Each time log in Step by Step to code:

1. Start env

cd C:\Users\angel\OneDrive\Desktop\Python Django\env\Scripts

activate

1. Display the packages install in the env

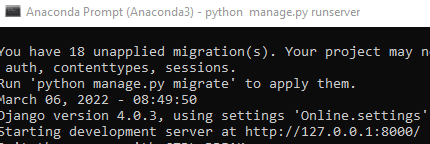
pip freeze

1. Run server to start coding

cd C:\Users\angel\OneDrive\Desktop\Python Django

python manage.py runserver

http://127.0.0.1:8000/



python manage.py makemigrations

python manage.py migrate

Website development Step by Step:

1. Find out the path name of the web development folder

cd C:\Users\angel\OneDrive\Desktop\Python Django

1. Installing virtualenv

py –m pip install –-user virtualenv

1. Creating a virtual environment folder

py –m venv env

1. Pointing to the activate the env folder

cd C:\Users\angel\OneDrive\Desktop\Python Django\env\Scripts

activate

1. Install Django package

pip install django

1. Display the packages install in the env

pip freeze

1. create project folder

cd C:\Users\angel\OneDrive\Desktop\Python Django

django-admin startproject Online .

1. Python start web site server

python manage.py runserver

http://127.0.0.1:8000/

1. Create a views.py in the online folder together with urls.py
2. Web Testing thru HTTP Response

urls.py

from django.contrib import admin

from django.urls import path

from . import views

urlpatterns = [

path('admin/', admin.site.urls),

path('',views.home, name='home'),

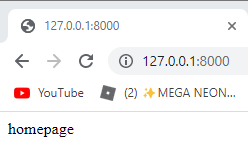
]

Views.py

from django.http import HttpResponse

def home(request):

return HttpResponse('homepage')



1. Create templates folder in the project folder
2. Create home.html in the templates folder

<h2>This is templates</h2>

1. Add templates folder name in the setting.py file in the project folder

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': ['templates'],

1. Adjust the view filer from HttpResponse to Render

#from django.http import HttpResponse

from django.shortcuts import render

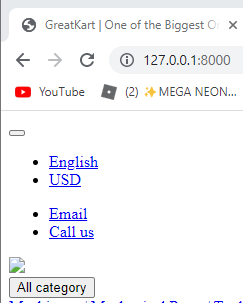
def home(request):

#return HttpResponse('homepage')

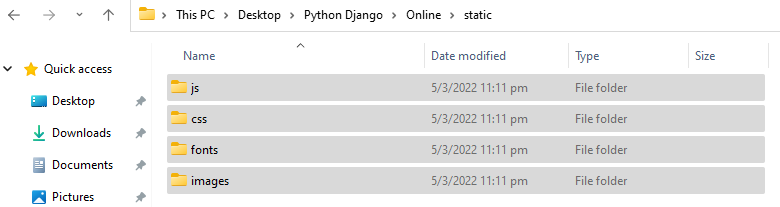
return render(request, 'home.html')



1. Copy and paste the index.html codes into home.html in the templates folder



1. Create static folder in the Online folder and copy and paste the 4 folders into the static folder



1. Adjust settings/py to prepare the linkage with pictures with the website from images folder

STATIC\_URL = '/static/'

STATIC\_ROOT = BASE\_DIR /'static'

STATICFILES\_DIRS =[

'online/static',

]

1. Change website logo with codes adjustment on home.html

Insert 1 sentence in the 1st line in home.html

{% load static %}

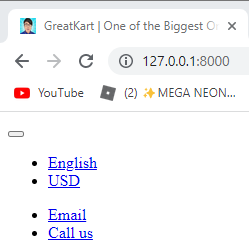
<!DOCTYPE HTML>

<html lang="en">

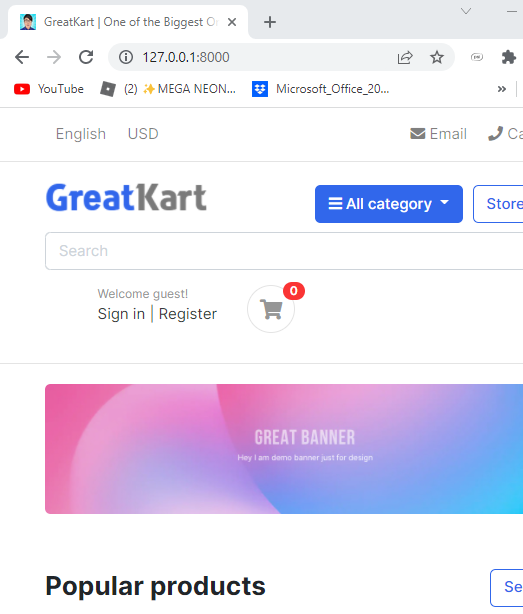
Replace old code with new codes

<link href="images/favicon.ico" rel="shortcut icon" type="image/x-icon">

<link href= {% static 'images/favicon.ico'%} rel="shortcut icon" type="image/x-icon">



1. Repeat the replacement process on js, css png, jpg



1. Create base.html in the templates folder
2. Copy the 1st line to </header> <!-- section-header.// --> from home.html to base.html
3. Add lines in base.html

1st line

{% load static %}

<!DOCTYPE HTML>

Last few lines

</header> <!-- section-header.// -->

{% block contents %}

<!-- content -->

{% endblock %}

1. Add lines in the homt.html

1st 3 lines

{% extends 'base.html' %}

{% load static %}}

{% block contents %}

<!-- ========================= SECTION MAIN ========================= -->

<section class="section-intro padding-y-sm">

Last 1 line

</html>

{% endblock %}

1. Create includes folder in the templates folder
2. Create footer.html & navbar.html in the includes folder
3. Copy the footer codes from home.html to footer.html

<!-- ========================= FOOTER ========================= -->

<footer class="section-footer border-top">

<div class="container">

<section class="footer-bottom border-top row">

<div class="col-md-2">

<p class="text-muted"> &copy 2019 Company name </p>

</div>

<div class="col-md-8 text-md-center">

<span class="px-2">info@pixsellz.io</span>

<span class="px-2">+879-332-9375</span>

<span class="px-2">Street name 123, Avanue abc</span>

</div>

<div class="col-md-2 text-md-right text-muted">

<i class="fab fa-lg fa-cc-visa"></i>

<i class="fab fa-lg fa-cc-paypal"></i>

<i class="fab fa-lg fa-cc-mastercard"></i>

</div>

</section>

</div><!-- //container -->

</footer>

<!-- ========================= FOOTER END // ========================= -->

1. Copy header sets of codes from base.html to navbar.html
2. Add lines in the base.html at bottom of the codes

<body>

<!-- navbar -->

{% include 'includes/navbar.html' %}

{% block contents %}

<!-- content -->

{% endblock %}

<!-- footer -->

{% include 'includes/footer.html' %}

1. Add 1 line at navbar.html

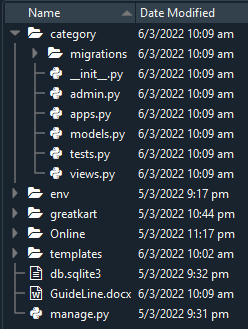
{% load static %}

1. Add 1 line in the foorer.html

{% load static %}

1. Create an category folder in the online folder

python manage.py startapp category



1. Register the category folder in the settings.py under online folder

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'category',

]

1. Create Models and Admin code for migration preparation

models.py

from django.db import models

# Create your models here.

class Category(models.Model):

category\_name = models.CharField(max\_length=50, unique = True)

slug = models.CharField(max\_length=100, unique=True)

descprtion = models.TextField(max\_length=255, blank=True)

cat\_image = models.ImageField(upload\_to='photos/categories', blank=True)

def \_\_str\_\_(self):

return self.category\_name

admin.py

from django.contrib import admin

from .models import Category

# Register your models here.

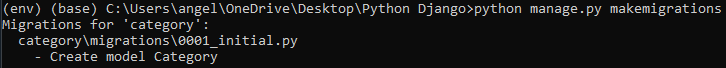
admin.site.register(Category)

1. Install pillow package to use ImageField (Cannot use ImageField because Pillow is not installed.)

Pip install pillow

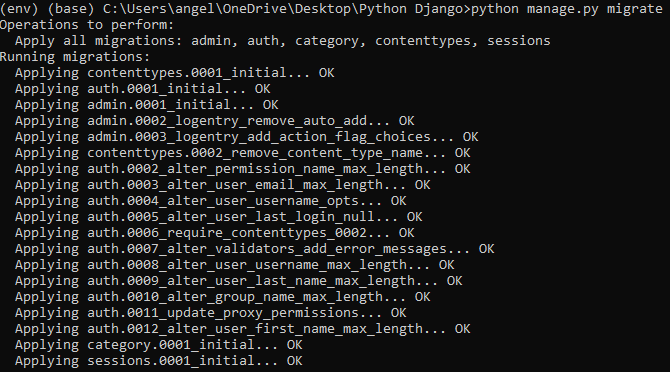
1. Type command in the Anaconda Prompt (Anaconda3) to create migrations folder

python manage.py makemigrations



1. Python run the migrate files 0001\_initial.py

python manage.py migrate



1. Create a super user

python manage.py createsuperuser

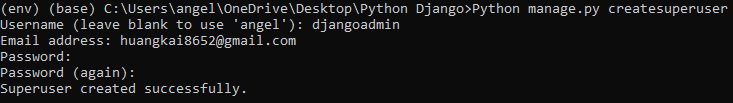
1. Create ID and Password for super user

Username: djangoadmin

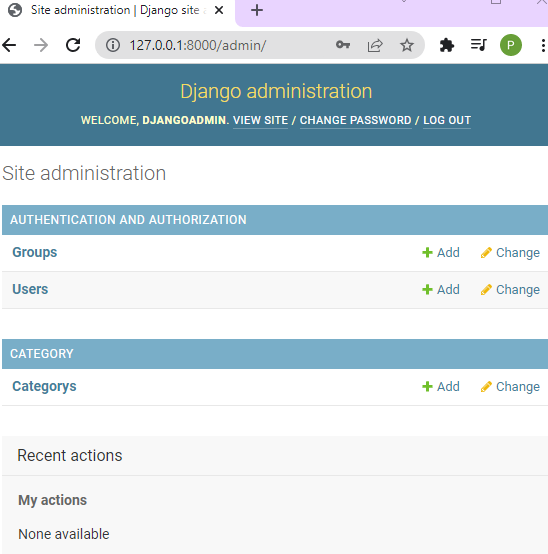
Email: huangkai8652@gamil.com

Password: !q2w3e4R

Password: !q2w3e4R



Django Admin Page (Logged In)



1. Edit the Categorys as above picture

Add lines in the models.py

cat\_image = models.ImageField(upload\_to='photos/categories', blank=True)

class Meta:

verbose\_name = 'Category'

verbose\_name\_plural = 'Categories'

def \_\_str\_\_(self):

1. Run migration command again

python manage.py makemigrations



1. Create account folder for email log in

python manage.py startapp accounts

1. Create linkage with settings.py in the online folder

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'category',

'accounts',

]

1. Add lines of code in the models.py in the accounts folder

from django.db import models

from django.contrib.auth.models import AbstractBaseUser, BaseUserManager

# Create your models here.

class MyAccountManager(BaseUserManager):

def create\_user(self, first\_name, last\_name, username, email, password=None):

if not email:

raise ValueError('User must have an email address')

if not username:

raise ValueError('User must have an username')

user = self.model(

email = self.normalize\_email(email),

username = username,

first\_name = first\_name,

last\_name = last\_name,

)

user.set\_password(password)

user.save(using=self.\_db)

return user

def create\_superuser(self, first\_name, last\_name, email, username, password):

user = self.create\_user(

email = self.normalize\_email(email),

username = username,

password = password,

first\_name = first\_name,

last\_name = last\_name,

)

user.is\_admin = True

user.is\_active = True

user.is\_staff = True

user.is\_superadmin = True

user.save(using=self.\_db)

return user

class Account(AbstractBaseUser):

first\_name = models.CharField(max\_length=50)

last\_name = models.CharField(max\_length=50)

username = models.CharField(max\_length=50, unique=True)

email = models.EmailField(max\_length=100, unique=True)

phone\_number = models.CharField(max\_length=50)

# required

date\_joined = models.DateTimeField(auto\_now\_add=True)

last\_login = models.DateTimeField(auto\_now\_add=True)

is\_admin = models.BooleanField(default=False)

is\_staff = models.BooleanField(default=False)

is\_active = models.BooleanField(default=False)

is\_superadmin = models.BooleanField(default=False)

USERNAME\_FIELD = 'email'

REQUIRED\_FIELDS = ['username', 'first\_name', 'last\_name']

objects = MyAccountManager()

def full\_name(self):

return f'{self.first\_name} {self.last\_name}'

def \_\_str\_\_(self):

return self.email

def has\_perm(self, perm, obj=None):

return self.is\_admin

def has\_module\_perms(self, add\_label):

return True

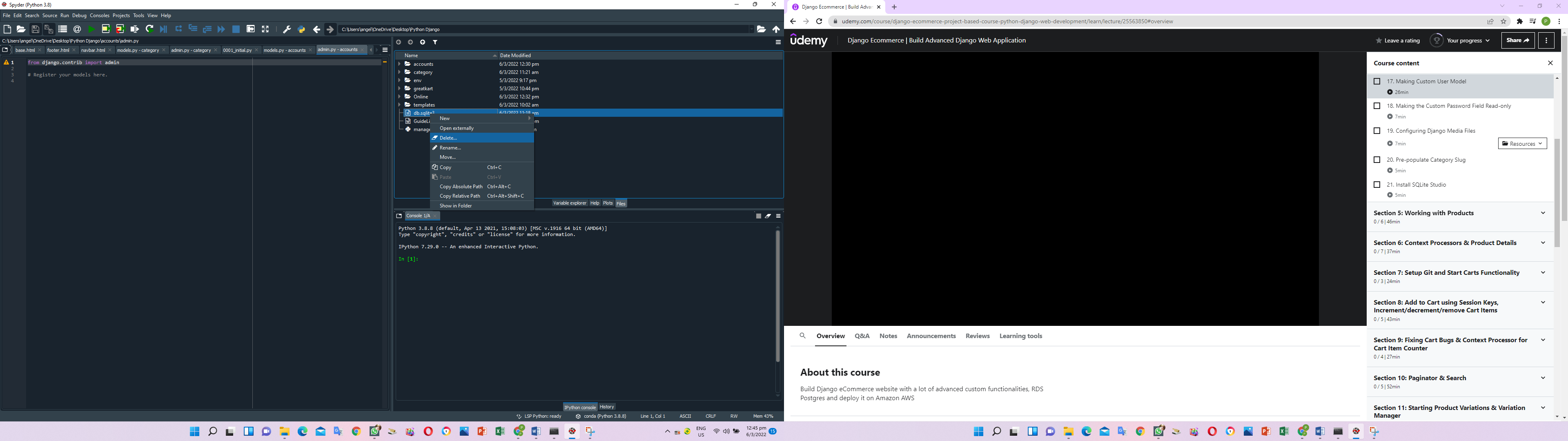
1. Add links in the settings.py file

WSGI\_APPLICATION = 'Online.wsgi.application'

AUTH\_USER\_MODEL = 'accounts.Account'

# Database

1. Delete the database db.sqlite3 in the online folder



1. Add lines in the admin.py in the accoutns folder

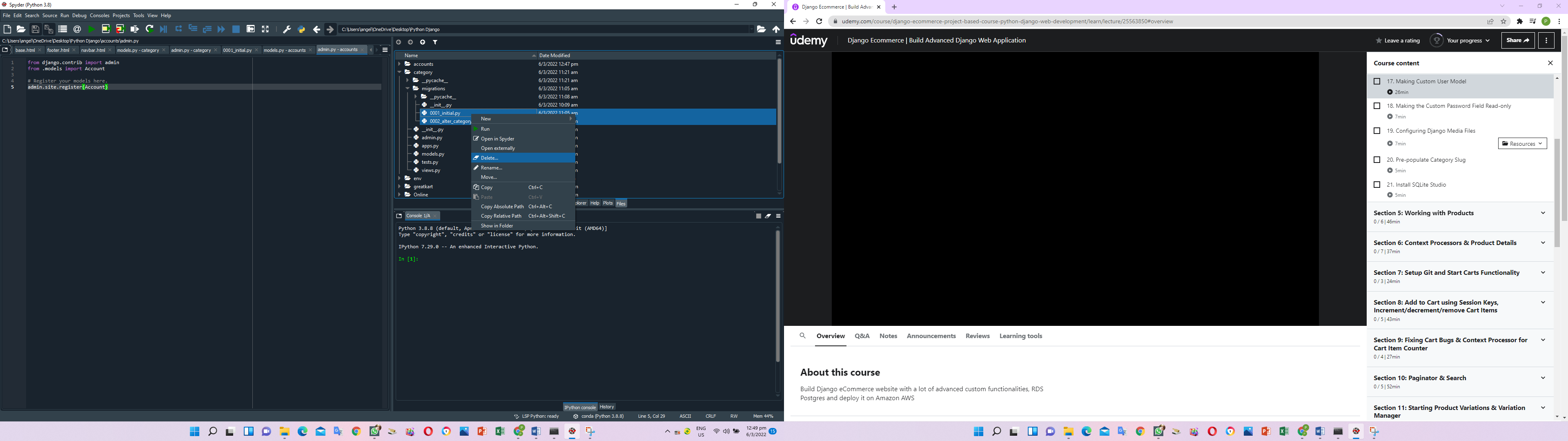
from django.contrib import admin

from .models import Account

# Register your models here.

admin.site.register(Account)

1. Delete the 0001\_initial.py & 0002\_alter\_category\_options.py in the migrations folder.

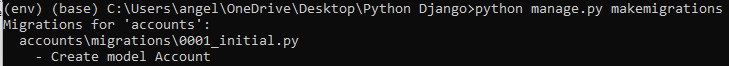


1. Create a db.sqlite3 in the online folder

python manage.py runserver

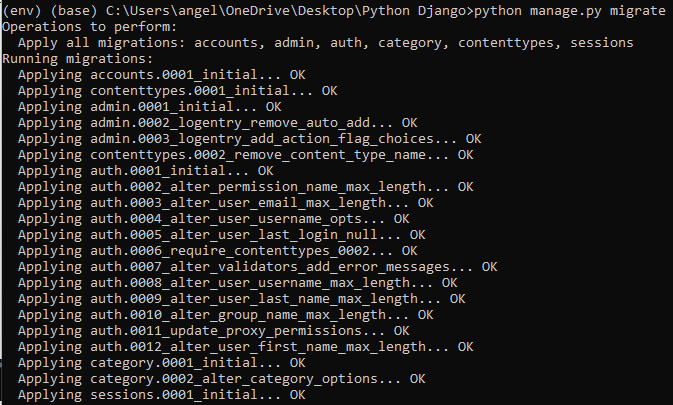
1. Migration folder again for the 0001\_inital.py

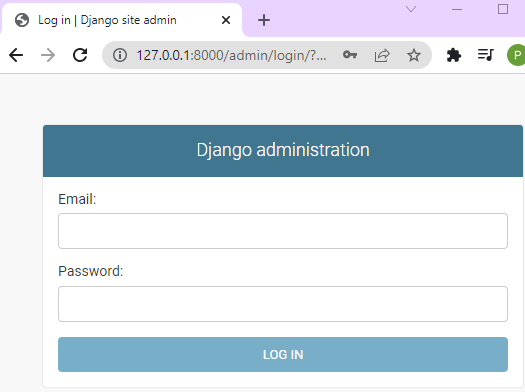
python manage.py makemigrations



1. Migrate the folder

python manage.py migrate





1. Create super user

Email: huangkai8652@gmail.com

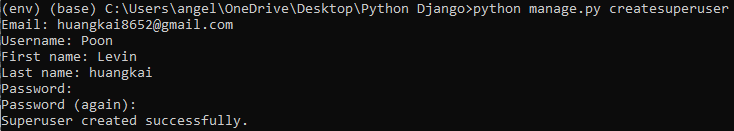
Username: Poon

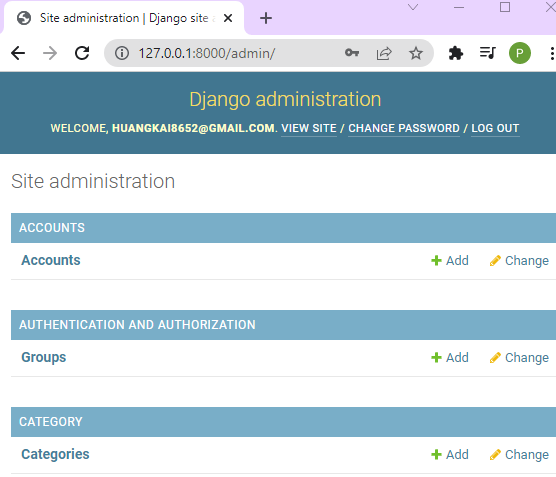
First name: Levin

Last name: huangkai

Password: !q2w3e4R

Password (again): !q2w3e4R





User able to log in thru email instead of username

1. Change password view to read only, create linkage between first name and last name to profile and seem last login timing

Add lines in admin.py under accounts

from django.contrib import admin

from django.contrib.auth.admin import UserAdmin

from .models import Account

# Register your models here.

class AccountAdmin(UserAdmin):

list\_display = ('email', 'first\_name', 'last\_name', 'username', 'last\_login', 'date\_joined', 'is\_active')

list\_display\_links = ('email', 'first\_name', 'last\_name')

readonly\_fields = ('last\_login', 'date\_joined')

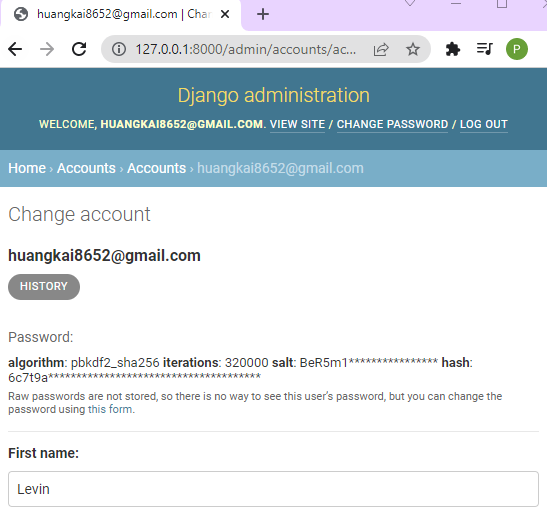
ordering = ('-date\_joined',)

filter\_horizontal = ()

list\_filter = ()

fieldsets = ()

admin.site.register(Account, AccountAdmin)



1. Add lines in the settings.py in the online folder (Configuring Media Files)

STATICFILES\_DIRS =[

'online/static',

]

# media files configuration

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR /'media'

1. Add lines in the urls.py in the online folder

from . import views

from django.conf.urls.static import static

from django.conf import settings

urlpatterns = [

path('admin/', admin.site.urls),

path('',views.home, name='home'),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

1. Save information on the django admin website under category

Category name: Shirts

Slug: shirts

Descriptions: This is the demo shirt category

Cat Image: shirts.jpg

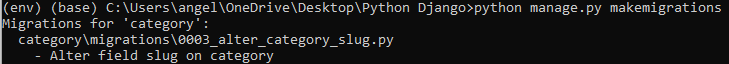
1. Slug name will auto copy from Category name

slug = models.CharField(max\_length=100, unique=True) change to =>

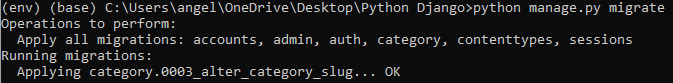
slug = models.SlugField(max\_length=100, unique=True)

1. Once the model in the category completed, user have to make migrate and migrate in CMD

python manage.py makemigrations



python manage.py migrate



1. Add lines in the admin.py under the category folder

# Register your models here.

class CategoryAdmin(admin.ModelAdmin):

prepopulated\_fields = {'slug':('category\_name',)}

list\_display = ('category\_name','slug')

admin.site.register(Category,CategoryAdmin)

1. Add new item of t shirt and slug will auto populate

Category name: T shirt

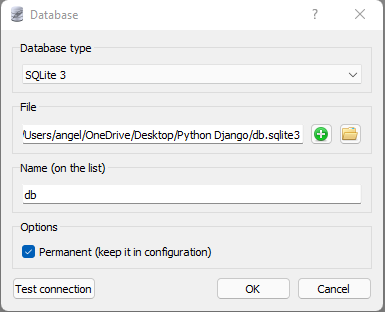
Slug: t-shirt

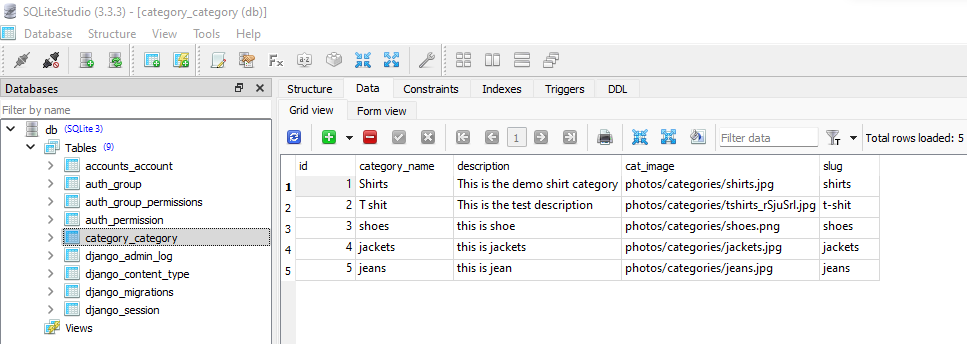
Descriptions: This is the test description

Cat Image: tshirts.jpg

Add the rest 4 category into the Django admin

1. The database is in the db.sqlite3 and download sqlitestudio from Goggle to view in human readable format





1. Crate store folder in the online folder

Python manage.py startapp store

1. Add store in the setting.py under online folder

'category',

'accounts',

'store',

]

1. Add line in models.py under store folder

from django.db import models

from category.models import Category

# Create your models here.

class Product(models.Model):

product\_name = models.CharField(max\_length=200, unique=True)

slug = models.SlugField(max\_length=200, unique=True)

description = models.TextField(max\_length=500, blank=True)

price = models.IntegerField()

images = models.ImageField(upload\_to='photos/products')

stock = models.IntegerField()

is\_available = models.BooleanField(default=True)

category = models.ForeignKey(Category, on\_delete=models.CASCADE)

created\_date = models.DateTimeField(auto\_now\_add=True)

modified\_date = models.DateTimeField(auto\_now=True)

def \_\_str\_\_(self):

return self.product\_name

1. Add lines in the admin.py under the store folder

from django.contrib import admin

from .models import Product

# Register your models here.

class ProductAdmin(admin.ModelAdmin):

list\_\_display = ('product\_name','price','stock','category','modified\_date','is\_available')

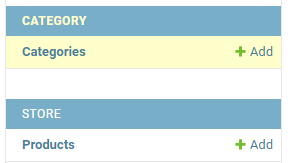
prepopluated\_fields = {'slug': ('product\_name',)}

admin.site.register(Product, ProductAdmin)

python manage.py makemigrations

python manage.py migrate

User able to see product appear on the admin page



1. Add all products under each category

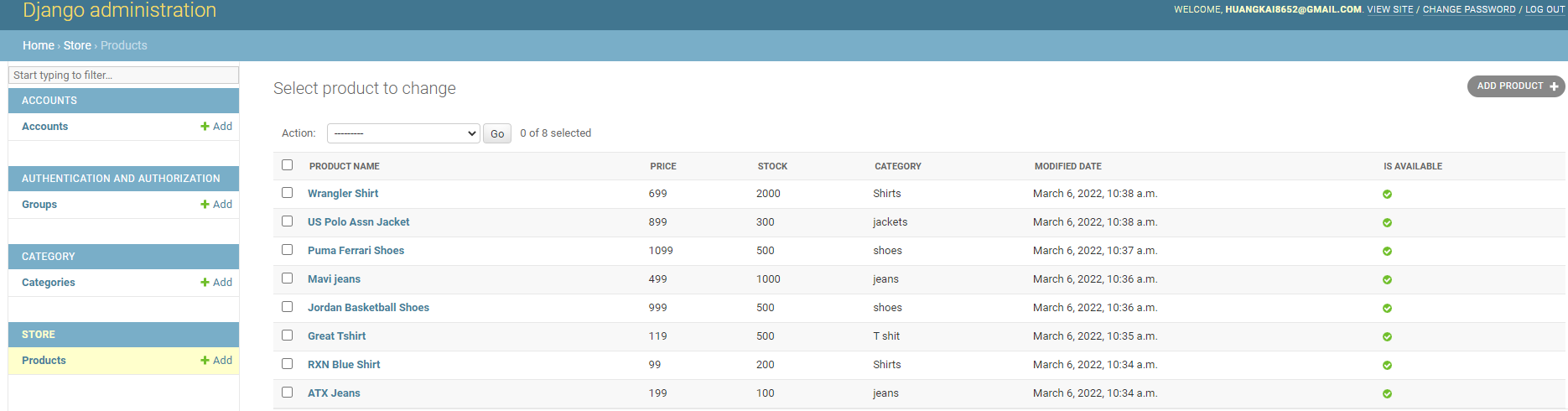
Product name:

Slug:

Description:

Price:

Images:



1. Add lines in the view.py in the online folder to display the products from database admin page

#from django.http import HttpResponse

from django.shortcuts import render

from store.models import Product

def home(request):

#return HttpResponse('homepage')

products = Product.objects.all().filter(is\_available = True)

context = {

'products': products,

}

return render(request, 'home.html', context)

1. Delete the static image from 2 to 9 in the home.html from templates folder

{% static 'images/items/2.jpg' %} to {% static 'images/items/9.jpg' %}

<!-- ========================= SECTION ========================= -->

<section class="section-name padding-y-sm">

<div class="container">

<header class="section-heading">

<a href="./store.html" class="btn btn-outline-primary float-right">See all</a>

<h3 class="section-title">Popular products</h3>

</header><!-- sect-heading -->

<div class="row">

<div class="col-md-3">

<div class="card card-product-grid">

<a href="./product-detail.html" class="img-wrap"> <img src={% static 'images/items/1.jpg' %}> </a>

<figcaption class="info-wrap">

<a href="./product-detail.html" class="title">Just another product name</a>

<div class="price mt-1">$179.00</div> <!-- price-wrap.// -->

</figcaption>

</div>

</div> <!-- col.// -->

</div> <!-- row.// -->

</div><!-- container // -->

</section>

<!-- ========================= SECTION END// ========================= -->

1. Add for loop to display the 8 products (we have insert manually at step 65)

<div class="row">

{% for product in products %}

<div class="col-md-3">

<div class="card card-product-grid">

<a href="./product-detail.html" class="img-wrap"> <img src={% static 'images/items/1.jpg' %}> </a>

<figcaption class="info-wrap">

<a href="./product-detail.html" class="title">Just another product name</a>

<div class="price mt-1">$179.00</div> <!-- price-wrap.// -->

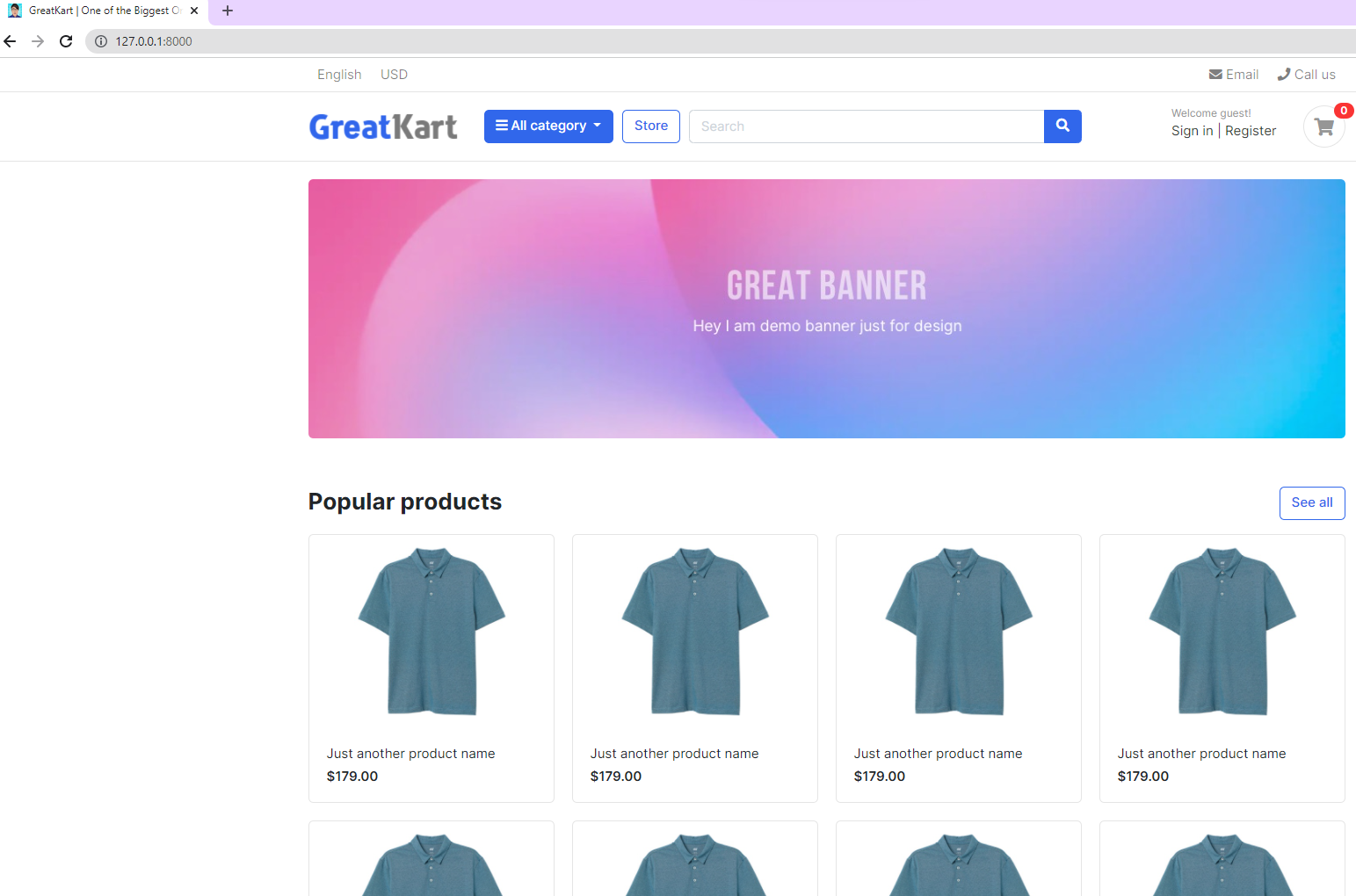
</figcaption>

</div>

</div> <!-- col.// -->

{% endfor %}

</div> <!-- row.// -->



1. Adjust the link to display the correct images, price and product name

<div class="row">

{% for product in products %}

<div class="col-md-3">

<div class="card card-product-grid">

<a href="./product-detail.html" class="img-wrap"> <img src="{{ product.images.url }}"> </a>

<figcaption class="info-wrap">

<a href="./product-detail.html" class="title">{{product.product\_name}}</a>

<div class="price mt-1">${{product.price}}</div> <!-- price-wrap.// -->

</figcaption>

</div>

</div> <!-- col.// -->

{% endfor %}

</div> <!-- row.// -->

</div><!-- container // -->

</section>

<!-- ========================= SECTION END// ========================= -->

1. Copy urls.py from online folder to store folder (making store button working)

from django.urls import path

from . import views

urlpatterns = [

path('', views.store, name='store'),

]

1. Add lines in urls.py in the online folder

from django.contrib import admin

from django.urls import path, include

from . import views

from django.conf.urls.static import static

from django.conf import settings

urlpatterns = [

path('admin/', admin.site.urls),

path('',views.home, name='home'),

path('store/', include('store.urls')),

] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

1. Add lines in views.py under store folder

from django.shortcuts import render

# Create your views here.

def store(request):

return render(request, 'store/store.html')

1. Chang the link at home.html in the template folder

<!-- ========================= SECTION ========================= -->

<section class="section-name padding-y-sm">

<div class="container">

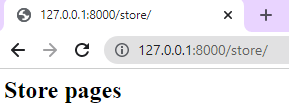
<header class="section-heading">

<a href="{% url 'store' %}" class="btn btn-outline-primary float-right">See all</a>

<h3 class="section-title">Popular products</h3>

</header><!-- sect-heading -->

1. Create store.html in the store folder in the template folder and refresh the website

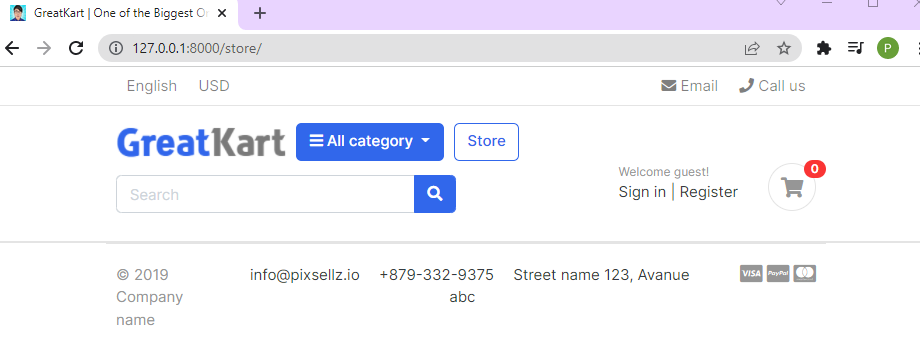


1. Replace codes in the store.html under the store folder under templates folder

{% extends 'base.html' %}

{% block content %}

{% endblock %}



1. Copy the codes store.html under sample project file and paste in store.html under store folder under templates folder

{% block content %}

<!-- ========================= SECTION PAGETOP ========================= -->

<section class="section-pagetop bg">

<div class="container">

<h2 class="title-page">Our Store</h2>

</div> <!-- container // -->

</section>

<!-- ========================= SECTION INTRO END// ========================= -->

1. Adjust line in the base.html under templates folder

<!-- navbar -->

{% include 'includes/navbar.html' %}

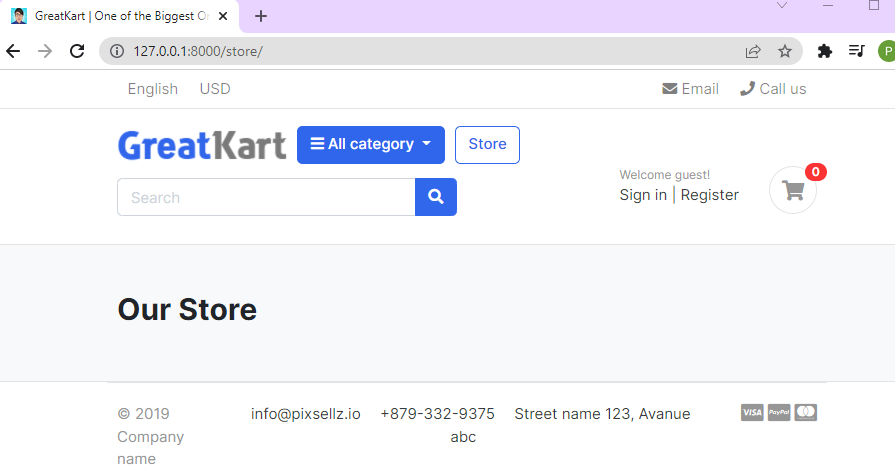
{% block content %}

<!-- content -->

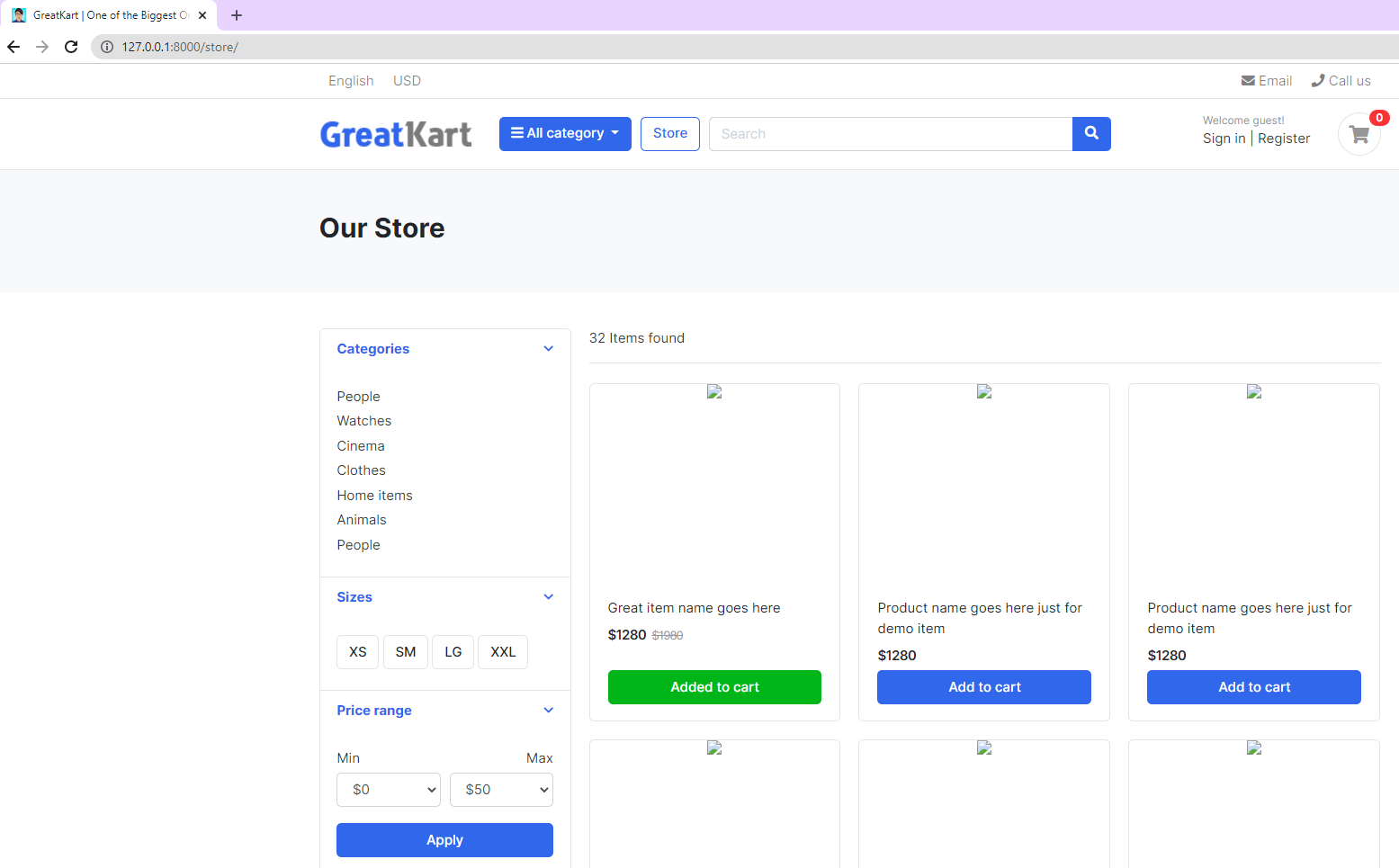
{% endblock %}

<!-- footer -->

{% include 'includes/footer.html' %}



1. Copy SECTION CONTENT to SECTION CONTENT END from store.html under sample project folder



1. Load pictures into the store.html under store at 2nd line

{% extends 'base.html' %}

{% load static %}

{% block content %}{% load static %}

1. Replace image code

<img src="images/items/1.jpg">

<img src="{% static 'images/items/1.jpg' %}">

1. Add lines in the views.py in the store folder

from django.shortcuts import render

from .models import Product

# Create your views here.

def store(request):

products = Product.objects.all().filter(is\_available = True)

context = {

'products': products,

}

return render(request, 'store/store.html', context)

1. Delete the image 2 to last image in the store.html under store folder under templates folder

<img src="{% static 'images/items/1.jpg' %}">

</div> <!-- img-wrap.// -->

<figcaption class="info-wrap">

<div class="fix-height">

<a href="./product-detail.html" class="title">Great item name goes here</a>

<div class="price-wrap mt-2">

<span class="price">$1280</span>

<del class="price-old">$1980</del>

</div> <!-- price-wrap.// -->

</div>

<a href="#" class="btn btn-block btn-success">Added to cart </a>

</figcaption>

</figure>

</div> <!-- col.// -->

</div> <!-- row end.// -->

1. Add loop to display the pictures

<div class="row">

{% for product in products %}

<div class="col-md-4">

<figure class="card card-product-grid">

<div class="img-wrap">

…

</figcaption>

</figure>

</div> <!-- col.// -->

{% endfor %}

</div> <!-- row end.// -->

1. User able to see the same short appear on web screen with blue button of add to cart

<a href="#" class="btn btn-block btn-primary">Add to cart </a>

1. Add line to adjust the prices

<span class="price">$ {{ product.price }}</span>

1. Delete 1 line

<del class="price-old">$1980</del>

1. Final adjustment to diplay the photo, product name and product price

<div class="row">

{% for product in products %}

<div class="col-md-4">

<figure class="card card-product-grid">

<div class="img-wrap">

<img src="{{ product.images.url }}">

</div> <!-- img-wrap.// -->

<figcaption class="info-wrap">

<div class="fix-height">

<a href="./product-detail.html" class="title">{{ product.product\_name }}</a>

<div class="price-wrap mt-2">

<span class="price">$ {{ product.price }}</span>

</div> <!-- price-wrap.// -->

</div>

<a href="#" class="btn btn-block btn-primary">Add to cart </a>

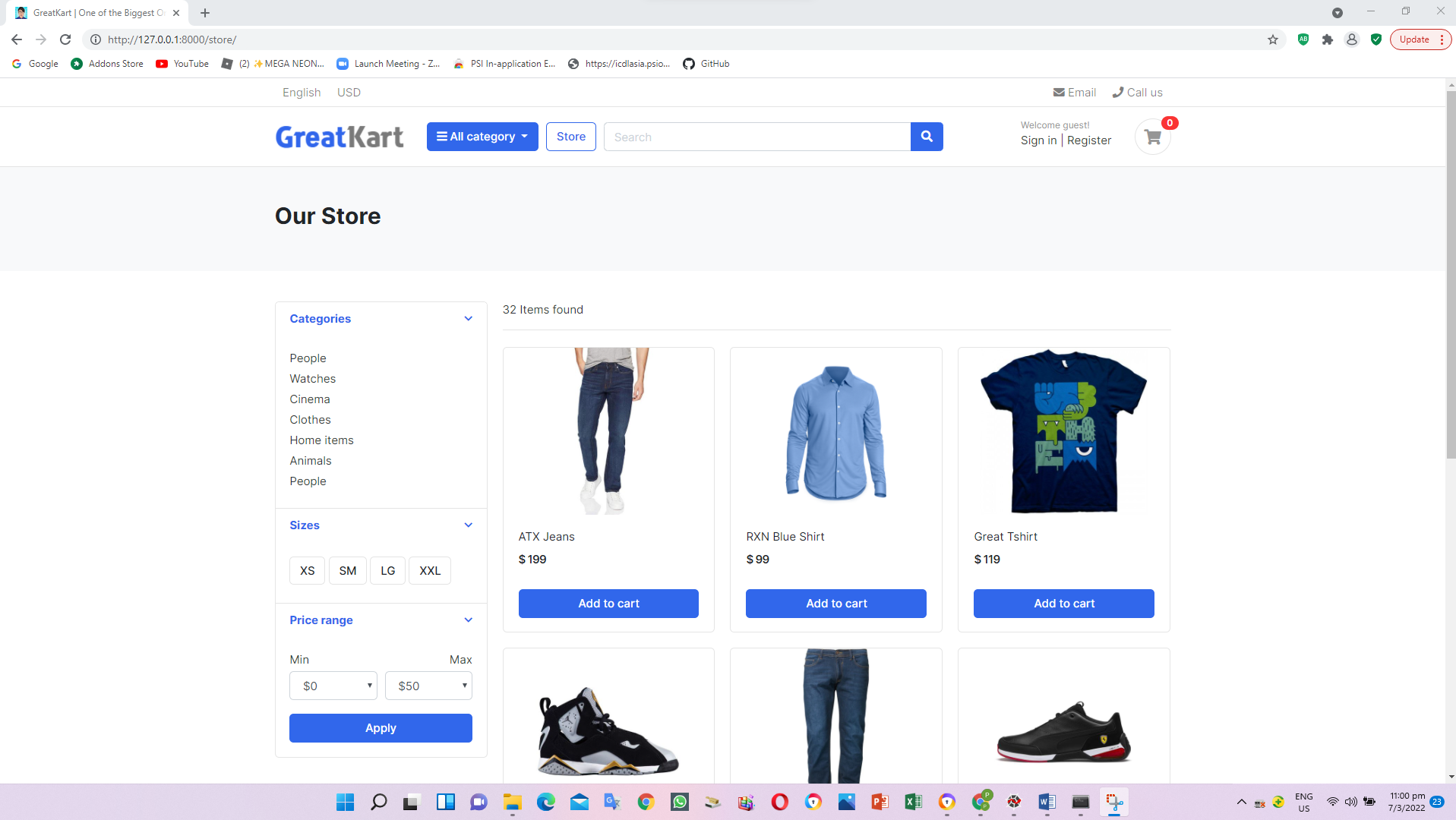
</figcaption>

</figure>

</div> <!-- col.// -->

{% endfor %}

</div> <!-- row end.// -->



1. Add 2 lines in the view.py under store folder

from django.shortcuts import render

from .models import Product

# Create your views here.

def store(request):

products = Product.objects.all().filter(is\_available = True)

product\_count = products.count()

context = {

'products': products,

'product\_count': product\_count,

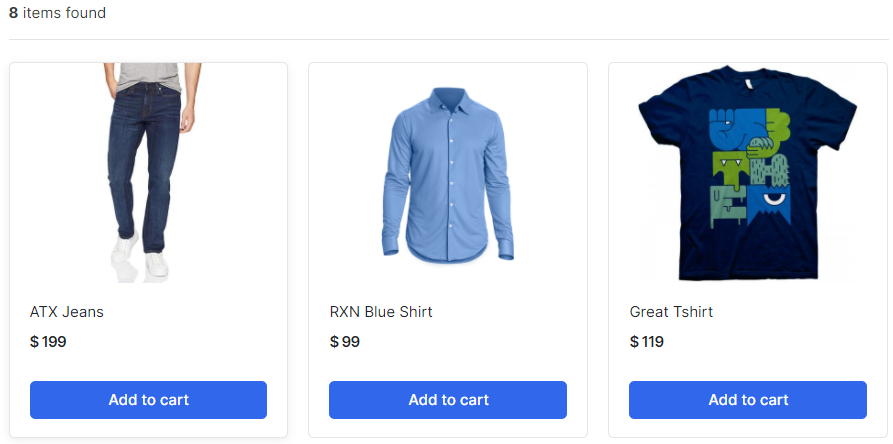
}

return render(request, 'store/store.html', context)

1. Adjust the found item from static to link to database items under store.html under store folder under template folder

<span class="mr-md-auto">32 Items found </span>

<span class="mr-md-auto"><b>{{ product\_count }}</b> items found </span>



1. Adjust the lines in the urls.py under store folder (fetch the items by categories)

from django.urls import path

from . import views

urlpatterns = [

path('', views.store, name='store'),

path('<slug:category\_slug>/', views.store, name='products\_by\_category'),

]

1. Adjust the lines in the views.py under store folder

from django.shortcuts import render, get\_object\_or\_404

from .models import Product

from category.models import Category

# Create your views here.

def store(request, category\_slug=None):

categories = None

products = None

if category\_slug != None:

categories = get\_object\_or\_404(Category, slug=category\_slug)

products = Product.objects.filter(category=categories, is\_available=True)

product\_count = products.count()

else:

products = Product.objects.all().filter(is\_available=True)

product\_count = products.count()

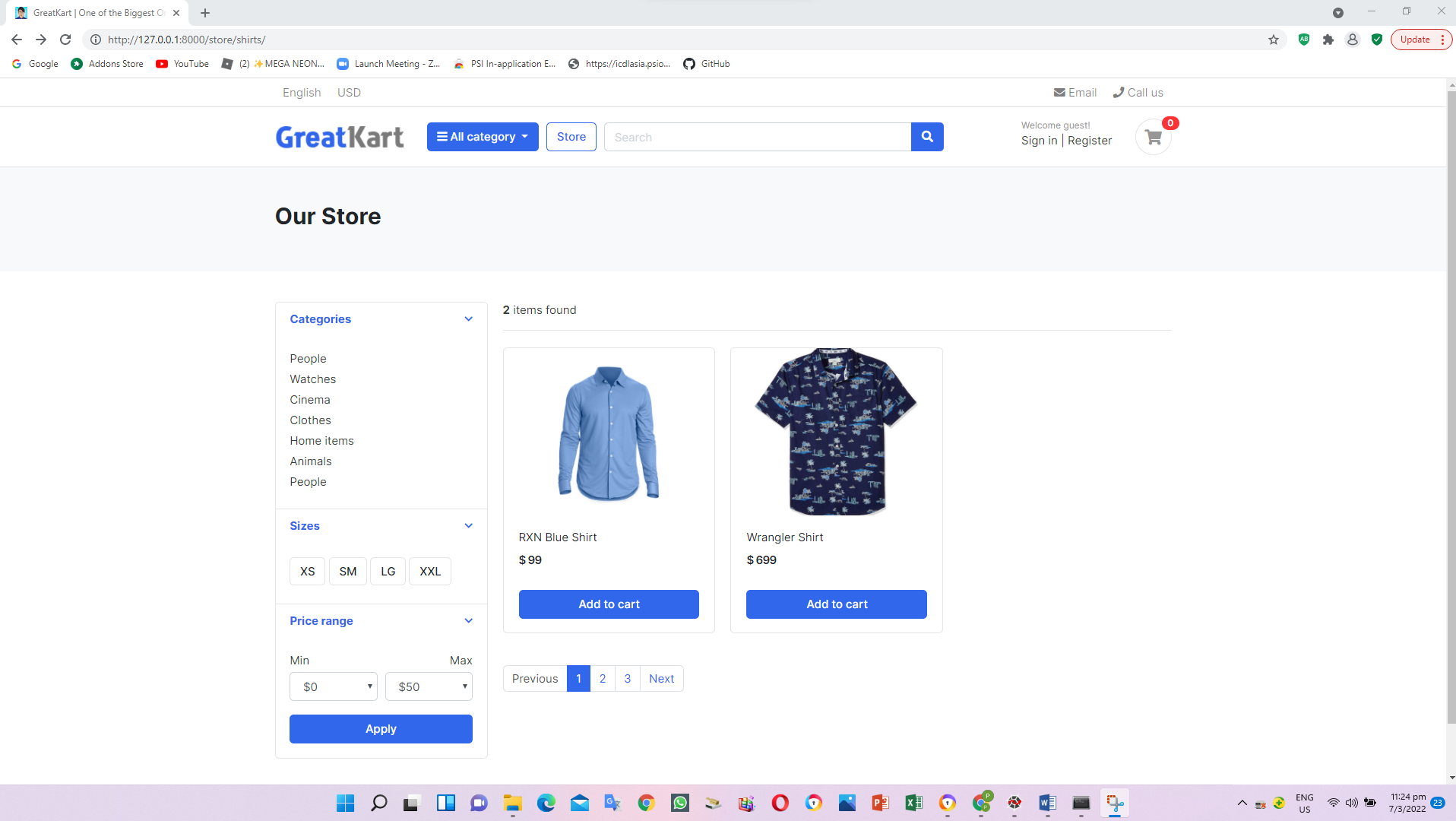
context = {

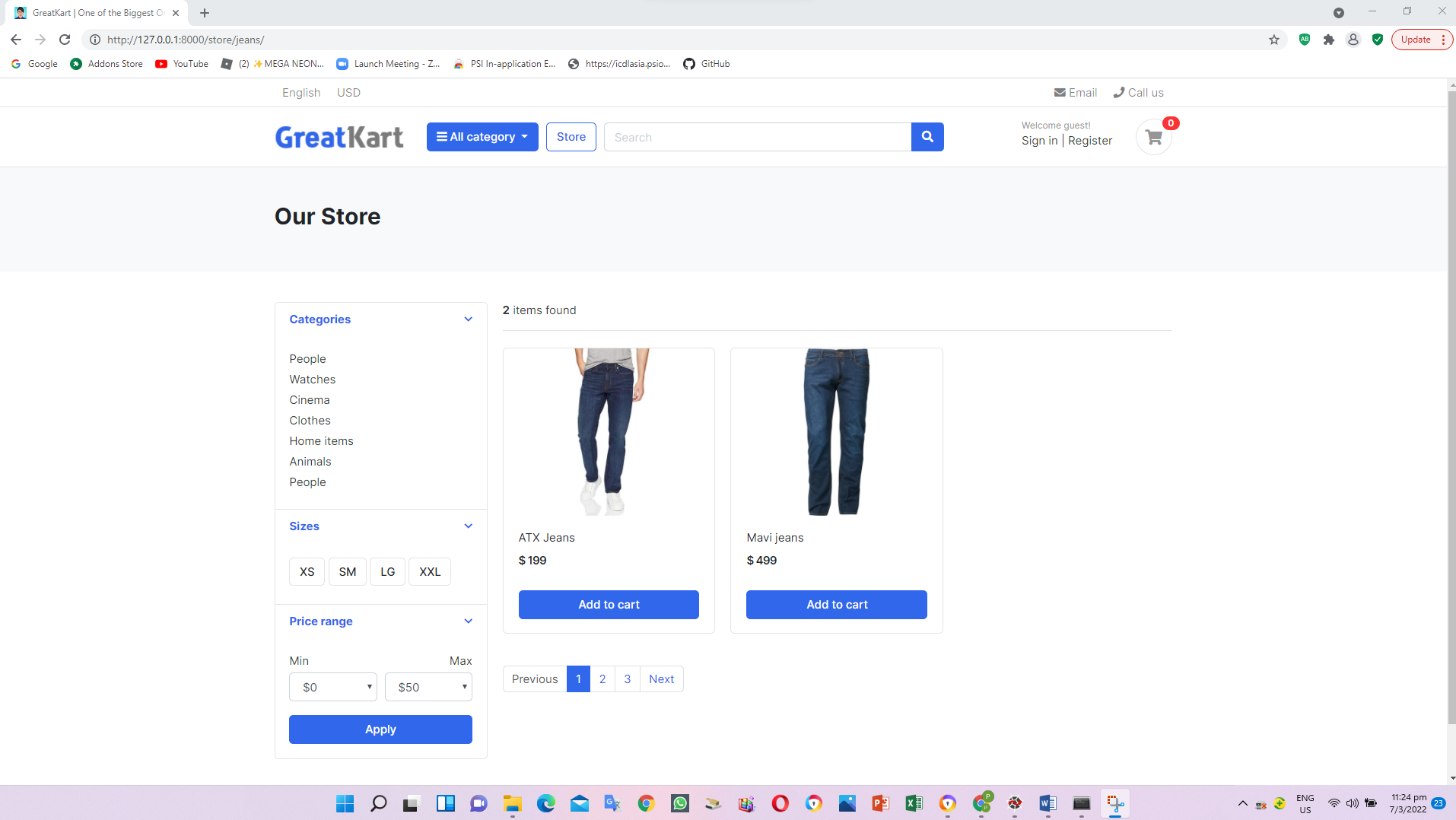
'products': products,

'product\_count': product\_count,

}

return render(request, 'store/store.html', context)





1. Create and add lines in the context\_processors.py under category folder (Create linkage between category to web click)

from .models import Category

def menu\_links(request):

links = Category.objects.all()

return dict(links=links)

1. Add line in the setting.py in the online folder

TEMPLATES = [

{

'BACKEND': 'django.template.backends.django.DjangoTemplates',

'DIRS': ['templates'],

'APP\_DIRS': True,

'OPTIONS': {

'context\_processors': [

'django.template.context\_processors.debug',

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

'django.contrib.messages.context\_processors.messages',

'category.context\_processors.menu\_links',

],

},

},

]

1. Edit the codes in the navbar.html under includes folder under templates folder

<i class="fa fa-bars"></i> All category

</button>

<div class="dropdown-menu">

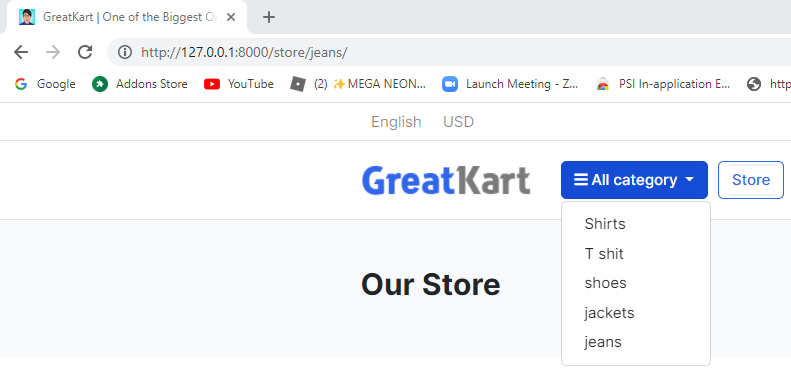
{% for category in links %}

<a class="dropdown-item" href="{{ category.get\_url }}">{{ category.category\_name}}</a>

{% endfor%}

</div>

</div> <!-- category-wrap.// -->



1. Adjust models.py under category folder

from django.db import models

from django.urls import reverse

# Create your models here.

class Category(models.Model):

category\_name = models.CharField(max\_length=50, unique = True)

slug = models.SlugField(max\_length=100, unique=True)

description = models.TextField(max\_length=255, blank=True)

cat\_image = models.ImageField(upload\_to='photos/categories', blank=True)

class Meta:

verbose\_name = 'Category'

verbose\_name\_plural = 'Categories'

def get\_url(self):

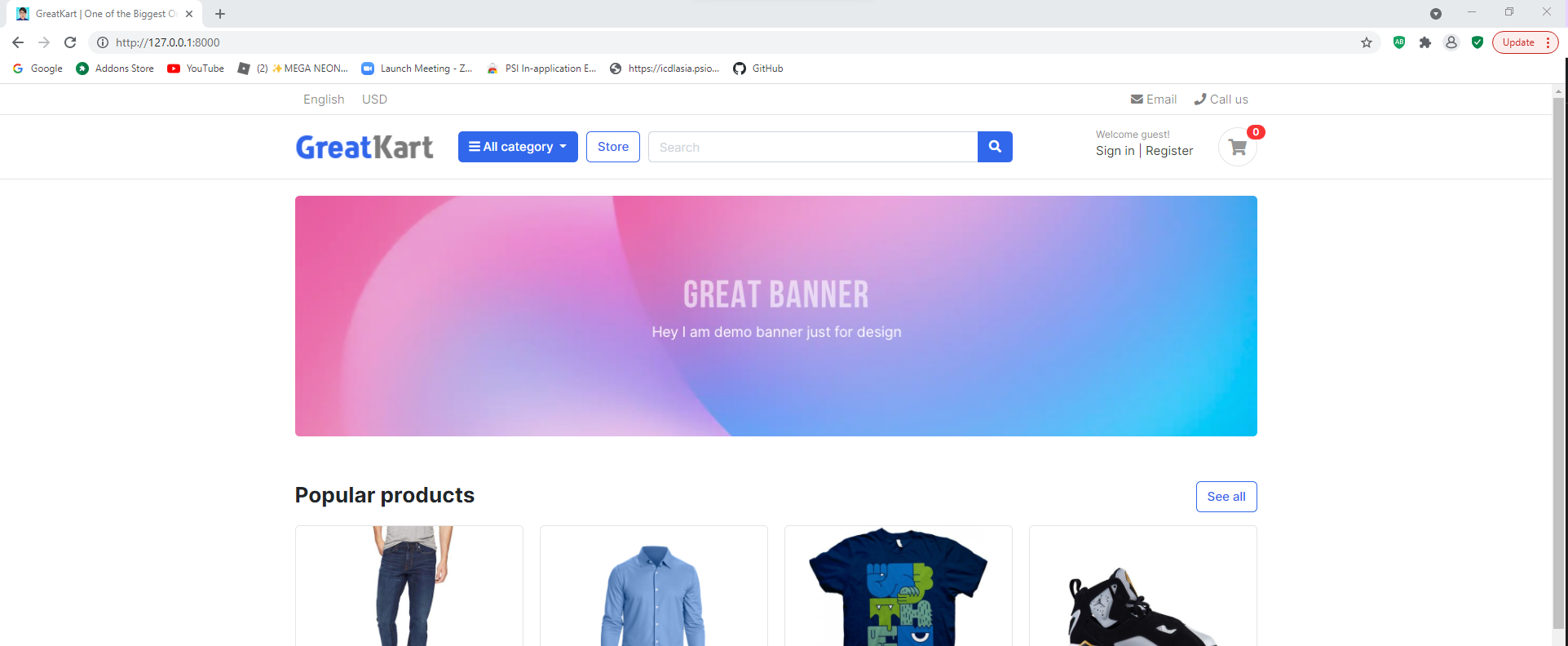
return reverse('products\_by\_category', args=[self.slug])

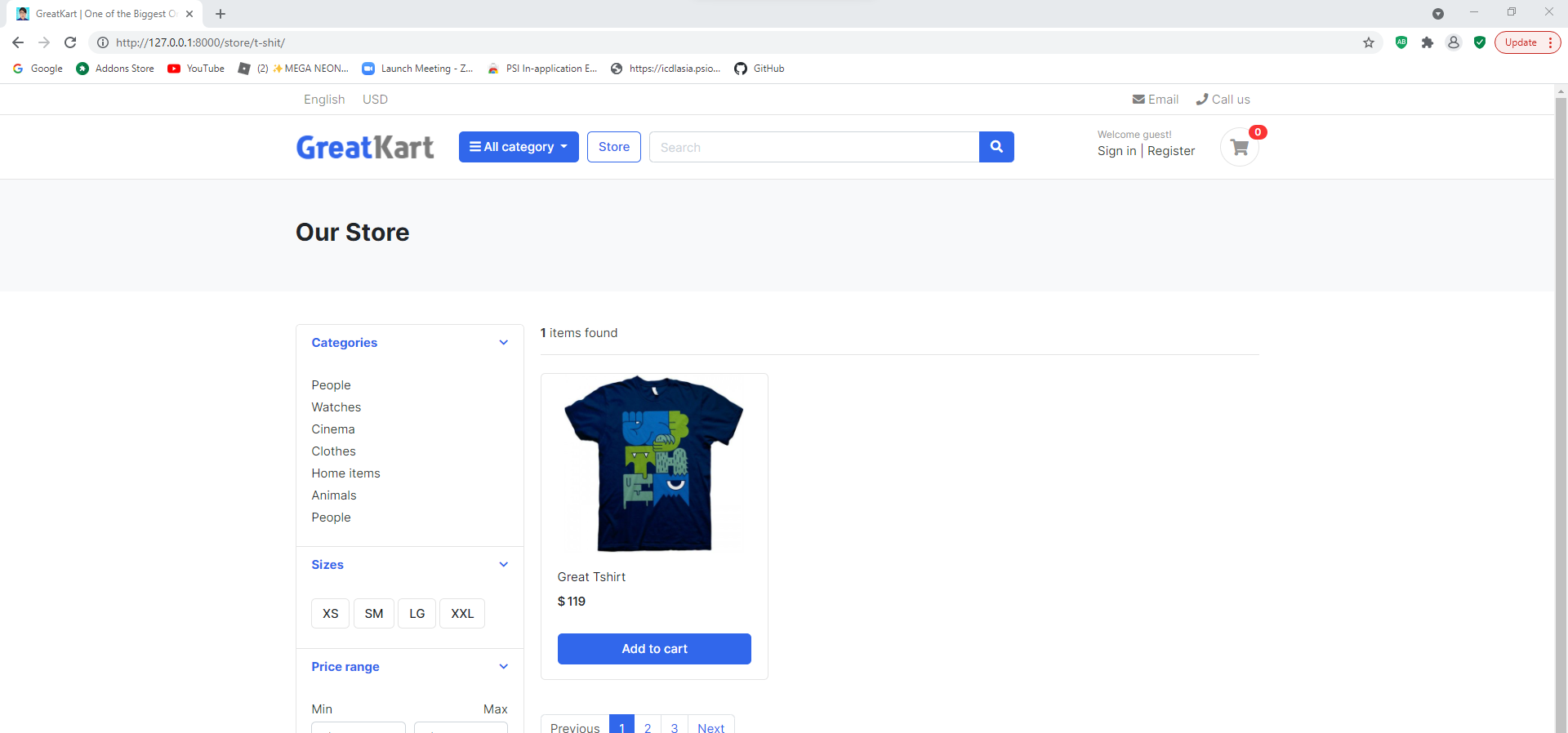
def \_\_str\_\_(self):

return self.category\_name

1. Adjust the home.html under templates folder (remove the s)

{% block content %}





1. Adjust codes in the store.html under store folder under templates folder (display category in the store page)

<div class="card-body">

<ul class="list-menu">

<li><a href="{% url 'store' %}">All Products </a></li>

{% for category in links %}

<li><a href="{{ category.get\_url }}">{{ category.category\_name }} </a></li>

{% endfor %}

</ul>

</div> <!-- card-body.// -->



1. Adjust codes in the navbar.html under includes folder under templates folder

<div class="dropdown-menu">

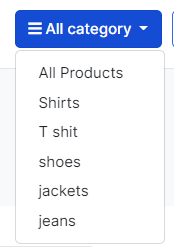
<a class="dropdown-item" href="{% url 'store' %}">All Products</a>

{% for category in links %}

<a class="dropdown-item" href="{{ category.get\_url }}">{{ category.category\_name}}</a>

{% endfor %}

</div>



1. Add line in the urls.py in the store folder (implement product detail URL and design)

from django.urls import path

from . import views

urlpatterns = [

path('', views.store, name='store'),

path('<slug:category\_slug>/', views.store, name='products\_by\_category'),

path('<slug:category\_slug>/<slug:product\_slug>/', views.product\_detail, name='product\_detail'),

]

1. Add lines in the views.py in the store folder

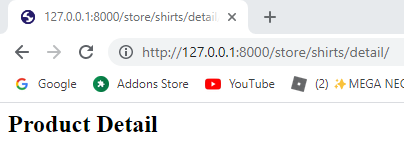
return render(request, 'store/store.html', context)

def product\_detail(request,category\_slug, product\_slug):

return render(request, 'store/product\_detail.html')

1. Create product\_details.html in the store folder in the templates folder

<h2>Product Detail</h2>

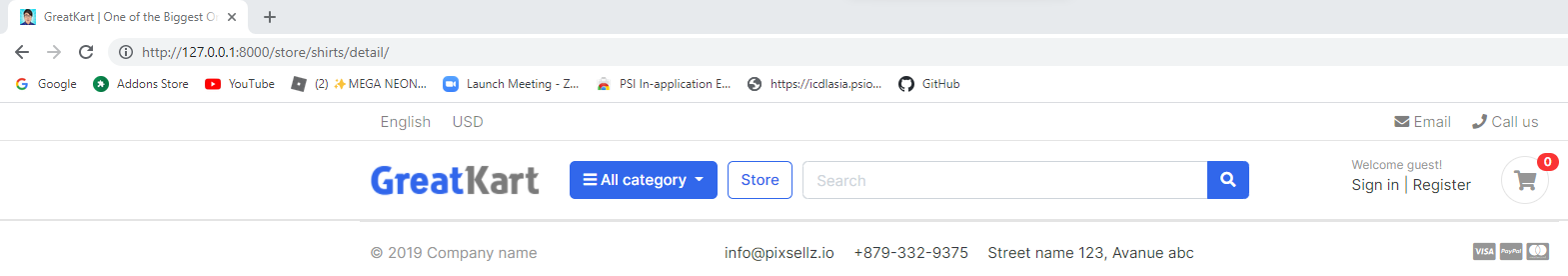


1. Replace the codes in the product\_details.html in the store folder in the templates folder

{% extends 'base.html' %}

{% block content %}

{% endblock %}



1. Copy the codes from product-detail.html from js folder from project sample folder to product\_details.html in the store folder in the templates folder

Copy

<section class="section-content padding-y bg">

…

<!-- ========================= SECTION CONTENT END// ========================= -->

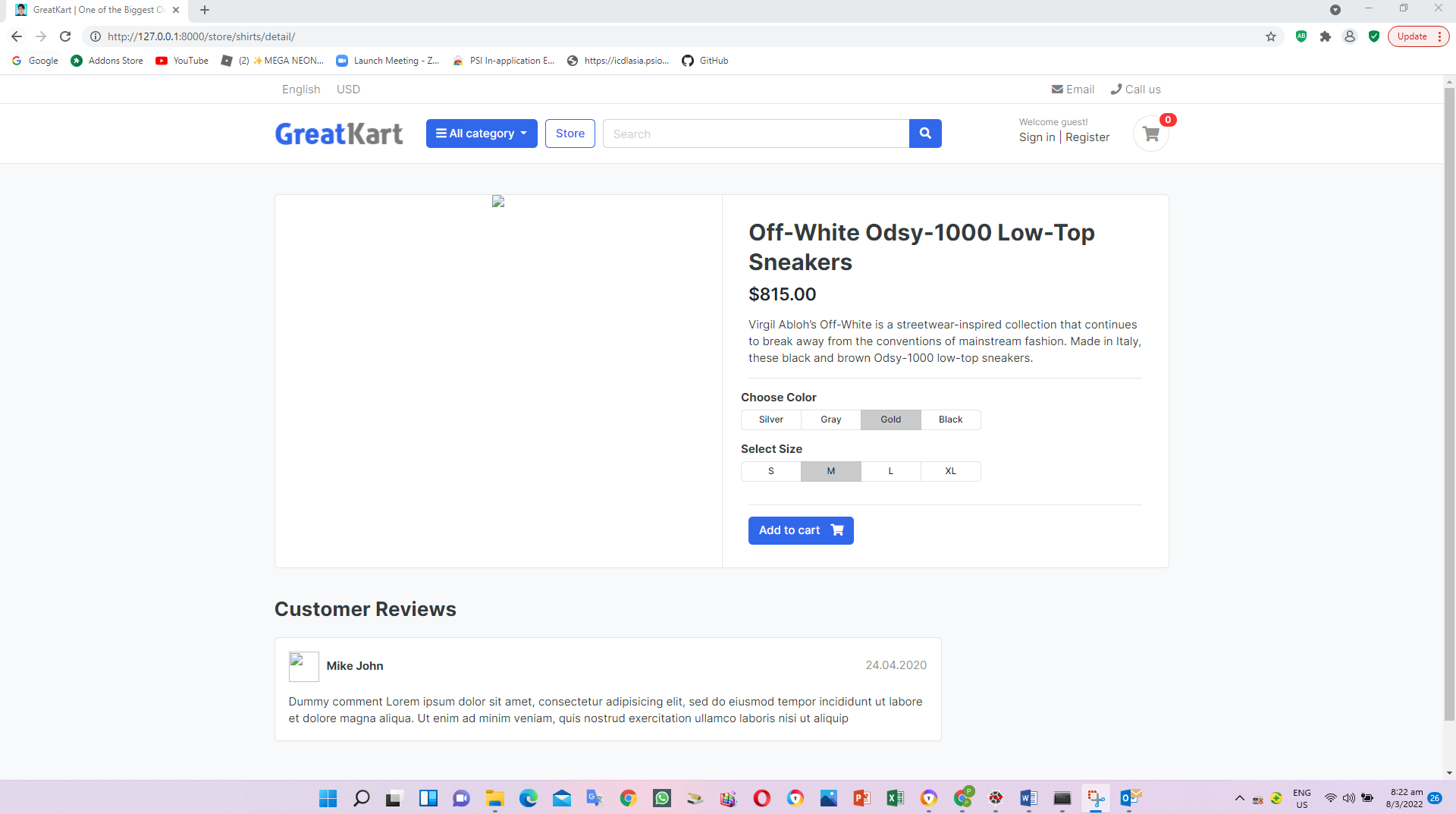
Paste

{% extends 'base.html' %}

{% block content %}

…

{% endblock %}



1. Adjust codes in the product\_details.html in the store folder in the templates folder

{% extends 'base.html' %}

{% load static %}

{% block content %}

<section class="section-content padding-y bg">

<div class="container">

<!-- ============================ COMPONENT 1 ================================= -->

<div class="card">

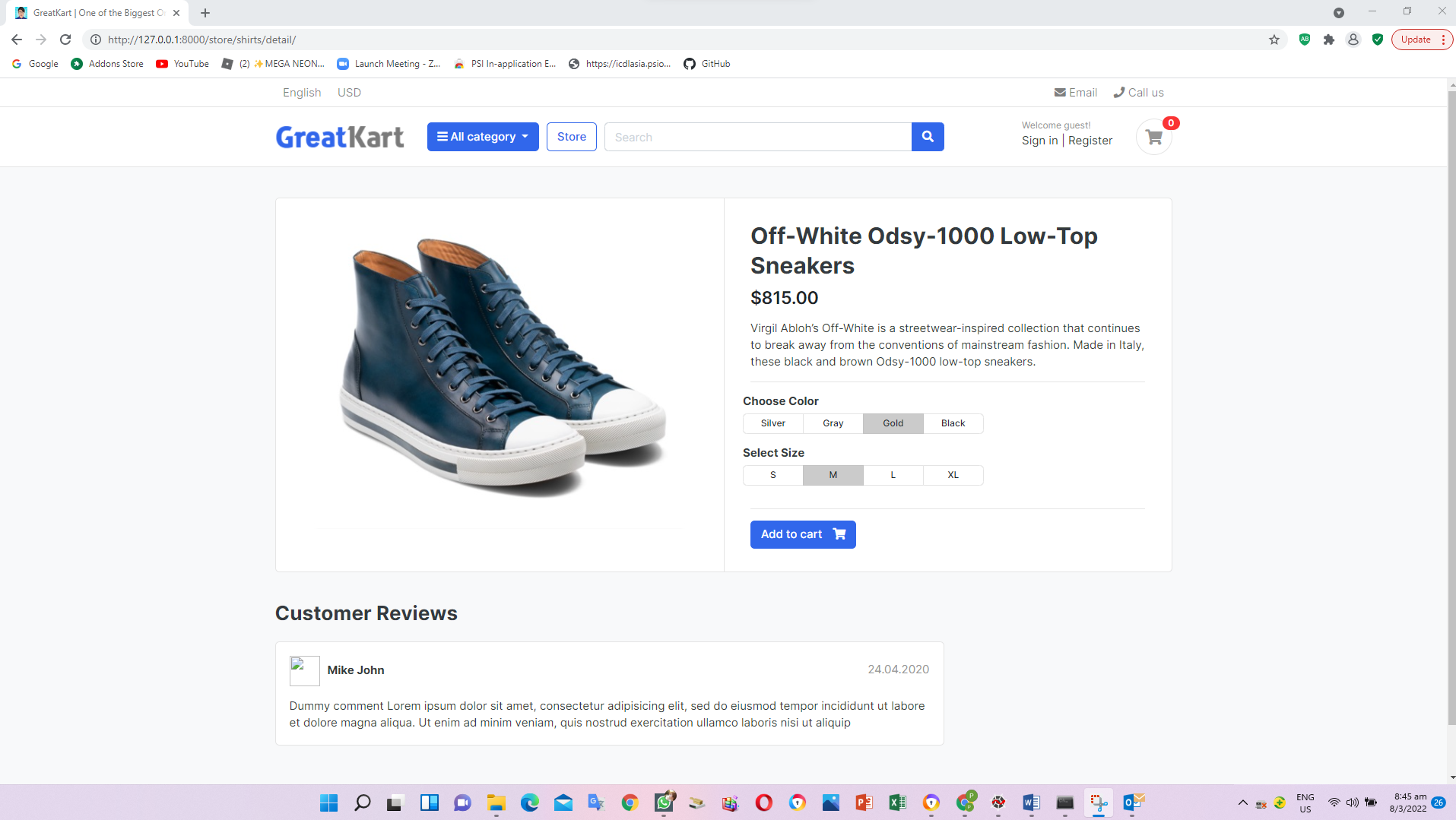
<div class="row no-gutters">

<aside class="col-md-6">

<article class="gallery-wrap">

<div class="img-big-wrap">

<a href="#"><img src="{% static './images/items/12.jpg' %}"></a>



1. Adjust codes in the product\_details.html in the store folder in the templates folder

<article class="box mb-3">

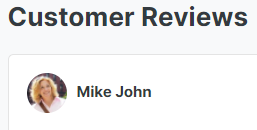
<div class="icontext w-100">

<img src="{% static './images/avatars/avatar1.jpg' %}" class="img-xs icon rounded-circle">

<div class="text">

<span class="date text-muted float-md-right">24.04.2020 </span>

<h6 class="mb-1">Mike John </h6>



1. Adjust codes in the views.py in the store folder (Single Product View)

def product\_detail(request,category\_slug, product\_slug):

try:

single\_product = Product.objects.get(category\_\_slug = category\_slug, slug=product\_slug)

except Exception as e:

raise e

context = {

'single\_product': single\_product,

}

return render(request, 'store/product\_detail.html', context)

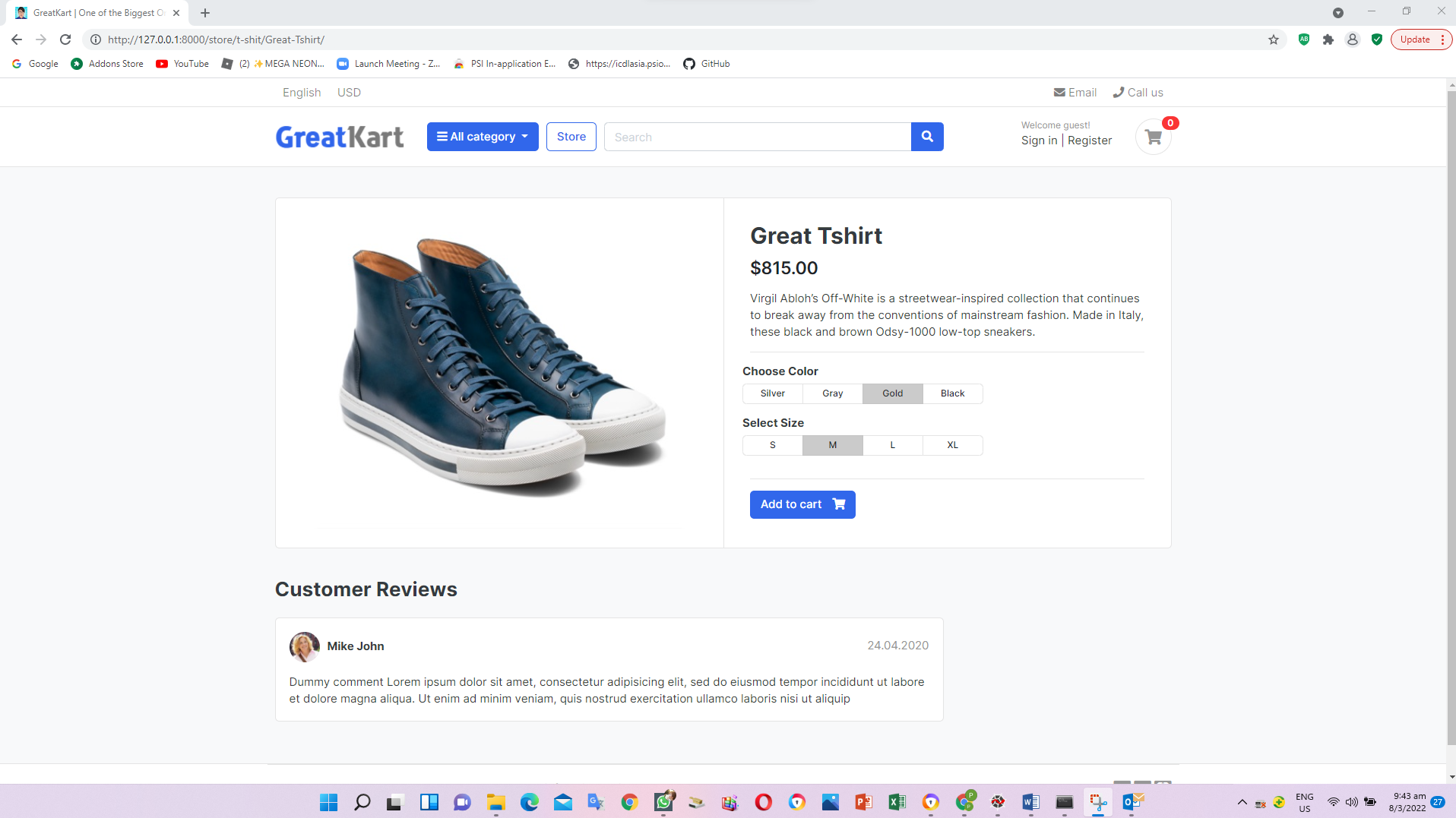
1. Adjust codes in the product\_details.html in the store folder in the templates folder

<article class="content-body">

<h2 class="title">{{ single\_product.product\_name }}</h2>

<div class="mb-3">

<var class="price h4">$815.00</var>



1. Adjust photo, price and description in the product\_details.html in the store folder in the templates folder

<article class="gallery-wrap">

<div class="img-big-wrap">

<a href="#"><img src="{{ single\_product.images.url }}"></a>

</div> <!-- img-big-wrap.// -->

</article> <!-- gallery-wrap .end// -->

</aside>

<main class="col-md-6 border-left">

<article class="content-body">

<h2 class="title">{{ single\_product.product\_name }}</h2>

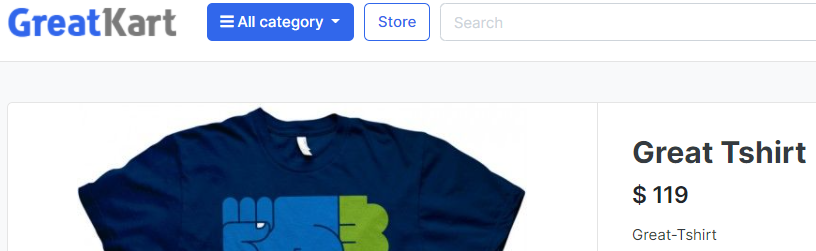
<div class="mb-3">

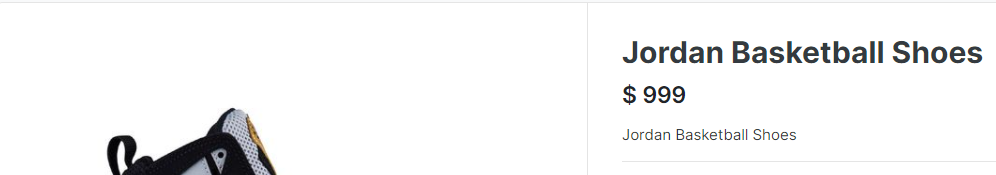
<var class="price h4">$ {{ single\_product.price }}</var>

</div>

<p>{{ single\_product.description }}</p>

<hr>





1. Adjust the line in the navbar.html under includes folder under templates folder (Fix Logo and store button)

Old

<a href="./store.html" class="btn btn-outline-primary">Store</a>

New

<a href="{% url 'store' %}" class="btn btn-outline-primary">Store</a>

Old

<a href="./" class="brand-wrap">

New

<a href="{% url 'home' %}" class="brand-wrap">

1. Adjust the line in the home.html under templates folder (Fix see all button)

Old

<a href="./store.html" class="btn btn-outline-primary float-right">See all</a>

New

<a href="{% url 'store' %}" class="btn btn-outline-primary float-right">See all</a>

1. Adjust store.html under store folder under templates folder (click photo and item tittle to see each item)

Old

<a href="./product-detail.html" class="title">{{ product.product\_name }}</a>

New

<a href="{{ product.get\_url }}" class="title">{{ product.product\_name }}</a>

Old

<img src="{{ product.images.url }}">

New

<a href="{{ product.get\_url }}"><img src="{{ product.images.url }}"></a>

User able to click the photo from our store page to individual page

1. User change the banker codes from home.html under templates folder (Change banner)

<img src={% static 'images/banners/cover.jpg' %} class="img-fluid rounded">

1. Product available in the product\_details.html in the store folder in the templates folder

{% if single\_product.stock <= 0 %}

<h5 class="text-danger">Out of Stock</h5>

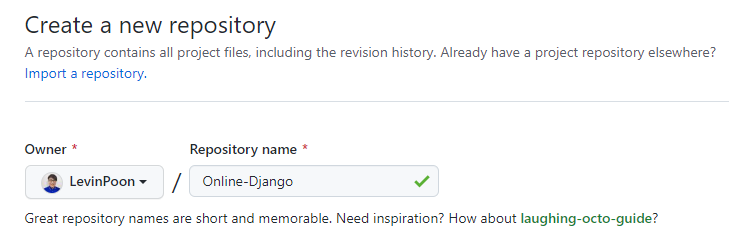
{% else %}

<a href="./product-detail.html" class="btn btn-primary"> <span class="text">Add to cart</span> <i class="fas fa-shopping-cart"></i> </a>

{% endif %}



1. Log in GitHub and create a new repository and press Create repository



1. Enter command in the Anaconda Prompt (Anaconda3)

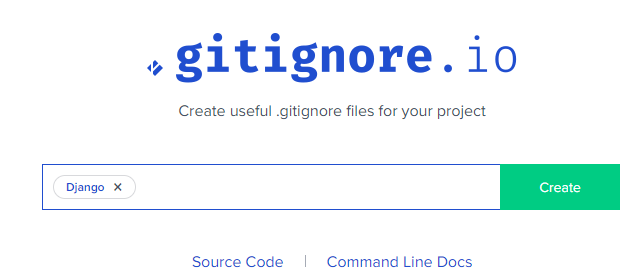
cd C:\Users\angel\OneDrive\Desktop\Python Django\env\Scripts

activate

cd C:\Users\angel\OneDrive\Desktop\Python Django

git init

1. Create an .gitignore file in online root folder (gitignore.io -> Django)



1. Copy and paste code from website to .gitignore file
2. Add codes to git repository

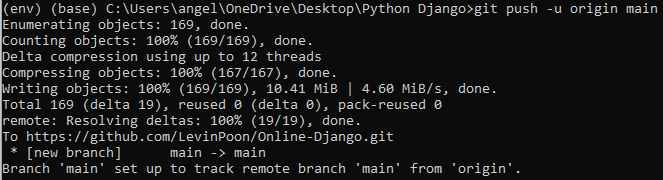
Git add –A

Git commit –m “this is my first commit”

git remote add origin https://github.com/LevinPoon/Online-Django.git

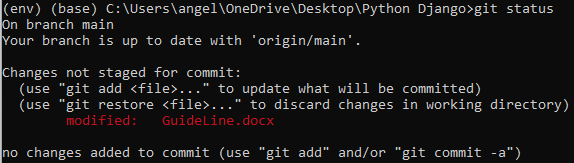
git branch -M main

git push -u origin main



User please reload the page

Git status check any updates



1. Enter CMD codes to create carts application folder

python manage.py startapp carts

1. Insert line in the setting.py under online folder

'store',

'carts',

]

1. Create urls.py in the carts folder

from django.urls import path

from . import views

urlpatterns = [

path('', views.cart, name='cart'),

]

1. Insert 1 line below store in the urls.py in the online folder

path('store/', include('store.urls')),

path('cart/', include('carts.urls')),

1. Edit views.py in the carts folder

from django.shortcuts import render

# Create your views here.

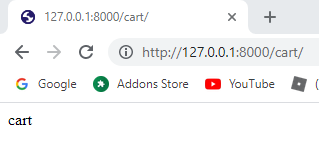
def cart(request):

return render(request, 'store/cart.html')

1. Create cart.html under store folder under templates folder

cart.html

cart



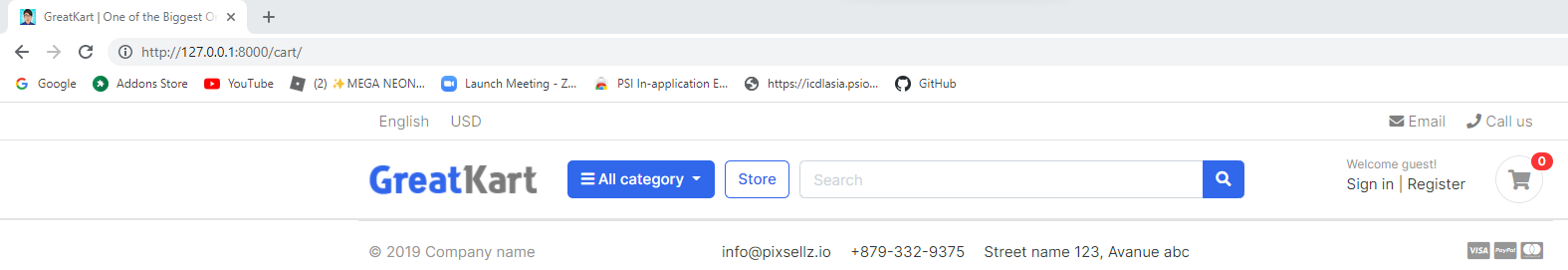
1. Adjust codes in cart.html under store folder under templates folder

{% extends 'base.html' %}

{% load static %}

{% block content %}

{% endblock %}



1. Copy session codes cart.html codes in project sample folder to cart.html under store folder under templates folder

<section class="section-content padding-y bg">

…

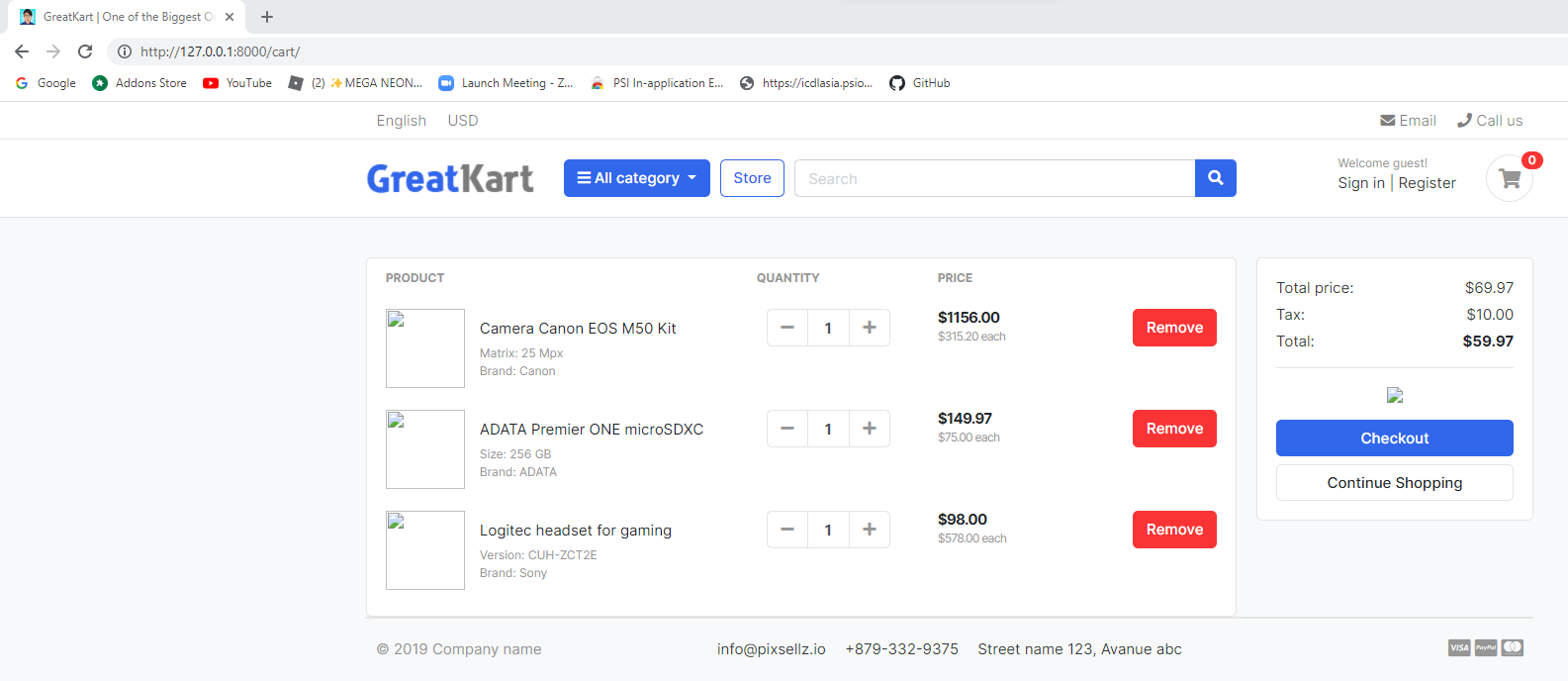
<!-- ========================= SECTION CONTENT END// ========================= -->

Paste in between

{% block content %}

…

{% endblock %}



1. Edit the cart.html to display photo in the store folder in the templates folder (display item and visa logo)

Old

<div class="aside"><img src="./images/items/11.jpg" class="img-sm"></div>

New

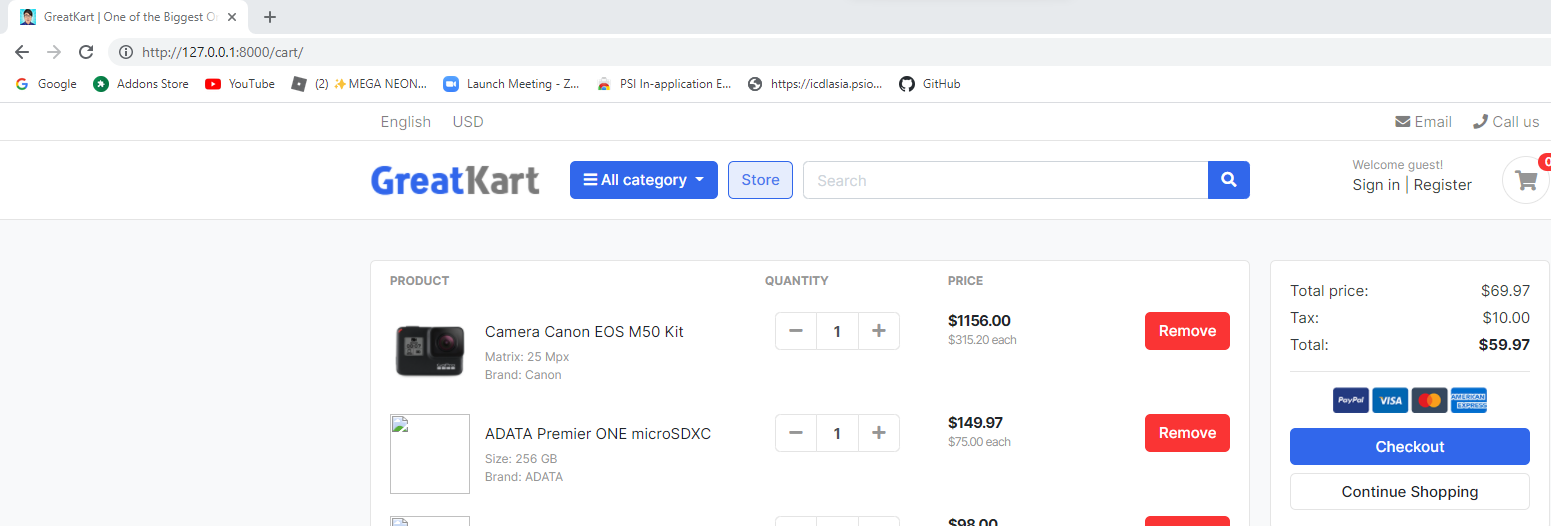
<div class="aside"><img src="{% static './images/items/11.jpg' %}" class="img-sm"></div>

Old

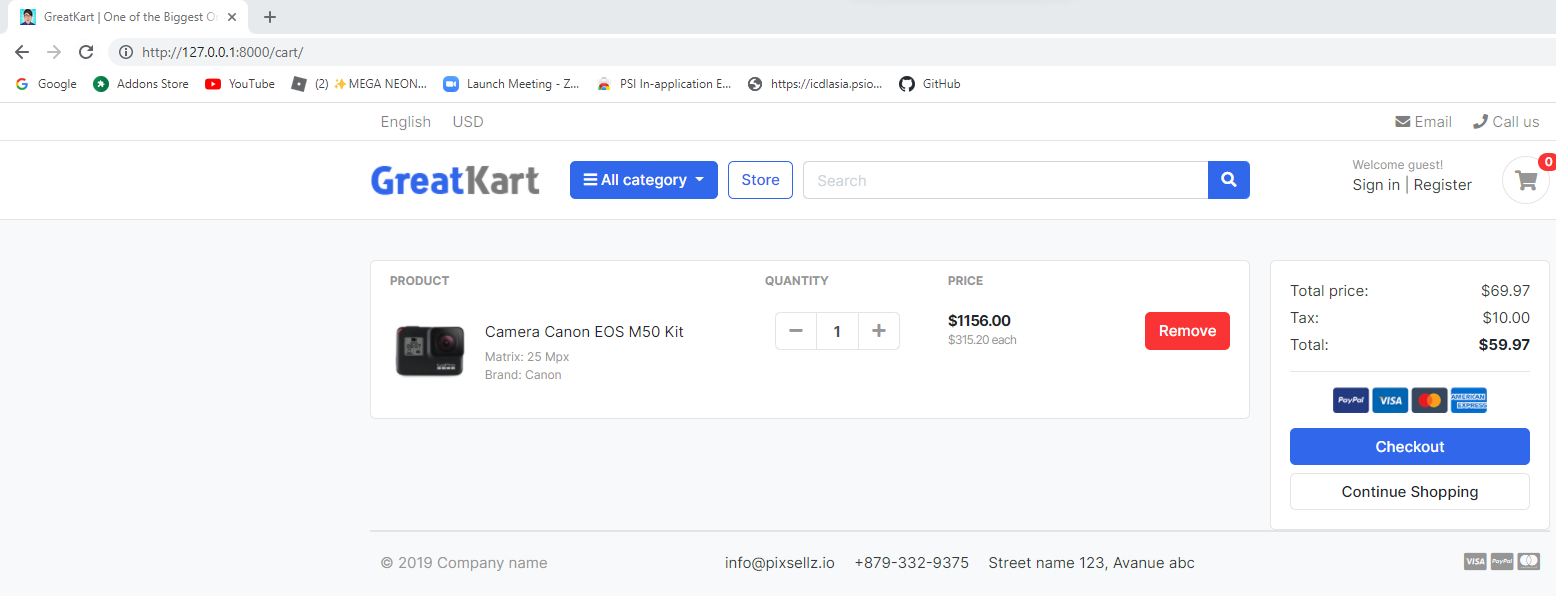
<img src="./images/misc/payments.png" height="26">

New

<img src="{% static './images/misc/payments.png' %}" height="26">



1. Delete 2nd tr to 3rd tr codes in the cart.html in the store folder in the templates folder



1. Add lines in models.py in the carts folder

from django.db import models

from store.models import Product

# Create your models here.

class Cart(models.Model):

cart\_id = models.CharField(max\_length=250, blank=True)

date\_added = models.DateField(auto\_now\_add=True)

def \_\_str\_\_(self):

return self.cart\_id

class CartItem(models.Model):

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

cart = models.ForeignKey(Cart, on\_delete=models.CASCADE)

quantity = models.IntegerField()

is\_active = models.BooleanField(default=True)

def \_\_str\_\_(self):

return self.product

1. Add lines in admin.py in the carts folder

from django.contrib import admin

from .models import Cart, CartItem

# Register your models here.

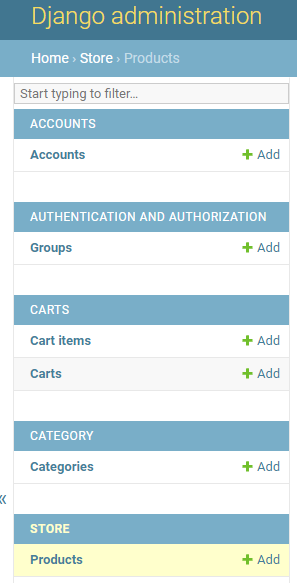
admin.site.register(Cart)

admin.site.register(CartItem)

1. Enter codes CMD to migrate in admin page

python manage.py makemigrations

python manage.py migrate



1. Add line to views.py under the carts folder (add item to cart from individual tab)

from django.shortcuts import render, redirect

from store.models import Product

from .models import Cart, CartItem

# Create your views here.

def \_cart\_id(request):

cart = request.session.session\_key

if not cart:

cart = request.session.create()

return cart

def add\_cart(request, product\_id):

product = Product.objects.get(id=product\_id) #get the product

try:

cart = Cart.objects.get(cart\_id=\_cart\_id(request)) #get the cart using the cart\_id present in the session

except Cart.DoesNotExist:

cart = Cart.objects.create(

cart\_id = \_cart\_id(request)

)

cart.save()

try:

cart\_item = CartItem.objects.get(product=product, cart=cart)

cart\_item.quantity += 1 # cart\_item.quantity = cart\_item.quantity + 1

cart\_item.save()

except CartItem.DoesNotExist:

cart\_item = CartItem.objects.create(

product = product,

quantity = 1,

cart = cart,

)

cart\_item.save()

return redirect('cart')

def cart(request):

return render(request, 'store/cart.html')

1. Add line to urls.py under the carts file

from django.urls import path

from . import views

urlpatterns = [

path('', views.cart, name='cart'),

path('add\_cart/<int:product\_id>/', views.add\_cart, name='add\_cart'),

]

1. Add lines in the product\_detail.html in the store folder in the templates folder

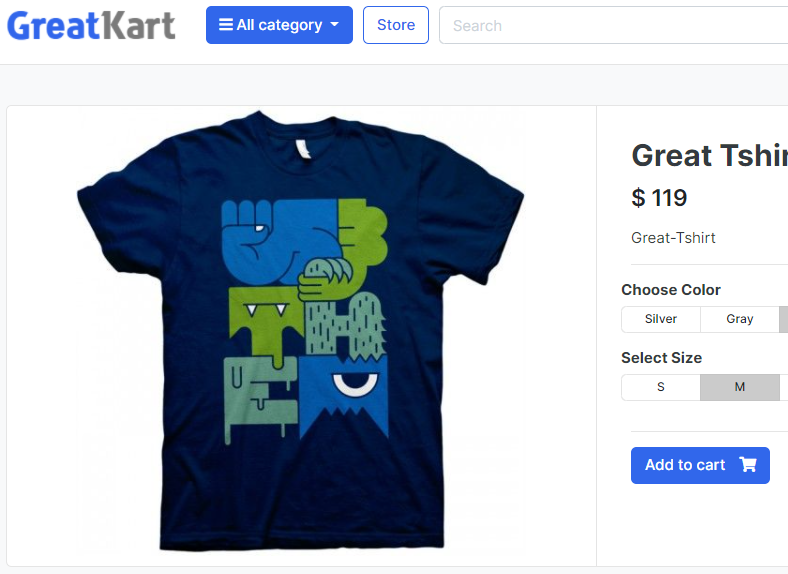
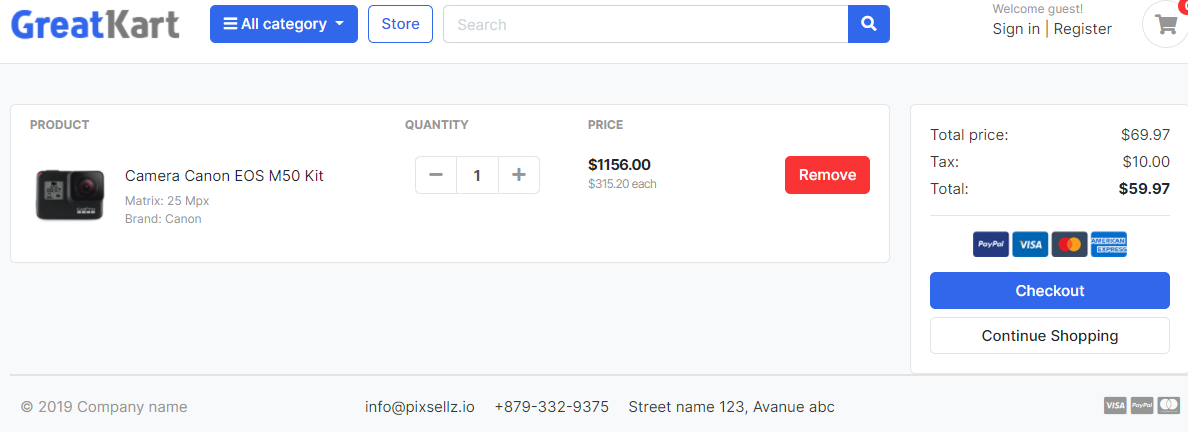
Old

<a href="./product-detail.html" class="btn btn-primary"> <span class="text">Add to cart</span> <i class="fas fa-shopping-cart"></i> </a>

New

<a href="{% url 'add\_cart' single\_product.id %}" class="btn btn-primary"> <span class="text">Add to cart</span> <i class="fas fa-shopping-cart"></i> </a>

User able to add to cart by press “Add to cart” button

1. Adjust line in the view.py under carts folder (Insert new item to cart)

From Django.core.exceptions import ObjectDoesNotExist

&

def cart(request, total=0, quantity=0, cart\_items=None):

try:

tax = 0

grand\_total = 0

cart = Cart.objects.get(cart\_id=\_cart\_id(request))

cart\_items = CartItem.objects.filter(cart=cart, is\_active=True)

for cart\_item in cart\_items:

total += (cart\_item.product.price \* cart\_item.quantity)

quantity += cart\_item.quantity

tax = (7 \* total)/100

grand\_total = total + tax

except ObjectDoesNotExist:

pass #just ignore

context = {

'total': total,

'quantity': quantity,

'cart\_items': cart\_items,

'tax': tax,

'grand\_total': grand\_total,

}

return render(request, 'store/cart.html', context)

1. Add for loop in front and end <tr> in the cart.html under store folder under templates folder

{% for cart\_item in cart\_items %}

<tr>

</tr>

{% endfor %}

1. Display cart item image in the cart.html under store folder under templates folder

Old

<div class="aside"><img src="{% static './images/items/11.jpg' %}" class="img-sm"></div>

New

<div class="aside"><img src="{{ cart\_item.product.images.url }}" class="img-sm"></div>

1. Display cart item name in the cart.html under store folder under templates folder

Old

<a href="#" class="title text-dark">Camera Canon EOS M50 Kit</a>

New

<a href="#" class="title text-dark">{{ cart\_item.product.product\_name }}</a>1

1. Display no of cart item in the cart.html under store folder under templates folder

Old

<input type="text" class="form-control" value="1">

New

<input type="text" class="form-control" value="{{ cart\_item.quantity }}">

1. Display total price of cart item in the cart.html under store folder under templates folder

Old

<var class="price">$1156.00</var>

<small class="text-muted"> $315.20 each </small>

New

<var class="price">$ {{ cart\_item.sub\_total }}</var>

<small class="text-muted"> $ {{ cart\_item.product.price }} each </small>

1. Add function codes in the models.py under carts folder

class CartItem(models.Model):

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

cart = models.ForeignKey(Cart, on\_delete=models.CASCADE)

quantity = models.IntegerField()

is\_active = models.BooleanField(default=True)

def sub\_total(self):

return self.product.price \* self.quantity

def \_\_str\_\_(self):

return self.product

1. Adjust codes to increase item in cart.html under store folder under templates folder

Old

<button class="btn btn-light" type="button" id="button-minus"> <i class="fa fa-plus"></i> </button>

New

<a href="{% url 'add\_cart' cart\_item.product.id %}" class="btn btn-light" type="button" id="button-minus"> <i class="fa fa-plus"></i> </a>

1. Adjust codes to display total price in cart.html under store folder under templates folder

Old

<dd class="text-right">$69.97</dd>

<dd class="text-right"> $10.00</dd>

<dd class="text-right text-dark b"><strong>$59.97</strong></dd>

New

<dd class="text-right">$ {{ total }}</dd>

<dd class="text-right"> $ {{ tax }}</dd>

<dd class="text-right text-dark b"><strong>${{ grand\_total }}</strong></dd>

1. Add lines in the view.py under the carts folder

from django.shortcuts import render, redirect, get\_object\_or\_404

&

def remove\_cart(request, product\_id):

cart = Cart.objects.get(cart\_id=\_cart\_id(request))

product = get\_object\_or\_404(Product, id=product\_id)

cart\_item = CartItem.objects.get(product=product, cart=cart)

if cart\_item.quantity > 1:

cart\_item.quantity -= 1

cart\_item.save()

else:

cart\_item.delete()

return redirect('cart')

1. Add lines in the view.py under the carts folder

path('add\_cart/<int:product\_id>/', views.add\_cart, name='add\_cart'),

path('remove\_cart/<int:product\_id>/', views.remove\_cart, name='remove\_cart'),

1. Adjust codes to display total price in cart.html under store folder under templates folder (decrement & remove)

Old

<button class="btn btn-light" type="button" id="button-plus"> <i class="fa fa-minus"></i> </button>

New

<a href="{% url 'remove\_cart' cart\_item.product.id %}" class="btn btn-light" type="button" id="button-plus"> <i class="fa fa-minus"></i> </a>

1. Add lines in the view.py under the carts folder

def remove\_cart\_item(request, product\_id):

cart = Cart.objects.get(cart\_id=\_cart\_id(request))

product = get\_object\_or\_404(Product, id=product\_id)

cart\_item = CartItem.objects.get(product=product, cart=cart)

cart\_item.delete()

return redirect('cart')

1. Add lines in the view.py under the carts folder

path('remove\_cart/<int:product\_id>/', views.remove\_cart, name='remove\_cart'),

path('remove\_cart\_item/<int:product\_id>/', views.remove\_cart\_item, name='remove\_cart\_item'),

1. Adjust codes to remove item in cart.html under store folder under templates folder

Old

<a href="" class="btn btn-danger"> Remove</a>

New

<a href="{% url 'remove\_cart\_item' cart\_item.product.id %}" class="btn btn-danger"> Remove</a>

1. Add if else code in the cart.html under store folder under templates folder

<!-- ============================ COMPONENT 1 ================================= -->

{% if not cart\_items %}

<h2 class="text-center" > Your Shopping Cart is Empty</h2>

<br>

<div class="text-center">

<a href="{% url 'store' %}" class="btn btn-primary">Continue Shopping</a>

</div>

<br>

{% else %}

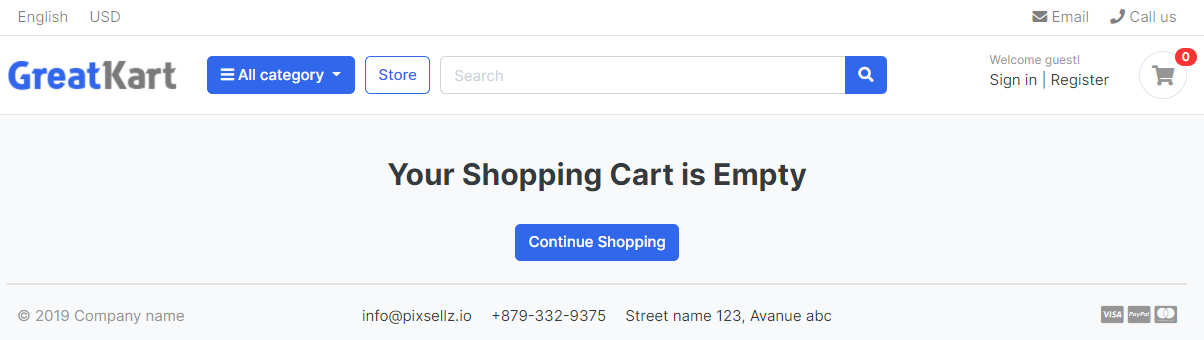
<div class="row">

…

</div> <!-- row.// -->

{% endif %}

<!-- ============================ COMPONENT 1 END .// ================================= -->



1. Adjust line to add to cart in the store page under store.html under store under templates

Old

<a href="#" class="btn btn-block btn-primary">Add to cart </a>

New

<a href="{% url 'add\_cart' product.id %}" class="btn btn-block btn-primary">Add to cart </a>

1. Zoom into each item page from cart page under cart.html under store folder under templates folder

Old

<a href="#" class="title text-dark">{{ cart\_item.product.product\_name }}</a>

New

<a href="{{ cart\_item.product.get\_url }}" class="title text-dark">{{ cart\_item.product.product\_name }}</a>

1. Continue shopping under cart.html under store folder under templates folder

Old

<a href="./store.html" class="btn btn-light btn-block">Continue Shopping</a>

New

<a href="{% url 'store' %}" class="btn btn-light btn-block">Continue Shopping</a>

1. Display added to cart logo under cart.html under store folder under templates folder

from django.shortcuts import render, get\_object\_or\_404

from .models import Product

from category.models import Category

from carts.models import CartItem

from carts.views import \_cart\_id

from django.http import HttpResponse

# Create your views here.

def store(request, category\_slug=None):

categories = None

products = None

if category\_slug != None:

categories = get\_object\_or\_404(Category, slug=category\_slug)

products = Product.objects.filter(category=categories, is\_available=True)

product\_count = products.count()

else:

products = Product.objects.all().filter(is\_available=True)

product\_count = products.count()

context = {

'products': products,

'product\_count': product\_count,

}

return render(request, 'store/store.html', context)

def product\_detail(request, category\_slug, product\_slug):

try:

single\_product = Product.objects.get(category\_\_slug = category\_slug, slug=product\_slug)

in\_cart = CartItem.objects.filter(cart\_\_cart\_id=\_cart\_id(request), product=single\_product).exists()

return HttpResponse(in\_cart)

exit()

except Exception as e:

raise e

context = {

'single\_product': single\_product,

'in\_cart': in\_cart,

}

return render(request, 'store/product\_detail.html', context)

User able to see true or false after add to cart to match with existing cart status.

1. Add lines in the product\_detail.html under store folder under templates folder

{% if single\_product.stock <= 0 %}

<h5 class="text-danger">Outof Stock</h5>

{% else %}

{% if in\_cart %}

<a href="#" class="btn btn-success"> <span class="text">Add to Cart</span> <i class="fas fa-check"></i> </a>

<a href="{% url 'cart' %}" class="btn btn-outline-primary"> <span class="text">View Cart</span> <i class="fas fa-eye"></i> </a>

{% else %}

<a href="{% url 'add\_cart' single\_product.id %}" class="btn btn-primary"> <span class="text">Add to Cart</span> <i class="fas fa-shopping-cart"></i> </a>

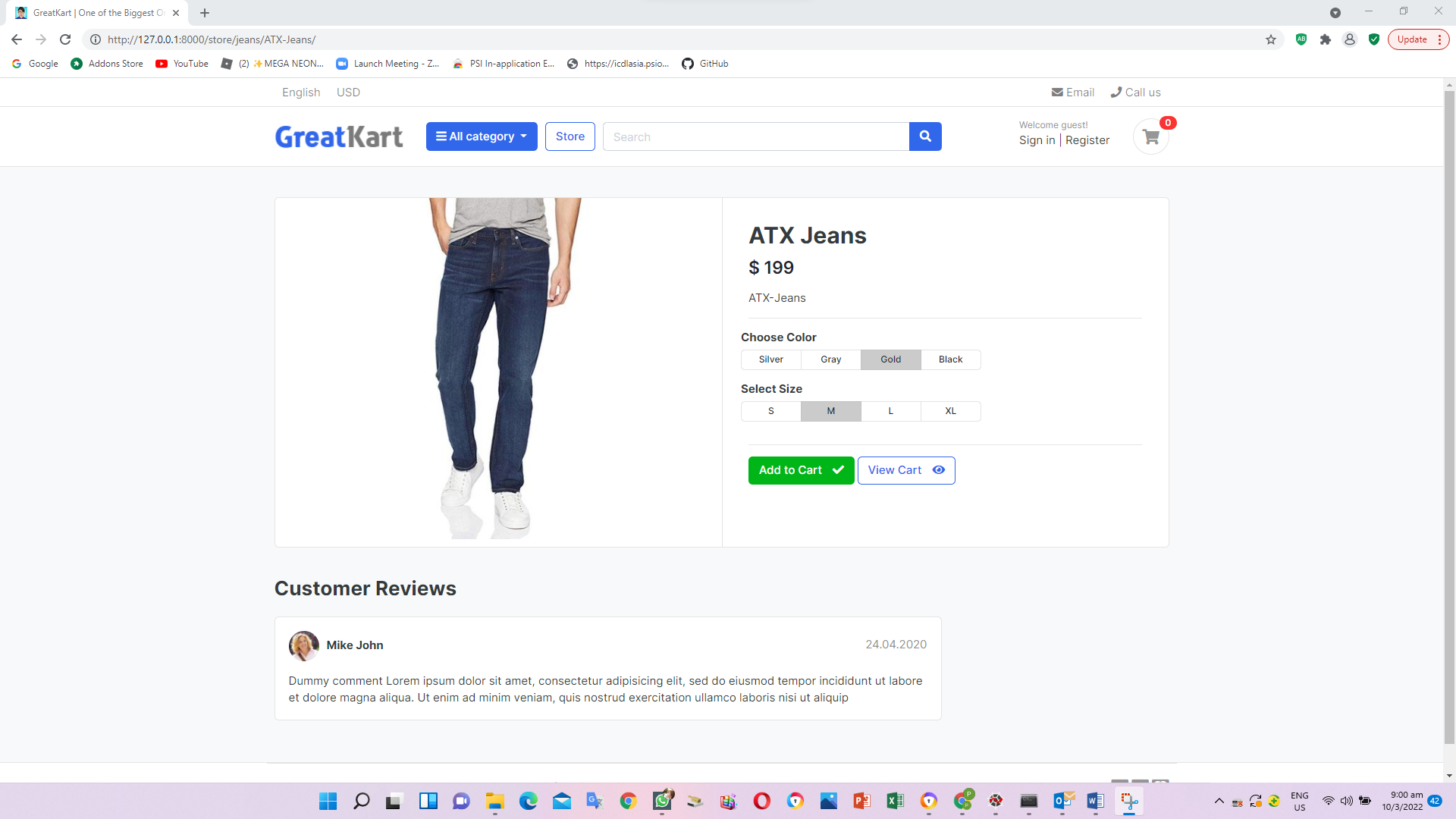
{% endif %}

{% endif %}

1. Remove code cart.html under store folder under templates folder

return HttpResponse(in\_cart)

exit()



1. Create context\_processors.py under carts folder (Linkage Cart Icon)

from .models import Cart, CartItem

from .views import \_cart\_id

def counter(request):

cart\_count = 0

if 'admin' in request.path:

return {}

else:

try:

cart = Cart.objects.filter(cart\_id=\_cart\_id(request))

cart\_items = CartItem.objects.all().filter(cart=cart[:1])

for cart\_item in cart\_items:

cart\_count += cart\_item.quantity

except Cart.DoesNotExist:

cart\_count = 0

return dict(cart\_count=cart\_count)

1. Add lines in settings.py under online folder

'category.context\_processors.menu\_links',

'carts.context\_processors.counter',

1. Add line in the navbar.html under includes under templates folder

Old

a href="./cart.html" class="widget-header pl-3 ml-3">

<span class="badge badge-pill badge-danger notify">0</span>

New

<a href="{% url 'cart' %}" class="widget-header pl-3 ml-3">

<span class="badge badge-pill badge-danger notify">{{ cart\_count }}</span>

1. Replace add to cart with view each item in the store.html under store folder under templates folder

Old

<a href="{% url 'add\_cart' product.id %}" class="btn btn-block btn-primary">Add to cart </a>

New

<a href="{{ product.get\_url }}" class="btn btn-block btn-primary">View Details </a>

1. Add lines in the view.py under store folder (view 6 items in the website)

from django.shortcuts import render, get\_object\_or\_404

from .models import Product

from category.models import Category

from carts.models import CartItem

from carts.views import \_cart\_id

from django.http import HttpResponse

from django.core.paginator import EmptyPage, PageNotAnInteger, Paginator

# Create your views here.

def store(request, category\_slug=None):

categories = None

products = None

if category\_slug != None:

categories = get\_object\_or\_404(Category, slug=category\_slug)

products = Product.objects.filter(category=categories, is\_available=True)

paginator = Paginator(products, 6)

page = request.GET.get('page')

paged\_products = paginator.get\_page(page)

product\_count = products.count()

else:

products = Product.objects.all().filter(is\_available=True).order\_by('id')

paginator = Paginator(products, 6)

page = request.GET.get('page')

paged\_products = paginator.get\_page(page)

product\_count = products.count()

context = {

'products': paged\_products,

'product\_count': product\_count,

}

return render(request, 'store/store.html', context)

def product\_detail(request, category\_slug, product\_slug):

try:

single\_product = Product.objects.get(category\_\_slug = category\_slug, slug=product\_slug)

in\_cart = CartItem.objects.filter(cart\_\_cart\_id=\_cart\_id(request), product=single\_product).exists()

except Exception as e:

raise e

context = {

'single\_product': single\_product,

'in\_cart': in\_cart,

}

return render(request, 'store/product\_detail.html', context)

1. Edit codes in the store.html under store folder under templates folder display paginator correctly

Old

<nav class="mt-4" aria-label="Page navigation sample">

<ul class="pagination">

<li class="page-item disabled"><a class="page-link" href="#">Previous</a></li>

<li class="page-item active"><a class="page-link" href="#">1</a></li>

<li class="page-item"><a class="page-link" href="#">2</a></li>

<li class="page-item"><a class="page-link" href="#">3</a></li>

<li class="page-item"><a class="page-link" href="#">Next</a></li>

</ul>

</nav>

New

<nav class="mt-4" aria-label="Page navigation sample">

{% if products.has\_other\_pages %}

<ul class="pagination">

{% if products.has\_previous %}

<li class="page-item"><a class="page-link" href="?page={{products.previous\_page\_number}}">Previous</a></li>

{% else %}

<li class="page-item disabled"><a class="page-link" href="#">Previous</a></li>

{% endif %}

{% for i in products.paginator.page\_range %}

{% if products.number == i %}

<li class="page-item active"><a class="page-link" href="#">{{i}}</a></li>

{% else %}

<li class="page-item"><a class="page-link" href="?page={{i}}">{{i}}</a></li>

{% endif %}

{% endfor %}

{% if products.has\_next %}

<li class="page-item"><a class="page-link" href="?page={{products.next\_page\_number}}">Next</a></li>

{% else %}

<li class="page-item disabled"><a class="page-link" href="#">Next</a></li>

{% endif %}

</ul>

{% endif %}

</nav>

1. Adjust lines in the urls.py under store folder (search function)

path('category/<slug:category\_slug>/', views.store, name='products\_by\_category'),

path('category/<slug:category\_slug>/<slug:product\_slug>/', views.product\_detail, name='product\_detail'),

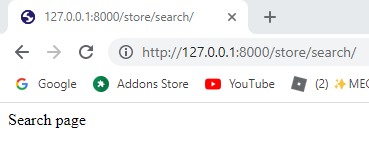
1. Add lines in the view.py under store folder (search function)

from django.http import HttpResponse

&

def search(request):

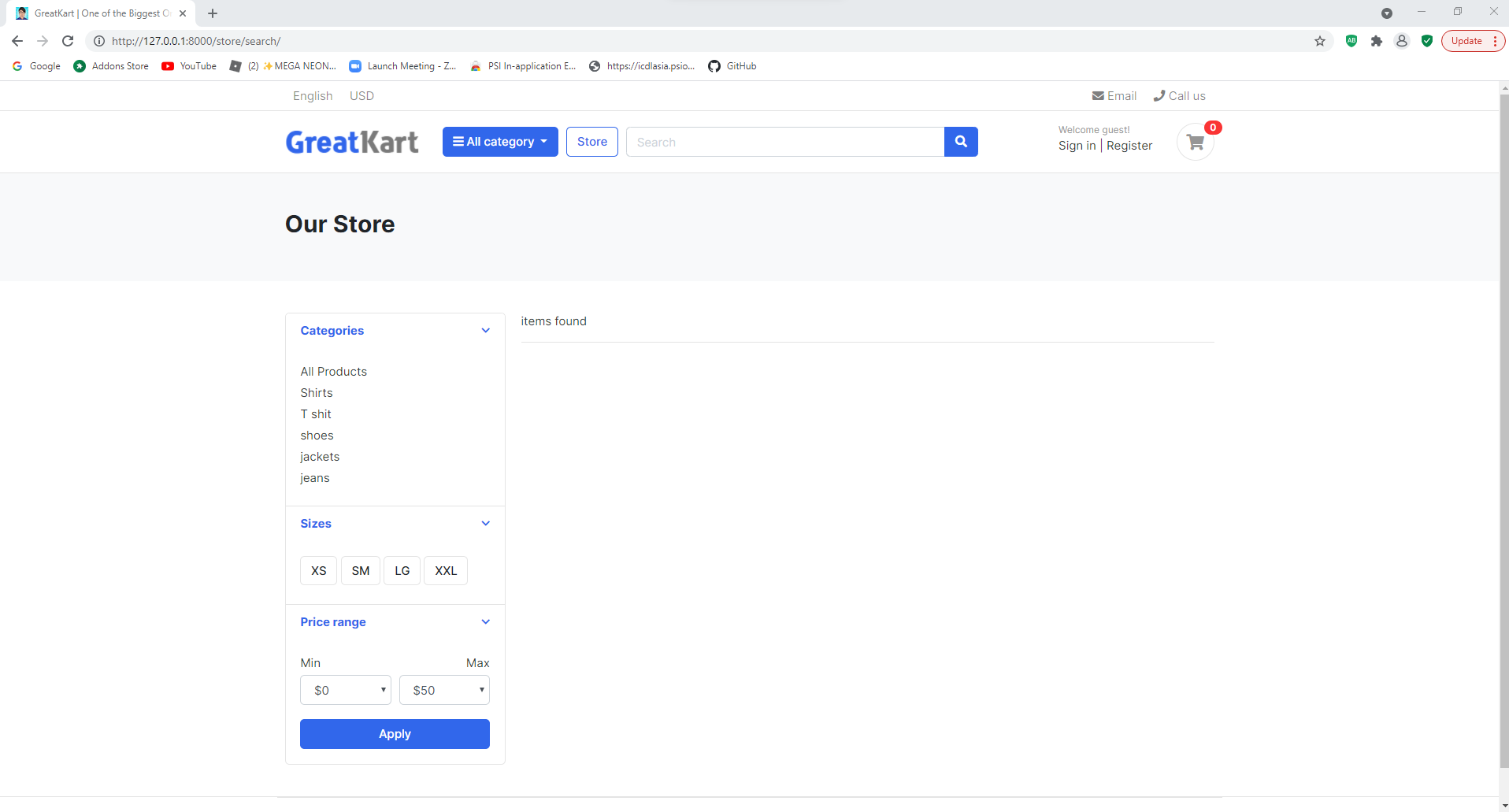
return HttpResponse('Search page')



1. Replace the codes in the view.py under store folder (search function)

def search(request):

return render(request, 'store/store.html')



1. Edit navbar.html under includes folder under templates

Old

<form action="#" class="search">

<div class="input-group w-100">

<input type="text" class="form-control" style="width:60%;" placeholder="Search">

New

<form action="{% url 'search' %}" class="search" method='GET'>

<div class="input-group w-100">

<input type="text" class="form-control" style="width:60%;" placeholder="Search" name="keyword">

1. Edit codes in views.py under store folder

Old

def search(request):

return render(request, 'store/store.html')

New

from django.db.models import Q

&

def search(request):

if 'keyword' in request.GET:

keyword = request.GET['keyword']

if keyword:

products = Product.objects.order\_by('-created\_date').filter(Q(description\_\_icontains=keyword) | Q(product\_name\_\_icontains=keyword))

product\_count = products.count()

context = {

'products': products,

'product\_count': product\_count,

}

return render(request, 'store/store.html', context)

1. Edits store.html under store folder under templates folder

<!-- ========================= SECTION PAGETOP ========================= -->

<section class="section-pagetop bg">

<div class="container">

{% if 'search' in request.path %}

<h2 class="title-page">Search Result</h2>

{% else %}

<h2 class="title-page">Our Store</h2>

{% endif %}

</div> <!-- container // -->

</section>

<!-- ========================= SECTION INTRO END// ========================= -->

&

<div class="row">

{% if products %}

{% for product in products %}

<div class="col-md-4">

<figure class="card card-product-grid">

<div class="img-wrap">

<a href="{{ product.get\_url }}"><img src="{{ product.images.url }}"></a>

</div> <!-- img-wrap.// -->

<figcaption class="info-wrap">

<div class="fix-height">

<a href="{{ product.get\_url }}" class="title">{{ product.product\_name }}</a>

<div class="price-wrap mt-2">

<span class="price">$ {{ product.price }}</span>

</div> <!-- price-wrap.// -->

</div>

<a href="{{ product.get\_url }}" class="btn btn-block btn-primary">View Details </a>

</figcaption>

</figure>

</div> <!-- col.// -->

{% endfor %}

{% else %}

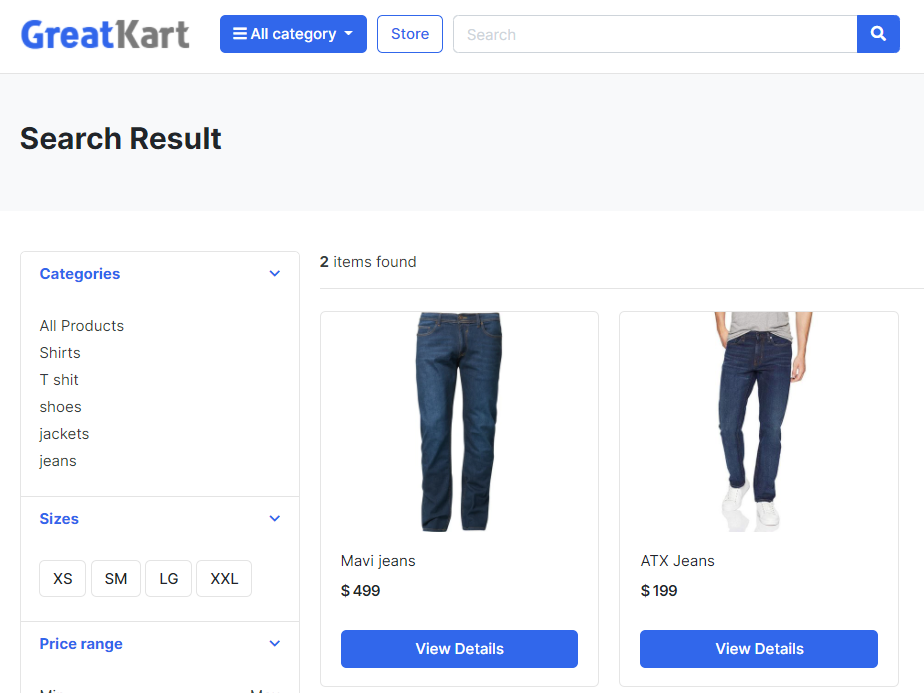
<div>

<h2> No result found. Please try again.</h2>

</div>

{% endif %}

</div> <!-- row end.// -->



1. Edit the product\_detail.html in store folder in templates folder (product variation color and size selection)

<form action="">

<article class="content-body">

…

</article> <!-- product-info-aside .// -->

</form>

1. Edit the product\_detail.html under store folder under templates folder (product variation)

Old

<a href="{% url 'add\_cart' single\_product.id %}" class="btn btn-primary"> <span class="text">Add to Cart</span> <i class="fas fa-shopping-cart"></i> </a>

New

<button type="submit” class="btn btn-primary"> <span class="text">Add to Cart</span> <i class="fas fa-shopping-cart"></i> </button>

&

Old

<form action="">

New

<form action="{% url 'add\_cart' single\_product.id %}" method="GET">

1. Remove the lines in the product\_detail.html under store folder under templates folder (color selection)

<div class="btn-group btn-group-sm btn-group-toggle" data-toggle="buttons">

<label class="btn btn-light">

<input type="radio" name="radio\_color"> Silver

</label>

<label class="btn btn-light">

<input type="radio" name="radio\_color" > Gray

</label>

<label class="btn btn-light active">

<input type="radio" name="radio\_color checked"> Gold

</label>

<label class="btn btn-light">

<input type="radio" name="radio\_color"> Black

</label>

</div>

1. Add the lines in the product\_detail.html under store folder under templates folder (color selection)

<div class="item-option-select">

<h6>Choose Color</h6>

<select name ="color">

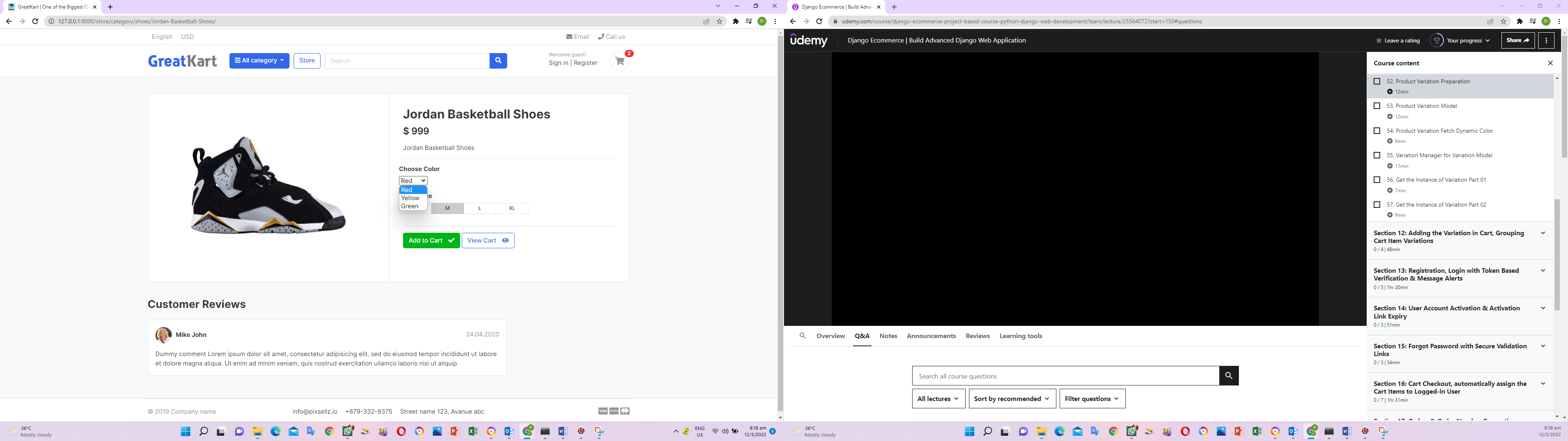
<option value="red">Red</option>

<option value="yellow">Yellow</option>

<option value="green">Green</option>

</select>

</div>



1. Remove the lines in the product\_detail.html under store folder under templates folder (size selection)

<div class="btn-group btn-group-sm btn-group-toggle" data-toggle="buttons">

<label class="btn btn-light">

<input type="radio" name="radio\_color"> S

</label>

<label class="btn btn-light active">

<input type="radio" name="radio\_color" checked> M

</label>

<label class="btn btn-light">

<input type="radio" name="radio\_color"> L

</label>

<label class="btn btn-light">

<input type="radio" name="radio\_color"> XL

</label>

</div>

1. Add the lines in the product\_detail.html under store folder under templates folder (size selection)

<div class="item-option-select">

<h6>Select Size</h6>

<select name ="size">

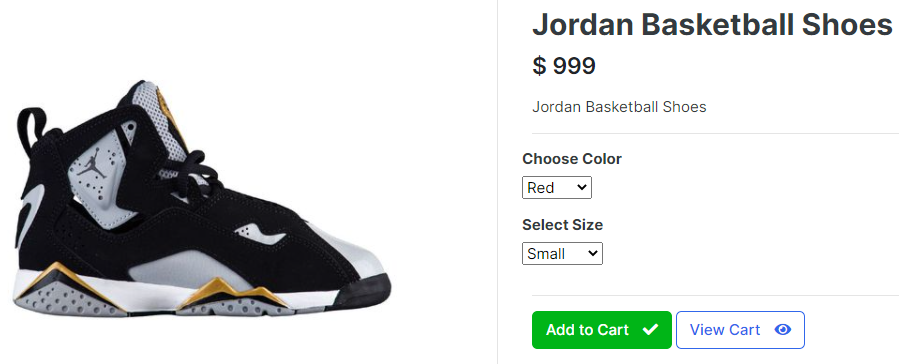
<option value="small">Small</option>

<option value="medium">Medium</option>

<option value="large">Large</option>

</select>

</div>



1. Replace the codes in the product\_detail.html under store folder under templates folder

Old

<select name ="color">

<select name ="size">

New

<select name ="size" class="form-control">

<select name ="color" class="form-control">

1. Add lines in the views.py from carts folder

from django.http import HttpResponse

&

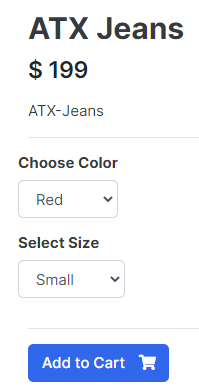
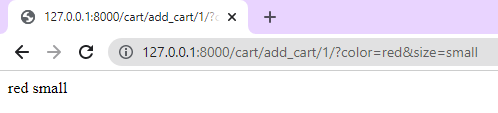
def add\_cart(request, product\_id):

color = request.GET['color']

size = request.GET['size']

return HttpResponse(color + ' ' + size)

exit()

1. Add lines in the models.py in the store folder

def \_\_str\_\_(self):

return self.product\_name

variation\_category\_choice = (

('color','color'),

('size','size'),

)

class Variation (models.Model):

product = models.ForeignKey(Product, on\_delete=models.CASCADE)

variation\_category = models.CharField(max\_length=100, choices=variation\_category\_choice)

variation\_value = models.CharField(max\_length=100)

is\_active = models.BooleanField(default=True)

created\_date = models.DateTimeField(auto\_now=True)

def \_\_unicode\_\_(self):

return self.product

1. Add lines in the admin.py in the store folder

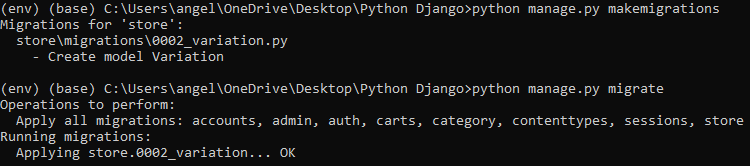
from .models import Product, Variation

admin.site.register(Variation)

1. Migrate the changes thru CMD

python manage.py makemigrations

python manage.py migrate



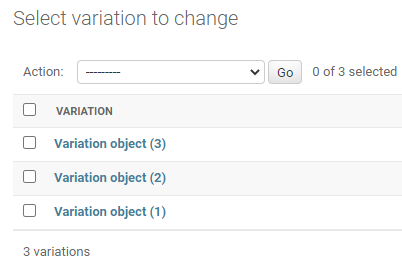
1. Log in admin page and ADD VARIATION in the Variations under STORE and press SAVE

Product: ATX Jeans

Variation category: color

Variation value: Red

Repeat the loop for Blue and Green



1. Add lines in the admin.py under store folder

class VariationAdmin(admin.ModelAdmin):

list\_display =('product', 'variation\_category', 'variation\_value', 'is\_active')

admin.site.register(Product, ProductAdmin)

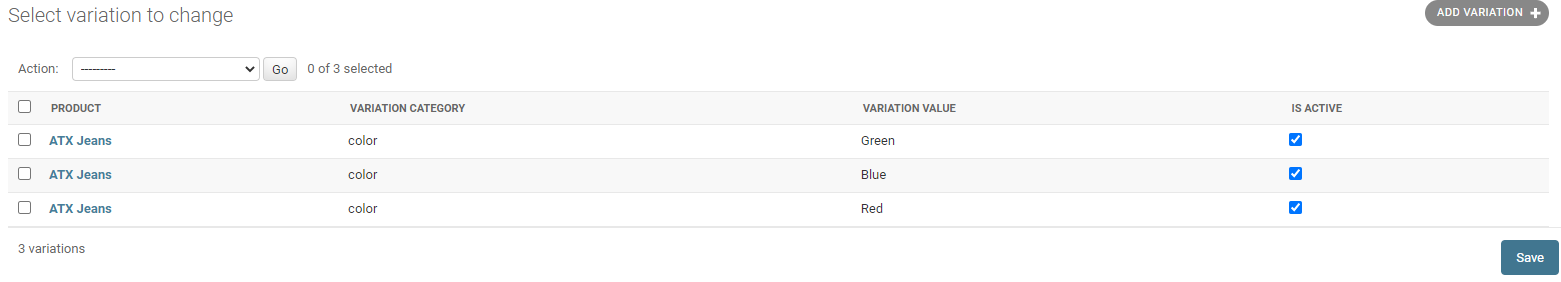
admin.site.register(Variation, VariationAdmin)



1. Add 1 line in the admin.py under store folder

list\_display =('product', 'variation\_category', 'variation\_value', 'is\_active')

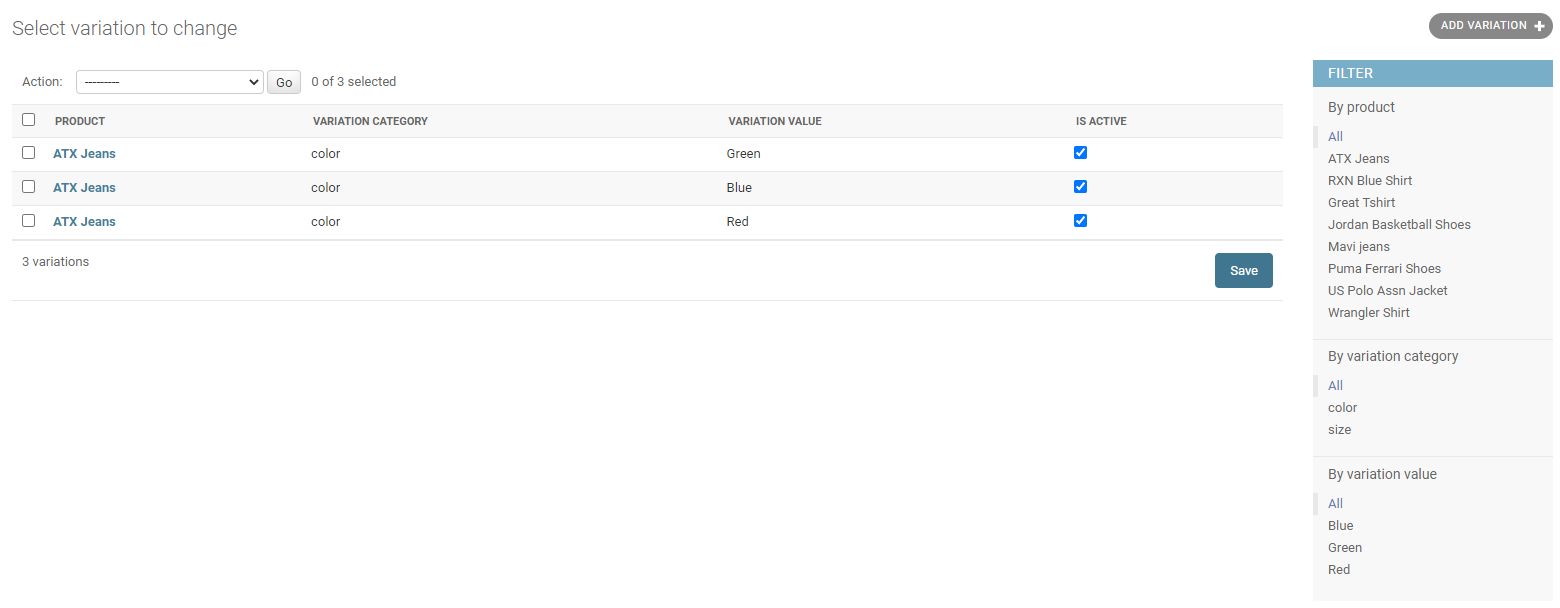
list\_editable = ('is\_active',)



1. Add 1 line in the admin.py under store folder

list\_editable = ('is\_active',)

list\_filter = ('product', 'variation\_category', 'variation\_value')



1. Edit lines in the product\_detail.html under store folder in template folder

Old

<select name ="color" class="form-control">

<option value="red">Red</option>

<option value="yellow">Yellow</option>

<option value="green">Green</option>

</select>

New

<select name ="color" class="form-control">

{% for i in single\_product.variation\_set.all %}

<option value="{{ i.variation\_value }}">{{ i.variation\_value }}</option>

{% endfor %}

</select>

& edit and add line as follow

<select name ="color" class="form-control" required>

<option value ="" disabled selected>Choose Color</option>

1. Add lines in the models.py under store folder

class VariationManager(models.Manager):

def colors(self):

return super (VariationManager, self).filter(variation\_category='color', is\_active=True)

def sizes(self):

return super (VariationManager, self).filter(variation\_category='size', is\_active=True)

1. Edit line in the product\_detail.html under store folder in template folder

Old

<div class="item-option-select">

<h6>Choose Color</h6>

<select name ="color" class="form-control" required>

<option value ="" disabled selected>Choose Color</option>

{% for i in single\_product.variation\_set.all %}

<option value="{{ i.variation\_value }}">{{ i.variation\_value }}</option>

{% endfor %}

</select>

</div>

</div> <!-- row.// -->

<div class="row">

<div class="item-option-select">

<h6>Select Size</h6>

<select name ="size" class="form-control">

<option value="small">Small</option>

<option value="medium">Medium</option>

<option value="large">Large</option>

</select>

</div>

New

<div class="item-option-select">

<h6>Choose Color</h6>

<select name ="color" class="form-control" required>

<option value ="" disabled selected>Choose Color</option>

{% for i in single\_product.variation\_set.colors %}

<option value="{{ i.variation\_value | lower }}">{{ i.variation\_value | capfirst }}</option>

{% endfor %}

</select>

</div>

</div> <!-- row.// -->

<div class="row">

<div class="item-option-select">

<h6>Select Size</h6>

<select name ="size" class="form-control">

{% for i in single\_product.variation\_set.sizes %}

<option value="{{ i.variation\_value | lower }}">{{ i.variation\_value | capfirst }}</option>

{% endfor %}

</select>

</div>

1. Insert 1 line in the models.py in the store folder

objects = VariationManager()

def \_\_str\_\_(self):

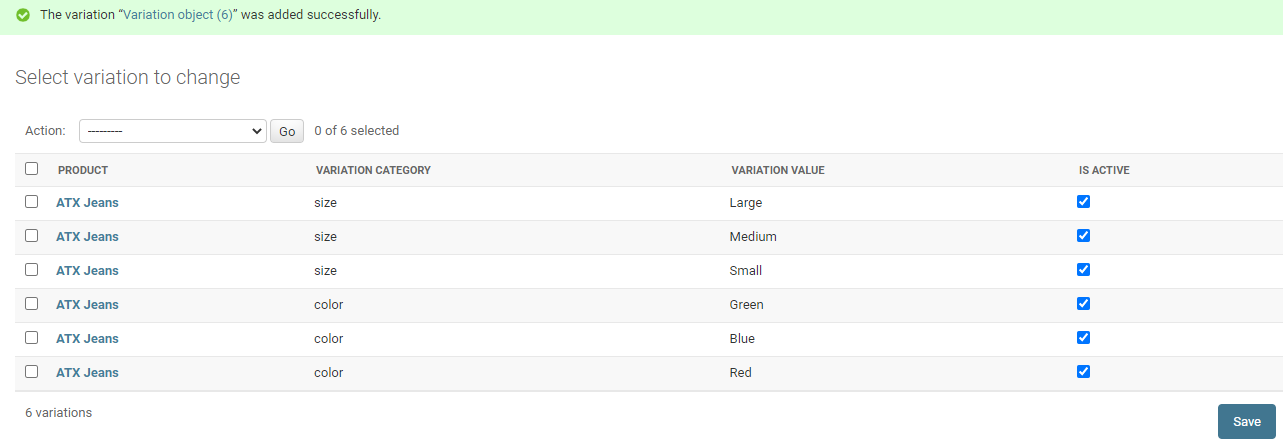
return self.variation\_value

1. Edit line in the product\_detail.html under store folder in template folder

<select name ="size" class="form-control" required>

<option value ="" disabled selected>Choose Size</option>

1. Add size variation in admin page as follow



1. Edit codes in the product\_detail.html under store folder in template folder

Old

<form action="{% url 'add\_cart' single\_product.id %}" method="GET">

New

<form action="{% url 'add\_cart' single\_product.id %}" method="POST">

{% csrf\_token %}

1. Edit codes in the views.py under carts folder

Old

def add\_cart(request, product\_id):

color = request.GET['color']

size = request.GET['size']

return HttpResponse(color + ' ' + size)

exit()

New

def add\_cart(request, product\_id):

if request.method == 'POST':

color = request.POST['color']

size = request.POST['size']

1. Replace codes in the views.py under carts folder

Old

if request.method == 'POST':

color = request.POST['color']

size = request.POST['size']

New

from store.models import Product, Variation

&

def add\_cart(request, product\_id):

product = Product.objects.get(id=product\_id) #get the product

product\_variation = []

if request.method == 'POST':

for item in request.POST:

key = item

value = request.POST[key]

try:

variation = Variation.objects.get(product=product, variation\_category\_\_iexact=key, variation\_value\_\_iexact=value)

product\_variation.append(variation)

except:

pass

1. Add line in the models.py under carts folder

from store.models import Product, Variation

&

variations = models.ManyToManyField(Variation, blank=True)

&

def \_\_unicode\_\_(self):

return self.product

1. Migrate the changes thru CMD

python manage.py makemigrations

python manage.py migrate



1. Edit admin.py under carts folder

from django.contrib import admin

from .models import Cart, CartItem

# Register your models here.

class CartAdmin(admin.ModelAdmin):

list\_display = ('cart\_id' , 'date\_added')

class CartItemAdmin(admin.ModelAdmin):

list\_display = ('product' , 'cart' , 'quantity','is\_active')

admin.site.register(Cart)

admin.site.register(CartItem, CartItemAdmin)

1. Add comments and codes in the view.py under carts folder

#product variation codes

product = Product.objects.get(id=product\_id) #get the product

&

#cart codes

try:

cart = Cart.objects.get(cart\_id=\_cart\_id(request)) #get the cart using the cart\_id present in the session

&

#cart item codes

try:

cart\_item = CartItem.objects.get(product=product, cart=cart)

&

cart\_item = CartItem.objects.get(product=product, cart=cart)

if len(product\_variation) > 0:

for item in product\_variation:

cart\_item.variations.add(item)

cart\_item.quantity += 1 # cart\_item.quantity = cart\_item.quantity + 1

&

cart = cart,

)

if len(product\_variation) > 0:

for item in product\_variation:

cart\_item.variations.add(item)

cart\_item.save()

return redirect('cart')

1. Edit the lines in product\_detail.html under store folder in template folder

Old

{% if single\_product.stock <= 0 %}

<h5 class="text-danger">Out of Stock</h5>

{% else %}

{% if in\_cart %}

<a href="#" class="btn btn-success"> <span class="text">Add to Cart</span> <i class="fas fa-check"></i> </a>

<a href="{% url 'cart' %}" class="btn btn-outline-primary"> <span class="text">View Cart</span> <i class="fas fa-eye"></i> </a>

{% else %}

<button type="submit" class="btn btn-primary"> <span class="text">Add to Cart</span> <i class="fas fa-shopping-cart"></i> </button>

{% endif %}

{% endif %}

New

{% if single\_product.stock <= 0 %}

<h5 class="text-danger">Out of Stock</h5>

{% else %}

<button type="submit" class="btn btn-primary"> <span class="text">Add to Cart</span> <i class="fas fa-shopping-cart"></i> </button>

{% endif %}

1. Edit views.py under carts folder

if len(product\_variation) > 0:

cart\_item.variations.clear()

for item in product\_variation:

cart\_item.variations.add(item)

cart\_item.quantity += 1 # cart\_item.quantity = cart\_item.quantity + 1

cart\_item.save()

except CartItem.DoesNotExist:

cart\_item = CartItem.objects.create(

product = product,

quantity = 1,

cart = cart,

)

if len(product\_variation) > 0:

cart\_item.variations.clear()

for item in product\_variation:

1. Edit cart.html under store folder under templates folder

Old

<p class="text-muted small">Matrix: 25 Mpx <br> Brand: Canon</p>

New

<p class="text-muted small">

{% if cart\_item.variations.all %}

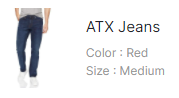
{% for item in cart\_item.variations.all %}

{{ item.variation\_category | capfirst }} : {{ item.variation\_value | capfirst }} <br>

{% endfor %}

{% endif %}

</p>



1. Edit views.py under carts folder

Old

#cart item codes

try:

cart\_item = CartItem.objects.get(product=product, cart=cart)

New

#cart item codes

try:

cart\_item = CartItem.objects.create(product=product, quantity=1, cart=cart)

if len(product\_variation) > 0:

cart\_item.variations.clear()

for item in product\_variation:

cart\_item.variations.add(item)

#cart\_item.quantity += 1 # cart\_item.quantity = cart\_item.quantity + 1



1. Edit codes in views.py under carts folder

#cart item codes

is\_cart\_item\_exists = CartItem.objects.filter(product=product, cart=cart).exists()

if is\_cart\_item\_exists:

cart\_item = CartItem.objects.filter(product=product, cart=cart)

# existing\_variation -> database

# current variation -> product\_variation

# item\_id -> database

ex\_var\_list = []

id = []

for item in cart\_item:

existing\_variation = item.variations.all()

ex\_var\_list.append(list(existing\_variation))

id.append(item.id)

print(ex\_var\_list)

if product\_variation in ex\_var\_list:

# increase the cart item quantity

index = ex\_var\_list.index(product\_variation)

item\_id = id[index]

item = CartItem.objects.get(product=product, id=item\_id)

item.quantity += 1

item.save()

else:

item = CartItem.objects.create(product=product, quantity=1, cart=cart)

if len(product\_variation) > 0:

item.variations.clear()

item.variations.add(\*product\_variation)

item.save()

else:

cart\_item = CartItem.objects.create(

product = product,

quantity = 1,

cart = cart,

)

if len(product\_variation) > 0:

cart\_item.variations.clear()

cart\_item.variations.add(\*product\_variation)

cart\_item.save()

return redirect('cart')

The - + remove button not working.

1. Edit codes in the cart.html in the store folder under the templates folder

<div class="input-group-append">

<form action="{% url 'add\_cart' cart\_item.product.id %}" method="POST">

{% csrf\_token %}

{% for item in cart\_item.variations.all %}

<input type="hidden" name="{{ item.variation\_category | lower }}" value="{{ item.variation\_value | capfirst }}">

{% endfor %}

<button class="btn btn-light" type="submit" id="button-minus"> <i class="fa fa-plus"></i> </button>

</form>

</div>

User able to add item no in the cart page

1. Edit views.py in the carts folder

def remove\_cart(request, product\_id, cart\_item\_id):

1. Edit codes in the cart.html in the store folder under the templates folder

<a href="{% url 'remove\_cart' cart\_item.product.id cart\_item.id %}" class="btn btn-light" type="button" id="button-plus"> <i class="fa fa-minus"></i> </a>

1. Edit codes in the urls.py under carts folder

path('remove\_cart/<int:product\_id>/<int:cart\_item\_id>/', views.remove\_cart, name='remove\_cart'),

1. Edit views.py in the carts folder

def remove\_cart(request, product\_id, cart\_item\_id):

cart = Cart.objects.get(cart\_id=\_cart\_id(request))

product = get\_object\_or\_404(Product, id=product\_id)

try:

cart\_item = CartItem.objects.get(product=product, cart=cart, id=cart\_item\_id)

if cart\_item.quantity > 1:

cart\_item.quantity -= 1

cart\_item.save()

else:

cart\_item.delete()

except:

pass

return redirect('cart')

User able to decrease the no of item in the cart page

1. Edit codes in the cart.html in the store folder under the templates folder

<a href="{% url 'remove\_cart\_item' cart\_item.product.id cart\_item.id %}" class="btn btn-danger"> Remove</a>

1. Edit codes in the urls.py under carts folder

path('remove\_cart\_item/<int:product\_id>/<int:cart\_item\_id>/', views.remove\_cart\_item, name='remove\_cart\_item'),

1. Edit views.py in the carts folder

def remove\_cart\_item(request, product\_id, cart\_item\_id):

cart = Cart.objects.get(cart\_id=\_cart\_id(request))

product = get\_object\_or\_404(Product, id=product\_id)

cart\_item = CartItem.objects.get(product=product, cart=cart, id=cart\_item\_id)

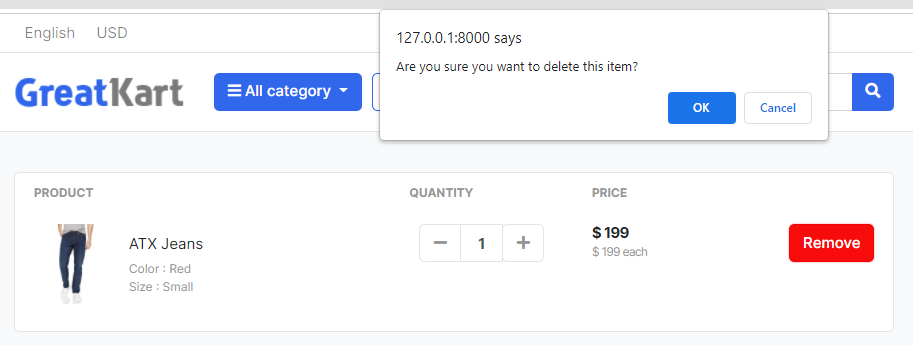
cart\_item.delete()

return redirect('cart')

user able to remove item in the cart page

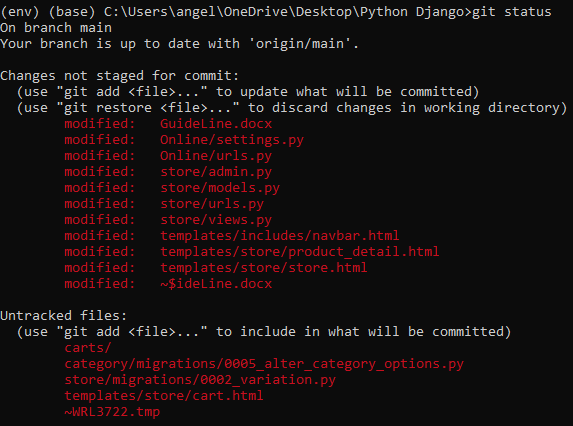
1. Edit code in the cart.html in the store folder under the templates folder

<a href="{% url 'remove\_cart\_item' cart\_item.product.id cart\_item.id %}" onclick="return confirm('Are you sure you want to delete this item?')" class="btn btn-danger"> Remove</a>



1. Push Code to GitHub on CMD

git status



git add –A

git push origin main

1. 1
2. 1
3. 1
4. 1
5. 1
6. 1
7. 1
8. 1
9. 1
10. 1
11. 1
12. 1
13. 1
14. 1
15. 1
16. 1