Let's build a **Django Blog Project** step by step with:-

- □ User Signup
- □ Login & Logout
- □ Create, Read, Update, Delete (CRUD) blog posts
- □ Each user manages only their own posts

We'll go step-by-step from installation to running the server.

□ STEP 1: Install Django

Make sure Python is installed:

```
python --version
```

Create Virtual Environment (Recommended)

```
python -m venv venv
```

Activate it:

Windows:

```
venv\Scripts\activate
```

Install Django

```
pip install django
```

Check version:

```
django-admin --version
```

□ STEP 2: Create Django Project

```
django-admin startproject myblog
```

Go into project folder:

```
cd myblog
```

Run server to test:

```
python manage.py runserver
```

Open browser:

```
http://127.0.0.1:8000/
```

Stop server with `CTRL + C`

□ STEP 3: Create Blog App

```
python manage.py startapp blog
```

Project structure now:

```
myblog/
    manage.py
    myblog/
    blog/
```

□ STEP 4: Register App in settings.py

Open:

```
myblog/settings.py
```

Find `INSTALLED_APPS` and add:

```
INSTALLED_APPS = [
    ...
    'blog',
]
```

□ STEP 5: Create Blog Model

Open:

```
blog/models.py
```

Replace with:

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

```
class Post(models.Model):
    title = models.CharField(max_length=200)
    content = models.TextField()
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title
```

□ STEP 6: Make Migrations

```
python manage.py makemigrations
python manage.py migrate
```

□ STEP 7: Create Superuser (Optional)

```
python manage.py createsuperuser
```

Follow prompts.

□ STEP 8: Create Forms

Create file:

```
blog/forms.py
```

Add:

```
from django import forms
from .models import Post
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User

class SignUpForm(UserCreationForm):
    email = forms.EmailField()

    class Meta:
        model = User
        fields = ['username', 'email', 'password1', 'password2']


class PostForm(forms.ModelForm):
    class Meta:
```

```
        model = Post
        fields = ['title', 'content']
```

□ STEP 9: Create Views

Open:

`blog/views.py`

Add:

```
from django.shortcuts import render, redirect, get_object_or_404
from .models import Post
from .forms import PostForm, SignUpForm
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import logout

# Home - List Posts
@login_required
def post_list(request):
    posts = Post.objects.filter(author=request.user)
    return render(request, 'blog/post_list.html', {'posts': posts})


# Create Post
@login_required
def post_create(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('post_list')
    else:
        form = PostForm()
    return render(request, 'blog/post_form.html', {'form': form})


# Update Post
@login_required
def post_update(request, pk):
    post = get_object_or_404(Post, pk=pk, author=request.user)
    form = PostForm(request.POST or None, instance=post)
    if form.is_valid():
        form.save()
        return redirect('post_list')
    return render(request, 'blog/post_form.html', {'form': form})
```

```
# Delete Post
@login_required
def post_delete(request, pk):
    post = get_object_or_404(Post, pk=pk, author=request.user)
    if request.method == 'POST':
        post.delete()
        return redirect('post_list')
    return render(request, 'blog/post_confirm_delete.html', {'post': post})


# Signup
def signup(request):
    if request.method == 'POST':
        form = SignUpForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('post_list')
    else:
        form = SignUpForm()
    return render(request, 'blog/signup.html', {'form': form})


# Logout
def user_logout(request):
    logout(request)
    return redirect('login')
```

□ STEP 10: Configure URLs

blog/urls.py (Create this file)

```
from django.urls import path
from . import views
from django.contrib.auth import views as auth_views

urlpatterns = [
    path('', views.post_list, name='post_list'),
    path('create/', views.post_create, name='post_create'),
    path('update/<int:pk>/', views.post_update, name='post_update'),
    path('delete/<int:pk>/', views.post_delete, name='post_delete'),
    path('signup/', views.signup, name='signup'),
    path('login/',
auth_views.LoginView.as_view(template_name='blog/login.html'), name='login'),
    path('logout/', views.user_logout, name='logout'),
]
```

Update Main URLs

Open:

`myblog/urls.py`

Modify:

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('blog.urls')),
]
```

□ STEP 11: Create Templates

Create folder structure:-

```
blog/
    templates/
        blog/
```

---

base.html

```
<!DOCTYPE html>
<html>
<head>
    <title>My Blog</title>
</head>
<body>

<h1>My Blog</h1>

{% if user.is_authenticated %}
    <a href="{% url 'post_create' %}">Create Post</a> |
    <a href="{% url 'logout' %}">Logout</a>
{% else %}
    <a href="{% url 'login' %}">Login</a> |
    <a href="{% url 'signup' %}">Signup</a>
{% endif %}

<hr>

{% block content %}
{% endblock %}
```

```
</body>
</html>
```

---

post_list.html

```
{% extends 'blog/base.html' %}

{% block content %}
<h2>Your Posts</h2>

{% for post in posts %}
    <h3>{{ post.title }}</h3>
    <p>{{ post.content }}</p>
    <a href="{% url 'post_update' post.pk %}">Edit</a>
    <a href="{% url 'post_delete' post.pk %}">Delete</a>
    <hr>
{% empty %}
    <p>No posts yet.</p>
{% endfor %}

{% endblock %}
```

---

post_form.html

```
{% extends 'blog/base.html' %}

{% block content %}
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Save</button>
</form>
{% endblock %}
```

---

post_confirm_delete.html

```
{% extends 'blog/base.html' %}

{% block content %}
<form method="post">
    {% csrf_token %}
    <p>Are you sure you want to delete "{{ post.title }}"?</p>
    <button type="submit">Yes</button>
</form>
{% endblock %}
```

---

signup.html

```
{% extends 'blog/base.html' %}

{% block content %}
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Signup</button>
</form>
{% endblock %}
```

---

login.html

```
{% extends 'blog/base.html' %}

{% block content %}
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
</form>
{% endblock %}
```

□ STEP 12: Configure Login Redirect

In `settings.py` add:

```
LOGIN_REDIRECT_URL = 'post_list'
LOGOUT_REDIRECT_URL = 'login'
LOGIN_URL = 'login'
```

□ STEP 13: Run Server

```
python manage.py runserver
```

Visit:

```
http://127.0.0.1:8000/signup/
```

□ DONE!

You now have:

- User Registration
- Login / Logout
- User-specific blog posts
- Full CRUD functionality

Perfect Let's upgrade your blog with:

- **Bootstrap styling**
- **Pagination**
- Clean professional UI

We'll do it step-by-step.

PART 1 — Add Bootstrap Styling

We'll use Bootstrap CDN (easiest method).

Step 1: Update `base.html`

Open:

```
blog/templates/blog/base.html
```

Replace with this:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Blog</title>

    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css
"
          rel="stylesheet">
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="{% url 'post_list' %}">My Blog</a>

    <div>
      {% if user.is_authenticated %}
        <a class="btn btn-success btn-sm" href="{% url 'post_create' %}">New
```

```
Post</a>
        <a class="btn btn-danger btn-sm" href="{% url 'logout' %}">Logout</a>
      {% else %}
        <a class="btn btn-primary btn-sm" href="{% url 'login' %}">Login</a>
        <a class="btn btn-warning btn-sm" href="{% url 'signup'
%}">Signup</a>
      {% endif %}
    </div>
  </div>
</nav>

<div class="container mt-4">
    {% block content %}
    {% endblock %}
</div>

</body>
</html>
```

□ Step 2: Improve `post_list.html`

Replace with:

```
{% extends 'blog/base.html' %}

{% block content %}
<h2 class="mb-4">Your Posts</h2>

{% for post in posts %}
<div class="card mb-3 shadow-sm">
  <div class="card-body">
    <h4 class="card-title">{{ post.title }}</h4>
    <p class="card-text">{{ post.content|truncatewords:30 }}</p>

    <a href="{% url 'post_update' post.pk %}" class="btn btn-sm btn-
primary">Edit</a>
    <a href="{% url 'post_delete' post.pk %}" class="btn btn-sm btn-
danger">Delete</a>
  </div>
</div>
{% empty %}
<div class="alert alert-info">No posts yet.</div>
{% endfor %}

{% endblock %}
```

□ Step 3: Improve Forms (`post_form.html`, login, signup)

Replace form pages with:

```
{% extends 'blog/base.html' %}

{% block content %}
<div class="card shadow p-4">
  <form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit" class="btn btn-success">Submit</button>
  </form>
</div>
{% endblock %}
```

Now your site looks modern

PART 2 — Add Pagination

Now we add pagination so only 5 posts show per page.

Step 1: Update `views.py`

Open:

```
blog/views.py
```

Modify `post_list`:

```
from django.core.paginator import Paginator

@login_required
def post_list(request):
    posts = Post.objects.filter(author=request.user).order_by('-created_at')

    paginator = Paginator(posts, 5)  # Show 5 posts per page
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)

    return render(request, 'blog/post_list.html', {'page_obj': page_obj})
```

□ Step 2: Update `post_list.html`

```
{% extends 'blog/base.html' %}

{% block content %}
<h2 class="mb-4">Your Posts</h2>

{% for post in page_obj %}
<div class="card mb-3 shadow-sm">
  <div class="card-body">
    <h4 class="card-title">{{ post.title }}</h4>
    <p class="card-text">{{ post.content|truncatewords:30 }}</p>

    <a href="{% url 'post_update' post.pk %}" class="btn btn-sm btn-primary">Edit</a>
    <a href="{% url 'post_delete' post.pk %}" class="btn btn-sm btn-danger">Delete</a>
  </div>
</div>
{% empty %}
<div class="alert alert-info">No posts yet.</div>
{% endfor %}

<!-- Pagination -->
<nav>
```

```
<ul class="pagination justify-content-center">

 {% if page_obj.has_previous %}
  <li class="page-item">
    <a class="page-link" href="?page=1">First</a>
  </li>
  <li class="page-item">
    <a class="page-link" href="?page={{ page_obj.previous_page_number }}">Previous</a>
  </li>
 {% endif %}

 <li class="page-item active">
  <span class="page-link">
    Page {{ page_obj.number }} of {{ page_obj.paginator.num_pages }}
  </span>
 </li>

 {% if page_obj.has_next %}
  <li class="page-item">
    <a class="page-link" href="?page={{ page_obj.next_page_number }}">Next</a>
  </li>
  <li class="page-item">
    <a class="page-link" href="?page={{ page_obj.paginator.num_pages }}">Last</a>
```

```
    </li>
  {% endif %}


 </ul>
</nav>


{% endblock %}
```

Restart Server

```
python manage.py runserver
```

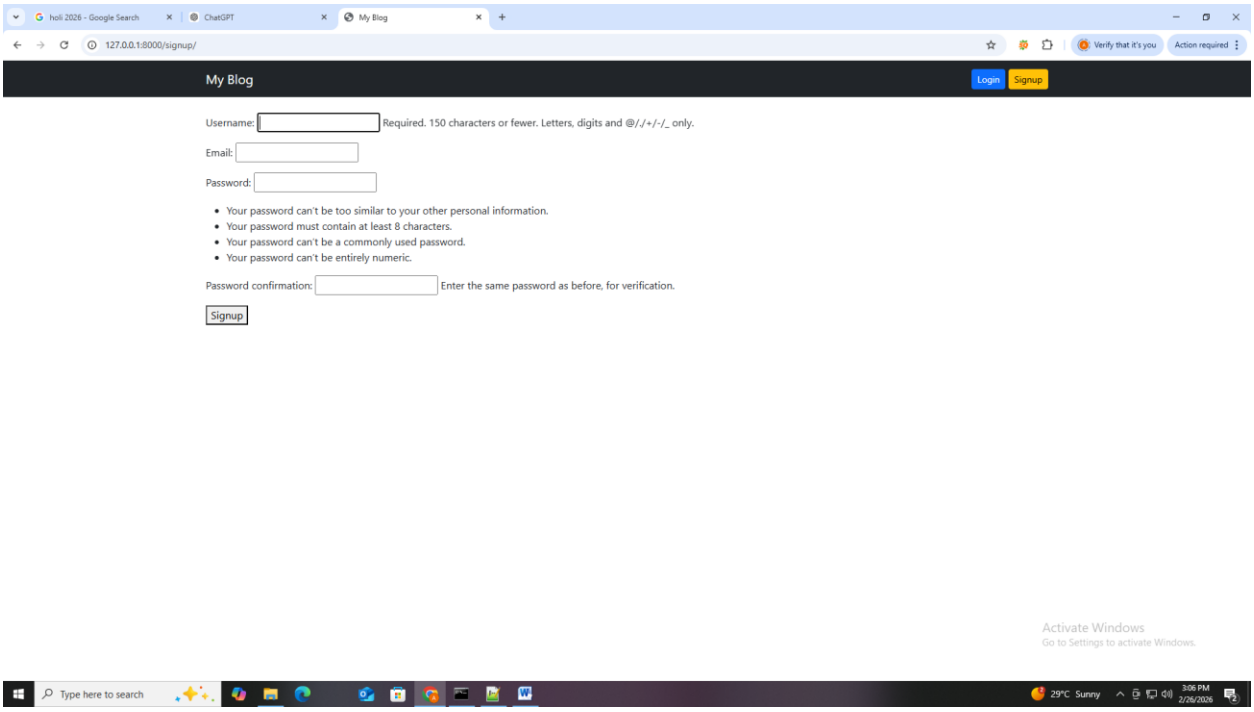
Now:

- ✔ Styled UI
- ✔ Cards layout
- ✔ Navbar
- ✔ Pagination working
- ✔ Professional look

You will see like this :-

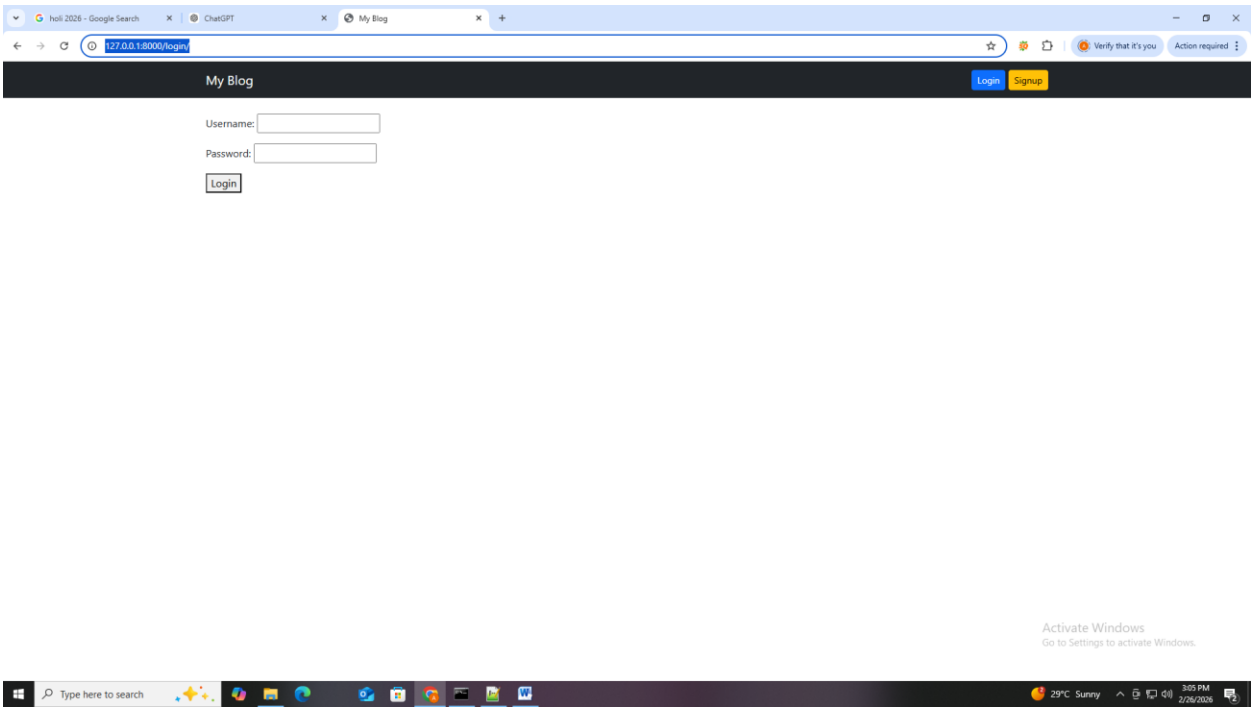
For signup :-

http://127.0.0.1:8000/signup/

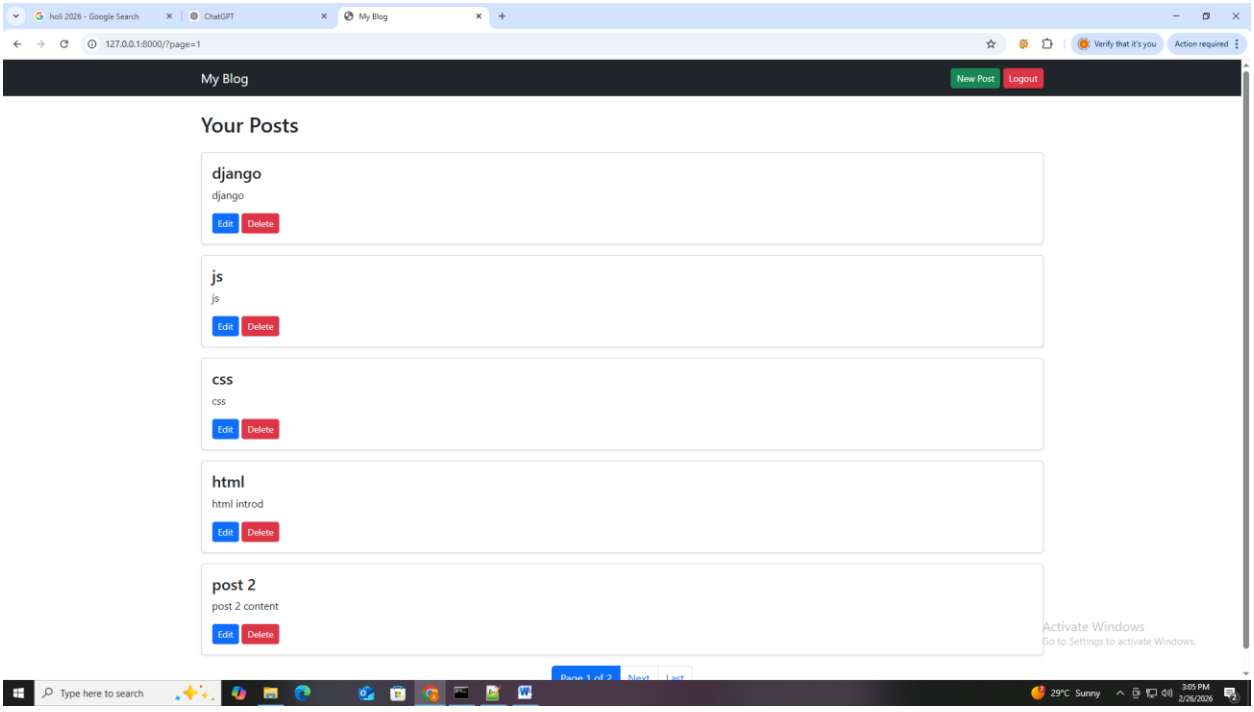


Login page :-

http://127.0.0.1:8000/login/



And after login you will see:-



2️⃣ Add a Function-Based Login View in views.py

Now create your own login function:

```
from django.contrib.auth import authenticate, login
from django.contrib.auth.forms import AuthenticationForm

def user_login(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data.get('username')
            password = form.cleaned_data.get('password')
            user = authenticate(username=username, password=password)

            if user is not None:
                login(request, user)
                return redirect('post_list')
    else:
        form = AuthenticationForm()

    return render(request, 'blog/login.html', {'form': form})
```

🔧 3️⃣ Update urls.py

Replace this:

```
path('login/', auth_views.LoginView.as_view(template_name='blog/login.html'),
name='login'),
```

With this:

```
from . import views

path('login/', views.user_login, name='login'),
```

🔧 4️⃣ Simple login.html Template Example

Make sure you have:

```
templates/blog/login.html
```

```
<h2>Login</h2>

<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
</form>
```

□ What's Happening in Your Login Function?

Step-by-step:

1. If method is POST → user submitted form
2. `AuthenticationForm` validates username & password
3. `authenticate()` checks credentials
4. `login()` creates session
5. Redirect to `'post_list'`
6. If GET request → just show empty form

□ Difference Between Class-Based and Function-Based

LoginView (CBV)	Function-Based
Shorter	More control
Built-in redirect handling	You manually control logic
Less code	More customizable