Old code:

<!--{% load static %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <title>My-Ecommerce</title>  
 <link rel="stylesheet"  
 href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"  
 integrity="sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdIwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=="  
 crossorigin="anonymous" referrerpolicy="no-referrer"/>  
 <style>  
 \* {  
 margin: 0;  
 padding: 0;  
 box-sizing: border-box;  
 font-family: 'Arial', sans-serif;  
 }  
  
 body {  
 background-color: #eaeded;  
 }  
  
 .navbar {  
 height: 60px;  
 background-color: #131921;  
 color: white;  
 display: flex;  
 align-items: center;  
 justify-content: space-evenly;  
 }  
  
 .nav-logo {  
 height: 50px;  
 width: 100px;  
 }  
  
 .logo {  
 background-image: url('{% static "images/logo.png" %}');  
 background-size: cover;  
 height: 50px;  
 width: 100%;  
 }  
  
 .border {  
 border: 1px solid transparent;  
 padding: 5px;  
 }  
  
 .border:hover {  
 border: 1px solid white;  
 }  
  
 .nav-search {  
 display: flex;  
 flex-grow: 1;  
 margin: 0 10px;  
 max-width: 600px;  
 height: 40px;  
 }  
  
 .search-select {  
 background-color: #f3f3f3;  
 width: 60px;  
 border: none;  
 border-top-left-radius: 4px;  
 border-bottom-left-radius: 4px;  
 }  
  
 .search-input {  
 flex-grow: 1;  
 border: none;  
 padding: 0 10px;  
 font-size: 1rem;  
 }  
  
 .search-icon {  
 background-color: #febd69;  
 width: 45px;  
 display: flex;  
 justify-content: center;  
 align-items: center;  
 border-top-right-radius: 4px;  
 border-bottom-right-radius: 4px;  
 cursor: pointer;  
 }  
  
 .panel {  
 height: 40px;  
 background-color: #232f3e;  
 color: white;  
 display: flex;  
 align-items: center;  
 justify-content: space-evenly;  
 }  
  
 .panel-options p {  
 display: inline;  
 margin: 0 10px;  
 }  
  
 .hero-section {  
 background-image: url('{% static "images/hero\_image.jpg" %}');  
 height: 300px;  
 background-size: cover;  
 display: flex;  
 justify-content: center;  
 align-items: flex-end;  
 }  
  
 .hero-msg {  
 background-color: white;  
 color: black;  
 padding: 10px 20px;  
 margin-bottom: 25px;  
 font-size: 0.9rem;  
 width: 90%;  
 text-align: center;  
 }  
  
 .hero-msg a {  
 color: #007185;  
 text-decoration: none;  
 }  
  
 .shop-section {  
 display: flex;  
 flex-wrap: wrap;  
 justify-content: space-evenly;  
 background-color: #e2e7e6;  
 padding: 20px 0;  
 }  
  
 .box {  
 height: 400px;  
 width: 23%;  
 background-color: white;  
 padding: 20px 0 15px;  
 margin: 15px 0;  
 }  
  
 .box-img {  
 height: 300px;  
 background-size: cover;  
 margin: 1rem 0;  
 }  
  
 footer {  
 margin-top: 30px;  
 }  
  
 .foot-panel1 {  
 background-color: #37475a;  
 color: white;  
 height: 50px;  
 display: flex;  
 align-items: center;  
 justify-content: center;  
 font-size: 1rem;  
 cursor: pointer;  
 }  
  
 .foot-panel2 {  
 background-color: #232f3e;  
 color: white;  
 display: flex;  
 justify-content: space-evenly;  
 padding: 20px 0;  
 }  
  
 .foot-panel2 ul {  
 list-style-type: none;  
 }  
  
 .foot-panel2 ul p {  
 font-weight: bold;  
 }  
  
 .foot-panel2 ul a {  
 display: block;  
 margin-top: 10px;  
 font-size: 0.9rem;  
 color: #ddd;  
 text-decoration: none;  
 }  
  
 .foot-panel3 {  
 background-color: #232f3e;  
 color: white;  
 display: flex;  
 justify-content: center;  
 padding: 20px 0;  
 }  
  
 .foot-panel3 .logo {  
 background-image: url('{% static "images/logo.png" %}');  
 background-size: cover;  
 height: 50px;  
 width: 100px;  
 }  
  
 .foot-panel4 {  
 background-color: #131a22;  
 color: white;  
 padding: 20px;  
 text-align: center;  
 font-size: 0.85rem;  
 }  
  
 .foot-panel4 .pages a {  
 margin: 0 10px;  
 color: #ddd;  
 text-decoration: none;  
 }  
  
 .foot-panel4 .copyright {  
 margin-top: 10px;  
 }  
 </style>  
</head>  
<body>  
<header>  
 <div class="navbar">  
 <div class="nav-logo border">  
 <div class="logo"></div>  
 </div>  
 <div class="nav-address border">  
 <p class="add-first">Deliver to</p>  
 <div class="add-icon">  
 <i class="fa-solid fa-location-dot"></i>  
 <p class="add-second">India</p>  
 </div>  
 </div>  
  
 <div class="nav-search">  
 <select class="search-select" name="search-select" id="search-select">  
 <option>All</option>  
 </select>  
 <input class="search-input" placeholder="Search My-Ecommerce">  
 <div class="search-icon">  
 <i class="fa-solid fa-magnifying-glass"></i>  
 </div>  
 </div>  
  
 <div class="nav-signin border">  
 <p><span>Hello, sign in</span></p>  
 <p><span>Accounts & Lists</span></p>  
 </div>  
 <div class="nav-return border">  
 <p><span>Returns</span></p>  
 <p><span>& Orders</span></p>  
 </div>  
 <div class="nav-cart border">  
 <i class="fa-solid fa-cart-shopping"></i>  
 <p><span>Cart</span></p>  
 </div>  
 </div>  
  
 <div class="panel">  
 <div class="panel-all">  
 <i class="fa-solid fa-bars"></i>  
 All  
 </div>  
 <div class="panel-options">  
 <p>Today's Deals</p>  
 <p>Customer Service</p>  
 <p>Registry</p>  
 <p>Gift Cards</p>  
 <p>Sell</p>  
 <p>My-Ecommerce Basics</p>  
 </div>  
 <div class="panel-deals">  
 <p>Shop in Electronics</p>  
 </div>  
 </div>  
</header>  
  
<div class="hero-section">  
 <div class="hero-msg">  
 <p>  
 You are on my-ecommerce.com, you can also shop on My-Ecommerce India for a better experience and millions of  
 products.  
 <a href="https://www.amazon.in/">Click here to go to My-Ecommerce.in</a>  
 </p>  
 </div>  
</div>  
  
<div class="shop-section">  
 <div class="box">  
 <div class="box-content">  
 <h2>Clothing</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/product.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Health & Personal care</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box2\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Furniture</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box3\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Electronics</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box4\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Beauty Picks</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box5\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Pet Care</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box6\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>New Arrival in Toys</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box7\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
  
 <div class="box">  
 <div class="box-content">  
 <h2>Fashion Trends</h2>  
 <div class="box-img" style="background-image: url('{% static 'images/box8\_image.jpg' %}')"></div>  
 <p>See more</p>  
 </div>  
 </div>  
</div>-->

### Django Project Structure

# project\_name/

# ├── manage.py

# ├── project\_name/

# │ ├── \_\_init\_\_.py

# │ ├── settings.py

# │ ├── urls.py

# │ └── wsgi.py

# └── community\_app/

# ├── \_\_init\_\_.py

# ├── admin.py

# ├── apps.py

# ├── models.py

# ├── urls.py

# ├── views.py

# ├── forms.py

# └── templates/

# └── community\_app/

# └── index.html (and other templates)

### 1. Models - community\_app/models.py

from django.db import models

from django.contrib.auth.models import User

from django.utils import timezone

from django.urls import reverse

class Profile(models.Model):

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

bio = models.TextField(max\_length=500, blank=True)

location = models.CharField(max\_length=100, blank=True)

profile\_picture = models.ImageField(upload\_to='profile\_pics', default='default.jpg')

joined\_date = models.DateTimeField(default=timezone.now)

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

return f'{self.user.username} Profile'

class Post(models.Model):

title = models.CharField(max\_length=200)

content = models.TextField()

date\_posted = models.DateTimeField(default=timezone.now)

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

image = models.ImageField(upload\_to='post\_images', blank=True, null=True)

likes = models.ManyToManyField(User, related\_name='post\_likes', blank=True)

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

return self.title

def get\_absolute\_url(self):

return reverse('post-detail', kwargs={'pk': self.pk})

def like\_count(self):

return self.likes.count()

class Comment(models.Model):

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

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

content = models.TextField()

date\_posted = models.DateTimeField(default=timezone.now)

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

return f'Comment by {self.author.username} on {self.post.title}'

class Event(models.Model):

title = models.CharField(max\_length=200)

description = models.TextField()

location = models.CharField(max\_length=200)

date = models.DateTimeField()

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

attendees = models.ManyToManyField(User, related\_name='events\_attending', blank=True)

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

return self.title

def get\_absolute\_url(self):

return reverse('event-detail', kwargs={'pk': self.pk})

class Message(models.Model):

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

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

content = models.TextField()

date\_sent = models.DateTimeField(default=timezone.now)

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

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

return f'Message from {self.sender.username} to {self.recipient.username}'

### 2. Forms - community\_app/forms.py

from django import forms

from django.contrib.auth.forms import UserCreationForm

from django.contrib.auth.models import User

from .models import Profile, Post, Comment, Event

class UserRegisterForm(UserCreationForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email', 'password1', 'password2']

class UserUpdateForm(forms.ModelForm):

email = forms.EmailField()

class Meta:

model = User

fields = ['username', 'email']

class ProfileUpdateForm(forms.ModelForm):

class Meta:

model = Profile

fields = ['bio', 'location', 'profile\_picture']

class PostForm(forms.ModelForm):

class Meta:

model = Post

fields = ['title', 'content', 'image']

class CommentForm(forms.ModelForm):

class Meta:

model = Comment

fields = ['content']

widgets = {

'content': forms.Textarea(attrs={'rows': 3}),

}

class EventForm(forms.ModelForm):

class Meta:

model = Event

fields = ['title', 'description', 'location', 'date']

widgets = {

'date': forms.DateTimeInput(attrs={'type': 'datetime-local'}),

}

class MessageForm(forms.ModelForm):

class Meta:

model = Message

fields = ['content']

widgets = {

'content': forms.Textarea(attrs={'rows': 3}),

}

### 3. Views - community\_app/views.py

from django.shortcuts import render, redirect, get\_object\_or\_404

from django.contrib import messages

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

from django.contrib.auth.mixins import LoginRequiredMixin, UserPassesTestMixin

from django.views.generic import (

ListView, DetailView, CreateView, UpdateView, DeleteView

)

from django.contrib.auth.models import User

from django.http import JsonResponse

from django.db.models import Q

from .models import Post, Comment, Profile, Event, Message

from .forms import (

UserRegisterForm, UserUpdateForm, ProfileUpdateForm,

PostForm, CommentForm, EventForm, MessageForm

)

def home(request):

context = {

'posts': Post.objects.all().order\_by('-date\_posted')[:5],

'events': Event.objects.filter(date\_\_gte=timezone.now()).order\_by('date')[:3],

}

return render(request, 'community\_app/home.html', context)

def about(request):

return render(request, 'community\_app/about.html', {'title': 'About'})

def register(request):

if request.method == 'POST':

form = UserRegisterForm(request.POST)

if form.is\_valid():

user = form.save()

username = form.cleaned\_data.get('username')

messages.success(request, f'Account created for {username}! You can now log in.')

# Create a profile for the new user

Profile.objects.create(user=user)

return redirect('login')

else:

form = UserRegisterForm()

return render(request, 'community\_app/register.html', {'form': form})

@login\_required

def profile(request, username=None):

if username:

user = get\_object\_or\_404(User, username=username)

else:

user = request.user

user\_posts = Post.objects.filter(author=user).order\_by('-date\_posted')

user\_events = Event.objects.filter(organizer=user).order\_by('-date')

context = {

'user': user,

'posts': user\_posts,

'events': user\_events,

}

return render(request, 'community\_app/profile.html', context)

@login\_required

def edit\_profile(request):

if request.method == 'POST':

u\_form = UserUpdateForm(request.POST, instance=request.user)

p\_form = ProfileUpdateForm(request.POST, request.FILES, instance=request.user.profile)

if u\_form.is\_valid() and p\_form.is\_valid():

u\_form.save()

p\_form.save()

messages.success(request, 'Your profile has been updated!')

return redirect('profile')

else:

u\_form = UserUpdateForm(instance=request.user)

p\_form = ProfileUpdateForm(instance=request.user.profile)

context = {

'u\_form': u\_form,

'p\_form': p\_form,

}

return render(request, 'community\_app/edit\_profile.html', context)

class PostListView(ListView):

model = Post

template\_name = 'community\_app/posts.html'

context\_object\_name = 'posts'

ordering = ['-date\_posted']

paginate\_by = 10

class UserPostListView(ListView):

model = Post

template\_name = 'community\_app/user\_posts.html'

context\_object\_name = 'posts'

paginate\_by = 10

def get\_queryset(self):

user = get\_object\_or\_404(User, username=self.kwargs.get('username'))

return Post.objects.filter(author=user).order\_by('-date\_posted')

class PostDetailView(DetailView):

model = Post

def get\_context\_data(self, \*\*kwargs):

context = super().get\_context\_data(\*\*kwargs)

context['comment\_form'] = CommentForm()

return context

class PostCreateView(LoginRequiredMixin, CreateView):

model = Post

form\_class = PostForm

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

class PostUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):

model = Post

form\_class = PostForm

def form\_valid(self, form):

form.instance.author = self.request.user

return super().form\_valid(form)

def test\_func(self):

post = self.get\_object()

return self.request.user == post.author

class PostDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):

model = Post

success\_url = '/'

def test\_func(self):

post = self.get\_object()

return self.request.user == post.author

@login\_required

def add\_comment(request, pk):

post = get\_object\_or\_404(Post, pk=pk)

if request.method == 'POST':

form = CommentForm(request.POST)

if form.is\_valid():

comment = form.save(commit=False)

comment.post = post

comment.author = request.user

comment.save()

messages.success(request, 'Comment added successfully!')

return redirect('post-detail', pk=post.pk)

return redirect('post-detail', pk=post.pk)

@login\_required

def like\_post(request, pk):

post = get\_object\_or\_404(Post, pk=pk)

if request.user in post.likes.all():

post.likes.remove(request.user)

liked = False

else:

post.likes.add(request.user)

liked = True

return JsonResponse({'liked': liked, 'count': post.like\_count()})

class EventListView(ListView):

model = Event

template\_name = 'community\_app/events.html'

context\_object\_name = 'events'

def get\_queryset(self):

return Event.objects.filter(date\_\_gte=timezone.now()).order\_by('date')

class EventDetailView(DetailView):

model = Event

class EventCreateView(LoginRequiredMixin, CreateView):

model = Event

form\_class = EventForm

def form\_valid(self, form):

form.instance.organizer = self.request.user

return super().form\_valid(form)

class EventUpdateView(LoginRequiredMixin, UserPassesTestMixin, UpdateView):

model = Event

form\_class = EventForm

def form\_valid(self, form):

form.instance.organizer = self.request.user

return super().form\_valid(form)

def test\_func(self):

event = self.get\_object()

return self.request.user == event.organizer

class EventDeleteView(LoginRequiredMixin, UserPassesTestMixin, DeleteView):

model = Event

success\_url = '/events/'

def test\_func(self):

event = self.get\_object()

return self.request.user == event.organizer

@login\_required

def attend\_event(request, pk):

event = get\_object\_or\_404(Event, pk=pk)

if request.user in event.attendees.all():

event.attendees.remove(request.user)

attending = False

else:

event.attendees.add(request.user)

attending = True

return JsonResponse({'attending': attending, 'count': event.attendees.count()})

@login\_required

def inbox(request):

messages\_received = Message.objects.filter(recipient=request.user).order\_by('-date\_sent')

messages\_sent = Message.objects.filter(sender=request.user).order\_by('-date\_sent')

context = {

'messages\_received': messages\_received,

'messages\_sent': messages\_sent,

}

return render(request, 'community\_app/inbox.html', context)

@login\_required

def send\_message(request, recipient\_username):

recipient = get\_object\_or\_404(User, username=recipient\_username)

if request.method == 'POST':

form = MessageForm(request.POST)

if form.is\_valid():

message = form.save(commit=False)

message.sender = request.user

message.recipient = recipient

message.save()

messages.success(request, f'Message sent to {recipient\_username}!')

return redirect('inbox')

else:

form = MessageForm()

context = {

'form': form,

'recipient': recipient,

}

return render(request, 'community\_app/send\_message.html', context)

@login\_required

def search(request):

query = request.GET.get('q', '')

if query:

user\_results = User.objects.filter(username\_\_icontains=query)

post\_results = Post.objects.filter(

Q(title\_\_icontains=query) | Q(content\_\_icontains=query)

)

event\_results = Event.objects.filter(

Q(title\_\_icontains=query) | Q(description\_\_icontains=query)

)

else:

user\_results = User.objects.none()

post\_results = Post.objects.none()

event\_results = Event.objects.none()

context = {

'query': query,

'users': user\_results,

'posts': post\_results,

'events': event\_results,

}

return render(request, 'community\_app/search\_results.html', context)

### 4. URLs - community\_app/urls.py

from django.urls import path

from . import views

from django.contrib.auth import views as auth\_views

urlpatterns = [

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

path('about/', views.about, name='about'),

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

path('login/', auth\_views.LoginView.as\_view(template\_name='community\_app/login.html'), name='login'),

path('logout/', auth\_views.LogoutView.as\_view(template\_name='community\_app/logout.html'), name='logout'),

# Password reset

path('password-reset/',

auth\_views.PasswordResetView.as\_view(template\_name='community\_app/password\_reset.html'),

name='password\_reset'),

path('password-reset/done/',

auth\_views.PasswordResetDoneView.as\_view(template\_name='community\_app/password\_reset\_done.html'),

name='password\_reset\_done'),

path('password-reset-confirm/<uidb64>/<token>/',

auth\_views.PasswordResetConfirmView.as\_view(template\_name='community\_app/password\_reset\_confirm.html'),

name='password\_reset\_confirm'),

path('password-reset-complete/',

auth\_views.PasswordResetCompleteView.as\_view(template\_name='community\_app/password\_reset\_complete.html'),

name='password\_reset\_complete'),

# Profile

path('profile/', views.profile, name='profile'),

path('profile/<str:username>/', views.profile, name='user-profile'),

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

# Posts

path('posts/', views.PostListView.as\_view(), name='posts'),

path('post/<int:pk>/', views.PostDetailView.as\_view(), name='post-detail'),

path('post/new/', views.PostCreateView.as\_view(), name='post-create'),

path('post/<int:pk>/update/', views.PostUpdateView.as\_view(), name='post-update'),

path('post/<int:pk>/delete/', views.PostDeleteView.as\_view(), name='post-delete'),

path('post/<int:pk>/comment/', views.add\_comment, name='add-comment'),

path('post/<int:pk>/like/', views.like\_post, name='like-post'),

# Events

path('events/', views.EventListView.as\_view(), name='events'),

path('event/<int:pk>/', views.EventDetailView.as\_view(), name='event-detail'),

path('event/new/', views.EventCreateView.as\_view(), name='event-create'),

path('event/<int:pk>/update/', views.EventUpdateView.as\_view(), name='event-update'),

path('event/<int:pk>/delete/', views.EventDeleteView.as\_view(), name='event-delete'),

path('event/<int:pk>/attend/', views.attend\_event, name='attend-event'),

# Messages

path('inbox/', views.inbox, name='inbox'),

path('send-message/<str:recipient\_username>/', views.send\_message, name='send-message'),

# Search

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

]

### 5. Project URLs - project\_name/urls.py

from django.contrib import admin

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

path('admin/', admin.site.urls),

path('', include('community\_app.urls')),

]

# Add these lines to serve media files in development

if settings.DEBUG:

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

### 6. Admin Configuration - community\_app/admin.py

from django.contrib import admin

from .models import Profile, Post, Comment, Event, Message

class CommentInline(admin.TabularInline):

model = Comment

extra = 0

class PostAdmin(admin.ModelAdmin):

list\_display = ('title', 'author', 'date\_posted')

list\_filter = ('date\_posted', 'author')

search\_fields = ('title', 'content')

inlines = [CommentInline]

class EventAdmin(admin.ModelAdmin):

list\_display = ('title', 'organizer', 'date')

list\_filter = ('date', 'organizer')

search\_fields = ('title', 'description', 'location')

class ProfileAdmin(admin.ModelAdmin):

list\_display = ('user', 'location', 'joined\_date')

search\_fields = ('user\_\_username', 'location')

class MessageAdmin(admin.ModelAdmin):

list\_display = ('sender', 'recipient', 'date\_sent', 'is\_read')

list\_filter = ('date\_sent', 'is\_read')

search\_fields = ('content', 'sender\_\_username', 'recipient\_\_username')

admin.site.register(Profile, ProfileAdmin)

admin.site.register(Post, PostAdmin)

admin.site.register(Comment)

admin.site.register(Event, EventAdmin)

admin.site.register(Message, MessageAdmin)

### 7. Settings Configuration - project\_name/settings.py (additions)

# Add these settings to your settings.py file

# Media files

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

MEDIA\_URL = '/media/'

# Authentication settings

LOGIN\_REDIRECT\_URL = 'home'

LOGIN\_URL = 'login'

# Email settings (for password reset, etc.)

EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'

EMAIL\_HOST = 'smtp.example.com' # Replace with your SMTP server

EMAIL\_PORT = 587

EMAIL\_USE\_TLS = True

EMAIL\_HOST\_USER = 'your-email@example.com' # Replace with your email

EMAIL\_HOST\_PASSWORD = 'your-password' # Replace with your password

# For development, you can use the console email backend

# EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'

### 8. App Configuration - community\_app/apps.py

from django.apps import AppConfig

class CommunityAppConfig(AppConfig):

default\_auto\_field = 'django.db.models.BigAutoField'

name = 'community\_app'

def ready(self):

import community\_app.signals

### 9. Signals - community\_app/signals.py

from django.db.models.signals import post\_save

from django.contrib.auth.models import User

from django.dispatch import receiver

from .models import Profile

@receiver(post\_save, sender=User)

def create\_profile(sender, instance, created, \*\*kwargs):

if created:

Profile.objects.create(user=instance)

@receiver(post\_save, sender=User)

def save\_profile(sender, instance, \*\*kwargs):

instance.profile.save()