DjangoLovers

قسمت اول: )ایجاد فولدر، نصب جنگو، ایجاد پروژه، انتقال به گیت هاپ(

1. ایجاد فولدری به نام DjangoLovers
2. باز کردن cmd داخل فولدر و ایجاد محیط مجازی: python -m venv env
3. داخل پوشه env و بعد از آن scripts شده و محیط مجازی را اکتیو میکنیم: activate
4. به پوشه اصلی برگشته و برای نصب جنگو اقدام میکنیم: python -m pip install Django
5. پروژه جدید را ایجاد میکنیم: django-admin startproject djangolovers .
6. وارد فولدر djangolovers می شویم و سرور را فعال میکنیم: manage.py runserver
7. حال به پورت <http://localhost:8000/> رفته پروژه را بالا می آوری.
8. در فولدر اصلی پروژه در cmd دستور git init را برای فایل گیت تایپ میکنیم.
9. برای اضافه کردن همه فایل ها به پروژه دستور git add . را تایپ میکنیم.
10. در cmd یک توضیح برای ارسال به گیت هاب مینویسیم: git commit -m 'start project'
11. در گیت هاپ یک ریپازیتوری جدید میسازیم به نام djangolovers
12. پروژه را به داخل ریپازیتوری میفرستیم :

git remote add origin https://github.com/SeyedAmirSamavatian/djangolovers.git

git branch -M main

git push -u origin main

قسمت دوم : (صفحه اول، برخی تنظیمات )

1. تنظیمات settings.py :

Import os

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

MEDIA\_URL = '/media/'

1. تنظیم urls.py در اپ اصلی:

if settings.DEBUG:

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

1. در تنظیمات settings.py مقدار DIRS در TEMPLATES را برابر 'templates' قرار میدهیم.
2. در فولدر اصلی پروژه یک فولدر با نام templates میسازیم و بعد هم یک پوشه به نام media در پوشه اصلی پروژه میسازیم و در داخل آن یک فولدر به نام imagesSite میسازیم و عکس پس زمینه صفحه اول سایت را در آن قرار میدهیم.
3. در پوشه templates یک فایل به نام index.html قرار میدهیم.
4. در فایل urls.py اپ اصلی آدرس زیر را اضافه میکنیم:

from views import welcome

urlpatterns = [

…..

path('/', welcome , name='welcome'),

]

1. در اپ اصلی فایل views.py را ایجاد کرده و فانکشن زیر را مینویسیم:

from django.shortcuts import render

def welcome(request):

return render(request,'index.html')

1. فایل فونت وزیر را دانلود میکنیم و در فولدر media و فولدر fonts میریزیم.
2. در فایل index.html کد های زیر را مینویسیم:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Django Lovers</title>

</head>

<style>

@font-face {

font-family: "vazir";

src: url("media/fonts/Vazir-Light.ttf");

}

.center{

display: flex;

align-items: center;

justify-content: center;

text-align: center;

flex-direction: column;

}

a{

width: 80px;

background: #7DC0F9;

text-decoration: none;

cursor: pointer;

border: solid 2px #ddd;

border-radius: 5px;

padding: 5px;

font-size: 25px;

margin: 35px;

font-family: 'vazir';

color: #fff;

box-shadow:2px 2px 4px rgba(0, 0, 0, 0.3);

}

p{

font-size:25px;

margin: 0 30px;

font-family: 'vazir';

}

div{

margin: 10px;

}

</style>

<body style="background-size: 100% 100%; background-image: url('media/imagesSite/djangoBack.jpg'); background-attachment: fixed; background-repeat: no-repeat;">

<div style="width: 100%; height: 100%; display: flex; align-items: center;justify-content: center;">

<div class="center" style="width: 50%;height: 100%;right:0%;bottom: 0px;position: absolute;">

<div class="center" style="width:60%;height: 100%;display: flex; flex-direction: column" >

<div>

<img width="150px" src="media/imagesSite/django-python-logo.png" alt="">

</div>

<div class="center" style="flex-direction: row;">

<p style="font-size:50px;">Django Lovers</p>

</div>

<div class="center" style="flex-direction: row;">

<p>...به ما بپیوندید </p>

<img width=80px src="media/imagesSite/cartoon.jpg" alt="">

</div>

<div class="center" style="flex-direction: row;">

<a href="" class="center">ثبت نام</a>

<a href="" class="center">ورود</a>

</div>

</div>

</div>

</div>

</body>

</html>

10\_ انتقال به github :

Git add .

Git commit -m "part 2"

git remote add origin https://github.com/SeyedAmirSamavatian/djangolovers.git

git push -u origin main

قسمت سوم (static و bootstap و css و base.html و ... ) :

1. فولدر static را در پوشه اصلی پروژه میسازیم
2. در setting.py کد زیر را مینویسیم:

STATIC\_URL = "static/" پیشفرض

STATICFILES\_DIRS = [BASE\_DIR, "satatic"] دستی وارد شود

1. مسیر فایل static را در فایل urls.py اپ اصلی اضافه میکنیم:

if settings.DEBUG:

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

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

1. در فولدر static فایل base.css و base.js را میسازیم.
2. در پوشه templates یک فایل به نام base.html میسازیم.
3. فولدر static را در فایل base.html وارد میکنیم: {% load static %}
4. لینک css های فایل base.html را از پوشه static به قسمت هدر این فایل اضافه می کنیم:

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

1. در فایل base.html در قسمت هدر Link و در پایین صفحه script مربوط به بوت استرپ 5.3.0 را اضافه میکنیم و فایل base.js را هم در آخر اضافه میکنیم:

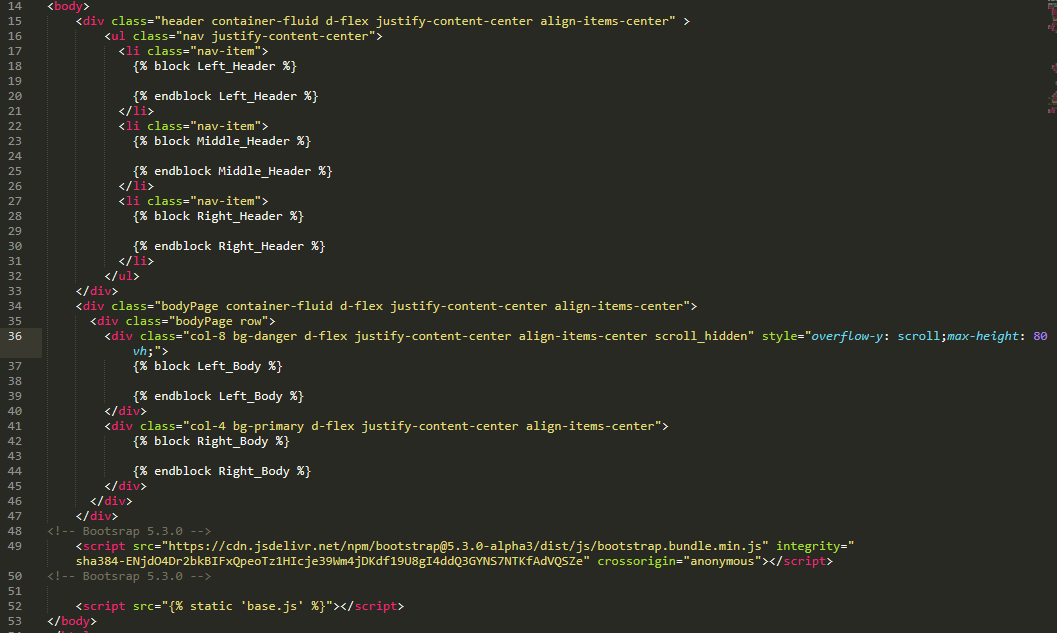
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-KK94CHFLLe+nY2dmCWGMq91rCGa5gtU4mk92HdvYe+M/SXH301p5ILy+dN9+nJOZ" crossorigin="anonymous">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap.bundle.min.js" integrity="sha384-ENjdO4Dr2bkBIFxQpeoTz1HIcje39Wm4jDKdf19U8gI4ddQ3GYNS7NTKfAdVQSZe" crossorigin="anonymous"></script>

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

1. در قسمت تگ title هم {% block title %} {% endblock title %} را مینویسیم.

10\_ داخل تگ body هم کد های زیر را مینویسیم:



11 – داخل فایل base.css هم کد های زیر را مینویسیم:

.header{

background: #CFC8C8;

height:50px;

width: 80% !important;

margin:30px 0;

border-radius: 5px;

}

.bodyPage{

width: 85% !important;

height: 80vh;

padding: 0 50px;

}

.bodyPage\_left,.bodyPage\_right{

height: 90%;

margin: 0 20px!important;

}

12\_ انتقال به github :

git add .

git commit -m "part 3"

git push -u origin main

قسمت چهارم (database و Admin و login و registry ) :

1. نصب mysql : pip install mysqlclient
2. ساخت یک دیتابیس به نام jangolovers در wampserver
3. تنظیم دیتابیس mysql در settings.py :

DATABASES = {

'default': {

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

'NAME': ' mydatabase',

'USER': 'root',

'PASSWORD': '',

'HOST': '127.0.0.1',

'PORT': '3306',

}

}

1. میگریشن و میگریت:

Manage.py makemigrations

Manage.py migrate

1. ایجاد سوپر یوزر: python manage.py createsuperuser
2. ساخت اَپ accounts : manage.py startapp accounts
3. اضافه کردن نام اَپ در فایل settings.py و اضافه کردن آدرس در urls.py اَپ اصلی.

settings.py:

INSTALLED\_APPS = [

…..

'accounts',

]

----------------------------------------------------------------------------

urls.py:

from django.urls import path , include

urlpatterns = [

…

path('accounts/', include( 'accounts.urls' )),

]

1. ساخت فایل urls.py در اَپ accounts و ایمپرت کردن path و ساخت آدرس registery برای صفحه ثبت نام و ایمپرت فانکشن رجیستری از فایل views.py:

from django.urls import path

from .views import registery

urlpatterns = [

path('registery/', registery , name='registery'),

]

1. ساخت فانکشن registery در صفحه views.py :

from django.shortcuts import render, redirect

from django.contrib.auth import authenticate, login, logout

from django.contrib import messages

from .forms import RegisteryUserForm

def registery(request):

if request.method == 'POST':

form = RegisteryUserForm (request.POST)

if form.is\_valid():

email = form.cleaned\_data['email']

user\_email\_count = User.objects.filter(email = email).count()

if user\_email\_count == 0:

form.save()

username = form.cleaned\_data['username']

password = form.cleaned\_data['password1']

user = authenticate(username = username , password = password)

login(request, user)

return redirect('welcome')

else:

messages.success(request, (' کاربر با این ایمیل قبلا ثبت نام کرده است '))

else:

form = RegisteryUserForm ()

return render(request, 'register.html' , {'form' : form})

10\_ در فایل models.py از اَپ accounts کد های زیر را اضافه میکنیم:

from django.db import models

from django.contrib.auth.models import User

class Profile(models.Model):

user = models.OneToOneField(User, on\_delete=models.CASCADE)

image = models.ImageField(upload\_to='profile\_pics', blank=True)

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

return self.user.email

11\_ ساخت فایل forms.py در اَپ accounts :

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from .models import Profile

class RegisteryUserForm(UserCreationForm):

email = forms.EmailField(label='ایمیل')

image = forms.ImageField(required =False)

class Meta:

model = User

fields = ( 'first\_name', 'last\_name', 'username', 'email', 'password1', 'password2', 'image' )

def save(self, commit=True):

user = super(RegistrationForm, self).save(commit=False)

user.email = self.cleaned\_data['email']

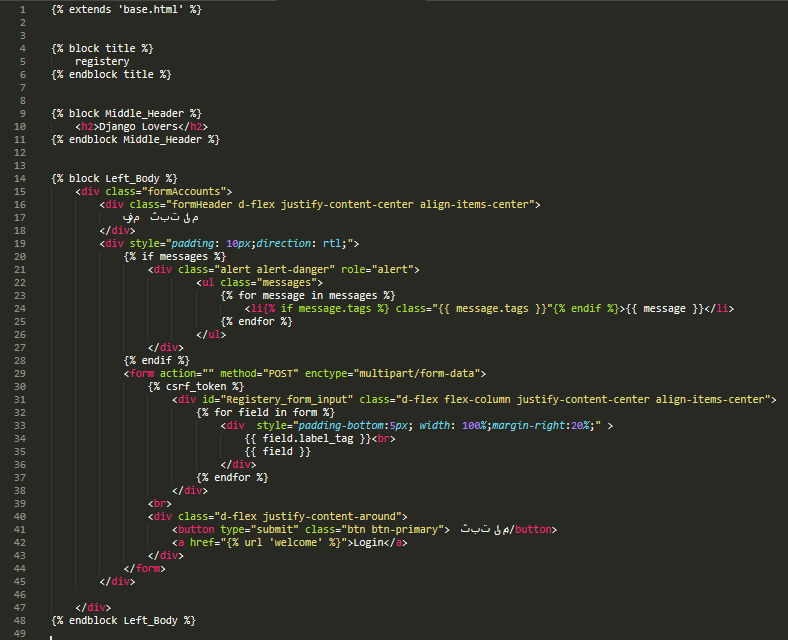
if commit:

user.save()

profile = Profile(user=user, image=self.cleaned\_data.get('image'))

profile.save();

return user

12 – ساخت فایل registery.html در templates :

13 – اضافه کردن کد های زیر به base.css :

.bodyPage\_left,.bodyPage\_right{

height: 80%;

margin: 0 20px!important;

}

.formAccounts{

width: 90%;

height: 100%;

background: #F5F5F5;

border-radius: 15px;

}

.formHeader{

width: 100%;

height: 30px;

background: #CFC8C8;

border-radius: 15px 15px 0 0;

}

14\_ بعد از انجام تنظیمات بالا باید برای کار با عکس ها اول پکیج Pillow را نصب کرد و بعد یکبار دیگر برای شناساندن مدل پروفایلی که در models.py نوشتیم عملیات میگریشن را انجام میدهیم:

python -m pip install Pillow

manage.py makemigrations

manage.py migrate

برای دیدن اطلاعات پروفایل در پنل ادمین در اَپ accounts در فایل admin.py کد زیر را وارد میکنیم:

from .models import Profile

admin.site.register(Profile)

برای نمایش بهتر فیلد های فرم در فایل base.js کد های زیر را مینویسیم:

const Registery\_form\_input = document.getElementById('Registery\_form\_input');

const inputList = Registery\_form\_input.getElementsByTagName('input');

if(Registery\_form\_input && inputList){

for (let i = 0; i < inputList.length; i++) {

inputList[i].classList.add('form-control');

inputList[i].style.width = '80%';

}

}

انتقال به github :

git add .

git commit -m "part 4"

git push -u origin main

قسمت پنجم( فرم لاگین ) :

1 – در فایل urls.py درون اَپ accounts آدرس login را اضافه میکنیم و فانکشن آن را از views.py ایمپرت میکنیم :

from django.urls import path

from .views import registery , login

urlpatterns = [

path('registery/', registery , name='registery'),

path('login/', login\_view, name='login'),

]

2 – اصلاح فانکشن RegisteryUserForm در فایل forms.py برای حذف username و جایگزین کردن ایمیل به جای آن:

class RegisteryUserForm(UserCreationForm):

email = forms.EmailField(label='ایمیل')

image = forms.ImageField(required =False)

class Meta:

model = User

fields = ('first\_name', 'last\_name','email', 'password1', 'password2', 'image') # username حذف شد

def save(self, commit=True):

user = super(RegisteryUserForm, self).save(commit=False)

user.email = self.cleaned\_data['email']

user.username = self.cleaned\_data['email']

if commit:

user.save()

profile = Profile(user=user, image=self.cleaned\_data.get('image'))

profile.save();

return user

همچنین فانکشن registery در فایل views.py راهم اصلاح کرده و مقدار username را مانند زیر قرار میدهیم:

username = form.cleaned\_data['email']

3 – فانکشن لاگین در فایل views.py را مینویسیم:

from .forms import RegisteryUserForm, LoginForm

def login\_view(request):

if request.method == 'POST':

form = LoginForm(request.POST)

if form.is\_valid():

email = form.cleaned\_data['email']

password = form.cleaned\_data['password']

user = authenticate(request, username = email, password = password)

if user is not None:

login(request, user)

return redirect('welcome')

else:

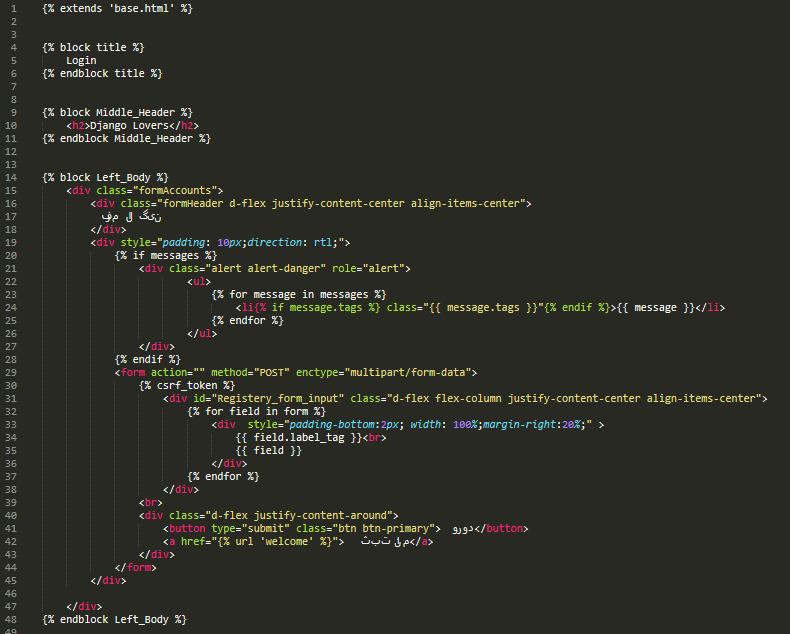
messages.success(request, 'ایمیل یا پسورد اشتباه است')

else:

form = LoginForm()

return render(request, "login.html", {'form':form})

4 – ساخت فایل login.html در templates :



انتقال به github :

git add .

git commit -m "part 5"

git push -u origin main

قسمت ششم ( Home ) :

1 – ساخت اَپ contents : python manage.py startapp contents و اضافه کردن اَپ به settings.py

2 – ساخت فایل urls.py در اپ contents و تنظیم آدرس url آن در اَپ اصلی پروژه:

Contents app:

from django.urls import path

from .views import home

urlpatterns = [

path( " " , home , name='home'),

]

Djangolovers app:

urlpatterns = [

……..

path('home/', include('contents.urls')),

]

3 – ساخت مدل مربوط به پست ها در فایل models.py :

from django.db import models

from django.contrib.auth.models import User

class Post(models.Model):

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

content = models.TextField()

likes = models.ManyToManyField(User, related\_name='liked\_posts')

author = models.ForeignKey(User, on\_delete=models.CASCADE)

date = models.DateTimeField(auto\_now\_add = True)

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

return self.title

4 – اضافه کردن کد زیر به فایل admin.py :

from .models import Post

admin.site.register(Post)

5 – میگریشن:

python manage.py makemigrations

python manage.py migrate

6 – فایل views.py :

from django.shortcuts import render

from .models import Post

def home(request):

user\_id = request.user.id

posts = Post.objects.filter(author\_id=user\_id)

return render(request, 'home.html', {'posts': posts})

7 – حذف اسکرول بار از طریق اضافه کردن کد css به فایل base.css :

.scroll\_hidden::-webkit-scrollbar { برای کروم و سافری و اُپرا

display: none;

}

.scroll\_hidden {

-ms-overflow-style: none; /\* IE and Edge \*/

scrollbar-width: none; /\* Firefox \*/

}

8 - اضافه کردن کد های سایت iconicons در بالای صفحه base.html :

<script type="module" src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"></script>

<script nomodule src="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"></script>

9 – ساخت فایل home.html در templates و تغییرات base.html :

Base.html



Home.html



انتقال به github :

git add .

git commit -m "part 6"

git push -u origin main

قسمت هفتم (دکوراتورها و لینک ها و ... ) :

1 – برای اینکه چک کنیم فانکشن home فقط زمانی اجرا شود که کاربر login کرده باشد ابتدا در بالای صفحه from django.contrib.auth.decorators import login\_required را وارد کرده و بعد بالای فانکشن home دکوراتور @login\_required را مینویسیم.

2 – در فایل index.html مقدار href لینک ورود و ثبت نام را به صورت زیر پر میکنیم:

<div class="center" style="flex-direction: row;">

<a href="{% url 'registery' %}" class="center">ثبت نام</a>

<a href="{% url 'login' %}" class="center">ورود</a>

</div>

3 – در خط اول فانکشن registery و login\_view در اَپ accounts شرط زیر را قرار میدهیم تا در صورت لاگین بودن کاربر به صفحه home ریدایرکت شود همچنین صفحه ریدایرکت کاربر بعد از لاگین یا ثبت نام را هم از welcome به home تغییر میدهیم:

if request.user.is\_authenticated:

return redirect('home')

4- در فایل registery.html در لینک ورود در پایین صفحه مقدار href را برابر {% url 'login' %} و در فایل login.html در قسمت لینک ثبت نام در پایین صفحه مقدار href را برابر {% url 'registery' %} قرار میدهیم.

5 – ساخت لینک logout : (بدون نیاز به views.py)

Urls.py:

from django.contrib.auth.views import LogoutView

urlpatterns = [

……

path('logout/', LogoutView.as\_view(next\_page='login'), name='logout'),

]

Home.html:

{% url 'logout' %} لینک آیکون خروج در تگ اِی

6 – تنظیمات پاک کردن پست:

Views.py:

@login\_required

def deletePost(request, id):

post\_del = Post.objects.get(id=id)

if post\_del.author\_id == request.user.id:

post\_del.delete()

posts = Post.objects.filter(author\_id=request.user.id)

return render(request, 'home.html', {'posts': posts})

urls.py:

from .views import home, deletePost

urlpatterns = [

path(' ', home , name='home'),

path('delete/<int:id>/', deletePost , name='delete\_Post'),

]

Home.html: تگ لینک مربوط به دیلیت

href="{% url 'delete\_Post' post.id %}"

قسمت هشتم ( like و ajax )

1 – ایجاد آدرس url برای عملیات like :

Urls.py

from .views import home, deletePost, likeDislike

urlpatterns = [

…….

path('like-post/<int:post\_id>/', likeDislike , name='like-post'),

]

2 – ساخت تابع ajax در فایل base.js :

function likeDislike(post\_id){

const url = '/home/like-post/' + post\_id + '/';

const csrf\_token = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

const xhr = new XMLHttpRequest();

post\_id = JSON.parse(post\_id)

xhr.open('POST', url);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.setRequestHeader('X-CSRFToken', csrf\_token);

xhr.onload = function() {

if (xhr.status === 200) {

let response = JSON.parse(xhr.responseText);

let element = document.getElementById('count'+post\_id)

let count = Number(element.innerHTML)

element.innerHTML = response.likes\_count;

let image\_Liked = document.getElementById('image\_Liked'+post\_id)

image\_Liked.innerHTML = response.liked\_pro;

}

}

xhr.send('post\_id=' + post\_id);

}

3 – اضافه کردن id به تگ های نمایش دهنده تعداد لایک ها و پروفایل لایک کننده ها در فایل home.html :

<span id="image\_Liked{{post.id}}" class="d-flex flex-row">

{% for like in post.likes.all|slice:":8" %}

<img src="{{ like.profile.image.url }}" alt="{{ like.profile.image }}" class="rounded-circle" style="width:30px;height: 30px;margin:0 -5px">

{% endfor %}

{% if post.likes.count > 8 %}

<h5 style="margin-left: 15px;">...</h5>

{% endif %}

</span>

<span class="d-flex flex-row">

<span onclick="likeDislike({{ post.id }})" class="like-button badge text-bg-danger d-flex flex-row" style="font-size:15px">

{% csrf\_token %}

<p id="count{{post.id}}">{{ post.likes.count }}</p>

<ion-icon style='color:#fff;font-size:30px' name="heart-outline"></ion-icon>

</span>

4 – در صفحه views.py کد های زیر را وارد میکنیم:

from django.http import JsonResponse

from django.http import HttpRequest

@login\_required

def likeDislike(request, post\_id):

if request.method == 'POST':

post = Post.objects.get(id=post\_id)

user = request.user

if user in post.likes.all():

post.likes.remove(user)

liked = False

else:

post.likes.add(user)

liked = True

likes\_count = post.likes.count()

posts = Post.objects.filter(author\_id=request.user.id)

liked\_pro = []

for index, like in enumerate(post.likes.all()):

profile\_Pic = f"<img src='{like.profile.image.url}' alt='{like.profile.image}' class='rounded-circle' style='width:30px;height: 30px;margin:0 -5px'>"

liked\_pro.append(profile\_Pic)

if index == 8:

break

return JsonResponse({'success': True, 'liked': liked, 'likes\_count': likes\_count, 'liked\_pro':liked\_pro})

return JsonResponse({'success': False})

قسمت نهم: (ارسال پست جدید)

1 – در فایل urls.py از اَپ contents آدرس زیر را اضافه کرده و فانکشن آن را ایمپرت میکینیم:

from .views import home, deletePost, likeDislike, send\_post

urlpatterns = [

……

path('new\_post', send\_post , name='new\_post'),

]

2 – فایل forms.py :

from django import forms

from .models import Post

class PostForm(forms.ModelForm):

class Meta:

model = Post

fields = ['title', 'content']

labels = {

'title': 'تیتر',

'content': 'متن',

}

3 – در فایل views.py :

from django.shortcuts import render , redirect

from .forms import PostForm

@login\_required

def send\_post(request):

if request.method == 'POST':

form = PostForm(request.POST)

if form.is\_valid():

post = form.save(commit=False)

post.author = request.user

post.save()

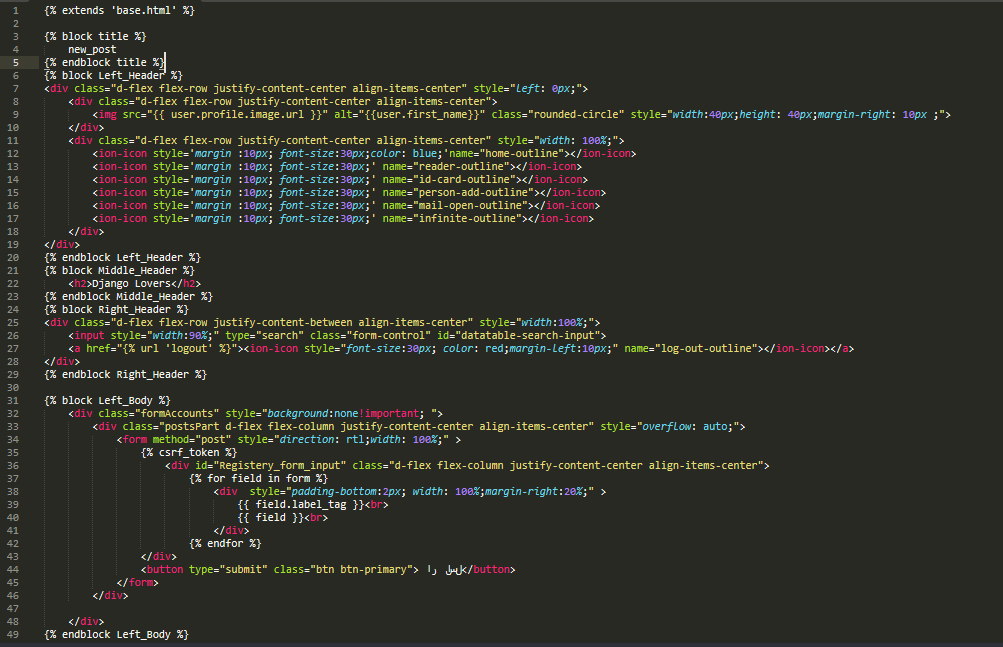
return redirect('home')

else:

form = PostForm()

return render(request, 'send\_post.html', {'form': form})

4 – فایل send\_post.html :



اصلاحات چند خط اول فایل base.js :

const Registery\_form\_input = document.getElementById('Registery\_form\_input');

const inputList = Registery\_form\_input.getElementsByTagName('input');

const textareaList = Registery\_form\_input.getElementsByTagName('textarea');

if(Registery\_form\_input && inputList){

for (let i = 0; i < inputList.length; i++) {

inputList[i].classList.add('form-control');

inputList[i].style.width = '80%';

}

for (let i = 0; i < textareaList.length; i++) {

textareaList[i].classList.add('form-control');

textareaList[i].style.width = '80%';

}

}

اضافه کردن فونت وزیر به فایل base.css برای کل سایت:

@font-face {

font-family: "vazir";

/\*src: url("media/fonts/Vazir-Bold.ttf");\*/

src: url("media/fonts/Vazir-Light.ttf");

}

\*{

font-family: 'vazir';

}

قسمت دهم : ( اضافه کردن قسمت کامنت گذاری )

Models.py:

class PostComments(models.Model):

content = models.TextField()

author = models.ForeignKey(User, on\_delete=models.CASCADE)

date = models.DateTimeField(auto\_now\_add=True)

post = models.ForeignKey(Post, on\_delete=models.CASCADE, related\_name='comments')

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

return f'{self.author.first\_name} commented on {self.post.title}'

urls.py:

from .views import home, deletePost, likeDislike, send\_post, add\_comment

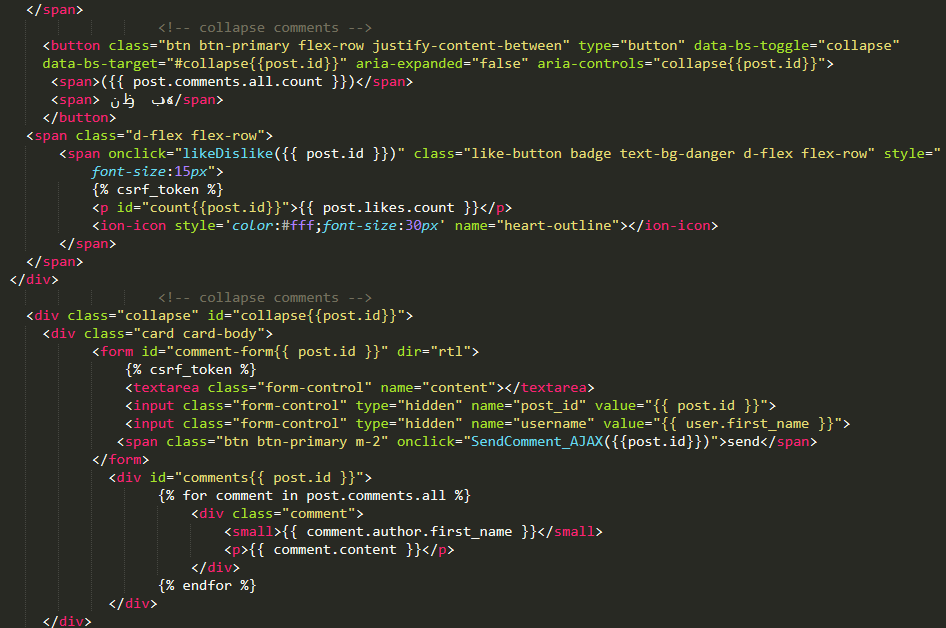
urlpatterns = [

…….

path('add\_comment/', add\_comment, name='add\_comment'),

]

Home.html:



base.js:

function SendComment\_AJAX(post\_id){

const form = document.querySelector('#comment-form'+post\_id);

const commentsSection = document.querySelector('#comments'+post\_id);

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

const formData = new FormData(form);

fetch('/home/add\_comment/', {

method: 'POST',

headers: {

'X-CSRFToken': csrftoken

},

body: formData

})

.then(response => response.json())

.then(data => {

if (data.success) {

commentsSection.innerHTML += `

<div class="comment">

<small>${formData.get('username')}</small>

<p>${formData.get('content')}</p>

</div>

`;

}

})

}

Views.py:

def add\_comment(request):

if request.method == 'POST':

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

post\_id = request.POST.get('post\_id')

author = request.user

post = Post.objects.get(id=post\_id)

comment = PostComments(content=content, author=author, post=post)

comment.save()

return JsonResponse({'success': True})

else:

return JsonResponse({'success': False})

قسمت یازدهم: ( حذف کامنت ها ):

1 - اصلاحات محل نمایش کامنت ها در home.html :

<div id="comments{{ post.id }}" >

{% for comment in post.comments.all %}

<div id="commentBox{{comment.id}}" class="comment d-flex flex-row justify-content-between align-items-center">

<div>

<small>{{ comment.author.first\_name }}</small>

<p>{{ comment.content }}</p>

</div>

<ion-icon onclick="DeleteComment({{comment.id}})" style='color:red; cursor: pointer;'; name="trash-outline"></ion-icon>

</div><hr>

{% endfor %}

</div>

2 – اضافه کردن jquery به base.html:

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

3 – اضافه کردن فانکشن ajax در base.js :

function DeleteComment(comment\_id){

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

$.ajax({

url: comment\_id + '/delete/',

type: 'POST',

beforeSend: function(xhr , settings){

xhr.setRequestHeader('X-CSRFToken', csrftoken);

},

data: {},

success: function(response) {

if (response.success) {

$(`#commentBox${comment\_id}` ).remove();

}

},

error: function() {

alert('An error occurred while deleting the comment.');

}

});

}

4 – اضافه کردن آدرس :

path('<int:comment\_id>/delete/', delete\_comment, name='delete\_comment'),

views.py:

from django.shortcuts import get\_object\_or\_404

# @csrf\_exempt

@login\_required

def delete\_comment(request, comment\_id):

print(comment\_id)

comment = get\_object\_or\_404(PostComments, id=comment\_id)

if request.user == comment.author:

comment.delete()

return JsonResponse({'success': True})

else:

return JsonResponse({'success': False})

انتقال به github :

git add .

git commit -m "part 11"

git push -u origin main

قسمت دوازدهم ( Edit پست ها به صورت کلاس بیس ):

1 - ایجاد آدرس جدید برای ویرایش:

from .views import home, deletePost, likeDislike, send\_post, add\_comment, delete\_comment, PostUpdateView

urlpatterns = [

…..

path('<int:pk>/edit/', PostUpdateView.as\_view(), name='post\_edit'),

]

2 - اصلاح لینک در home.html :

<li>

<a class="dropdown-item" href="{% url 'post\_edit' post.id %} ">

<ion-icon style='margin-right:15px;' name="create-outline"></ion-icon>Edit

</a>

</li>

3 – ایجاد کلاس برای آپدیت :

from django.views.generic.edit import UpdateView

from django.urls import reverse\_lazy

from django.contrib.auth.mixins import LoginRequiredMixin

class PostUpdateView(UpdateView):

model = Post

fields = ['title', 'content']

template\_name = "post\_edit.html"

success\_url = reverse\_lazy('home')

def get\_queryset(self): برای اینکه هر کاربر فقط بتواند پست خودش را ادیت کند

queryset = super().get\_queryset()

return queryset.filter(author=self.request.user)

4 – ایجاد فایل post\_edit.html :

یک کپی از فایل send\_post.html میگیریم و فقط لینک بالای صفحه را که در هدر صفحه ارسال پست را رنگی نشان میدهد به home تغییر میدهیم

انتقال به github :

git add .

git commit -m "part 12"

git push -u origin main

قسمت سیزدهم ( پروفایل دیگران )

1 – ساخت اَپ کاربران دیگر : manage.py startapp otheruser

2 – اضافه کردن اَپ otheruser به لیست اَپ های پروژه در settings.py

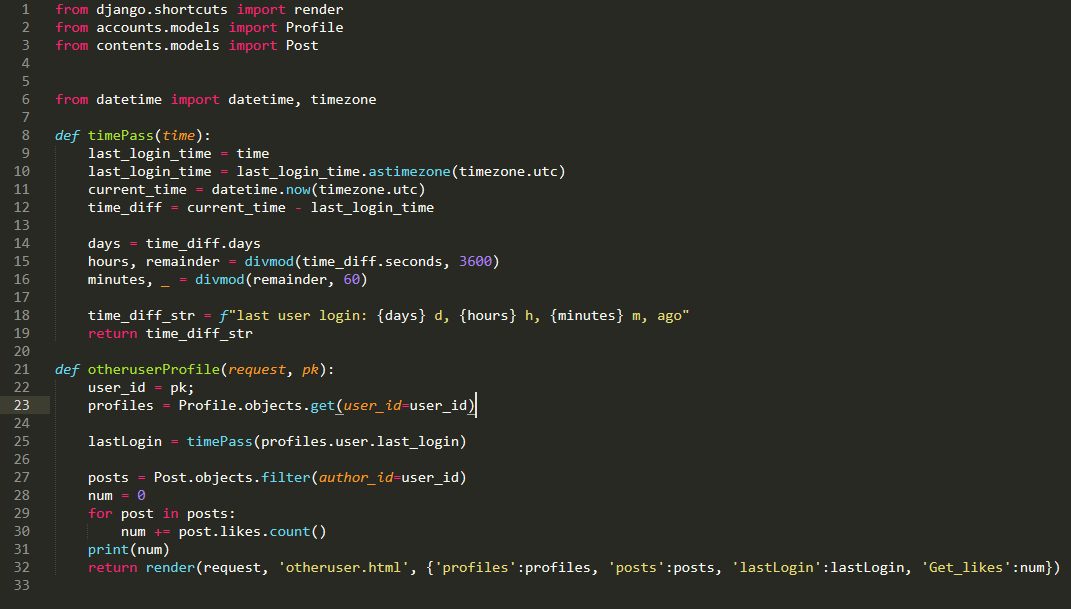
3 – اضافه کردن آدرس اَپ به urls.py اَپ اصلی:

path('otheruser/', include('otheruser.urls')),

4 – ساخت فایل urls.py و آدرس در اَپ otheruser :

path('<int:pk>', otheruserProfile , name='otheruser'),

5 – ساخت فانکشن otheruserProfile در فایل views.py :



6 – یک کپی از home.html به نام otheruser.html ذخیره میکنیم و پس از تغییرات در هدر و تایتل صفحه و پاک کردن بخش ادیت و دیلیت در دراپ باکس، در انتهای صفحه کد های block Right\_Body را مینویسیم :

{% block Right\_Body %}

<div class="card" dir="rtl">

<img src="{{ profiles.image.url }}" class="card-img-top" alt="...">

<div class="card-body">

<h5 class="card-title">{{ profiles.user.first\_name }}</h5>

<div class="d-flex flex-row justify-content-around align-items-center m-3">

<button type="button" class="btn btn-info" style="color: #fff;">Fallow</button>

<button type="button" class="btn btn-info" style="color: #fff;">Message</button>

</div>

<p class="card-text">{{ profiles.user.email }}</p>

<p class="card-text"> تعداد پست ها : {{ posts.count }}</p>

<p class="card-text">} تعداد لایک ها : {{ Get\_likes }}</p>

<p class="card-text"></p>

<p class="card-text"><small class="text-body-secondary">{{lastLogin}}</small></p>

</div>

</div>

{% endblock Right\_Body %}

قسمت چهاردهم ( following ) :

1 - models.py :

from django.contrib.auth.models import User

class Follower(models.Model):

follower = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='following')

following = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='followers')

class Meta:

unique\_together = ('follower', 'following')

2 – migrations:

Manage.py makemigrations

Manage.py migrate

3 – اضافه کردن کدهای جدید به فانکشن otheruserProfile برای ارسال اطلاعات فالوور ها به صفحه otheruser.html :

from .models import User, Follower

def otheruserProfile(request, pk):

user\_id = pk;

profiles = Profile.objects.get(user\_id=user\_id)

lastLogin = timePass(profiles.user.last\_login)

posts = Post.objects.filter(author\_id=user\_id)

num = 0

for post in posts:

num += post.likes.count()

users = User.objects.all()

followers = Follower.objects.filter(following\_id = user\_id)

following = []

for follow in followers:

print(follow.follower.id)

following.append(follow.follower.id)

return render(request, 'otheruser.html', {'profiles':profiles, 'posts':posts, 'lastLogin':lastLogin, 'Get\_likes':num, 'users':users, 'following': following})

4 - urls.py:

path('<int:user\_to\_follow>/follow/', follow\_user , name='follow'),

5 – در صفحه base.js فانکشن follow را برای ارسال داده به صورت ای جکس مینویسیم:

function follow(){

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

$.ajax({

url: window.location.href+"follow/",

type: 'POST',

beforeSend: function(xhr , settings){

xhr.setRequestHeader('X-CSRFToken', csrftoken);

},

data: {},

success: function(response) {

if (response.success) {

$('#follow\_btn').html(response.follow);

}

},

error: function() {

alert('An error occurred while following.');

}

});

}

6 – ایجاد فانکشن follow\_user در views.py برای انجام حذف یا اضافه کردن در جدول follower:

from django.http import JsonResponse

def follow\_user(request, user\_to\_follow):

user\_to\_follow = get\_object\_or\_404(User, id=user\_to\_follow)

if request.user != user\_to\_follow:

follower, created = Follower.objects.get\_or\_create(follower=request.user, following=user\_to\_follow)

follow = 'Unfollow'

if not created:

follower.delete()

follow = 'Follow'

return JsonResponse({'success': True, 'follow': follow })

انتقال به github :

git add .

git commit -m "part 13 , part 14"

git push -u origin main

قسمت پانزدهم ( پیام خصوصی ) :

1 – ساخت اَپ chat : manage.py startapp chat

اضافه کردن اَپ به settings.py و اضافه کردن آدرس url به اَپ اصلی :

path(chat/', include(chat.urls')),

2 – ساخت فایل urls.py و اضافه کردن آدرس :

from django.urls import path

from .views import new\_message

urlpatterns = [

path('<int:pk>/', new\_message, name=new\_message),

]

3 – models.py:

from django.db import models

from django.contrib.auth.models import User

class Message(models.Model):

sender = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='sent\_messages')

recipient = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='received\_messages')

timestamp = models.DateTimeField(auto\_now\_add=True)

message = models.TextField()

is\_read = models.BooleanField(default=False)

4 – manage.py makemigrations و manage.py migrate

5 – اضافه کردن مُدال از bootstrap به صفحه otheuser.html و تغییر دکمه message



7 - در فایل base.js فانکشن ارسال اطلاعات به شکل api را مینویسیم:

function sendMessage(user\_id){

const form = document.querySelector('#send-message-form');

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

const formData = new FormData(form);

fetch( `/chat/${user\_id}/` , {

method: 'POST',

headers: {

'X-CSRFToken': csrftoken

},

body: formData

})

.then(response => response.json())

.then(data => {

if (data.success) {

$('#message\_textarea').val('');

alert(data.alert)

let close = $('#close\_modal').click()

}

})

}

8 – ساخت view :

from django.shortcuts import render, redirect

from django.contrib.auth.decorators import login\_required

from .models import Message

from django.contrib.auth.models import User

from django.http import JsonResponse

@login\_required

def new\_message(request,pk):

if request.method == 'POST':

sender\_message = int(request.POST.get('sender\_id'))

recipient\_message = int(request.POST.get('recipient\_id'))

if request.user.id == sender\_message:

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

sender\_id = User.objects.get(id=sender\_message)

recipient\_id = User.objects.get(id=recipient\_message)

sendMessage = Message(sender=sender\_id, recipient = recipient\_id, message= message)

sendMessage.save()

return JsonResponse({'success': True,'alert':' پیام شما با موفقیت ارسال شد'})

else:

return JsonResponse({'success': False})

انتقال به github :

git add .

git commit -m "part 15"

git push -u origin main

قسمت شانزدهم ( چت ) :

1 – یک کپی از مدل message برای چت میسازیم و نام آن را به chatBox تغییر میدهیم و مقدار related\_name آن را نیز تغییر میدهیم.

class ChatBox(models.Model):

sender = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='sent\_chat')

recipient = models.ForeignKey(User, on\_delete=models.CASCADE, related\_name='received\_chat')

timestamp = models.DateTimeField(auto\_now\_add=True)

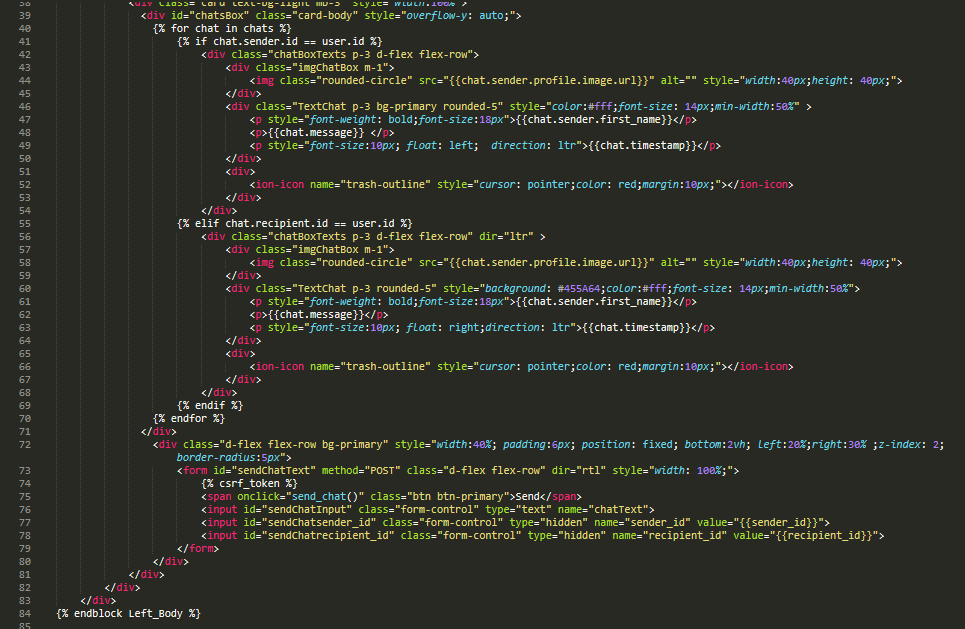
message = models.TextField()

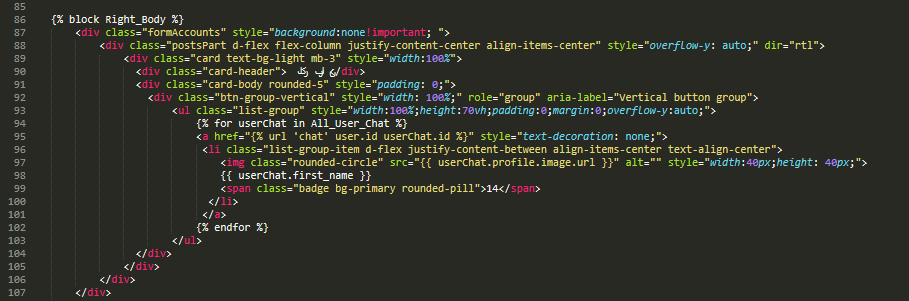
is\_read = models.BooleanField(default=False)

2 – manage.py makemigrations و manage.py migrate

3 – بعد از ساخته شدن جداول در دیتابیس چند نمونه دستی وارد جدول میکنیم تا ظاهر جدول را هنگام ساختن صفحه ببینیم

4 – صفحه chat.html را از طریق کپی از صفحه home و تغییرات مربوطه می سازیم:





در انتهای صفحه chat.html کد جاوا اسکریپت زیر را وارید میکنیم:

<script>

function scrollDone(){

var chatBox = $('#divScroll');

chatBox.scrollTop(chatBox.prop("scrollHeight"));

}

window.onload = scrollDone();

var chatBoxHTML = $('#divScroll').html();

function updateChat(){

const recipient\_id = document.getElementById('sendChatrecipient\_id').value;

const sender\_id = document.getElementById('sendChatsender\_id').value;

const form = document.querySelector('#sendChatText');

const url = `/chat/update/` + recipient\_id + '/';

const csrf\_token = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

const xhr = new XMLHttpRequest();

xhr.open('POST', url);

xhr.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');

xhr.setRequestHeader('X-CSRFToken', csrf\_token);

xhr.onload = function() {

if (xhr.status === 200) {

let chatBox = $('#chatsBox');

chatBox.html('');

let response = JSON.parse(xhr.responseText);

let dataChat = response.dataChat;

let element='';

for(let i=0; dataChat.length>i; i++){

if(dataChat[i]['sender\_id'] == sender\_id){

element = `

<div class="chatBoxTexts p-3 d-flex flex-row">

<div class="imgChatBox m-1">

<img class="rounded-circle" src="${response.img\_sender}" alt="" style="width:40px;height: 40px;">

</div>

<div class="TextChat p-3 bg-primary rounded-5" style="color:#fff;font-size: 14px;min-width:50%" >

<p style="font-weight: bold;font-size:18px">${response.first\_name\_sender}</p>

<p>${dataChat[i]['message']} </p>

<p style="font-size:10px; float: left; direction: ltr">${dataChat[i]['timestamp']}</p>

</div>

<div>

<ion-icon name="trash-outline" style="cursor: pointer;color: red;margin:10px;"></ion-icon>

</div>

</div>

`;

chatBox.append(element)

}else if(dataChat[i]['sender\_id'] == recipient\_id){

element = `

<div class="chatBoxTexts p-3 d-flex flex-row" dir="ltr" >

<div class="imgChatBox m-1">

<img class="rounded-circle" src="${response.img\_recipient}" alt="" style="width:40px;height: 40px;">

</div>

<div class="TextChat p-3 rounded-5" style="background: #455A64;color:#fff;font-size: 14px;min-width:50%">

<p style="font-weight: bold;font-size:18px">${response.first\_name\_recipient}</p>

<p>${dataChat[i]['message']}</p>

<p style="font-size:10px; float: right;direction: ltr">${dataChat[i]['timestamp']}</p>

</div>

<div>

<ion-icon name="trash-outline" style="cursor: pointer;color: red;margin:10px;"></ion-icon>

</div>

</div>

`;

chatBox.append(element)

}

}

if(chatBoxHTML != $('#divScroll').html()){

chatBoxHTML = document.getElementById('divScroll').innerHTML;

scrollDone();

}

}

}

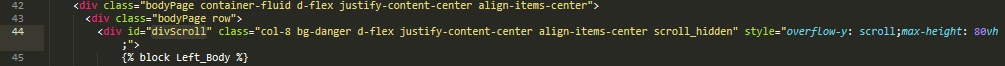
xhr.send('recipient\_id=' + recipient\_id);

}

setInterval(updateChat,5000)

</script>

برای انجام اسکریپت آخر صفحه باید یک آی دی divscroll به تگ مورد نظر در base.html اضافه کنیم:



urls.py:

from .views import new\_message, chat , chat\_send, chat\_update

urlpatterns = [

…….

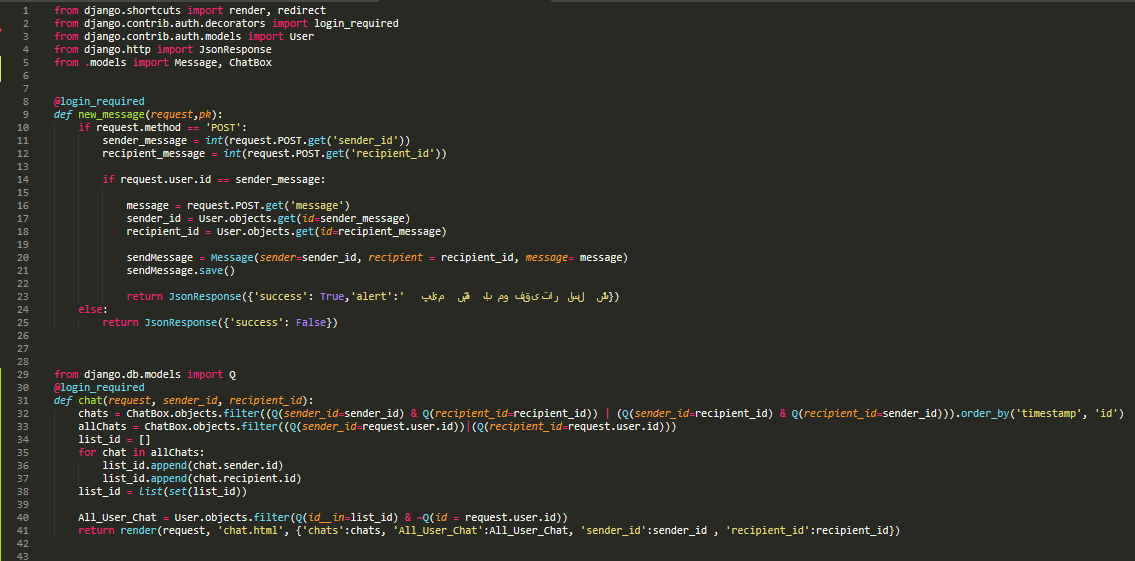
path('<int:sender\_id>/<int:recipient\_id>/', chat , name='chat'),

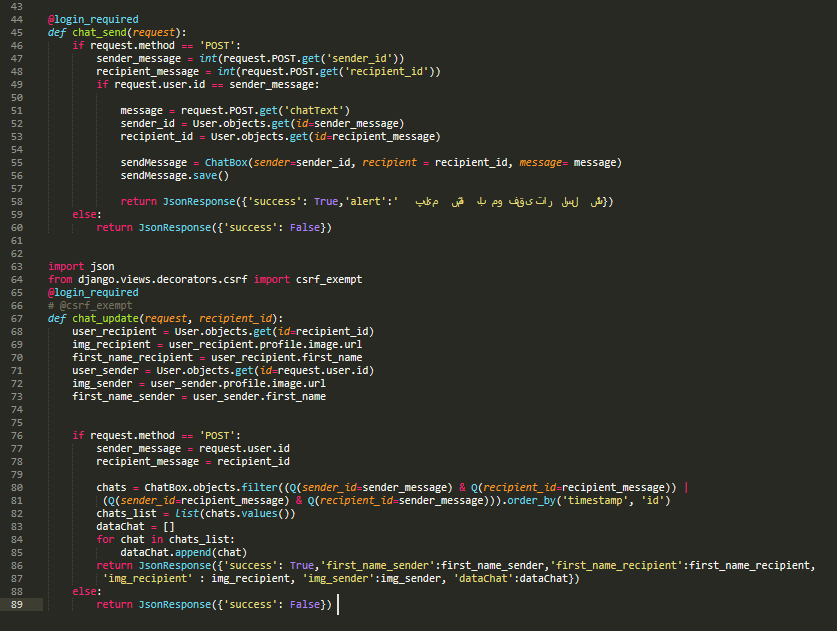
path('send/', chat\_send , name='chat\_send'),

path('update/<int:recipient\_id>/', chat\_update , name='chat\_update'),

]

Views.py:





در فایل base.js فانکشن مربوط به ارسال چت را مینویسیم:

function send\_chat(){

const form = document.querySelector('#sendChatText');

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

const formData = new FormData(form);

fetch(`/chat/send/`, {

method: 'POST',

headers: {

'X-CSRFToken': csrftoken

},

body: formData

})

.then(response => response.json())

.then(data => {

if (data.success) {

$('#sendChatInput').val(' ');

}

انتقال به github :

git add .

git commit -m "part 16"

git push -u origin main

قسمت هفدهم (حذف چت ):

urls.py:

path('delete/<int:id>/', delete\_chat , name='delete\_chat'),

views.py:  
def delete\_chat(request, id):

chat\_del = ChatBox.objects.get(id=id)

if ((chat\_del.sender\_id == request.user.id) | (chat\_del.recipient\_id == request.user.id)):

chat\_del.delete()

return JsonResponse({'success': True})

else:

return JsonResponse({'success': False})

در chat.py به آیکن حذف چت اتریبیوت زیر را اضافه میکنیم:

onclick='deleteChat({{chat.id}})'

و در فانکشن updateChat هم به همین تگ مورد زیر را اضافه میکنیم:

onclick='deleteChat(${dataChat[i]['id']})'

در صفحه base.js فانکشن مربوط به پاک کردن چت را مینویسیم:

function deleteChat(id){

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

$.ajax({

url: '/chat/delete/'+ id+'/' ,

type: 'POST',

beforeSend: function(xhr , settings){

xhr.setRequestHeader('X-CSRFToken', csrftoken);

},

data: {},

success: function(response) {

if (response.success) {

alert("پیام مورد نظر حذف شد");

}

},

error: function() {

alert('حذف پیام با مشکل روبرو شد! دوباره امتحان کنید');

}

});

}

انتقال به github :

git add .

git commit -m "part 17"

git push -u origin main

قسمت هجدهم ( ویرایش پروفایل و تغییر پسورد ) :

الف) ویرایش پروفایل:

urls.py:

from .views import registery , login\_view , edit\_profile

urlpatterns = [

……

path('profile/', edit\_profile, name='edit\_profile'),

]

ایجاد صفحه profile.html با کپی از home و تغییر آن:



برای نمایش عکس پروفایل قبل از تغییر آن کد جاوا اسکریپت زیر را به آخر صفحه profile.html قبل از بسته شدن تگ endblack Left\_Body مینویسیم:

<script>

function previewImage(event) {

var preview = document.getElementById('preview-image');

var file = event.target.files[0];

var reader = new FileReader();

reader.onload = function() {

preview.src = reader.result;

}

if (file) {

reader.readAsDataURL(file);

}

}

</script>

forms.py:

class UpdateUserForm(forms.ModelForm):

first\_name = forms.CharField(required=True, max\_length=30)

last\_name = forms.CharField(required=True, max\_length=30)

image = forms.ImageField(required=False, widget=forms.ClearableFileInput(attrs={'onchange': 'previewImage(event)'}))

class Meta:

model = User

fields = ('first\_name', 'last\_name', 'image')

def save(self, commit=True):

user = super(UpdateUserForm, self).save(commit=False)

if commit:

user.save()

try:

profile = user.profile

except Profile.DoesNotExist:

profile = Profile(user = user)

if self.cleaned\_data['image'] is not None:

profile.image = self.cleaned\_data['image']

profile.save()

return user

views.py:

from .forms import RegisteryUserForm, LoginForm, UpdateUserForm

from django.contrib.auth.decorators import login\_required

@login\_required

def edit\_profile(request):

if request.method == 'POST':

user\_form = UpdateUserForm(request.POST, request.FILES, instance=request.user)

if user\_form.is\_valid():

user = user\_form.save()

return render(request, 'profile.html', {'user\_form': user\_form})

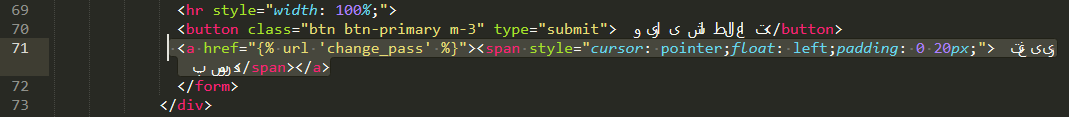
else:

user\_form = UpdateUserForm(instance=request.user)

return render(request, 'profile.html', {'user\_form': user\_form})

ب ) تغییر پسورد:

اضافه کردن لینک صفحه تغییر پسورد به profile.html :



urls.py

from .views import registery , login\_view , edit\_profile, passwordChangeView

urlpatterns = [

….

path('profile/change\_pass', passwordChangeView.as\_view(), name='change\_pass'),

]

forms.py

from django.contrib.auth.forms import PasswordChangeForm

class ChangePassword(PasswordChangeForm):

old\_password = forms.CharField(label='پسورد فعلی' , widget = forms.PasswordInput(attrs={'class':'form-control'}))

new\_password1 = forms.CharField(label='پسورد جدید' , widget = forms.PasswordInput(attrs={'class':'form-control'}))

new\_password2 = forms.CharField(label='تکرار پسورد جدید' , widget = forms.PasswordInput(attrs={'class':'form-control'}))

views.py

from django.contrib.auth.views import PasswordChangeView

from django.urls import reverse\_lazy

class passwordChangeView(PasswordChangeView):

form\_class = ChangePassword

template\_name = 'change\_password.html'

success\_url = reverse\_lazy('edit\_profile')

ایجاد صفحه change\_password.html از کپی profile.html و تغییر آن:



انتقال به github :

git add .

git commit -m "part 18"

git push -u origin main

قسمت نوزدهم ( نمایش پیام های خصویصی و حذف آن ها ):

در urls.py سه آدرس میسازیم یکی برای نمایش همه کسانی که پیام خصوصی دادند و دیگری برای نمایش پیام های هر فرد و یکی هم برای پاک کردن پیام ها:

urls.py:

from .views import new\_message, chat , chat\_send, chat\_update, delete\_chat, privateMessage, userPrivate\_message, Del\_Private\_message

urlpatterns = [

….

path('privateMessage/', privateMessage , name='privateMessage'),

path('privateMessage/<int:id>/', userPrivate\_message , name='userPrivate\_message'),

path('privateMessage/delete/<int:id>/<int:sender\_id>', Del\_Private\_message , name='Del\_Private\_message'),

]

در views.py برای هر یک از آدرس ها یک فانکشن میسازیم:

Views.py:

from contents.models import Post

@login\_required

def privateMessage(request):

user\_id = request.user.id

user = User.objects.get(id = user\_id)

senders = Message.objects.filter(recipient\_id = request.user.id)

senders\_user =User.objects.filter(

Q(sent\_messages\_\_recipient=request.user)

).distinct();

posts = Post.objects.filter(author\_id=user\_id)

return render(request, 'private\_message.html', {'senders\_user' : senders\_user, 'posts':posts})

@login\_required

def userPrivate\_message(request, id):

user\_id = request.user.id

sender\_id = id

messages = Message.objects.filter(Q(recipient\_id = user\_id) & Q(sender\_id = id)).order\_by('-id')

senders\_user =User.objects.filter(

Q(sent\_messages\_\_recipient=request.user)

).distinct();

return render(request, 'private\_message\_user.html', {'senders\_user' : senders\_user, 'messages':messages})

from django.http import HttpResponse

from django.shortcuts import get\_object\_or\_404

@login\_required

def Del\_Private\_message(request, id, sender\_id):

message = get\_object\_or\_404(Message, id=id)

if message.sender\_id == request.user.id or message.recipient\_id == request.user.id:

message.delete()

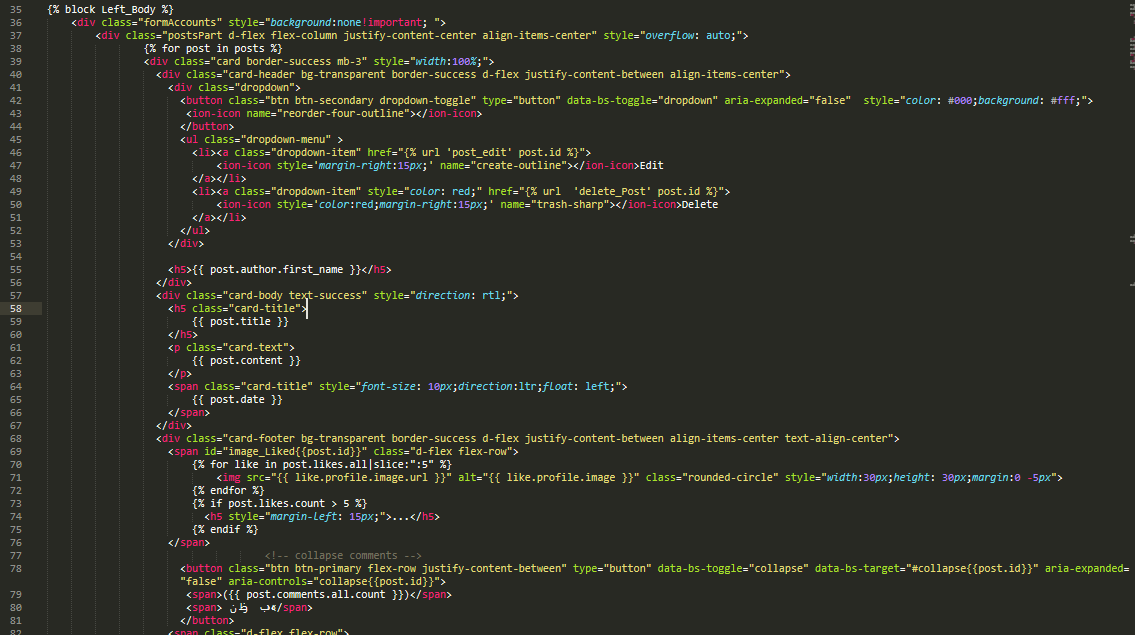
return redirect('userPrivate\_message', id=sender\_id)

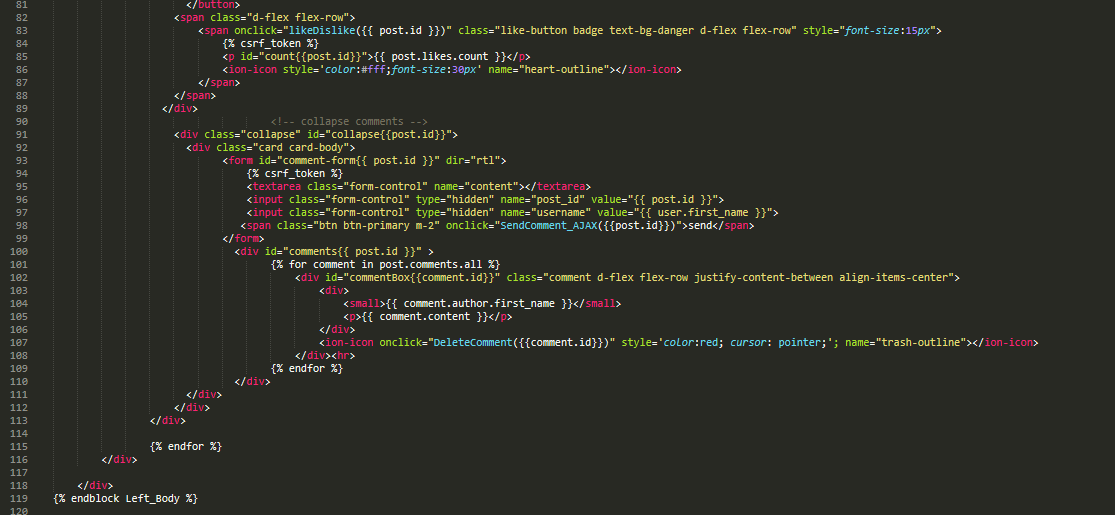
else:

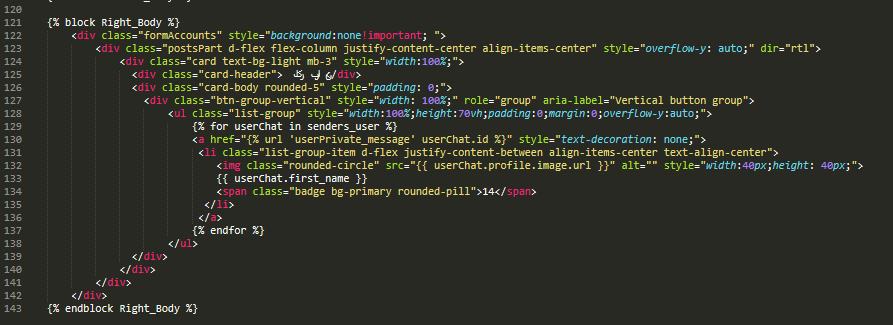
return HttpResponse("You don't have permission to delete this message")

حال دو صفحه برای فانکشن ها میسازیم یکی برای نمایش تمام پیغام دهندگان به همراه پست های کاربر ( private\_message.html ) و دیگری برای نمایش پیغام دهندگان به همراه پیغام های یک پیغام دهنده خاص( private\_message\_user.html ):

Private\_message.html:

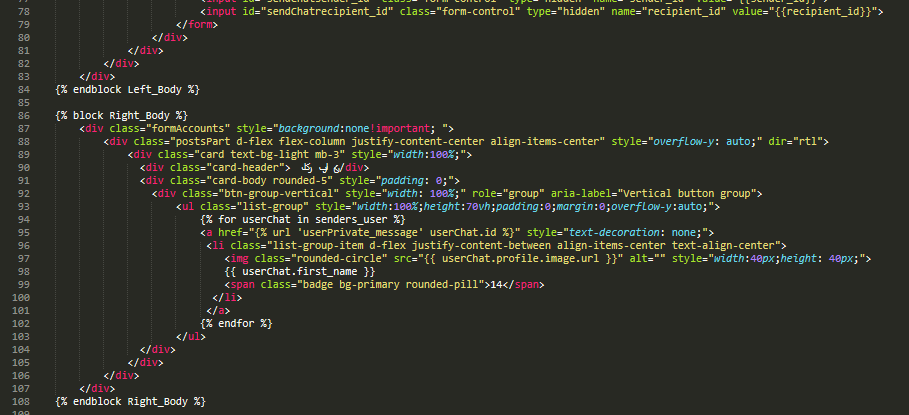






Private\_message\_user.html:





انتقال به github :

git add .

git commit -m "part 19"

git push -u origin main

قسمت بیستم ( نمایش همه پست های ارسالی و سرچ ) :

الف) نمایش همه پست های ارسالی:

contents urls.py:

urlpatterns = [

…..

path('all\_post/', all\_post, name='all\_post'),

]

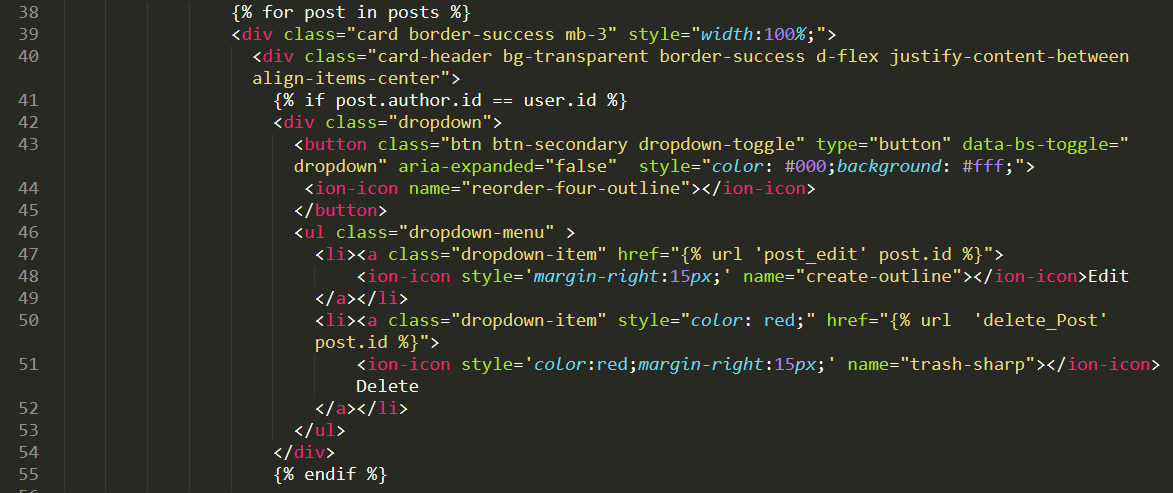
views.py:

def all\_post(request):

posts = Post.objects.all().order\_by('-id')

return render(request, 'all\_post.html', {'posts': posts})

برای ساخت صفحه all\_post.html یک کپی از home گرفته و اصلاحات لازم را انجام میدهیم:



ب) سرچ :

در اَپ contents آدرس زیر را میسازیم و فانکشن search را در بالای صفحه ایمپرت میکنیم:

urlpatterns = [

……

path('search/', search, name='search'),

]

در view از اَپ contents کد زیر را مینویسیم:

from django.db.models import Q

def search(request):

data = [ ]

if request.headers.get('x-requested-with') == 'XMLHttpRequest': برای چک کردن وجود درخواست ایجکس

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

q = User.objects.filter(Q(first\_name\_\_icontains = search) | Q(last\_name\_\_icontains = search))

if len(q) > 0 and len(search) > 0:

for pos in q:

item = {

'pk': pos.pk,

'first\_name': pos.first\_name,

'last\_name': pos.last\_name,

'img' : pos.profile.image.url,

}

data.append(item)

return JsonResponse({'success': True, 'data' : data})

else:

return JsonResponse({'success': False})

در فولدر static یک فایل به نام search.js برای نوشتن کدهای ایجکس سرچ میسازیم:

search.js:

const searching = document.getElementById('datatable-search-input');

const sendSearchData = (search)=>{

const csrftoken = document.querySelector('input[name="csrfmiddlewaretoken"]').value;

$.ajax({

type:'POST',

beforeSend: function(xhr , settings){

xhr.setRequestHeader('X-CSRFToken', csrftoken);

},

url:`${window.location.origin + '/home/' + 'search/'}`,

data:{

'search': search,

},

success:(res)=>{

data = res.data;

if(Array.isArray(data) && data.length > 0){

let element = document.getElementById('search\_res');

element.style.display = 'flex';

element.innerHTML=' ';

data.forEach(user =>{

document.getElementById('search\_res').innerHTML += `

<div class='list-group-item'>

<a href='${window.location.origin+'/otheruser/' + user.pk }'>

<img src="${user.img}" alt="${user.img}" class="rounded-circle" style="width:30px;height: 30px;margin:5px">

<span>${user.first\_name}</span>

<span>${user.last\_name}</span>

</a>

</div><hr>

`

})

}

if(data.length==0){

let element = document.getElementById('search\_res');

element.style.display = 'none';

}

},

error:(err)=>{

let element = document.getElementById('search\_res');

element.style.display = 'none';

}

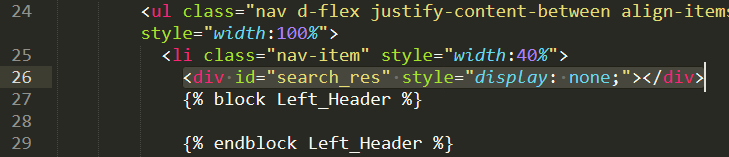
})

}

searching.addEventListener('keyup', e=>{

sendSearchData(e.target.value)

})

در صفحه base.html تگ دیو زیر را اضافه میکنیم:

در bace.css تنظیمات زیر را وارد میکنیم:

#search\_res{

min-width:25%;

position: absolute;

right:10%;

top: 75px;

border-radius: 5px;

background: #ddd;

z-index: 10;

padding: 20px;

display: flex;

flex-flow: column nowrap;

}

در آخر در فایل urls.py همه اَپ هایی که میخواهیم سرچ روی آن فعال باشد search را ایمپرت میکنیم و آدرس مورد نظر را میسازیم مانند اَپ چت و ... :

from contents.views import search

urlpatterns = [

…..

path('search/', search, name='search'),

]

انتقال به github :

git add .

git commit -m "part 20"

git push -u origin main