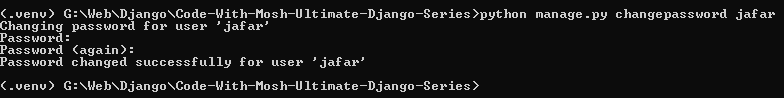
To Change The Admin Password, We Can Run: *python manage.py changepassword user-name*

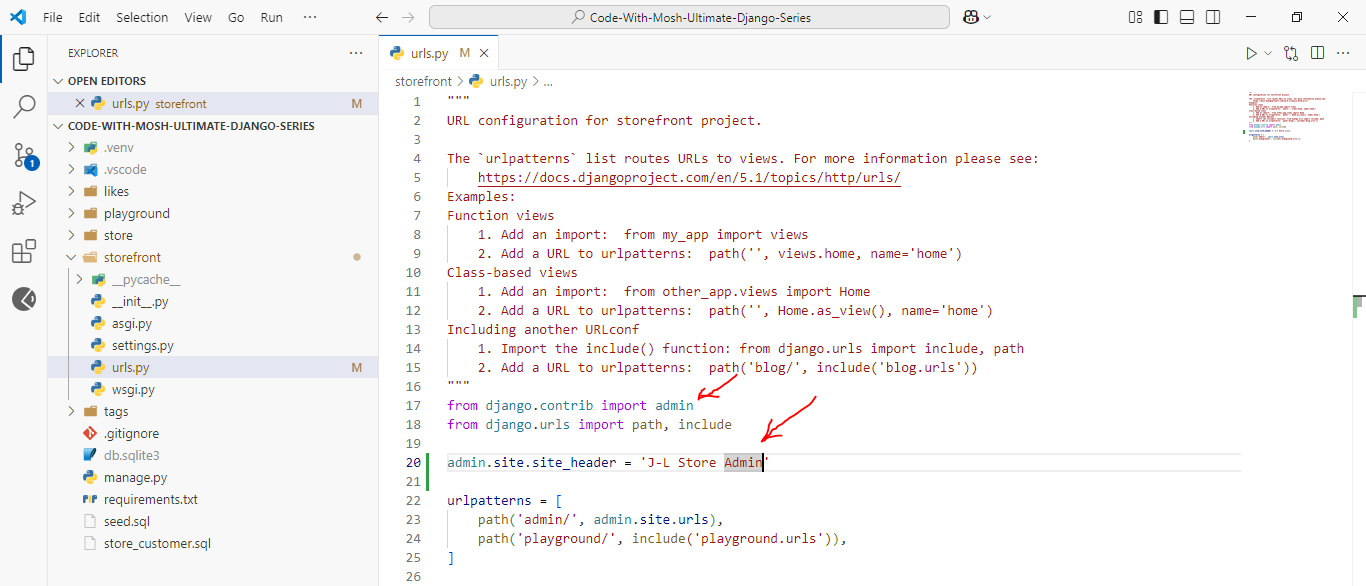
* **Ex**: *python manage.py changepassword jafar*

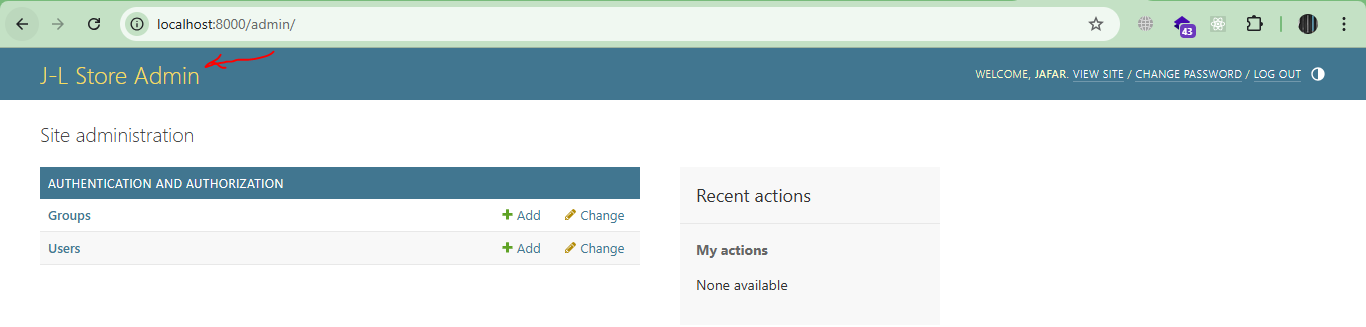


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Change The Header Of The Admin Site:

* From The Main urls.py-File, That Include The urls For Admin Site:
* Then We Change: *admin.site.site\_header = 'New Header Test Will be Here'*
  + Ex*: admin.site.site\_header = 'J-L Store Admin'*

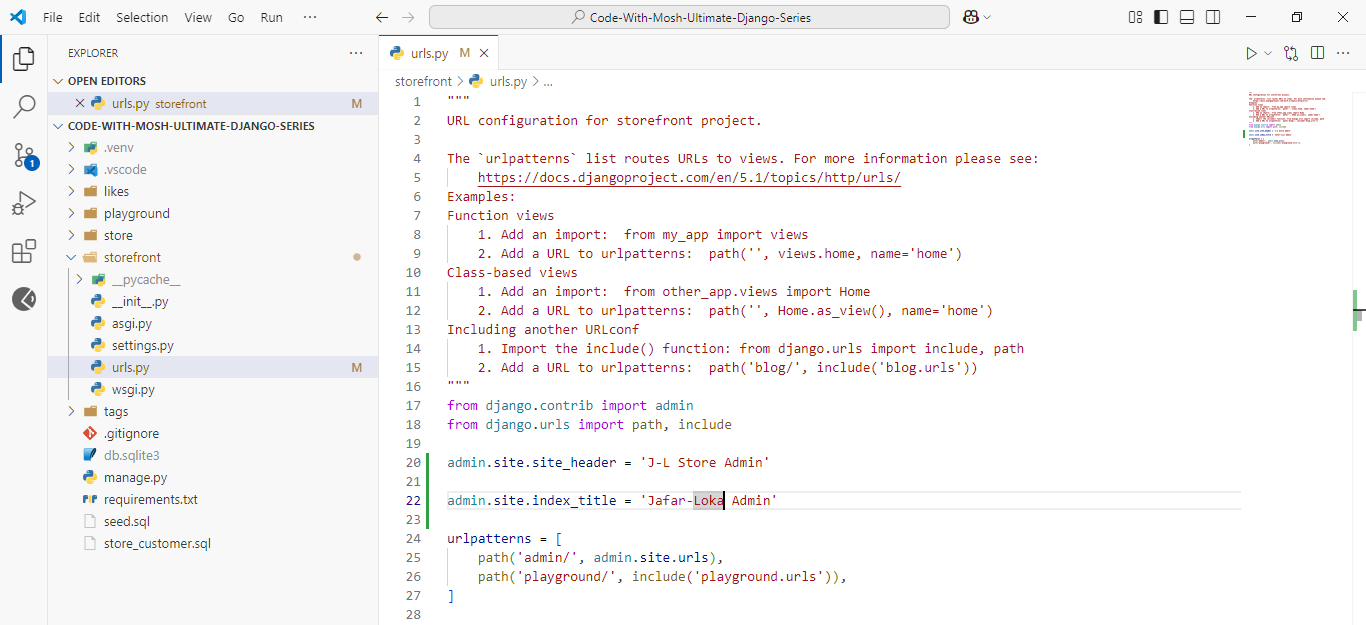


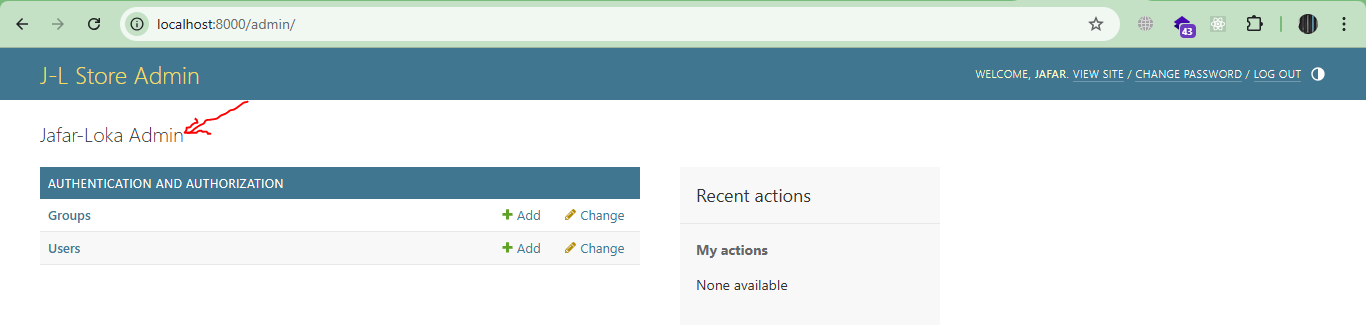


\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Change The Index Title, From The Main App Of Project (storefront):

* We Do: admin.site.index\_title = 'Text Here'
  + Ex: *admin.site.index\_title = 'Jafar-Loka Admin'*





\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Register Models To Our Admin Site:

* Every App Has admin.py-File
* Inside It We Register The Models, Using: admin.site.register(Model-Here)

from django.contrib import admin

from . import models

# Register your models here.

admin.site.register(models.Collection)

admin.site.register(models.Product)

admin.site.register(models.Order)

admin.site.register(models.OrderItem)

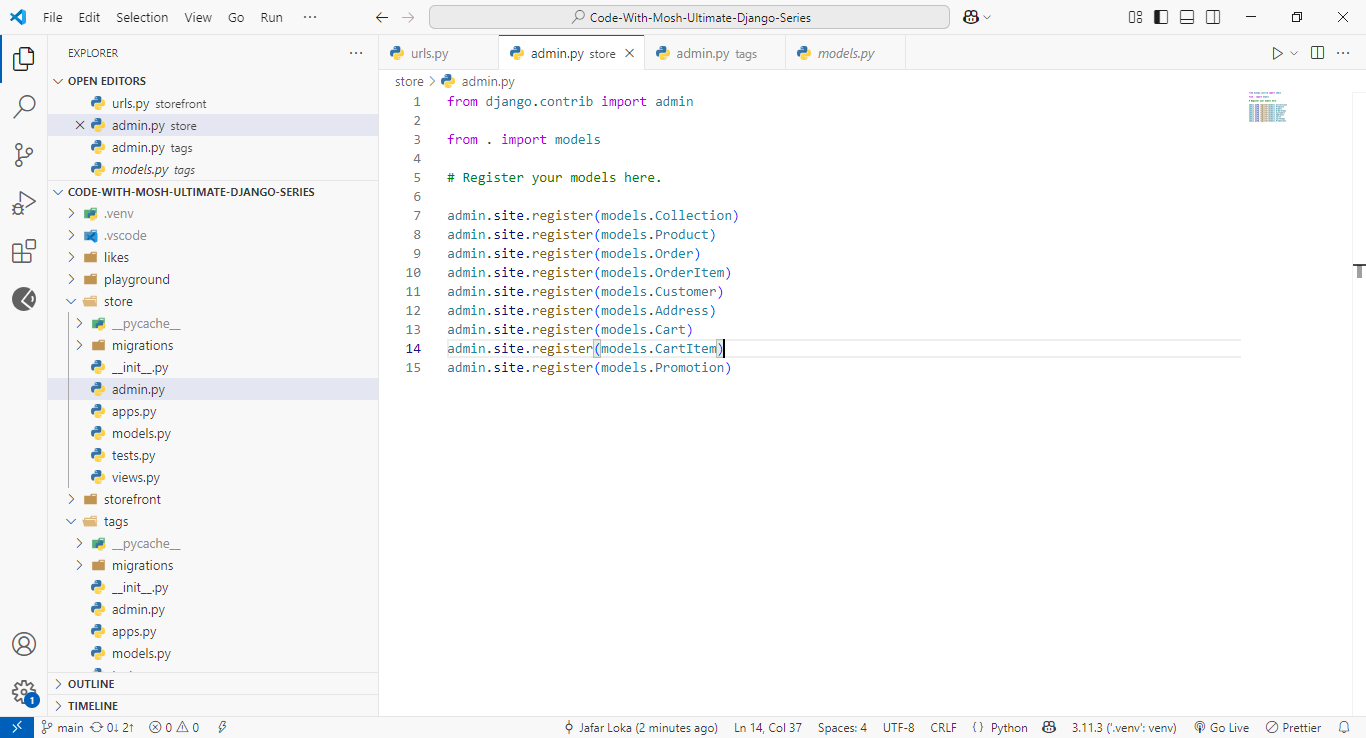
admin.site.register(models.Customer)

admin.site.register(models.Address)

admin.site.register(models.Cart)

admin.site.register(models.CartItem)

admin.site.register(models.Promotion)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Change The Representation Of Model Inside Admin Site, We Can Override The \_\_str\_\_(self)-Method Inside The Model-Class:

class Collection(models.Model):

    title = models.CharField(max\_length=255, unique=True)

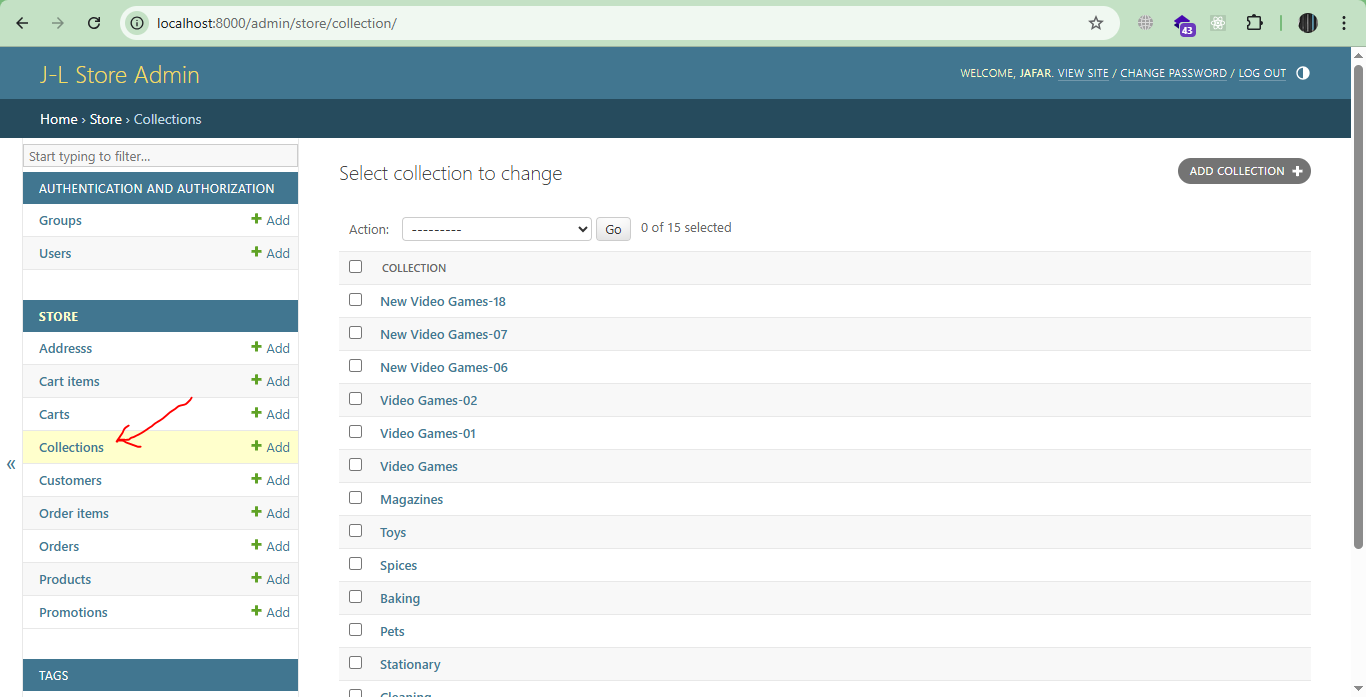
    featured\_product = models.ForeignKey(

        to='Product', on\_delete=models.SET\_NULL, null=True, related\_name='+'

    )

    def \_\_str\_\_(self):

        return self.title



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Override The Ordering Of Model Inside The Admin Site, We Can Use Meta-Class:

class Collection(models.Model):

    title = models.CharField(max\_length=255, unique=True)

    featured\_product = models.ForeignKey(

        to='Product', on\_delete=models.SET\_NULL, null=True, related\_name='+'

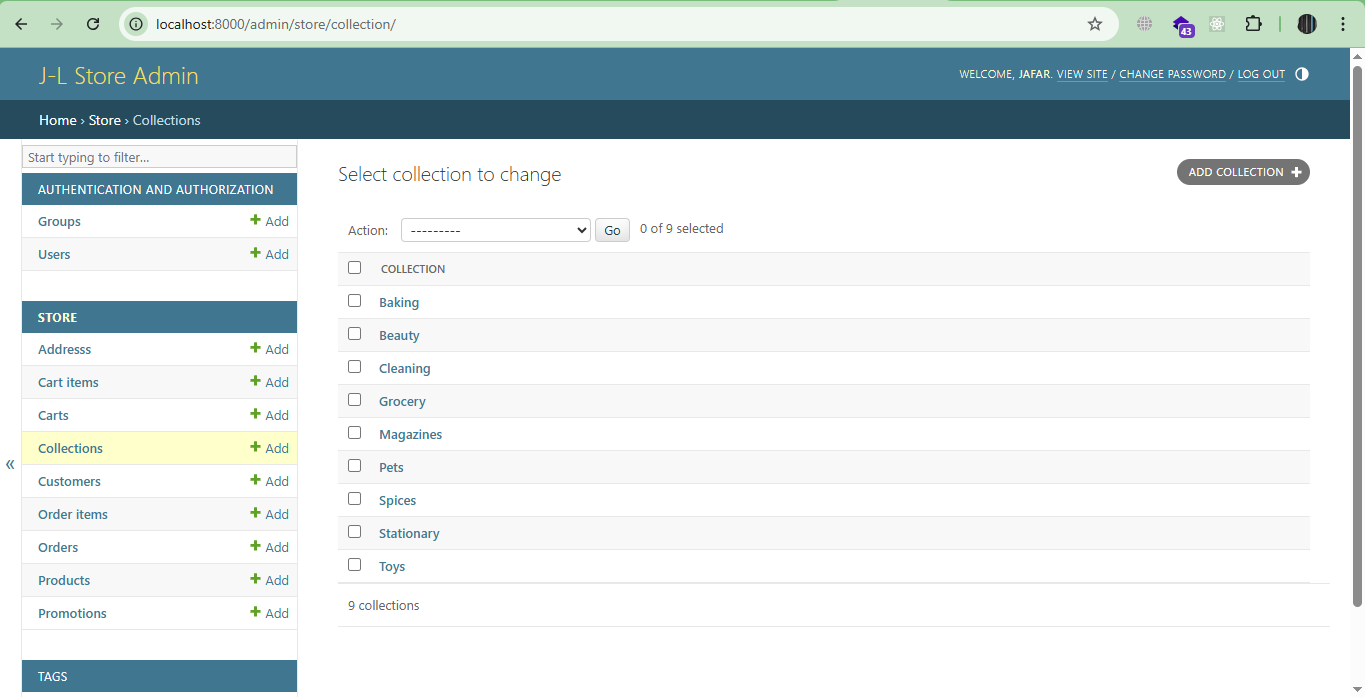
    )

    def \_\_str\_\_(self):

        return self.title

    class Meta:

        ordering = ['title']



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Change The Display Of Model Inside The Admin Site:

* First, We Create Class That Inherit From admin.ModelAdmin
* Second, We Define The Attribute list\_display AS List:

list\_display = ['attr-01', 'attr-02', ...]

class ProductAdmin(admin.ModelAdmin):

    list\_display = ['id', 'title', 'unit\_price', ]

* Then We Have Two Ways To Make It For Product Model:

admin.site.register(models.Product, ProductAdmin)

* OR, We Can Use Decorator
  + **Note**: In This Way We Must Delete The Model From *admin.site.register*

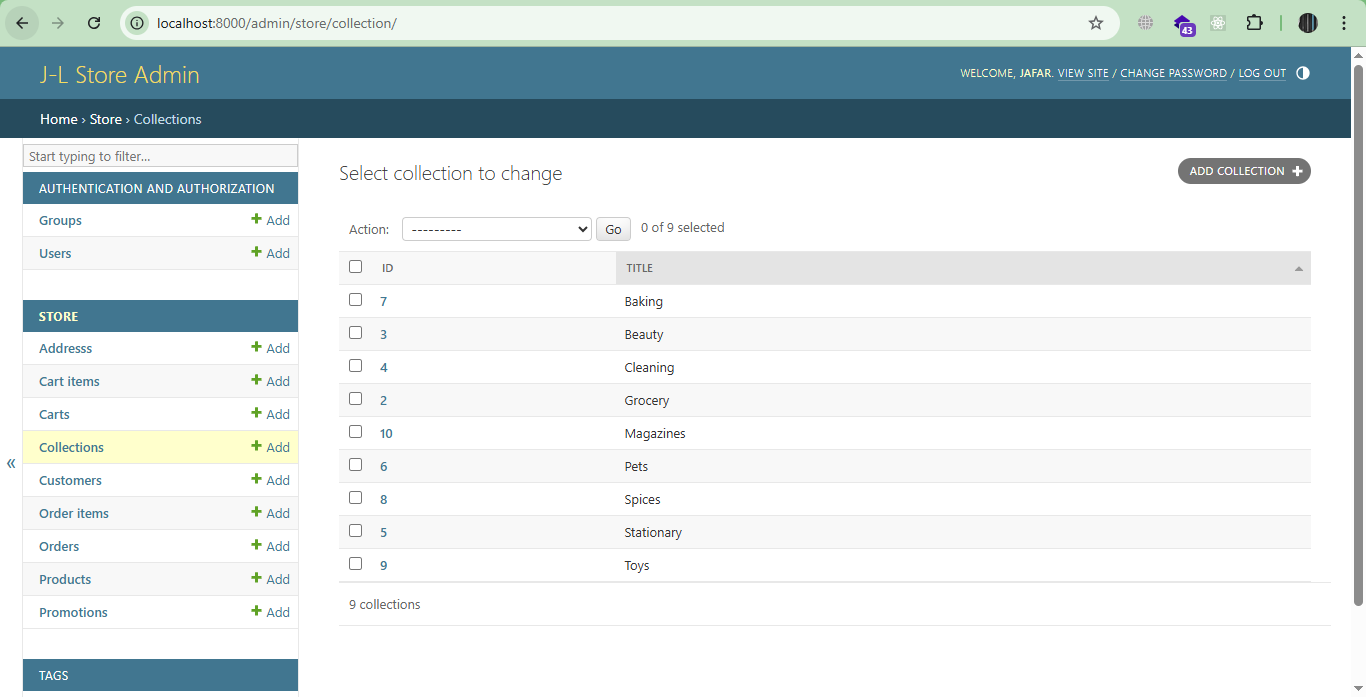
@admin.register(models.Collection)

class CollectionAdmin(admin.ModelAdmin):

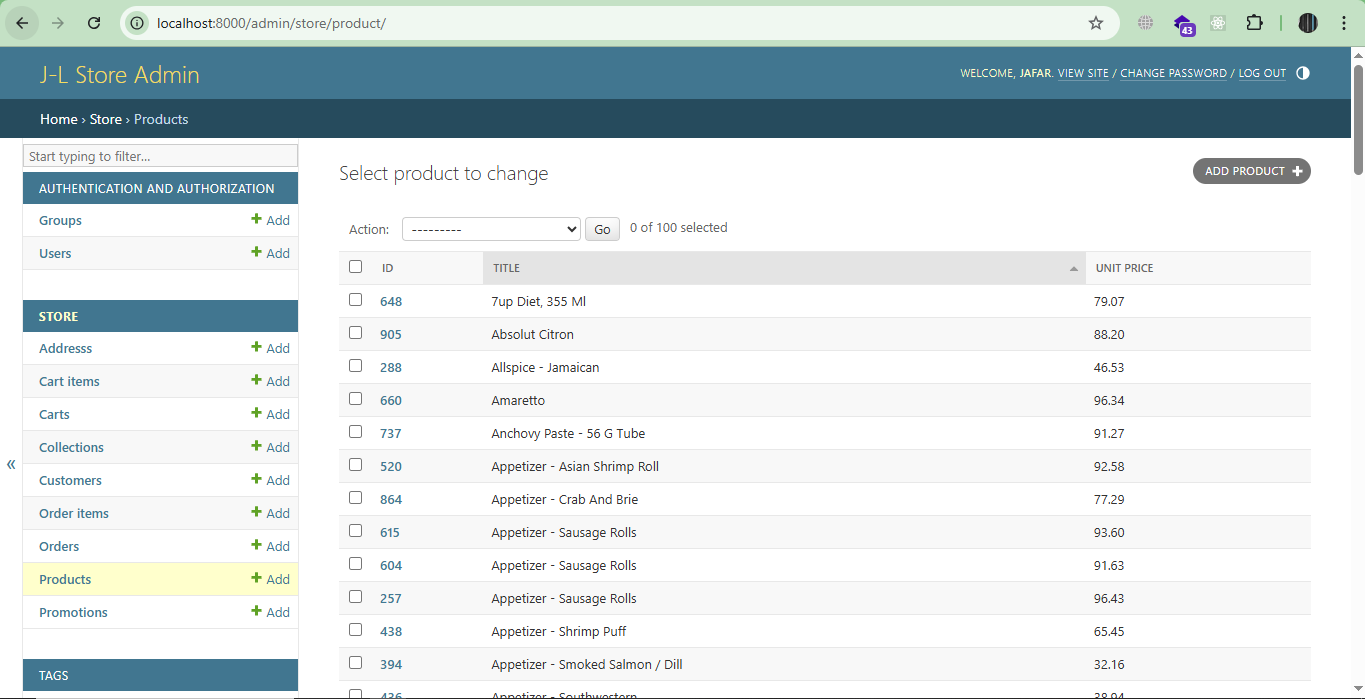
    list\_display = ['id', 'title']

# admin.site.register(models.Collection) # In This Way We Comment This Line

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Note**: The Previous Usage Of list\_display Override The \_\_str\_\_(self)-Method Of Collection, And Product Models

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Basic Transaction Management

1. Using the transaction decorator

from django.db import transaction

@transaction.atomic

def my\_view(request):

# This code executes inside a transaction

if some\_condition:

# Everything will be committed if we reach here

return HttpResponse("Success")

else:

# Explicitly rollback

transaction.set\_rollback(True)

return HttpResponse("Failure", status=400)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 2. Using context managers

from django.db import transaction

def my\_view(request):

try:

with transaction.atomic():

# Database operations here

if not some\_condition:

# Raise an exception to trigger rollback

raise ValueError("Condition not met")

# If we get here, commit happens automatically

return HttpResponse("Success")

except ValueError as e:

return HttpResponse(str(e), status=400)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

### 3. Nested transactions with savepoints

from django.db import transaction, IntegrityError

def complex\_operation():

with transaction.atomic(): # Outer transaction

try:

# Create savepoint

sid = transaction.savepoint()

# Perform some operations

obj1 = Model1.objects.create(field1='value1')

if some\_condition:

# Commit these changes by doing nothing

pass

else:

# Rollback to savepoint

transaction.savepoint\_rollback(sid)

return False

# More operations

obj2 = Model2.objects.create(field2='value2')

return True

except IntegrityError:

# Automatic rollback on exception

return False

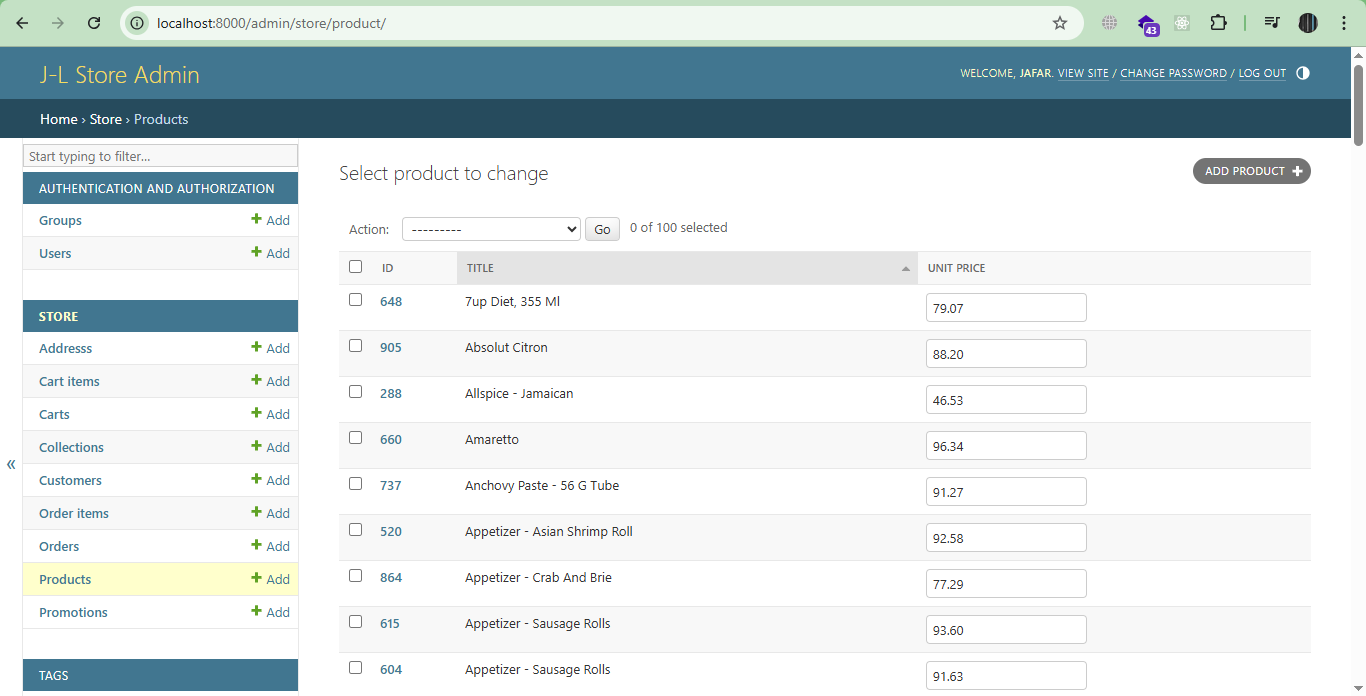
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If We Want To Set The List Of Fields That We Can Edit In Admin Panel:

class ProductAdmin(admin.ModelAdmin):

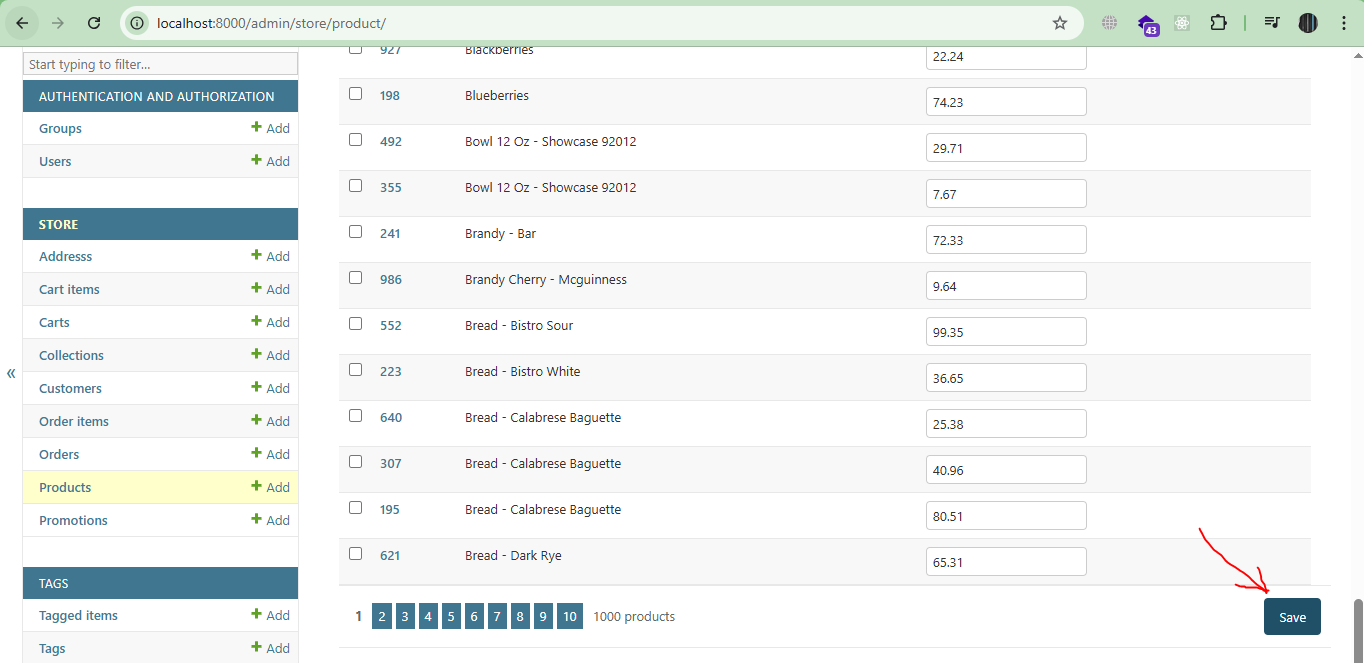
    list\_display = [ 'id', 'title', 'unit\_price', ]

    list\_editable = [ 'unit\_price' ]



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To Save Any Changes To Any Editable Fields:



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*