```
#manage.py
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ECom.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if _name_ == '_main_':
    main()

#admin.py
# Register your models here.
from django.contrib import admin

from .models import *

admin.site.register(Customer)
admin.site.register(Product)
admin.site.register(Order)
admin.site.register(OrderItem)
admin.site.register(ShippingAddress)


#apps.py

from django.apps import AppConfig
class StoreConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'store'

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

```
#  models.py
# Create your models here.

class Customer(models.Model):
	user = models.OneToOneField(User, null=True, blank=True, on_delete=models.CASCADE)
	name = models.CharField(max_length=200, null=True)
	email = models.CharField(max_length=200)

	def _str_(self):
		return self.name


class Product(models.Model):
	name = models.CharField(max_length=200)
	price = models.FloatField()
	digital = models.BooleanField(default=False,null=True, blank=True)
	image = models.ImageField(null=True, blank=True)

	def _str_(self):
		return self.name

@property
def imageURL(self):
		try:
			url = self.image.url
		except:
			url = ''
		return url


class Order(models.Model):
	customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True, blank=True)
	date_ordered = models.DateTimeField(auto_now_add=True)
	complete = models.BooleanField(default=False)
	transaction_id = models.CharField(max_length=100, null=True)

	def _str_(self):
		return str(self.id)

@property
def get_cart_total(self):
	orderitems=self.orderitem_set.all()
	total=sum([item.get_total for item in orderitems])
	return total

@property
```

```
def get_cart_items(self):
        orderitems=self.orderitem_set.all()
        total=sum([item.quantity for item in orderitems])
        return total



class OrderItem(models.Model):
        product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)
        order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
        quantity = models.IntegerField(default=0, null=True, blank=True)
        date_added = models.DateTimeField(auto_now_add=True)

@property
def get_total(self):
        total=self.product.price * self.quantity
        return total


class ShippingAddress(models.Model):
        customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True)
        order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
        address = models.CharField(max_length=200, null=False)
        city = models.CharField(max_length=200, null=False)
        state = models.CharField(max_length=200, null=False)
        zipcode = models.CharField(max_length=200, null=False)
        date_added = models.DateTimeField(auto_now_add=True)

        def _str_(self):
                return self.address


#tests.py
from django.test import TestCase

# Create your tests here.

#urls.py
from django.urls import path
from .import views

urlpatterns = [
    path('home/',views.home,name="home"),
    path('store/',views.store,name="store"),
    path('cart/',views.cart,name='cart'),
    path('checkout/',views.checkout,name='checkout'),
    path('signup/', views.signup, name='signup'),
    path('login/', views.login_view, name='login')
```

```
    path('update_item',views.updateItem,name='update_item')

]
```

[9:27 am, 23/11/2023] +91 99237 71062: from django.db import models
from django.contrib.auth.models import User

```
# Create your models here.

class Customer(models.Model):
        user = models.OneToOneField(User, null=True, blank=True,
on_delete=models.CASCADE)
        name = models.CharField(max_length=200, null=True)
        email = models.CharField(max_length=200)

        def _str_(self):
                return self.name


class Product(models.Model):
        name = models.CharField(max_length=200)
        price = models.FloatField()
        digital = models.BooleanField(default=False,null=True, blank=True)
        image = models.ImageField(null=True, blank=True)

        def _str_(self):
                return self.name

@property
def imageURL(self):
                try:
                        url = self.image.url
                except:
                        url = ''
                return url


class Order(models.Model):
        customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True,
blank=True)
        date_ordered = models.DateTimeField(auto_now_add=True)
        complete = models.BooleanField(default=False)
        transaction_id = models.CharField(max_length=100, null=True)

        def _str_(self):
                return str(self.id)
```

```
@property
def get_cart_total(self):
        orderitems=self.orderitem_set.all()
        total=sum([item.get_total for item in orderitems])
        return total

@property
def get_cart_items(self):
        orderitems=self.orderitem_set.all()
        total=sum([item.quantity for item in orderitems])
        return total



class OrderItem(models.Model):
        product = models.ForeignKey(Product, on_delete=models.SET_NULL, null=True)
        order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
        quantity = models.IntegerField(default=0, null=True, blank=True)
        date_added = models.DateTimeField(auto_now_add=True)

@property
def get_total(self):
        total=self.product.price * self.quantity
        return total


class ShippingAddress(models.Model):
        customer = models.ForeignKey(Customer, on_delete=models.SET_NULL, null=True)
        order = models.ForeignKey(Order, on_delete=models.SET_NULL, null=True)
        address = models.CharField(max_length=200, null=False)
        city = models.CharField(max_length=200, null=False)
        state = models.CharField(max_length=200, null=False)
        zipcode = models.CharField(max_length=200, null=False)
        date_added = models.DateTimeField(auto_now_add=True)

        def _str_(self):
                return self.address


   models.py
[9:27 am, 23/11/2023] +91 99237 71062: from django.test import TestCase

# Create your tests here.
tests.py
[9:27 am, 23/11/2023] +91 99237 71062: urls.py  from django.urls import path
from .import views
```

```
urlpatterns = [
    path('home/',views.home,name="home"),
    path('store/',views.store,name="store"),
    path('cart/',views.cart,name='cart'),
    path('checkout/',views.checkout,name='checkout'),
    path('signup/', views.signup, name='signup'),
    path('login/', views.login_view, name='login'),
    path('update_item',views.updateItem,name='update_item')

]



#views.py
from django.shortcuts import render,redirect
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth import login
from django.http import JsonResponse
import json
from django.contrib.auth.forms import AuthenticationForm
from .models import *

# Create your views here.

def home(request):
    return render(request,'home.html')

def store(request):
    context={}
    customer=request.user.is_authenticated
    products=Product.objects.all()
    context={'products':products}
    return render(request,'store.html',context)









def cart(request):
    if request.user.is_authenticated:
        customer=request.user.customer
        order,created=Order.objects.get_or_create(customer=customer,complete=False)
        items=order.orderitem_set.all()

    else:
```

```
        items=[]
        order={'get_cart_total':0,'get_cart_items':0}


    context={'items':items,'order':order}
    return render(request,'cart.html',context)

def checkout(request):
    if request.user.is_authenticated:
        customer=request.user.customer
        order,created=Order.objects.get_or_create(customer=customer,complete=False)
        items=order.orderitem_set.all()
    else:
        items=[]
        order={'get_cart_total':0,'get_cart_items':0}

    context={'items':items,'order':order}


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



def signup(request):
    if request.method == 'POST':
        form = UserCreationForm(request.POST)
        if form.is_valid():
            user = form.save()
            login(request, user)
            return redirect('store')  # Redirect to the store page after signup
    else:
        form = UserCreationForm()
    return render(request, 'signup.html', {'form': form})


def login_view(request):
    if request.method == 'POST':
        form = AuthenticationForm(data=request.POST)
        if form.is_valid():
            user = form.get_user()
            login(request, user)
            return redirect('store')  # Redirect to the dashboard page after login
    else:
        form = AuthenticationForm()
    return render(request, 'login.html', {'form': form})

def updateItem(request):
    data=json.loads(request.data)
```

```
    productId=data['productId']
    action=data['action']


    print('action:',action)
    print('productID:',productId)

    customer=request.user.customer
    product=Product.objects.get(id=productId)
    order,created=Order.objects.get_or_create(customer=customer,complete=False)

    orderItem,created=OrderItem.objects.get_or_create(order=order,product=product)

    if action=='add':
        orderItem.quantity=(orderItem.quantity +1)

    elif action=='remove':
        orderItem.quantity=(orderItem.quantity -1)

    orderItem.save()

    if orderItem.quantity<=0:
        orderItem.delete()
```

cart.js

```
var updateBtns = document.getElementsByClassName('update-cart')

for (i = 0; i < updateBtns.length; i++) {
        updateBtns[i].addEventListener('click', function(){
                var productId = this.dataset.product
                var action = this.dataset.action
                console.log('productId:', productId, 'action:', action)

        console.log('USER:',user)

        if (user == 'AnonymousUser'){
                        console.log('not logged in')
                }else{
                        updateUserOrder(productId,action)
                }
    })
}

function updateUserOrder(productId, action){
        console.log('User is authenticated, sending data...')

    var url = '/update_item/'
```

```
    fetch(url, {
        method:'POST',
        headers:{
            'Content-Type':'application/json',
            'X-CSRFToken':csrftoken,
        },
        body:JSON.stringify({'productId':productId, 'action':action})
    })

    .then((response) => {
        return response.json();
     })
     .then((data) => {
         location.reload()
     });

}
```