Login , signup , logout code explanation :-

```python
# views.py
from django.shortcuts import render, redirect
from django.contrib.auth import authenticate, login, logout
from django.contrib import messages
from .forms import SignupForm
from django.contrib.auth.forms import AuthenticationForm
from django.contrib.auth.decorators import login_required

def signup_view(request):
    if request.method == 'POST':
        form = SignupForm(request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, 'Account created successfully!')
            return redirect('login')
    else:
        form = SignupForm()
    return render(request, 'signup.html', {'form': form})

def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(request, data=request.POST)
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
            if user is not None:
                login(request, user)
                return redirect('home')
            else:
                messages.error(request, 'Invalid username or password.')
        else:
            messages.error(request, 'Invalid form.')
    else:
        form = AuthenticationForm()
    return render(request, 'login.html', {'form': form})

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

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

Explanation given below:-

forms.py file code-

🔍 Full Code:

python Copy Edit

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

class SignupForm(UserCreationForm):
    email = forms.EmailField(required=True)

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

📁 Imports Breakdown

```python
from django import forms
```

- Imports Django's `forms` module. It's used to define and manage form fields in Django (like `CharField`, `EmailField`, etc).

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

- Imports the built-in `User` model. This model represents users in Django's authentication system.

```python
from django.contrib.auth.forms import UserCreationForm
```

- Imports the default Django form for creating new users. This includes:
  - `username`
  - `password1`
  - `password2` (for confirmation)

`SignupForm` Class

```python
class SignupForm(UserCreationForm):
```

- We're creating a **custom form** by extending the default `UserCreationForm`.
- Inheriting from `UserCreationForm` allows us to reuse its built-in validation and fields.

Adding Email Field

```python
    email = forms.EmailField(required=True)
```

- Adds an extra `email` field to the form (not included in the default `UserCreationForm`).
- `required=True`: This makes sure the user must provide a valid email to register.

`Meta` Class

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

- The `Meta` class tells Django **which model the form is related to**.
- `model = User`: This form is linked to the built-in `User` model.
- `fields = [...]`: Specifies the fields that should appear on the form and in what order.

💡 Why customize `UserCreationForm`?

Django's default `UserCreationForm` doesn't include an email field, even though the `User` model has one. So, this custom `SignupForm`:

1. Adds the `email` field.
2. Ensures it's required.
3. Still uses Django's built-in validation (like password matching and username uniqueness).

views.py file code:-

🔁 Full Code for Reference:

python Copy Edit

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

🔹 Imports

`render` and `redirect`

python Copy Edit

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

- `render(request, template, context)`: Loads an HTML template and passes data (context).
- `redirect('url_name')`: Redirects the user to another URL, usually after an action (e.g. login or logout).

Auth Tools

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

- `authenticate()`: Checks if username and password are correct.
- `login()`: Logs the user in (starts session).
- `logout()`: Logs the user out (clears session).

Messages

```python
from django.contrib import messages
```

- Django's messaging framework.
- Allows you to show success or error messages (like "Login failed").

Forms

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

- `SignupForm`: Your custom form from `forms.py`.
- `AuthenticationForm`: Django's built-in form for login.

Login Required Decorator

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

- Protects views from being accessed by anonymous users.
- If a user is not logged in, they will be redirected to the login page.

🔸 View 1: `signup_view`

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

- Handles user registration.

If form is submitted (POST):

```python
    if request.method == 'POST':
        form = SignupForm(request.POST)
```

- Creates a `SignupForm` with data from the request (what user filled in).

- Creates a `SignupForm` with data from the request (what user filled in).

```python
        if form.is_valid():
            form.save()
            messages.success(request, 'Account created successfully!')
            return redirect('login')
```

- If the form is valid:
  - Save the user to the database.
  - Show a success message.
  - Redirect to the login page.

If it's a GET request (user just opened the page):

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

- Show a blank signup form.

Render the signup page:

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

- Loads the signup template and passes the form to it.

◆ View 2: `login_view`

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

- Handles user login.

If it's a POST request (form submitted):

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

- Creates an instance of Django's login form with posted data.

If login form is valid:

```python
        if form.is_valid():
            username = form.cleaned_data['username']
            password = form.cleaned_data['password']
            user = authenticate(request, username=username, password=password)
```

- Extracts the username and password.
- Authenticates the user (verifies credentials).

If user is valid:

```python
            if user is not None:
                login(request, user)
                return redirect('home')
```

- Logs the user in.
- Redirects to the home page.

If authentication fails:

```python
            else:
                messages.error(request, 'Invalid username or password.')
```

If form is invalid:

```python
        else:
            messages.error(request, 'Invalid form.')
```

If it's a GET request (just opening login page):

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

- Shows a blank login form.

Render login page:

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

🔸 View 3: `logout_view`

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

- Logs out the current user.
- Redirects them to the login page.

🔸 View 4: `home`

python Copy Edit

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

- This view is only accessible if the user is logged in.
- If not logged in, Django will redirect them to the login page.
- If logged in, it renders `home.html`.