

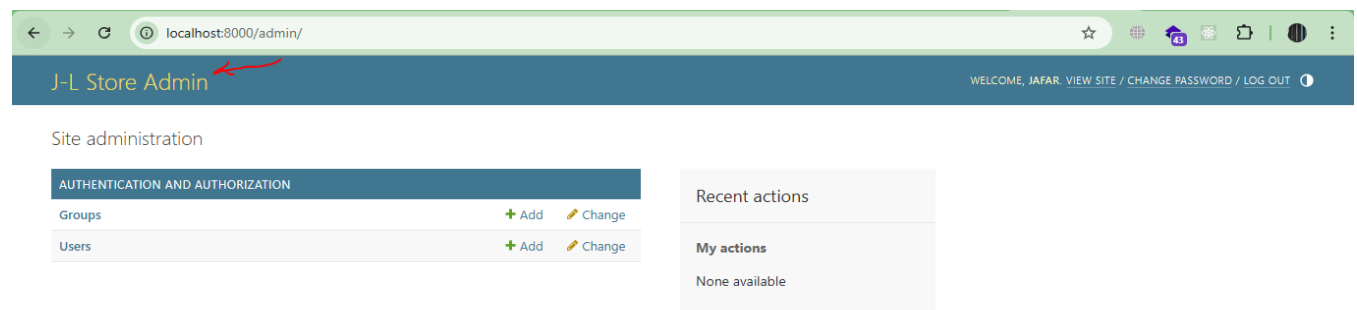
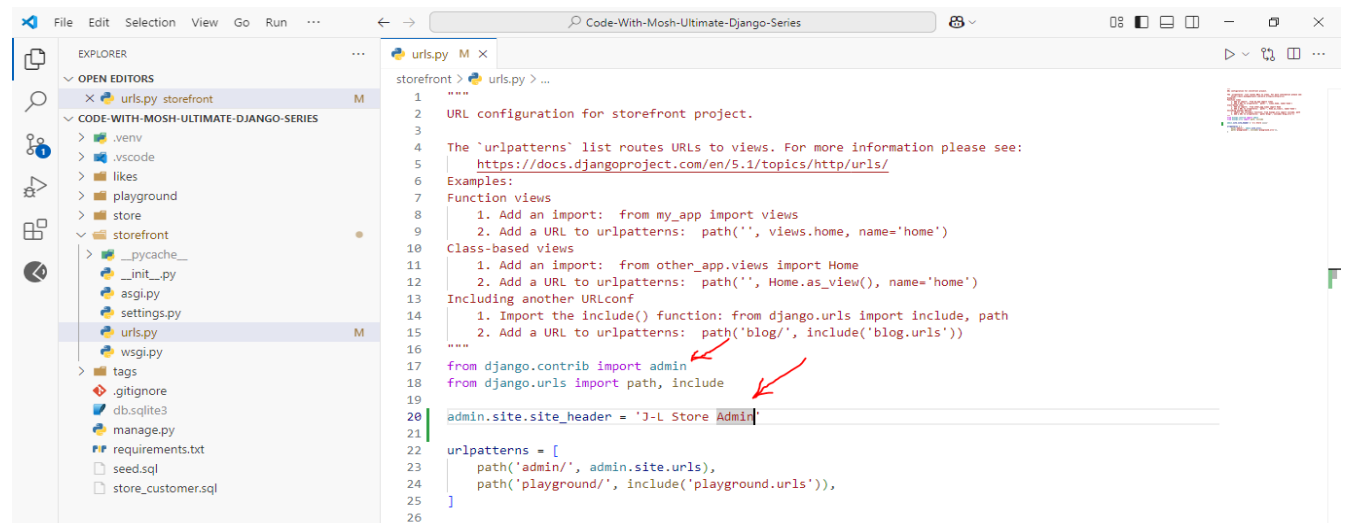
To Change The Admin Password, We Can Run: *python manage.py changepassword user-name*

- Ex: *python manage.py changepassword jafar*

```
<.venv> G:\Web\Django\Code-With-Mosh-Ultimate-Django-Series>python manage.py changepassword jafar
Changing password for user 'jafar'
Password:
Password (again):
Password changed successfully for user 'jafar'
<.venv> G:\Web\Django\Code-With-Mosh-Ultimate-Django-Series>
```

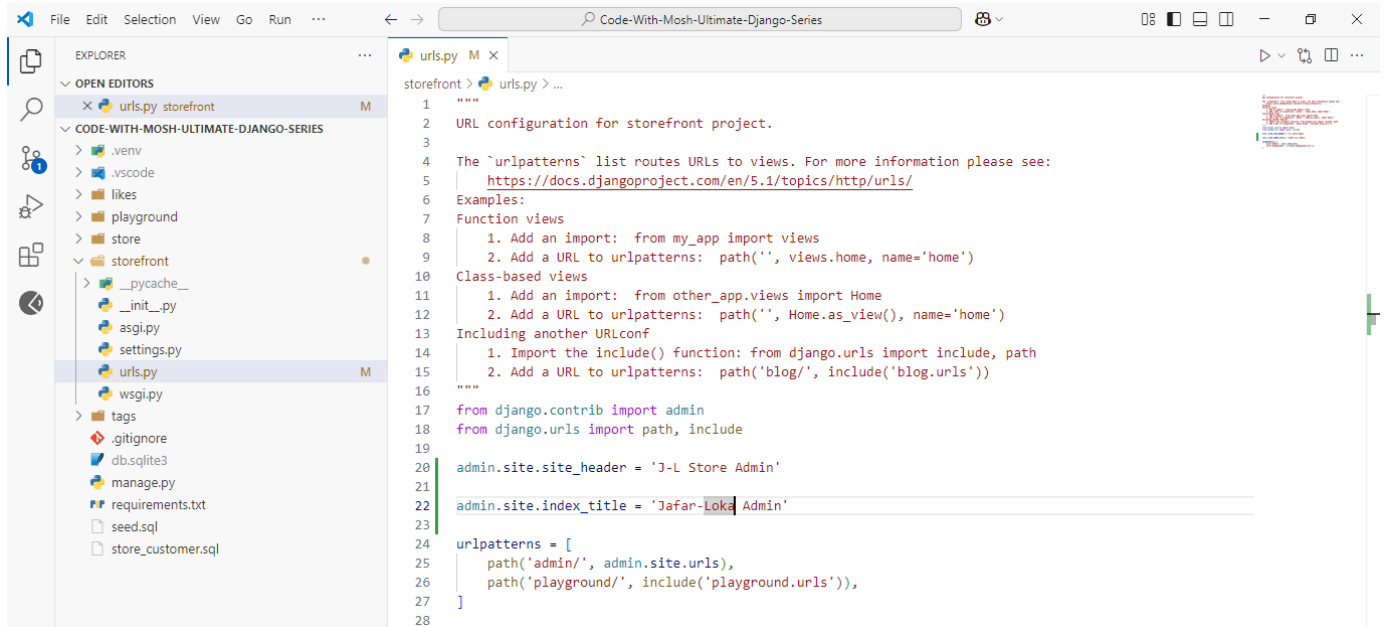
To Change The Header Of The Admin Site:

- 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'*

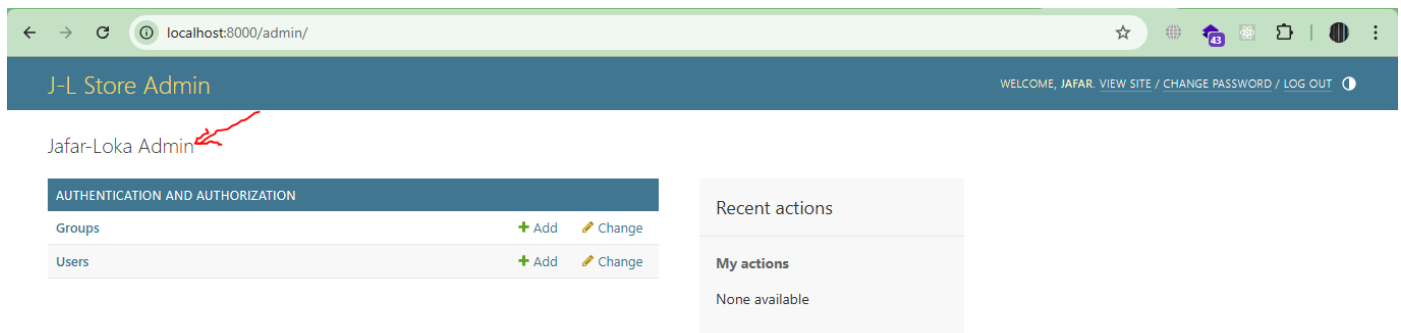


To Change The Index Title, From The Main App Of Project (storefront):

- We Do: `admin.site.index_title = 'Text Here'`
 - Ex: `admin.site.index_title = 'Jafar-Loka Admin'`



```
1  """
2  URL configuration for storefront project.
3
4  The 'urlpatterns' list routes URLs to views. For more information please see:
5  https://docs.djangoproject.com/en/5.1/topics/http/urls/
6  Examples:
7  Function views
8      1. Add an import: from my_app import views
9      2. Add a URL to urlpatterns: path('', views.home, name='home')
10 Class-based views
11     1. Add an import: from other_app.views import Home
12     2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
13 Including another URLconf
14     1. Import the include() function: from django.urls import include, path
15     2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 admin.site.site_header = 'J-L Store Admin'
21
22 admin.site.index_title = 'Jafar-Loka Admin'
23
24 urlpatterns = [
25     path('admin/', admin.site.urls),
26     path('playground/', include('playground.urls')),
27 ]
28
```



To Register Models To Our Admin Site:

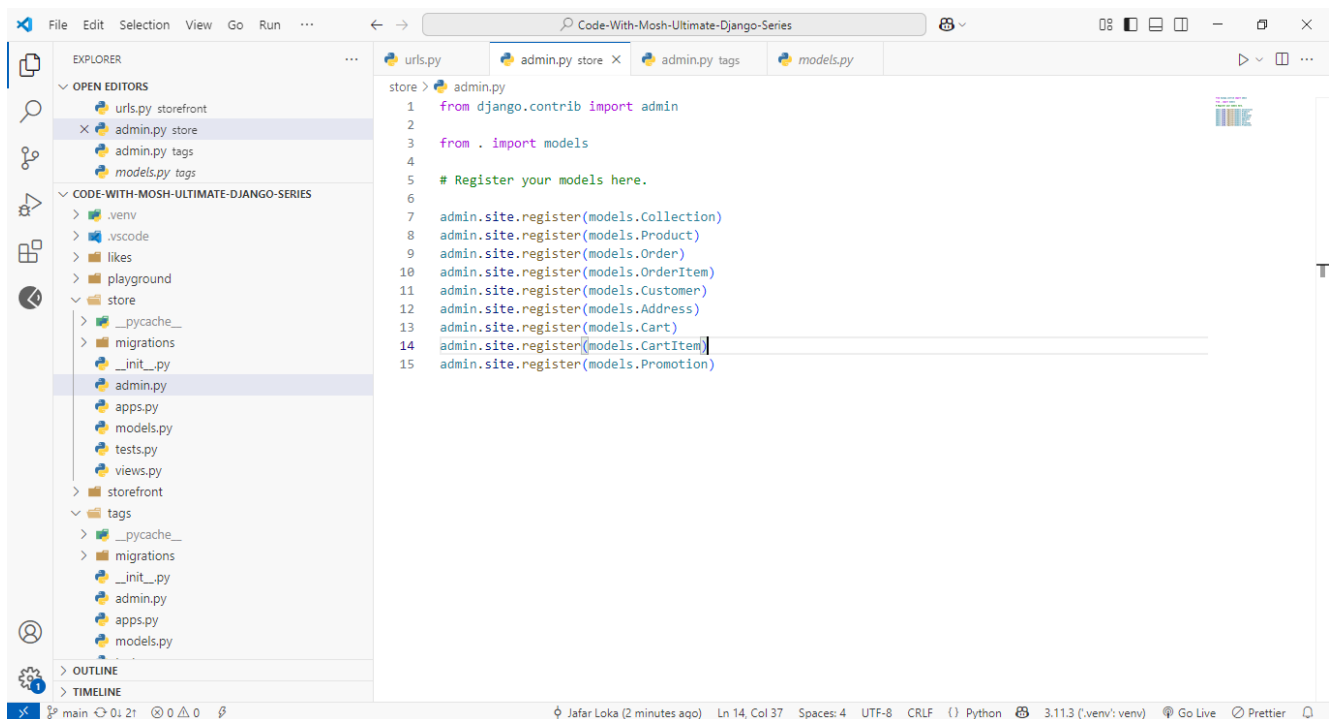
- Every App Has admin.py-File
- Inside It We Register The Models, Using: admin.site.register(Model-Here)

```
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)
```

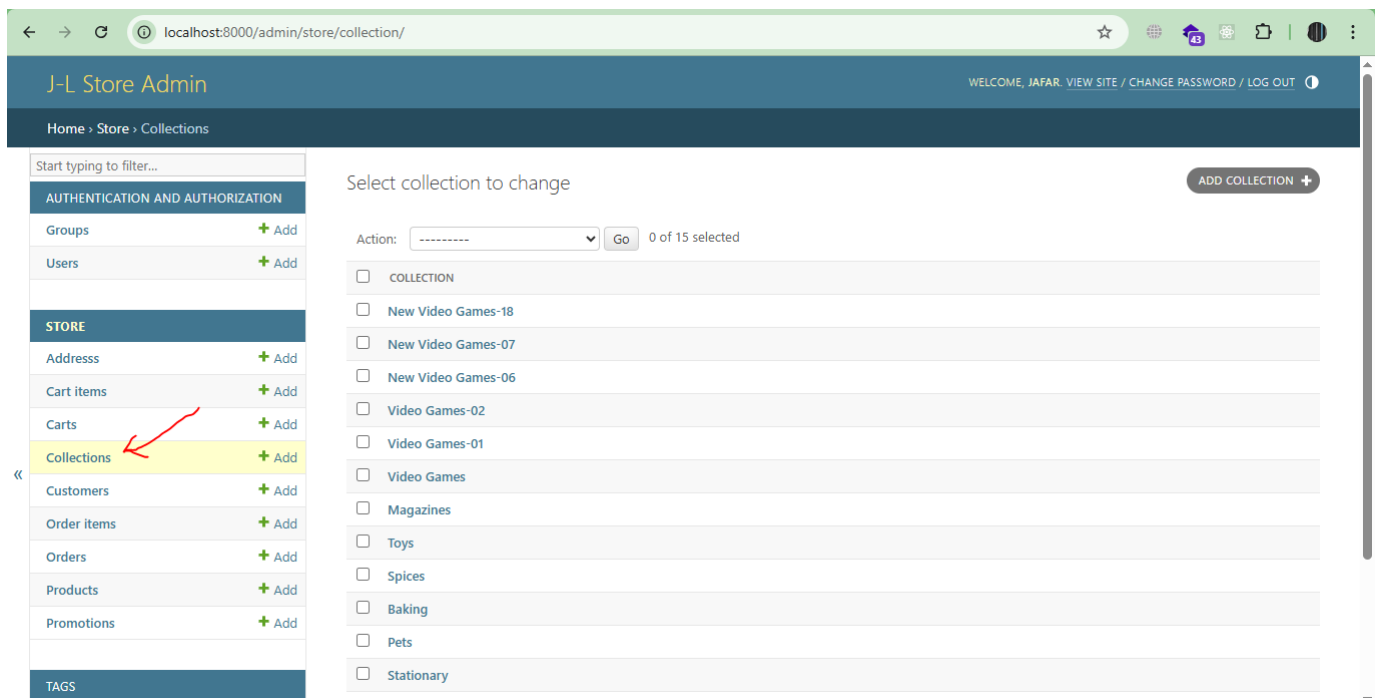


To Change The Representation Of Model Inside Admin Site, We Can Override The `__str__(self)`-Method Inside The Model-Class:

```
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
```



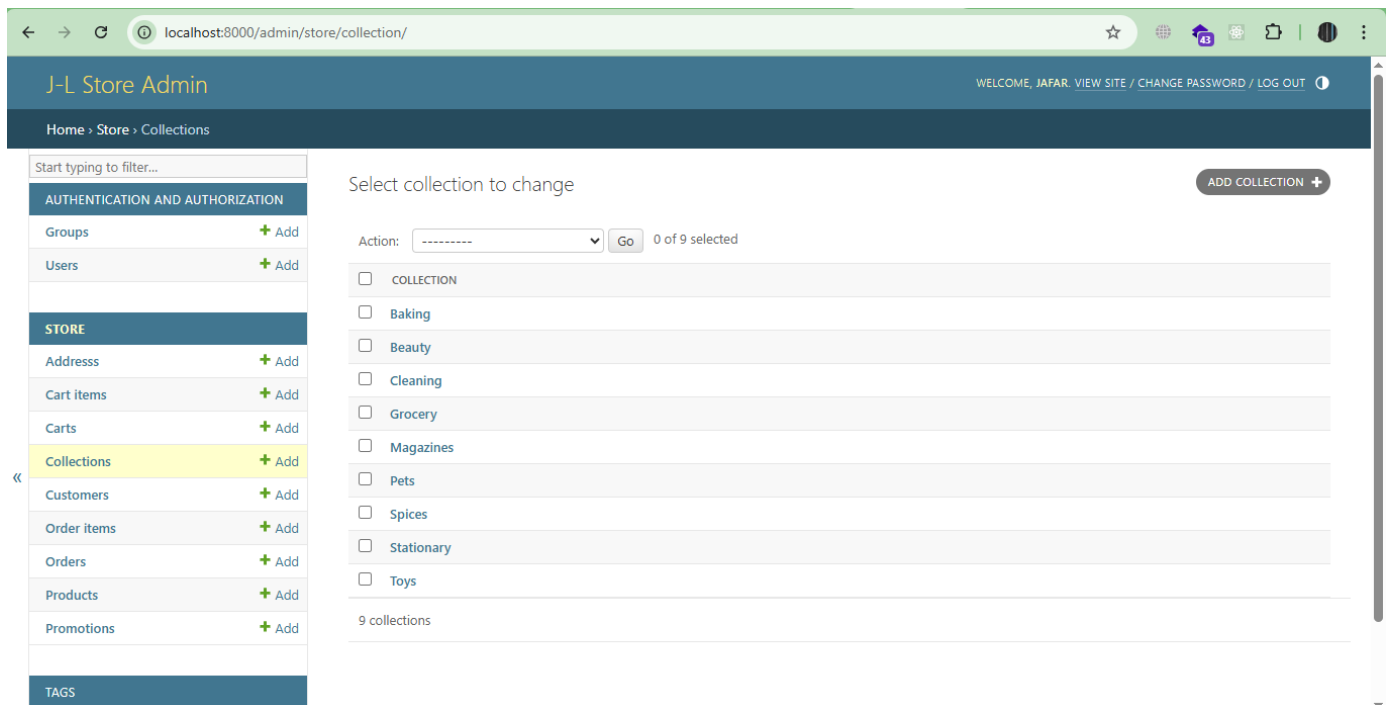
To Override The Ordering Of Model Inside The Admin Site, We Can Use Meta-Class:

```
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:

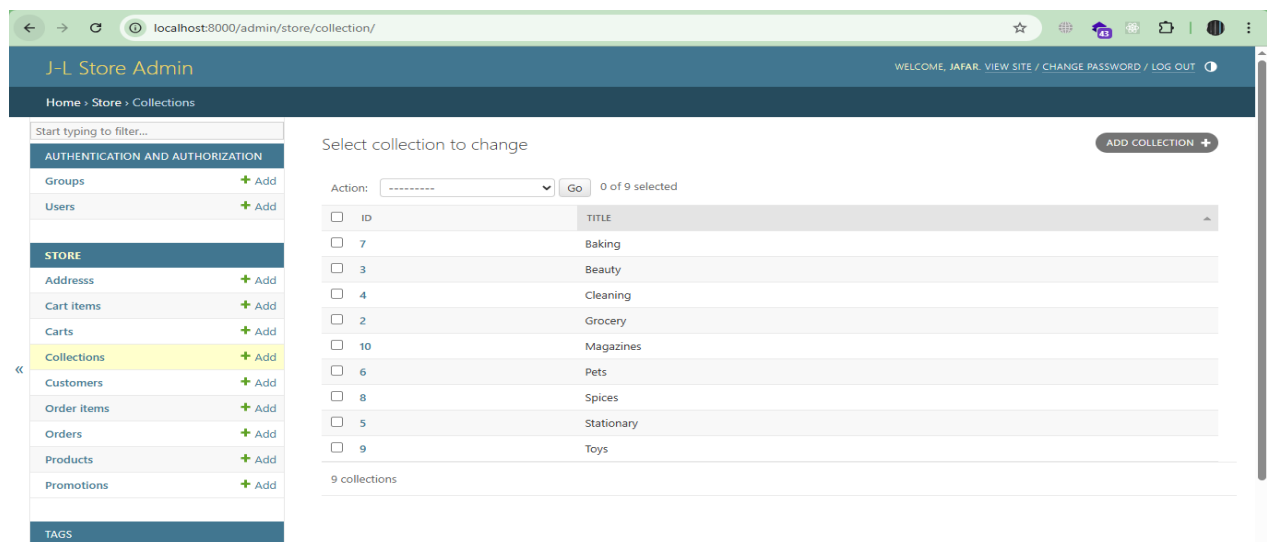
```
admin.site.register(models.Product, ProductAdmin)
```

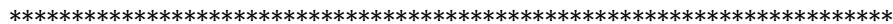
- OR, We Can Use Decorator

- **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





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' ]
```

The screenshot shows the J-L Store Admin interface. The top navigation bar includes the store name, a welcome message for JAFAR, and links for viewing the site, changing the password, and logging out. The breadcrumb trail indicates the current location is Home > Store > Products. The left sidebar menu is categorized into AUTHENTICATION AND AUTHORIZATION (Groups, Users), STORE (Address, Cart items, Carts, Collections, Customers, Order items, Orders, Products, Promotions), and TAGS. The main content area, titled 'Select product to change', features a table of products with columns for ID, TITLE, and UNIT PRICE. Each row includes a checkbox for selection and an input field for the unit price. The table lists 10 products, including 7up Diet, Absolut Citron, Allspice - Jamaican, Amaretto, Anchovy Paste, Appetizer - Asian Shrimp Roll, Appetizer - Crab And Brie, Appetizer - Sausage Rolls, and others. An 'ADD PRODUCT' button is located in the top right corner of the main area.

| ID | TITLE | UNIT PRICE |
|-----|-------------------------------|------------|
| 648 | 7up Diet, 355 Ml | 79.07 |
| 905 | Absolut Citron | 88.20 |
| 288 | Allspice - Jamaican | 46.53 |
| 660 | Amaretto | 96.34 |
| 737 | Anchovy Paste - 56 G Tube | 91.27 |
| 520 | Appetizer - Asian Shrimp Roll | 92.58 |
| 864 | Appetizer - Crab And Brie | 77.29 |
| 615 | Appetizer - Sausage Rolls | 93.60 |
| 604 | Appetizer - Sausage Rolls | 91.63 |

To Save Any Changes To Any Editable Fields:

The screenshot shows a web application interface for managing products. The browser address bar indicates the URL is `localhost:8000/admin/store/product/`. On the left, there is a sidebar menu with sections: 'AUTHENTICATION AND AUTHORIZATION' (containing 'Groups' and 'Users'), 'STORE' (containing 'Address', 'Cart items', 'Carts', 'Collections', 'Customers', 'Order items', 'Orders', 'Products' (highlighted), and 'Promotions'), and 'TAGS' (containing 'Tagged items' and 'Tags'). The main content area displays a table of products. Each row includes a checkbox, a product ID, a product name, and a price field. A red arrow points to the 'Save' button at the bottom right of the table. Below the table, there is a pagination bar showing '10' products and a 'Save' button.

| ID | Product Name | Price |
|-----|-----------------------------|-------|
| 927 | Blackberries | 22.24 |
| 198 | Blueberries | 74.23 |
| 492 | Bowl 12 Oz - Showcase 92012 | 29.71 |
| 355 | Bowl 12 Oz - Showcase 92012 | 7.67 |
| 241 | Brandy - Bar | 72.33 |
| 986 | Brandy Cherry - McGuinness | 9.64 |
| 552 | Bread - Bistro Sour | 99.35 |
| 223 | Bread - Bistro White | 36.65 |
| 640 | Bread - Calabrese Baguette | 25.38 |
| 307 | Bread - Calabrese Baguette | 40.96 |
| 195 | Bread - Calabrese Baguette | 80.51 |
| 621 | Bread - Dark Rye | 65.31 |

1 2 3 4 5 6 7 8 9 10 1000 products

Save
