**Role based Registration in Django 5**

First If we want customer only access customer page and seller only access seller page so update this In

Base.html:

            {% if user.is\_authenticated %}

                {% if user.is\_seller %}

                <li><a href="{% url 'seller\_dashboard' %}" class="block py-2 px-3">Seller Dashboard</a></li>

              {% elif user.is\_customer %}

                <li><a href="{% url 'customer\_dashboard' %}" class="block py-2 px-3">Customer Dashboard</a></li>

              {% endif %}

IN THIS WE CAN GAVE THE POWER TO THE USER TO MAKE THE ACCOUNT AS A SELLER OR AS A USER BOTH

Add this code in

accounts/forms.py:

    ROLE\_CHOICES = (

        ('customer','Customer'),

        ('seller','Seller'),

    )

    role = forms.ChoiceField(choices=ROLE\_CHOICES,widget=forms.Select)

complete updated accounts/forms.py:

from django import forms

from account.models import User

class RegistrationForm(forms.ModelForm):

    ROLE\_CHOICES = (

        ('customer','Customer'),

        ('seller','Seller'),

    )

    role = forms.ChoiceField(choices=ROLE\_CHOICES,widget=forms.Select)

    """

    Simple registration form:

    - email, name from the model

    - password + confirm\_password are plain fields here; we hash password in the view

    """

    password = forms.CharField(widget=forms.PasswordInput)

    confirm\_password = forms.CharField(widget=forms.PasswordInput)

    class Meta:

        model = User

        fields = ["email", "name", "password", "confirm\_password"]

    # validate that password and confirm\_password match

    def clean(self):

        cleaned\_data = super().clean()

        password = cleaned\_data.get('password')

        confirm = cleaned\_data.get('confirm\_password')

        if password and confirm and password != confirm:

            self.add\_error('confirm\_password', 'Password and Confirm Password do not match.')

        return cleaned\_data

    # ensure email uniqueness at the form level

    def clean\_email(self):

        email = self.cleaned\_data.get('email')

        if email and User.objects.filter(email=email).exists():

            raise forms.ValidationError("A user with this email already exists.")

        return email

class PasswordResetForm(forms.Form):

    """

    Minimal 'forgot password' form that only requires an email.

    """

    email = forms.EmailField(

        max\_length=255,

        required=True,

        widget=forms.EmailInput(attrs={'placeholder': 'you@example.com'})

    )

    def clean\_email(self):

        """

        Ensure we only proceed if a user with this email actually exists.

        """

        email = self.cleaned\_data.get('email')

        if not User.objects.filter(email=email).exists():

            raise forms.ValidationError('No account is associated with this email address.')

        return email

AND add this in account/templates/account/register.html:

      <!-- role -->

      <div>

        <label for="role" class="block text-sm font-medium">Role</label>

        <select name="role" id="role" class="block w-full mt-1 border-gray-300 rounded-md shadow-sm bg-gray-700 p-2"

          required>

          <option value="customer">Customer</option>

          <option value="seller">Seller</option>

        </select>

      </div>

      <!-- Submit -->

      <button type="submit"

        class="w-full px-4 py-2 text-white bg-rose-600 rounded-md hover:bg-rose-700 focus:outline-none">Register</button>

    </form>