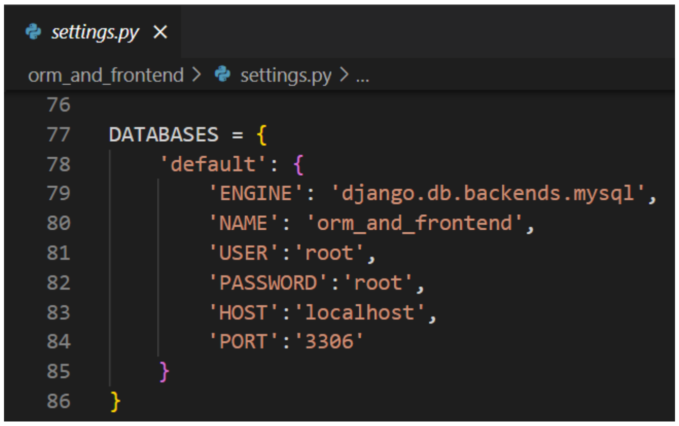
1. Create project and application
    a. Django>django-admin startproject orm_and_frontend
    b. Django>cd orm_and_frontend
    c. Django\orm_and_frontend>python manage.py startapp product
2. Open project in vs code
3. Register application in settings.py
4. Create database in workbench
   create database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

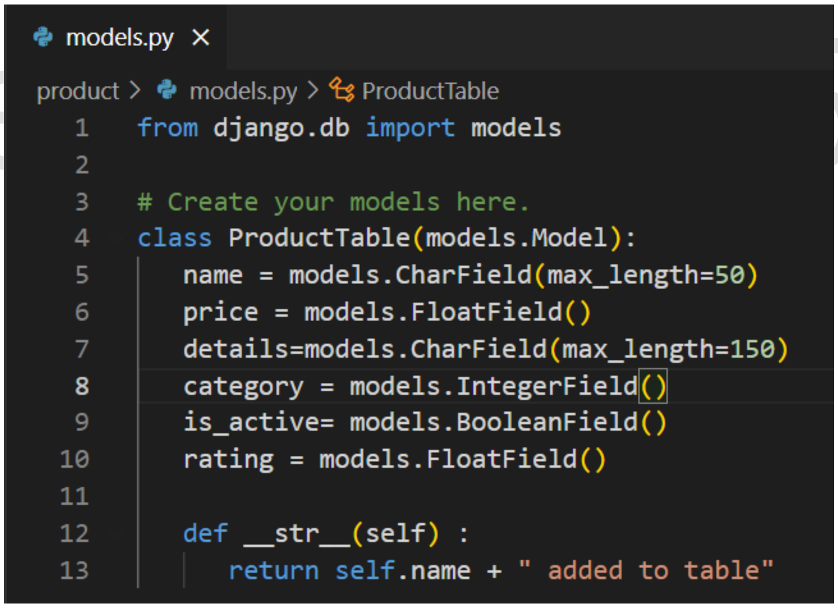


settings.py ×

orm_and_frontend > settings.py > ...

```python

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'orm_and_frontend',
        'USER':'root',
        'PASSWORD':'root',
        'HOST':'localhost',
        'PORT':'3306'
    }
}
```

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

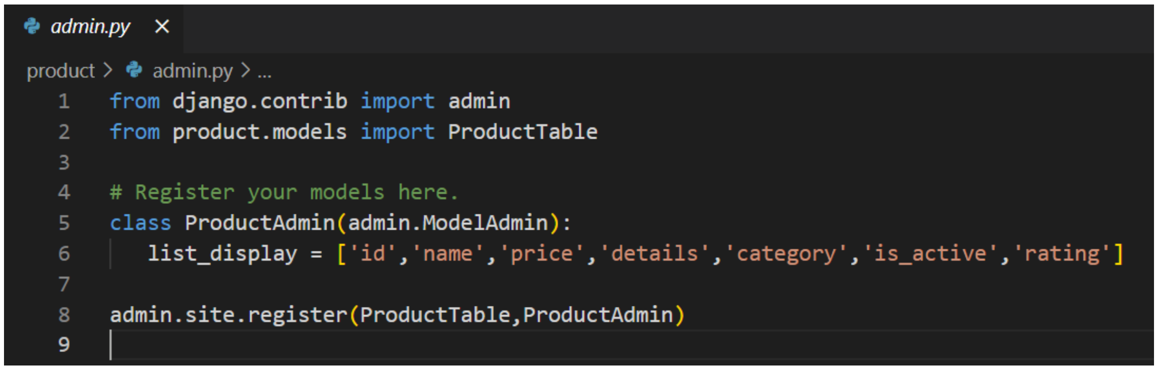


models.py ×

product > models.py > ProductTable

```python
from django.db import models

# Create your models here.
class ProductTable(models.Model):
    name = models.CharField(max_length=50)
    price = models.FloatField()
    details=models.CharField(max_length=150)
    category = models.IntegerField()
    is_active= models.BooleanField()
    rating = models.FloatField()

    def __str__(self) :
        return self.name + " added to table"
```

7. Register model class in admin.py



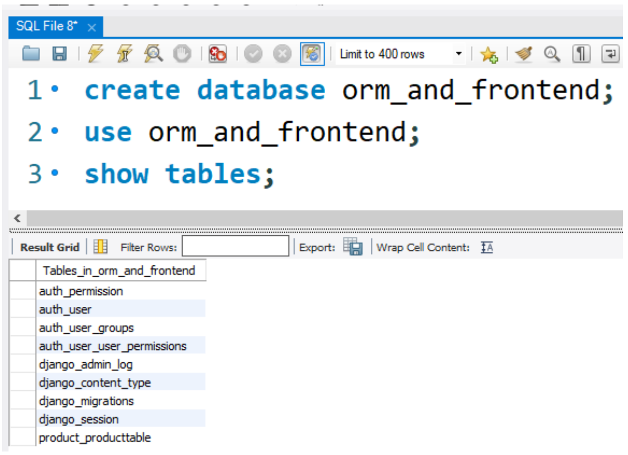
admin.py ×

product > admin.py > ...

```python
from django.contrib import admin
from product.models import ProductTable

# Register your models here.
class ProductAdmin(admin.ModelAdmin):
    list_display = ['id','name','price','details','category','is_active','rating']

admin.site.register(ProductTable,ProductAdmin)

```

Te 59859415

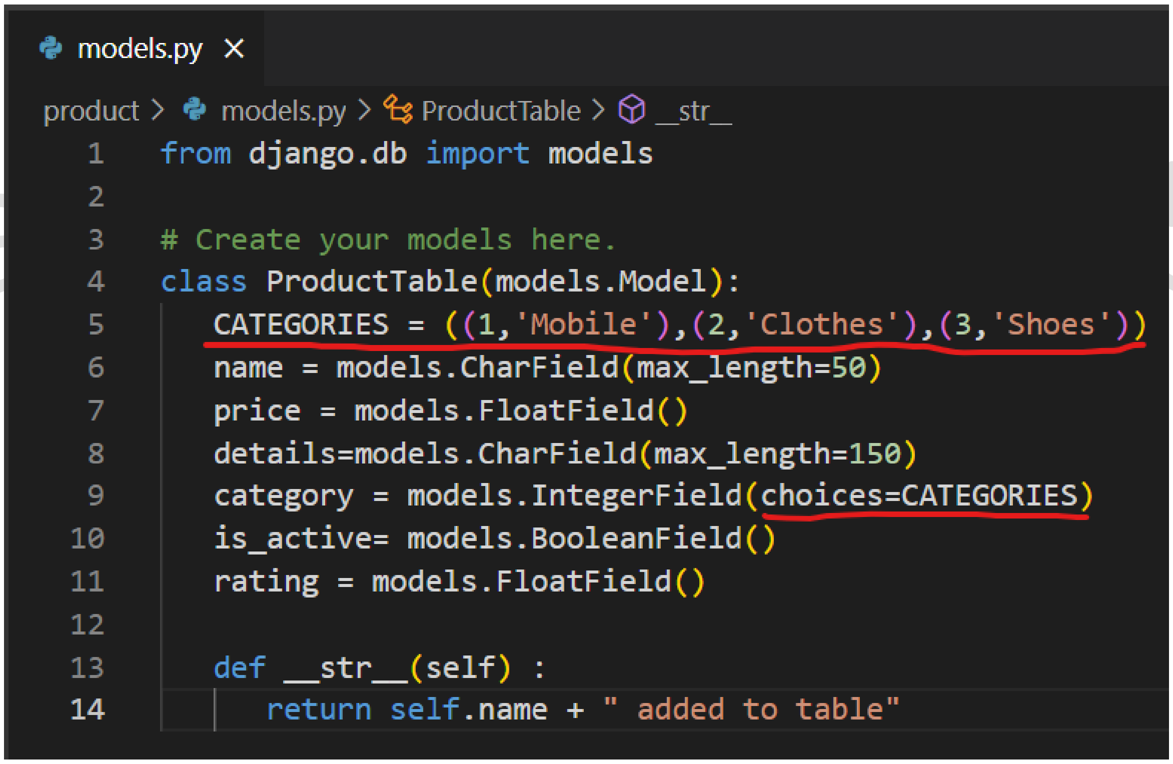
8. Makemigrations
    a. Django\orm_and_frontend>python manage.py makemigrations
9. Migrate
    a. Django\orm_and_frontend>python manage.py migrate
10. Check tables in orm_and_frontend database (in workbench)



```sql
create database orm_and_frontend;
use orm_and_frontend;
show tables;
```

Tables_in_orm_and_frontend
auth_permission
auth_user
auth_user_groups
auth_user_user_permissions
django_admin_log
django_content_type
django_migrations
django_session
product_producttable

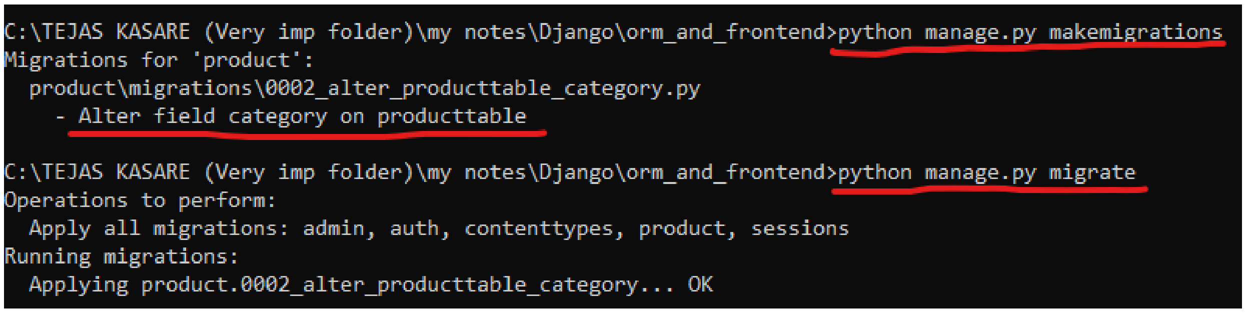
11. Providing categories and drop down option
    a. Make change in ProductTable model



product > models.py > ProductTable > __str__

```python
from django.db import models

# Create your models here.
class ProductTable(models.Model):
    CATEGORIES = ((1,'Mobile'),(2,'Clothes'),(3,'Shoes'))
    name = models.CharField(max_length=50)
    price = models.FloatField()
    details=models.CharField(max_length=150)
    category = models.IntegerField(choices=CATEGORIES)
    is_active= models.BooleanField()
    rating = models.FloatField()

    def __str__(self) :
        return self.name + " added to table"
```

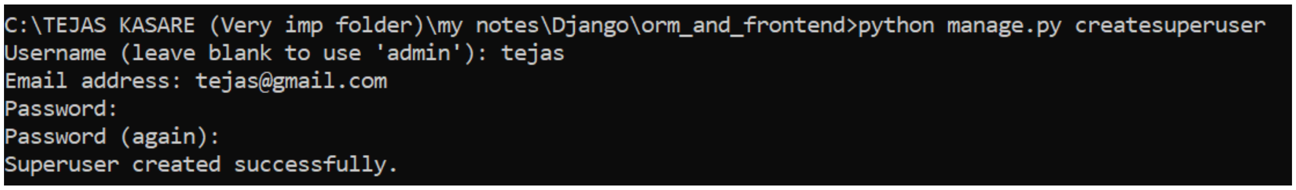
    b. Since we have made changes in model, we have to do makemigrations and migrate



```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py makemigrations
Migrations for 'product':
  product\migrations\0002_alter_producttable_category.py
    - Alter field category on producttable

C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, product, sessions
Running migrations:
  Applying product.0002_alter_producttable_category... OK
```

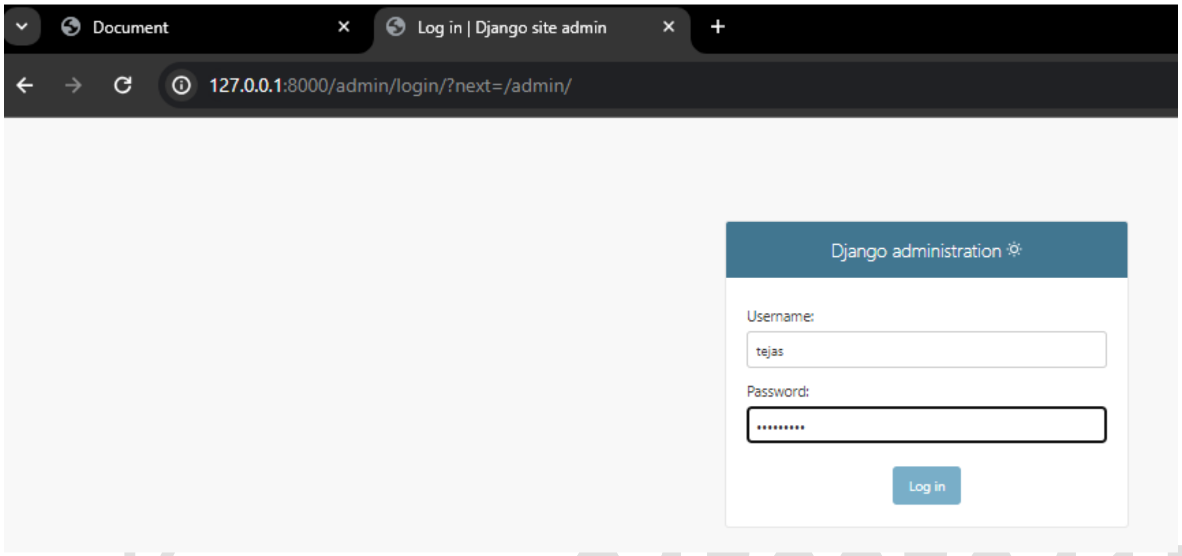
12. Adding some products into product table from admin panel

    a. Create superuser



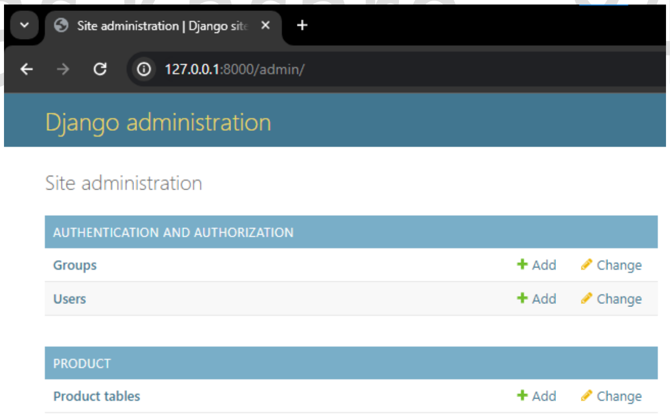
```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py createsuperuser
Username (leave blank to use 'admin'): tejas
Email address: tejas@gmail.com
Password:
Password (again):
Superuser created successfully.
```

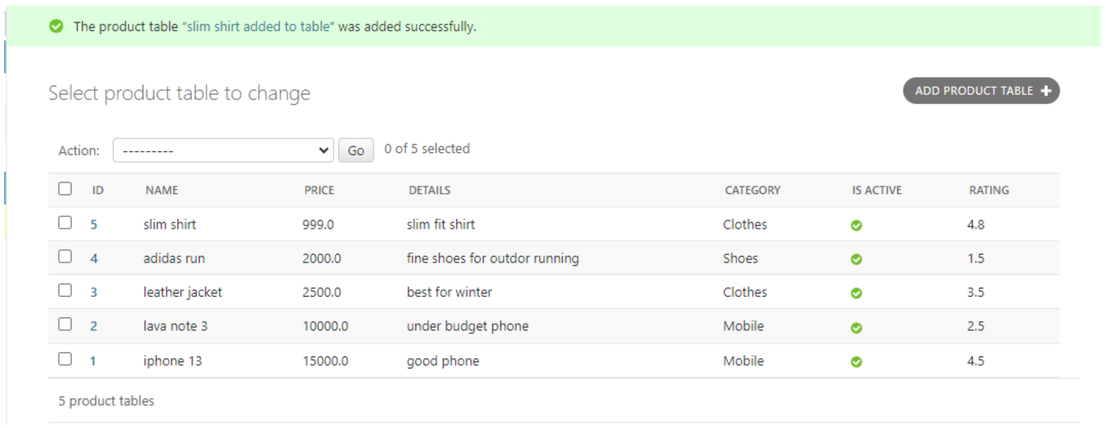
    b. runserver
    c. Login to admin panel


Document
Log in | Django site admin
127.0.0.1:8000/admin/login/?next=/admin/
Django administration
Username:
tejas
Password:
Log in

Tejas Kasare 8459859415

    d. Add 4-5 procducts (min 1 of each catergory)





The product table "slim shirt added to table" was added successfully.

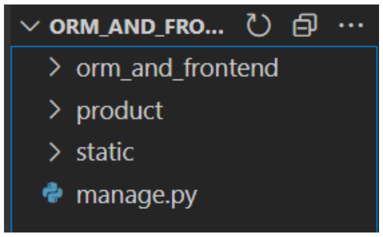
Select product table to change

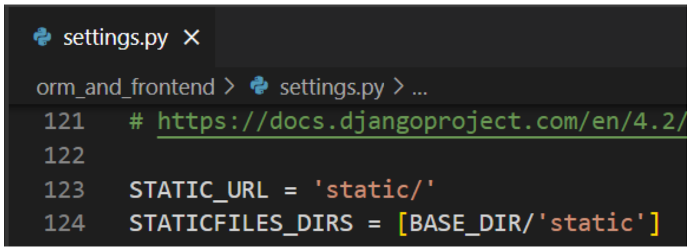
ADD PRODUCT TABLE

Action: --------- Go 0 of 5 selected

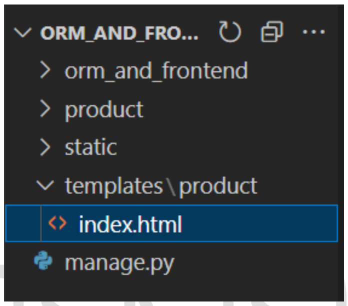
ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING
5	slim shirt	999.0	slim fit shirt	Clothes	✓	4.8
4	adidas run	2000.0	fine shoes for outdor running	Shoes	✓	1.5
3	leather jacket	2500.0	best for winter	Clothes	✓	3.5
2	lava note 3	10000.0	under budget phone	Mobile	✓	2.5
1	iphone 13	15000.0	good phone	Mobile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

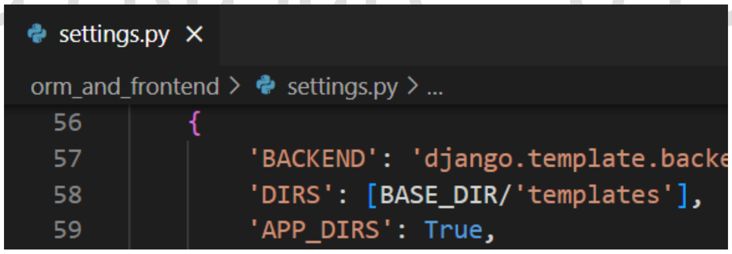
5 product tables

13. Creating static folder to store css file
    a. 

14. Register static folder in settings.py
    a. 

```
# https://docs.djangoproject.com/en/4.2/

STATIC_URL = 'static/'
STATICFILES_DIRS = [BASE_DIR/'static']
```

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

16. Register templates folder
    a. 

```
    {
        'BACKEND': 'django.template.backe
        'DIRS': [BASE_DIR/'templates'],
        'APP_DIRS': True,
```

17. In static folder do following :
    a. Create css folder > create product_style.css
    b. Create images folder > add one shirt image in it (200x200 size)
    c. 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;

}
```

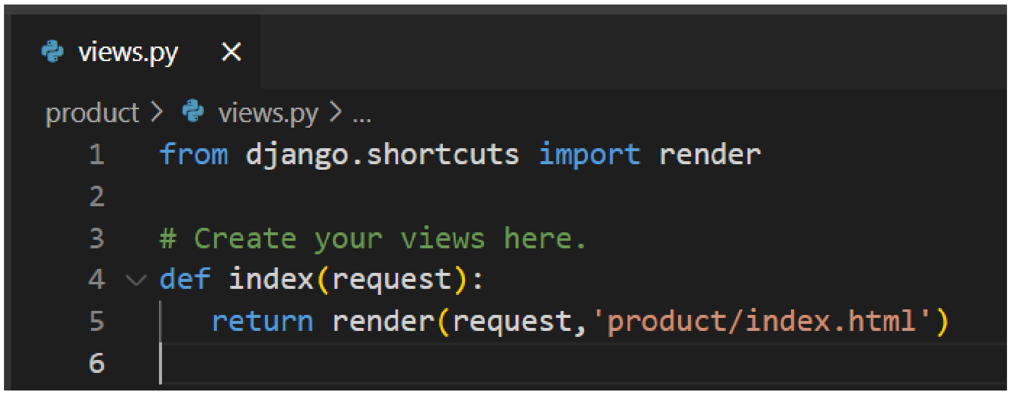
Tejas Kasare - 8459859415

```css
.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
    a. create product folder > create index.html
    b. add following code in it

```html
<!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 Categoty</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 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>
      <div class="product_area">
         <div class="card">
            <div class="card-items">
               <img src="{% static 'images/shirt.webp' %}" alt="">
               <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

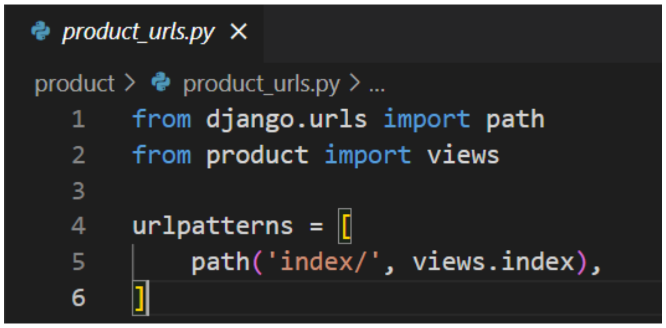


views.py ×

product > views.py > ...

```python
from django.shortcuts import render

# Create your views here.
def index(request):
    return render(request,'product/index.html')

```

20. create product_urls.py in product folder

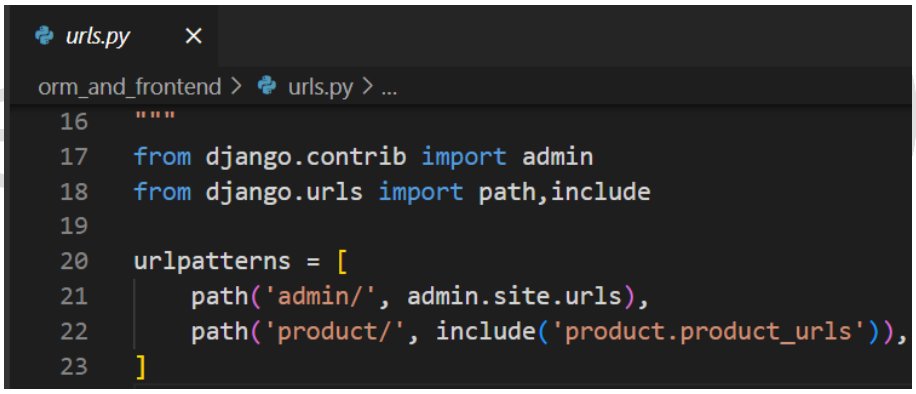


product_urls.py ×

product > product_urls.py > ...

```python
from django.urls import path
from product import views

urlpatterns = [
    path('index/', views.index),
]
```

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



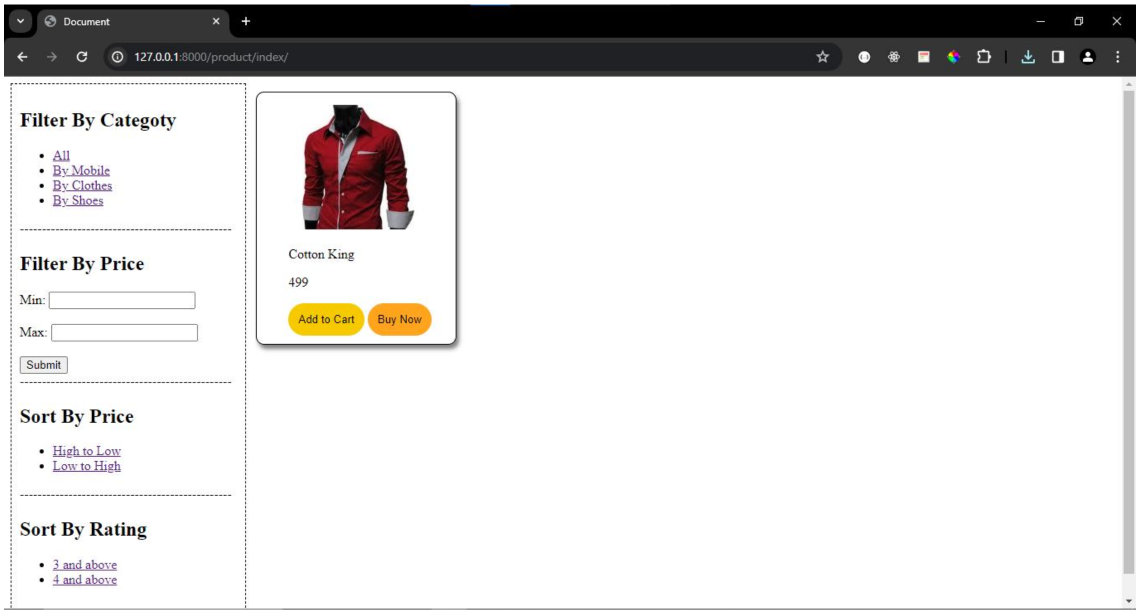
urls.py ×

orm_and_frontend > urls.py > ...

```python
"""
from django.contrib import admin
from django.urls import path,include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('product/', include('product.product_urls')),
]
```

Te 859415

22. runserver and check for product/index url



Document

127.0.0.1:8000/product/index/

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

Min:

Max:

Submit

Sort By Price

- High to Low
- Low to High

Sort By Rating

- 3 and above
- 4 and above

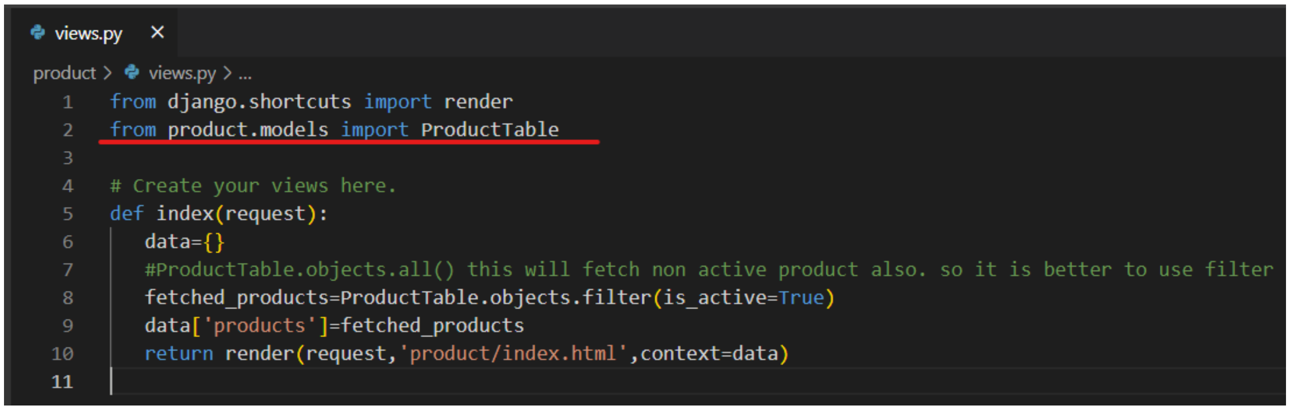
Cotton King

499

Add to Cart Buy Now

OPERATIONS

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

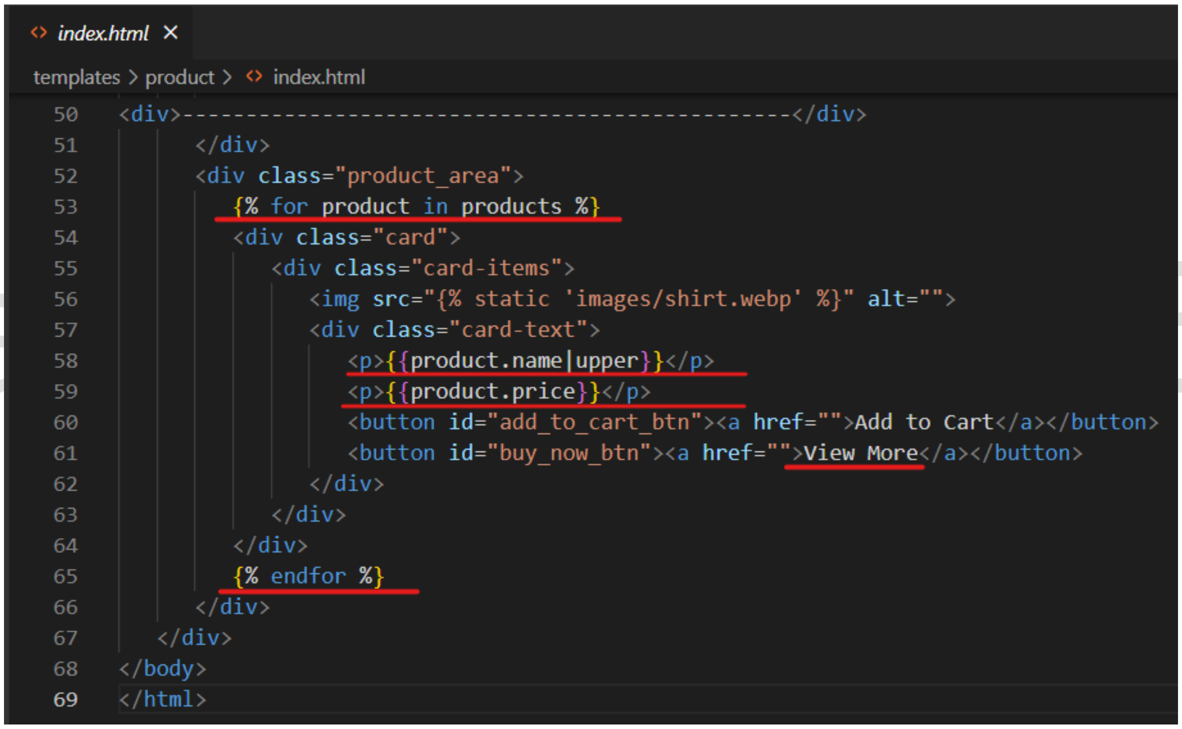


views.py ✕

product > views.py > ...

```python
from django.shortcuts import render
from product.models import ProductTable

# Create your views here.
def index(request):
    data={}
    #ProductTable.objects.all() this will fetch non active product also. so it is better to use filter
    fetched_products=ProductTable.objects.filter(is_active=True)
    data['products']=fetched_products
    return render(request,'product/index.html',context=data)

```

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



index.html ✕

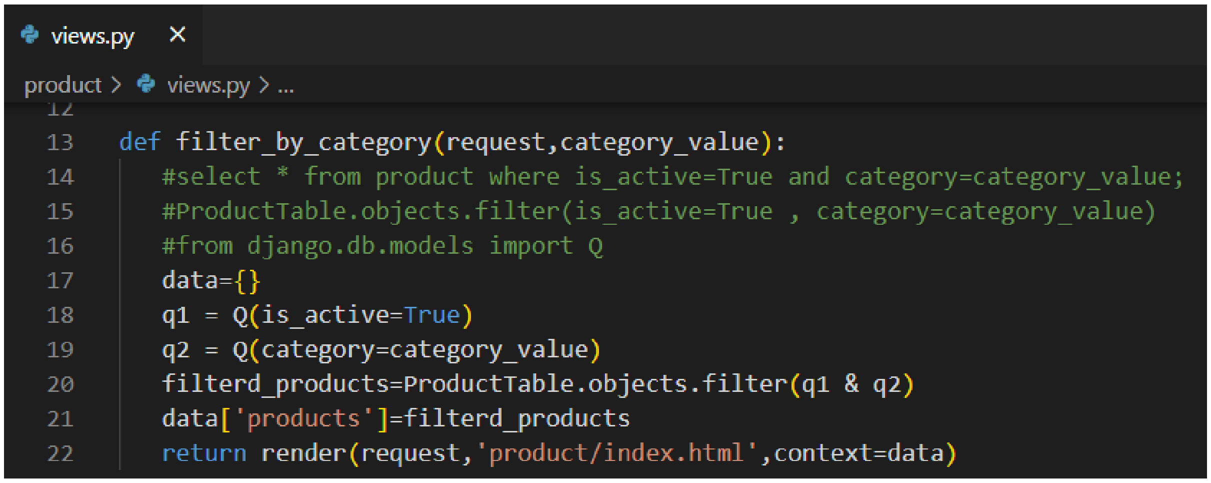
templates > product > index.html

```html
<div>-------------------------------------------------</div>
        </div>
        <div class="product_area">
          {% for product in products %}
          <div class="card">
            <div class="card-items">
              <img src="{% static 'images/shirt.webp' %}" alt="">
              <div class="card-text">
                <p>{{product.name|upper}}</p>
                <p>{{product.price}}</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>
            </div>
          </div>
          {% endfor %}
        </div>
    </div>
</body>
</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)

Teja

2. Implementing Filter by category logic
    a. Create view

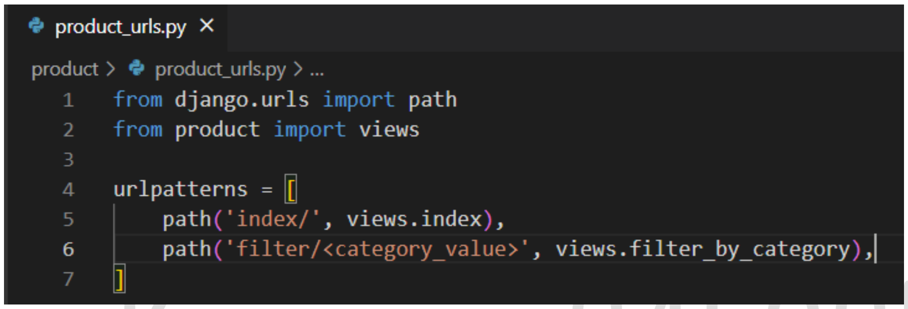


views.py

product > views.py > ...

```
def filter_by_category(request,category_value):
    #select * from product where is_active=True and category=category_value;
    #ProductTable.objects.filter(is_active=True , category=category_value)
    #from django.db.models import Q
    data={}
    q1 = Q(is_active=True)
    q2 = Q(category=category_value)
    filterd_products=ProductTable.objects.filter(q1 & q2)
    data['products']=filterd_products
    return render(request,'product/index.html',context=data)
```

    b. Create url for view



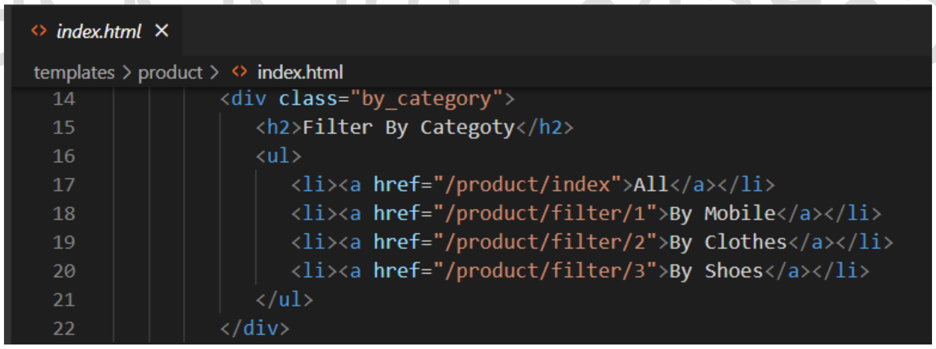
product_urls.py

product > product_urls.py > ...

```
from django.urls import path
from product import views

urlpatterns = [
    path('index/', views.index),
    path('filter/<category_value>', views.filter_by_category),
]
```

Tejas Kasare 8459859415

    c. Use url in index.html

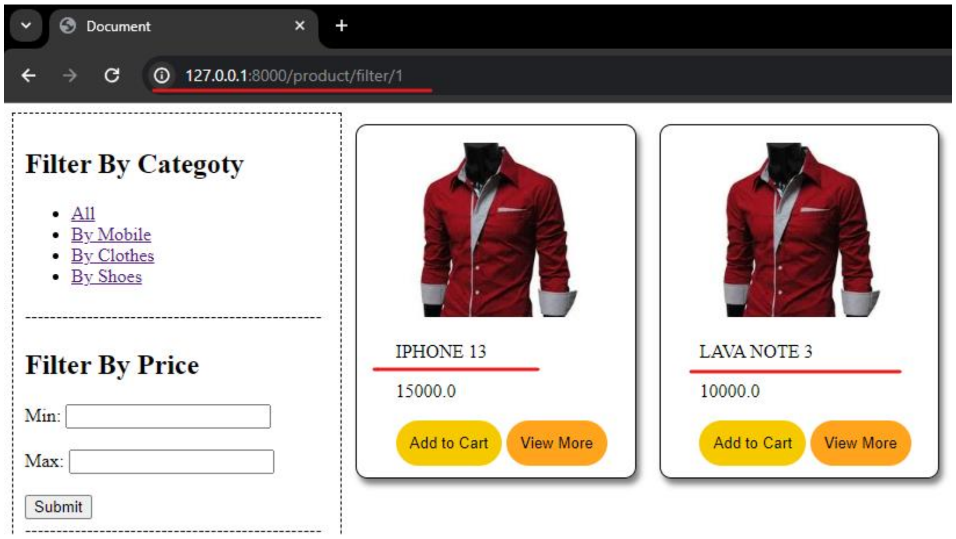


index.html

templates > product > index.html

```
            <div class="by_category">
                <h2>Filter By Categoty</h2>
                <ul>
                    <li><a href="/product/index">All</a></li>
                    <li><a href="/product/filter/1">By Mobile</a></li>
                    <li><a href="/product/filter/2">By Clothes</a></li>
                    <li><a href="/product/filter/3">By Shoes</a></li>
                </ul>
            </div>
```

Result :



Document

127.0.0.1:8000/product/filter/1

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

Min:

Max:

Submit

IPHONE 13

15000.0

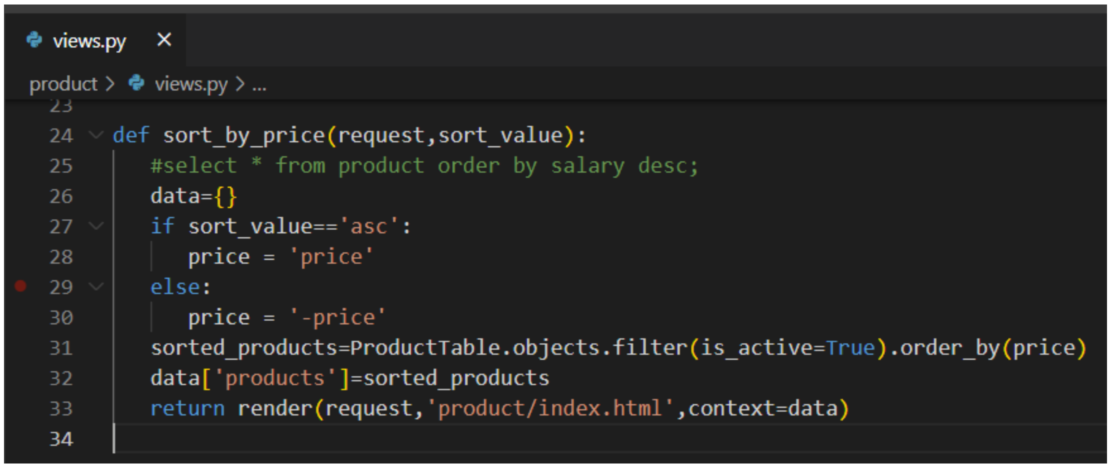
Add to Cart View More

LAVA NOTE 3

10000.0

Add to Cart View More

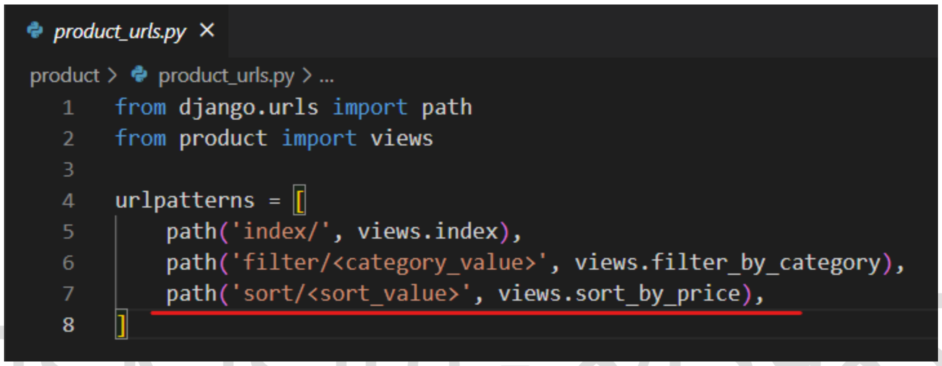
3. Implementing sorting logic (high to low and low to high)
   a. Create view



views.py
product > views.py > ...

```
def sort_by_price(request,sort_value):
    #select * from product order by salary desc;
    data={}
    if sort_value=='asc':
        price = 'price'
    else:
        price = '-price'
    sorted_products=ProductTable.objects.filter(is_active=True).order_by(price)
    data['products']=sorted_products
    return render(request,'product/index.html',context=data)
```

   b. Create url for view

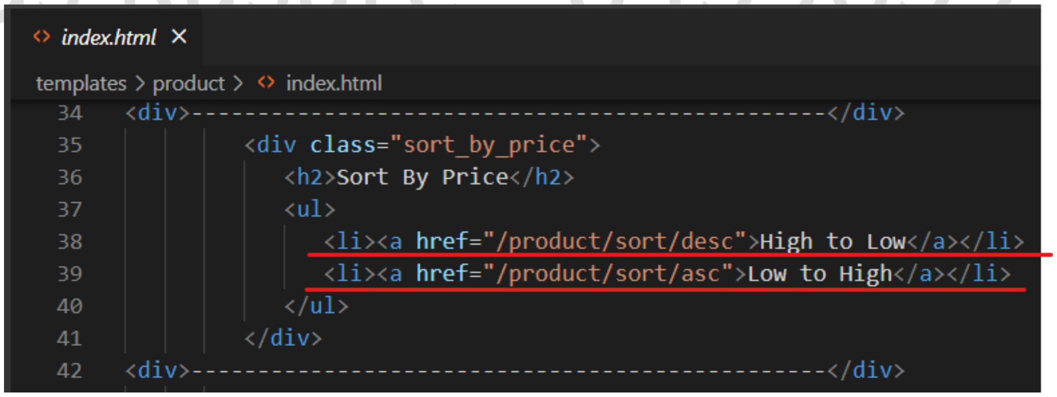


product_urls.py
product > product_urls.py > ...

```
from django.urls import path
from product import views

urlpatterns = [
    path('index/', views.index),
    path('filter/<category_value>', views.filter_by_category),
    path('sort/<sort_value>', views.sort_by_price),
]
```

Tejas Kasare - 8459859415

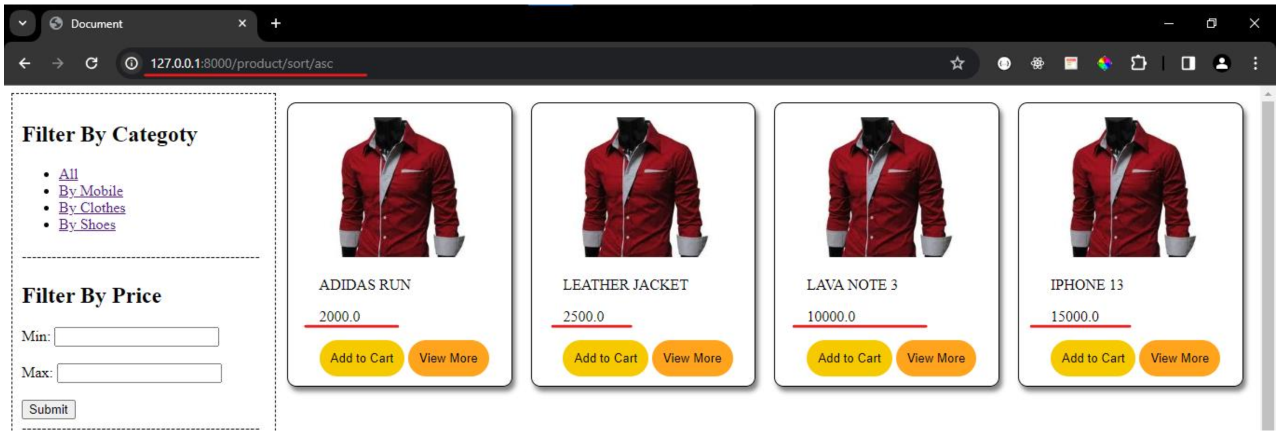
   c. Use url in index.html



index.html
templates > product > index.html

```
<div>-------------------------------------------------</div>
        <div class="sort_by_price">
            <h2>Sort By Price</h2>
            <ul>
                <li><a href="/product/sort/desc">High to Low</a></li>
                <li><a href="/product/sort/asc">Low to High</a></li>
            </ul>
        </div>
<div>-------------------------------------------------</div>
```

   d. Result



Document
127.0.0.1:8000/product/sort/asc

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

Min:
Max:
Submit

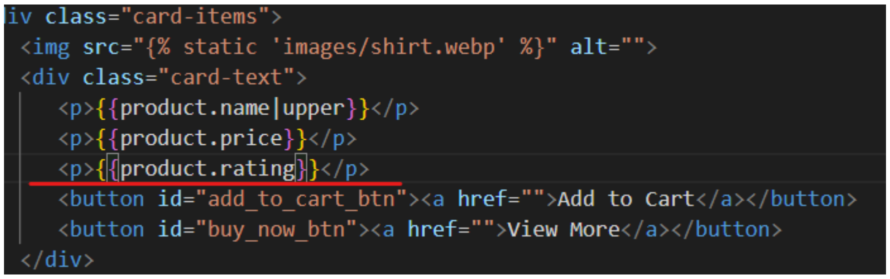
ADIDAS RUN
2000.0

LEATHER JACKET
2500.0

LAVA NOTE 3
10000.0

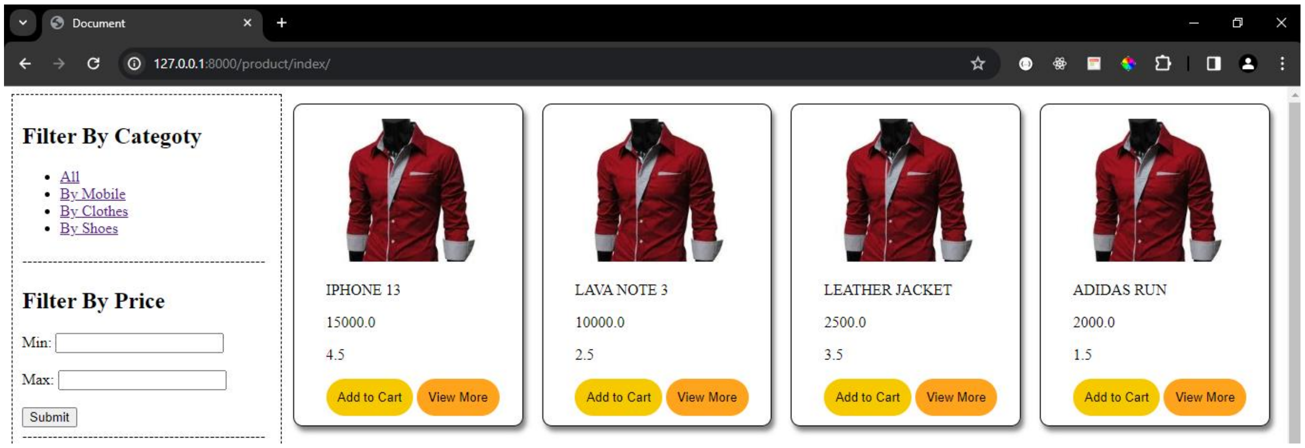
IPHONE 13
15000.0

4. Implementing filter by rating logic
   a. Since we are not displaying ratings in index.html, lets display first



```
iv class="card-items">
 <img src="{% static 'images/shirt.webp' %}" alt="">
 <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



Document

127.0.0.1:8000/product/index/

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

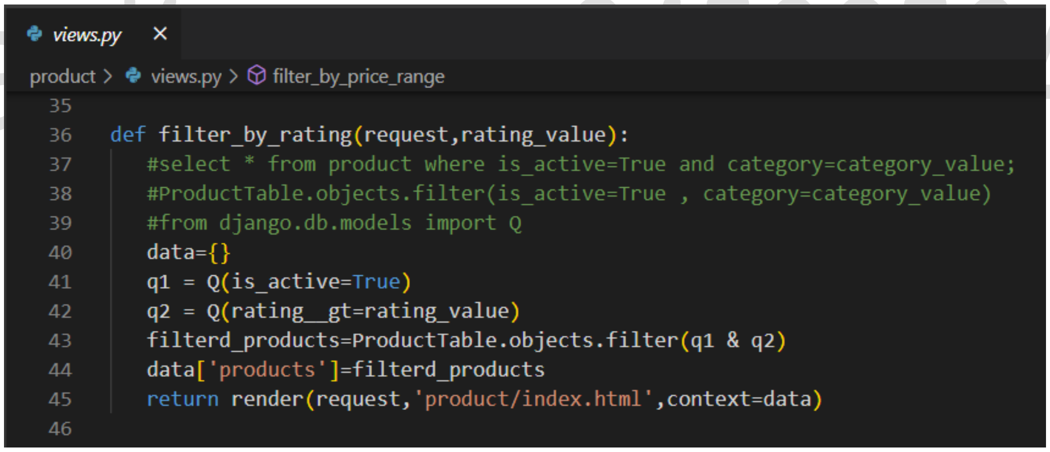
Min:

Max:

Submit

IPHONE 13	LAVA NOTE 3	LEATHER JACKET	ADIDAS RUN
15000.0	10000.0	2500.0	2000.0
4.5	2.5	3.5	1.5
Add to Cart View More	Add to Cart View More	Add to Cart View More	Add to Cart View More

   c. Create view

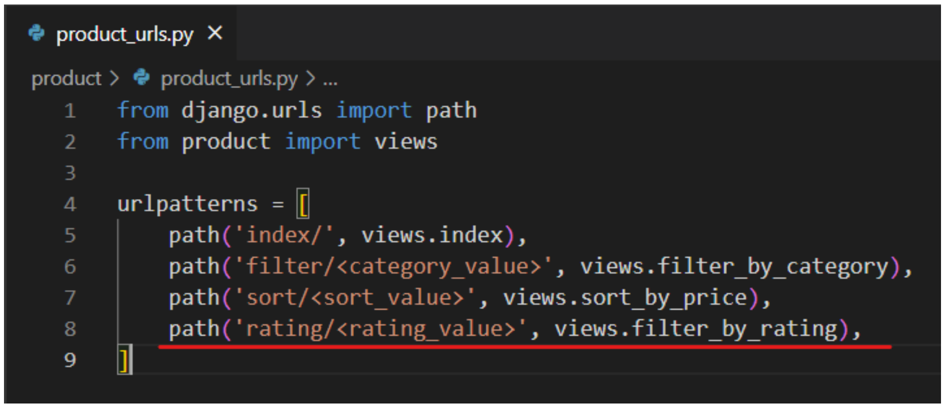


views.py

product > views.py > filter_by_price_range

```python

def filter_by_rating(request,rating_value):
    #select * from product where is_active=True and category=category_value;
    #ProductTable.objects.filter(is_active=True , category=category_value)
    #from django.db.models import Q
    data={}
    q1 = Q(is_active=True)
    q2 = Q(rating__gt=rating_value)
    filterd_products=ProductTable.objects.filter(q1 & q2)
    data['products']=filterd_products
    return render(request,'product/index.html',context=data)

```

   d. Create url for view



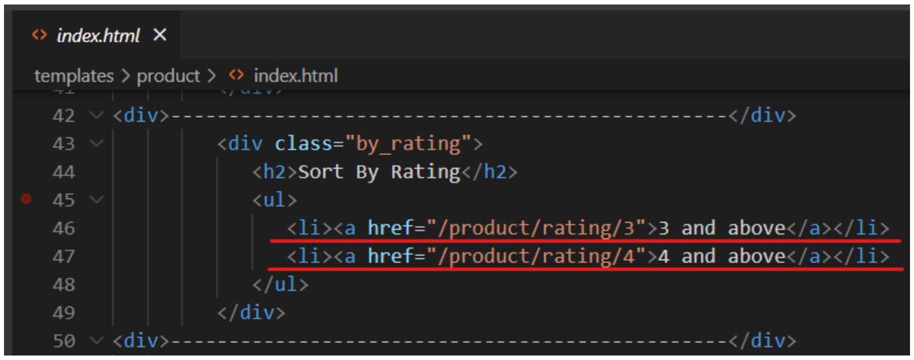
product_urls.py

product > product_urls.py > ...

```python
from django.urls import path
from product import views

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),
]
```

Teja 8459859415

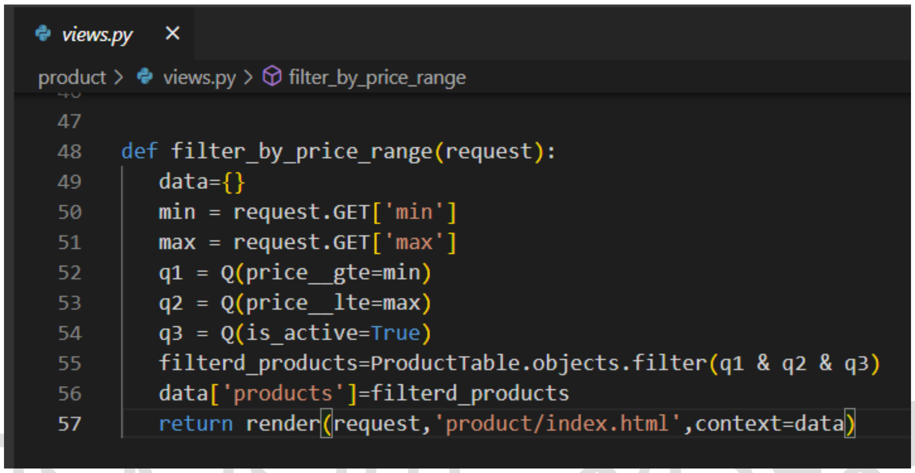
e. Use url in index.html



```html
<div>-----------------------------------------------</div>
        <div class="by_rating">
            <h2>Sort By Rating</h2>
            <ul>
                <li><a href="/product/rating/3">3 and above</a></li>
                <li><a href="/product/rating/4">4 and above</a></li>
            </ul>
        </div>
<div>-----------------------------------------------</div>
```

5. Filter by price range

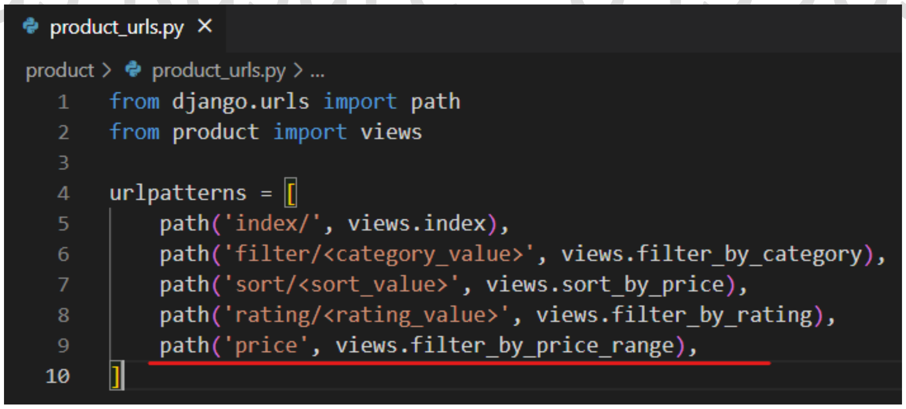
a. Create view



```python

def filter_by_price_range(request):
    data={}
    min = request.GET['min']
    max = request.GET['max']
    q1 = Q(price__gte=min)
    q2 = Q(price__lte=max)
    q3 = Q(is_active=True)
    filterd_products=ProductTable.objects.filter(q1 & q2 & q3)
    data['products']=filterd_products
    return render(request,'product/index.html',context=data)
```

Tejas Kasare - 8459859415

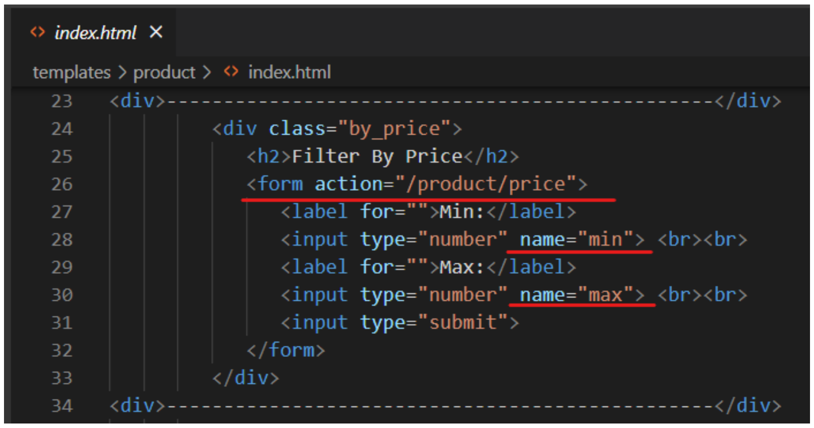
b. Create url for view



```python
from django.urls import path
from product import views

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),
]
```

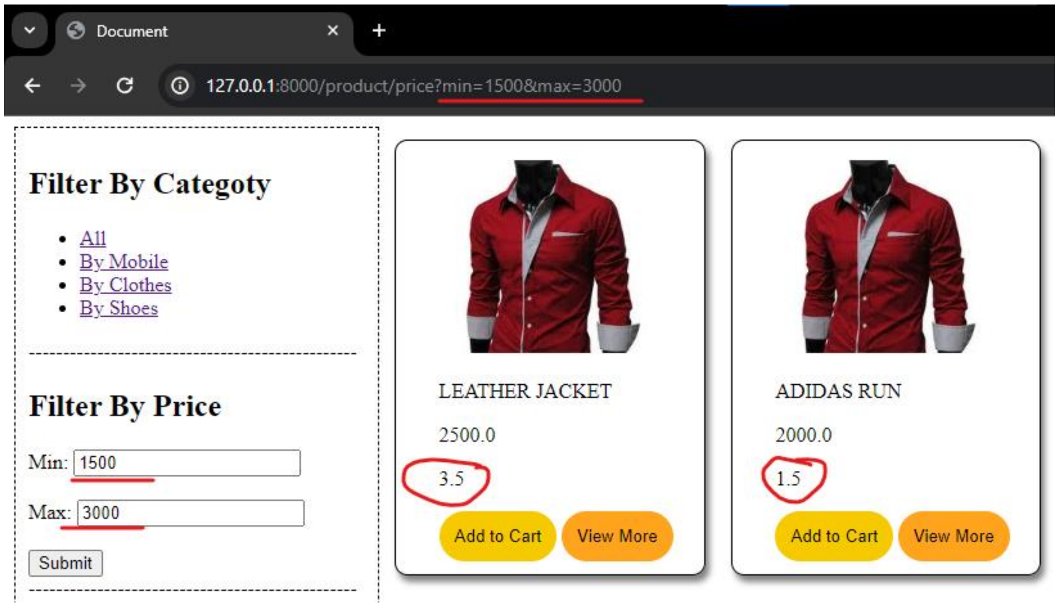
IMPORTANT : there is no / after price

c. User url in index.html

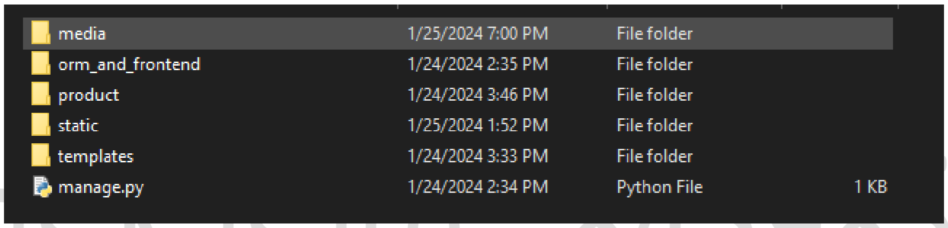


```html
<div>-----------------------------------------------</div>
        <div class="by_price">
            <h2>Filter By Price</h2>
            <form action="/product/price">
                <label for="">Min:</label>
                <input type="number" name="min"> <br><br>
                <label for="">Max:</label>
                <input type="number" name="max"> <br><br>
                <input type="submit">
            </form>
        </div>
<div>-----------------------------------------------</div>
```

Result :



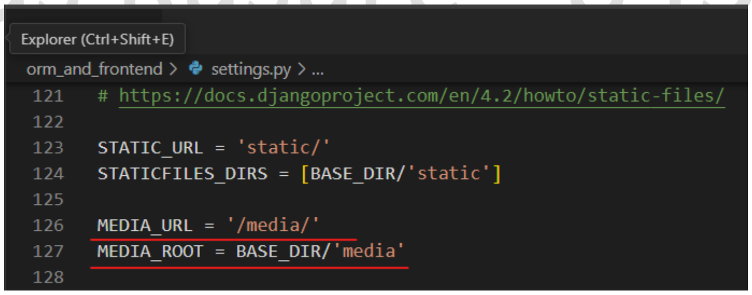
6. Uploading Image
   a. Create media project in project like we create templates folder



```
media            1/25/2024 7:00 PM   File folder
orm_and_frontend 1/24/2024 2:35 PM   File folder
product          1/24/2024 3:46 PM   File folder
static           1/25/2024 1:52 PM   File folder
templates        1/24/2024 3:33 PM   File folder
manage.py        1/24/2024 2:34 PM   Python File   1 KB
```

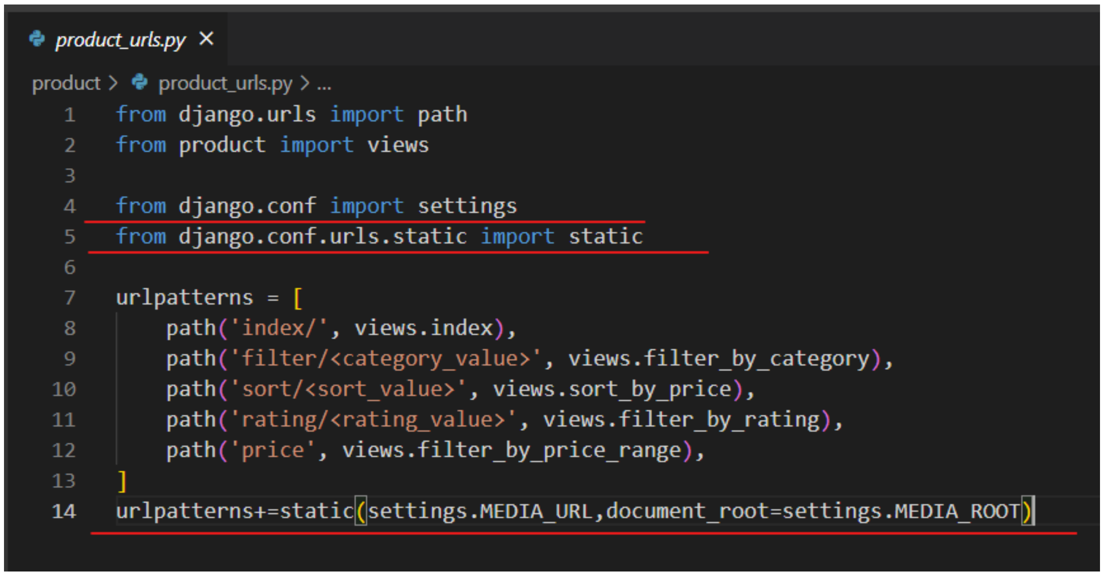
Tejas Kasare - 8459859415

   b. Register media folder in settings.py



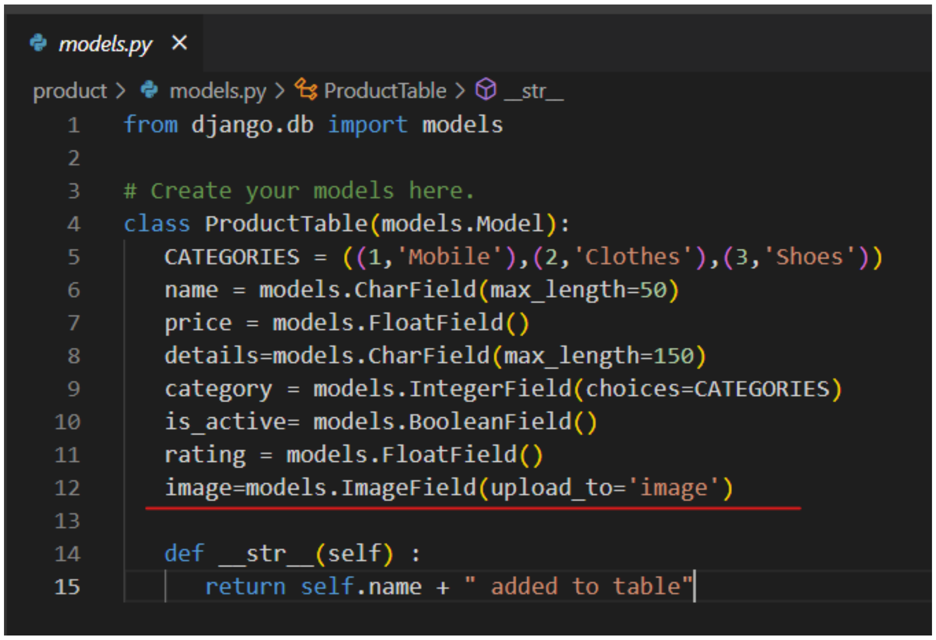
```python
# https://docs.djangoproject.com/en/4.2/howto/static-files/

STATIC_URL = 'static/'
STATICFILES_DIRS = [BASE_DIR/'static']

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR/'media'
```

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



```python
from django.urls import path
from product import views

from django.conf import settings
from django.conf.urls.static import static

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),
]
urlpatterns+=static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

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

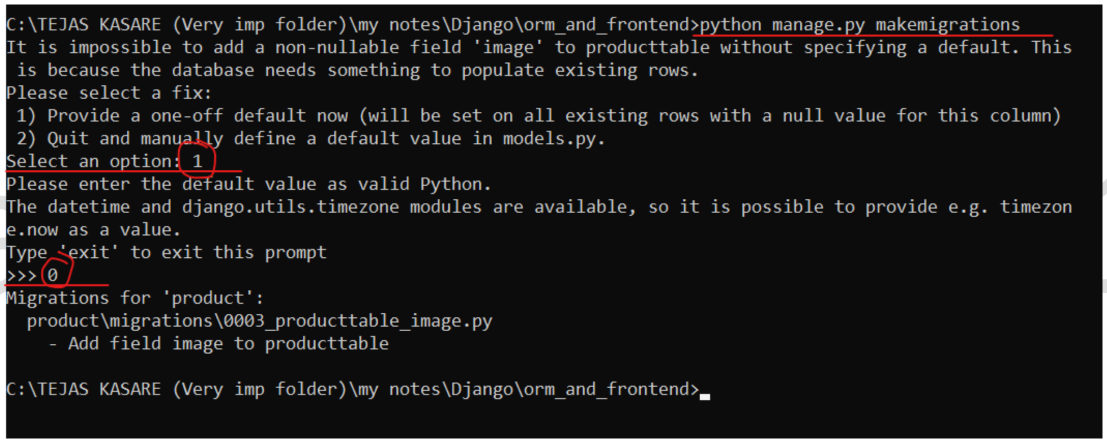


models.py

product > models.py > ProductTable > __str__

```python
from django.db import models

# Create your models here.
class ProductTable(models.Model):
    CATEGORIES = ((1,'Mobile'),(2,'Clothes'),(3,'Shoes'))
    name = models.CharField(max_length=50)
    price = models.FloatField()
    details=models.CharField(max_length=150)
    category = models.IntegerField(choices=CATEGORIES)
    is_active= models.BooleanField()
    rating = models.FloatField()
    image=models.ImageField(upload_to='image')

    def __str__(self) :
        return self.name + " added to table"
```

e. Makemigrations

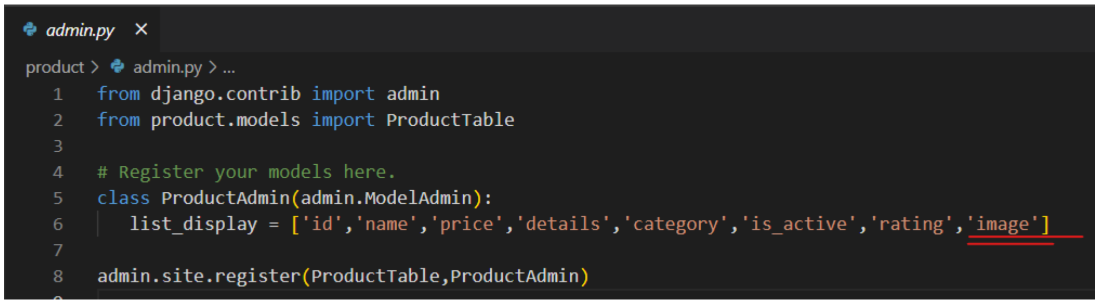


```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py makemigrations
It is impossible to add a non-nullable field 'image' to producttable without specifying a default. This
 is because the database needs something to populate existing rows.
Please select a fix:
 1) Provide 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. timezon
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



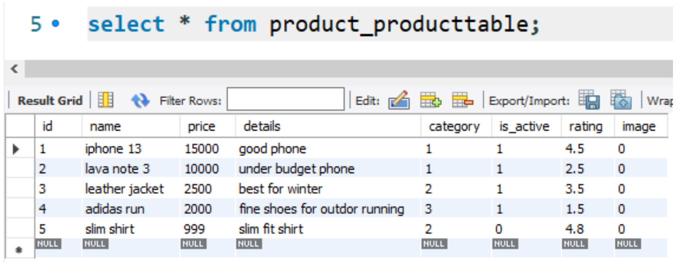
admin.py

product > admin.py > ...

```python
from django.contrib import admin
from product.models import ProductTable

# Register your models here.
class ProductAdmin(admin.ModelAdmin):
    list_display = ['id','name','price','details','category','is_active','rating','image']

admin.site.register(ProductTable,ProductAdmin)
```

h. Runserver

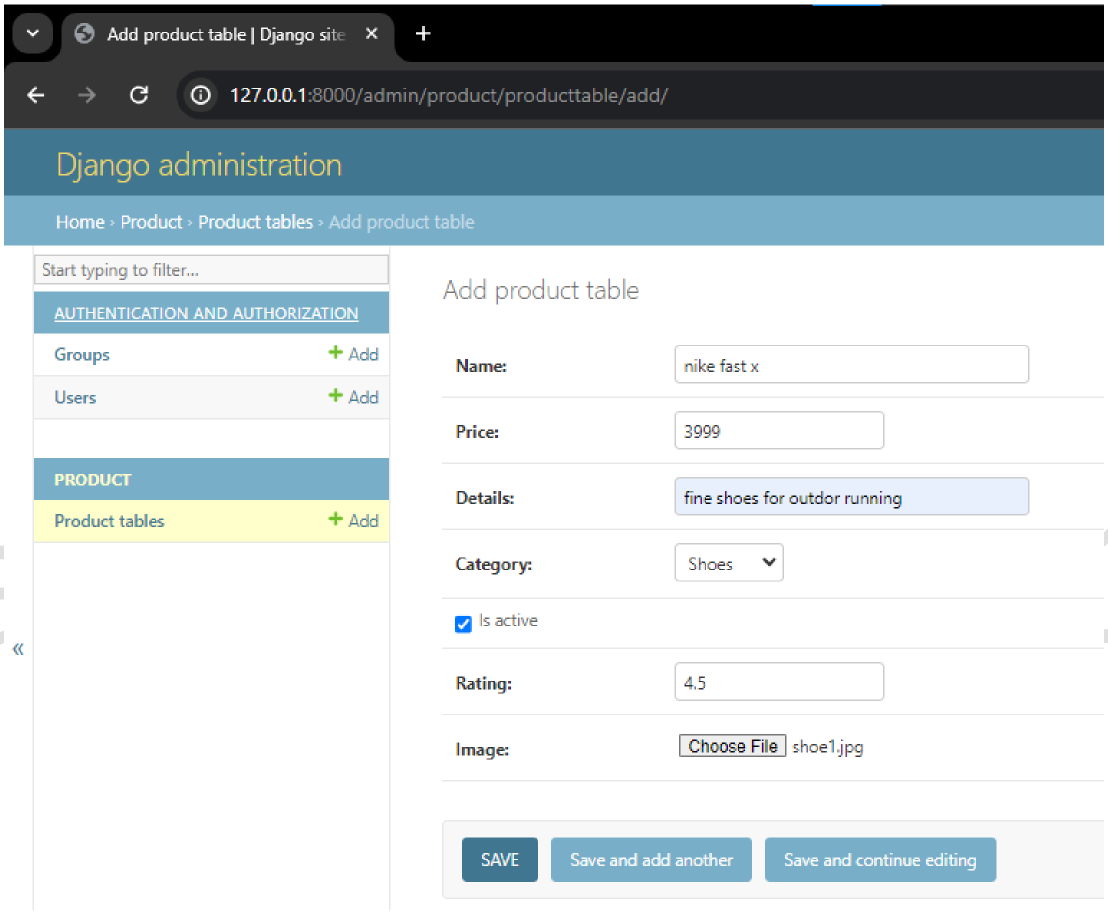
i. Check table in workbench, you will get image filed with default value 0



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

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

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



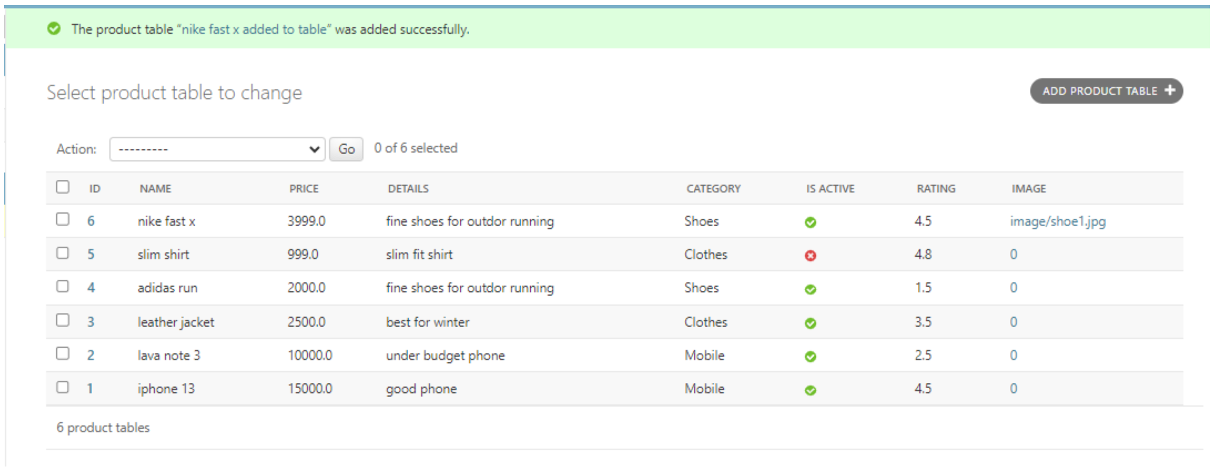
Add product table | Django site
127.0.0.1:8000/admin/product/producttable/add/
Django administration
Home › Product › Product tables › Add product table
Start typing to filter...
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
PRODUCT
Product tables + Add
«

Add product table

Name:	nike fast x
Price:	3999
Details:	fine shoes for outdor running
Category:	Shoes
☑ Is active	
Rating:	4.5
Image:	Choose File shoe1.jpg

SAVE
Save and add another
Save and continue editing

Teja 15



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

Select product table to change

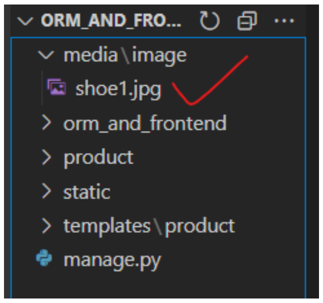
	ID	NAME	PRICE	DETAILS	CATEGORY	IS ACTIVE	RATING	IMAGE
☐	6	nike fast x	3999.0	fine shoes for outdor running	Shoes	✓	4.5	image/shoe1.jpg
☐	5	slim shirt	999.0	slim fit shirt	Clothes	✗	4.8	0
☐	4	adidas run	2000.0	fine shoes for outdor running	Shoes	✓	1.5	0
☐	3	leather jacket	2500.0	best for winter	Clothes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

6 product tables

k. Check in media folder –

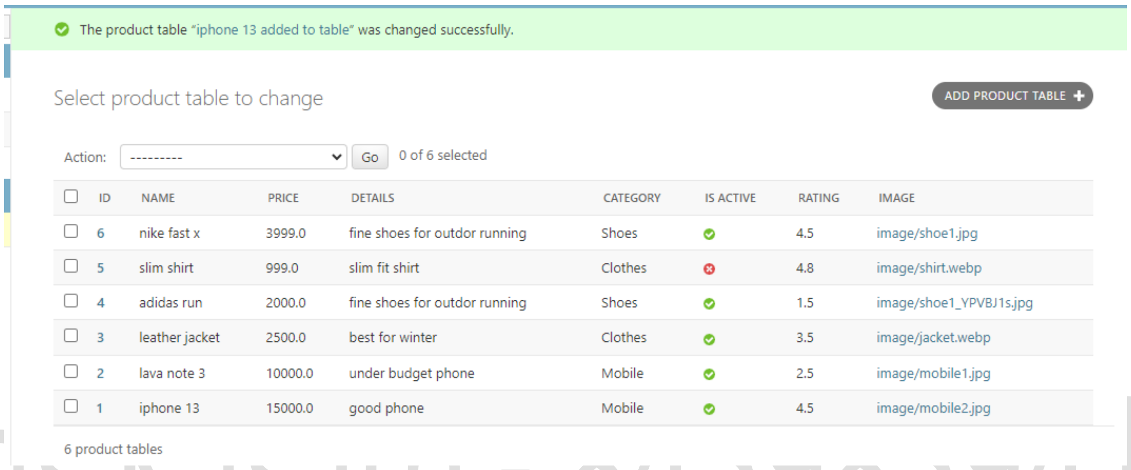
   i. You will get image folder

   ii. In image folder you will get your uploaded image



```
ORM_AND_FRO...
  media\image
    shoe1.jpg
  > orm_and_frontend
  > product
  > static
  > templates\product
  manage.py
```

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

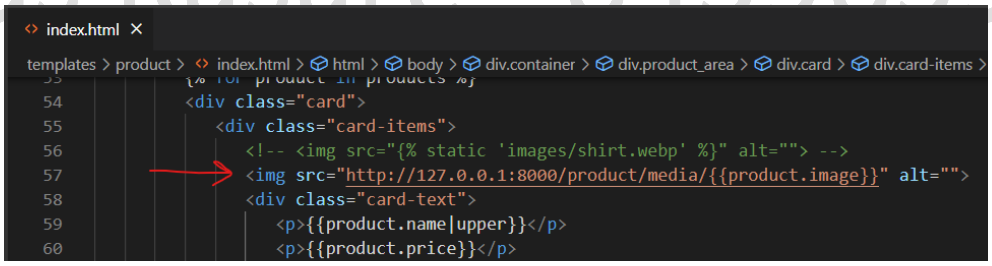
Action: --------- Go 0 of 6 selected

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

6 product tables

Tejas Kasare - 8459859415

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

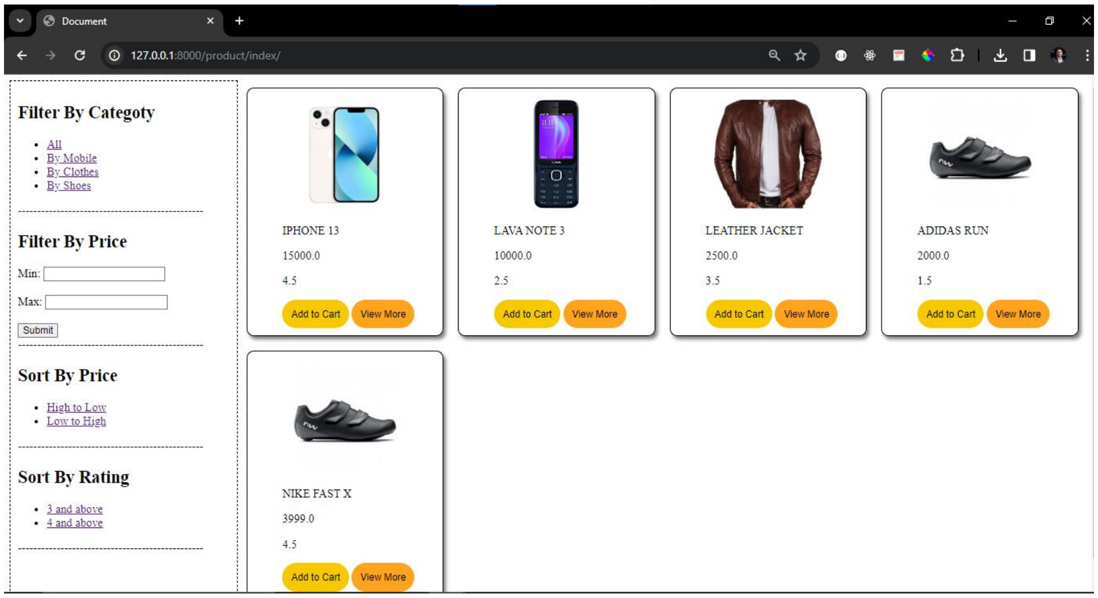


index.html

templates > product > index.html > html > body > div.container > div.product_area > div.card > div.card-items >

```
            {% for product in products %}
            <div class="card">
                <div class="card-items">
                    <!-- <img src="{% static 'images/shirt.webp' %}" alt=""> -->
                    <img src="http://127.0.0.1:8000/product/media/{{product.image}}" alt="">
                    <div class="card-text">
                        <p>{{product.name|upper}}</p>
                        <p>{{product.price}}</p>
```

OUTPUT :



Document

127.0.0.1:8000/product/index/

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

Min:

Max:

Submit

Sort By Price

- High to Low
- Low to High

Sort By Rating

- 3 and above
- 4 and above

IPHONE 13
15000.0
4.5
Add to Cart View More

LAVA NOTE 3
10000.0
2.5
Add to Cart View More

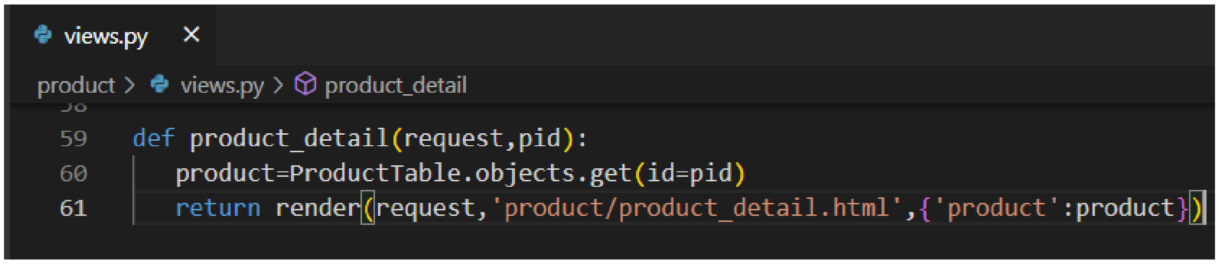
LEATHER JACKET
2500.0
3.5
Add to Cart View More

ADIDAS RUN
2000.0
1.5
Add to Cart View More

NIKE FAST X
3999.0
4.5
Add to Cart View More

7. View More Functionality (View individual product details)

   a. Create view to fetch product based on id

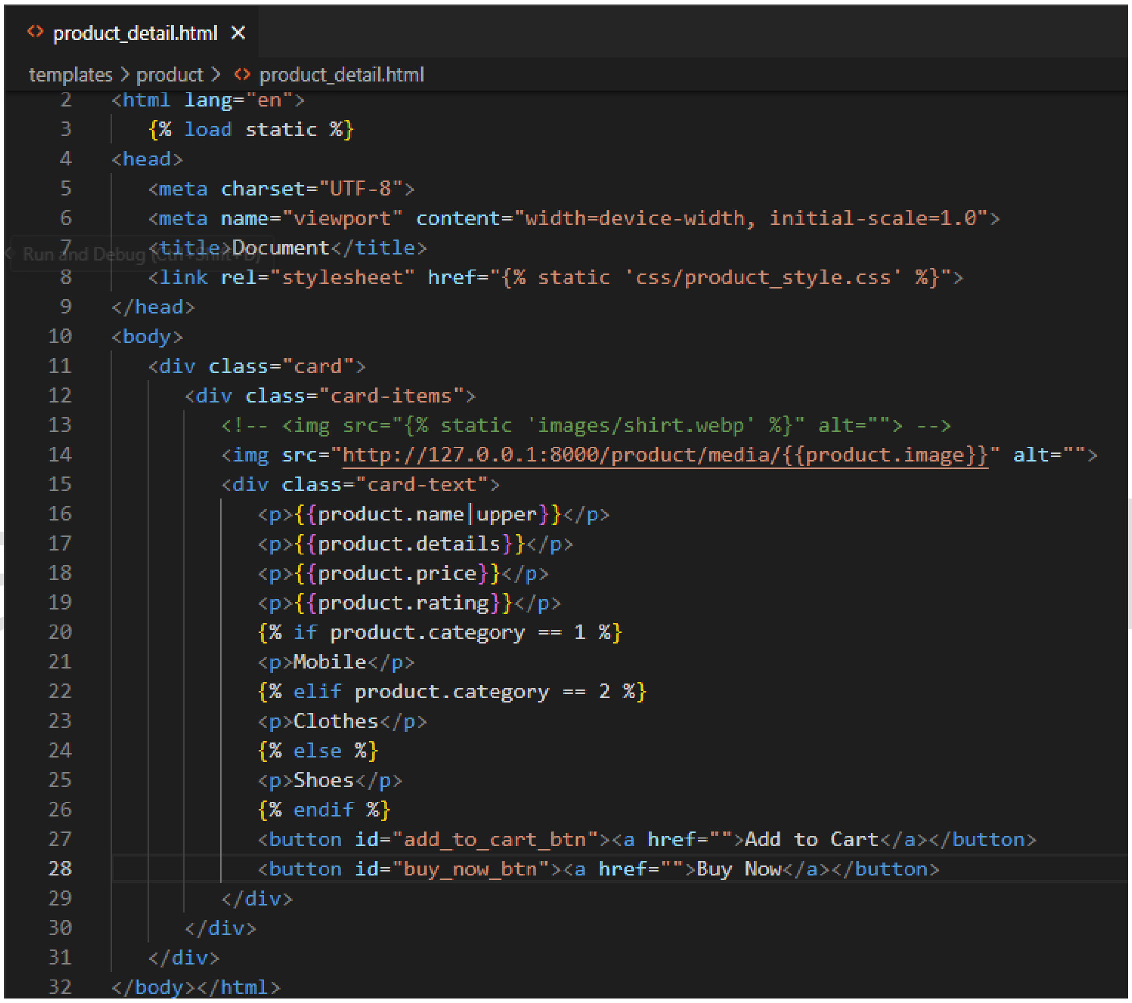


views.py

product > views.py > product_detail

```python
def product_detail(request,pid):
    product=ProductTable.objects.get(id=pid)
    return render(request,'product/product_detail.html',{'product':product})
```

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

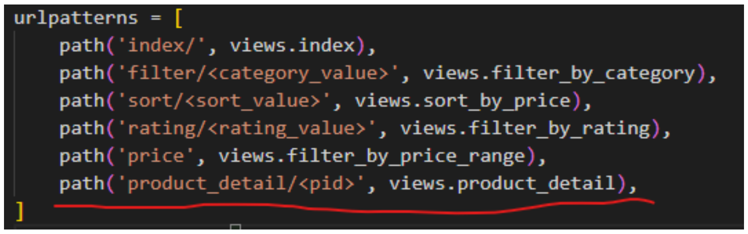


product_detail.html

templates > product > product_detail.html

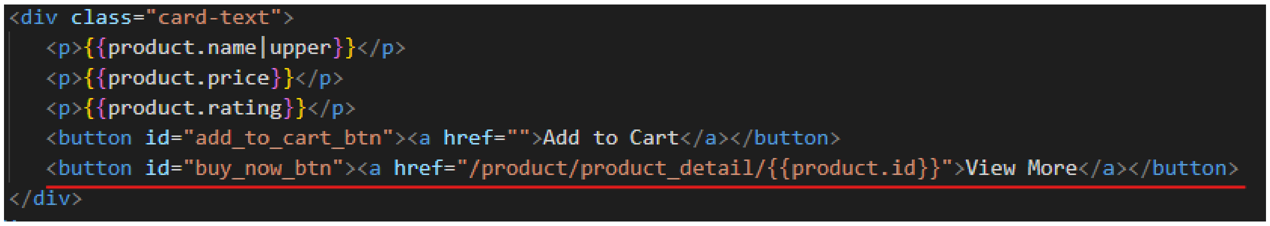
```html
<html lang="en">
    {% load static %}
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="{% static 'css/product_style.css' %}">
</head>
<body>
    <div class="card">
        <div class="card-items">
            <!-- <img src="{% static 'images/shirt.webp' %}" alt=""> -->
            <img src="http://127.0.0.1:8000/product/media/{{product.image}}" alt="">
            <div class="card-text">
                <p>{{product.name|upper}}</p>
                <p>{{product.details}}</p>
                <p>{{product.price}}</p>
                <p>{{product.rating}}</p>
                {% if product.category == 1 %}
                <p>Mobile</p>
                {% elif product.category == 2 %}
                <p>Clothes</p>
                {% else %}
                <p>Shoes</p>
                {% endif %}
                <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>
</body></html>
```

   c. Create url for above view



```python
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),
]
```

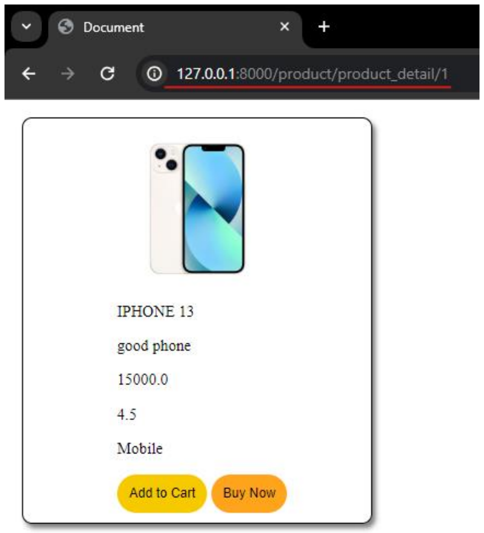
   d. Use url in index.html on view more button :



```html
<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>
```

Teja 15

e. Check output :



127.0.0.1:8000/product/product_detail/1

IPHONE 13

good phone

15000.0

4.5

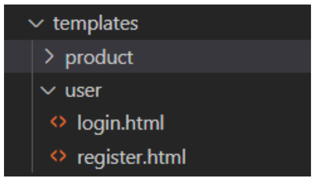
Mobile

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)

Tejas

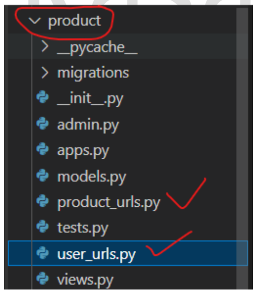
```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Document</title>
</head>
<body>
   <table align="center" border="1" cellpadding="5" cellspacing="5">
      <form method="POST">
         {% csrf_token %}
      <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)

```python
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']='Fileds 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 + ' alredy 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')
```

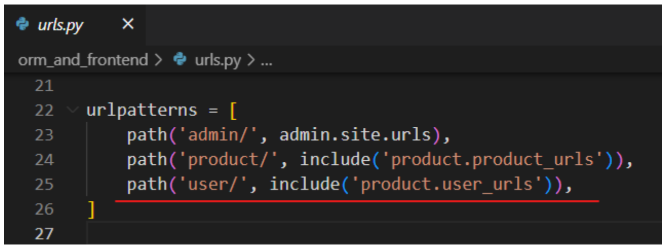
Tejas Kasare - 8459859415

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



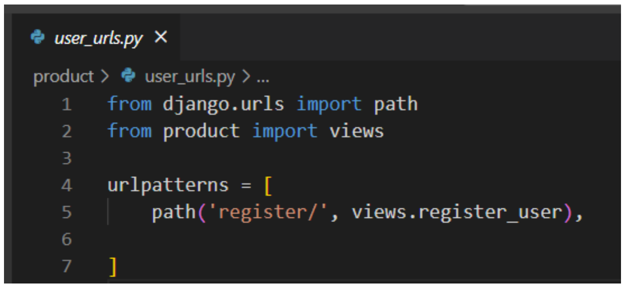
```
product
  > __pycache__
  > migrations
  __init__.py
  admin.py
  apps.py
  models.py
  product_urls.py
  tests.py
  user_urls.py
  views.py
```

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



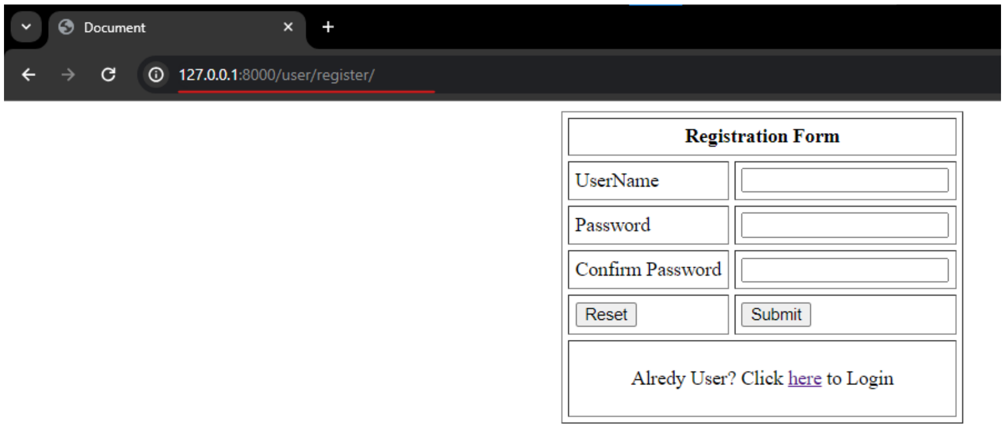
```
urls.py
orm_and_frontend > urls.py > ...

urlpatterns = [
    path('admin/', admin.site.urls),
    path('product/', include('product.product_urls')),
    path('user/', include('product.user_urls')),
]
```

Finally, url to display above resgistration form



```python
user_urls.py
product > user_urls.py > ...
from django.urls import path
from product import views

urlpatterns = [
    path('register/', views.register_user),

]
```

Output



Registration Form	
UserName	
Password	
Confirm Password	
Reset	Submit
Alredy User? Click here to Login	

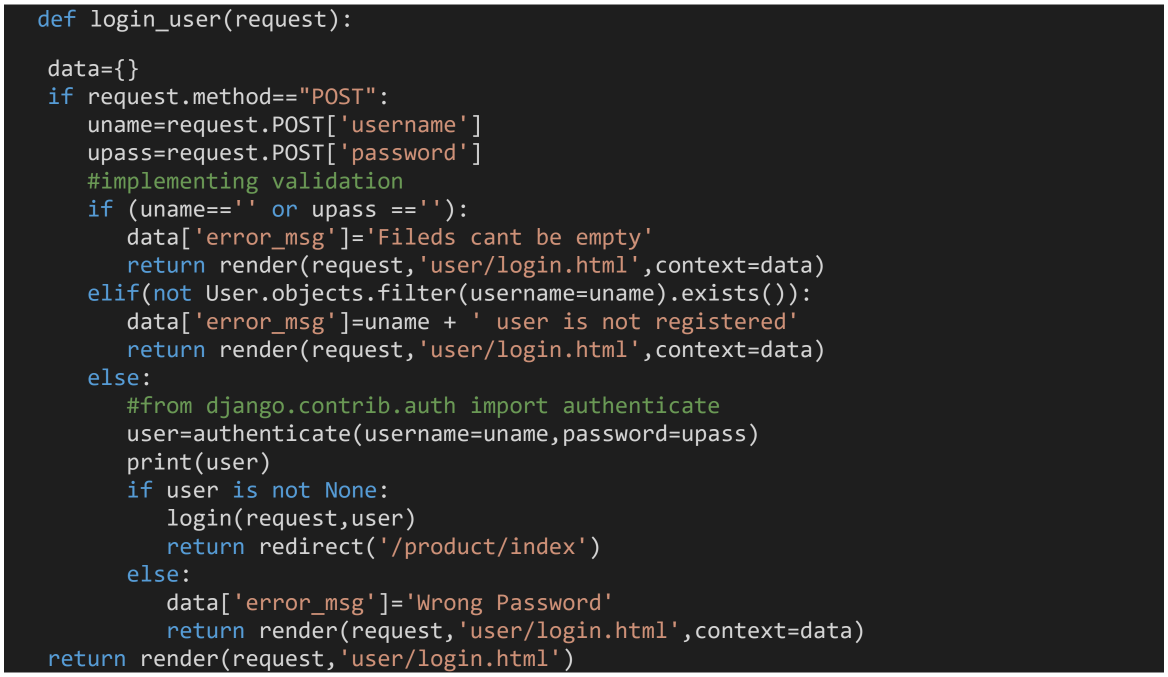
b. login user:

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>
```

Tejas

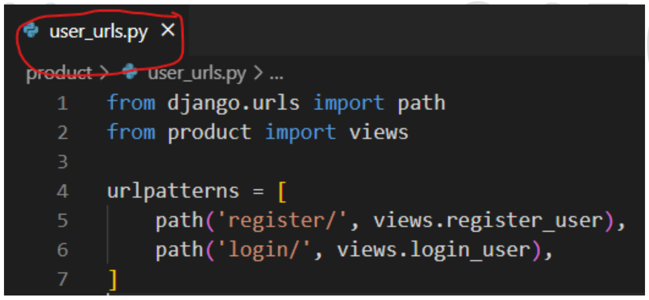
ii. create view to show login form and logic to login(I have used entire same logic same as we gave done in previous project number 8 : user login registration and session)



```
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']='Fileds 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

Tejas 9859415

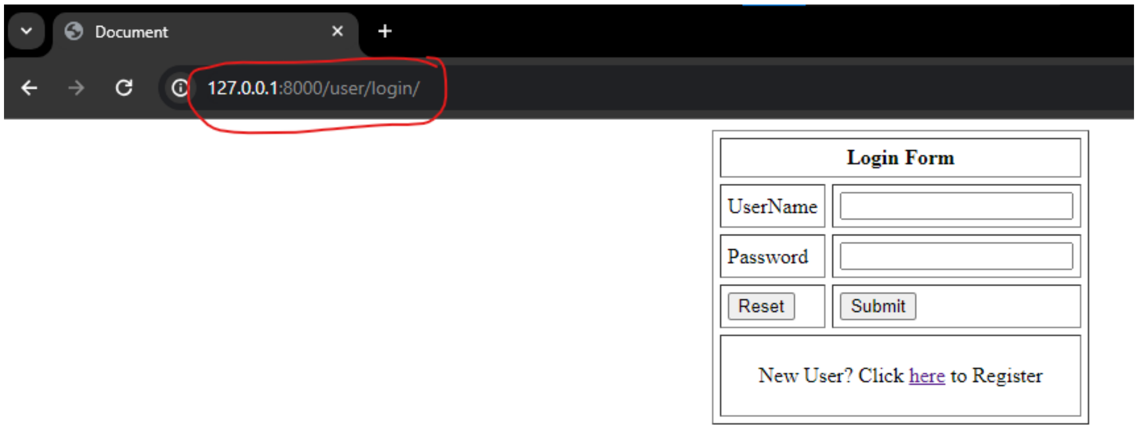


user_urls.py ×

product > user_urls.py > ...

```
from django.urls import path
from product import views

urlpatterns = [
    path('register/', views.register_user),
    path('login/', views.login_user),
]
```

Output



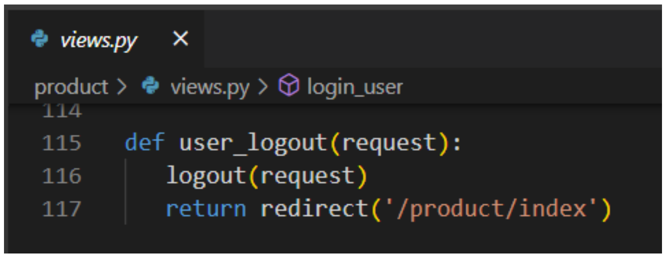
Document

127.0.0.1:8000/user/login/

Login Form	
UserName	
Password	
Reset	Submit
New User? Click here to Register	

c. Logout

i. Create view for logout

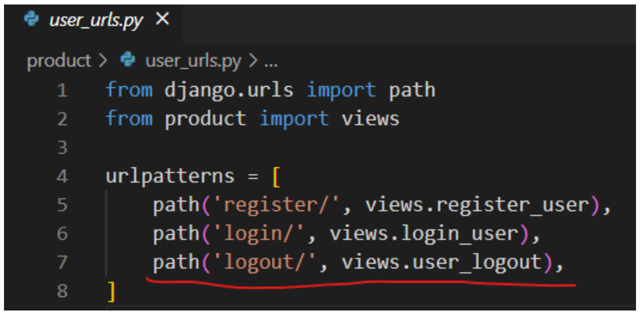


views.py

product > views.py > login_user

```
def user_logout(request):
    logout(request)
    return redirect('/product/index')
```

ii. Create url for above view



user_urls.py

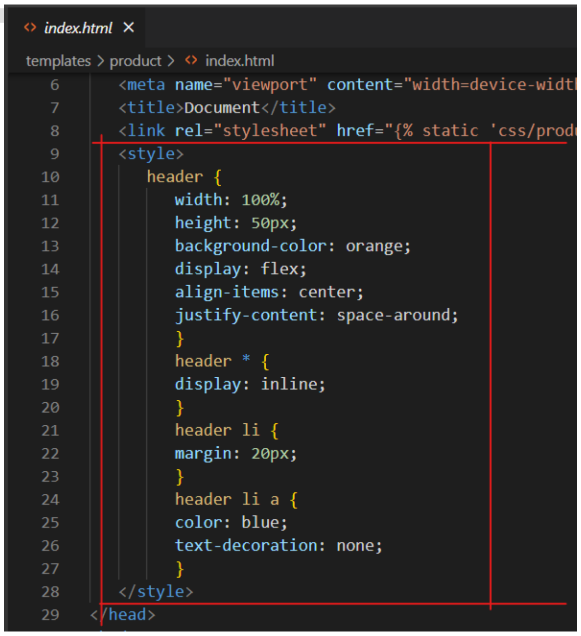
product > user_urls.py > ...

```
from django.urls import path
from product import views

urlpatterns = [
    path('register/', views.register_user),
    path('login/', views.login_user),
    path('logout/', views.user_logout),
]
```

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

i. In our index.html, add following code

Inline css for navbar

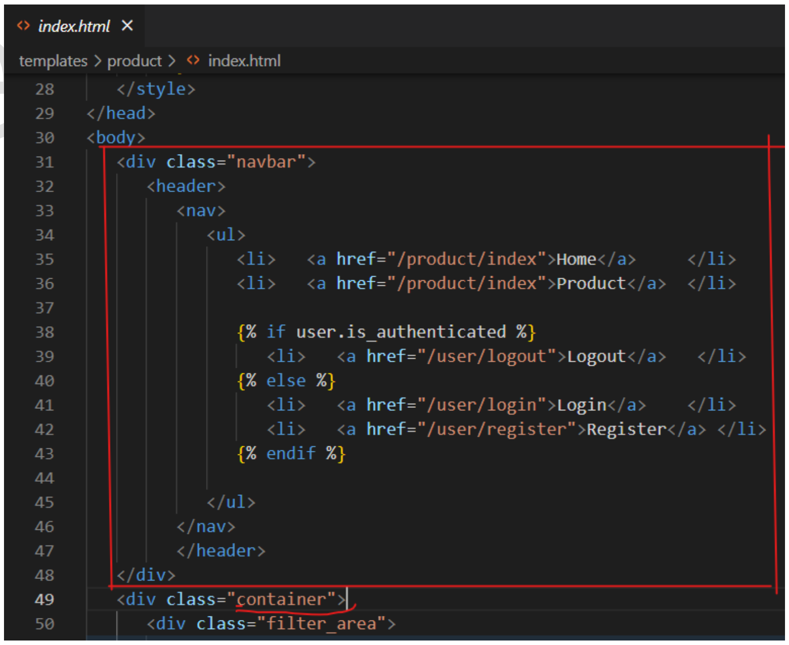


index.html

templates > product > index.html

```
    <meta name="viewport" content="width=device-widt
    <title>Document</title>
    <link rel="stylesheet" href="{% static 'css/prod
    <style>
        header {
            width: 100%;
            height: 50px;
            background-color: orange;
            display: flex;
            align-items: center;
            justify-content: space-around;
            }
            header * {
            display: inline;
            }
            header li {
            margin: 20px;
            }
            header li a {
            color: blue;
            text-decoration: none;
            }
    </style>
</head>
```

Navbar code

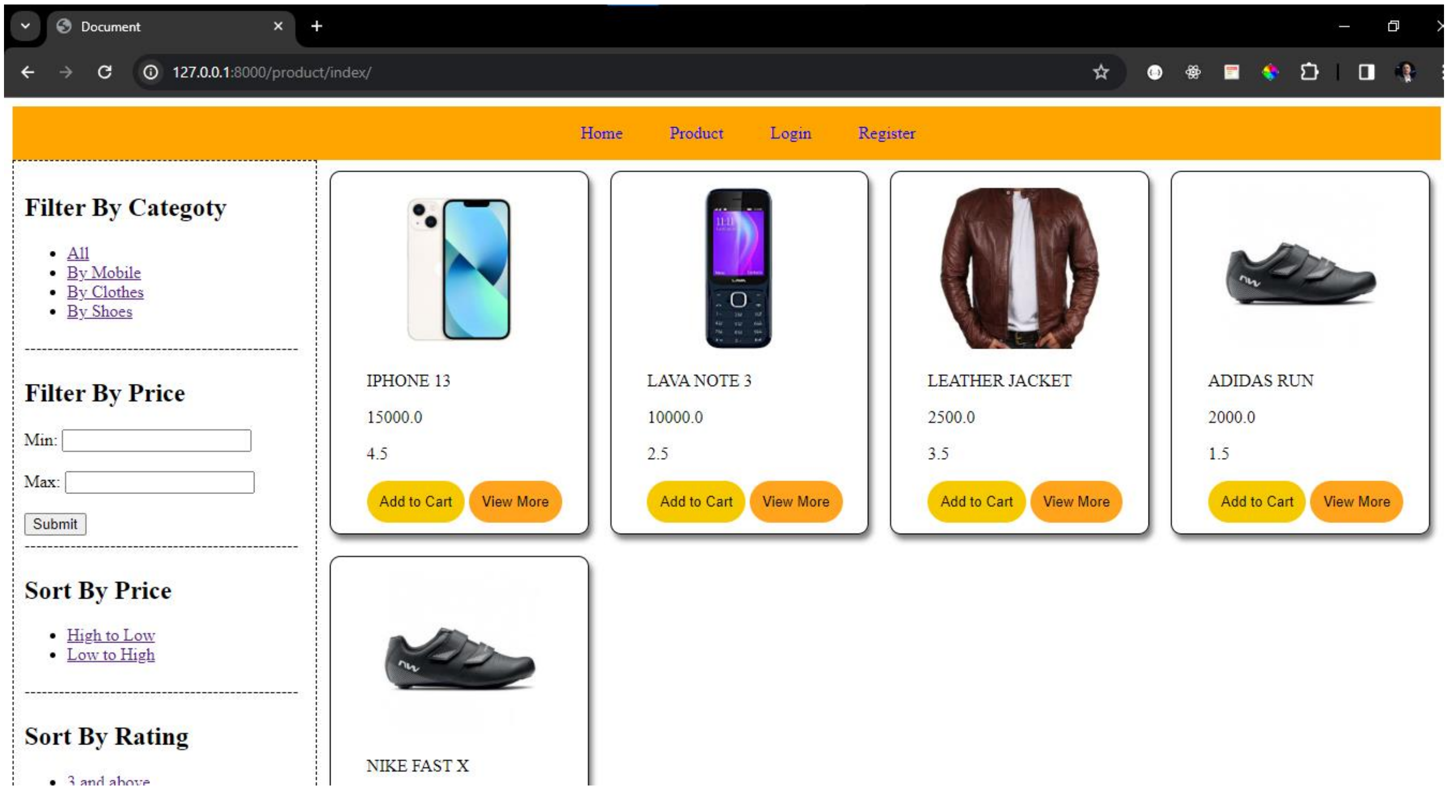


index.html

templates > product > index.html

```
    </style>
</head>
<body>
    <div class="navbar">
        <header>
            <nav>
                <ul>
                    <li>    <a href="/product/index">Home</a>     </li>
                    <li>    <a href="/product/index">Product</a>  </li>

                    {% if user.is_authenticated %}
                        <li>    <a href="/user/logout">Logout</a>    </li>
                    {% else %}
                        <li>    <a href="/user/login">Login</a>     </li>
                        <li>    <a href="/user/register">Register</a> </li>
                    {% endif %}

                </ul>
            </nav>
        </header>
    </div>
    <div class="container">
        <div class="filter_area">
```

e. Final output


Document

127.0.0.1:8000/product/index/

Home Product Login Register

Filter By Categoty

- All
- By Mobile
- By Clothes
- By Shoes

Filter By Price

Min:

Max:

Submit

Sort By Price

- High to Low
- Low to High

Sort By Rating

- 3 and above

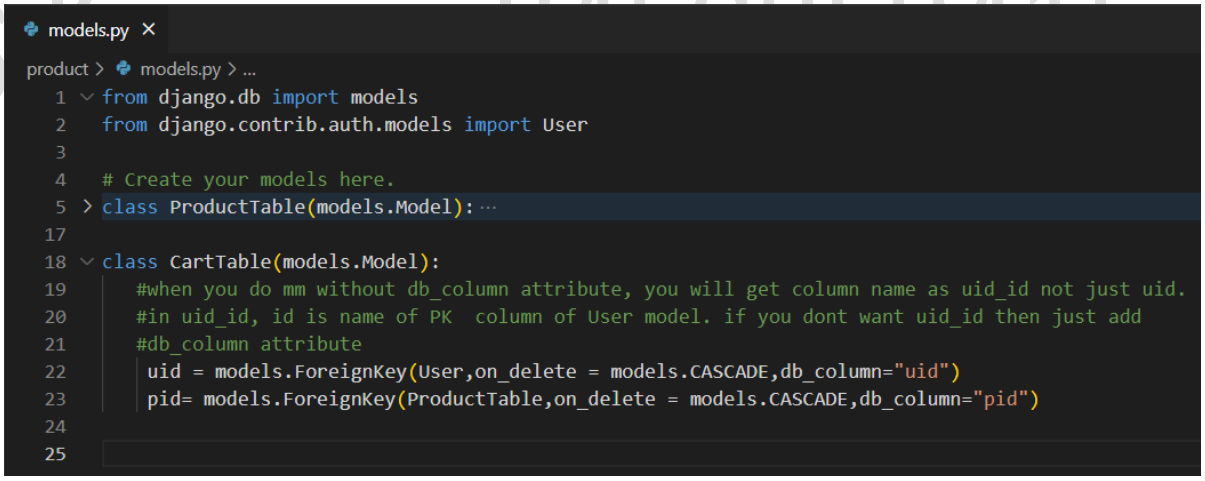
IPHONE 13	LAVA NOTE 3	LEATHER JACKET	ADIDAS RUN
15000.0	10000.0	2500.0	2000.0
4.5	2.5	3.5	1.5
Add to Cart View More	Add to Cart View More	Add to Cart View More	Add to Cart View More

NIKE FAST X

f. Finally, add to cart functionality :

i. Create model for cart

Tejas

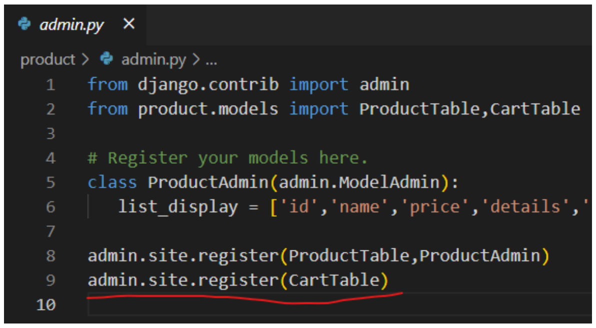

models.py

product > models.py > ...

```python
from django.db import models
from django.contrib.auth.models import User

# Create your models here.
class ProductTable(models.Model): ...

class CartTable(models.Model):
    #when you do mm without db_column attribute, you will get column name as uid_id not just uid.
    #in uid_id, id is name of PK  column of User model. if you dont want uid_id then just add
    #db_column attribute
     uid = models.ForeignKey(User,on_delete = models.CASCADE,db_column="uid")
     pid= models.ForeignKey(ProductTable,on_delete = models.CASCADE,db_column="pid")

```

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

ii. Register model in admin.py

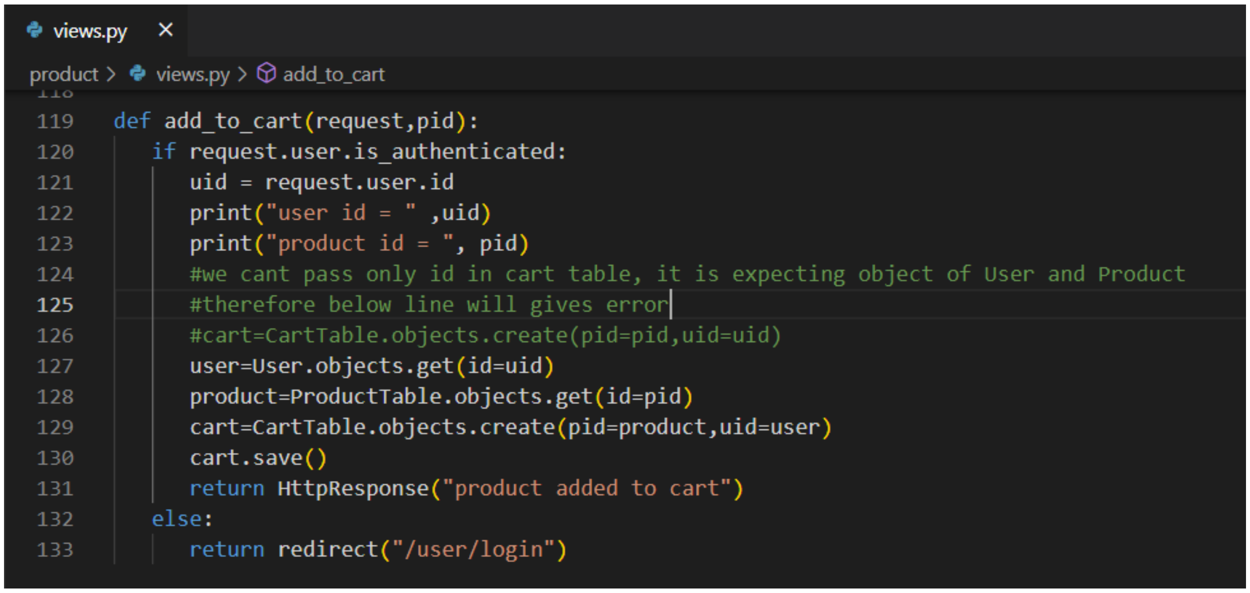

admin.py

product > admin.py > ...

```python
from django.contrib import admin
from product.models import ProductTable,CartTable

# Register your models here.
class ProductAdmin(admin.ModelAdmin):
    list_display = ['id','name','price','details','

admin.site.register(ProductTable,ProductAdmin)
admin.site.register(CartTable)

```

iii. MM

iv. View (basic functionality to check working)

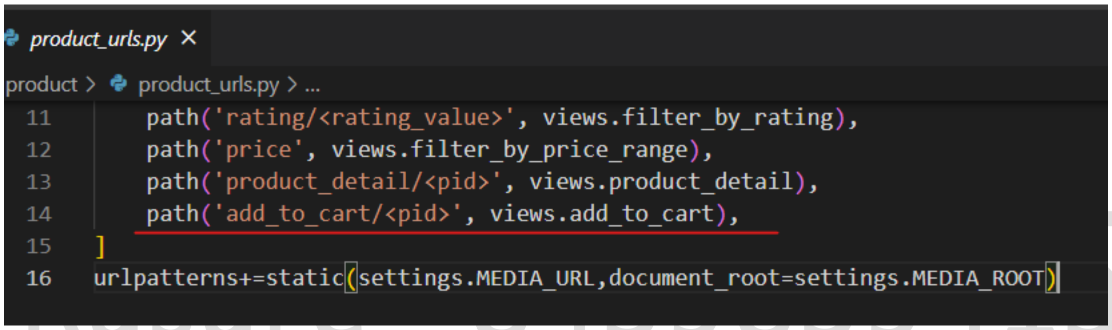


views.py

product > views.py > add_to_cart

```python
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)
        cart=CartTable.objects.create(pid=product,uid=user)
        cart.save()
        return HttpResponse("product added to cart")
    else:
        return redirect("/user/login")
```

v. url



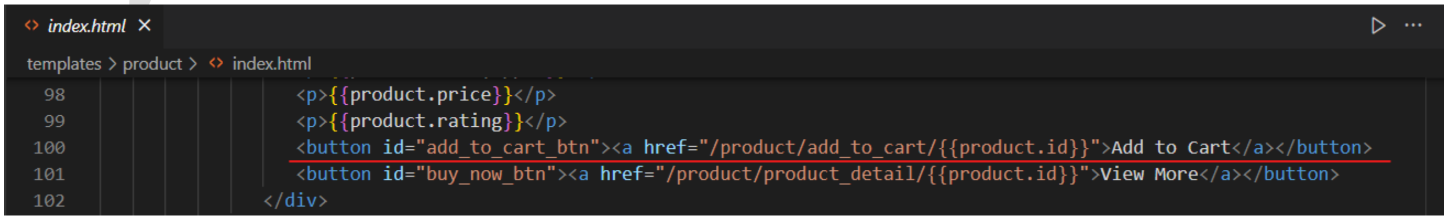
product_urls.py

product > product_urls.py > ...

```python
    path('rating/<rating_value>', views.filter_by_rating),
    path('price', views.filter_by_price_range),
    path('product_detail/<pid>', views.product_detail),
    path('add_to_cart/<pid>', views.add_to_cart),
]
urlpatterns+=static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

Tejas Kasare 8459859415

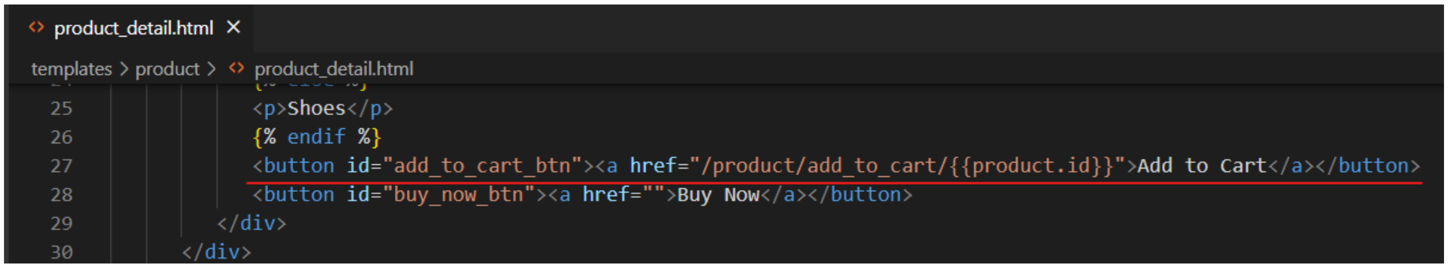
vi. use url – we need to use url on in 2 files –index.html and product_detail.html



index.html

templates > product > index.html

```html
                <p>{{product.price}}</p>
                <p>{{product.rating}}</p>
                <button id="add_to_cart_btn"><a href="/product/add_to_cart/{{product.id}}">Add to Cart</a></button>
                <button id="buy_now_btn"><a href="/product/product_detail/{{product.id}}">View More</a></button>
            </div>
```



product_detail.html

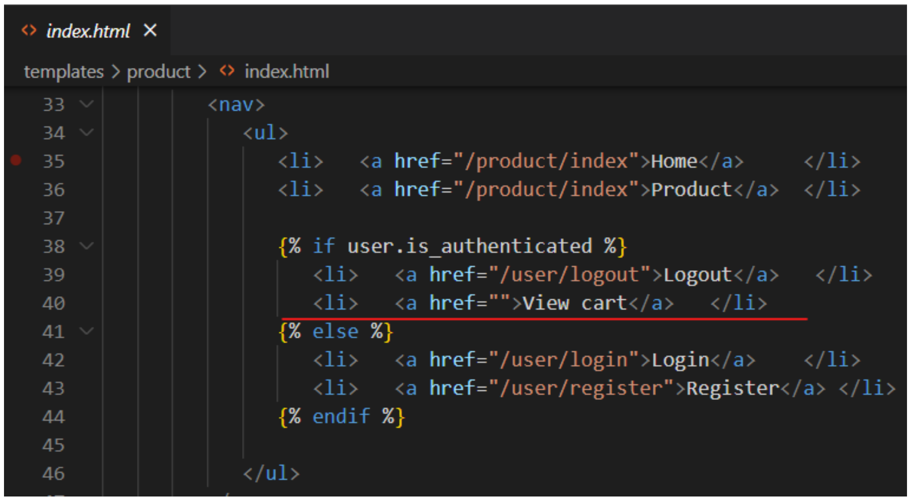
templates > product > product_detail.html

```html
                <p>Shoes</p>
                {% endif %}
                <button id="add_to_cart_btn"><a href="/product/add_to_cart/{{product.id}}">Add to Cart</a></button>
                <button id="buy_now_btn"><a href="">Buy Now</a></button>
            </div>
        </div>
```

DONEEE!!!

9. View Cart Functionality

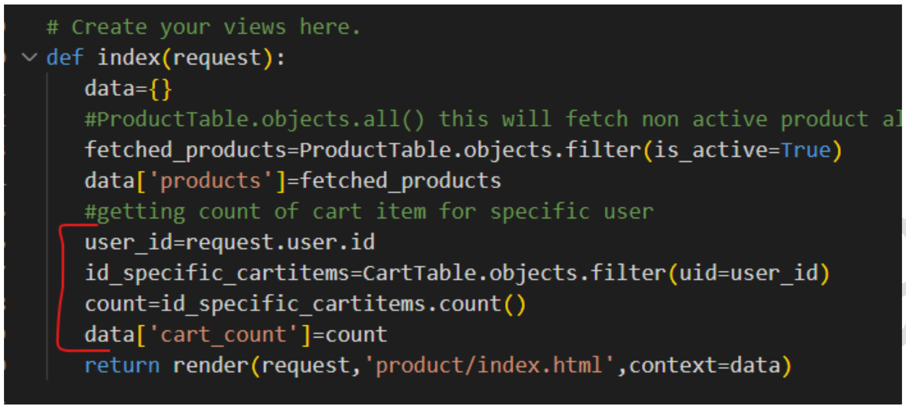
   a. Add view cart option in navbar – and only visible if user in logged in



```
index.html ×
templates > product > index.html
            <nav>
                <ul>
                    <li>    <a href="/product/index">Home</a>      </li>
                    <li>    <a href="/product/index">Product</a>   </li>

                    {% if user.is_authenticated %}
                        <li>    <a href="/user/logout">Logout</a>     </li>
                        <li>    <a href="">View cart</a>    </li>
                    {% else %}
                        <li>    <a href="/user/login">Login</a>     </li>
                        <li>    <a href="/user/register">Register</a> </li>
                    {% endif %}

                </ul>
```

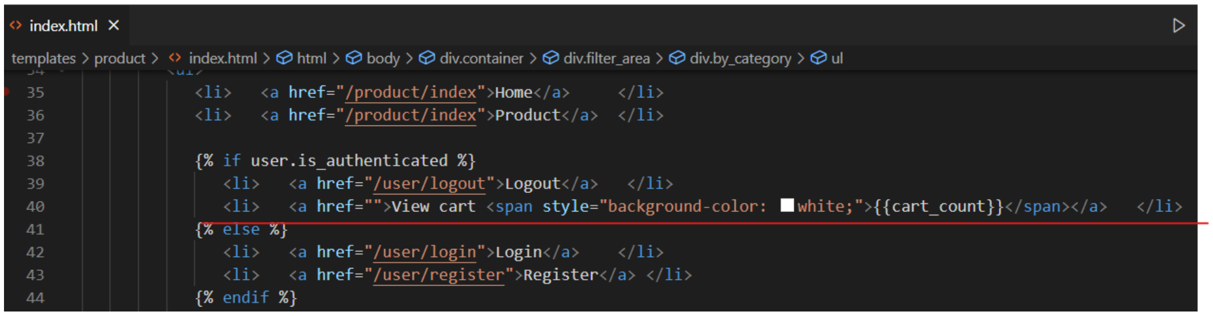
   b. Displaying number of products added in cart into navbar (cart total)

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



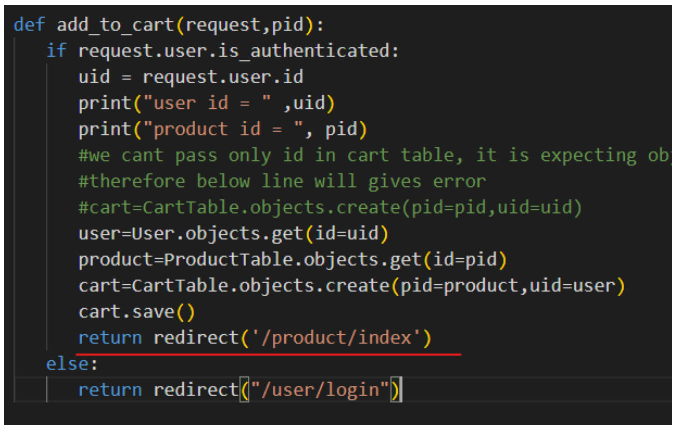
```
# Create your views here.
def index(request):
    data={}
    #ProductTable.objects.all() this will fetch non active product al
    fetched_products=ProductTable.objects.filter(is_active=True)
    data['products']=fetched_products
    #getting count of cart item for specific user
    user_id=request.user.id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

      ii. Display counter in html (index.html)



```
index.html ×
templates > product > index.html > html > body > div.container > div.filter_area > div.by_category > ul
                    <li>    <a href="/product/index">Home</a>      </li>
                    <li>    <a href="/product/index">Product</a>   </li>

                    {% if user.is_authenticated %}
                        <li>    <a href="/user/logout">Logout</a>    </li>
                        <li>    <a href="">View cart <span style="background-color: white;">{{cart_count}}</span></a>    </li>
                    {% else %}
                        <li>    <a href="/user/login">Login</a>     </li>
                        <li>    <a href="/user/register">Register</a> </li>
                    {% endif %}
```

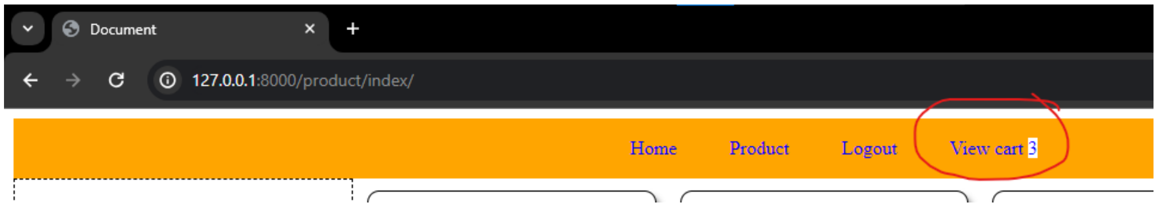
      iii. Redirecting to index page after clicking on add to cart button



```
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")
```

Tejas 9415

Output :



c. Displaying cart items on cart.html page

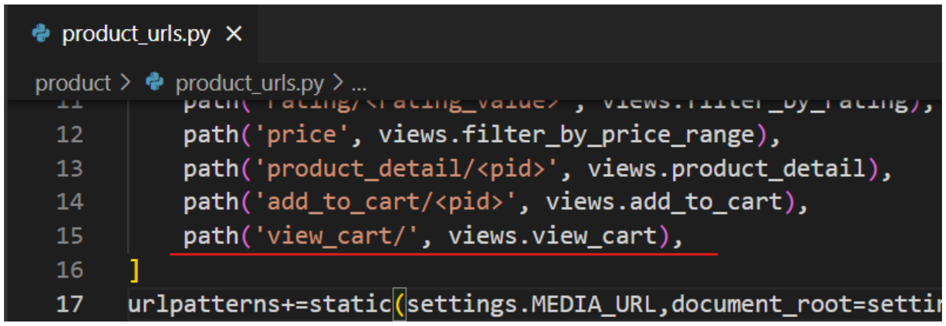
i. Create cart.html in templates> product> cart.html

```
<!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">
                  <img src="http://127.0.0.1:8000/product/media/{{product.pid.image}}" alt="">
               </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-
C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/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

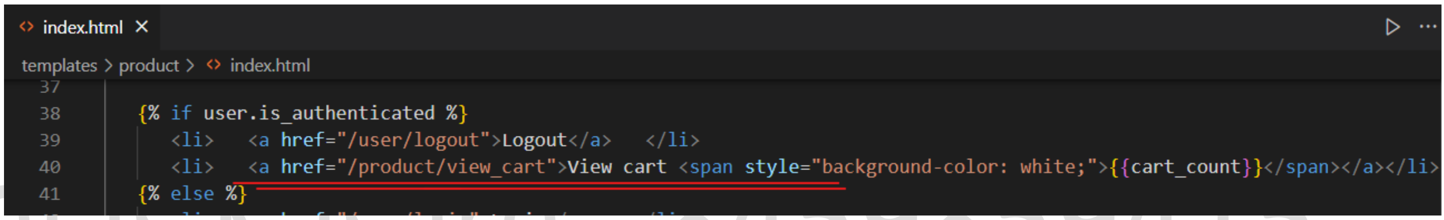


product_urls.py ×

product > product_urls.py > ...

```
    path('price', views.filter_by_price_range),
    path('product_detail/<pid>', views.product_detail),
    path('add_to_cart/<pid>', views.add_to_cart),
    path('view_cart/', views.view_cart),
]
urlpatterns+=static(settings.MEDIA_URL,document_root=settin
```

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



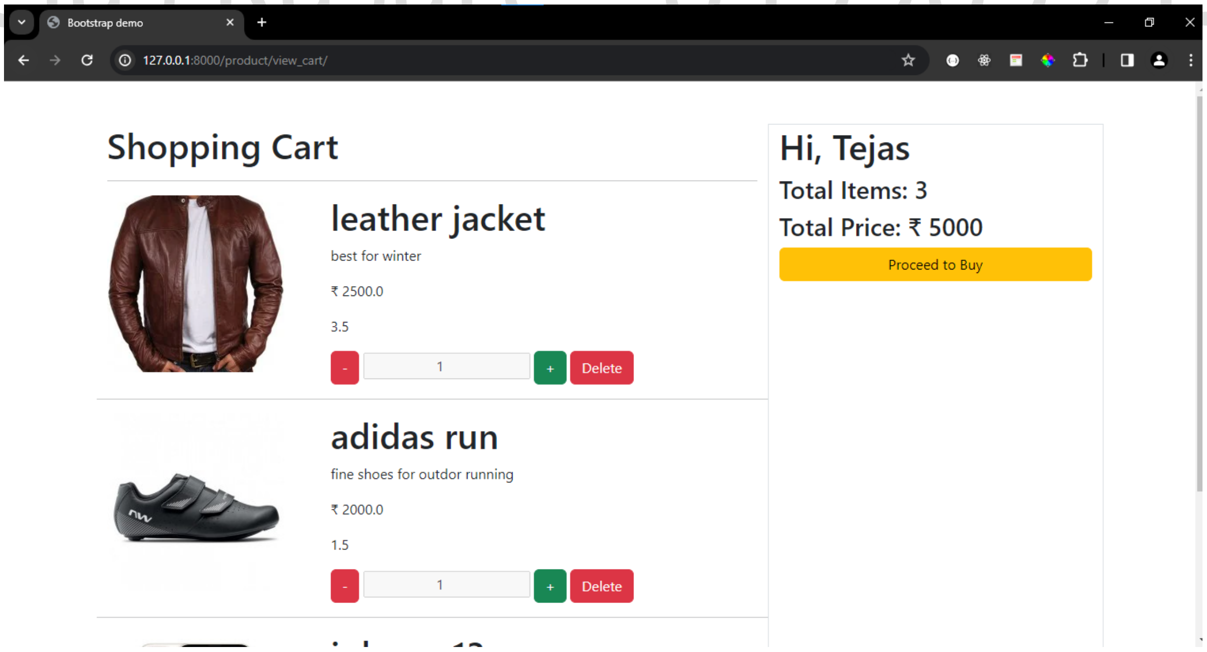
index.html ×

templates > product > index.html

```
{% if user.is_authenticated %}
    <li>    <a href="/user/logout">Logout</a>    </li>
    <li>    <a href="/product/view_cart">View cart <span style="background-color: white;">{{cart_count}}</span></a></li>
{% else %}
```

Tejas Kasare - 8459859415

OutPut :



Bootstrap demo

127.0.0.1:8000/product/view_cart/

Shopping Cart

leather jacket

best for winter

₹ 2500.0

3.5

- 1 + Delete

adidas run

fine shoes for outdor running

₹ 2000.0

1.5

- 1 + Delete

Hi, Tejas

Total Items: 3

Total Price: ₹ 5000

Proceed to Buy

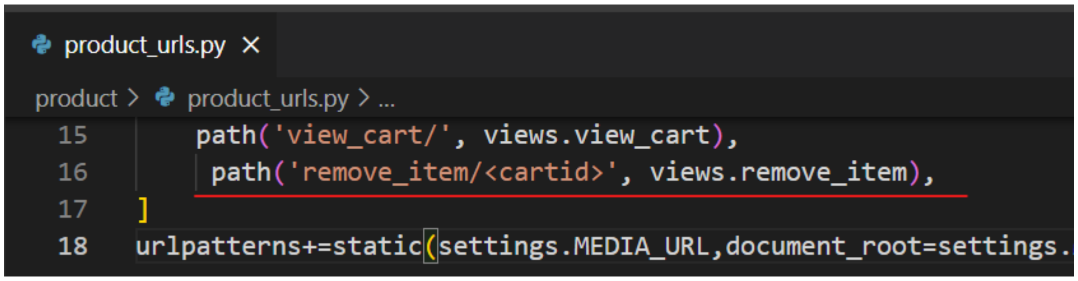
10. Remove item from cart functionality

We are going to remove cart item based on cart id

a. 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')
```

b. Create url form above remove_item() view

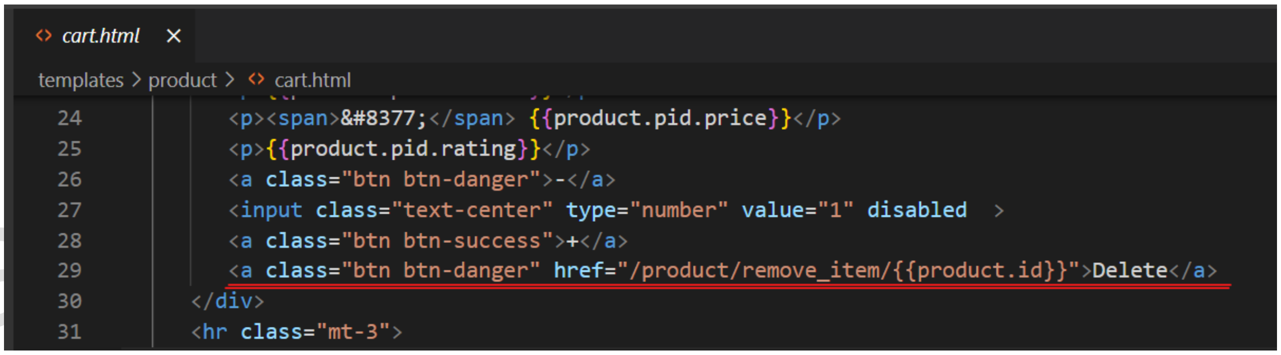


product_urls.py ×

product > product_urls.py > ...

```
        path('view_cart/', views.view_cart),
         path('remove_item/<cartid>', views.remove_item),
    ]
urlpatterns+=static(settings.MEDIA_URL,document_root=settings.
```

c. Use above url on delete button in cart.html



cart.html ×

templates > product > cart.html

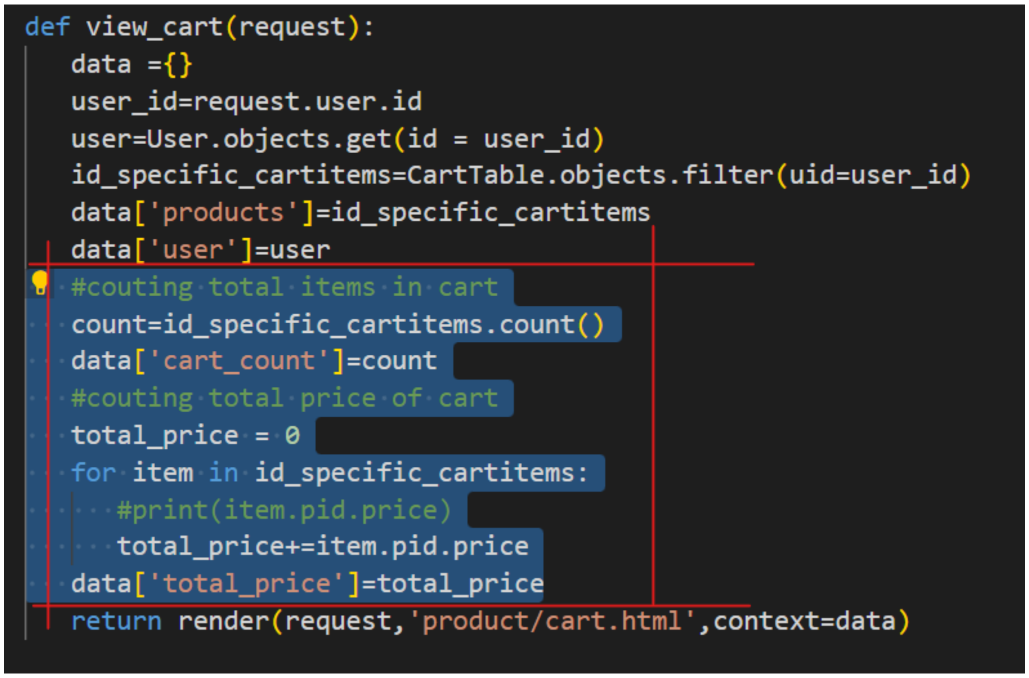
```
            <p><span>&#8377;</span> {{product.pid.price}}</p>
            <p>{{product.pid.rating}}</p>
            <a class="btn btn-danger">-</a>
            <input class="text-center" type="number" value="1" disabled  >
            <a class="btn btn-success">+</a>
            <a class="btn btn-danger" href="/product/remove_item/{{product.id}}">Delete</a>
        </div>
        <hr class="mt-3">
```

Teja

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

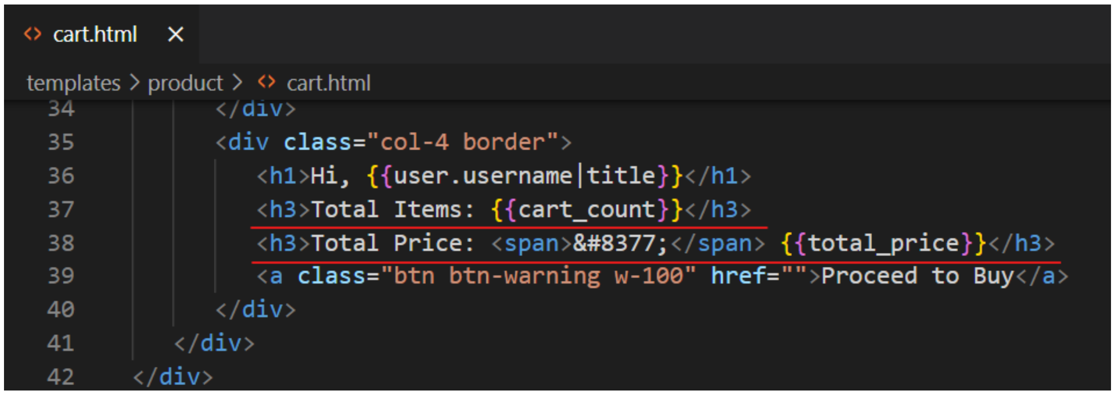
11. Total items and total price calculation

a. 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
    #couting total items in cart
    count=id_specific_cartitems.count()
    data['cart_count']=count
    #couting 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

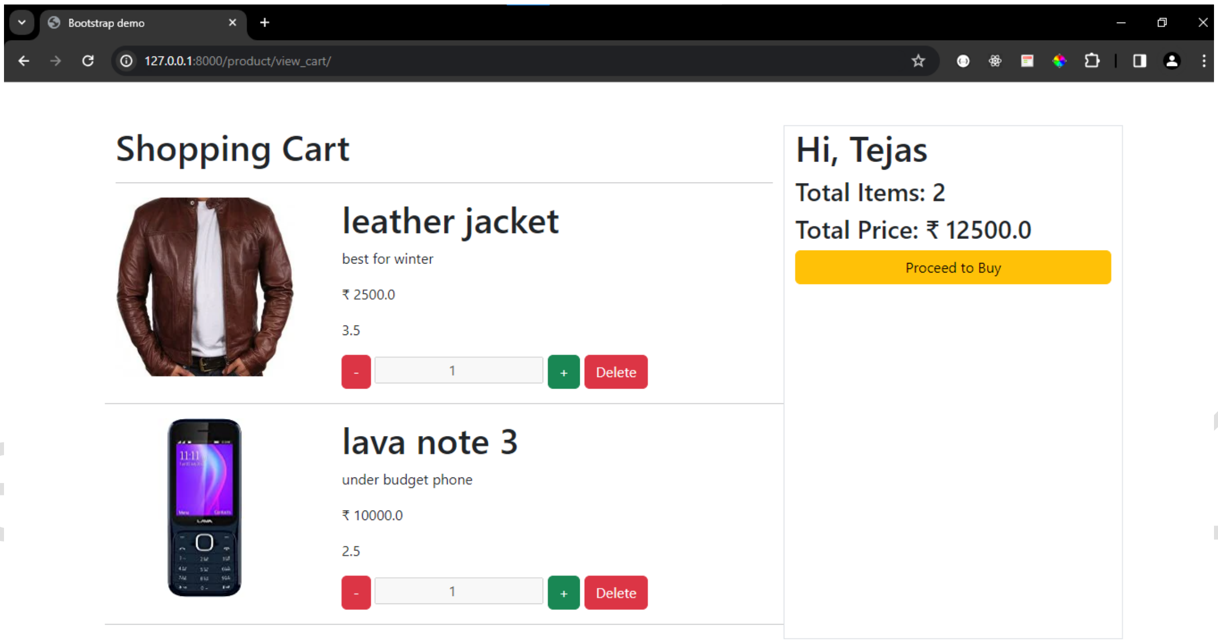


cart.html

templates > product > cart.html

```html
        </div>
        <div class="col-4 border">
            <h1>Hi, {{user.username|title}}</h1>
            <h3>Total Items: {{cart_count}}</h3>
            <h3>Total Price: <span>&#8377;</span> {{total_price}}</h3>
            <a class="btn btn-warning w-100" href="">Proceed to Buy</a>
        </div>
    </div>
</div>
```

c. Output



Bootstrap demo

127.0.0.1:8000/product/view_cart/

Shopping Cart

leather jacket

best for winter

₹ 2500.0

3.5

- 1 + Delete

lava note 3

under budget phone

₹ 10000.0

2.5

- 1 + Delete

Hi, Tejas

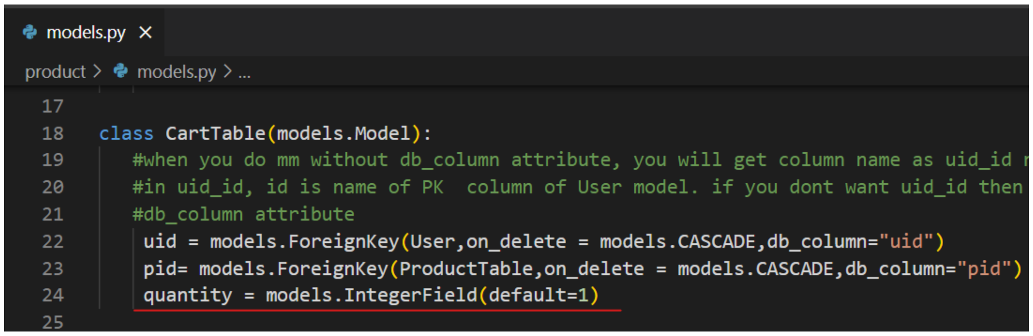
Total Items: 2

Total Price: ₹ 12500.0

Proceed to Buy

Te 15

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

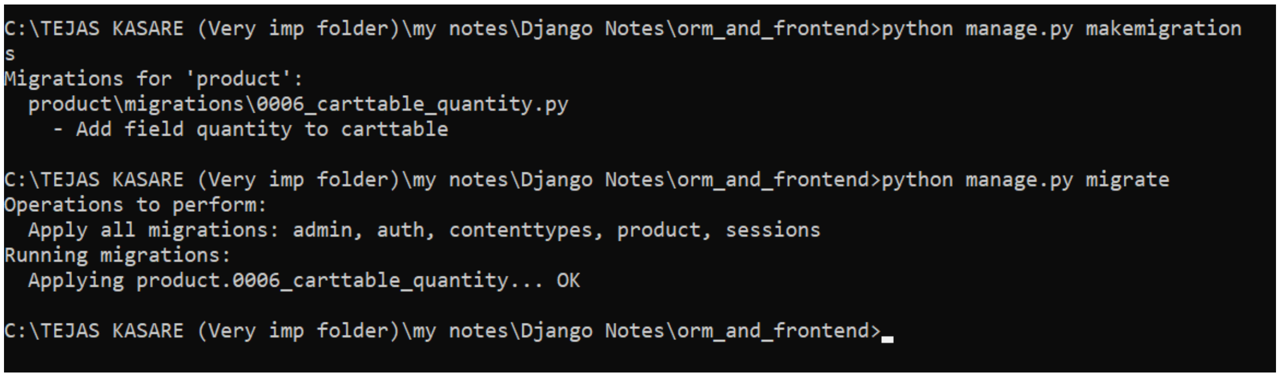


models.py

product > models.py > ...

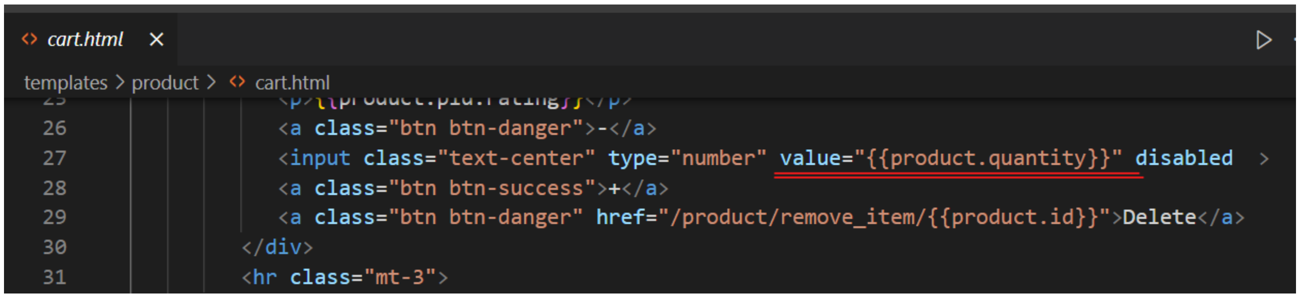
```python

class CartTable(models.Model):
    #when you do mm without db_column attribute, you will get column name as uid_id r
    #in uid_id, id is name of PK  column of User model. if you dont want uid_id then
    #db_column attribute
     uid = models.ForeignKey(User,on_delete = models.CASCADE,db_column="uid")
     pid= models.ForeignKey(ProductTable,on_delete = models.CASCADE,db_column="pid")
     quantity = models.IntegerField(default=1)

```

b. MM



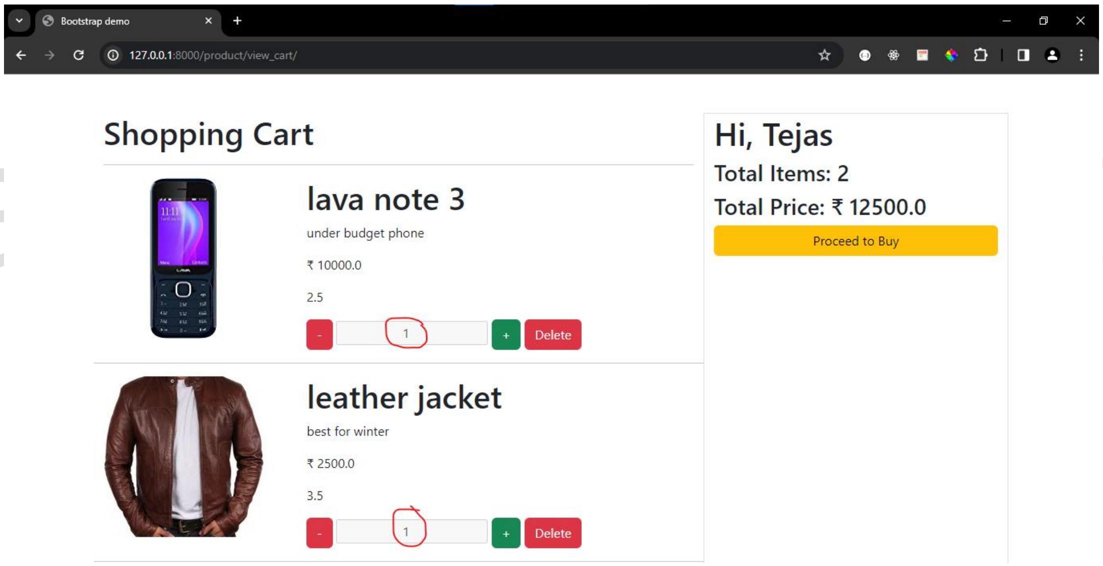
```
C:\TEJAS KASARE (Very imp folder)\my notes\Django Notes\orm_and_frontend>python manage.py makemigration
s
Migrations for 'product':
  product\migrations\0006_carttable_quantity.py
    - Add field quantity to carttable

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



```
            <a class="btn btn-danger">-</a>
            <input class="text-center" type="number" value="{{product.quantity}}" disabled
            <a class="btn btn-success">+</a>
            <a class="btn btn-danger" href="/product/remove_item/{{product.id}}">Delete</a>
        </div>
        <hr class="mt-3">
```

d. Output :



Shopping Cart

lava note 3

under budget phone

₹ 10000.0

2.5

leather jacket

best for winter

₹ 2500.0

3.5

Hi, Tejas

Total Items: 2

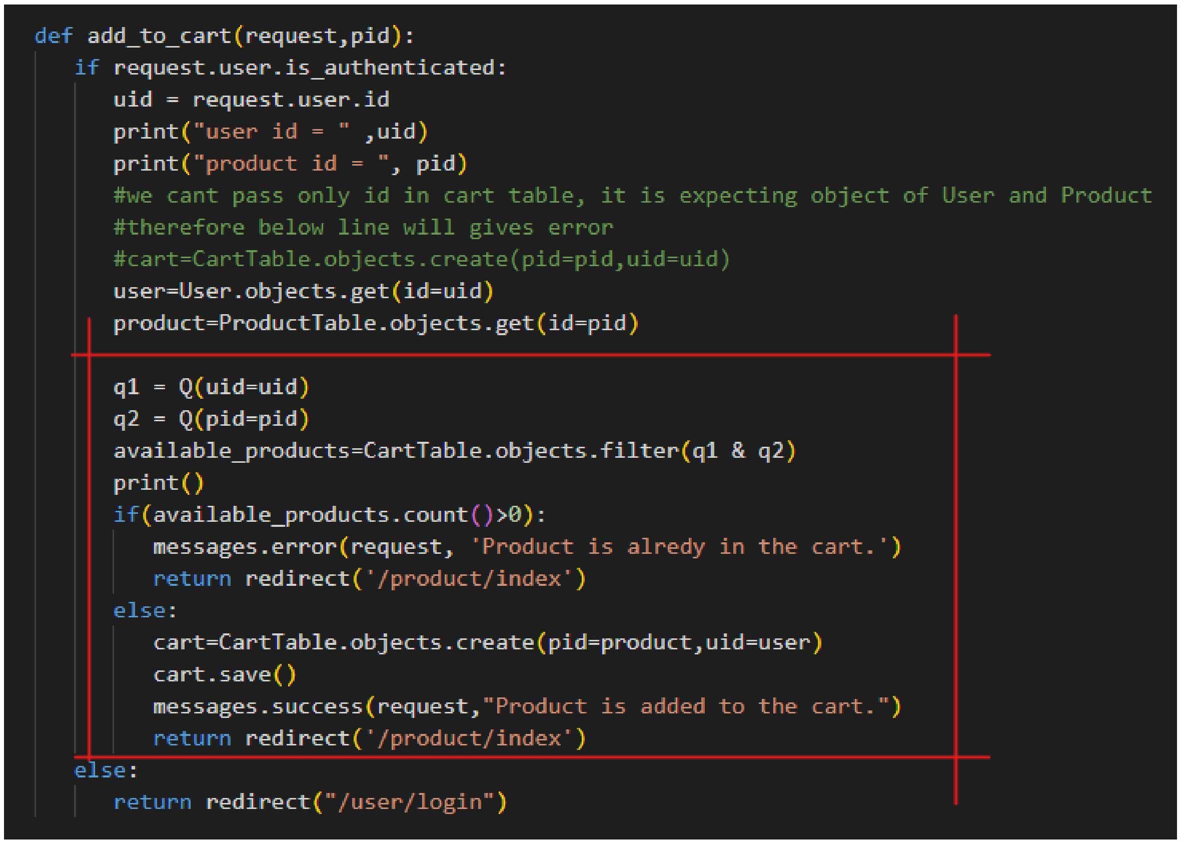
Total Price: ₹ 12500.0

Teja 15

13. Product already in cart functionality

a. In views.py import django messages as - from django.contrib import messages

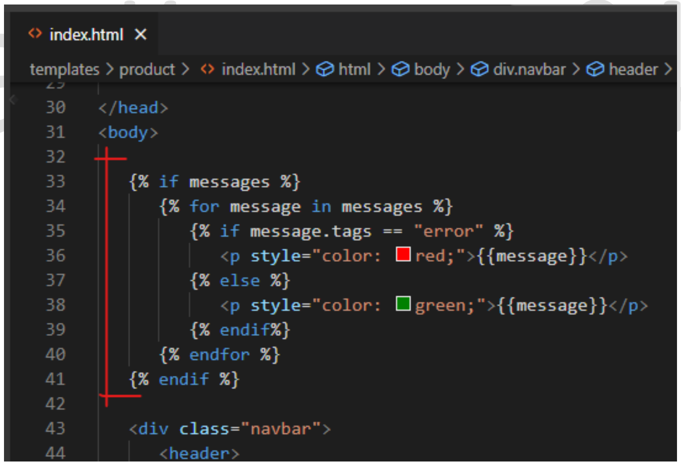
b. Update add_to_cart()and check whether that product already in cart



```python
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 alredy 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

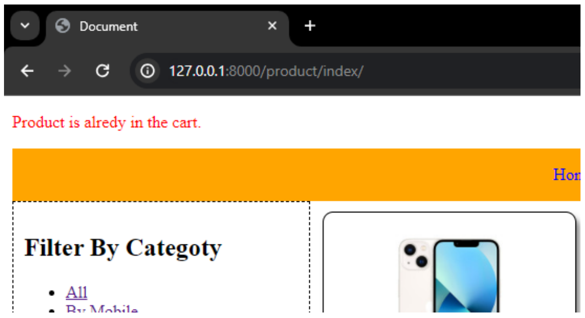
Tejas 8459859415



index.html
templates > product > index.html > html > body > div.navbar > header

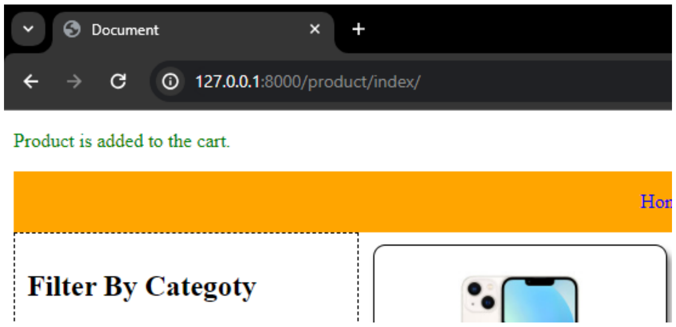
```html
</head>
<body>

    {% if messages %}
        {% for message in messages %}
            {% if message.tags == "error" %}
                <p style="color: red;">{{message}}</p>
            {% else %}
                <p style="color: green;">{{message}}</p>
            {% endif%}
        {% endfor %}
    {% endif %}

    <div class="navbar">
        <header>
```

d. Output



Product is alredy in the cart.

Filter By Categoty
- All
- By Mobile

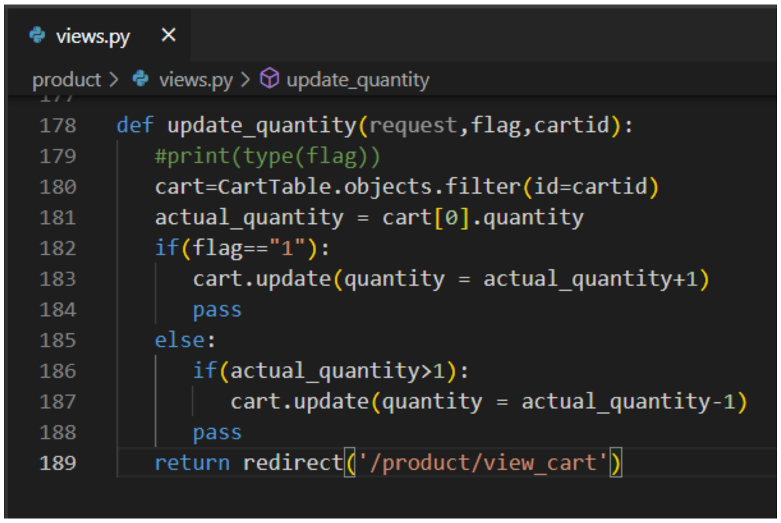


Product is added to the cart.

Filter By Categoty

14. Quantity inc/dec functionality

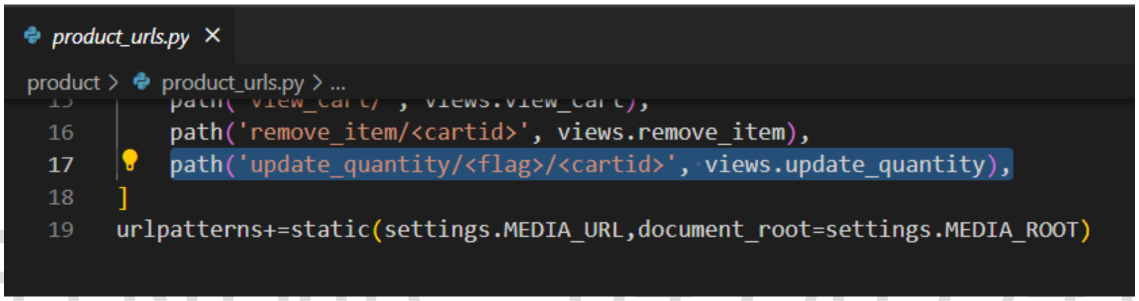
a. Creating view: update_qunatity() to change quantity



```
views.py
product > views.py > update_quantity
def update_quantity(request,flag,cartid):
    #print(type(flag))
    cart=CartTable.objects.filter(id=cartid)
    actual_quantity = cart[0].quantity
    if(flag=="1"):
        cart.update(quantity = actual_quantity+1)
        pass
    else:
        if(actual_quantity>1):
            cart.update(quantity = actual_quantity-1)
        pass
    return redirect('/product/view_cart')
```

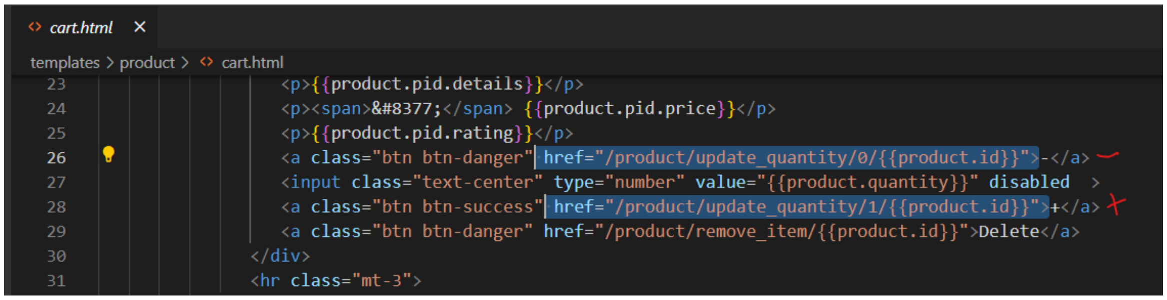
Here flag is 1 to increment and 0 is for decrement

b. Create url for above view



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

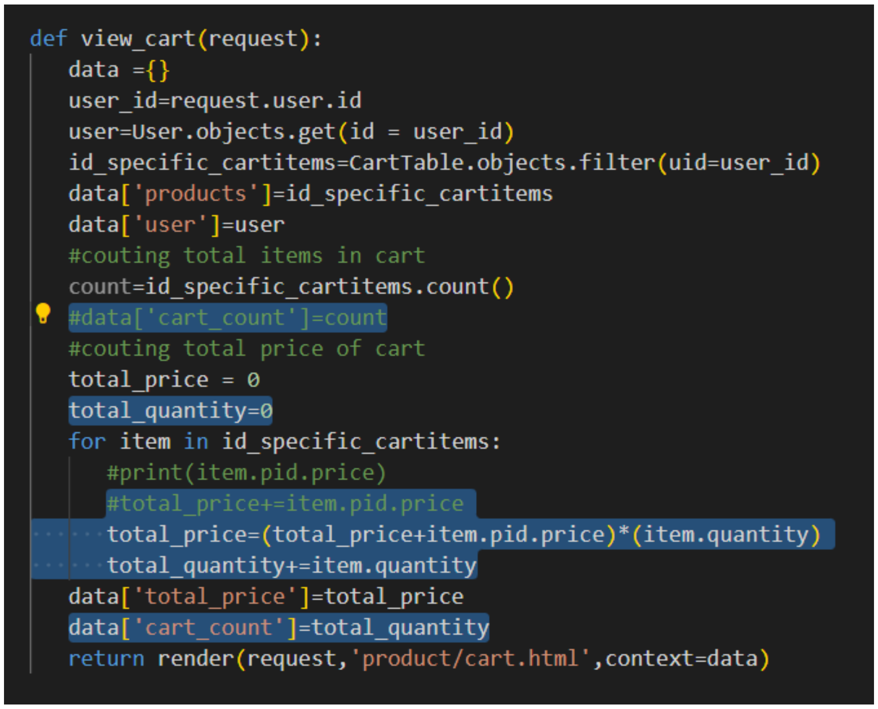
c. Use above url on plus(+) and minus(-) button in cart.html



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

d.

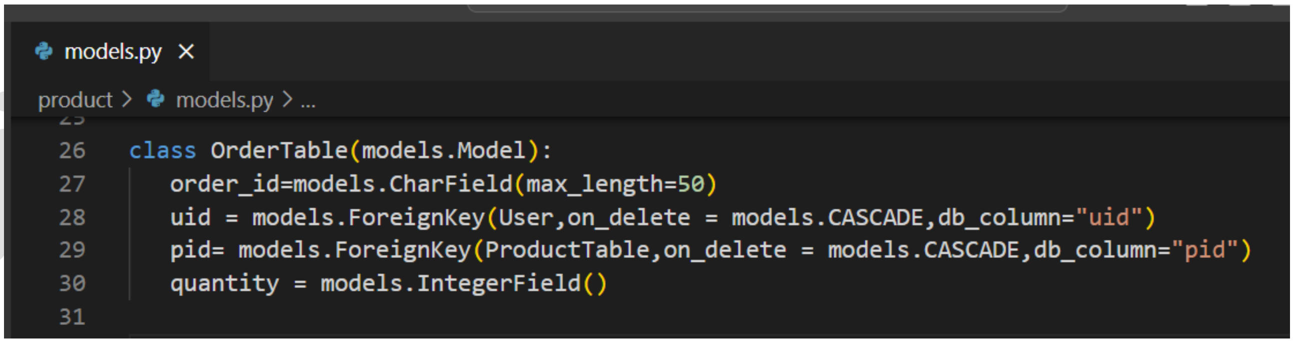
15. Changing item count and total price according to quantity



```python
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
    #couting total items in cart
    count=id_specific_cartitems.count()
    #data['cart_count']=count
    #couting 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

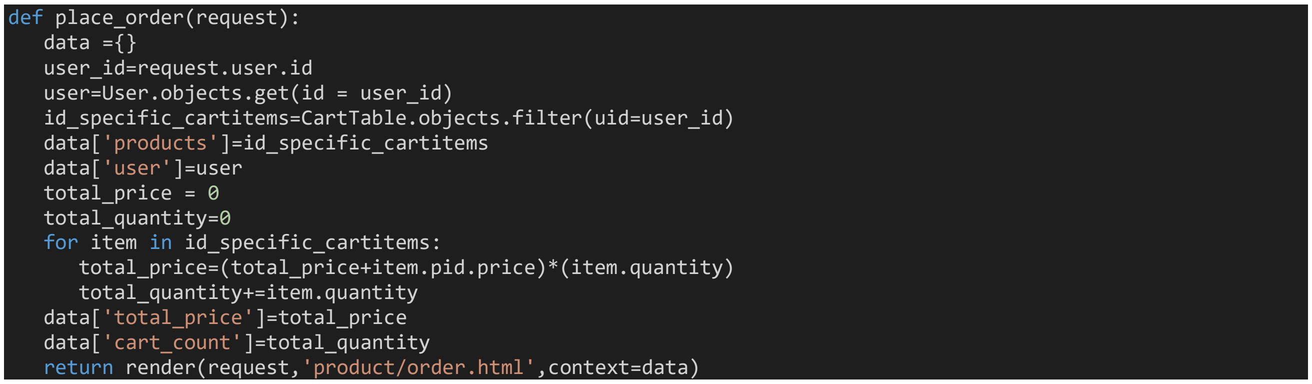
   a. We have to create OrderTable class in models.py



```python
class OrderTable(models.Model):
    order_id=models.CharField(max_length=50)
    uid = models.ForeignKey(User,on_delete = models.CASCADE,db_column="uid")
    pid= models.ForeignKey(ProductTable,on_delete = models.CASCADE,db_column="pid")
    quantity = models.IntegerField()
```

   b. MM

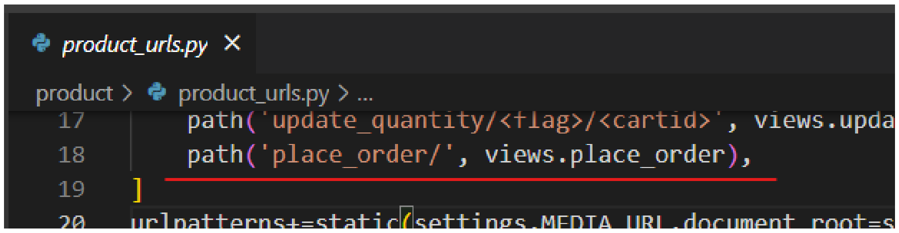
   c. Create view place_order()



```python
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



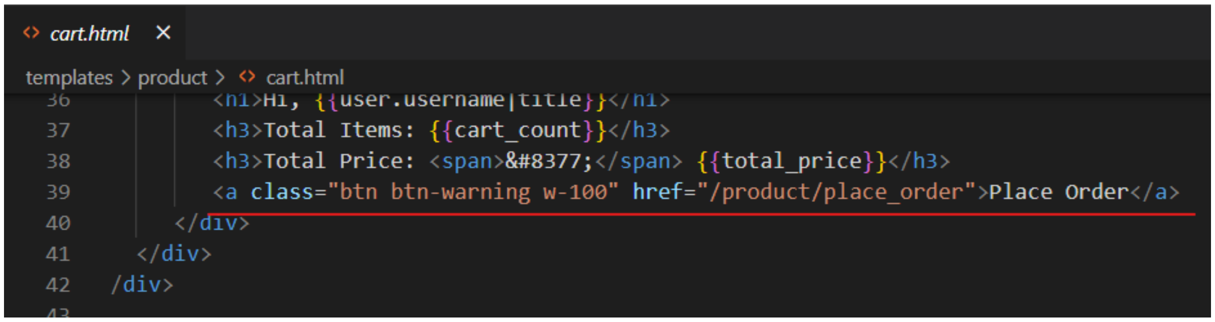
product_urls.py ×

product > product_urls.py > ...

```
    path('update_quantity/<flag>/<cartid>', views.upda
    path('place_order/', views.place_order),
]
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 ×

templates > product > cart.html

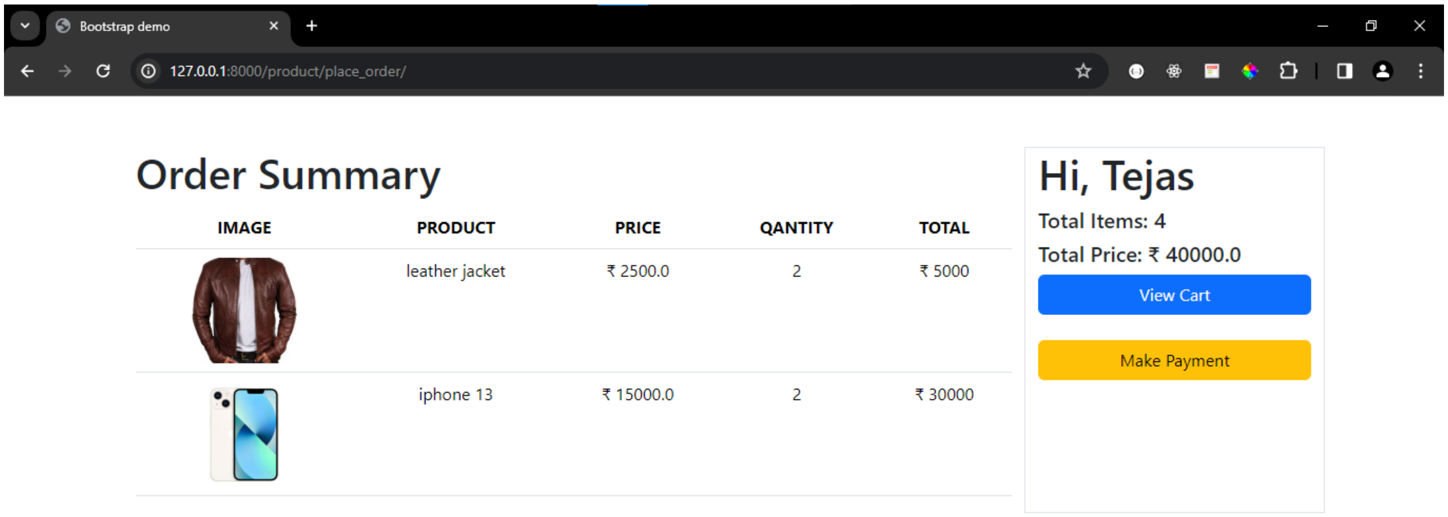
```
            <h1>Hi, {{user.username|title}}</h1>
            <h3>Total Items: {{cart_count}}</h3>
            <h3>Total Price: <span>&#8377;</span> {{total_price}}</h3>
            <a class="btn btn-warning w-100" href="/product/place_order">Place Order</a>
         </div>
      </div>
   /div>
```

f. Creating order.html page to show order details

Teja

```
<!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-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">QANTITY</th>
                   <th scope="col">TOTAL</th>
                 </tr>
               </thead>
               <tbody>
                  {% for product in products %}
                 <tr class="text-center">
                   <th scope="row">
                     <img src="http://127.0.0.1:8000/product/media/{{product.pid.image}}" alt=""
height="100px" width="100px">
                   </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> -->
                   </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>
            <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-C6RzsynM9kWDrMNeT87bh95OGNyZPhcTNXj1NW7RuBCsyN/o0jlpcV8Qyq46cDfL"
crossorigin="anonymous"></script>
  </body>
</html>
```

g. Output :



Order Summary

IMAGE	PRODUCT	PRICE	QANTITY	TOTAL
	leather jacket	₹ 2500.0	2	₹ 5000
	iphone 13	₹ 15000.0	2	₹ 30000

Hi, Tejas

Total Items: 4

Total Price: ₹ 40000.0

Tejas Kasare - 8459859415

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

d. Create model