**Video54 codefile:45**

**Set User Permission Programmatically in Django 5**

Import **PermissionsMixin** in

Accounts/models.py:

from django.contrib.auth.models import BaseUserManager, AbstractBaseUser,PermissionsMixin

and than add in this

class User(AbstractBaseUser,PermissionsMixin):

and than go into the account/admin.py:

add in this

"groups", "user\_permissions"

    fieldsets = [

        ("User Credentials", {"fields": ["email", "password"]}),

        ("Personal Information", {"fields": ["name", "city"]}),

        ("Permissions", {"fields": [

         "is\_active", "is\_staff", "is\_superuser", "is\_customer", "is\_seller", "groups", "user\_permissions"]})

    ]

After that do makemigrations and migrate

Than make new app name (product)

Product/models.py

from django.db import models

class Product(models.Model):

    name = models.CharField(max\_length=255)

    description = models.TextField()

    price = models.DecimalField(max\_digits=10, decimal\_places=2)

    created\_at = models.DateTimeField(auto\_now\_add=True)

    updated\_at = models.DateTimeField(auto\_now=True)

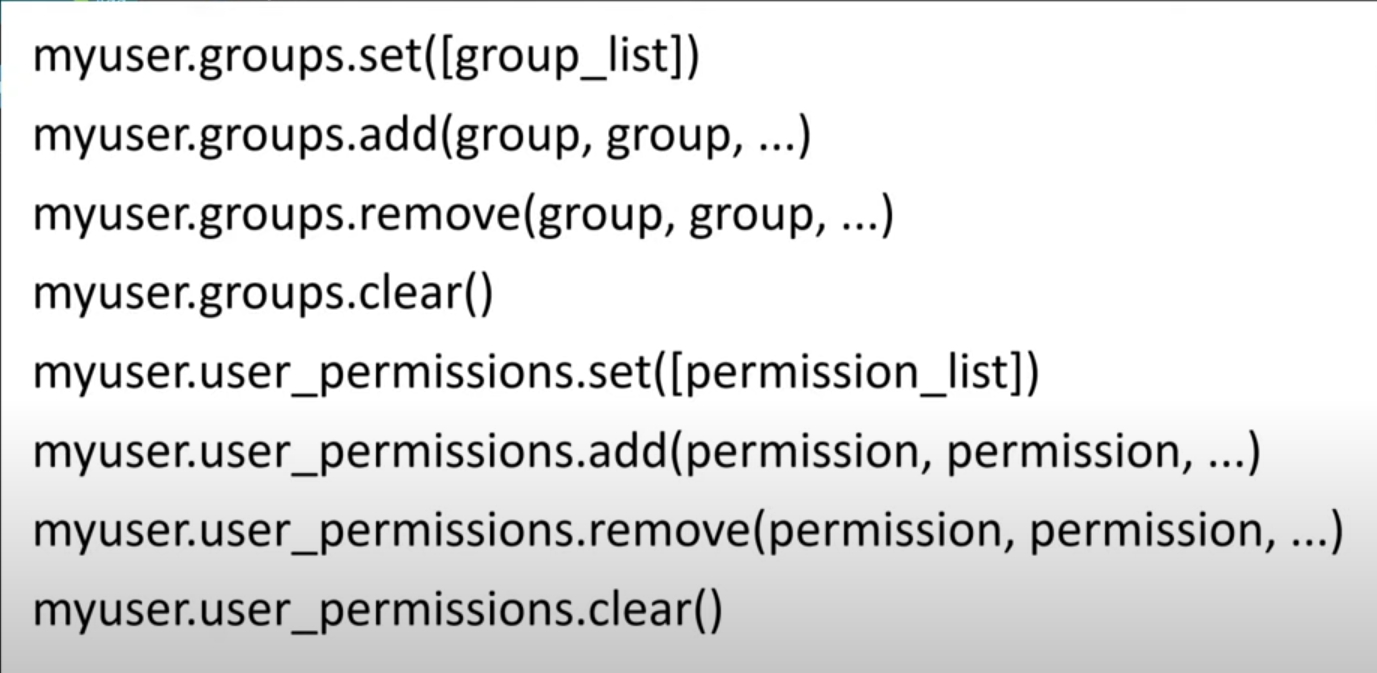
    def \_\_str\_\_(self):

        return self.name

In that we can modified the things

Like add delete view edit through code

Through group:



Go into core app we can working in that we can also work inside the account but core is best so

In core make file name sas permission\_config.py:

from product.models import Product

PERMISSION\_CONFIG = {

    "customer": {

        Product:["view"],

        #here if we have order so order we can also work with that order[]

    },

    "seller":{

        Product:["view","add","change",],

    }

#by this we can gave permissions to seller (view,add,change )

}

And make another file name utils.py:

Core/utils.py:

from django.contrib.auth.models import Permission

from django.contrib.contenttypes.models import ContentType

from core.permission\_config import PERMISSION\_CONFIG

def assign\_permission(user, role):

  role\_permission = PERMISSION\_CONFIG.get(role, {})

  for model, permissions in role\_permission.items():

    content\_type = ContentType.objects.get\_for\_model(model)

    for perm\_codename in permissions:

      permission = Permission.objects.get(

        content\_type=content\_type,

        codename=f"{perm\_codename}\_{model.\_meta.model\_name}"

        )

      user.user\_permissions.add(permission)

this function can give the permission

now next if we want these permssions can assign on registration(means user register and permissions automatically assigns) so go into the account views.py:

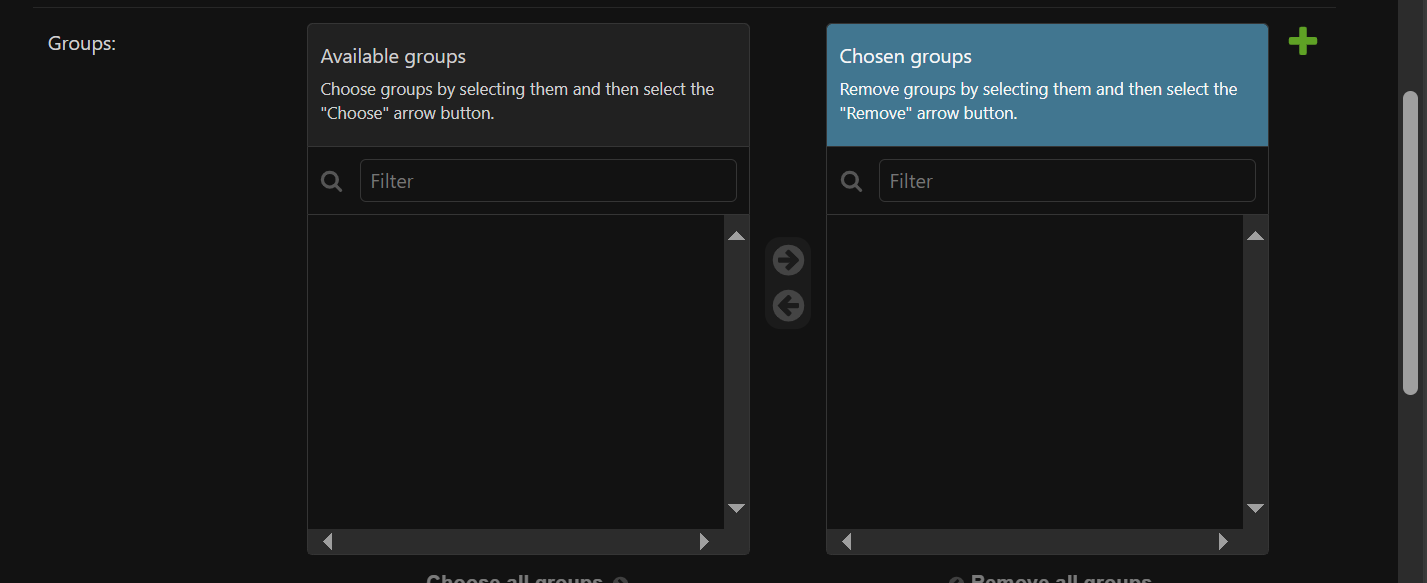
from core.utils import assign\_permission

and than go into the register function and after saving write code ;

def register\_view(request):

            user.save()

            assign\_permission(user,role)

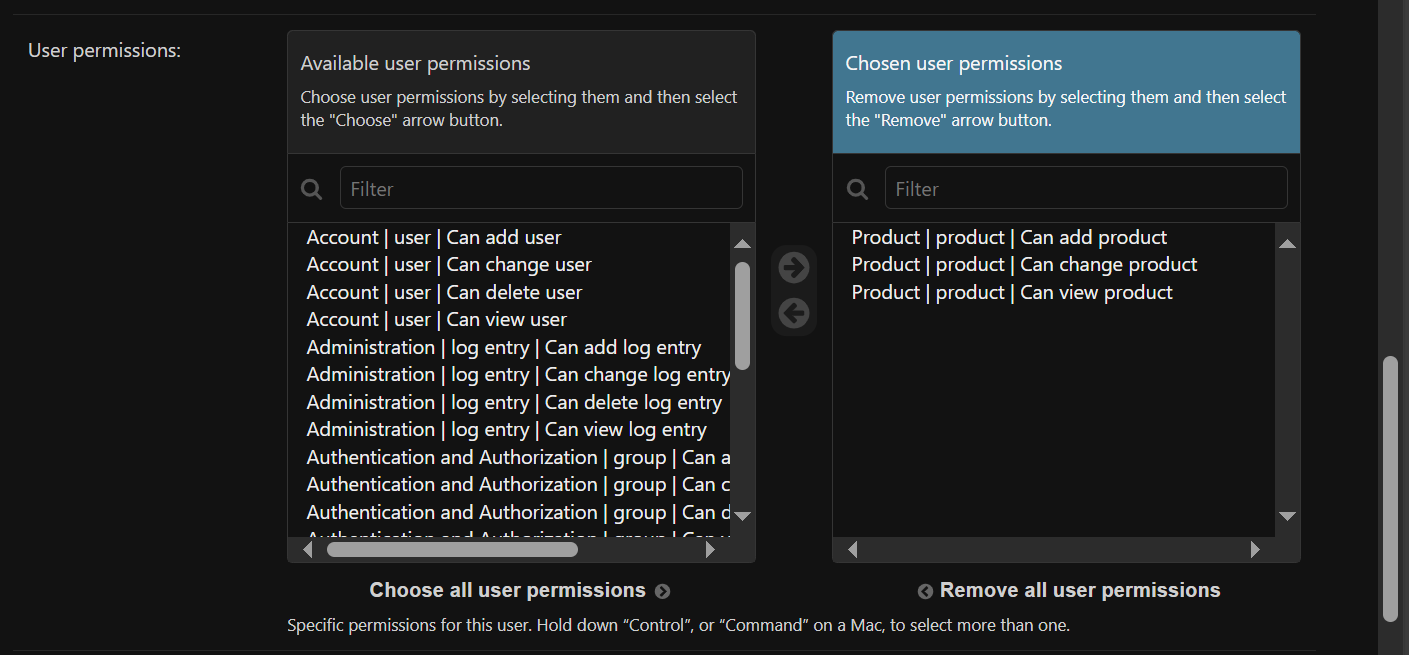
after that in permisisons (chhosen group cannot be select so )the choosen group cannot be shown so for that we go into the

account/admin.py:

    filter\_horizontal = ["groups"]

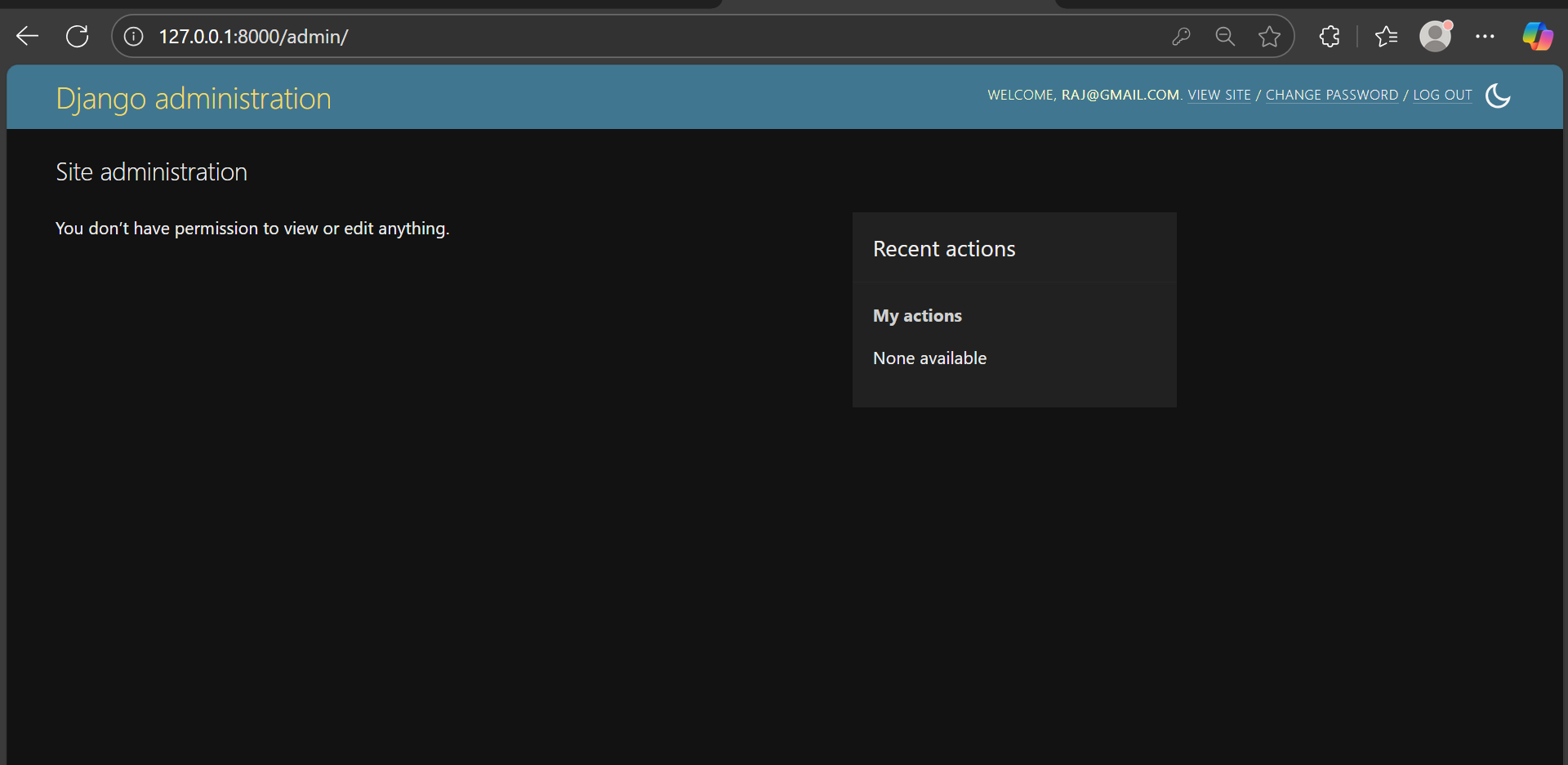
and also for permissions :

    filter\_horizontal = ["groups","user\_permissions"]



Now after that we have problem

A non super user cannot access the product



So this is good for security but we want to modfiy this so

We go account/models.py:

    # Permissions: keep it simple for this project (superuser = all perms)

    def has\_perm(self, perm, obj=None):

        if self.is\_superuser:

            return True

        return super().has\_perm(perm,obj) # non user can asscess data accourding to permissions

and aslo do modified the has\_module\_perms:

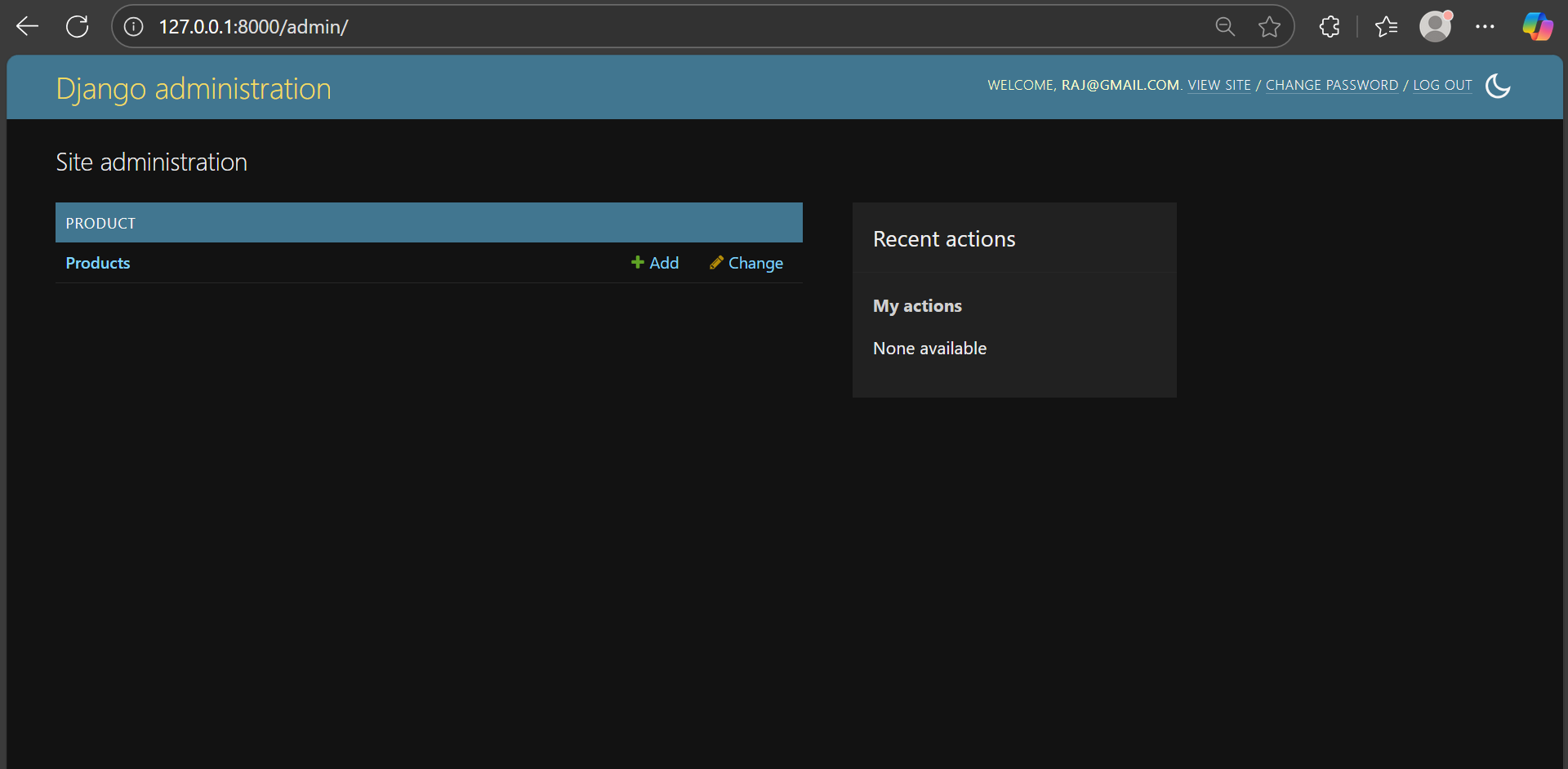
    def has\_module\_perms(self, app\_label):

        if self.is\_superuser:

            return True

        return super().has\_module\_perms(app\_label)# also do for the permissions

after that permissions are shows



Now here seller can add and change the product and customer can view the product

All the upper work is for admin now here we work for our own template

And their view

Product/views.py

Int this we can use decorators for restrict seller and customer for crud operation use only seller

from django.contrib.auth.decorators import permission\_required

in template we add these :

  {% if perms.product.change\_product  %}

  <a href="{% url 'product\_edit' product.pk %}" class="bg-orange-500 px-4 py-2">Edit</a>

  {% endif %}

  {% if perms.product.delete\_product  %}

  <a href="{% url 'product\_delete' product.pk %}" class="bg-red-500 px-4 py-2">Delete</a>

  {% endif %}

We can use this in other tempaltes accourding to our logic to handle things

Product/views.py

from django.shortcuts import render, get\_object\_or\_404, redirect

from django.contrib.auth.decorators import permission\_required

from product.models import Product

from product.forms import ProductForm

@permission\_required('product.view\_product', raise\_exception=True)

def product\_list(request):

    products = Product.objects.all()

    return render(request, 'product/list.html', {'products': products})

@permission\_required('product.view\_product', raise\_exception=True)

def product\_detail(request, pk):

    product = get\_object\_or\_404(Product, pk=pk)

    return render(request, 'product/detail.html', {'product': product})

@permission\_required('product.add\_product', raise\_exception=True)

def product\_add(request):

    if request.method == 'POST':

        form = ProductForm(request.POST)

        if form.is\_valid():

            form.save()

            return redirect('product\_list')

    else:

        form = ProductForm()

    return render(request, 'product/add.html', {'form': form})

@permission\_required('product.change\_product', raise\_exception=True)

def product\_edit(request, pk):

    product = get\_object\_or\_404(Product, pk=pk)

    if request.method == 'POST':

        form = ProductForm(request.POST, instance=product)

        if form.is\_valid():

            form.save()

            return redirect('product\_list')

    else:

        form = ProductForm(instance=product)

    return render(request, 'product/add.html', {'form': form})

@permission\_required('product.delete\_product', raise\_exception=True)

def product\_delete(request, pk):

    product = get\_object\_or\_404(Product, pk=pk)

    if request.method == 'POST':

        product.delete()

        return redirect('product\_list')

    return render(request, 'product/delete.html', {'product': product})

product/urls.py:

from django.urls import path

from . import views

urlpatterns = [

    path('list/', views.product\_list, name='product\_list'),

    path('<int:pk>/', views.product\_detail, name='product\_detail'),

    path('add/', views.product\_add, name='product\_add'),

    path('<int:pk>/edit/', views.product\_edit, name='product\_edit'),

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

]

Now other codes see templates and urls.py and other things in product app