Complete beginner-friendly Django project structure with **Signup, Login, Logout using Function-Based Views (FBV)** + templates + everything wired together.

□ 1. Project Structure

```
myproject/
│
├── myproject/
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
│
├── accounts/          ← your app
│   ├── migrations/
│   ├── __init__.py
│   ├── views.py
│   ├── urls.py
│   └── models.py
│
├── templates/
│   ├── signup.html
│   ├── login.html
│   └── home.html
│
├── db.sqlite3
└── manage.py
```

□ 2. Create Project & App (Commands)

```
django-admin startproject myproject
cd myproject
python manage.py startapp accounts
```

□ 3. settings.py (Important Changes)

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

# Add templates folder
import os
TEMPLATES = [
    {
        ...
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        ...
    },
]

# Redirect after login
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = 'home'
```

□ 4. Main URLs (myproject/urls.py)

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

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

□ 5. App URLs (accounts/urls.py)

```
from django.urls import path
from .views import signup_view, login_view, logout_view, home

urlpatterns = [
    path('', home, name='home'),
    path('signup/', signup_view, name='signup'),
    path('login/', login_view, name='login'),
    path('logout/', logout_view, name='logout'),
]
```

□□ 6. Views (accounts/views.py)

```
from django.shortcuts import render, redirect
from django.contrib.auth.models import User
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required


def signup_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        if User.objects.filter(username=username).exists():
            return render(request, 'signup.html', {'error': 'User already
exists'})

        user = User.objects.create_user(username=username, password=password)
        user.save()
        return redirect('login')

    return render(request, 'signup.html')


def login_view(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(request, username=username, password=password)

        if user is not None:
            login(request, user)
            return redirect('home')
        else:
            return render(request, 'login.html', {'error': 'Invalid
credentials'})

    return render(request, 'login.html')


def logout_view(request):
    logout(request)
    return redirect('login')


@login_required
def home(request):
    return render(request, 'home.html')
```

□ 7. Templates

□ signup.html

```
<h2>Signup</h2>

<form method="post">
    {% csrf_token %}
    <input type="text" name="username" placeholder="Username"
required><br><br>
    <input type="password" name="password" placeholder="Password"
required><br><br>
    <button type="submit">Signup</button>
</form>

<p style="color:red;">{{ error }}</p>

<a href="/login/">Login</a>
```

□ login.html

```
<h2>Login</h2>

<form method="post">
    {% csrf_token %}
    <input type="text" name="username" placeholder="Username"><br><br>
    <input type="password" name="password" placeholder="Password"><br><br>
    <button type="submit">Login</button>
</form>

<p style="color:red;">{{ error }}</p>

<a href="/signup/">Signup</a>
```

□ home.html

```
<h2>Welcome {{ request.user.username }}</h2>

<a href="/logout/">Logout</a>
```

□ 8. Run Migrations

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

▶ 9. Run Server

```
python manage.py runserver
```

 Open:

- http://127.0.0.1:8000/signup/
- http://127.0.0.1:8000/login/

Super Easy Revision Trick (Interview)

 Flow:

```
Signup → create_user
Login → authenticate → login
Logout → logout
```

 Security keyword: `csrf_token` (VERY IMPORTANT)

 Protection: `@login_required`

Let's break **each line in simple words** so you can *remember + explain easily*.

□ IMPORTS (Top Lines)

```
from django.shortcuts import render, redirect
```

- `render` → used to **show HTML page**
- `redirect` → used to **go to another URL/page**

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

- `User` → Django's **built-in user table/model**
- Used to **create, store, and manage users**

```
from django.contrib.auth import authenticate, login, logout
```

- `authenticate()` → **checks username & password**
- `login()` → **logs user in (creates session)**
- `logout()` → **logs user out (clears session)**

```
from django.contrib.auth.decorators import login_required
```

- `login_required` → **restricts access**
- Only logged-in users can open that view

□ □ SIGNUP VIEW

```
def signup_view(request):
```

- Function that handles **user signup**
- `request` → contains all data (GET, POST, user info)

```
if request.method == 'POST':
```

- Check if form is submitted
- `POST` means user sent data (form submit)

```
username = request.POST['username']
password = request.POST['password']
```

- Get data from form
- `request.POST` → dictionary of form values

```
if User.objects.filter(username=username).exists():
```

- Check if username already exists in DB
- Prevent duplicate users

```
return render(request, 'signup.html', {'error': 'User already exists'})
```

- Show signup page again
- Send error message to HTML

```
user = User.objects.create_user(username=username, password=password)
```

- Create new user
- Automatically hashes password (important!)

```
user.save()
```

- Save user to database

```
return redirect('login')
```

- After signup → go to login page

```
return render(request, 'signup.html')
```

- If request is GET → just show signup form

□ LOGIN VIEW

```
def login_view(request):
```

- Function to handle login

```
if request.method == 'POST':
```

- Check if form submitted

```
username = request.POST['username']
password = request.POST['password']
```

- Get user input

```
user = authenticate(request, username=username, password=password)
```

- Check credentials with database
- Returns:
  - user object → if correct
  - None → if wrong

```
if user is not None:
```

- If login successful

```
login(request, user)
```

- Logs user in
- Creates session (user stays logged in)

```
return redirect('home')
```

- Go to home page after login

```
else:
    return render(request, 'login.html', {'error': 'Invalid credentials'})
```

- If login fails → show error

```
return render(request, 'login.html')
```

- If GET request → show login page

□ LOGOUT VIEW

```
def logout_view(request):
```

- Function for logout

```
logout(request)
```

- Clears session
- Logs user out

```
return redirect('login')
```

- Redirect to login page

□ HOME VIEW (PROTECTED)

```
@login_required
```

- Only logged-in users can access
- If not logged in → redirect to login page

```
def home(request):
```

- Home page function

```
return render(request, 'home.html')
```

- Show home page

□ FINAL INTERVIEW SUMMARY (SUPER IMPORTANT)

□ In 1 line each:

- **render** → show HTML
- **redirect** → move to another page
- **authenticate** → check credentials
- **login** → create session
- **logout** → destroy session
- **login_required** → protect page

□ Quick Memory Trick

```
Signup → create_user
Login → authenticate → login
Logout → logout
```

Let’s create a **complete beginner-friendly Django authentication flow** using **UserCreationForm + AuthenticationForm** with **Signup, Login, Logout, protected Home page**, all ready for interviews. I’ll explain clearly so you can remember easily.

---

1. Project Structure (Simplified)

```
myproject/
├── myproject/
│   └── settings.py, urls.py, wsgi.py
├── accounts/
│   ├── views.py
│   ├── urls.py
│   └── forms.py
├── templates/
│   ├── signup.html
│   ├── login.html
│   └── home.html
└── manage.py
```

---

2. forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, AuthenticationForm
from django.contrib.auth.models import User

# Signup form with username, password, and password confirmation
class SignupForm(UserCreationForm):
    class Meta:
        model = User
        fields = ['username', 'password1', 'password2']

# We can use Django's built-in AuthenticationForm for login
```

□□ 3. views.py

```
from django.shortcuts import render, redirect
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth import login, logout
from django.contrib.auth.decorators import login_required
from .forms import SignupForm

# Signup View
def signup_view(request):
    if request.method == 'POST':
        form = SignupForm(request.POST)
        if form.is_valid():
            form.save()  # Automatically hashes password
            return redirect('login')
        return render(request, 'signup.html', {'form': form})
    else:
        form = SignupForm()
    return render(request, 'signup.html', {'form': form})

# Login View
def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            user = form.get_user()  # get the user object from form
            login(request, user)    # logs the user in
            return redirect('home')
        return render(request, 'login.html', {'form': form})
    else:
        form = AuthenticationForm()
    return render(request, 'login.html', {'form': form})

# Logout View
def logout_view(request):
    logout(request)           # clears session
    return redirect('login')  # redirect to login page

# Protected Home Page
@login_required
def home(request):
    # request.user now contains the logged-in user
    return render(request, 'home.html', {'user': request.user})
```

---

4. urls.py (accounts/urls.py)

```
from django.urls import path
from .views import signup_view, login_view, logout_view, home

urlpatterns = [
    path('', home, name='home'),
    path('signup/', signup_view, name='signup'),
    path('login/', login_view, name='login'),
    path('logout/', logout_view, name='logout'),
]
```

And include it in my**project urls.py**:

```
from django.urls import path, include

urlpatterns = [
    path('', include('accounts.urls')),
]
```

---

5. Templates

signup.html

```
<h2>Signup</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Signup</button>
</form>
<a href="/login/">Already have an account? Login</a>
```

---

login.html

```
<h2>Login</h2>
<form method="post">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Login</button>
</form>
<a href="/signup/">Don't have an account? Signup</a>
```

---

home.html

```
<h2>Welcome, {{ user.username }}</h2>
<a href="/logout/">Logout</a>
```

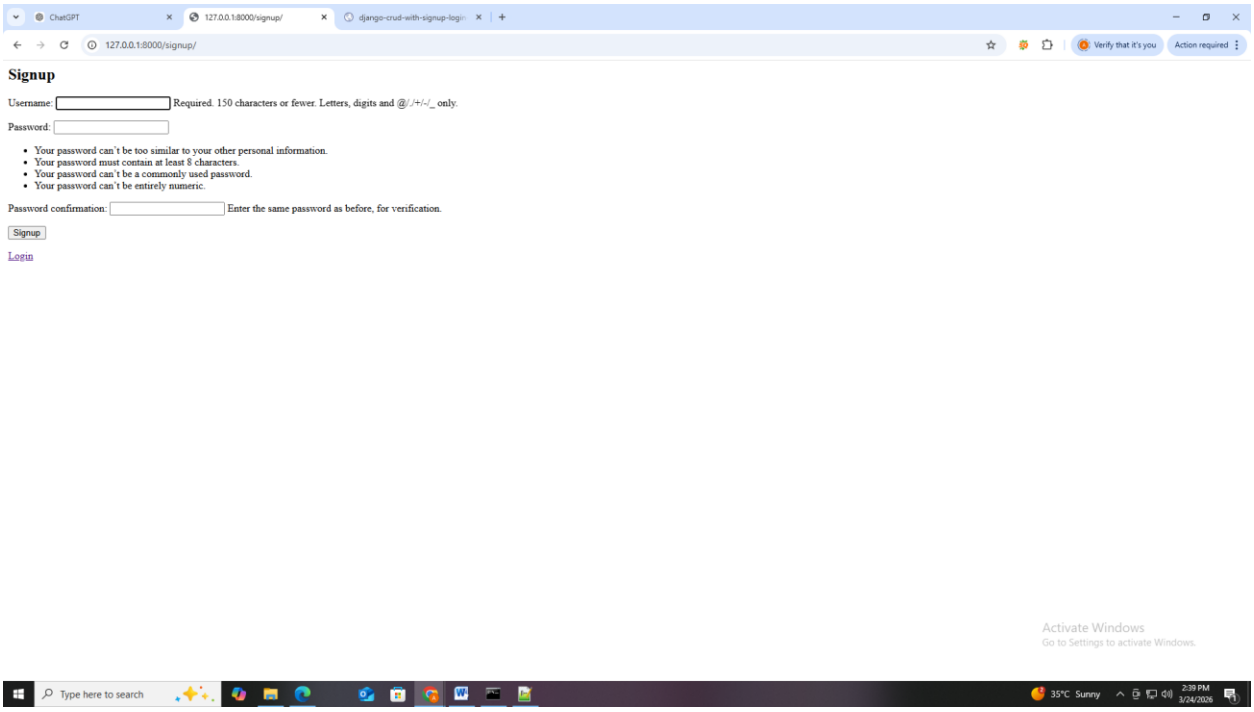
□ 6. Key Notes for Interviews

- **Signup** → `UserCreationForm` → handles password hash + confirmation □
- **Login** → `AuthenticationForm` → handles validation □
- **Logout** → `logout(request)` clears session □
- **Protected pages** → `@login_required` ensures only logged-in users can access □
- **`request.user`** → gives the currently logged-in user

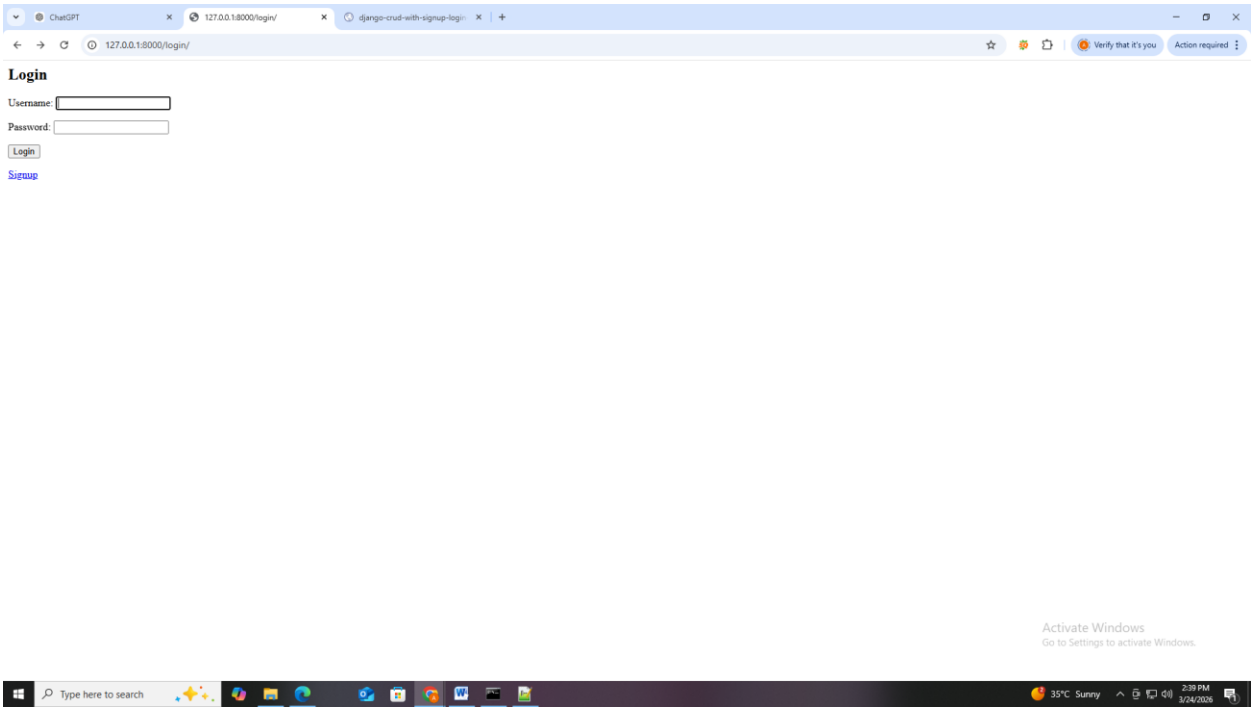
□ Flow in Simple Words

```
Signup → form.save() → password hashed → redirect login
Login → form.is_valid() → form.get_user() → login(request, user) → redirect
home
Home → login_required → request.user.username
Logout → logout(request) → redirect login
```

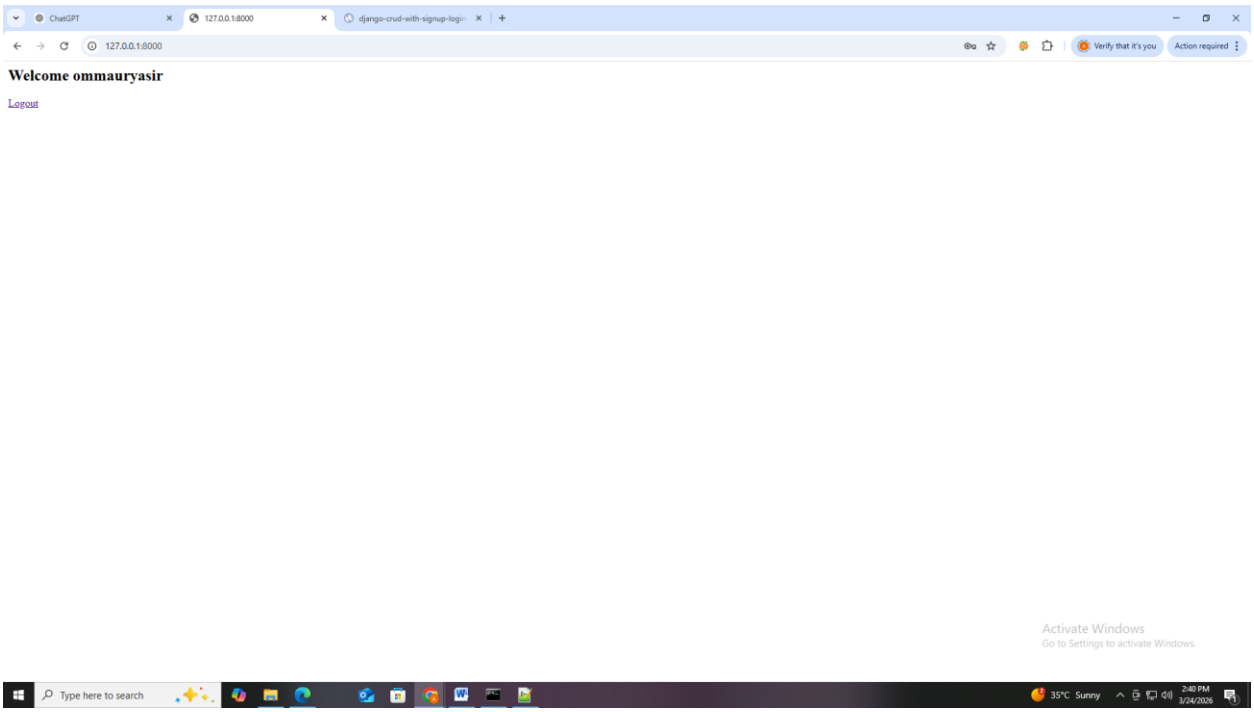
Signup:-



Login :-



After login :-



Let's go **line by line** for your code so you can explain it clearly in interviews. I'll break it into sections: imports, signup, login, logout, and home.

□ IMPORTS

```
from django.shortcuts import render, redirect
```

- **render** → displays an HTML template to the user.
- **redirect** → sends the user to a different URL/page.

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

- **User** → Django's built-in user model. Used to create or query users in the database.

```
from django.contrib.auth import login, logout
```

- **login()** → logs a user in and creates a session.
- **logout()** → logs a user out and clears the session.

```
from django.contrib.auth.forms import AuthenticationForm
```

- **AuthenticationForm** → Django's built-in form for login, handles validation of username/password.

```
from django.contrib.auth.decorators import login_required
```

- **login_required** → decorator to protect views, only allows logged-in users to access.

```
from .forms import SignupForm
```

- Imports your custom **SignupForm**, which is based on `UserCreationForm` (handles password hashing and validation).

□□ SIGNUP VIEW

```
def signup_view(request):
```

- Defines the signup page logic. Takes **request** as input (all form data is in request).

```
if request.method == 'POST':
```

- Checks if the user submitted the form (POST request).

```
form = SignupForm(request.POST)
```

- Creates a form instance filled with submitted data.

```
if form.is_valid():
```

- Validates the form: checks required fields, password match, username uniqueness, etc.

```
form.save()   # automatically hashes password □
```

- Saves the user to the database.
- **Automatically hashes password** (so it's secure).

```
return redirect('login')
```

- After successful signup → redirect the user to the login page.

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

- If form is invalid → re-display the signup page with the form (and errors).

```
else:
    form = SignupForm()
```

- If GET request → display an empty signup form.

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

- Render the signup page with empty form (GET request).

□ LOGIN VIEW

```
def login_view(request):
```

- Defines the login page logic.

```
if request.method == 'POST':
    form = AuthenticationForm(request, data=request.POST)
```

- If form submitted → create AuthenticationForm with submitted data and request.

```
if form.is_valid():
```

- Validates username and password automatically.

```
user = form.get_user()   # get logged-in user
```

- Retrieves the authenticated user object.

```
login(request, user)     # create session
```

- Logs the user in by creating a session.

```
return redirect('home')
```

- After login → redirect to home page.

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

- If form invalid → show login page again with errors.

```
else:
    form = AuthenticationForm()
```

- If GET request → show empty login form.

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

- Render login page (empty form for GET request).

□ LOGOUT VIEW

```
def logout_view(request):
    logout(request)        # clears session (logs user out)
    return redirect('login')
```

- **logout(request)** → destroys the session, logs out the user.
- Redirects to login page after logout.

□ HOME VIEW

```
@login_required
def home(request):
    return render(request, 'home.html')
```

- `@login_required` → only logged-in users can access this page.
- `render(request, 'home.html')` → shows home page template.
- `request.user` can be used inside template to show username (e.g., `{{ user.username }}`).

□ INTERVIEW MEMORY TRICKS

- **Signup:** `form.is_valid() → form.save() → redirect('login')`
- **Login:** `AuthenticationForm → is_valid → form.get_user() → login(request, user)`
- **Logout:** `logout(request) → redirect('login')`
- **Protected page:** `@login_required → render(...)`

 Very clean, ready to explain in interviews.