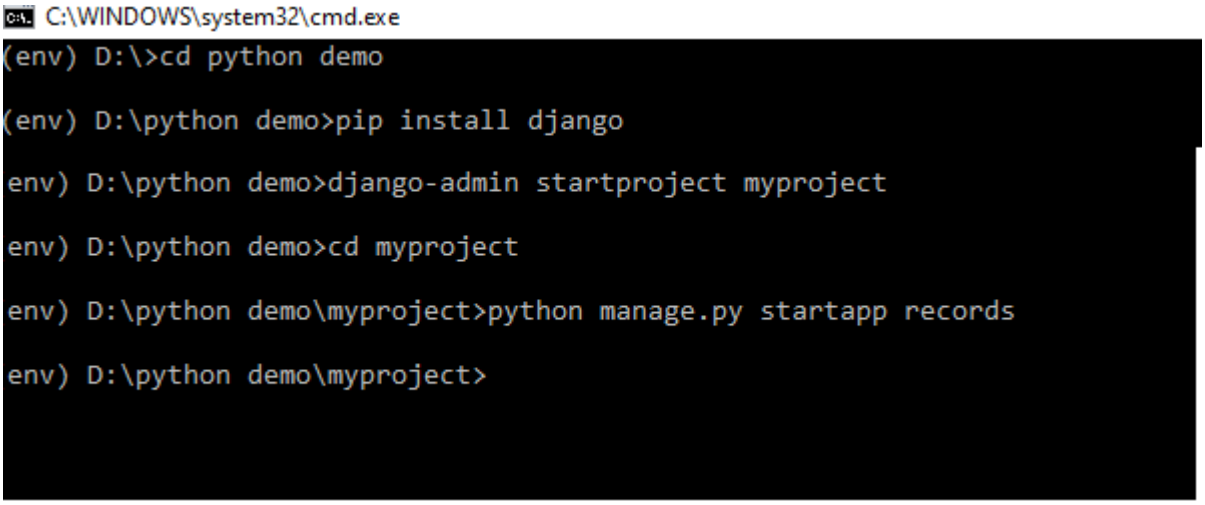
Here is a **complete Django CRUD + User Authentication project** (Signup, Login, Logout, Create, Read, Update, Delete). This is a **very common internship/interview project** and good for your portfolio.



---

1 Project Structure

Your Django project should look like this:

```
myproject/
│
├── myproject/
│   ├── settings.py
│   ├── urls.py
│
├── records/
│   ├── migrations/
│   ├── templates/
│   │   ├── records/
│   │   │   ├── record_list.html
│   │   │   ├── record_form.html
│   │   │   └── record_confirm_delete.html
│   │   └── registration/
│   │       ├── login.html
│   │       └── signup.html
│   │
│   ├── models.py
│   ├── forms.py
│   ├── views.py
│   ├── urls.py
│
└── manage.py
```

2️⃣ models.py

```
from django.db import models

class Record(models.Model):

    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=15)
    city = models.CharField(max_length=100)

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

Run migrations:-

python manage.py makemigrations
python manage.py migrate

```
(env) D:\python demo\myproject>python manage.py makemigrations
Migrations for 'records':
  records\migrations\0001_initial.py
    + Create model Record

(env) D:\python demo\myproject>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, records, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying records.0001_initial... OK
  Applying sessions.0001_initial... OK
```

3 forms.py

```
from django import forms
from .models import Record


class RecordForm(forms.ModelForm):

    class Meta:
        model = Record
        fields = ['name', 'email', 'phone', 'city']
```

---

4 views.py

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import login
from django.contrib.auth.decorators import login_required

from .models import Record
from .forms import RecordForm


# Signup
def signup(request):

    if request.method == "POST":
        form = UserCreationForm(request.POST)

        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('record_list')

    else:
        form = UserCreationForm()

    return render(request, 'registration/signup.html', {'form': form})


# READ
@login_required
def record_list(request):

    records = Record.objects.all()

    return render(request, 'records/record_list.html', {'records': records})
```

```
# CREATE
@login_required
def record_create(request):

    if request.method == "POST":

        form = RecordForm(request.POST)

        if form.is_valid():
            form.save()
            return redirect('record_list')

    else:
        form = RecordForm()

    return render(request, 'records/record_form.html', {'form': form})


# UPDATE
@login_required
def record_update(request, pk):

    record = get_object_or_404(Record, pk=pk)

    if request.method == "POST":

        form = RecordForm(request.POST, instance=record)

        if form.is_valid():
            form.save()
            return redirect('record_list')

    else:
        form = RecordForm(instance=record)

    return render(request, 'records/record_form.html', {'form': form})


# DELETE
@login_required
def record_delete(request, pk):

    record = get_object_or_404(Record, pk=pk)

    if request.method == "POST":
        record.delete()
        return redirect('record_list')

    return render(request, 'records/record_confirm_delete.html', {'record':
record})
```

5️⃣ records/urls.py

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

urlpatterns = [

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

    path('create/', views.record_create, name='record_create'),

    path('update/<int:pk>/', views.record_update, name='record_update'),

    path('delete/<int:pk>/', views.record_delete, name='record_delete'),

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

]
```

6️⃣ Main urls.py

Inside **project urls.py**

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

urlpatterns = [

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

    path('', include('records.urls')),

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

    path('logout/', auth_views.LogoutView.as_view(), name='logout'),

]
```

7️⃣ settings.py

Add this at the bottom:

```
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = '/login/'
LOGIN_URL = '/login/'
```

8️⃣ Templates

record_list.html

```
<h2>Records</h2>

<a href="{% url 'record_create' %}">Add Record</a>
<form method="post" action="{% url 'logout' %}">
    {% csrf_token %}
    <button type="submit">Logout</button>
</form>

<table border="1">

<tr>
<th>Name</th>
<th>Email</th>
<th>phone</th>
<th>city </th>
<th>Actions</th>
</tr>

{% for record in records %}

<tr>

<td>{{ record.name }}</td>
<td>{{ record.email }}</td>
<td>{{ record.phone}}</td>
<td>{{ record.city}}</td>

<td>

<a href="{% url 'record_update' record.id %}">Edit</a>
```

```
<a href="{% url 'record_delete' record.id %}">Delete</a>

</td>

</tr>

{% endfor %}

</table>
```

---

record_form.html

```
<h2>Record Form</h2>

<form method="POST">

{% csrf_token %}

{{ form.as_p }}

<button type="submit">Save</button>

</form>

<a href="{% url 'record_list' %}">Back</a>
```

---

record_confirm_delete.html

```
<h2>Delete Record</h2>

<p>Are you sure you want to delete {{ record.name }}?</p>

<form method="POST">

{% csrf_token %}

<button type="submit">Delete</button>

</form>

<a href="{% url 'record_list' %}">Cancel</a>
```

---

signup.html

```
<h2>Signup</h2>

<form method="POST">

{% csrf_token %}
```

```
{{ form.as_p }}

<button type="submit">Signup</button>

</form>

<a href="{% url 'login' %}">Login</a>
```

---

login.html

```
<h2>Login</h2>

<form method="POST">

{% csrf_token %}

{{ form.as_p }}

<button type="submit">Login</button>

</form>

<a href="{% url 'signup' %}">Signup</a>
```

---

9 Run Project

Run migrations:

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

Start server:

```
python manage.py runserver
```

Open:

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

What This Project Shows in Interviews

This project proves you understand:

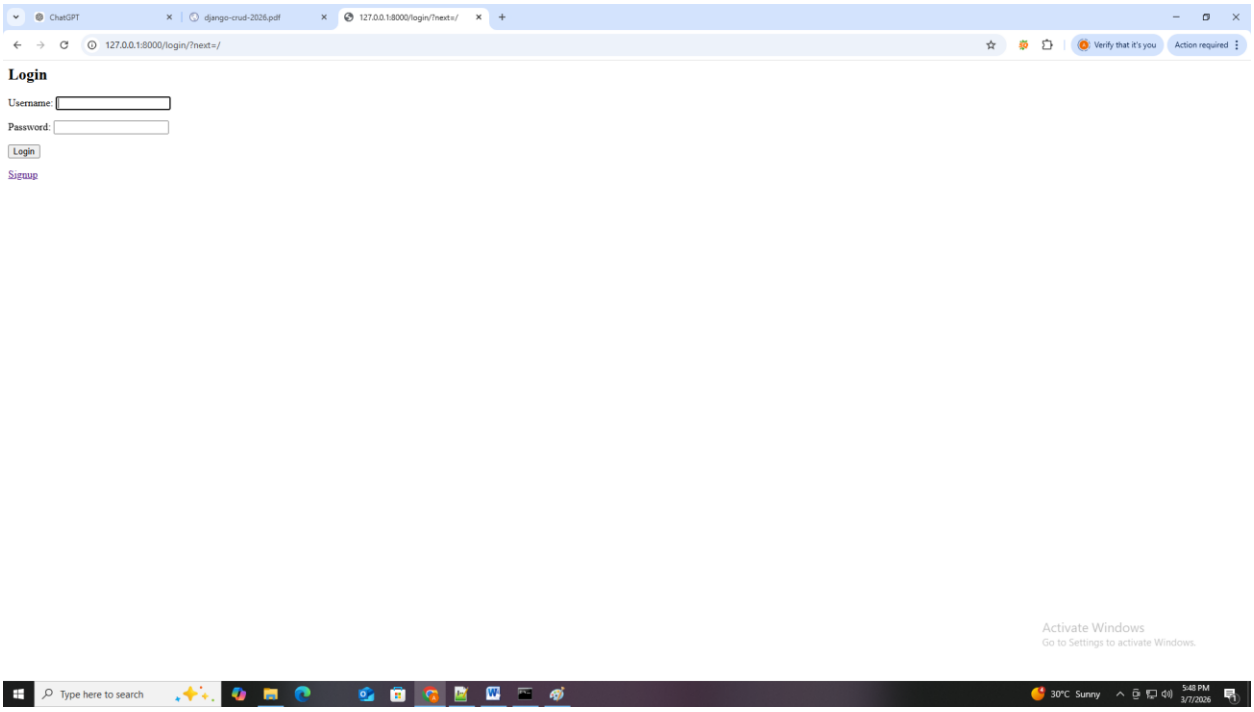
 Django Models
 Django Forms
 CRUD operations
 Authentication system
 URL routing
 Templates
 Login protection

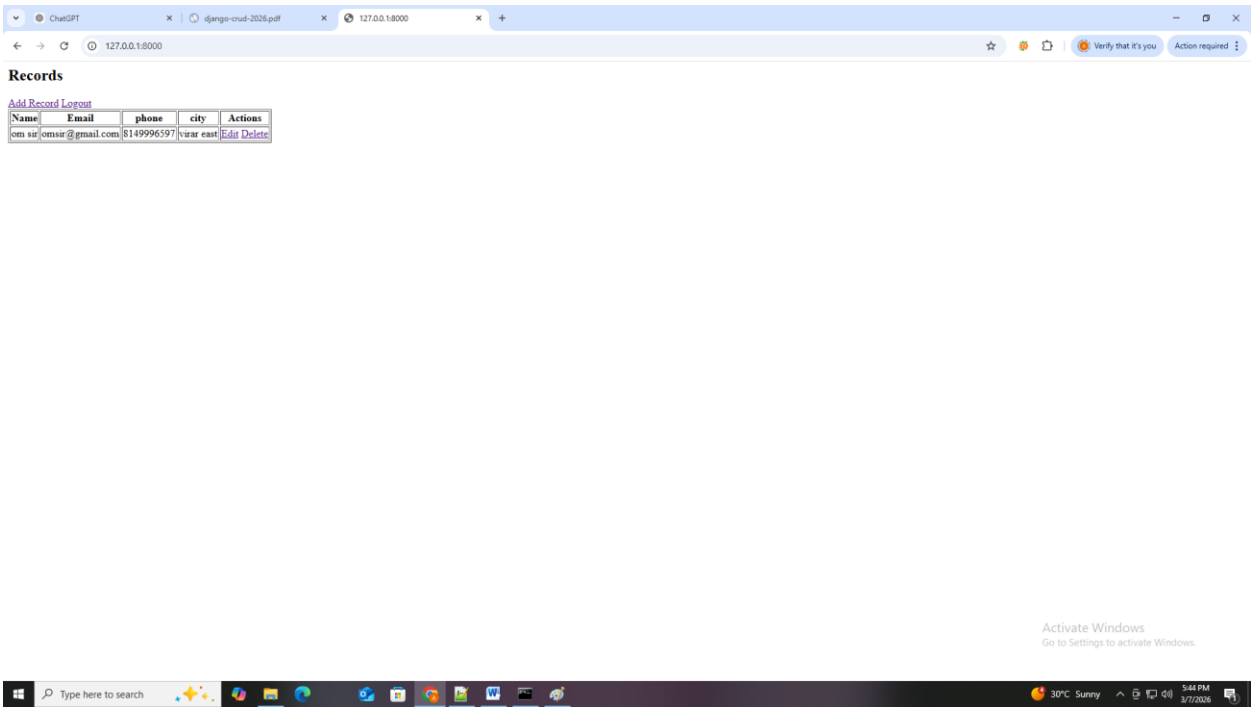
This is **exactly the type of project recruiters expect from beginners.**

Finally run :-

```
python manage.py runserver
```

http://127.0.0.1:8000




ChatGPT
django-crud-2026.pdf
127.0.0.1:8000
127.0.0.1:8000
Verify that it's you
Action required
Records
Add Record Logout
Name
Email
phone
city
Actions
om sir
omsir@gmail.com
8149996597
virar east
Edit
Delete
Activate Windows
Go to Settings to activate Windows.
Type here to search
30°C Sunny
5:44 PM
3/7/2026

If You want **Login and Logout handled via custom function-based views** instead of Django's class-based views. I'll rewrite your **views.py** and **urls.py** for a **function-based authentication system** that works with your CRUD app.

---

1 views.py (Function-Based Authentication + CRUD)

```
from django.shortcuts import render, redirect, get_object_or_404
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth import login, logout, authenticate
from django.contrib.auth.decorators import login_required
from .models import Record
from .forms import RecordForm


# ----------------------------
# AUTHENTICATION VIEWS
# ----------------------------

def signup_view(request):
    if request.method == "POST":
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)  # automatically login after signup
            return redirect('record_list')
    else:
        form = UserCreationForm()
    return render(request, 'registration/signup.html', {'form': form})


def login_view(request):
    if request.method == "POST":
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('record_list')
    else:
        form = AuthenticationForm()
    return render(request, 'registration/login.html', {'form': form})


def logout_view(request):
    if request.method == "POST":  # logout must be POST for security
        logout(request)
        return redirect('login')
    # optional: show confirmation page for logout
    return render(request, 'registration/logout.html')
```

```
# ----------------------------
# CRUD VIEWS
# ----------------------------

@login_required
def record_list(request):
    records = Record.objects.all()
    return render(request, 'records/record_list.html', {'records': records})


@login_required
def record_create(request):
    if request.method == "POST":
        form = RecordForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('record_list')
    else:
        form = RecordForm()
    return render(request, 'records/record_form.html', {'form': form})


@login_required
def record_update(request, pk):
    record = get_object_or_404(Record, pk=pk)
    if request.method == "POST":
        form = RecordForm(request.POST, instance=record)
        if form.is_valid():
            form.save()
            return redirect('record_list')
    else:
        form = RecordForm(instance=record)
    return render(request, 'records/record_form.html', {'form': form})


@login_required
def record_delete(request, pk):
    record = get_object_or_404(Record, pk=pk)
    if request.method == "POST":
        record.delete()
        return redirect('record_list')
    return render(request, 'records/record_confirm_delete.html', {'record':
record})
```

---

2 urls.py (Function-Based Authentication + CRUD)

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

urlpatterns = [
```

```
    # Authentication
    path('signup/', views.signup_view, name='signup'),
    path('login/', views.login_view, name='login'),
    path('logout/', views.logout_view, name='logout'),

    # CRUD
    path('', views.record_list, name='record_list'),
    path('create/', views.record_create, name='record_create'),
    path('update/<int:pk>/', views.record_update, name='record_update'),
    path('delete/<int:pk>/', views.record_delete, name='record_delete'),
]
```

---

3 Templates Changes

Logout Button (POST request)

```
<form method="post" action="{% url 'logout' %}">
    {% csrf_token %}
    <button type="submit">Logout</button>
</form>
```

Login & Signup forms remain the same

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

Key Points

1. **Logout must use POST** → prevents CSRF and HTTP 405 errors.
2. **login_view** uses `AuthenticationForm` and `authenticate()`.
3. `login_required` ensures CRUD pages are protected.
4. After signup or login, user is redirected to `record_list`.
5. URLs are simple and all function-based.

If You want **user-specific CRUD**, so that **each user can only edit/delete the records they added**. This is very common in real apps and highly valued in interviews.

Here's how to implement it step by step.

---

1 Update your models.py

Add a **ForeignKey to the user** for each record.

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

class Record(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField()
    phone = models.CharField(max_length=15)
    city = models.CharField(max_length=100)

    # NEW: track which user created this record
    user = models.ForeignKey(User, on_delete=models.CASCADE)

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

 Now each record belongs to a **specific user**.

---

2 Update forms.py (No changes needed for user field)

We **do NOT want users to select user manually**, so exclude `user`:

```
from django import forms
from .models import Record

class RecordForm(forms.ModelForm):
    class Meta:
        model = Record
        fields = ['name', 'email', 'phone', 'city']
```

---

3 Update views.py

3.1 Create

When saving, assign `request.user` to the record:

```
@login_required
def record_create(request):
    if request.method == "POST":
        form = RecordForm(request.POST)
        if form.is_valid():
            record = form.save(commit=False)
            record.user = request.user  # assign the current user
            record.save()
            return redirect('record_list')
    else:
        form = RecordForm()
    return render(request, 'records/record_form.html', {'form': form})
```

3.2 Read (List only the user's records)

```
@login_required
def record_list(request):
    records = Record.objects.filter(user=request.user)  # only show records
added by current user
    return render(request, 'records/record_list.html', {'records': records})
```

3.3 Update / Delete (Only allow if record belongs to user)

```
@login_required
def record_update(request, pk):
    record = get_object_or_404(Record, pk=pk, user=request.user)  # user-
specific
    if request.method == "POST":
        form = RecordForm(request.POST, instance=record)
        if form.is_valid():
            form.save()
            return redirect('record_list')
    else:
        form = RecordForm(instance=record)
    return render(request, 'records/record_form.html', {'form': form})


@login_required
def record_delete(request, pk):
    record = get_object_or_404(Record, pk=pk, user=request.user)  # user-
specific
    if request.method == "POST":
        record.delete()
        return redirect('record_list')
    return render(request, 'records/record_confirm_delete.html', {'record':
record})
```

---

4 Run Migrations

After updating models:

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

What This Does

1. Only the **user who created the record** can **edit/delete it**.
2. Users see **only their own records** in the list.
3. Security is enforced via `get_object_or_404(..., user=request.user)`.