1. Frontend (HTML/CSS/JavaScript)

HTML

This is the basic structure for a user profile page where posts, comments, likes, and follow features will be displayed.

html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Social Media Platform</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>My Social Media</h1>
        <nav>
            <a href="/profile">Profile</a>
            <a href="/feed">Feed</a>
        </nav>
    </header>

    <main>
        <div id="profile">
            <h2>User Profile</h2>
            <button id="follow-btn">Follow</button>
            <p id="followers-count">Followers: 0</p>
        </div>

        <section id="posts">
            <h2>Posts</h2>
            <div id="post-list">
                <!-- Posts will be dynamically loaded here -->
```

```
            </div>
        </section>

        <section id="create-post">
            <h2>Create a New Post</h2>
            <textarea id="post-content" placeholder="What's on
your mind?"></textarea>
            <button id="post-btn">Post</button>
        </section>
    </main>

    <script src="scripts.js"></script>
</body>
</html>
```

CSS (styles.css)

```
css
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header {
    background-color: #333;
    color: white;
    padding: 15px;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

nav a {
    color: white;
    margin-left: 20px;
    text-decoration: none;
```

```
}

main {
    margin: 20px;
}

#posts, #create-post {
    margin-top: 20px;
}

#post-list .post {
    border: 1px solid #ddd;
    padding: 15px;
    margin-bottom: 10px;
}

button {
    background-color: #28a745;
    color: white;
    padding: 10px;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: #218838;
}
```

JavaScript (scripts.js)
This handles post creation, comments, likes, and following functionality.

```
javascript
let posts = [];
let followersCount = 0;
```

```
document.addEventListener('DOMContentLoaded', () => {
    loadPosts();
    document.getElementById('post-
btn').addEventListener('click', createPost);
    document.getElementById('follow-
btn').addEventListener('click', followUser);
});

function loadPosts() {
    const postList = document.getElementById('post-list');
    postList.innerHTML = ''; // Clear the post list

    posts.forEach((post, index) => {
        const postDiv = document.createElement('div');
        postDiv.classList.add('post');
        postDiv.innerHTML = `
            <p>${post.content}</p>
            <button onclick="likePost(${index})">Like
(${post.likes})</button>
            <button
onclick="showComments(${index})">Comments
(${post.comments.length})</button>
            <div id="comments-${index}" class="comments">
                ${post.comments.map(comment =>
`<p>${comment}</p>`).join('')}
                <textarea id="comment-${index}" placeholder="Add
a comment"></textarea>
                <button
onclick="addComment(${index})">Comment</button>
            </div>
        `;
        postList.appendChild(postDiv);
    });
}

function createPost() {
```

```
    const content = document.getElementById('post-content').value;
    if (content.trim()) {
        posts.push({ content, likes: 0, comments: [] });
        document.getElementById('post-content').value = '';
        loadPosts();
    }
}

function likePost(index) {
    posts[index].likes += 1;
    loadPosts();
}

function showComments(index) {
    const commentsDiv = document.getElementById(`comments-${index}`);
    commentsDiv.style.display = commentsDiv.style.display === 'none' ? 'block' : 'none';
}

function addComment(index) {
    const commentText = document.getElementById(`comment-${index}`).value;
    if (commentText.trim()) {
        posts[index].comments.push(commentText);
        document.getElementById(`comment-${index}`).value = '';
        loadPosts();
    }
}

function followUser() {
    followersCount += 1;
    document.getElementById('followers-count').innerText = `Followers: ${followersCount}`;
}
```

---

2. Backend Setup

We will now set up the backend using **Django** or **Express.js** for user profiles, posts, comments, likes, and follows.

Django Backend

1. Install Django:

```
pip install django
```

2. Set Up Project and App:

```
django-admin startproject socialmedia
cd socialmedia
django-admin startapp platform
```

3. Models for User, Post, Comment, and Follow (platform/models.py):

```
from django.contrib.auth.models import User
from django.db import models

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

class Comment(models.Model):
    post = models.ForeignKey(Post,
on_delete=models.CASCADE)
```

```
    user = models.ForeignKey(User,
on_delete=models.CASCADE)
    content = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

class Follow(models.Model):
    follower = models.ForeignKey(User,
related_name="following", on_delete=models.CASCADE)
    followed = models.ForeignKey(User,
related_name="followers", on_delete=models.CASCADE)
```

4. Views to Handle Posts, Likes, Comments, and Follows (platform/views.py):

```
from django.http import JsonResponse
from django.shortcuts import get_object_or_404
from .models import Post, Comment, Follow

def post_list(request):
    posts = Post.objects.all()
    post_data = [{
        "id": p.id,
        "content": p.content,
        "likes": p.likes.count(),
        "comments": [c.content for c in p.comment_set.all()]
    } for p in posts]
    return JsonResponse(post_data, safe=False)

def create_post(request):
    if request.method == 'POST':
        content = request.POST['content']
        post = Post.objects.create(user=request.user,
content=content)
        return JsonResponse({"status": "Post created", "post_id":
post.id})

def like_post(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_count": post.likes.count()})

def follow_user(request, user_id):
    followed = get_object_or_404(User, id=user_id)
    Follow.objects.get_or_create(follower=request.user,
followed=followed)
    return JsonResponse({"status": "Following"})
```

5.URLs (socialmedia/urls.py):

```
from django.contrib import admin
from django.urls import path
from platform import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('posts/', views.post_list),
    path('posts/create/', views.create_post),
    path('posts/<int:post_id>/like/', views.like_post),
    path('users/<int:user_id>/follow/', views.follow_user),
]
```

6. Migrate and Run:

```
python manage.py migrate
python manage.py runserver
```

Express.js Backend

1. Install Dependencies:

```
npm init -y
```

```
npm install express mongoose body-parser
```

2. Set Up Express (server.js):

```
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');

const app = express();
app.use(bodyParser.json());

mongoose.connect('mongodb://localhost:27017/socialmedia',
{ useNewUrlParser: true });

const UserSchema = new mongoose.Schema({ name: String });
const User = mongoose.model('User', UserSchema);

const PostSchema = new mongoose.Schema({
    user: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    content: String,
    likes: [{ type: mongoose.Schema.Types.ObjectId, ref: 'User' }]
});
const Post = mongoose.model('Post', PostSchema);

app.post('/posts', async (req, res) => {
    const
```