Models.py

```
from django.contrib.auth.models import User
from django.db import models
from django.utils.timezone import now

class Post(models.Model):
    user = models.ForeignKey(User,
on_delete=models.CASCADE, related_name="posts")
    content = models.TextField()
    timestamp = models.DateTimeField(default=now)
    likes = models.ManyToManyField(User,
related_name="liked_posts", blank=True)

    def like_count(self):
        return self.likes.count()

    def __str__(self):
        return f"{self.user.username}: {self.content[:30]}"

class Profile(models.Model):
    user = models.OneToOneField(User,
on_delete=models.CASCADE, related_name="profile")
    followers = models.ManyToManyField(User,
related_name="following", blank=True)

    def follower_count(self):
        return self.followers.count()

    def following_count(self):
        return self.user.following.count()
```

Views.py

```
from django.shortcuts import render, redirect,
get_object_or_404
from django.contrib.auth.decorators import login_required
from django.core.paginator import Paginator
from .models import Post, Profile
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
import json

def all_posts(request):
    posts = Post.objects.all().order_by("-timestamp")
    paginator = Paginator(posts, 10)  # 10 posts per page
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, "app/all_posts.html", {"page_obj":
page_obj})

@login_required
def new_post(request):
    if request.method == "POST":
        content = request.POST["content"]
        Post.objects.create(user=request.user,
content=content)
        return redirect("all_posts")
    return render(request, "app/new_post.html")

def profile_page(request, username):
    profile = get_object_or_404(Profile,
user__username=username)
    posts = profile.user.posts.all().order_by("-timestamp")
    is_following = request.user in profile.followers.all() if
request.user.is_authenticated else False
    return render(request, "app/profile.html", {
        "profile": profile,
        "posts": posts,
        "is_following": is_following
    })

@login_required
def follow_unfollow(request, username):
    profile = get_object_or_404(Profile,
user__username=username)
    if request.user in profile.followers.all():
        profile.followers.remove(request.user)
    else:
        profile.followers.add(request.user)
    return redirect("profile", username=username)

@login_required
def following(request):
    user_following = request.user.following.all()
    posts =
Post.objects.filter(user__profile__in=user_following).order_
by("-timestamp")
    paginator = Paginator(posts, 10)  # 10 posts per page
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    return render(request, "app/following.html", {"page_obj":
page_obj})

@csrf_exempt
@login_required
def edit_post(request, post_id):
    post = get_object_or_404(Post, id=post_id,
user=request.user)
    if request.method == "PUT":
        data = json.loads(request.body)
        post.content = data["content"]
        post.save()
        return JsonResponse({"message": "Post updated
successfully."}, status=200)

@csrf_exempt
@login_required
def toggle_like(request, post_id):
    post = get_object_or_404(Post, id=post_id)
    if request.user in post.likes.all():
        post.likes.remove(request.user)
    else:
        post.likes.add(request.user)
    return JsonResponse({"likes": post.like_count()},
status=200)
```

Templates

new_post.html

```
<form method="POST">
    {% csrf_token %}
    <textarea name="content" rows="4" placeholder="Write
your post here..."></textarea>
    <button type="submit">Post</button>
</form>
```

all_posts.html

```
{% for post in page_obj %}
<div>
    <h3>{{ post.user.username }}</h3>
    <p>{{ post.content }}</p>
    <p>Likes: {{ post.like_count }}</p>
    <button onclick="toggleLike({{ post.id }})">
        {% if user in post.likes.all %} Unlike {% else %} Like {% endif
%}
    </button>
</div>
{% endfor %}
<div>
    {% if page_obj.has_previous %}
        <a href="?page={{ page_obj.previous_page_number
}}">Previous</a>
    {% endif %}
    {% if page_obj.has_next %}
        <a href="?page={{ page_obj.next_page_number
}}">Next</a>
    {% endif %}
</div>
```

app.js

```
function toggleLike(postId) {
    fetch(`/toggle-like/${postId}`, {
        method: "POST",
        headers: {"X-CSRFToken": getCsrfToken() }
    })
    .then(response => response.json())
    .then(data => {
        document.querySelector(`#likes-${postId}`).innerText
= data.likes;
    });
}

function getCsrfToken() {
    return
document.querySelector("[name=csrfmiddlewaretoken]").
value;
}
```

urls.py

```
from django.urls import path
from . import views

urlpatterns = [
    path("", views.all_posts, name="all_posts"),
    path("new", views.new_post, name="new_post"),
    path("profile/<str:username>", views.profile_page,
name="profile"),
    path("follow-unfollow/<str:username>",
views.follow_unfollow, name="follow_unfollow"),
    path("following", views.following, name="following"),
    path("edit-post/<int:post_id>", views.edit_post,
name="edit_post"),
    path("toggle-like/<int:post_id>", views.toggle_like,
name="toggle_like"),
]
```