Creating an e-commerce mini-project with Django is a great way to understand the fundamentals of web development, database handling, and user management. Here’s an outline of what you can build, including features and steps to set up the project:

**Features to Implement:**

1. **User Authentication**: Registration, login, logout, and password reset.
2. **Product Management**: Create, Read, Update, and Delete (CRUD) operations for products.
3. **Product Categories**: Organize products into different categories.
4. **Shopping Cart**: Allow users to add items to the cart, update quantities, and remove items.
5. **Checkout System**: Process orders, manage order summaries, and store orders in the database.
6. **Payment Integration**: Simulate payment processing (Stripe, PayPal, or a simple dummy payment method).
7. **Admin Dashboard**: Manage products, view orders, and track sales.

**Project Setup**

Here’s a step-by-step guide to set up the project:

1. **Set Up Django Project**:

bash

Copy code

django-admin startproject ecommerce

cd ecommerce

django-admin startapp shop

1. **Configure settings.py**:
   * Add shop to INSTALLED\_APPS.
   * Set up the database (SQLite for simplicity):

python

Copy code

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.sqlite3',

'NAME': BASE\_DIR / 'db.sqlite3',

}

}

* + Set up MEDIA and STATIC files:

python

Copy code

MEDIA\_URL = '/media/'

MEDIA\_ROOT = BASE\_DIR / 'media'

STATIC\_URL = '/static/'

STATICFILES\_DIRS = [BASE\_DIR / 'static']

1. **Create Models**: In shop/models.py, create models for Product, Category, and Order.

python

Copy code

from django.db import models

from django.contrib.auth.models import User

class Category(models.Model):

name = models.CharField(max\_length=200)

slug = models.SlugField(unique=True)

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

return self.name

class Product(models.Model):

name = models.CharField(max\_length=200)

slug = models.SlugField(unique=True)

description = models.TextField()

price = models.DecimalField(max\_digits=10, decimal\_places=2)

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

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

available = models.BooleanField(default=True)

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

return self.name

class Order(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

products = models.ManyToManyField(Product)

total\_price = models.DecimalField(max\_digits=10, decimal\_places=2)

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

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

return f'Order {self.id} by {self.user}'

1. **Migrate and Create Superuser**:

bash

Copy code

python manage.py makemigrations

python manage.py migrate

python manage.py createsuperuser

1. **Create Views for Product Listing and Details**: In shop/views.py:

python

Copy code

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

from .models import Product, Category

def product\_list(request):

products = Product.objects.filter(available=True)

return render(request, 'shop/product\_list.html', {'products': products})

def product\_detail(request, slug):

product = get\_object\_or\_404(Product, slug=slug)

return render(request, 'shop/product\_detail.html', {'product': product})

1. **Set Up URLs**: In shop/urls.py:

python

Copy code

from django.urls import path

from . import views

urlpatterns = [

path('', views.product\_list, name='product\_list'),

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

]

Include these URLs in the main ecommerce/urls.py:

python

Copy code

from django.urls import path, include

from django.conf import settings

from django.conf.urls.static import static

urlpatterns = [

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

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

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

1. **Create Templates**:
   * Create product\_list.html to display a list of products.
   * Create product\_detail.html for product details.
   * Add a base template to structure the layout.
2. **Shopping Cart Functionality**: Create a Cart class using Django sessions to manage the cart, allowing users to add, update, and remove products.
3. **Order Placement**: Create a form for users to provide details for order placement and store the order in the database.
4. **Admin Dashboard**: Use Django's admin panel to manage products, categories, and orders. Customize the admin panel if needed.

This will provide a simple e-commerce platform with basic functionalities. As you get more comfortable, you can add features like search, product reviews, and more detailed user profiles. Would you like further details on any part of this setup?

4o

**You said:**

7,8,9,10 point full coding and html templetes coding

**ChatGPT said:**

ChatGPT

I'll provide you with code snippets for points 7, 8, 9, and 10, including the necessary Django views, models, and templates for those parts. Let's dive into each section:

**7. Shopping Cart Functionality**

For the cart, we’ll create a class to manage cart items using Django sessions. We'll also include the views and templates for adding to the cart, viewing it, updating quantities, and removing items.

**Models**

We can use Django sessions for storing cart items, so there's no need for an additional database model.

**Cart Logic (shop/cart.py)**

Create a file named cart.py in your shop app to handle cart operations.

python

Copy code

from decimal import Decimal

from django.conf import settings

from .models import Product

class Cart:

def \_\_init\_\_(self, request):

self.session = request.session

cart = self.session.get(settings.CART\_SESSION\_ID)

if not cart:

cart = self.session[settings.CART\_SESSION\_ID] = {}

self.cart = cart

def add(self, product, quantity=1, update\_quantity=False):

product\_id = str(product.id)

if product\_id not in self.cart:

self.cart[product\_id] = {'quantity': 0, 'price': str(product.price)}

if update\_quantity:

self.cart[product\_id]['quantity'] = quantity

else:

self.cart[product\_id]['quantity'] += quantity

self.save()

def save(self):

self.session.modified = True

def remove(self, product):

product\_id = str(product.id)

if product\_id in self.cart:

del self.cart[product\_id]

self.save()

def \_\_iter\_\_(self):

product\_ids = self.cart.keys()

products = Product.objects.filter(id\_\_in=product\_ids)

for product in products:

self.cart[str(product.id)]['product'] = product

for item in self.cart.values():

item['price'] = Decimal(item['price'])

item['total\_price'] = item['price'] \* item['quantity']

yield item

def \_\_len\_\_(self):

return sum(item['quantity'] for item in self.cart.values())

def get\_total\_price(self):

return sum(Decimal(item['price']) \* item['quantity'] for item in self.cart.values())

def clear(self):

del self.session[settings.CART\_SESSION\_ID]

self.save()

**Cart Views (shop/views.py)**

python

Copy code

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

from .models import Product

from .cart import Cart

def cart\_add(request, product\_id):

cart = Cart(request)

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

cart.add(product=product)

return redirect('cart\_detail')

def cart\_remove(request, product\_id):

cart = Cart(request)

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

cart.remove(product)

return redirect('cart\_detail')

def cart\_detail(request):

cart = Cart(request)

return render(request, 'shop/cart\_detail.html', {'cart': cart})

**Cart URLs (shop/urls.py)**

python

Copy code

from django.urls import path

from . import views

urlpatterns = [

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

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

path('cart/', views.cart\_detail, name='cart\_detail'),

]

**Cart Template (templates/shop/cart\_detail.html)**

html

Copy code

{% extends 'base.html' %}

{% block content %}

<h2>Your Cart</h2>

<table>

<thead>

<tr>

<th>Product</th>

<th>Quantity</th>

<th>Price</th>

<th>Total</th>

<th>Action</th>

</tr>

</thead>

<tbody>

{% for item in cart %}

<tr>

<td>{{ item.product.name }}</td>

<td>{{ item.quantity }}</td>

<td>${{ item.price }}</td>

<td>${{ item.total\_price }}</td>

<td>

<a href="{% url 'cart\_remove' item.product.id %}">Remove</a>

</td>

</tr>

{% endfor %}

</tbody>

</table>

<p>Total: ${{ cart.get\_total\_price }}</p>

<a href="{% url 'order\_create' %}">Proceed to Checkout</a>

{% endblock %}

**8. Order Placement**

We'll create a form for users to fill in their details and place an order.

**Models (shop/models.py)**

Update the Order model to store user information:

python

Copy code

from django.db import models

from django.contrib.auth.models import User

class Order(models.Model):

user = models.ForeignKey(User, on\_delete=models.CASCADE)

address = models.CharField(max\_length=250)

city = models.CharField(max\_length=100)

postal\_code = models.CharField(max\_length=20)

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

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

return f'Order {self.id} by {self.user}'

**Forms (shop/forms.py)**

Create a form for the order:

python

Copy code

from django import forms

from .models import Order

class OrderCreateForm(forms.ModelForm):

class Meta:

model = Order

fields = ['address', 'city', 'postal\_code']

**Order Views (shop/views.py)**

python

Copy code

from django.shortcuts import render, redirect

from .forms import OrderCreateForm

from .models import Order

from .cart import Cart

def order\_create(request):

cart = Cart(request)

if request.method == 'POST':

form = OrderCreateForm(request.POST)

if form.is\_valid():

order = form.save(commit=False)

order.user = request.user

order.save()

for item in cart:

order.products.add(item['product'])

cart.clear()

return redirect('order\_success')

else:

form = OrderCreateForm()

return render(request, 'shop/order\_create.html', {'form': form, 'cart': cart})

def order\_success(request):

return render(request, 'shop/order\_success.html')

**Order URLs (shop/urls.py)**

python

Copy code

urlpatterns += [

path('order/create/', views.order\_create, name='order\_create'),

path('order/success/', views.order\_success, name='order\_success'),

]

**Order Templates (templates/shop/order\_create.html)**

html

Copy code

{% extends 'base.html' %}

{% block content %}

<h2>Checkout</h2>

<form method="post">

{% csrf\_token %}

{{ form.as\_p }}

<button type="submit">Place Order</button>

</form>

{% endblock %}

**Success Template (templates/shop/order\_success.html)**

html

Copy code

{% extends 'base.html' %}

{% block content %}

<h2>Thank you for your order!</h2>

<p>Your order has been placed successfully.</p>

<a href="{% url 'product\_list' %}">Continue Shopping</a>

{% endblock %}

**9. Admin Dashboard**

Use Django's built-in admin panel for managing products, categories, and orders. You can customize it if needed.

**Register Models (shop/admin.py)**

python

Copy code

from django.contrib import admin

from .models import Product, Category, Order

@admin.register(Product)

class ProductAdmin(admin.ModelAdmin):

list\_display = ['name', 'price', 'available']

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

@admin.register(Category)

class CategoryAdmin(admin.ModelAdmin):

list\_display = ['name']

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

@admin.register(Order)

class OrderAdmin(admin.ModelAdmin):

list\_display = ['user', 'created\_at']

Now you can log in to Django's admin interface at http://127.0.0.1:8000/admin/ to manage your products and orders.

**10. Admin Customization**

You can further customize the admin interface by adding inline editing, filters, and search fields. Here’s an example of adding search capabilities:

python

Copy code

@admin.register(Order)

class OrderAdmin(admin.ModelAdmin):

list\_display = ['user', 'created\_at', 'address', 'city', 'postal\_code']

search\_fields = ['user\_\_username', 'address', 'city', 'postal\_code']

list\_filter = ['created\_at']

With these additions, your e-commerce mini-project should now have a fully functional shopping cart, order placement, and admin management. Let me know if you need more details on any specific part!

4o