Django Online Shop Main Steps

**Main Modules**

* Product Store – Which will display all products
* Cart – Which will display added products to cart
* Checkout – Checkout for added products i.e adding address and process order

1. Start Project – onlineshop
2. Start App – eshop
3. Add app name in settings.py (installed apps)

**Part-1**

**Setup Templates with Boostrap Styling**

1. Creating templates folder in project app directory

* base.html – main base html which all templates inherits from this base template
* home.html - home page for displaying all products
* cart.html – user selected product displaying cart
* checkout.html – user checkout page

1. Now configure temlates directories in settings ( in TEMPLATES -> DIRS: ) as 'DIRS': [os.path.join(BASE\_DIR, 'templates')],
2. And import os in settings with command import os
3. Now create 3 views for home, cart and checkout html pages with rendering respective html pages
4. Go to urls.py file in onlineshop project directory include eshop urls path with empty directory such as path('', include('eshop.urls')),
5. Create urls.py in eshop app directory and add configure 3 urls paths as below

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

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

path('checkout/', views.checkout, name='check'),

1. Add some stuff in each html file and make it sure all urls are working as home.html 🡪<h1>Product Page</h1> and run server in local host and check all three urls
2. Now create static file folder in project directory for styling
3. Now configure static directories in settings (add at last line) as STATICFILES\_DIRS =[os.path.join(BASE\_DIR, 'static')]
4. Create sub folder called images and css in static folder for displaying images as products
5. Create main.css file and copy code provided in Part-1 Source Code files
6. And add add css file link in base.html file under title as
7. Now go to link <https://getbootstrap.com/docs/4.5/getting-started/introduction/> to make your templates stylish
8. Now go to base.html file to edit and make it common for all templates
9. Go to base html file and add type html and press tab button to setup html tags in base.html
10. Add static tag in base.html under <!DOCTYPE html> as {% load static %} to load/link files under static file folder
11. Add title that what you want and under title add CSS files link from Boostrap website link as provided above
12. And add Java files link from same Boostrap website under body in base.html file
13. Add navbar(choose your own nav bar ) html code from same website (comonents->navbars) and paste it under body in base.html file (before java link files)
14. And edit nav bar code in base.html as your wish such as nav bar title, home button, cart button
15. And change urls for home and cart in base.html such as href="{% url 'home' %}">Home</a>
16. Now add cart image and its related css file code (static->css folder->main.css as mentioned in source files provided by us
17. After finishing the nav bar set up we can call it for base html file for all templates and now add tags {% block content %} {% endblock %} to process the other templates
18. Now go to home.html and load base.html file as {% extends 'base.html' %}
19. Add {% block content %} write your home product html code under these tags {% endblock %}
20. Do 22, 23, 24 for remaining templates
21. Go to home.html and add 3 product cards (take card design from boostrap wev site as mentioned above) (see Part-1 Source Code for your reference) and add any images that you want sell in your website
22. And do as for cart and checkout too (see Part-1 Source Code)

**Part-2**

**Create Models and Rendering Data into Created Templates**

Here we are going to create 5 different kinds of models

* User Customer model for the customer authenticated by user (in this case self)
* Product model for kind of product, price and its details
* Order model for no of orders for that particular user/customer
* Order Item model for items in that particular order
* ShippingAddress model for shipping info for that particular customer and his order

1. Go to models and create user model, for this we need to import user form django auth such as from django.contrib.auth.models import User
2. Create Customer Model with one to one filed for the user because this user have one customer (he is nothing but user) and other required fields (see Part-2 Source Code)
3. And create product model with product name, image and price
4. For this we need to install module called pillow which is image processing library for the Django image filed in models
5. For this use command pip install pillow
6. For the image filed you need to root the media and image url And you can add image from anywhere and you must root (in settings such as MEDIA\_ROOT = os.path.join(BASE\_DIR, 'static/images')) the media folder which will save all images that you uploaded
7. For rendering images into templates, you need to set image URL just like static URL (in settings set MEDIA\_URL = '/images/')
8. And now set the path in project url file like urlpatterns += static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)
9. Before that import these into urls sile: from django.conf.urls.static import static and from django.conf import settings
10. Now image media root and url configuration is finished, now we ned to make a function under product models for image field, why because if you are rendering image into template and if there is no image in model table, it will show Django create error page, to overcome this and put empty image when there is no image, we need to make a function in product model as below,

@property

def imageURL(self):

try:

url = self.image.url

except:

url = ''

return url

1. Call this function instead of url in templates, it will return image if there otherwise it will show empty image(no image in that particular field/row)
2. And create remaining models (see Part-2 Source Code)
3. Now register all models to admin sites in application directory and makemigrations and migrate and then run server
4. Open admin panel see all your register models there
5. Now add some stuff into your models such as customer details, products, order detailsand order items
6. Delete images in product images folder and put in somewhere and upload (rom product admin panel) from somewhere locations and then you can see images in images folder
7. Now we need to retrieve data from tables to templates using functioning views
8. Now go to views and then product view to render products from model tables
9. For this, get all objects from product model and put it in context dictionary and call required parameter in template
10. Except one column delete all columns in home template (under rows class) and call all products using for loop
11. And call product details with template tags using for loop from vies context dictionary (see Part-2 Source Code)
12. Now you can restart server and you can see product data
13. Now we need to go to the cart view to render the cart template
14. For this, first we need to check whether is authenticated or not, in our case we login to admin through admin panel, so we are authorised
15. Now get customer related to user
16. And get filter order details with customer and status of order completion
17. And now collect/set all items from related customer and put this in dictionary
18. Call cart details from this dictionary into cart template same as home template
19. Now you can see the cart the cart items in cart view but not total items and total cost for the particular item and total cost for all the items
20. For this we need to define function under order details for calculate total items for that particular order and then define another function for total price under items list (this total is for the particular item. Then define another function for total all items (for all added products) under order details (see Part-2 Source Code)
21. Now call the required items total, total items price and total cart price into the cart template from the views (see Part-2 Source Code)
22. Set empty values in cart view if user is not authenticated (total cart items and price become 0)
23. Logout admin and verify cart, now it shows empty cart
24. And do same as for checkout view (same as cart view) and verify checkout template view