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 > ...
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

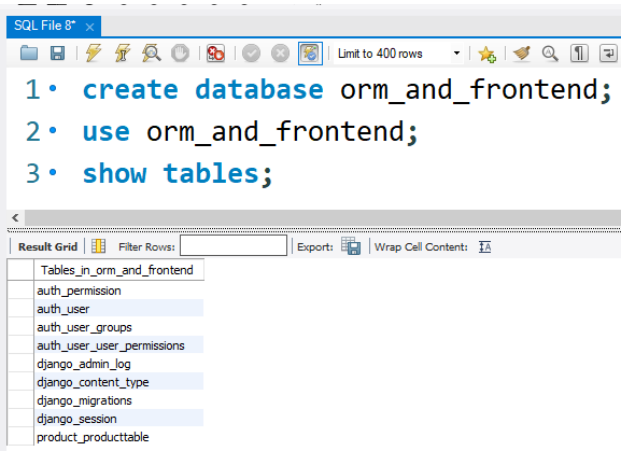
```
product > models.py > ProductTable
1   from django.db import models
2
3   # Create your models here.
4   class ProductTable(models.Model):
5       name = models.CharField(max_length=50)
6       price = models.FloatField()
7       details=models.CharField(max_length=150)
8       category = models.IntegerField()
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

```
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
    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)



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

```python
# models.py
# product > models.py > ProductTable > __str__
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"
```

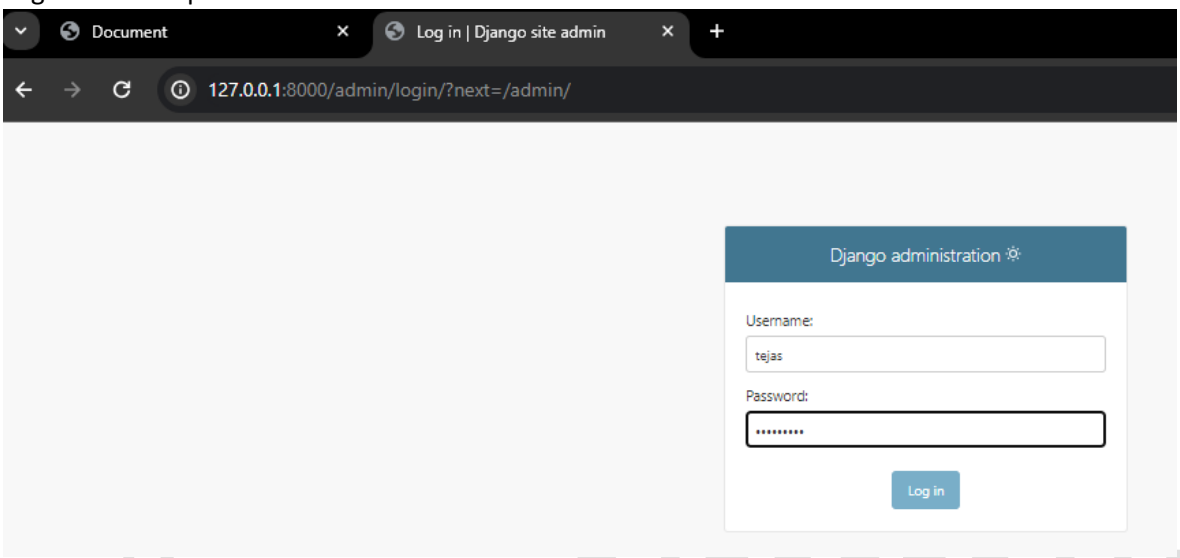
    b. Since we have made changes in model, we have to do makemigrations and migrate

```
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
```

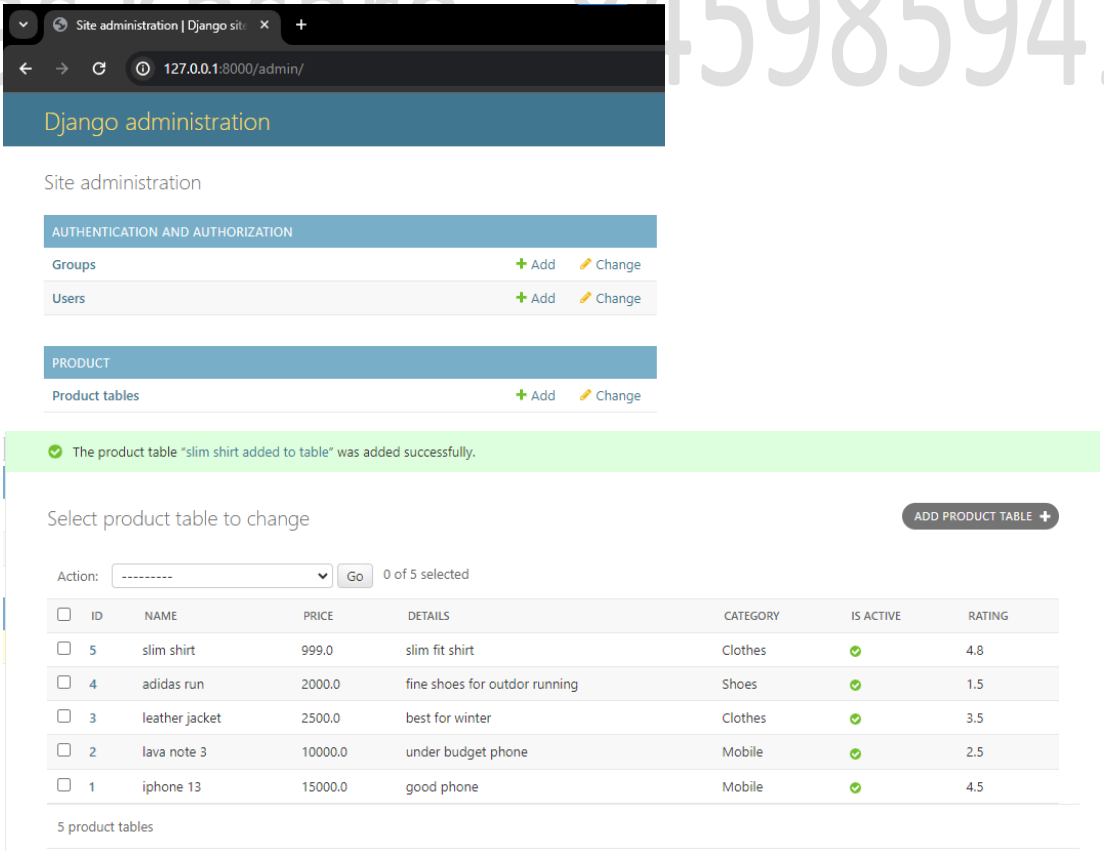
12. Adding some products into product table from admin panel
    a. Create superuser

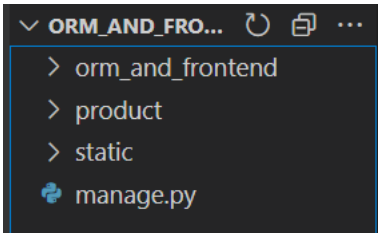
```
C:\TEJAS KASARE (Very imp folder)\my notes\Django\orm_and_frontend>python manage.py createsuperuser
Username (leave blank to use 'admin'): tejas
Email address: tejas@gmail.com
Password:
Password (again):
Superuser created successfully.
```

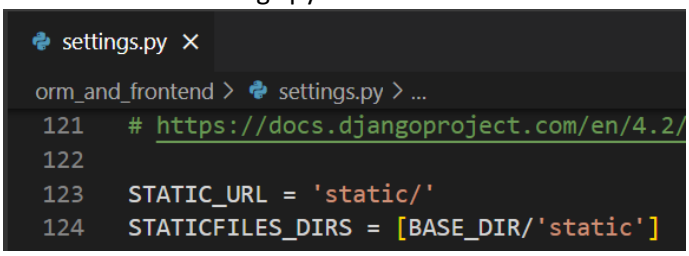
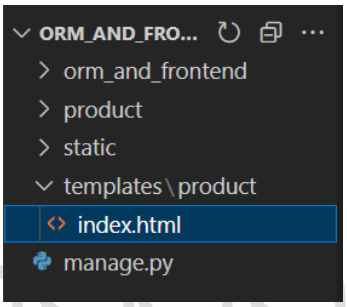
    b. runserver
    c. Login to admin panel

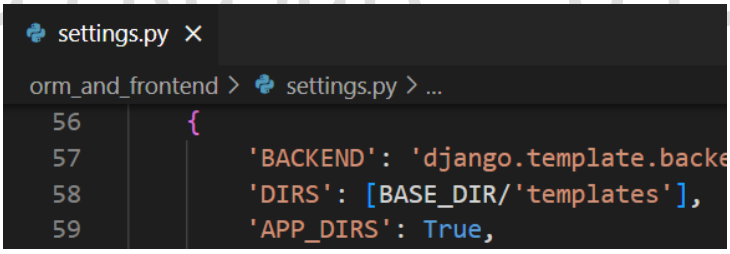


    d. Add 4-5 procducts (min 1 of each catergory)



13. Creating static folder to store css file
    a. 

14. Register static folder in settings.py
    a. 
```
settings.py
orm_and_frontend > settings.py > ...
121  # https://docs.djangoproject.com/en/4.2/
122
123  STATIC_URL = 'static/'
124  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. 
```
settings.py
orm_and_frontend > settings.py > ...
56      {
57          'BACKEND': 'django.template.backe
58          'DIRS': [BASE_DIR/'templates'],
59          '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

```css
.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;

}
```

```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

```
<!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 > ...
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

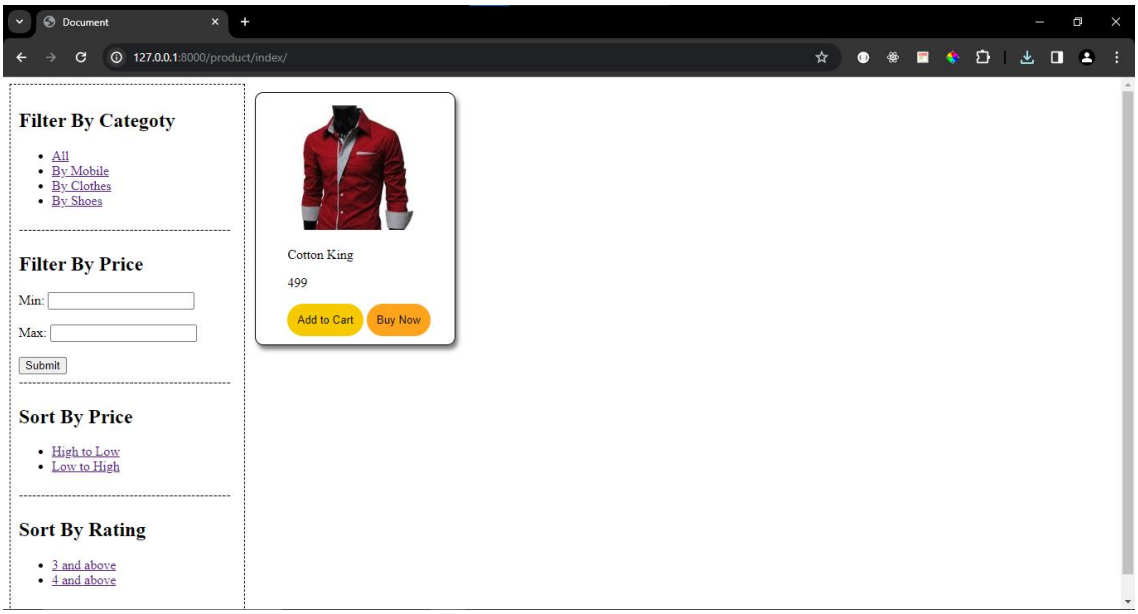
```
product > product_urls.py > ...
1   from django.urls import path
2   from product import views
3
4   urlpatterns = [
5       path('index/', views.index),
6   ]
```

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

urls.py

```
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

```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

```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)

2. Implementing Filter by category logic
    a. Create view

views.py

```python
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

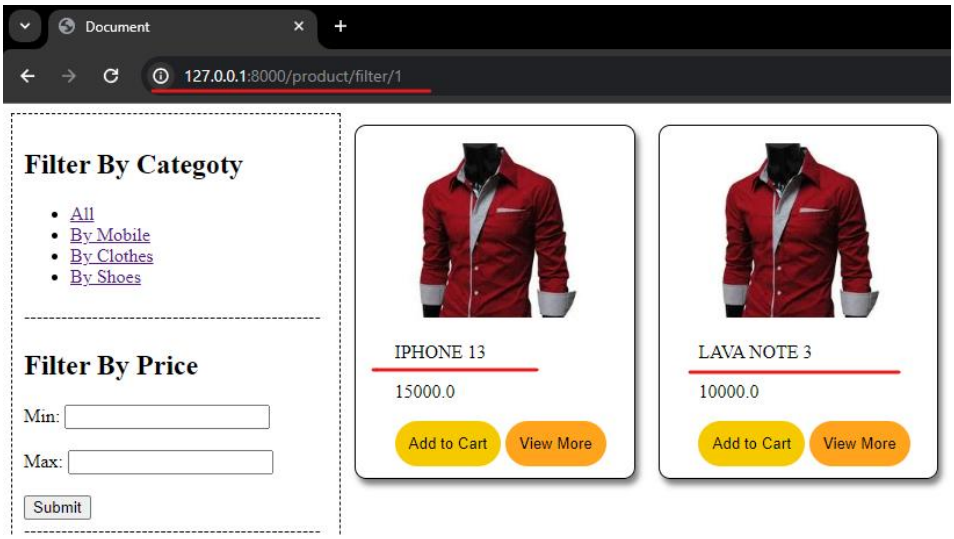
```python
from django.urls import path
from product import views

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


    c. Use url in index.html

index.html

```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 :



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

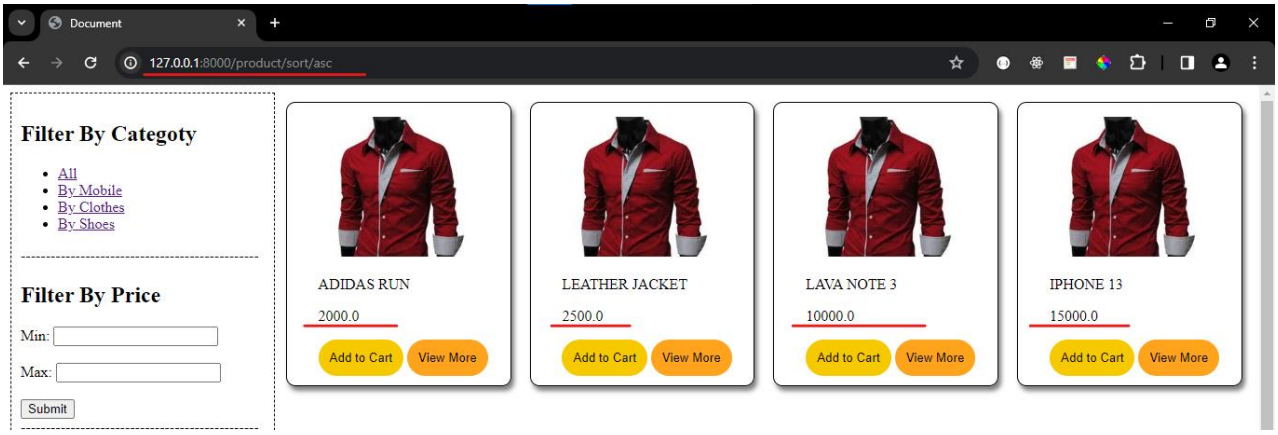
```
product > product_urls.py > ...
1  from django.urls import path
2  from product import views
3
4  urlpatterns = [
5      path('index/', views.index),
6      path('filter/<category_value>', views.filter_by_category),
7      path('sort/<sort_value>', views.sort_by_price),
8  ]
```

    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/desc">High to Low</a></li>
39                  <li><a href="/product/sort/asc">Low to High</a></li>
40              </ul>
41          </div>
42  <div>--------------------------------------------------</div>
```

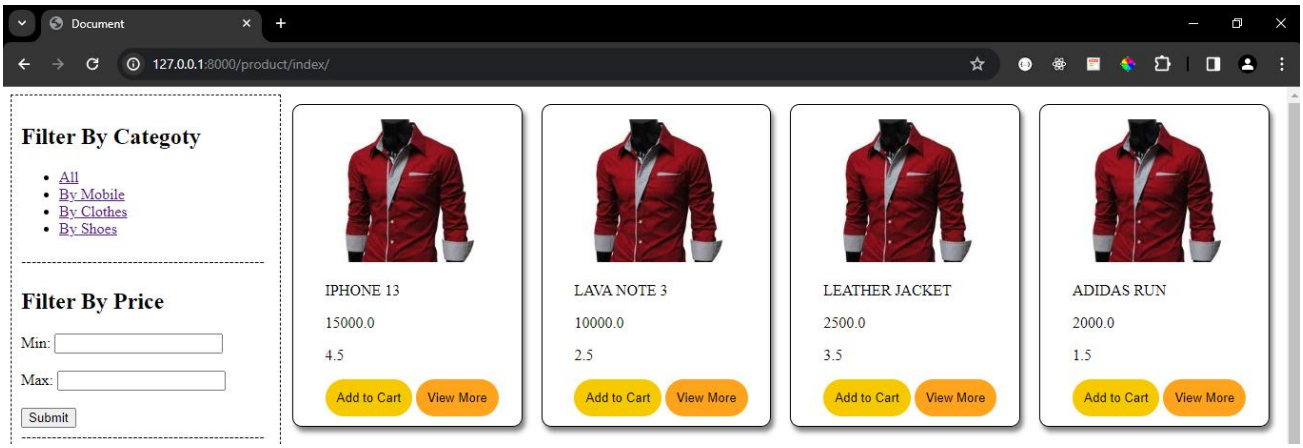
    d. Result



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



    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      filterd_products=ProductTable.objects.filter(q1 & q2)
44      data['products']=filterd_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

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

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      filterd_products=ProductTable.objects.filter(q1 & q2 & q3)
56      data['products']=filterd_products
57      return render(request,'product/index.html',context=data)
```


   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       path('rating/<rating_value>', views.filter_by_rating),
9       path('price', views.filter_by_price_range),
10  ]
```

IMPORTANT : there is no / after price

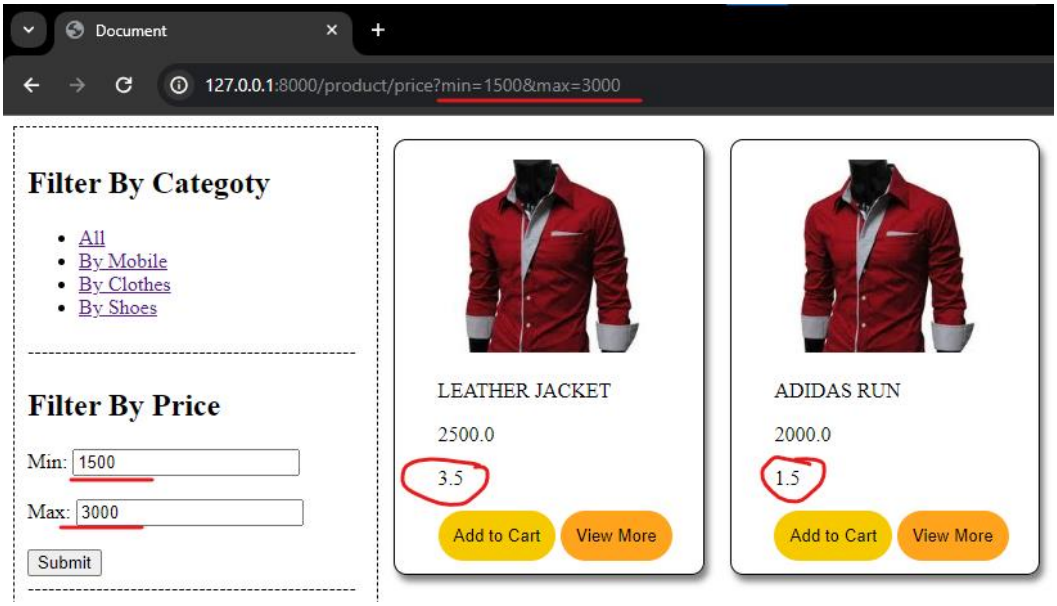
   c. User url in index.html

index.html

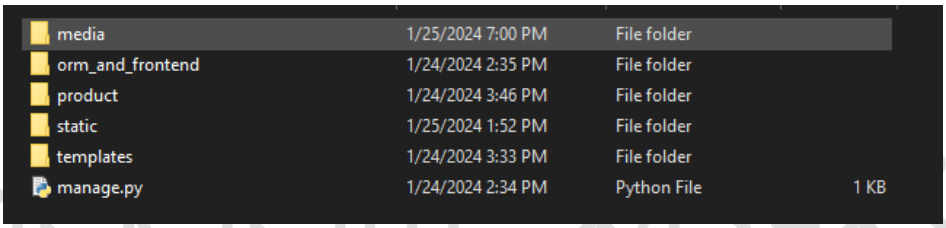
templates > product > index.html

```
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 :

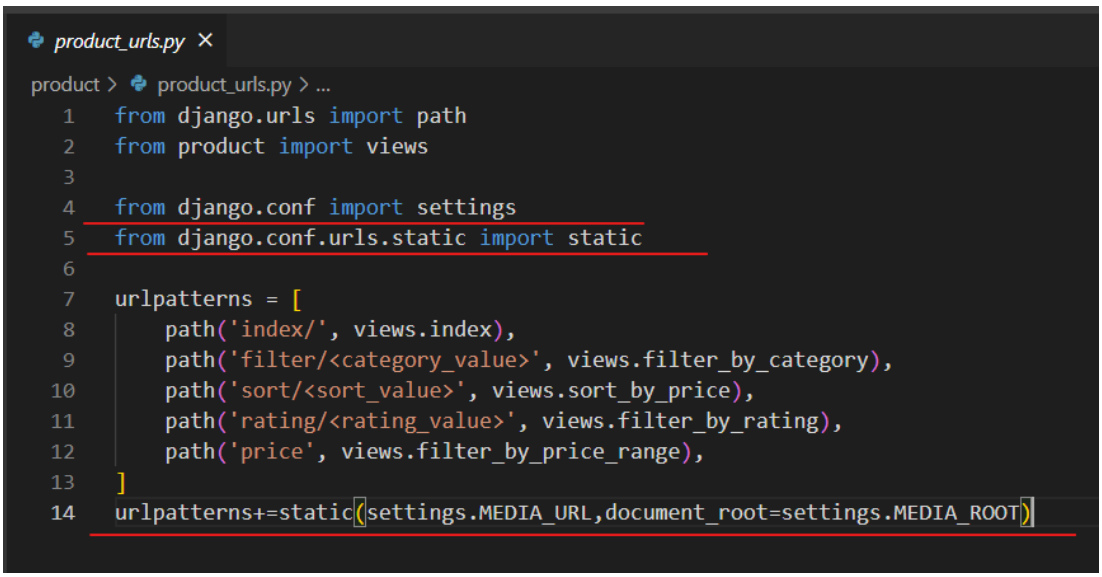


6. Uploading Image
   a. Create media project in project like we create templates folder



   b. Register media folder in settings.py

   c. 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

```
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. 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 > ...
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

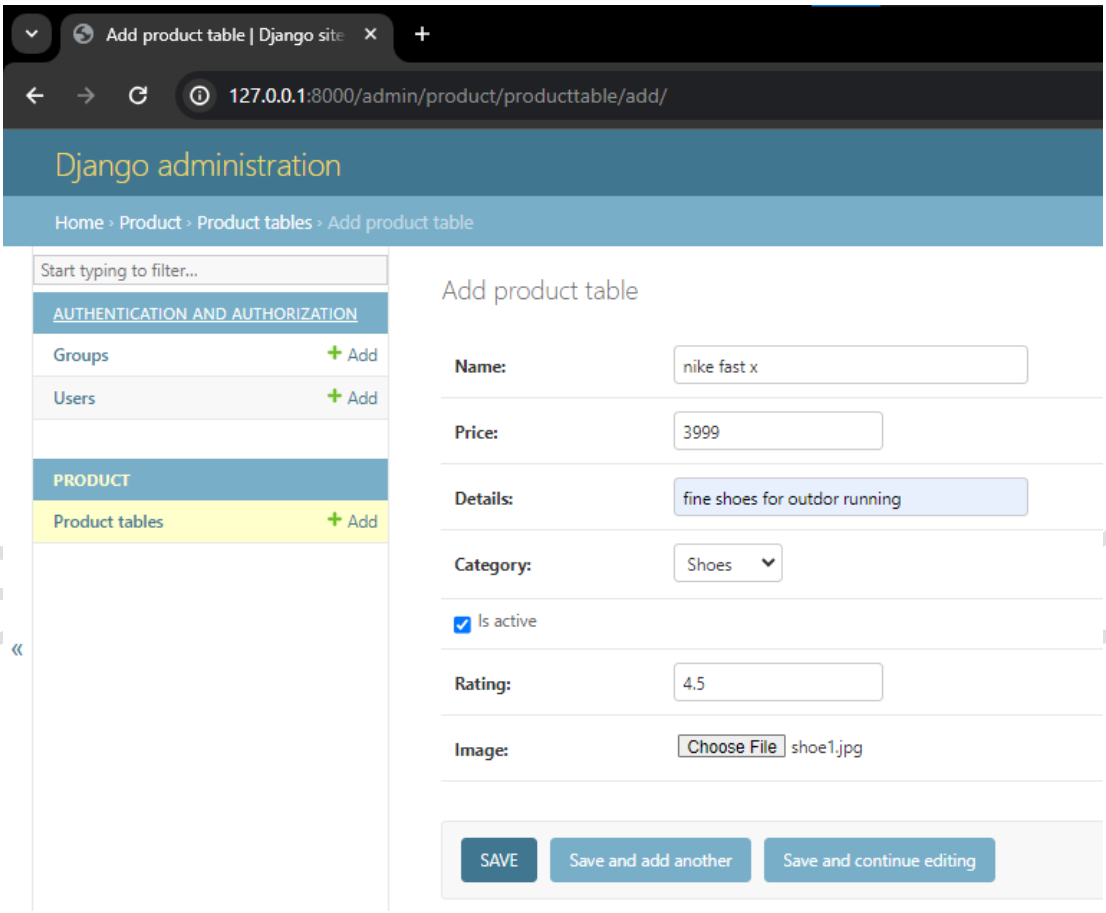
i. Check table in workbench, you will get image filed with default value 0

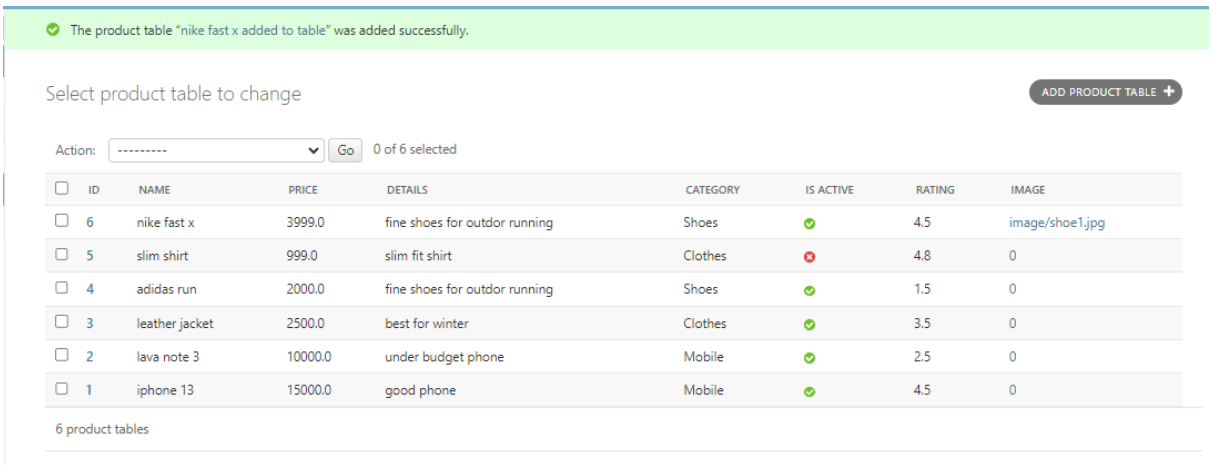
```
5 •  select * from product_producttable;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap

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

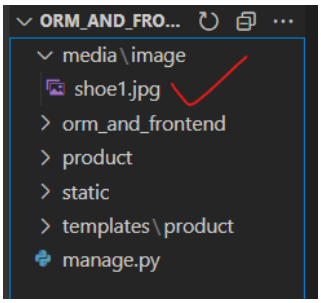




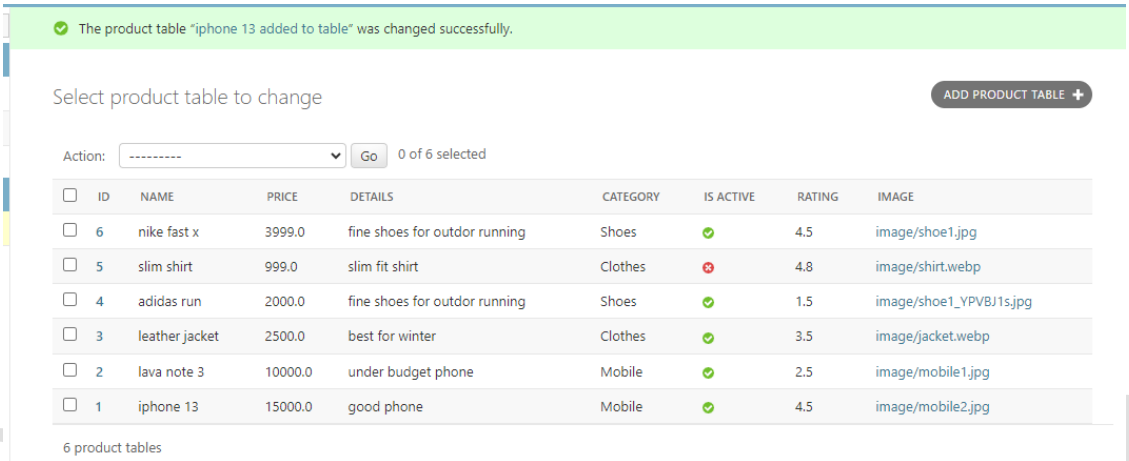
k. Check in media folder –

i. You will get image folder

ii. In image folder you will get your uploaded image



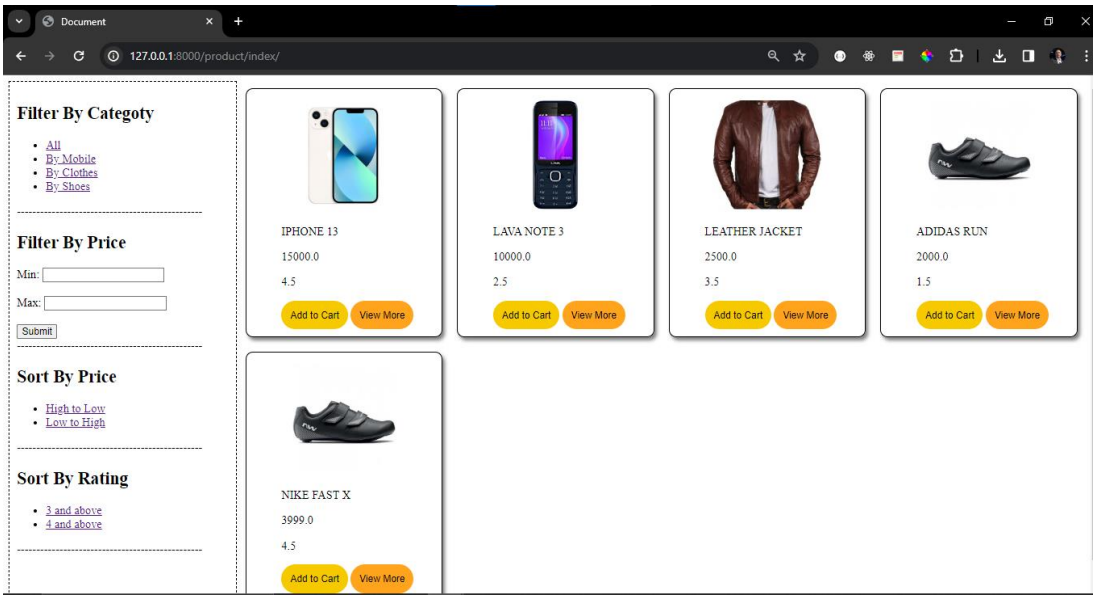
l. Add image for other products



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

```
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                  <!-- <img src="{% static 'images/shirt.webp' %}" alt=""> -->
57                  <img src="http://127.0.0.1:8000/product/media/{{product.image}}" alt="">
58                  <div class="card-text">
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

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>
```

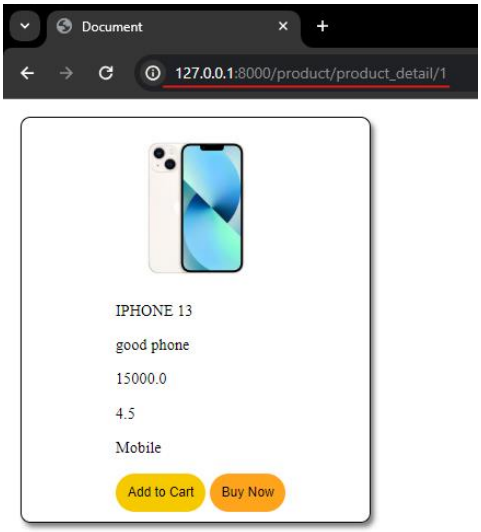
    c. Create url for above view

```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>
```

e. Check output :

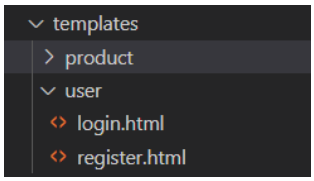


8. Add to cart functionality :

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



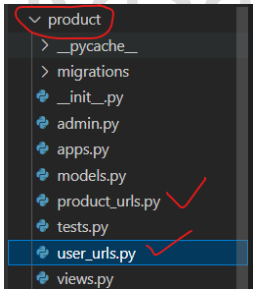
ii. code for register.html (not image. You can copy paste)

```
<!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)

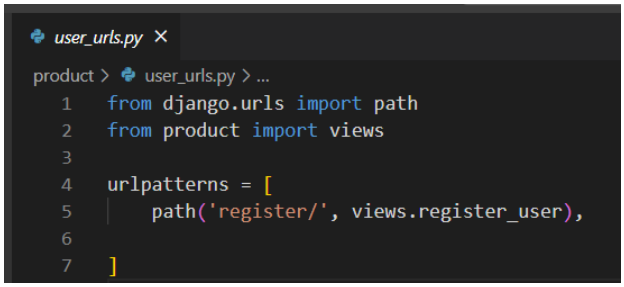
```
def register_user(request):
   data={}
   if request.method=="POST":
      uname=request.POST['username']
      upass=request.POST['password']
      uconf_pass=request.POST['password2']
      #implementing validation
      if (uname=='' or upass =='' or uconf_pass ==''):
         data['error_msg']='Fileds 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')
```


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

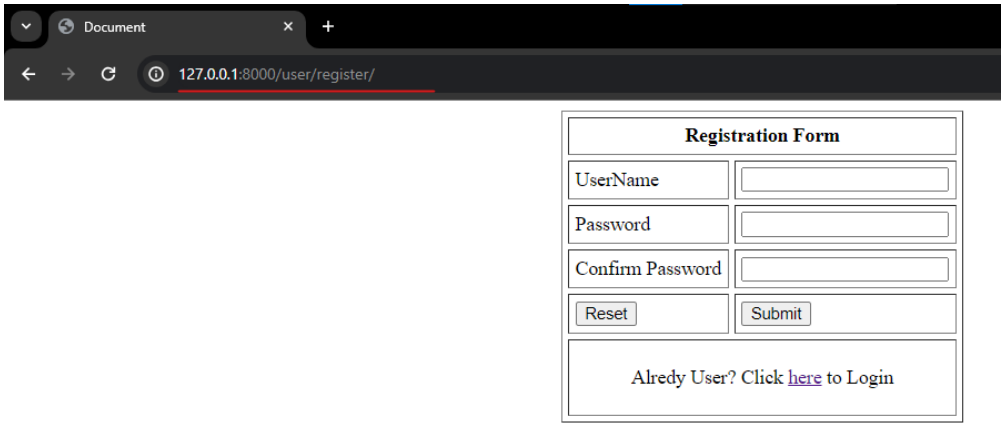


Register this user_urls.py in urls.py (project level)

Finally, url to display above resgistration form



Output



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)

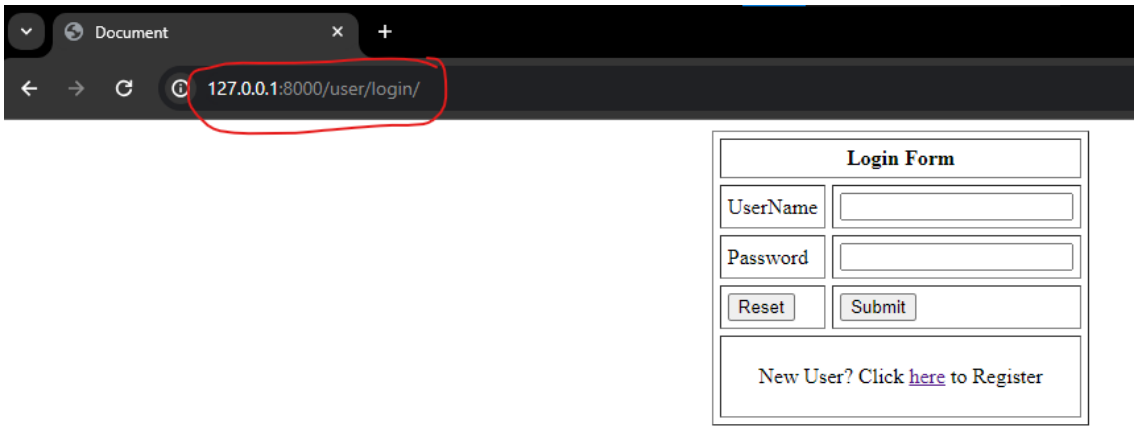
```
def login_user(request):

 data={}
 if request.method=="POST":
    uname=request.POST['username']
    upass=request.POST['password']
    #implementing validation
    if (uname=='' or upass ==''):
       data['error_msg']='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

user_urls.py

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

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


Output



c. Logout

i. Create view for logout

views.py

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

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

templates > product > index.html

```
6   <meta name="viewport" content="width=device-width
7   <title>Document</title>
8   <link rel="stylesheet" href="{% static 'css/prod
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>
```

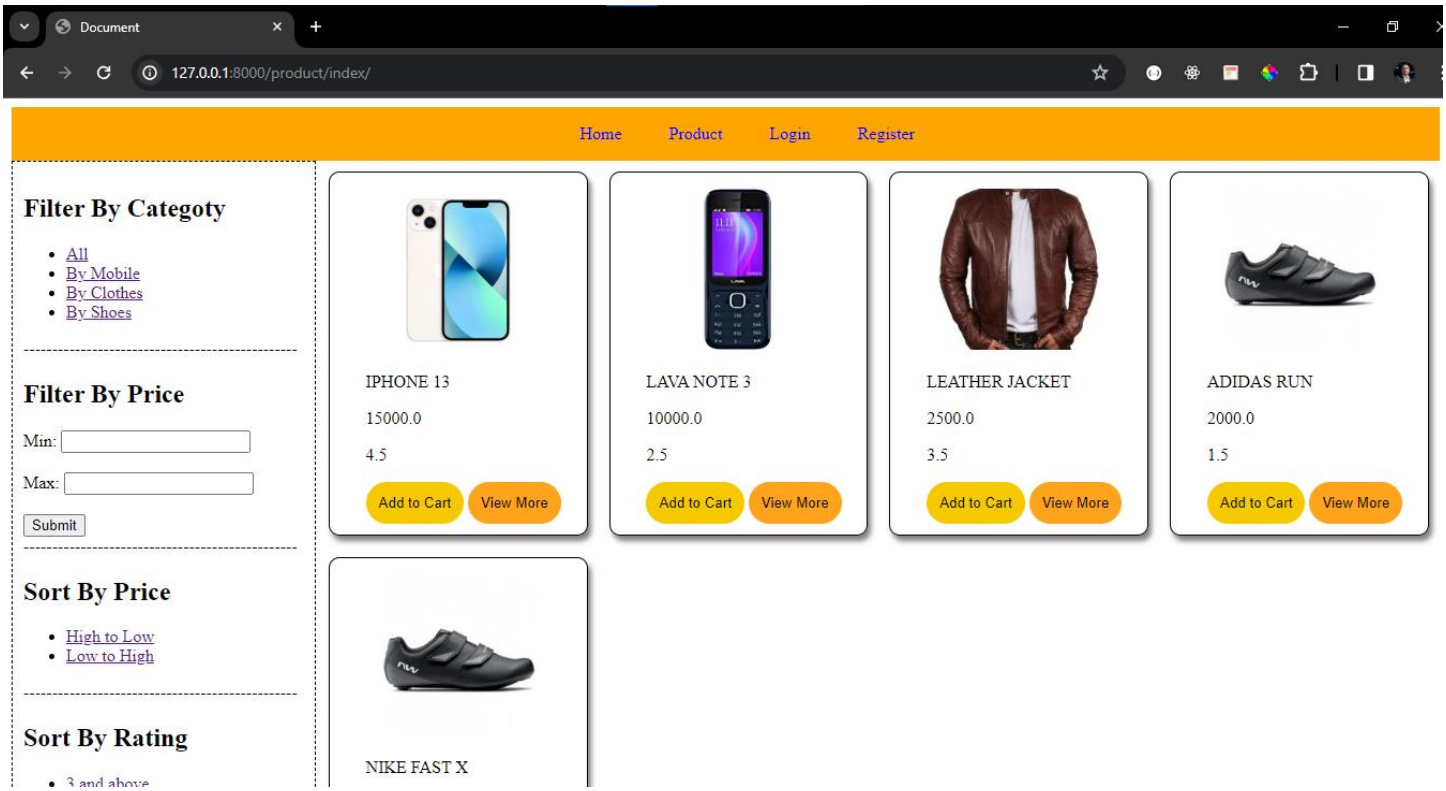
Navbar code

index.html

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
45          </ul>
46        </nav>
47        </header>
48    </div>
49    <div class="container">
50      <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

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)
```

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!!

Created By – Tejas Kasare(Trainer and Developer) ☏ +91 845 985 9415 ✉ tejaskasare14@gmail.com

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

```
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>
```

    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

```
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 %}
```

        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")
```

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

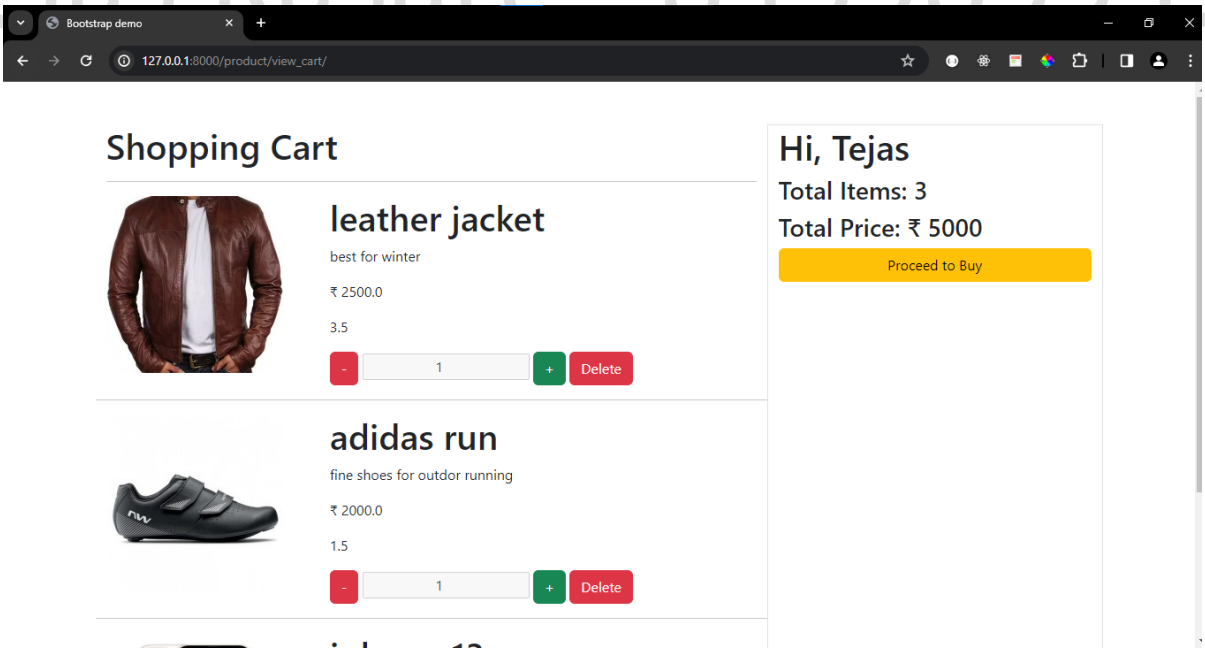
```
12      path('price', views.filter_by_price_range),
13      path('product_detail/<pid>', views.product_detail),
14      path('add_to_cart/<pid>', views.add_to_cart),
15      path('view_cart/', views.view_cart),
16  ]
17  urlpatterns+=static(settings.MEDIA_URL,document_root=settin
```

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

index.html

```
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

    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

```
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.
```

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

cart.html

```
24          <p><span>&#8377;</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

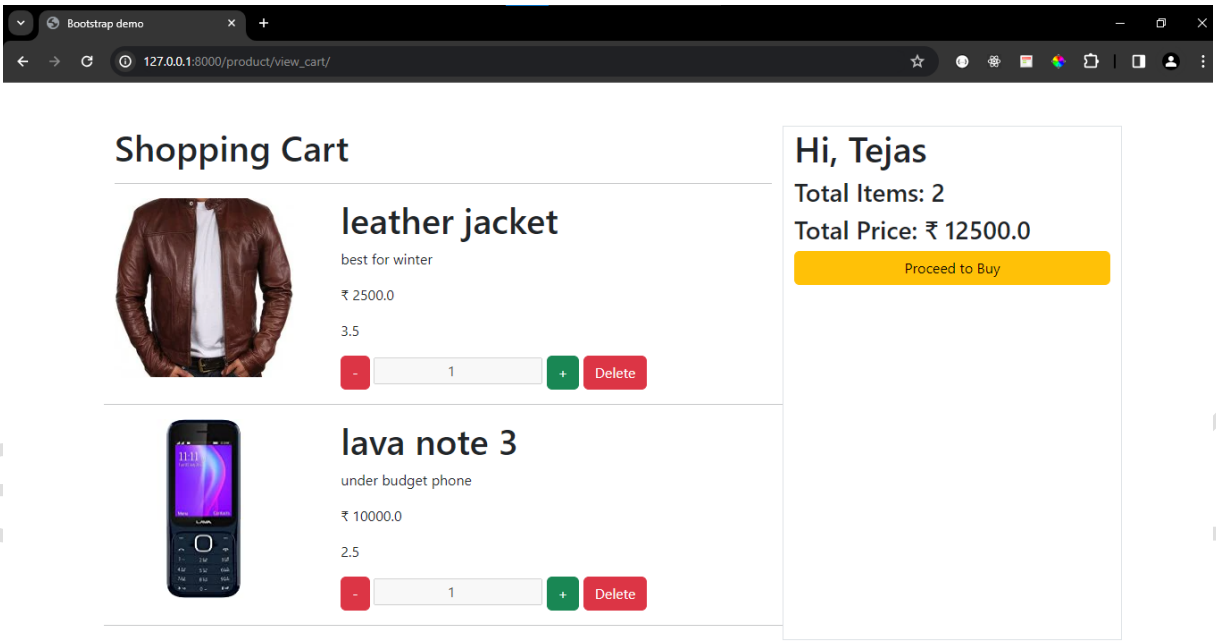
    a. We have to write a logic to calculate total price and total item in view_cart() function

```
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

```
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

```
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
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 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>
```

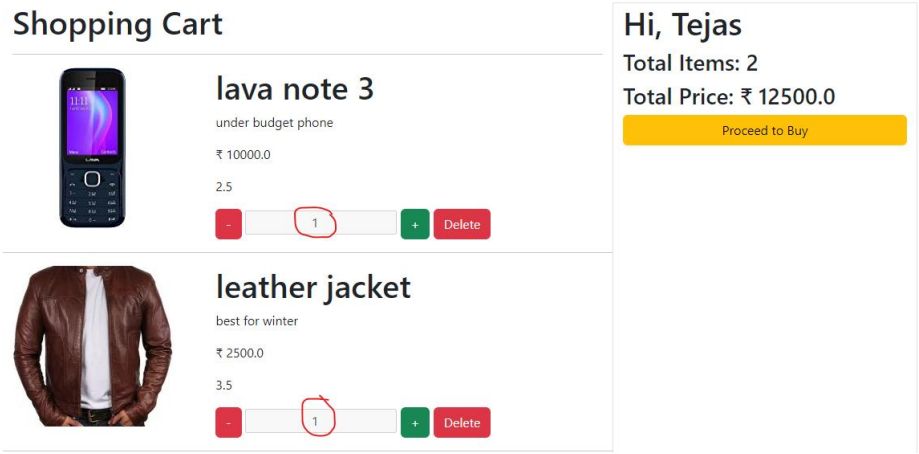
c. Fetching quantity and displaying in cart.html

cart.html

templates > product > cart.html

```
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
    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")
```

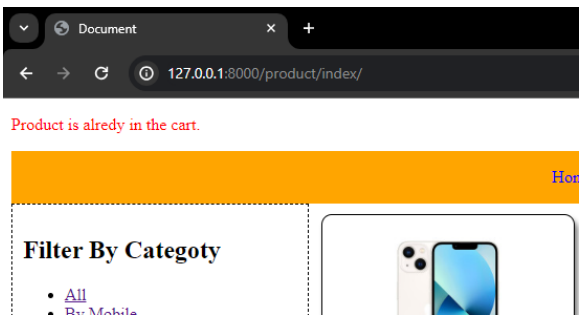
c. Display message in html

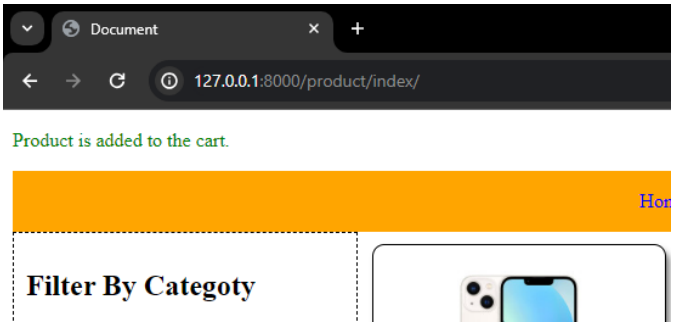
index.html

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





14. Quantity inc/dec functionality
    a. Creating view: update_qunatity() to change quantity

views.py

product > views.py > update_quantity

```python
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 > ...

```python
    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

models.py ×

product > 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

```
<!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

View Cart

Make Payment

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

```
{% 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 insterd into Customer

views.py

product > views.py > register_user

```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
            customer=CustomerDetails.objects.create(uid=user)
            customer.save()
            # return HttpResponse("Registraion done")
            return redirect('/user/login')
    return render(request,'user/register.html')

```

e. Create view to update customer details edit_profile()

```python
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

product > user_urls.py > ...

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

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

h. Use url in navar Edit Profile button

index.html

templates > product > index.html

```
<li>  <a href="/product/index">Product</a>  </li>

{% if user.is_authenticated %}
    <li>  <a href="/user/logout">Logout</a>  </li>
    <li>  <a href="/product/view_cart">View cart <span style="ba
    <li>  <a href="/user/edit_profile">Edit Profile </a></li>
{% else %}
    <li>  <a href="/user/login">Login</a>  </li>
```

18. Adding customer address on order.html page

a. In place_order() view, fetch customer based on user id,Pass this cutomer to order.html

```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)
    customer = CustomerDetails.objects.get(uid = user_id)
    data['customer']=customer
    data['products']=id_specific_cartitems
```

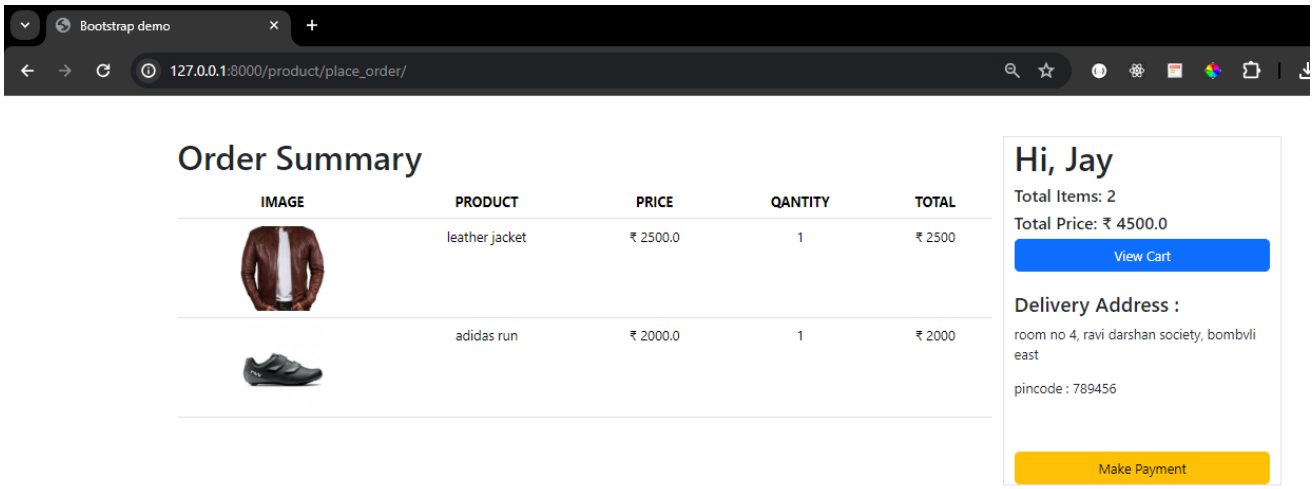
b. Access address and pincode from customer object and display

order.html

templates > product > order.html > html > body > div.container.mt-5 > div.row

```
</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>
    <h4>Delivery Address :</h4>
    <p>{{customer.full_address}}</p>
    <p>pincode : {{customer.pincode}}</p>
    <br><br>
    <a class="btn btn-warning w-100" href="/product/place_order">Make Payment</a>
</div>
```

c. Output



19. Creating account with razor pay
    a. Visit website
    b. Signup
    c. Verify OTP
    d. Name
    e. Answer the questions asked
    f. After completing registration, don't click on COMPLETE KYC (and don't close the tab)
    g. Open another tab and search for razorpay dashboard > login
    h. In razorpay dashboard
        i. Account and settings
        ii. Website and app settings
        iii. API keys
        iv. Generate test keys and downlad it

20. Make Payment functionality
    a. Create view makepayment() in views.py

```
def make_payment(request):
    return HttpResponse("Payment done")
```

    b. Create url for above view

```
path('make_payment/', views.make_payment),
```

    c. Use above url on "Make Payment" button in order.html

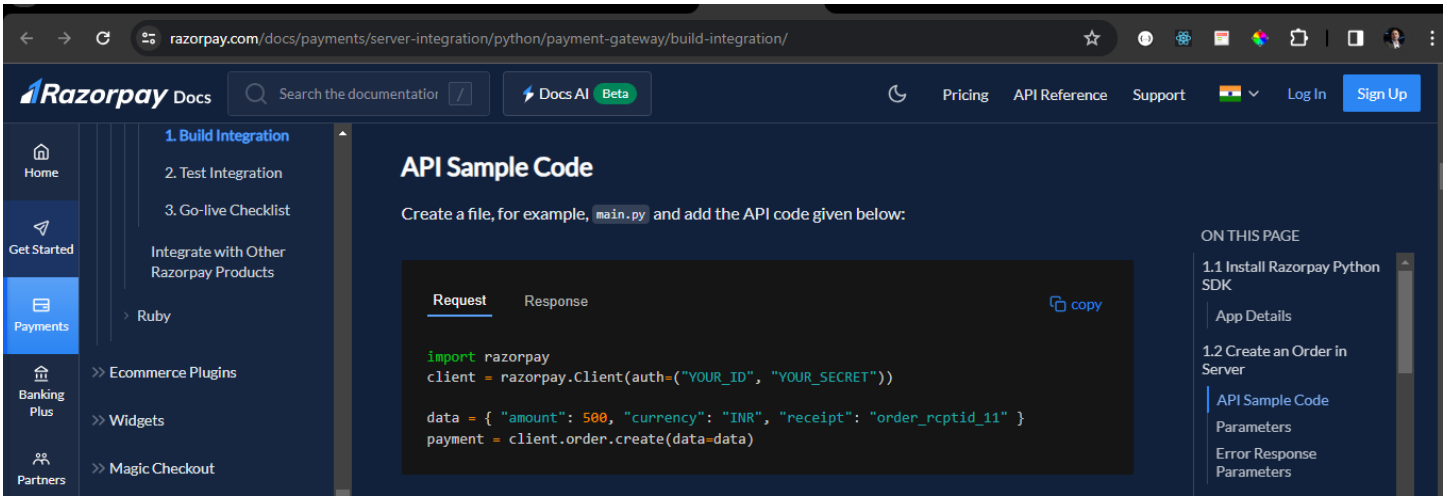
order.html

templates > product > order.html

```
47                  <h4>Delivery Address :</h4>
48                  <p>{{customer.full_address}}</p>
49                  <p>pincode : {{customer.pincode}}</p>
50                  <br><br>
51                  <a class="btn btn-warning w-100" href="/product/make_payment">Make Payment</a>
52              </div>
```

    d. Go to google and search for razrorpay python server integration document or visit https://razorpay.com/docs/payments/server-integration/python/install/
    e. Stop server and install razorpay as
        i. >pip install razorpay
    f. Start server

g. If you got error stating : ModuleNotFoundError: No module named 'pkg_resources'
    i. Do following :
        1. Stop server and run >pip install --upgrade setuptools
h. Connecting with razor pay
    i. Get API sample code from website and add it to the view




```
import razorpay
def make_payment(request):
    client = razorpay.Client(auth=("rzp_test_ls3ufyYec3WWtv", "Cm3iRE0s8JpvOE5hTg9dS8tP"))
    data = { "amount": 500, "currency": "INR", "receipt": "order_rcptid_11" }
    payment = client.order.create(data=data)
    print(payment)
    return HttpResponse("Payment done")
```

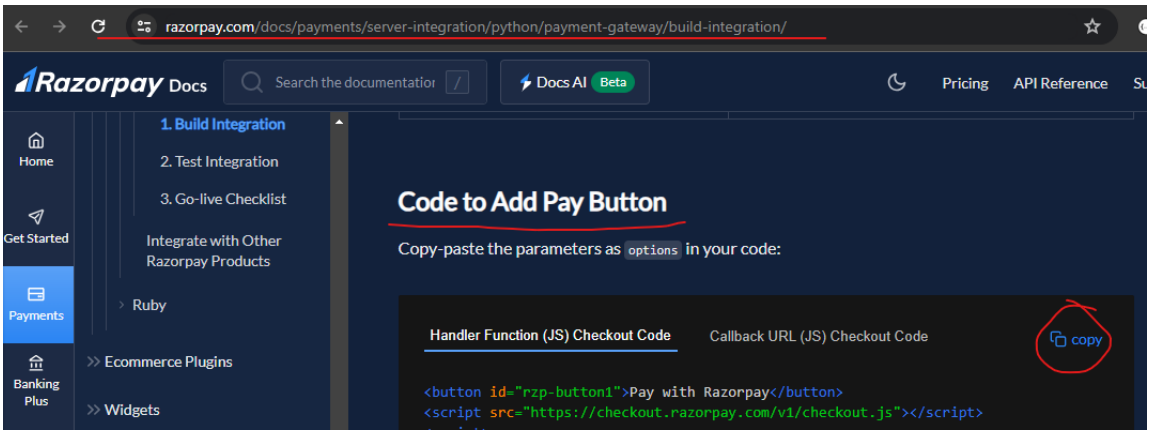

    ii. Check print's output on console

```
[15/Feb/2024 15:02:15] "GET /product/make_payment HTTP/1.1" 301 0
{'id': 'order_Nb6gnotFP2xC5C', 'entity': 'order', 'amount': 500, 'amount_paid': 0, 'amount_due': 500, '
currency': 'INR', 'receipt': 'order_rcptid_11', 'offer_id': None, 'status': 'created', 'attempts': 0, '
notes': [], 'created_at': 1707989535}
[15/Feb/2024 15:02:16] "GET /product/make_payment/ HTTP/1.1" 200 12
```

i. Changing total amount

```
import razorpay
def make_payment(request):
    #getting total amount
    user_id=request.user.id
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    total_price = 0
    for item in id_specific_cartitems:
        total_price=(total_price+item.pid.price)*(item.quantity)

    client = razorpay.Client(auth=("rzp_test_ls3ufyYec3WWtv", "Cm3iRE0s8JpvOE5hTg9dS8tP"))
    data = { "amount": total_price*100, "currency": "INR", "receipt": "order_rcptid_11" }
    payment = client.order.create(data=data)
    print(payment)
    return HttpResponse("Payment Done")
```

j. Scroll down the razorpay webpage (the page from which we copy sample API code). You will get “Code to add pay button”



i. Copy the code

ii. Create pay.html file in templates > product

iii. Render pay.html from our view make_payment

```python
import razorpay
def make_payment(request):
    #getting total amount
    user_id=request.user.id
    id_specific_cartitems=CartTable.objects.filter(uid=user_id)
    total_price = 0
    for item in id_specific_cartitems:
        total_price=(total_price+item.pid.price)*(item.quantity)

    client = razorpay.Client(auth=("rzp_test_ls3ufyYec3WWtv", "Cm3iRE0s8JpvOE5hTg9dS8tP"))
    data = { "amount": total_price*100, "currency": "INR", "receipt": "order_rcptid_11" }
    payment = client.order.create(data=data)
    print(payment)
    #return HttpResponse("Payment Done")
    return render(request,'product/pay.html',context=data)
```

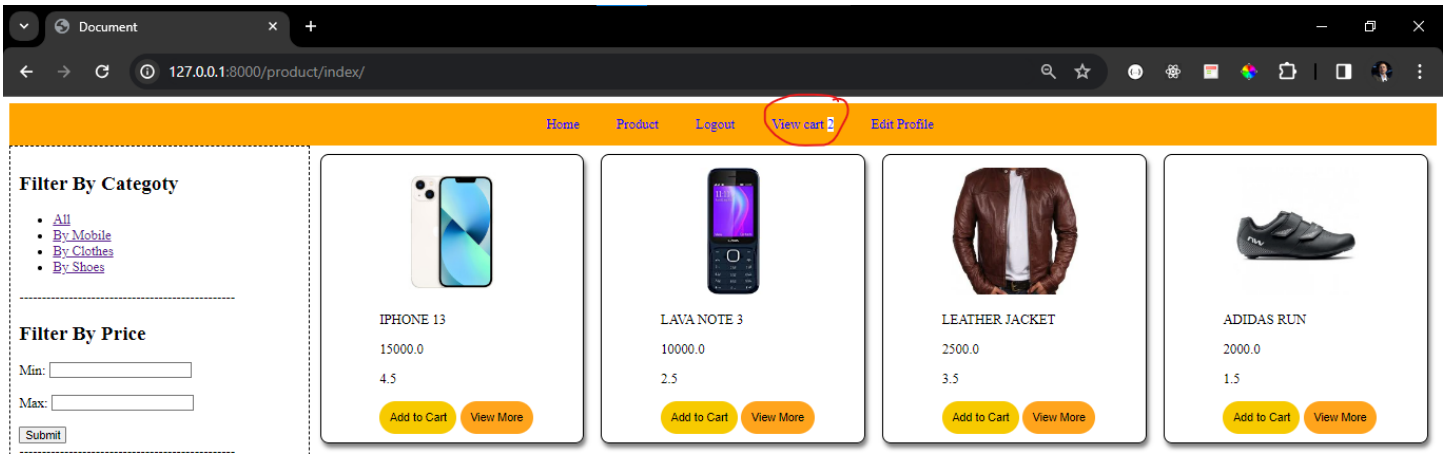

iv. Paste the copied code into body tag and do following changes

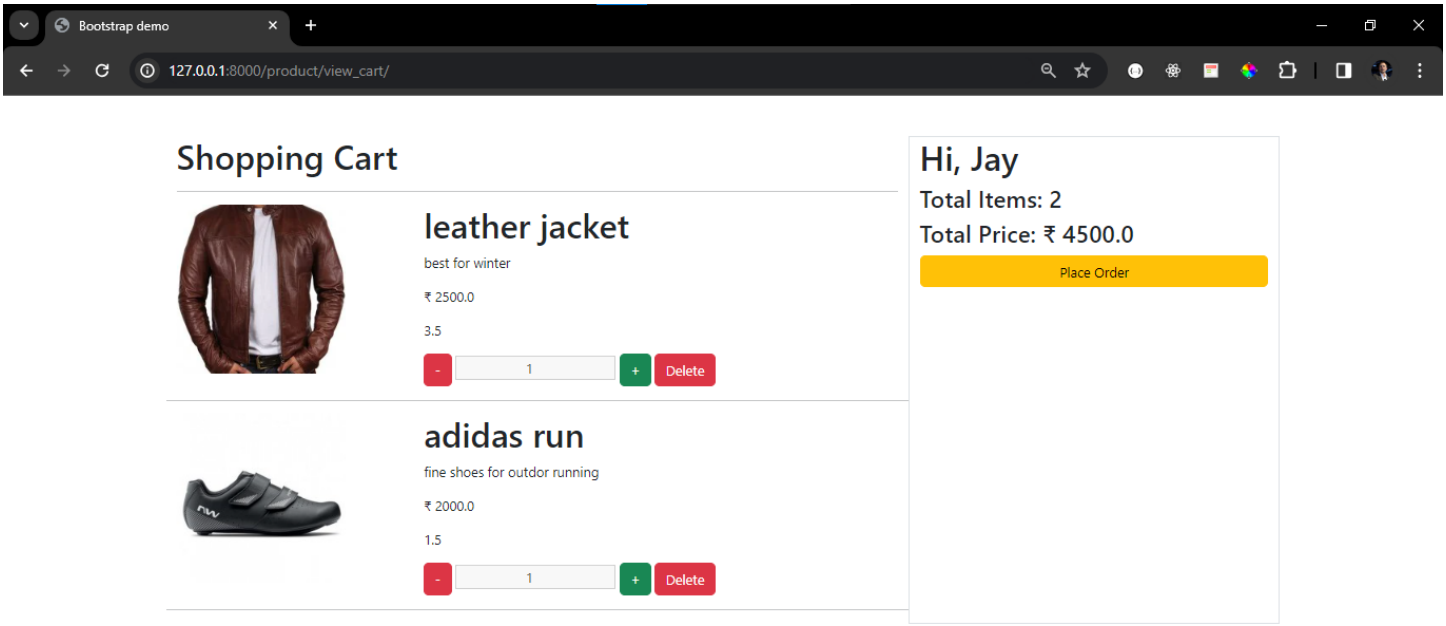
pay.html

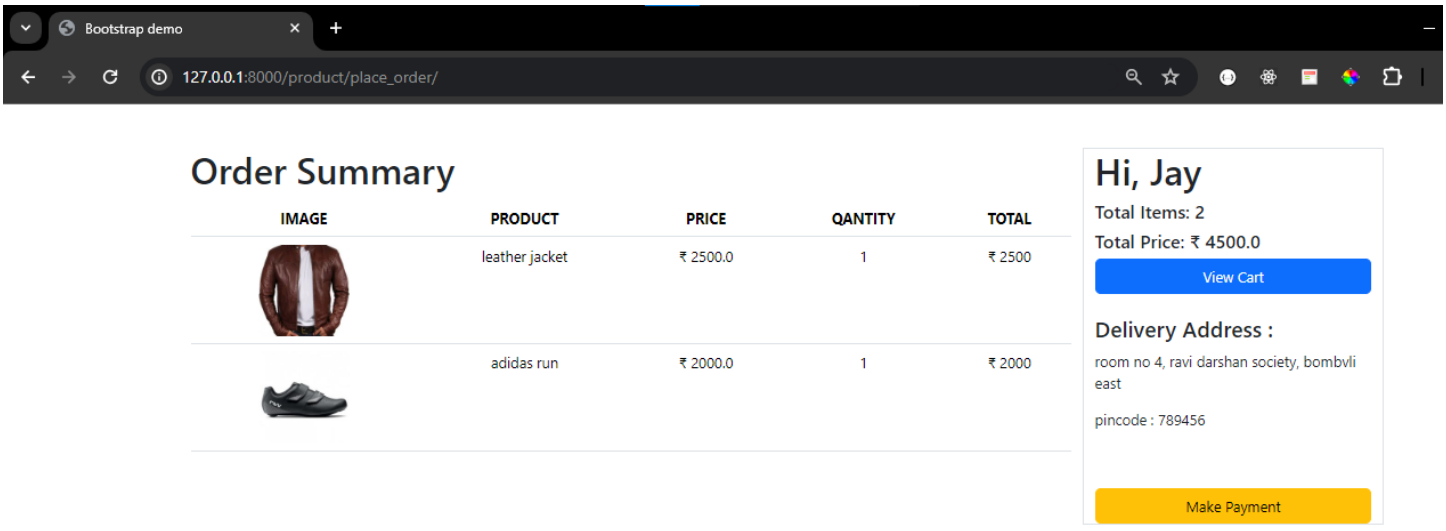
templates > product > pay.html > html > body > script > options > "prefill" > "contact"

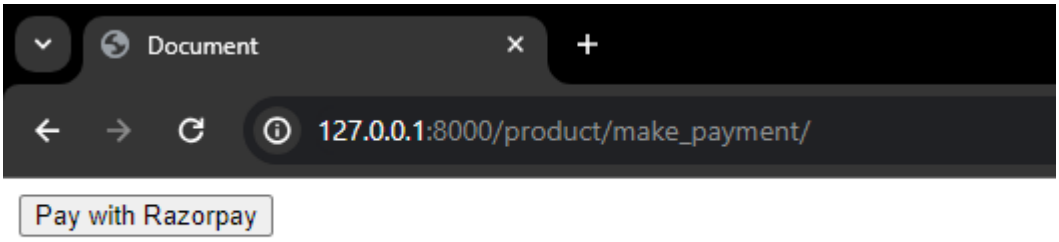
```
8   <body>
9       <button id="rzp-button1">Pay with Razorpay</button>
10  <script src="https://checkout.razorpay.com/v1/checkout.js"></script>
11  <script>
12  var options = {
13      "key": "rzp_test_ls3ufyYec3WWtv", // Enter the Key ID generated from the Dashboard
14      "amount": "{{amount}}", // Amount is in currency subunits. Default currency is INR. Hence, 50000 refers to 50000 paise
15      "currency": "INR",
16      "name": "Kasare Industries",
17      "description": "Test Transaction",
18      "image": "https://cdn.iconscout.com/icon/free/png-256/free-industry-1430036-1207834.png?f=webp",
19      "order_id": "{{id}}", //This is a sample Order ID. Pass the `id` obtained in the response of Step 1
20      "handler": function (response){
21          alert(response.razorpay_payment_id);
22          alert(response.razorpay_order_id);
23          alert(response.razorpay_signature)
24      },
25      "prefill": {
26          "name": "Tejas Kasare",
27          "email": "tejaskasare14@gmail.com",
28          "contact": "8459859415"
29      },
30      "notes": {
31          "address": "Razorpay Corporate Office"
```

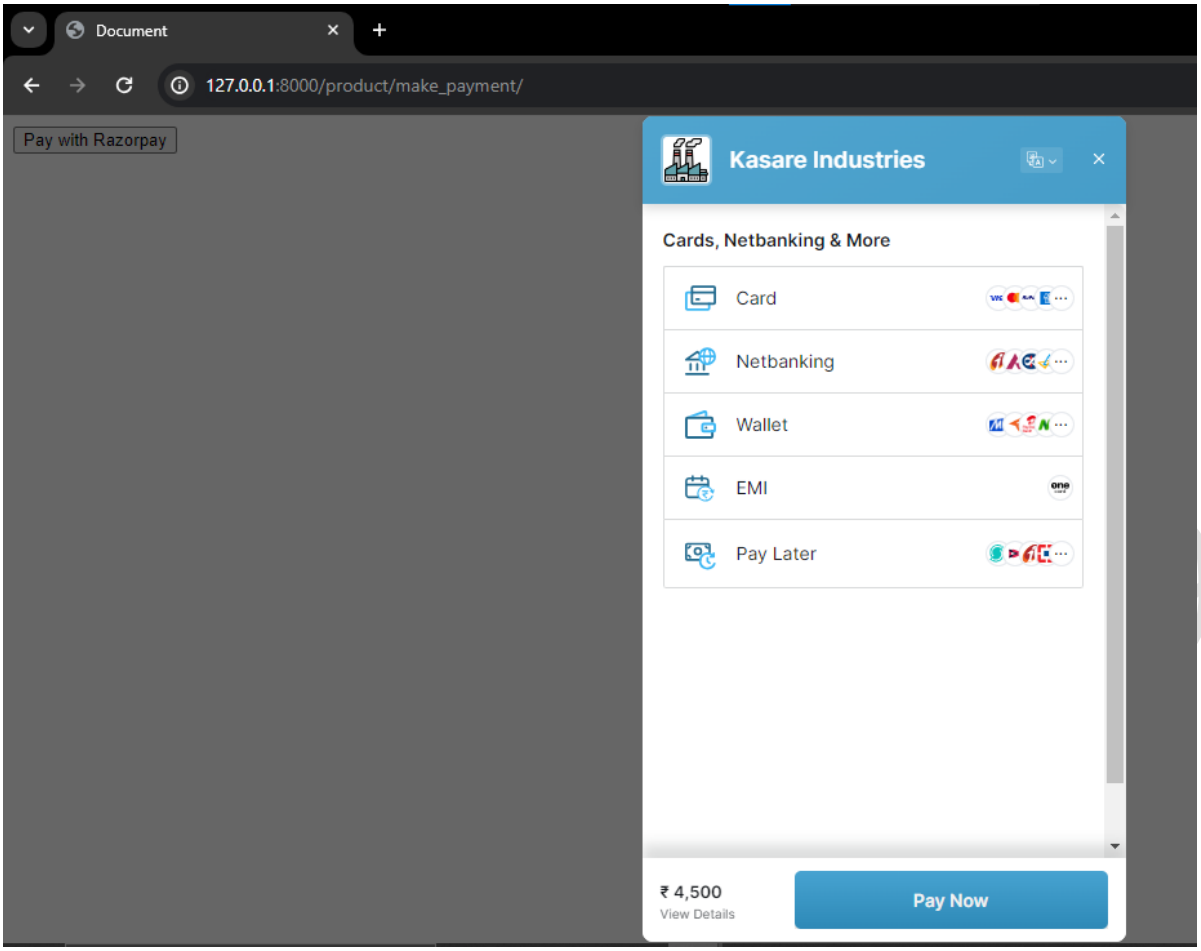
v. Final output



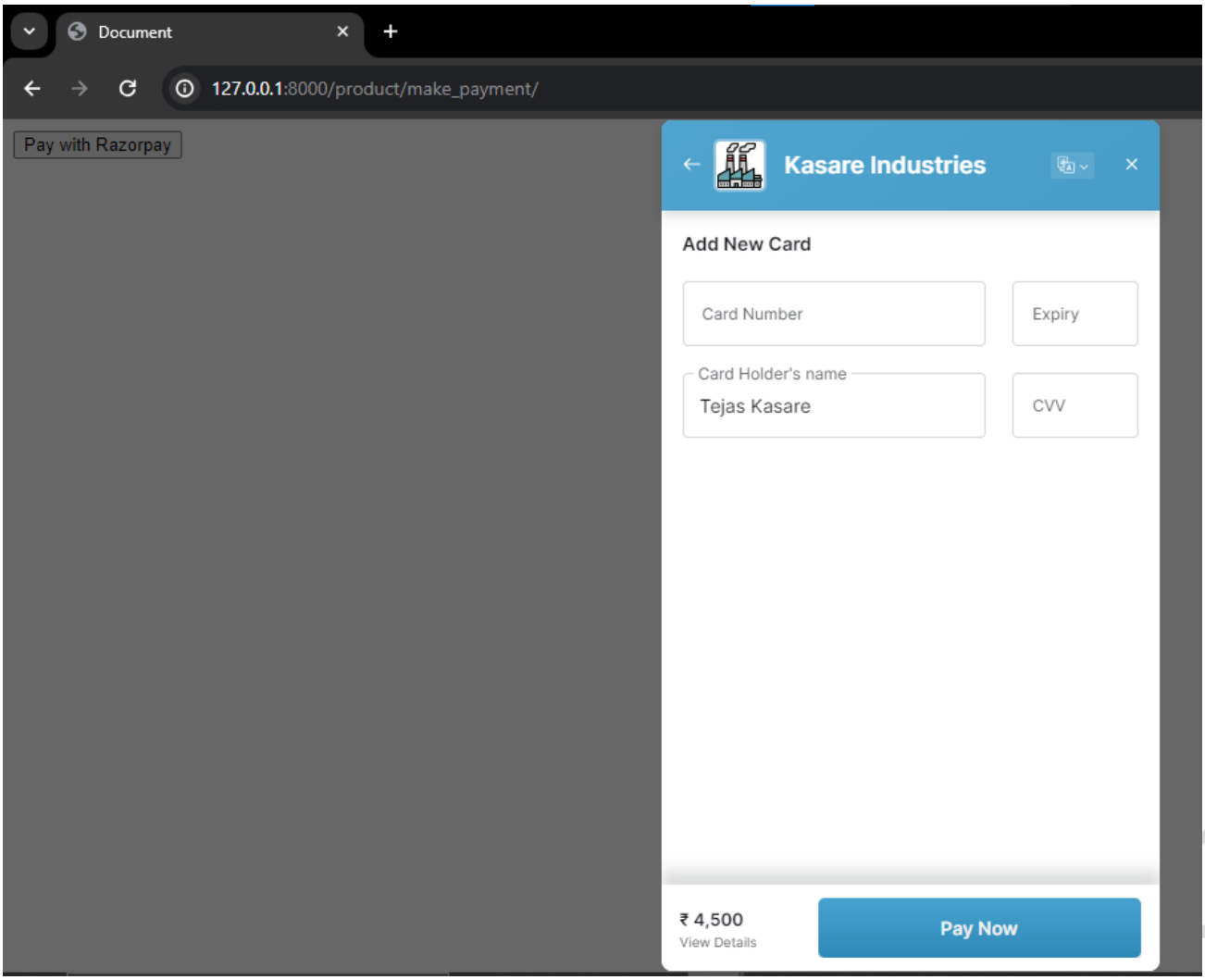




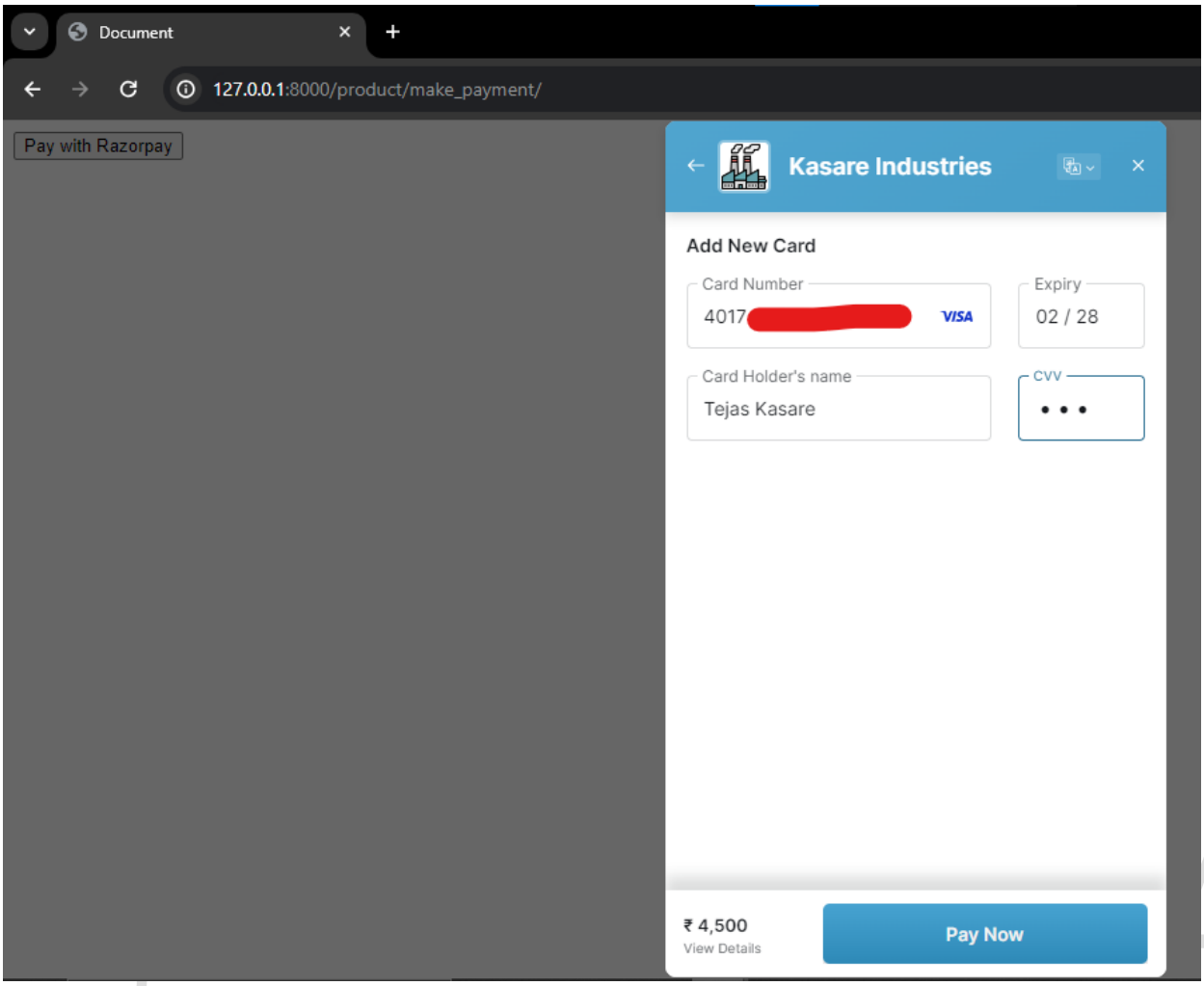




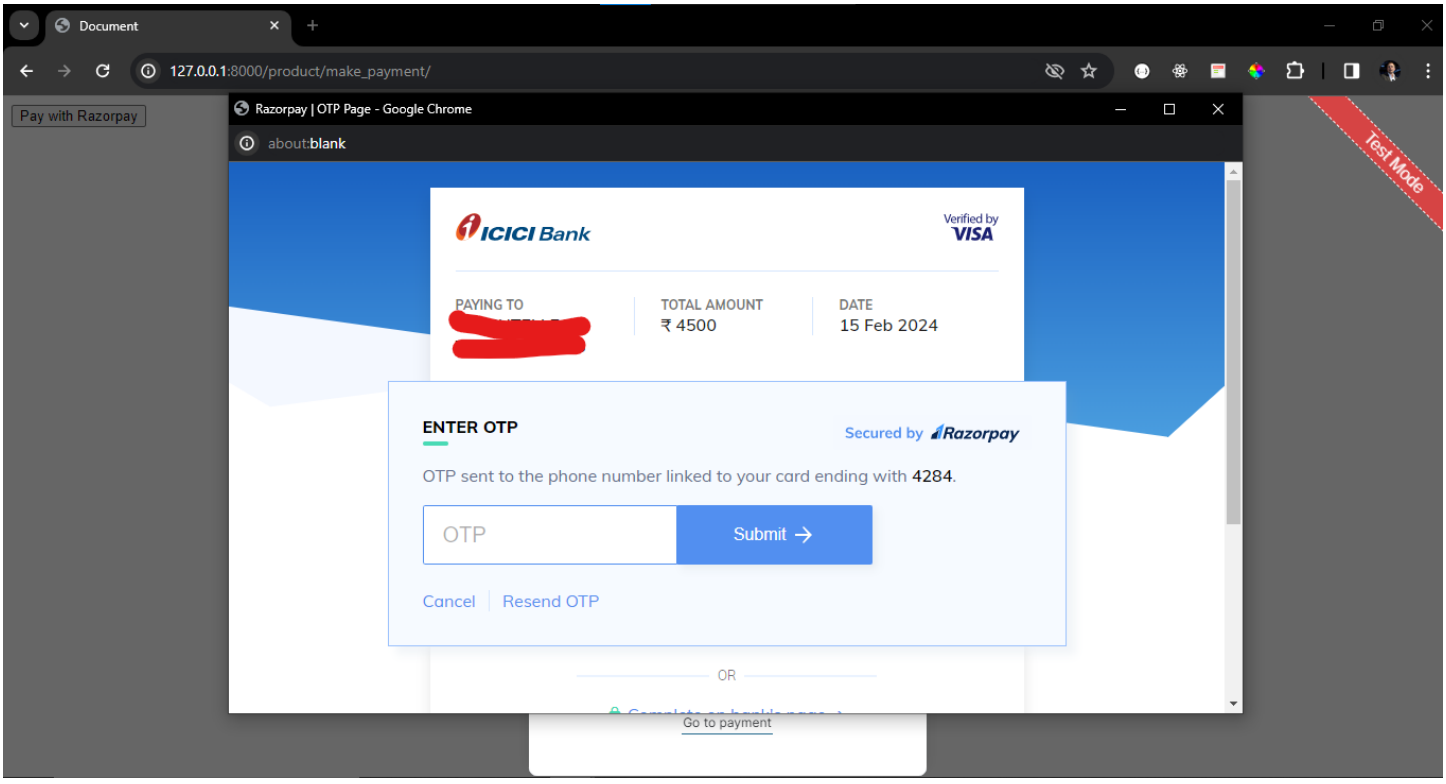
Click on card and add correct card number but you can add dummy or wrong cvv and exp date

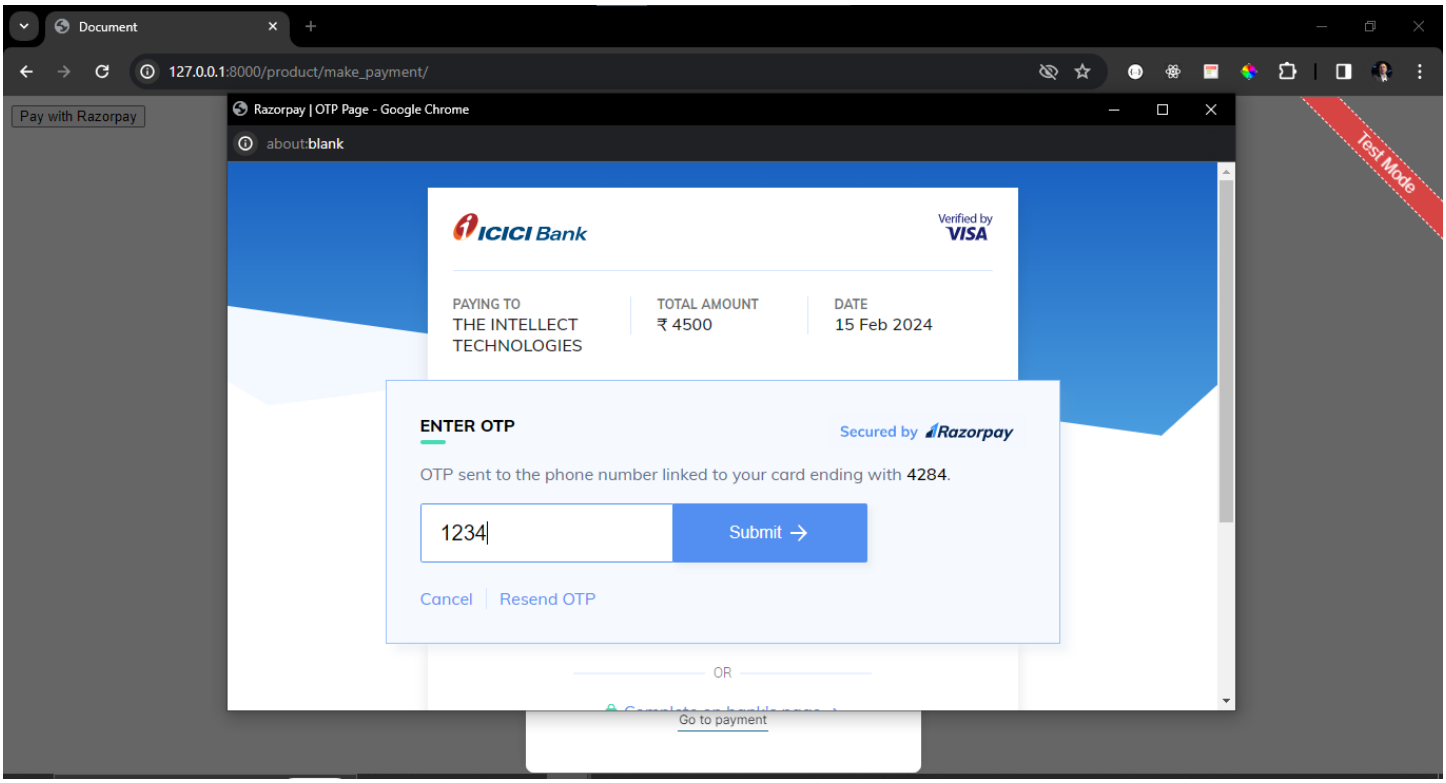

Document
127.0.0.1:8000/product/make_payment/
Pay with Razorpay
Kasare Industries
Add New Card
Card Number
Expiry
Card Holder's name
Tejas Kasare
CVV
₹ 4,500
View Details
Pay Now

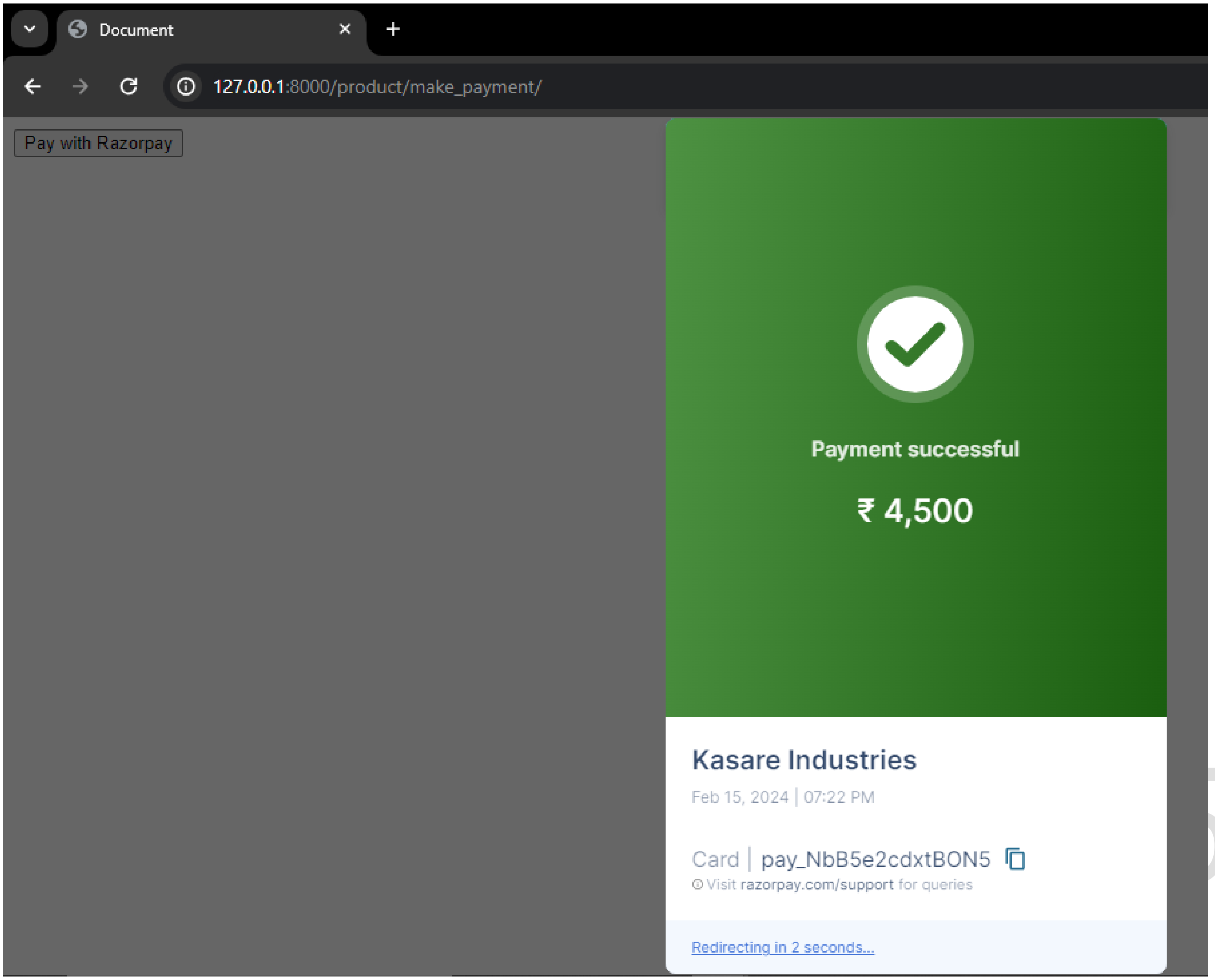
15



Click on pay now and you make ask for OTP. Here you have to add any 4 digits (4 digits compulsory)







DONEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE

vi.