



National Institute of Technology Andhra Pradesh

Department of Computer Science & Engineering

WEB TECHNOLOGIES

LAB MANUAL

COURSE CODE :CS311

CLASS: BTech.III I Sem(CSE)

Name of Student : Priyanka K Gupta

Roll No :411963

Faculty Incharge:

Mr Y.Gireesh

PROJECT:EDUCATIONAL BLOG EDUBLOG

Framework: DJANGO

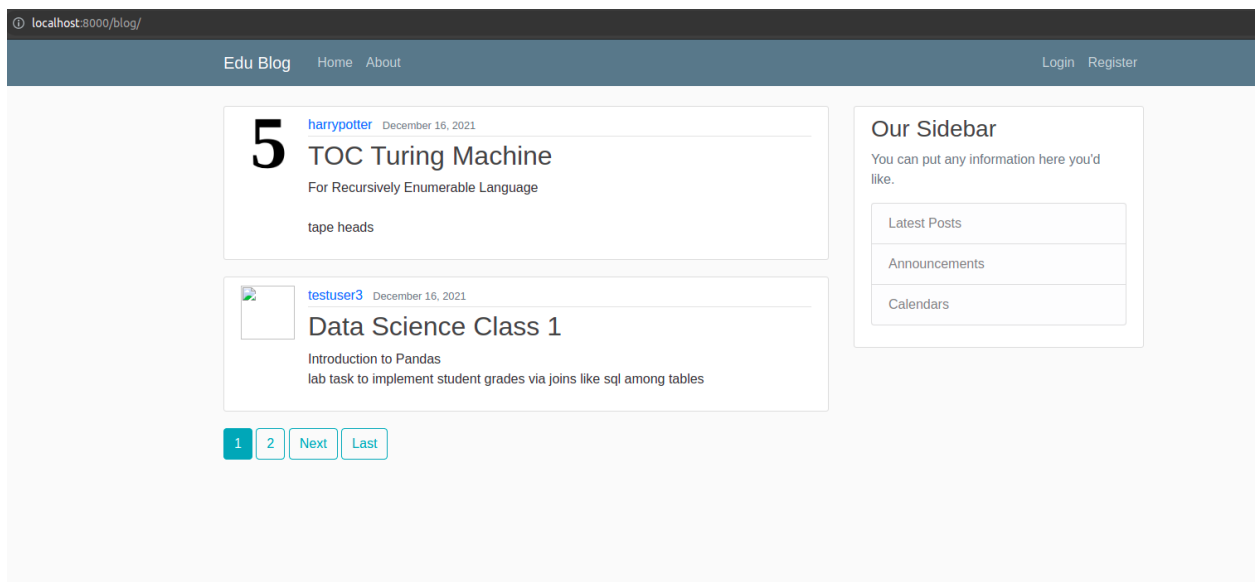
Frontend- HTML,CSS,JavaScript

Backend DB: sqlite3

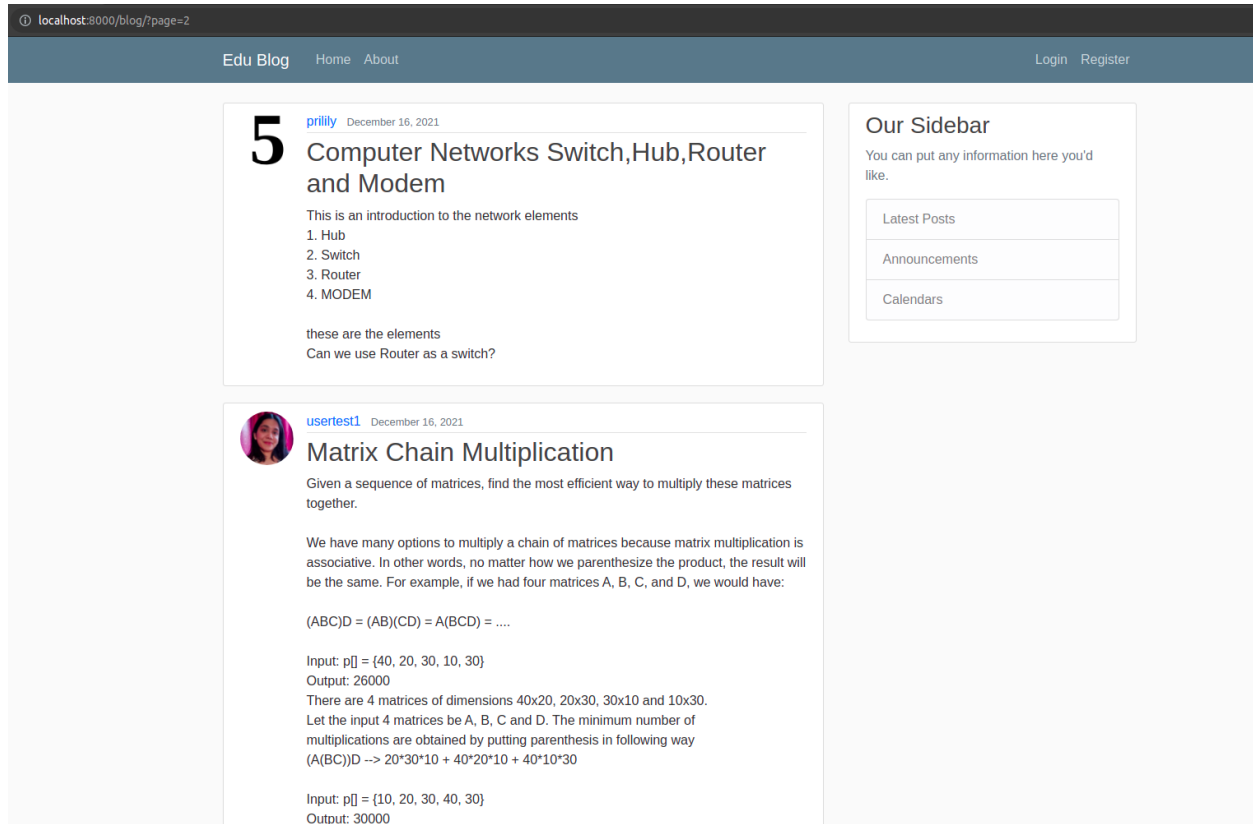
FRONTEND

1. The home page

Where all posts created by users appear, with the latest post on the top.



On the bottom is navigation to go to next/previous page



NEW USER REGISTRATION

To create post new users can be created who have the functionalities of making a post

Updating the post created by same user

Deleting the post created by same user

Updating profile information

localhost:8000/register/

Edu Blog Home About Login Register

Join Today

Username*

hagrid

Required: 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

hagrid@gmail.com

Password*

- Your password can't be too similar to your other personal information.
- Your password must contain at least 8 characters.
- Your password can't be a commonly used password.
- Your password can't be entirely numeric.

Password confirmation*

Enter the same password as before, for verification.

Sign Up

Already Have An Account? [Sign In](#)

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

Login after successful registration:

localhost:8000/login/

Edu Blog Home About Login Register

Your Account has been Created ! YOu can now log-in!

Log In

Username*

Password*

Login

New User? [Sign In](#)

Our Sidebar

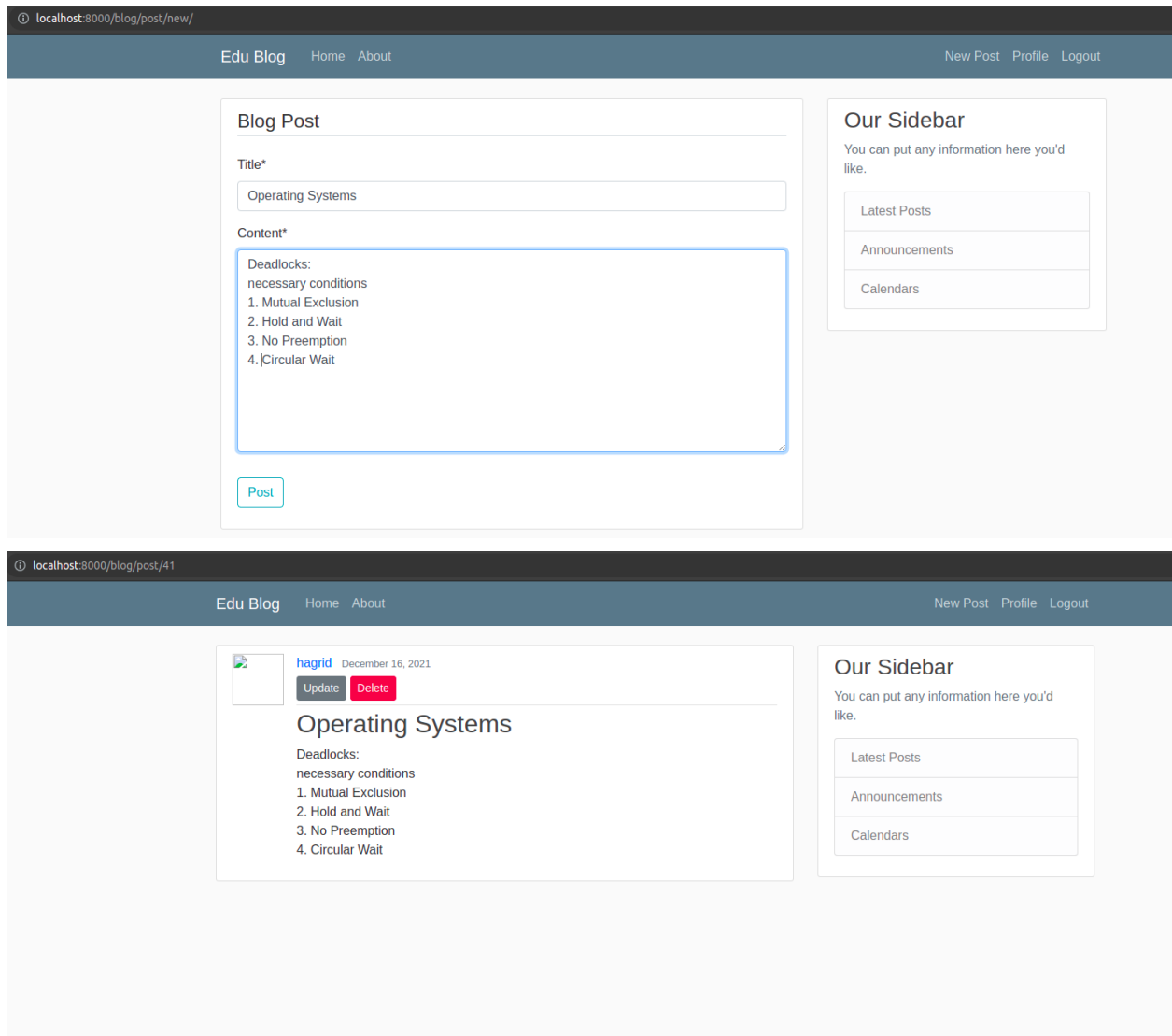
You can put any information here you'd like.

Latest Posts

Announcements

Calendars

To create a new post by NewPost button on navbar



option of update/delete only for author of post not viewers

localhost:8000/blog/post/41/update

Edu BlogHomeAboutNew PostProfileLogout

Blog Post

Title*

Operating Systems

Content*

Deadlocks:
Deadlock is a situation which involves the interaction of more than one resources and processes with each other

necessary conditions
1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

Post

Our Sidebar

You can put any information here you'd like.

Latest Posts


Announcements

Calendars

UPDATING POST

localhost:8000/blog/

Edu BlogHomeAboutNew PostProfileLogout



hagrid

December 16, 2021

Operating Systems

Deadlocks:
Deadlock is a situation which involves the interaction of more than one resources and processes with each other

necessary conditions

1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

5

harrypotter

December 16, 2021

TOC Turing Machine

For Recursively Enumerable Language

tape heads

1

2

3

Next

Last

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

LOGGING OUT

localhost:8000/logout/

Edu BlogHomeAboutLoginRegister

You have been Logged out

[Login Again](#) [Sign In](#)

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

New login via prilily-user

localhost:8000/blog/

Edu BlogHomeAboutNew PostProfileLogout

5

prilly

December 16, 2021


Ways of Dealing with Deadlock

1. Deadlock ignorance. It is the most popular method and it acts as if no deadlock and the user will restart.

2. Deadlock prevention. It means that we design such a system where there is no chance of having a deadlock.

3. Deadlock avoidance. ...

4. Detection and recovery.



hagrid

December 16, 2021

Operating Systems

Deadlocks:

Deadlock is a situation which involves the interaction of more than one resources and processes with each other

necessary conditions

1. Mutual Exclusion

2. Hold and Wait

3. No Preemption

4. Circular Wait

1

2

3

Next

Last

Our Sidebar

You can put any information here you'd like.

Latest Posts


Announcements

Calendars

Delete Post

localhost:8000/blog/post/38

Edu BlogHomeAboutNew PostProfileLogout



prillyDecember 16, 2021

UpdateDelete

Computer Networks Switch,Hub,Router and Modem

This is an introduction to the network elements

1. Hub
2. Switch
3. Router
4. MODEM

these are the elements

Can we use Router as a switch?

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

localhost:8000/blog/post/38/delete

Edu BlogHomeAboutNew PostProfileLogout

Delete Post

Are you sure you want to delete the post "Computer Networks Switch,Hub,Router and Modem"

Yes,DeleteCancel

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

localhost:8000/blog/?page=2

Edu BlogHomeAboutNew PostProfileLogout

5


harrypotter

December 16, 2021

TOC Turing Machine

For Recursively Enumerable Language

tape heads



testuser3December 16, 2021

Data Science Class 1

Introduction to Pandas

lab task to implement student grades via joins like sql among tables

First

Previous

1

2

3

Next

Last

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

PROFILE UPDATE

9

411963

localhost:8000/profile/

Edu Blog Home About New Post Profile Logout

5
prilily
priyankamessage@gmail.com

Profile Info

Username*
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Image*

Currently: [profile_pics/s1_aMc4aHR.png](#)

Change:

No file chosen

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

localhost:8000/profile/

Edu Blog Home About New Post Profile Logout

5
prilily
priyankamessage@gmail.com

Profile Info

Username*
Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.

Email*

Image*

Currently: [profile_pics/s1_aMc4aHR.png](#)

Change:

Untitled Workspace.png

Our Sidebar

You can put any information here you'd like.

Latest Posts

Announcements

Calendars

BACKEND:
2 apps
Blog and User
USER MODELS

```
models.py users X home.html post_form.html views.py blog models.py blog U
users > models.py > Profile > save
1 from django.db import models
2
3 # Create your models here.
4 from django.contrib.auth.models import User
5 from PIL import Image
6
7 #for profile to add picture etc, existing django model extend
8
9 class Profile(models.Model):
10     user=models.OneToOneField(User,on_delete=models.CASCADE)
11     image=models.ImageField(default='default.jpg', upload_to='profile_pics')
12
13
14     def __str__(self):
15         return f'{self.user.username} Profile'
16
17     def save(self, *args, **kwargs):
18         super().save(*args, **kwargs)
19
20         img=Image.open(self.image.path)
21
22         if(img.height > 300 or img.width >300):
23             output_size =(300,300)
24             img.thumbnail(output_size)
25             img.save(self.image.path)
```

POST MODEL(in blog app)

```

blog > models.py > ...
1  from django.db import models
2  from django.utils import timezone
3  # Create your models here.
4  from django.contrib.auth.models import User
5  from django.urls import reverse
6
7
8
9  class Post(models.Model):
10     title=models.CharField(max_length=100)
11     content=models.TextField()
12     date_posted=models.DateTimeField(default= timezone.now) #add is f
13     # do default which takes our timezone
14     # a post only have 1 author but author many post
15     author= models.ForeignKey(User, on_delete= models.CASCADE)
16     # re run migrations to reflect changes
17
18     #special methods with __
19     def __str__(self):
20         return self.title
21
22     def get_absolute_url(self):
23         return reverse('post-detail',kwargs={'pk':self.pk})
24
25

```

USER REGISTRATION FORM

```

from django import forms
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import Profile

class UserRegisterForm(UserCreationForm):
    email=forms.EmailField()
    #can give required=true for compulsory
    class Meta:
        model= User
        fields = ['username','email','password1','password2']

#form to update user model

class UserUpdateForm(forms.ModelForm):
    email=forms.EmailField()

    class Meta:
        model=User
        fields=['username','email']

class ProfileUpdateForm(forms.ModelForm):
    class Meta:
        model=Profile
        fields=['image']

#now put it on profile views

```

```

users > templates > users > <> register.html > ...
1  {% extends "blog/base.html" %}
2  {% load crispy_forms_tags %}
3  {% block content %}
4      <div class="content-section">
5          <form method="POST">
6              {% csrf_token %}
7              <fieldset class="form-group">
8                  <legend class="border-bottom mb-4">Join Today</legend>
9                  {{ form|crispy }}
10             </fieldset>
11             <div class="form-group">
12                 <button class="btn btn-outline-info" type="submit">Si
13             </div>
14         </form>
15         <div class="border-top pt-3">
16             <small class="text-muted">
17                 Already Have An Account? <a class="ml-2" href="{% url
18             </small>
19         </div>
20     </div>
21 {% endblock content %}

```

LOGIN/LOGOUT PAGES

```

j  register.html  urls.py django_project  signals.py  apps.py  login.html x  logout.html
users > templates > users > <> login.html > div.content-section > div.border-top.pt-3 > small.text-muted > a.ml-2
1  {% extends "blog/base.html" %}
2  {% load crispy_forms_tags %}
3  {% block content %}
4      <div class="content-section">
5          <form method="POST">
6              {% csrf_token %}
7              <fieldset class="form-group">
8                  <legend class="border-bottom mb-4">Log In</legend>
9                  {{ form|crispy }}
10             </fieldset>
11             <div class="form-group">
12                 <button class="btn btn-outline-info" type="submit">Login</button>
13             </div>
14         </form>
15         <div class="border-top pt-3">
16             <small class="text-muted">
17                 New User? <a class="ml-2" href="{% url 'register' %}">Sign In</a>
18             </small>
19         </div>
20     </div>
21 {% endblock content %}

```

```
users > templates > users > logout.html > h2
1 {% extends "blog/base.html" %}
2 {% block content %}
3     <h2>You have been Logged out</h2>
4     <div class="border-top pt-3">
5         <small class="text-muted">
6             Login Again <a class="ml-2" href="{% url 'login' %}">Sign In</a>
7         </small>
8     </div>
9 {% endblock content %}
```

Blog APP views and urls

```
views.py blog X  models.py blog  urls.py blog  # style.css  post_confirm_delete.html  pos
blog > views.py > home
1  from django.shortcuts import render
2  from django.contrib.auth.mixins import LoginRequiredMixin , UserPassesTestMixin
3  from django.http import HttpResponse
4  from .models import Post
5  from django.views.generic import ListView,DetailView,CreateView,UpdateView, DeleteView
6
7  # Create your views here.
8
9  '''
10 to be used as http response
11 def home(request):
12     return HttpResponse('<h1> Blog Home </h1>')
13
14 def about(request):
15     return HttpResponse('<h1> About page </h1>')'''
16
17 # as render
18 # so the dict we are passing from our render will be accesible to the page that can be
19 # via the key (i.e here context)
20 def home(request):
21     context={
22         'posts':Post.objects.all()
23     }
24     return render(request, 'blog/home.html',context)
25
26 class PostListView(ListView):
27     model=Post
28     template_name= 'blog/home.html'
29     context_object_name= 'posts' # why? in home template context is post i.e why
30     # <app>/<model>_<view_type>.html  template naming convention for class view
31     ordering=['-date_posted']
32     paginate_by=2
33
34 class PostDetailView(DetailView):
35     model=Post
36
37 class PostCreateView(LoginRequiredMixin,CreateView):
38     model=Post
39     fields=['title','content']
40
41     def form_valid(self,form):
42         form.instance.author=self.request.user
43         return super().form_valid(form)
44
45 class PostUpdateView(UserPassesTestMixin,LoginRequiredMixin,UpdateView):
46     model=Post
47     fields=['title','content']
48
```






```

49     def test_func(self):
50         post=self.get_object()
51         if(self.request.user == post.author):
52             return True
53         return False
54
55     def form_valid(self,form):
56         form.instance.author=self.request.user
57         return super().form_valid(form)
58
59     def about(request):
60         return render(request, 'blog/about.html')
61
62     class PostDeleteView(UserPassesTestMixin,LoginRequiredMixin,DeleteView):
63         #user login + author
64         model=Post
65         success_url='/blog'
66         def test_func(self):
67             post=self.get_object()
68             if(self.request.user == post.author):
69                 return True
70             return False
71
72
73

```

Urls.py (blog)

```

blog >  urls.py >  urlpatterns
1  from django.urls import path
2  from . import views
3  from .views import PostListView, PostDetailView, PostCreateView, PostUpdateView, PostDeleteView
4
5
6
7  urlpatterns=[
8      path('',PostListView.as_view(),name='blog-home'),
9      path('post/<int:pk>',PostDetailView.as_view(),name='post-detail'),
10     path('post/new/',PostCreateView.as_view(),name='post-create'),
11     path('post/<int:pk>/update',PostUpdateView.as_view(),name='post-update'),
12      path('post/<int:pk>/delete',PostDeleteView.as_view(),name='post-delete'),
13     path('about/',views.about,name='blog-about'), # this about is in the url after home
14 ]
15
16 #blog/post_list.html  template with the name
17 # <app>/<model>_<view_type>.html blog/post_list.html
18 #so, change it in views.py
19
20
21 # route to specific f+post
22 '''blog/
23
24 url with variable like post/1 orpost/2
25 so id of post part of the route,
26 post/<int:pk>
27
28 pk is post we want to view and say what kind of variable it is
29
30 now a template for post details
31 with convention as <app>/<model>_<viewtype> i.e blog/post_detail.html
32 '''

```

VIEWS APP Forms.py

```
{ } post.json  admin.py  views.py users  profile.html  forms.py x  base.html  p
users > forms.py > ProfileUpdateForm > Meta > fields
1  '''
2  here we will see to add email, name etc and all those
3  fields
4  a new form that inherits from usercreationform
5  so new file is this
6
7  meta class
8  this gives us nested namespace and keep config in one place
9  and we say in user model
10
11 after this go in views and add the form we just created
12
13 '''
14
15 from django import forms
16 from django.contrib.auth.models import User
17 from django.contrib.auth.forms import UserCreationForm
18 from .models import Profile
19
20 class UserRegisterForm(UserCreationForm):
21     email=forms.EmailField()
22     #can give required=true for compulsory
23     class Meta:
24         model= User
25         fields = ['username','email','password1','password2']
26
27 #form to update user model
28
29 class UserUpdateForm(forms.ModelForm):
30     email=forms.EmailField()
31
32     class Meta:
33         model=User
34         fields=['username','email']
35
36
37 class ProfileUpdateForm(forms.ModelForm):
38     class Meta:
39         model=Profile
40         fields=['image']
41
42 #now put it on profile views
```

Views.py

```
post.json  admin.py  views.py users X  profile.html  forms.py  base.html  profile.png

users > views.py > register
#decorator for login- add functionality to function

9
10 '''
11
12 # Create your views here.
13 def register(request):
14     if request.method== 'POST':
15         form=UserRegisterForm(request.POST)
16         if form.is_valid():
17             form.save()
18             username=form.cleaned_data.get('username')
19             messages.success(request,f'Your Account has been Created ! Y0u can now log-in!')
20             return redirect('login')
21     else:
22         form=UserRegisterForm()
23         context={
24             'form':form
25         }
26     return render(request,'users/register.html',{'form':form})
27
28 @login_required
29 def profile(request):
30     if request.method == 'POST':
31         u_form = UserUpdateForm(request.POST, instance=request.user)
32         p_form = ProfileUpdateForm(request.POST,
33                                   request.FILES,
34                                   instance=request.user.profile)
35         if u_form.is_valid() and p_form.is_valid():
36             u_form.save()
37             p_form.save()
38             messages.success(request, f'Your account has been updated!')
39             return redirect('profile')
40
41     else:
42         u_form = UserUpdateForm(instance=request.user)
43         p_form = ProfileUpdateForm(instance=request.user.profile)
44
45     context = {
46         'u_form': u_form,
47         'p_form': p_form
48     }
49
50     return render(request, 'users/profile.html', context)
51
```