

1. Create project and application
 - a. Django>django-admin startproject orm_and_frontend
 - b. Django>cd orm_and_frontend
 - c. Django\orm_and_frontend>python manage.py startapp product
2. Open project in vs code
3. Register application in settings.py
4. Create database in workbench
create database orm_and_frontend;
use orm_and_frontend;
5. Register database in settings.py

```
settings.py X
orm_and_frontend > settings.py > ...
76
77 DATABASES = {
78     'default': {
79         'ENGINE': 'django.db.backends.mysql',
80         'NAME': 'orm_and_frontend',
81         'USER': 'root',
82         'PASSWORD': 'root',
83         'HOST': 'localhost',
84         'PORT': '3306'
85     }
86 }
```

6. Create model class for Products, so in models.py

```
models.py X
product > models.py > ProductTable
1 from django.db import models
2
3 # Create your models here.
4 class ProductTable(models.Model):
5     name = models.CharField(max_length=50)
6     price = models.FloatField()
7     details=models.CharField(max_length=150)
8     category = models.IntegerField()
9     is_active= models.BooleanField()
10    rating = models.FloatField()
11
12    def __str__(self) :
13        return self.name + " added to table"
```

7. Register model class in admin.py

```
admin.py X
product > admin.py > ...
1 from django.contrib import admin
2 from product.models import ProductTable
3
4 # Register your models here.
5 class ProductAdmin(admin.ModelAdmin):
6     list_display = ['id', 'name', 'price', 'details', 'category', 'is_active', 'rating']
7
8 admin.site.register(ProductTable, ProductAdmin)
9
```

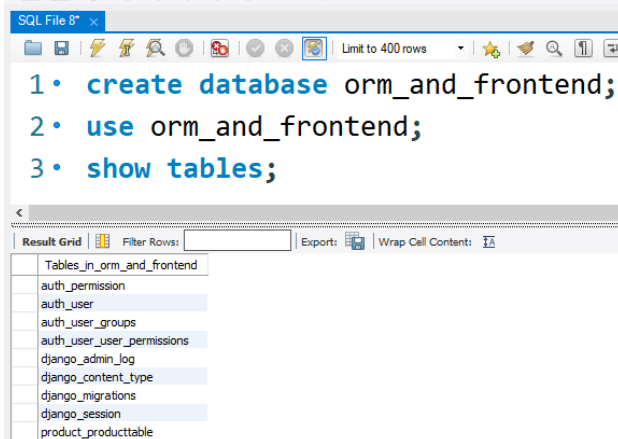
8. Makemigrations

- Django\orm_and_frontend>python manage.py makemigrations

9. Migrate

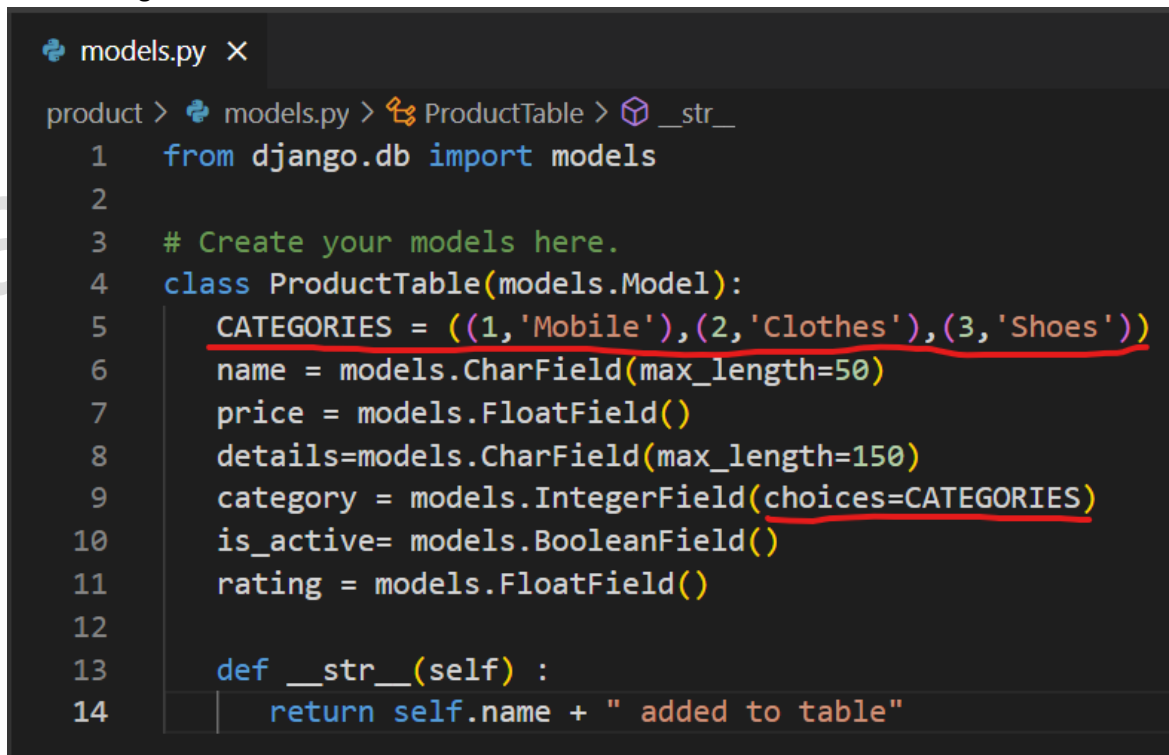
- Django\orm_and_frontend>python manage.py migrate

10. Check tables in orm_and_frontend database (in workbench)

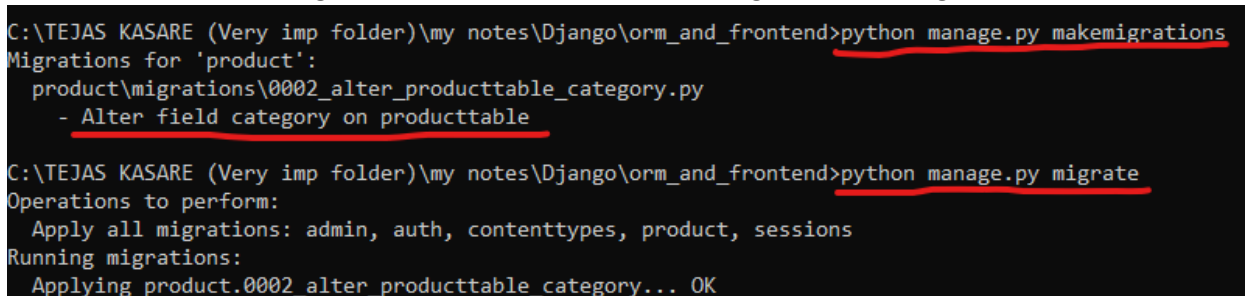


11. Providing categories and drop down option

- Make change in ProductTable model



- Since we have made changes in model, we have to do makemigrations and migrate



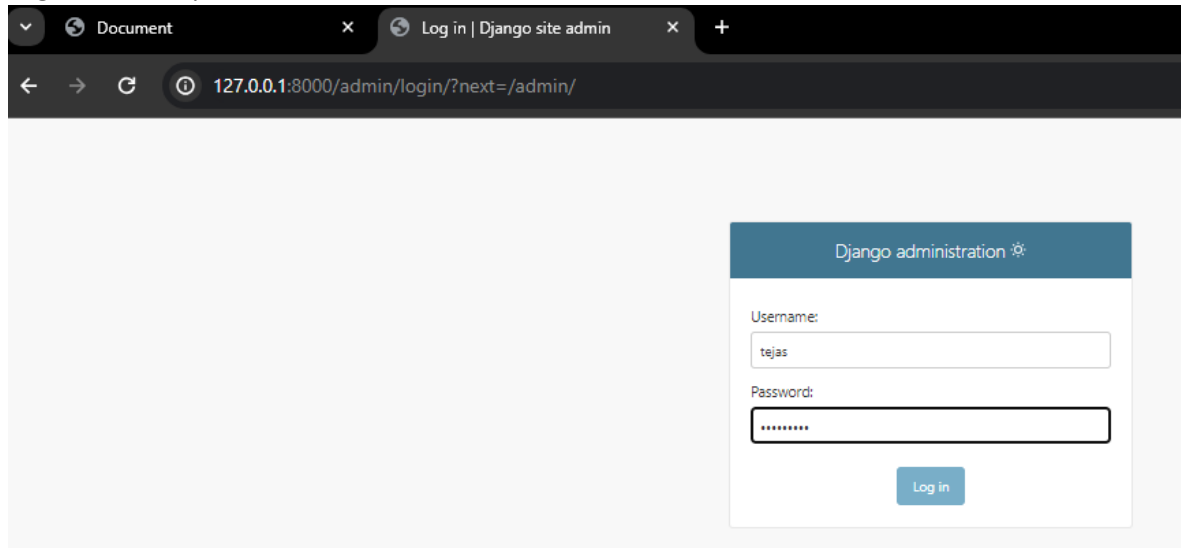
12. Adding some products into product table from admin panel

a. Create superuser

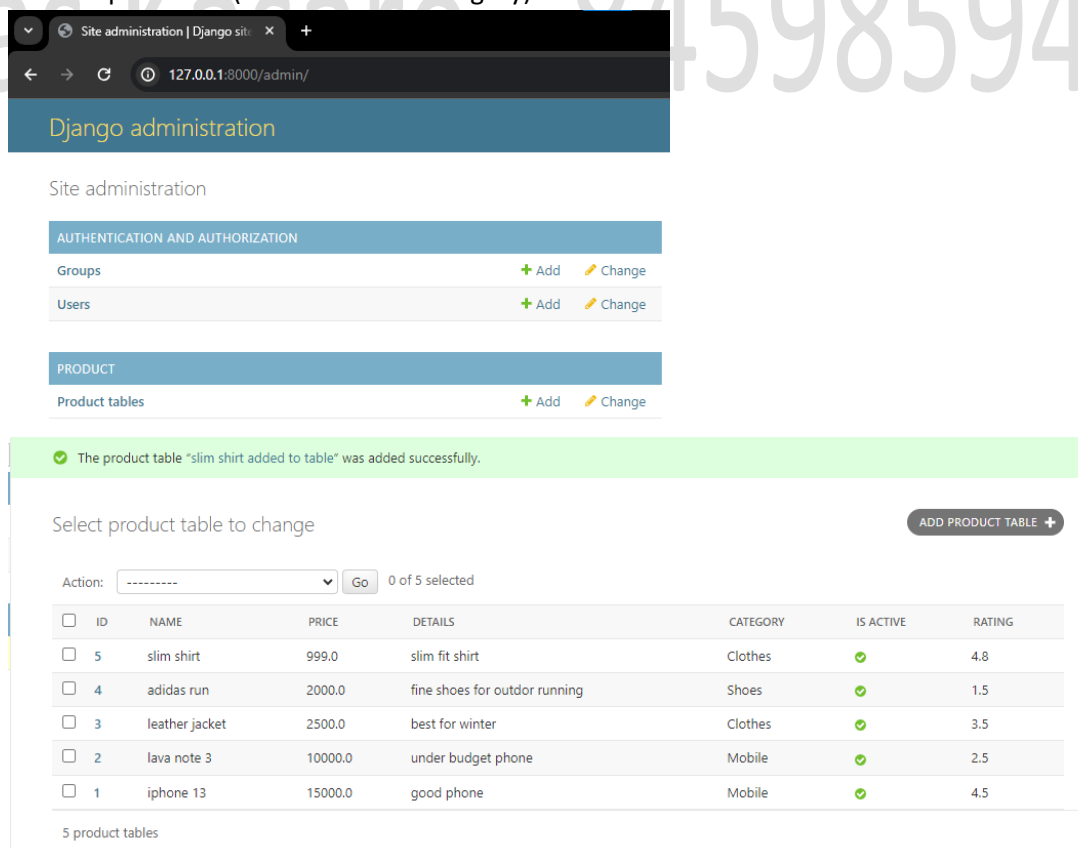
```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py createsuperuser
Username (leave blank to use 'admin'): tejas
Email address: tejas@gmail.com
Password:
Password (again):
Superuser created successfully.
```

b. runserver

c. Login to admin panel

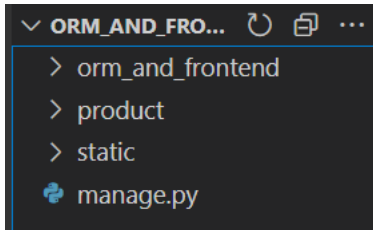


d. Add 4-5 products (min 1 of each category)



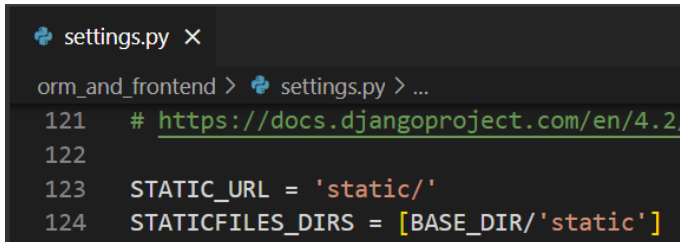
ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING
5	slim shirt	999.0	slim fit shirt	Clothes	✓	4.8
4	adidas run	2000.0	fine shoes for outdoor running	Shoes	✓	1.5
3	leather jacket	2500.0	best for winter	Clothes	✓	3.5
2	lava note 3	10000.0	under budget phone	Mobile	✓	2.5
1	iphone 13	15000.0	good phone	Mobile	✓	4.5

13. Creating static folder to store css file



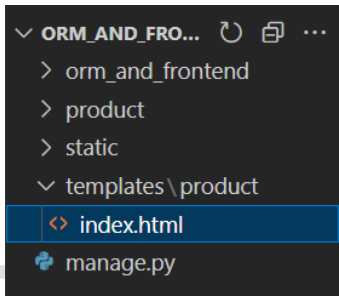
a.

14. Register static folder in settings.py



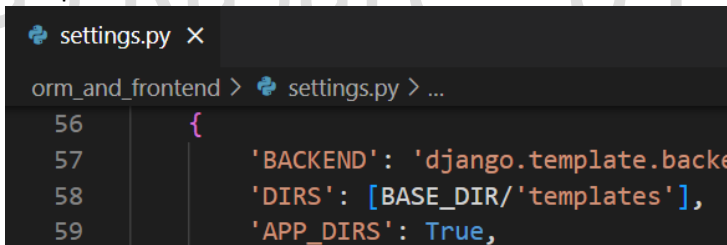
a.

15. Create templates folder, inside templates folder create product folder, create index.html in product folder



a.

16. Register templates folder



a.

17. In static folder do following :

- Create css folder > create product_style.css
- Create images folder > add one shirt image in it (200x200 size)
- add following code in it

```
.container
{
    width: 100%;
    display: flex;
    flex-direction: row;
}

.filter_area
{
    width: 20%;
    display: flex;
    /* background-color: aqua; */
    border: 1px dashed black;
    flex-direction: column;
    padding: 10px;
}
```

```
.product_area
{
  width: 80%;
  display: flex;
  margin-left: 2px;
  /* border: 1px dashed black; */
  flex-wrap: wrap;
}

.card
{
  display: flex;
  flex-direction: column;
  width: 25%;
}

.card .card-items
{
  border: 1px solid black;
  padding: 10px;
  border-radius: 10px;
  display: flex;
  flex-direction: column;
  margin: 10px;
  align-items: center;
  justify-content: center;
  box-shadow: 3px 5px 5px grey;
}

.card .card-items img
{
  border-radius: 5%;
  height: 150px;
  width: 150px;
  padding: 5px;
}

.card .card-items button
{
  border-radius: 20px;
  padding: 12px;
  border: none;
}

.card .card-items #add_to_cart_btn
{
  background-color: #F7CA00;
}

.card .card-items #buy_now_btn
{
  background-color: #FFA41C;
}

.card .card-items button a
{
  text-decoration: none;
  color: black;
}

.card .card-items .card-text
{
  margin-left: 10px;
}
```

18. in templates folder do following

- create product folder > create index.html
- add following code in it

```
<!DOCTYPE html>
<html lang="en">
  {% load static %}
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="{% static 'css/product_style.css' %}">
</head>
<body>
  <div class="navbar"></div>
  <div class="container">
    <div class="filter_area">
      <div class="by_category">
        <h2>Filter By Category</h2>
        <ul>
          <li><a href="">All</a></li>
          <li><a href="">By Mobile</a></li>
          <li><a href="">By Clothes</a></li>
          <li><a href="">By Shoes</a></li>
        </ul>
      </div>
    </div>
    <div>-----</div>
    <div class="by_price">
      <h2>Filter By Price</h2>
      <form action="">
        <label for="">Min:</label>
        <input type="number"> <br><br>
        <label for="">Max:</label>
        <input type="number"> <br><br>
        <input type="submit">
      </form>
    </div>
    <div>-----</div>
    <div class="sort_by_price">
      <h2>Sort By Price</h2>
      <ul>
        <li><a href="">High to Low</a></li>
        <li><a href="">Low to High</a></li>
      </ul>
    </div>
    <div>-----</div>
    <div class="by_rating">
      <h2>Sort By Rating</h2>
      <ul>
        <li><a href="">3 and above</a></li>
        <li><a href="">4 and above</a></li>
      </ul>
    </div>
    <div>-----</div>
    <div class="product_area">
      <div class="card">
        <div class="card-items">
          
          <div class="card-text">
            <p>Cotton King</p>
            <p>499</p>
            <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
            <button id="buy_now_btn"><a href="">Buy Now</a></button>
          </div>
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

19. create view to display index.html file

```
views.py X
product > views.py > ...
1 from django.shortcuts import render
2
3 # Create your views here.
4 def index(request):
5     return render(request, 'product/index.html')
6
```

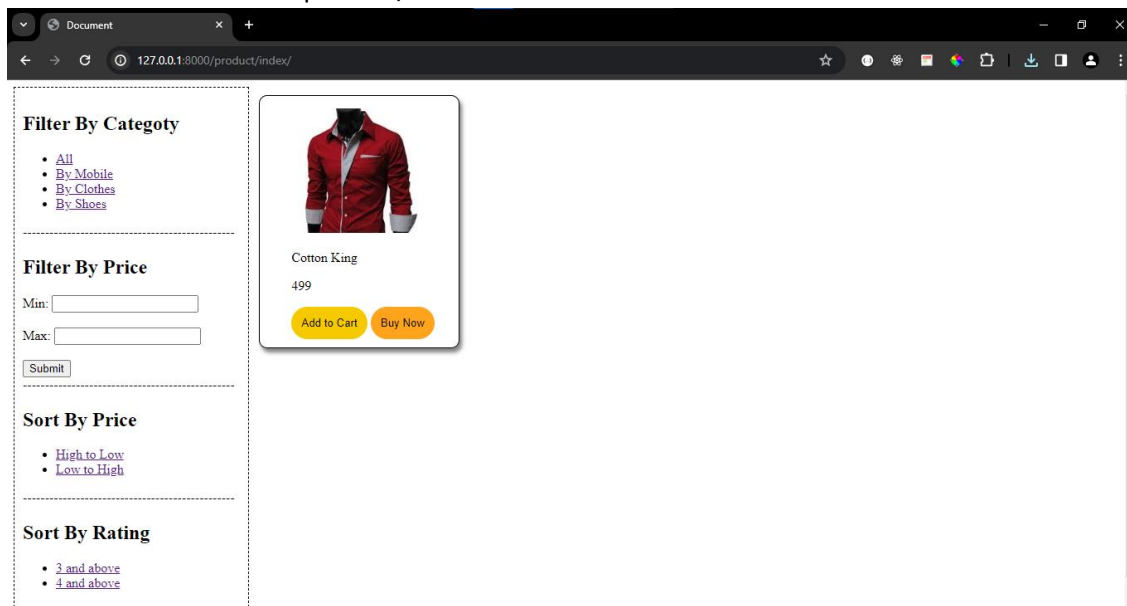
20. create product_urls.py in product folder

```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6 ]
```

21. create url for above application level url in project level url (urls.py)

```
urls.py X
orm_and_frontend > urls.py > ...
16 """
17 from django.contrib import admin
18 from django.urls import path, include
19
20 urlpatterns = [
21     path('admin/', admin.site.urls),
22     path('product/', include('product.product_urls')),
23 ]
```

22. runserver and check for product/index url



OPERATIONS

1. get all products and display in index.html
 - a. add logic to fetch data in index() view

```
views.py x
product > views.py > ...
1  from django.shortcuts import render
2  from product.models import ProductTable
3
4  # Create your views here.
5  def index(request):
6      data={}
7      #ProductTable.objects.all() this will fetch non active product also. so it is better to use filter
8      fetched_products=ProductTable.objects.filter(is_active=True)
9      data['products']=fetched_products
10     return render(request,'product/index.html',context=data)
11
```

- b. pass fetched data to index.html and display using loop

```
index.html x
templates > product > index.html
50 <div>-----</div>
51 </div>
52 <div class="product_area">
53     {% for product in products %}
54     <div class="card">
55         <div class="card-items">
56             
57             <div class="card-text">
58                 <p>{{product.name|upper}}</p>
59                 <p>{{product.price}}</p>
60                 <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
61                 <button id="buy_now_btn"><a href="">View More</a></button>
62             </div>
63         </div>
64     </div>
65     {% endfor %}
66 </div>
67 </div>
68 </body>
69 </html>
```

IMPORTANT : in above code I have changed Buy Now button to View More
We will add buy now option when we show each product indivisibly (product details)

2. Implementing Filter by category logic

a. Create view

```
views.py X
product > views.py > ...
12
13 def filter_by_category(request,category_value):
14     #select * from product where is_active=True and category=category_value;
15     #ProductTable.objects.filter(is_active=True , category=category_value)
16     #from django.db.models import Q
17     data={}
18     q1 = Q(is_active=True)
19     q2 = Q(category=category_value)
20     filtered_products=ProductTable.objects.filter(q1 & q2)
21     data['products']=filtered_products
22     return render(request,'product/index.html',context=data)
```

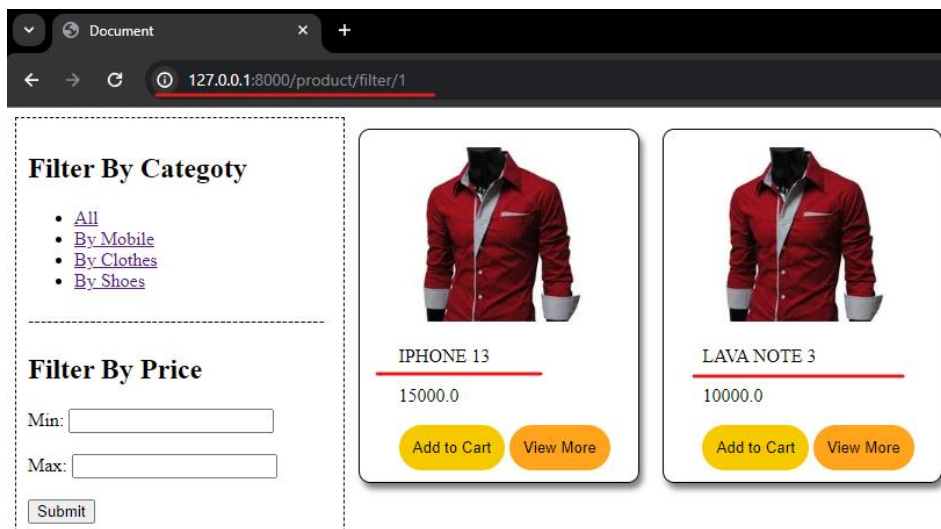
b. Create url for view

```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7 ]
```

c. Use url in index.html

```
index.html X
templates > product > index.html
14 <div class="by_category">
15     <h2>Filter By Category</h2>
16     <ul>
17         <li><a href="/product/index">All</a></li>
18         <li><a href="/product/filter/1">By Mobile</a></li>
19         <li><a href="/product/filter/2">By Clothes</a></li>
20         <li><a href="/product/filter/3">By Shoes</a></li>
21     </ul>
22 </div>
```

Result :



3. Implementing sorting logic (high to low and low to high)

a. Create view

```
views.py
product > views.py > ...
23
24 def sort_by_price(request, sort_value):
25     #select * from product order by salary desc;
26     data={}
27     if sort_value=='asc':
28         price = 'price'
29     else:
30         price = '-price'
31     sorted_products=ProductTable.objects.filter(is_active=True).order_by(price)
32     data['products']=sorted_products
33     return render(request, 'product/index.html', context=data)
34
```

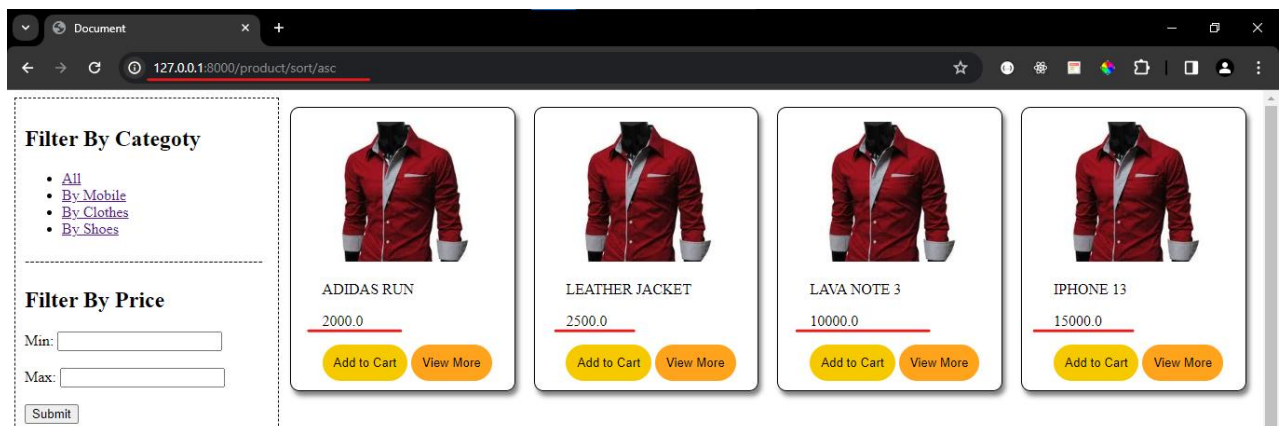
b. Create url for view

```
product_urls.py
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8 ]
```

c. Use url in index.html

```
index.html
templates > product > index.html
34 <div>-----</div>
35     <div class="sort_by_price">
36         <h2>Sort By Price</h2>
37         <ul>
38             <li><a href="/product/sort/asc">High to Low</a></li>
39             <li><a href="/product/sort/asc">Low to High</a></li>
40         </ul>
41     </div>
42 </div>-----</div>
```

d. Result

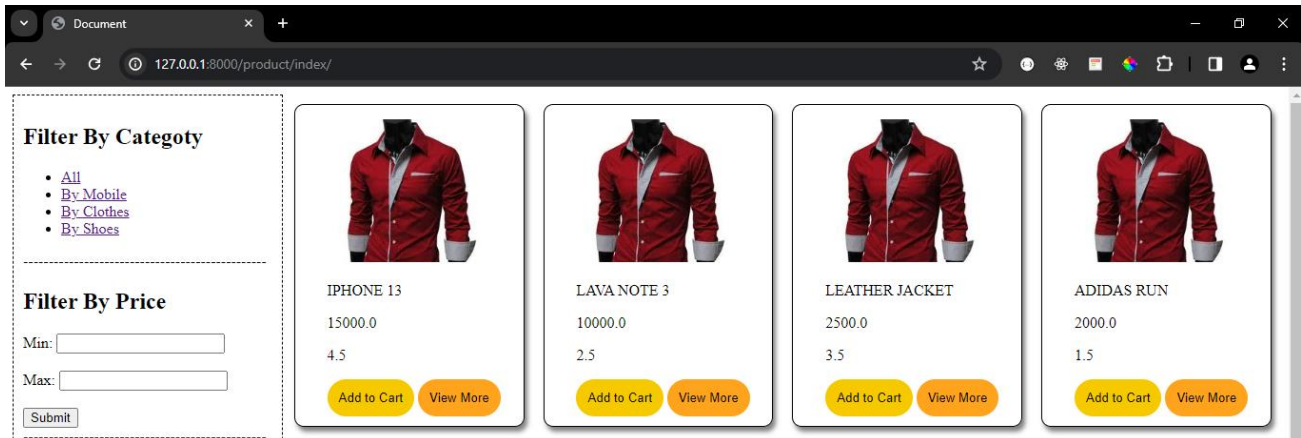


4. Implementing filter by rating logic

- a. Since we are not displaying ratings in index.html, let's display first

```
div class="card-items">
  
  <div class="card-text">
    <p>{{product.name|upper}}</p>
    <p>{{product.price}}</p>
    <p>{{product.rating}}</p>
    <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
    <button id="buy_now_btn"><a href="">View More</a></button>
  </div>
```

- b. Result



- c. Create view

```
views.py
product > views.py > filter_by_price_range
35
36 def filter_by_rating(request, rating_value):
37     #select * from product where is_active=True and category=category_value;
38     #ProductTable.objects.filter(is_active=True, category=category_value)
39     #from django.db.models import Q
40     data={}
41     q1 = Q(is_active=True)
42     q2 = Q(rating__gt=rating_value)
43     filtered_products=ProductTable.objects.filter(q1 & q2)
44     data['products']=filtered_products
45     return render(request, 'product/index.html', context=data)
46
```

- d. Create url for view

```
product_urls.py
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8     path('rating/<rating_value>', views.filter_by_rating),
9 ]
```

- e. Use url in index.html

```
index.html X
templates > product > index.html
42 <div>-----</div>
43 <div class="by_rating">
44   <h2>Sort By Rating</h2>
45   <ul>
46     <li><a href="/product/rating/3">3 and above</a></li>
47     <li><a href="/product/rating/4">4 and above</a></li>
48   </ul>
49 </div>
50 <div>-----</div>
```

5. Filter by price range

- a. Create view

```
views.py X
product > views.py > filter_by_price_range
47
48 def filter_by_price_range(request):
49     data={}
50     min = request.GET['min']
51     max = request.GET['max']
52     q1 = Q(price__gte=min)
53     q2 = Q(price__lte=max)
54     q3 = Q(is_active=True)
55     filtered_products=ProductTable.objects.filter(q1 & q2 & q3)
56     data['products']=filtered_products
57     return render(request,'product/index.html',context=data)
```

- b. Create url for view

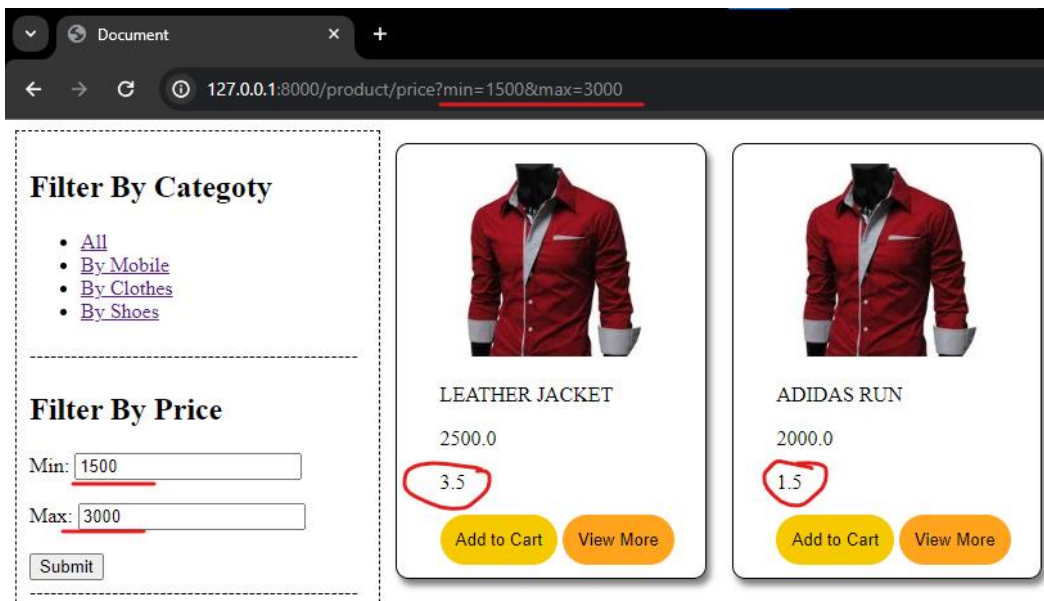
```
product_urls.py X
product > product_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('index/', views.index),
6     path('filter/<category_value>', views.filter_by_category),
7     path('sort/<sort_value>', views.sort_by_price),
8     path('rating/<rating_value>', views.filter_by_rating),
9     path('price', views.filter_by_price_range),
10 ]
```

IMPORTANT : there is no / after price

- c. User url in index.html

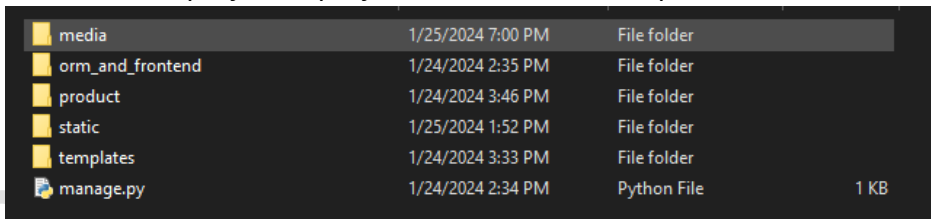
```
index.html X
templates > product > index.html
23 <div>-----</div>
24 <div class="by_price">
25   <h2>Filter By Price</h2>
26   <form action="/product/price">
27     <label for="">Min:</label>
28     <input type="number" name="min"> <br><br>
29     <label for="">Max:</label>
30     <input type="number" name="max"> <br><br>
31     <input type="submit">
32   </form>
33 </div>
34 <div>-----</div>
```

Result :

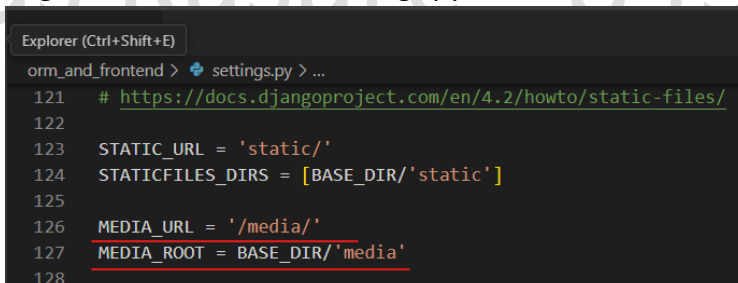


6. Uploading Image

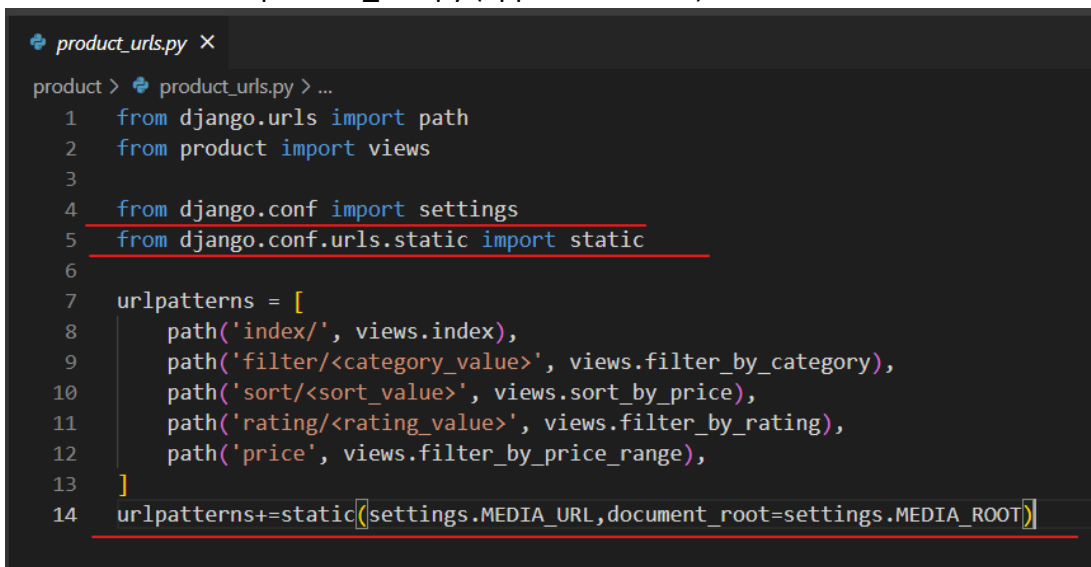
- Create media project in project like we create templates folder



- Register media folder in settings.py



- Create media url in product_urls.py (application level)



- d. Since we are not having image field in our Product model, so lets add image field in it

```
models.py X
product > models.py > ProductTable > __str__
1 from django.db import models
2
3 # Create your models here.
4 class ProductTable(models.Model):
5     CATEGORIES = ((1,'Mobile'),(2,'Clothes'),(3,'Shoes'))
6     name = models.CharField(max_length=50)
7     price = models.FloatField()
8     details=models.CharField(max_length=150)
9     category = models.IntegerField(choices=CATEGORIES)
10    is_active= models.BooleanField()
11    rating = models.FloatField()
12    image=models.ImageField(upload_to='image')
13
14    def __str__(self) :
15        return self.name + " added to table"
```

- e. Makemigrations

```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py makemigrations
It is impossible to add a non-nullable field 'image' to producttable without specifying a default. This
is because the database needs something to populate existing rows.
Please select a fix:
1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
2) Quit and manually define a default value in models.py.
Select an option: 1
Please enter the default value as valid Python.
The datetime and django.utils.timezone modules are available, so it is possible to provide e.g. timezone
e.now as a value.
Type 'exit' to exit this prompt
>>> 0
Migrations for 'product':
  product\migrations\0003_producttable_image.py
    - Add field image to producttable

C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>
```

After running makemigrations, you will get this error because

- f. Migrate
g. Add "image" in list_display in admin.py

```
admin.py X
product > admin.py > ...
1 from django.contrib import admin
2 from product.models import ProductTable
3
4 # Register your models here.
5 class ProductAdmin(admin.ModelAdmin):
6     list_display = ['id','name','price','details','category','is_active','rating','image']
7
8     admin.site.register(ProductTable,ProductAdmin)
```

- h. Runserver

- i. Check table in workbench, you will get image filed with default value 0

```
5 • select * from product_producttable;
```

	id	name	price	details	category	is_active	rating	image
1	1	iphone 13	15000	good phone	1	1	4.5	0
2	2	lava note 3	10000	under budget phone	1	1	2.5	0
3	3	leather jacket	2500	best for winter	2	1	3.5	0
4	4	adidas run	2000	fine shoes for outdoor running	3	1	1.5	0
5	5	slim shirt	999	slim fit shirt	2	0	4.8	0
*		NULL	NULL	NULL	NULL	NULL	NULL	NULL

- j. Download one shoe image (200x200) and add one shoe product with image

Teja 15

Add product table | Django site

127.0.0.1:8000/admin/product/producttable/add/

Django administration

Home > Product > Product tables > Add product table

Start typing to filter...

AUTHENTICATION AND AUTHORIZATION

Groups [+ Add](#)

Users [+ Add](#)

PRODUCT

Product tables [+ Add](#)

Add product table

Name:

Price:

Details:

Category:

☒ Is active

Rating:

Image: shoe1.jpg

[SAVE](#) [Save and add another](#) [Save and continue editing](#)

✓ The product table "nike fast x added to table" was added successfully.

Select product table to change [ADD PRODUCT TABLE +](#)

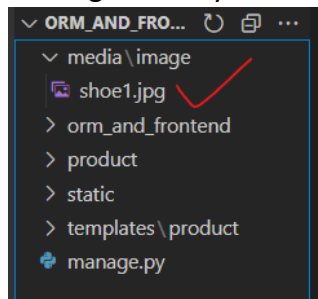
Action: Go 0 of 6 selected

<input type="checkbox"/>	ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING	IMAGE
<input type="checkbox"/>	6	nike fast x	3999.0	fine shoes for outdoor running	Shoes	✓	4.5	image/shoe1.jpg
<input type="checkbox"/>	5	slim shirt	999.0	slim fit shirt	Clothes	✗	4.8	0
<input type="checkbox"/>	4	adidas run	2000.0	fine shoes for outdoor running	Shoes	✓	1.5	0
<input type="checkbox"/>	3	leather jacket	2500.0	best for winter	Clothes	✓	3.5	0
<input type="checkbox"/>	2	lava note 3	10000.0	under budget phone	Mobile	✓	2.5	0
<input type="checkbox"/>	1	iphone 13	15000.0	good phone	Mobile	✓	4.5	0

6 product tables

k. Check in media folder –

- i. You will get image folder
- ii. In image folder you will get your uploaded image



l. Add image for other products

✓ The product table "iphone 13 added to table" was changed successfully.

Select product table to change ADD PRODUCT TABLE +

Action: Go 0 of 6 selected

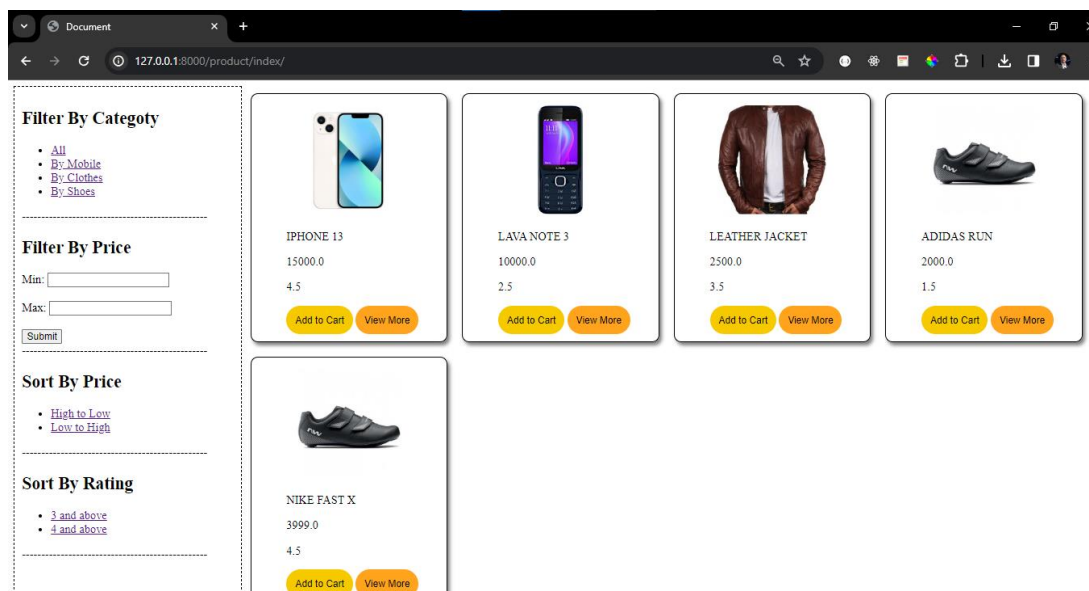
<input type="checkbox"/>	ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING	IMAGE
<input type="checkbox"/>	6	nike fast x	3999.0	fine shoes for outdoor running	Shoes	✓	4.5	image/shoe1.jpg
<input type="checkbox"/>	5	slim shirt	999.0	slim fit shirt	Clothes	✗	4.8	image/shirt.webp
<input type="checkbox"/>	4	adidas run	2000.0	fine shoes for outdoor running	Shoes	✓	1.5	image/shoe1_YPVBJ1s.jpg
<input type="checkbox"/>	3	leather jacket	2500.0	best for winter	Clothes	✓	3.5	image/jacket.webp
<input type="checkbox"/>	2	lava note 3	10000.0	under budget phone	Mobile	✓	2.5	image/mobile1.jpg
<input type="checkbox"/>	1	iphone 13	15000.0	good phone	Mobile	✓	4.5	image/mobile2.jpg

6 product tables

m. Show images dynamically in index.html for each product

```
index.html X
templates > product > index.html > html > body > div.container > div.product_area > div.card > div.card-items >
54 <div class="card">
55 <div class="card-items">
56 <!-- 
57 
59 <p>{{product.name|upper}}</p>
60 <p>{{product.price}}</p>
```

OUTPUT :



7. View More Functionality (View individual product details)

a. Create view to fetch product based on id

```
views.py X
product > views.py > product_detail
59 def product_detail(request,pid):
60     product=ProductTable.objects.get(id=pid)
61     return render(request,'product/product_detail.html',{'product':product})
```

b. Create html file : product_detail.html in templates>product> product_detail.html

```
product_detail.html X
templates > product > product_detail.html
2 <html lang="en">
3     {% load static %}
4 <head>
5     <meta charset="UTF-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1.0">
7     <title>Document</title>
8     <link rel="stylesheet" href="{% static 'css/product_style.css' %}">
9 </head>
10 <body>
11     <div class="card">
12         <div class="card-items">
13             <!-- 
14             
15             <div class="card-text">
16                 <p>{{product.name|upper}}</p>
17                 <p>{{product.details}}</p>
18                 <p>{{product.price}}</p>
19                 <p>{{product.rating}}</p>
20                 {% if product.category == 1 %}
21                 <p>Mobile</p>
22                 {% elif product.category == 2 %}
23                 <p>Clothes</p>
24                 {% else %}
25                 <p>Shoes</p>
26                 {% endif %}
27                 <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
28                 <button id="buy_now_btn"><a href="">Buy Now</a></button>
29             </div>
30         </div>
31     </div>
32 </body></html>
```

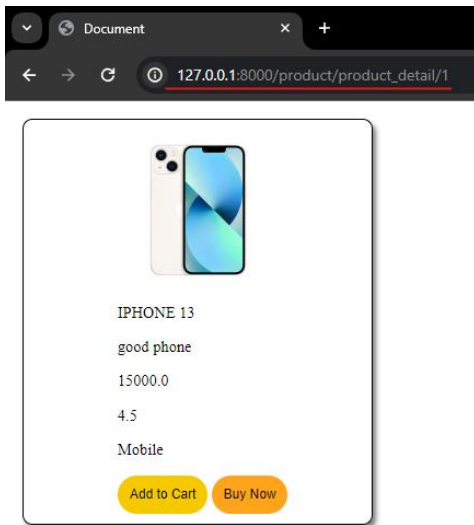
c. Create url for above view

```
urlpatterns = [
    path('index/', views.index),
    path('filter/<category_value>', views.filter_by_category),
    path('sort/<sort_value>', views.sort_by_price),
    path('rating/<rating_value>', views.filter_by_rating),
    path('price', views.filter_by_price_range),
    path('product_detail/<pid>', views.product_detail),
]
```

d. Use url in index.html on view more button :

```
<div class="card-text">
    <p>{{product.name|upper}}</p>
    <p>{{product.price}}</p>
    <p>{{product.rating}}</p>
    <button id="add_to_cart_btn"><a href="">Add to Cart</a></button>
    <button id="buy_now_btn"><a href="/product/product_detail/{{product.id}}">View More</a></button>
</div>
```

e. Check output :



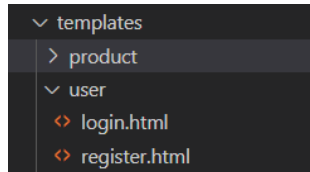
Tejas Kasare - 8459859415

8. Add to cart functionality :

To add product in the cart, we need user id. Since we havnt implemented user login function in this project ,so let's add that functionality

a. Register User

- i. Create user folder in templates folder and create register.html and login.html file in it



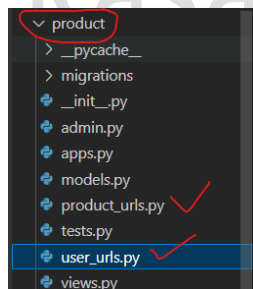
- ii. code for register.html (not image. You can copy paste)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <table align="center" border="1" cellpadding="5" cellspacing="5">
    <form method="POST">
      {% csrf_token %}
      <thead>
        <tr>
          <th colspan="2">
            Registration Form
            {% if error_msg %}
            <p style="color: red; font-weight: lighter;">{{error_msg}}</p>
            {% endif %}
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td><label for="username">UserName</label></td>
          <td><input type="text" id="username" name="username" value=""></td>
        </tr>
        <tr>
          <td><label for="password">Password</label></td>
          <td><input type="password" id="password" name="password" value=""></td>
        </tr>
        <tr>
          <td><label for="password2">Confirm Password</label></td>
          <td><input type="password" id="password2" name="password2" value=""></td>
        </tr>
        <tr>
          <td><input type="reset"></td>
          <td><input type="submit"></td>
        </tr>
        <tr>
          <td colspan="2">
            <p align="center">Alredy User? Click <a href="/user/login">here</a> to Login</p>
          </td>
        </tr>
      </tbody>
    </form>
  </table>
</body>
</html>
```

- iii. create view to show registration form and logic to register(I have used entire same logic same as we gave done in previous project number 8 : user login registration and session)

```
def register_user(request):
    data={}
    if request.method=="POST":
        uname=request.POST['username']
        upass=request.POST['password']
        uconf_pass=request.POST['password2']
        #implementing validation
        if (uname==' ' or upass =='' or uconf_pass ==''):
            data['error_msg']='Fields cant be empty'
            return render(request,'user/register.html',context=data)
        elif(upass!=uconf_pass):
            data['error_msg']='Password and confirm password does not matched'
            return render(request,'user/register.html',context=data)
        elif(User.objects.filter(username=uname).exists()):
            data['error_msg']=uname + ' already exist'
            return render(request,'user/register.html',context=data)
        else:
            user=User.objects.create(username=uname)
            #here username and password aee column names present inside auth_user table
            user.set_password(upass) #encrypting passowrd
            user.save() #saving data into table
            # return HttpResponse("Registraion done")
            return redirect('/user/login')
    return render(request,'user/register.html')
```

- iv. create url pattern for above view at application level
I have created user_urls.py (like product_urls.py) to manage user level urls



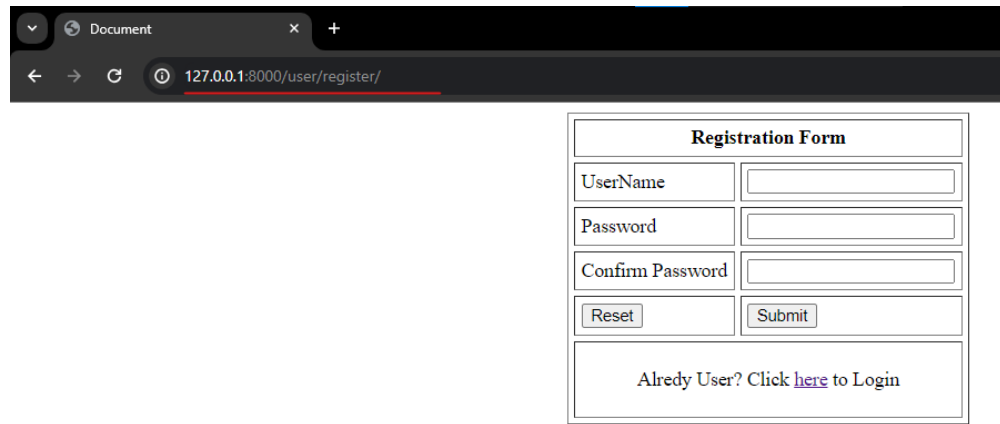
Register this user_urls.py in urls.py (project level)

```
urls.py
orm_and_frontend > urls.py > ...
21
22 urlpatterns = [
23     path('admin/', admin.site.urls),
24     path('product/', include('product.product_urls')),
25     path('user/', include('product.user_urls')),
26 ]
27
```

Finally, url to display above resgistration form

```
user_urls.py
product > user_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6
7 ]
```

Output



Document x +

127.0.0.1:8000/user/register/

Registration Form	
UserName	<input type="text"/>
Password	<input type="password"/>
Confirm Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>
Already User? Click here to Login	

b. login user:

i. Code for login.html (not image. You can copy paste)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <table align="center" border="1" cellpadding="5" cellspacing="5">
    <form method="POST">
      {% csrf_token %}
      <thead>
        <tr>
          <th colspan="2">
            Login Form
            {% if error_msg %}
            <p style="color: red; font-weight: lighter;">{{error_msg}}</p>
            {% endif %}
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td><label for="username">UserName</label></td>
          <td><input type="text" id="username" name="username" value=""></td>
        </tr>
        <tr>
          <td><label for="password">Password</label></td>
          <td><input type="password" id="password" name="password" value=""></td>
        </tr>
        <tr>
          <td><input type="reset"></td>
          <td><input type="submit"></td>
        </tr>
        <tr>
          <td colspan="2">
            <p align="center">New User? Click <a href="/user/register">here</a> to
            Register</p>
          </td>
        </tr>
      </tbody>
    </form>
  </table>
</body>
</html>
```

- ii. create view to show login form and logic to login(I have used entire same logic same as we gave done in previous project number 8 : user login registration and session)

```
def login_user(request):  
    data={}  
    if request.method=="POST":  
        uname=request.POST['username']  
        upass=request.POST['password']  
        #implementing validation  
        if (uname==' ' or upass == ''):  
            data['error_msg']='Fields cant be empty'  
            return render(request,'user/login.html',context=data)  
        elif(not User.objects.filter(username=uname).exists()):  
            data['error_msg']=uname + ' user is not registered'  
            return render(request,'user/login.html',context=data)  
        else:  
            #from django.contrib.auth import authenticate  
            user=authenticate(username=uname,password=upass)  
            print(user)  
            if user is not None:  
                login(request,user)  
                return redirect('/product/index')  
            else:  
                data['error_msg']='Wrong Password'  
                return render(request,'user/login.html',context=data)  
    return render(request,'user/login.html')
```

- iii. create url pattern for above view in user_urls.py

```
user_urls.py X  
product > user_urls.py > ...  
1 from django.urls import path  
2 from product import views  
3  
4 urlpatterns = [  
5     path('register/', views.register_user),  
6     path('login/', views.login_user),  
7 ]
```

Output



Login Form	
UserName	<input type="text"/>
Password	<input type="password"/>
<input type="button" value="Reset"/>	<input type="button" value="Submit"/>
New User? Click here to Register	

c. Logout

i. Create view for logout

```
views.py X
product > views.py > login_user
114
115 def user_logout(request):
116     logout(request)
117     return redirect('/product/index')
```

ii. Create url for above view

```
user_urls.py X
product > user_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6     path('login/', views.login_user),
7     path('logout/', views.user_logout),
8 ]
```

d. Create navbar to provide link for login, register and logout.

i. In our index.html, add following code

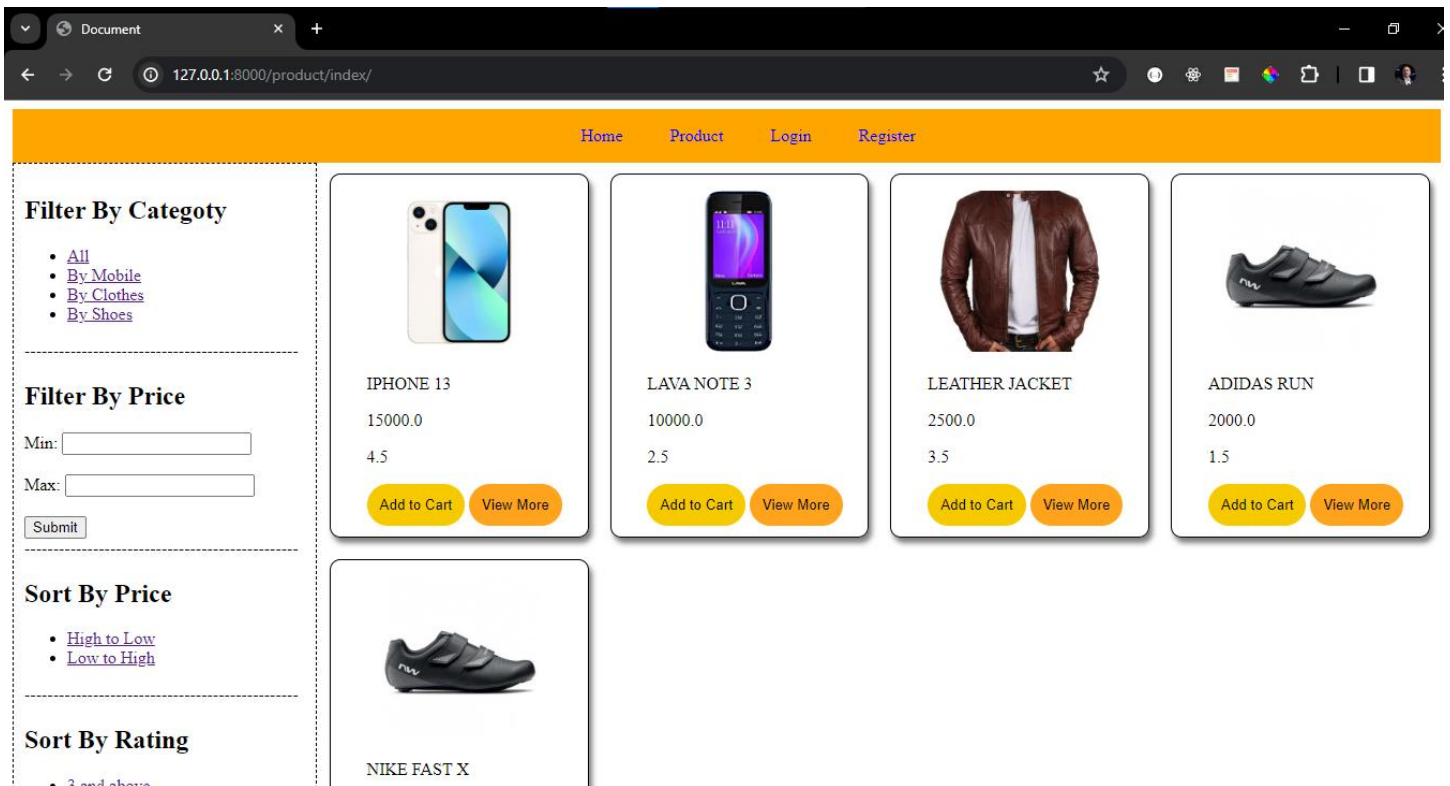
Inline css for navbar

```
index.html X
templates > product > index.html
6 <meta name="viewport" content="width=device-width" />
7 <title>Document</title>
8 <link rel="stylesheet" href="{% static 'css/product.css' %}" />
9 <style>
10     header {
11         width: 100%;
12         height: 50px;
13         background-color: orange;
14         display: flex;
15         align-items: center;
16         justify-content: space-around;
17     }
18     header * {
19         display: inline;
20     }
21     header li {
22         margin: 20px;
23     }
24     header li a {
25         color: blue;
26         text-decoration: none;
27     }
28 </style>
29 </head>
```

Navbar code

```
index.html X
templates > product > index.html
28 </style>
29 </head>
30 <body>
31     <div class="navbar">
32         <header>
33             <nav>
34                 <ul>
35                     <li> <a href="/product/index">Home</a> </li>
36                     <li> <a href="/product/index">Product</a> </li>
37
38                     {% if user.is_authenticated %}
39                     <li> <a href="/user/logout">Logout</a> </li>
40                     {% else %}
41                     <li> <a href="/user/login">Login</a> </li>
42                     <li> <a href="/user/register">Register</a> </li>
43                     {% endif %}
44                 </ul>
45             </nav>
46         </header>
47     </div>
48     <div class="container">
49         <div class="filter_area">
```

e. Final output



f. Finally, add to cart functionality :

i. Create model for cart

```
models.py
product > models.py > ...
1 from django.db import models
2 from django.contrib.auth.models import User
3
4 # Create your models here.
5 > class ProductTable(models.Model): ...
17
18 > class CartTable(models.Model):
19     #when you do mm without db_column attribute, you will get column name as uid_id not just uid.
20     #in uid_id, id is name of PK column of User model. if you dont want uid_id then just add
21     #db_column attribute
22     uid = models.ForeignKey(User, on_delete = models.CASCADE, db_column="uid")
23     pid = models.ForeignKey(ProductTable, on_delete = models.CASCADE, db_column="pid")
24
25
```

Don't forget to import User (above line 2)

ii. Register model in admin.py

```
admin.py
product > admin.py > ...
1 from django.contrib import admin
2 from product.models import ProductTable, CartTable
3
4 # Register your models here.
5 class ProductAdmin(admin.ModelAdmin):
6     list_display = ['id', 'name', 'price', 'details', '
7
8 admin.site.register(ProductTable, ProductAdmin)
9 admin.site.register(CartTable)
10
```

iii. MM

iv. View (basic functionality to check working)

```
views.py X
product > views.py > add_to_cart
118
119 def add_to_cart(request,pid):
120     if request.user.is_authenticated:
121         uid = request.user.id
122         print("user id = ",uid)
123         print("product id = ", pid)
124         #we cant pass only id in cart table, it is expecting object of User and Product
125         #therefore below line will gives error|
126         #cart=CartTable.objects.create(pid=pid,uid=uid)
127         user=User.objects.get(id=uid)
128         product=ProductTable.objects.get(id=pid)
129         cart=CartTable.objects.create(pid=product,uid=user)
130         cart.save()
131         return HttpResponseRedirect("product added to cart")
132     else:
133         return redirect("/user/login")
```

v. url

```
product_urls.py X
product > product_urls.py > ...
11     path('rating/<rating_value>', views.filter_by_rating),
12     path('price', views.filter_by_price_range),
13     path('product_detail/<pid>', views.product_detail),
14     path('add_to_cart/<pid>', views.add_to_cart),
15 ]
16 urlpatterns+=static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

- vi. use url – we need to use url on in 2 files –index.html and product_detail.html

```

98     <p>{{product.price}}</p>
99     <p>{{product.rating}}</p>
100    <button id="add_to_cart_btn"><a href="/product/add_to_cart/{{product.id}}">Add to Cart</a></button>
101    <button id="buy_now_btn"><a href="/product/product_detail/{{product.id}}">View More</a></button>
102  </div>

```

```

> product_detail.html
templates > product > product_detail.html
25     <p>Shoes</p>
26     {% endif %}
27     <button id="add_to_cart_btn"><a href="/product/add_to_cart/{{product.id}}">Add to Cart</a></button>
28     <button id="buy_now_btn"><a href="">Buy Now</a></button>
29 </div>
30 </div>

```

DONEEE!!

9. View Cart Functionality

- Add view cart option in navbar – and only visible if user is logged in

```
index.html X
templates > product > index.html
33 <nav>
34 <ul>
35 <li> <a href="/product/index">Home</a> </li>
36 <li> <a href="/product/index">Product</a> </li>
37
38 {% if user.is_authenticated %}
39 <li> <a href="/user/logout">Logout</a> </li>
40 <li> <a href="">View cart</a> </li>
41 {% else %}
42 <li> <a href="/user/login">Login</a> </li>
43 <li> <a href="/user/register">Register</a> </li>
44 {% endif %}
45
46 </ul>
```

- Displaying number of products added in cart into navbar (cart total)

- Logic to fetch cart items for logged in user (in index()) and pass counter to the html

```
# Create your views here.
def index(request):
    data={}
    #ProductTable.objects.all() this will fetch non active product all
    fetched_products=ProductTable.objects.filter(is_active=True)
    data['products']=fetched_products
    #getting count of cart item for specific user
    user_id=request.user.id
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    count=id_specific_cartitems.count()
    data['cart_count']=count
    return render(request, 'product/index.html', context=data)
```

- Display counter in html (index.html)

```
index.html X
templates > product > index.html > html > body > div.container > div.filter_area > div.by_category > ul
35 <li> <a href="/product/index">Home</a> </li>
36 <li> <a href="/product/index">Product</a> </li>
37
38 {% if user.is_authenticated %}
39 <li> <a href="/user/logout">Logout</a> </li>
40 <li> <a href="">View cart <span style="background-color: white;">{{cart_count}}</span></a> </li>
41 {% else %}
42 <li> <a href="/user/login">Login</a> </li>
43 <li> <a href="/user/register">Register</a> </li>
44 {% endif %}
```

- Redirecting to index page after clicking on add to cart button

```
def add_to_cart(request, pid):
    if request.user.is_authenticated:
        uid = request.user.id
        print("user id = ", uid)
        print("product id = ", pid)
        #we cant pass only id in cart table, it is expecting ob
        #therefore below line will gives error
        #cart=CartTable.objects.create(pid=pid,uid=uid)
        user=User.objects.get(id=uid)
        product=ProductTable.objects.get(id=pid)
        cart=CartTable.objects.create(pid=product,uid=user)
        cart.save()
        return redirect('/product/index')
    else:
        return redirect("/user/login")
```

Output :



c. Displaying cart items on cart.html page

i. Create cart.html in templates> product> cart.html

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN" crossorigin="anonymous">
  </head>
  <body>
    <div class="container mt-5">
      <div class="row">
        <div class="col-8">
          <h1>Shopping Cart</h1>
          <hr>
          {% for product in products %}
          <div class="row">
            <div class="col-4">
              
            </div>

            <div class="col-8">
              <h1>{{product.pid.name}}</h1>
              <p>{{product.pid.details}}</p>
              <p><span>&#8377;</span> {{product.pid.price}}</p>
              <p>{{product.pid.rating}}</p>
              <a class="btn btn-danger">-</a>
              <input class="text-center" type="number" value="1" disabled >
              <a class="btn btn-success">+</a>
              <a class="btn btn-danger">Delete</a>
            </div>
            <hr class="mt-3">
          </div>
          {% endfor %}
        </div>
        <div class="col-4 border">
          <h1>Hi, {{user.username|title}}</h1>
          <h3>Total Items: 3</h3>
          <h3>Total Price: <span>&#8377;</span> 5000</h3>
          <a class="btn btn-warning w-100" href="">Proceed to Buy</a>
        </div>
      </div>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js" integrity="sha384-
    C6RzsynM9kWDrmNeT87bh950GNYZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL" crossorigin="anonymous"></script>
  </body>
</html>
```

ii. Create view to display above cart.html and fetch cart product based on user id

```
def view_cart(request):
    data = {}
    user_id=request.user.id
    user=User.objects.get(id = user_id)
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    data['products']=id_specific_cartitems
    data['user']=user
    return render(request,'product/cart.html',context=data)
```

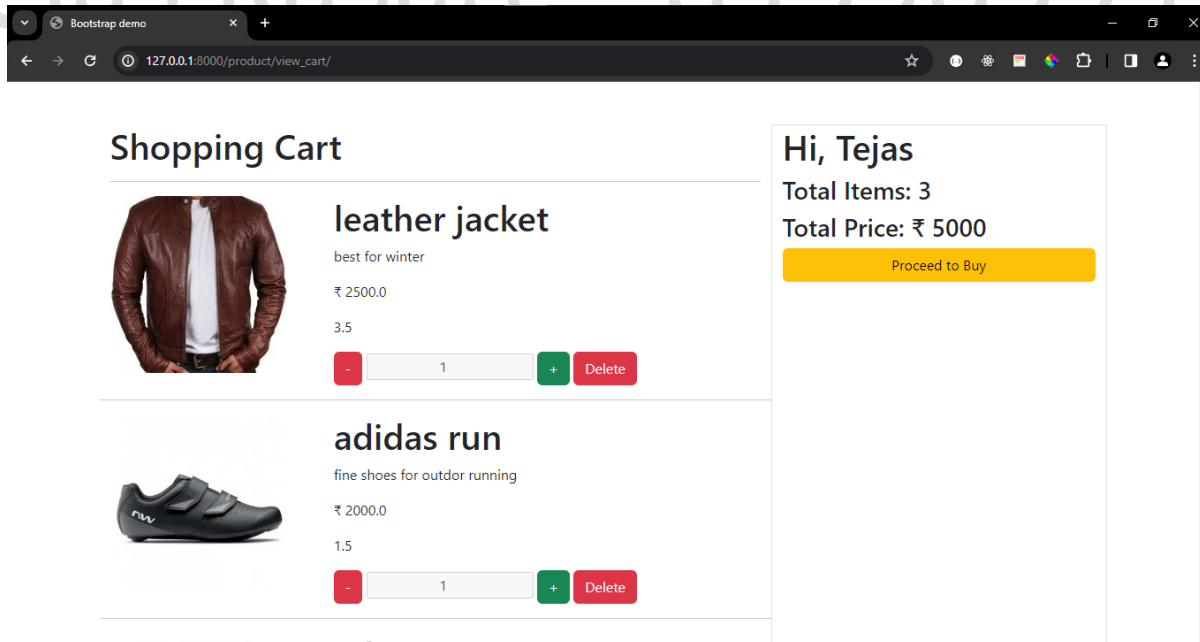
iii. Create url for above view

```
product_urls.py X
product > product_urls.py > ...
11 path('rating/<rating_value>', views.filter_by_rating),
12 path('price', views.filter_by_price_range),
13 path('product_detail/<pid>', views.product_detail),
14 path('add_to_cart/<pid>', views.add_to_cart),
15 path('view_cart/', views.view_cart),
16 ]
17 urlpatterns+=static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

iv. Use above url in index.html on view cart button

```
index.html X
templates > product > index.html
37
38 {% if user.is_authenticated %}
39 <li> <a href="/user/logout">Logout</a> </li>
40 <li> <a href="/product/view_cart">View cart <span style="background-color: white;">{{cart_count}}</span></a></li>
41 {% else %}
```

OutPut :



10. Remove item from cart functionality

We are going to remove cart item based on cart id

- Create logic to delete cart item based on cart id

```
def remove_item(request, cartid):  
    cart=CartTable.objects.get(id=cartid)  
    cart.delete()  
    return redirect('/product/view_cart')
```

- Create url form above remove_item() view

```
product_urls.py X  
product > product_urls.py > ...  
15     path('view_cart/', views.view_cart),  
16     path('remove_item/<cartid>', views.remove_item),  
17 ]  
18 urlpatterns+=static(settings.MEDIA_URL,document_root=settings.
```

- Use above url on delete button in cart.html

```
cart.html X  
templates > product > cart.html  
24     <p><span>₹</span> {{product.pid.price}}</p>  
25     <p>{{product.pid.rating}}</p>  
26     <a class="btn btn-danger"></a>  
27     <input class="text-center" type="number" value="1" disabled >  
28     <a class="btn btn-success">+</a>  
29     <a class="btn btn-danger" href="/product/remove_item/{{product.id}}">Delete</a>  
30 </div>  
31 <hr class="mt-3">
```

In Cart Table we are having 3 columns –id, pid, uid where id is actually a cart id

11. Total items and total price calculation

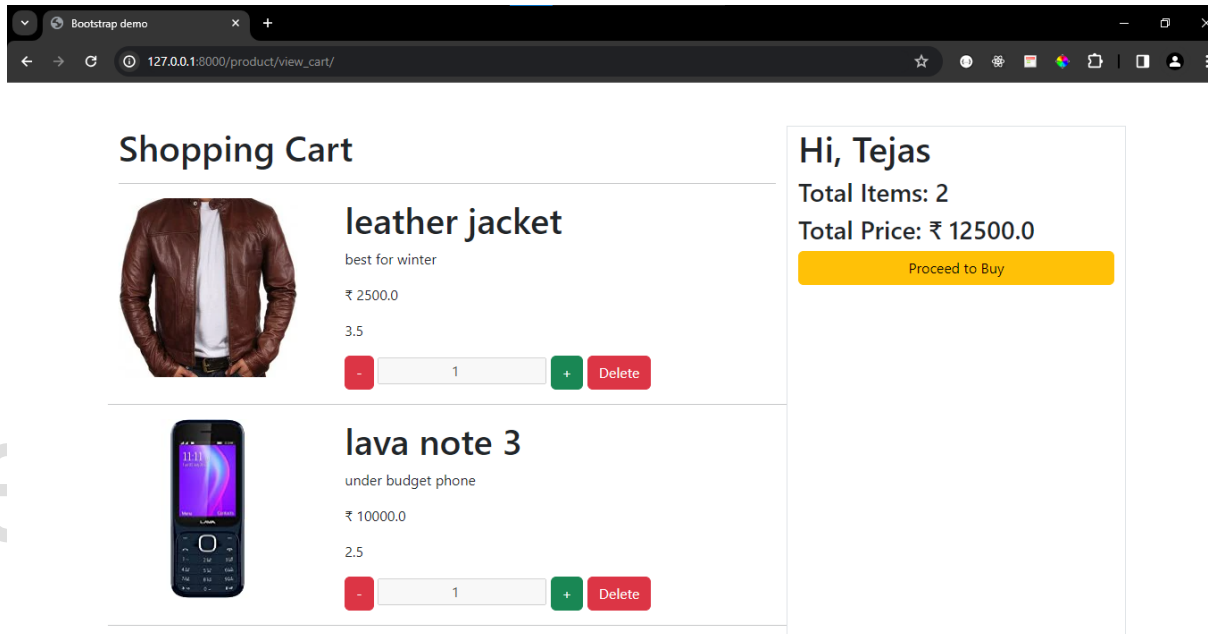
- We have to write a logic to calculate total price and total item in view_cart() function

```
def view_cart(request):  
    data = {}  
    user_id=request.user.id  
    user=User.objects.get(id = user_id)  
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)  
    data['products']=id_specific_cartitems  
    data['user']=user  
    #counting total items in cart  
    count=id_specific_cartitems.count()  
    data['cart_count']=count  
    #counting total price of cart  
    total_price = 0  
    for item in id_specific_cartitems:  
        #print(item.pid.price)  
        total_price+=item.pid.price  
    data['total_price']=total_price  
    return render(request, 'product/cart.html', context=data)
```

b. Displaying in html

```
cart.html x
templates > product > cart.html
34     </div>
35     <div class="col-4 border">
36         <h1>Hi, {{user.username|title}}</h1>
37         <h3>Total Items: {{cart_count}}</h3>
38         <h3>Total Price: <span>₹8377;</span> {{total_price}}</h3>
39         <a class="btn btn-warning w-100" href="">Proceed to Buy</a>
40     </div>
41 </div>
42 </div>
```

c. Output



12. product quantity functionality in the cart

- a. Since we are not having quantity column in cart table, lets add it from CartTable model (models.py) and setting default value to 1

```
models.py x
product > models.py > ...
17
18 class CartTable(models.Model):
19     #when you do mm without db_column attribute, you will get column name as uid_id r
20     #in uid_id, id is name of PK column of User model. if you dont want uid_id then
21     #db_column attribute
22     uid = models.ForeignKey(User,on_delete = models.CASCADE,db_column="uid")
23     pid= models.ForeignKey(ProductTable,on_delete = models.CASCADE,db_column="pid")
24     quantity = models.IntegerField(default=1)
25
```

b. MM

```
C:\TEJAS KASARE (Very imp folder)\my notes\Django Notes\orm_and_frontend>python manage.py makemigrations
Migrations for 'product':
  product\migrations\0006_carttable_quantity.py
    - Add field quantity to carttable

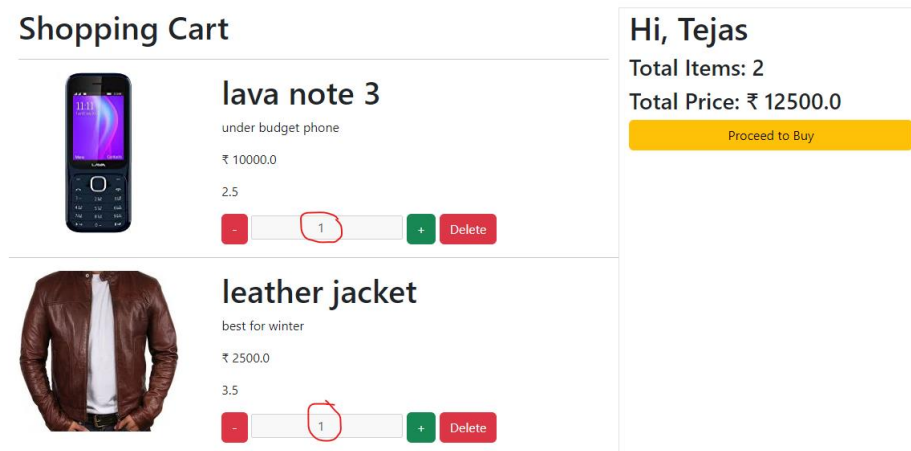
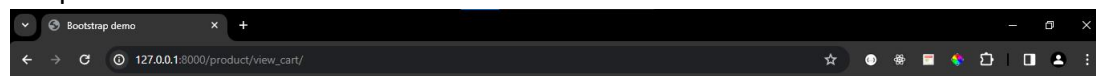
C:\TEJAS KASARE (Very imp folder)\my notes\Django Notes\orm_and_frontend>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, product, sessions
Running migrations:
  Applying product.0006_carttable_quantity... OK

C:\TEJAS KASARE (Very imp folder)\my notes\Django Notes\orm_and_frontend>_
```

c. Fetching quantity and displaying in cart.html

```
templates > product > cart.html
25 <!-- {{ product.id }} -->
26 <a class="btn btn-danger" -->-</a>
27 <input class="text-center" type="number" value="{{ product.quantity }}" disabled -->
28 <a class="btn btn-success" -->+</a>
29 <a class="btn btn-danger" href="/product/remove_item/{{ product.id }}" -->Delete</a>
30 </div>
31 <hr class="mt-3">
```

d. Output :



13. Product already in cart functionality

- In views.py import django messages as - from django.contrib import messages
- Update add_to_cart() and check whether that product already in cart

```

def add_to_cart(request,pid):
    if request.user.is_authenticated:
        uid = request.user.id
        print("user id = ",uid)
        print("product id = ", pid)
        #we cant pass only id in cart table, it is expecting object of User and Product
        #therefore below line will gives error
        #cart=CartTable.objects.create(pid=pid,uid=uid)
        user=User.objects.get(id=uid)
        product=ProductTable.objects.get(id=pid)

        q1 = Q(uid=uid)
        q2 = Q(pid=pid)
        available_products=CartTable.objects.filter(q1 & q2)
        print()
        if(available_products.count(>0):
            messages.error(request, 'Product is already in the cart.')
            return redirect('/product/index')
        else:
            cart=CartTable.objects.create(pid=product,uid=user)
            cart.save()
            messages.success(request,"Product is added to the cart.")
            return redirect('/product/index')
    else:
        return redirect("/user/login")

```

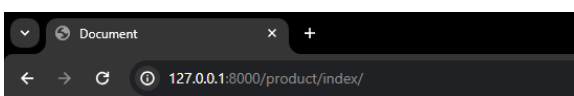
c. Display message in html

```

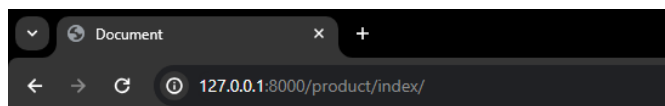
index.html x
templates > product > index.html > html > body > div.navbar > header >
30 </head>
31 <body>
32
33 {% if messages %}
34     {% for message in messages %}
35         {% if message.tags == "error" %}
36             <p style="color: red;">{{message}}</p>
37         {% else %}
38             <p style="color: green;">{{message}}</p>
39         {% endif %}
40     {% endfor %}
41 {% endif %}
42
43 <div class="navbar">
44     <header>

```

d. Output



Product is already in the cart.



Product is added to the cart.



14. Quantity inc/dec functionality

- a. Creating view: update_quantity() to change quantity

```
views.py X
product > views.py > update_quantity
178 def update_quantity(request, flag, cartid):
179     #print(type(flag))
180     cart=CartTable.objects.filter(id=cartid)
181     actual_quantity = cart[0].quantity
182     if(flag=="1"):
183         cart.update(quantity = actual_quantity+1)
184         pass
185     else:
186         if(actual_quantity>1):
187             cart.update(quantity = actual_quantity-1)
188             pass
189     return redirect('/product/view_cart')
```

Here flag is 1 to increment and 0 is for decrement

- b. Create url for above view

```
product_urls.py X
product > product_urls.py > ...
16 path('view_cart/', views.view_cart),
17 path('remove_item/<cartid>', views.remove_item),
18 path('update_quantity/<flag>/<cartid>', views.update_quantity),
19 ]
urlpatterns+=static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

- c. Use above url on plus(+) and minus(-) button in cart.html

```
cart.html X
templates > product > cart.html
23 <p>{{product.pid.details}}</p>
24 <p><span>&#8377;</span> {{product.pid.price}}</p>
25 <p>{{product.pid.rating}}</p>
26 <a class="btn btn-danger" href="/product/update_quantity/0/{{product.id}}">-</a>
27 <input class="text-center" type="number" value="{{product.quantity}}" disabled >
28 <a class="btn btn-success" href="/product/update_quantity/1/{{product.id}}">+</a>
29 <a class="btn btn-danger" href="/product/remove_item/{{product.id}}">Delete</a>
30 </div>
31 <hr class="mt-3">
```

- d.

15. Changing item count and total price according to quantity

```
def view_cart(request):
    data = {}
    user_id=request.user.id
    user=User.objects.get(id = user_id)
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    data['products']=id_specific_cartitems
    data['user']=user
    #counting total items in cart
    count=id_specific_cartitems.count()
    #data['cart_count']=count
    #counting total price of cart
    total_price = 0
    total_quantity=0
    for item in id_specific_cartitems:
        #print(item.pid.price)
        #total_price+=item.pid.price
        total_price=(total_price+item.pid.price)*(item.quantity)
        total_quantity+=item.quantity
    data['total_price']=total_price
    data['cart_count']=total_quantity
    return render(request, 'product/cart.html', context=data)
```

16. Creating place order functionality

a. We have to create OrderTable class in models.py

```
models.py X
product > models.py > ...
26 class OrderTable(models.Model):
27     order_id=models.CharField(max_length=50)
28     uid = models.ForeignKey(User,on_delete = models.CASCADE,db_column="uid")
29     pid= models.ForeignKey(ProductTable,on_delete = models.CASCADE,db_column="pid")
30     quantity = models.IntegerField()
31
```

b. MM

c. Create view place_order()

```
def place_order(request):
    data = {}
    user_id=request.user.id
    user=User.objects.get(id = user_id)
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    data['products']=id_specific_cartitems
    data['user']=user
    total_price = 0
    total_quantity=0
    for item in id_specific_cartitems:
        total_price=(total_price+item.pid.price)*(item.quantity)
        total_quantity+=item.quantity
    data['total_price']=total_price
    data['cart_count']=total_quantity
    return render(request, 'product/order.html', context=data)
```

our code for place order will be same as view cart since we are placing order for items present inside cart

- d. Create url for view

```
product_urls.py X
product > product_urls.py > ...
17 path('update_quantity/<+lag>/<cartid>', views.upda
18 path('place_order/', views.place_order),
19 ]
20 urlpatterns+=static(settings.MEDIA_URL, document_root=s
```

- e. Use url on place order button

- i. In our cart.html, we have proceed to pay button, change it to place order

```
cart.html X
templates > product > cart.html
36 <h1>Hi, {{user.username|title}}</h1>
37 <h3>Total Items: {{cart_count}}</h3>
38 <h3>Total Price: <span>&#8377;</span> {{total_price}}</h3>
39 <a class="btn btn-warning w-100" href="/product/place_order">Place Order</a>
40 </div>
41 </div>
42 /div>
43
```

- f. Creating order.html page to show order details

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Bootstrap demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet"
    integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwvyk2MPK8M2HN"
    crossorigin="anonymous">
  </head>
  <body>
    <div class="container mt-5">
      <div class="row">
        <div class="col-9">
          <h1>Order Summary</h1>
          <table class="table">
            <thead>
              <tr class="text-center">
                <th scope="col">IMAGE</th>
                <th scope="col">PRODUCT</th>
                <th scope="col">PRICE</th>
                <th scope="col">QUANTITY</th>
                <th scope="col">TOTAL</th>
              </tr>
            </thead>
            <tbody>
              {% for product in products %}
              <tr class="text-center">
                <th scope="row">
                  
                </th>
                <td><p>{{product.pid.name}}</p></td>
                <td><p><span>&#8377;</span> {{product.pid.price}}</p></td>
                <td><p>{{product.quantity}}</p></td>
                <td><p><span>&#8377;</span> {% widthratio product.quantity 1 product.pid.price
                %}</p></td>
                <!-- <td><p> {{product.quantity}}*{{product.pid.price}}</p></td> -->
              </tr>
              {% endfor %}
            </tbody>
          </table>
        </div>
        <div class="col-3 border">
          <h1>Hi, {{user.username|title}}</h1>
          <h5>Total Items: {{cart_count}}</h5>
          <h5>Total Price: <span>&#8377;</span> {{total_price}}</h5>
        </div>
      </div>
    </div>
  </body>
</html>
```

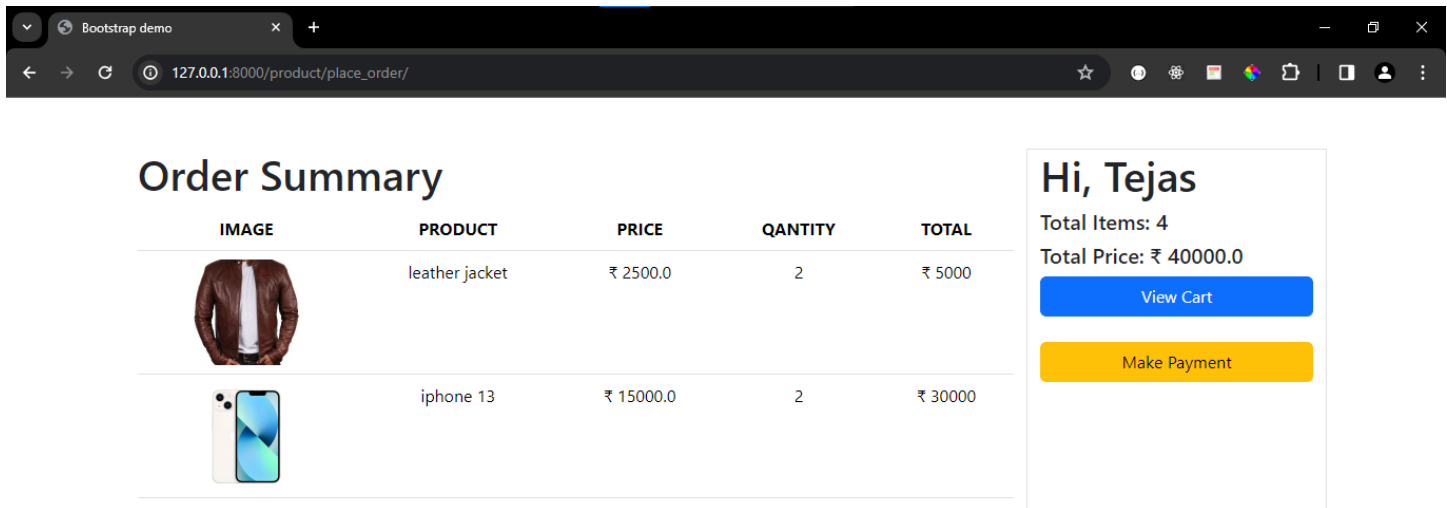
```

        <a class="btn btn-primary w-100" href="/product/view_cart">View Cart</a>
        <br><br>
        <a class="btn btn-warning w-100" href="/product/place_order">Make Payment</a>
    </div>
</div>
</div>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-C6RzsynM9kWDrmNeT87bh950GNYZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
</body>
</html>

```

g. Output :



17. Providing Edit profile functionality (to add address and other details of customer)

a. Create model for customer details

```

class CustomerDetails(models.Model):
    ADDRESS_TYPE = (('home', 'Home'), ('office', 'Office'), ('other', 'Other'))
    uid = models.ForeignKey(User, on_delete = models.CASCADE, db_column="uid")
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)
    phone = models.CharField(max_length=50)
    email = models.EmailField(max_length=50)
    address_type = models.CharField(max_length=10, choices=ADDRESS_TYPE)
    full_address = models.CharField(max_length=200)
    pincode = models.CharField(max_length=10)

```

b. MM

c. Provide Edit profile in navbar only for authenticated user

```

index.html
templates > product > index.html
49
50
51
52
53
54
    {% if user.is_authenticated %}
    <li> <a href="/user/logout">Logout</a> </li>
    <li> <a href="/product/view_cart">View cart <span style="background-color: wh
    <li> <a href="">Edit Profile </a></li>
    {% else %}

```

d. Add details into Customer table while registering user so we will get user id to be inserted into Customer

```

views.py X
product > views.py > register_user

71
72 def register_user(request):
73     data={}
74     if request.method=="POST":
75         uname=request.POST['username']
76         upass=request.POST['password']
77         uconf_pass=request.POST['password2']
78         #implementing validation
79         if (uname==' ' or upass =='' or uconf_pass ==''):
80             data['error_msg']='Fields cant be empty'
81             return render(request,'user/register.html',context=data)
82         elif(upass!=uconf_pass):
83             data['error_msg']='Password and confirm password does not matched'
84             return render(request,'user/register.html',context=data)
85         elif(User.objects.filter(username=uname).exists()):
86             data['error_msg']=uname + ' alredy exist'
87             return render(request,'user/register.html',context=data)
88         else:
89             user=User.objects.create(username=uname)
90             #here username and password aee column names present inside auth_user table
91             user.set_password(upass) #encrypting passowrd
92             user.save() #saving data into table
93             customer=CustomerDetails.objects.create(uid=user)
94             customer.save()
95             # return HttpResponse("Registraion done")
96             return redirect('/user/login')
97     return render(request,'user/register.html')
98

```

e. Create view to update customer details edit_profile()

```

from product.models import CustomerDetails
def edit_profile(request):
    data={}
    user_id=request.user.id
    customer_querySet=CustomerDetails.objects.filter(uid=user_id)
    customer = customer_querySet[0]
    data['customer']=customer
    if request.method=="POST":
        first_name=request.POST['first_name']
        last_name=request.POST['last_name']
        phone=request.POST['phone']
        email=request.POST['email']
        address_type=request.POST['address_type']
        full_address=request.POST['full_address']
        pincode=request.POST['pincode']
        print(first_name,last_name,phone,email,address_type,full_address,pincode)
        customer_querySet.update(first_name=first_name,last_name=last_name,phone=phone,
        email=email,address_type=address_type,full_address=full_address,pincode=pincode)
        return redirect('/product/index')
    return render(request,'user/edit_profile.html',context=data)

```

f. Create edit_profile.html (in templates folder > user folder)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <table align="center" border="1" cellpadding="5" cellspacing="5">
    <form method="POST">
      {% csrf_token %}
      <thead>
        <tr>
          <th colspan="2">
            Edit Profile
            {% if error_msg %}
            <p style="color: red; font-weight: lighter;">{{error_msg}}</p>
            {% endif %}
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td><label for="first_name">First Name</label></td>
          <td><input type="text" id="first_name" name="first_name" value="{{customer.uid}}"></td>
        </tr>
        <tr>
          <td><label for="last_name">Last Name</label></td>
          <td><input type="text" id="last_name" name="last_name" value="{{customer.last_name}}"></td>
        </tr>
        <tr>
          <td><label for="phone">Phone</label></td>
          <td><input type="tel" id="phone" name="phone" value="{{customer.phone}}"></td>
        </tr>
        <tr>
          <td><label for="email">Email</label></td>
          <td><input type="email" id="email" name="email" value="{{customer.email}}"></td>
        </tr>
        <tr>
          <td><label for="address_type">Type</label></td>
          <td>
            <select name="address_type" id="address_type">
              <option value="home">Home</option>
              <option value="office">Office</option>
              <option value="other">Other</option>
            </select>
          </td>
        </tr>
        <tr>
          <td><label for="full_address">Address</label></td>
          <td>
            <textarea name="full_address" id="full_address" cols="30" rows="5">
              {{customer.full_address}}
            </textarea>
          </td>
        </tr>
        <tr>
          <td><label for="pincode">Pincode</label></td>
          <td><input type="number" id="pincode" name="pincode" value="{{customer.pincode}}"></td>
        </tr>
        <tr>
          <td><input type="reset"></td>
          <td><input type="submit"></td>
        </tr>
      </tbody>
    </form>
  </table>
</body>
</html>
```

g. Create url for above view

```
user_urls.py X
product > user_urls.py > ...
1 from django.urls import path
2 from product import views
3
4 urlpatterns = [
5     path('register/', views.register_user),
6     path('edit_profile/', views.edit_profile),
7     path('login/', views.login_user),
8     path('logout/', views.user_logout),
9 ]
10
```

h. Use url in navbar Edit Profile button

```
index.html X
templates > product > index.html
47 </li> <a href="/product/index/home/">Home</a> </li>
48 <li> <a href="/product/index/">Product</a> </li>
49
50 {% if user.is_authenticated %}
51 <li> <a href="/user/logout/">Logout</a> </li>
52 <li> <a href="/product/view_cart/">View cart <span style="border: 1px solid black; padding: 2px 5px;">0</span></li>
53 <li> <a href="/user/edit_profile/">Edit Profile </a></li>
54 {% else %}
55 <li> <a href="/user/login/">Login</a> </li>
```

i.

Tejas Kasare - 8459859415