NB:The migrations before if not a problem may occure during modification of User model

# GUIDELINES

## Project startup

* Create a project folder and a virtual environment in the project folder : virtualenv env

Then activate the virtual environment : env\Scripts\activate

* In your project folder, install Django(in case you do not have it yet) : pip install Django  
  Then create a Django project : django-admin startproject e\_commerceprj
* In the Django-project create your Django app : django-admin startapp core
* Install the app in the **settings.py**
* Configure the template (static and media files)
* Import os in the settings
* Join the base directory to the template folder and create the template folder
* Join the static root: STATIC\_ROOT = os.path.join(BASE\_DIR, ‘static’)
* Add the static route and the static\_root to the project urls : urlpatterns += static(settings.STATIC\_URL, document\_root = settings.STATIC\_ROOT)
* Join the static files directories : STATICFILES\_DIR = os.path.join(BASE\_DIR, ‘staticfiles’)
* Configure the media urls: MEDIA\_URL = ‘/media’
* Join the media root: MEDIA\_ROOT = os.path.join(BASE\_DIR, ‘media)
* Configure new views, urls and templates then run server
* Configure template inheritance and partials

## Configure Admin Page, Superuser and Jazzmin

* Install Jazzmin : pip install Django-jazzmin
* Add jazzmin in INSTALLED\_APPS
* Add jazmin Cingig Code in **settings.py**
* Create Superuser
* Login to admin Sector

## Custom User Model

* Create new app userauths
* Install the app in the **settings.py**
* Create a custom class User(AbstractUser): in **models.py**
* Add AUTH\_USER\_MODEL = ‘userauths.User’ in **settings.py**
* Run Makemigrations and migrate
* Create new superuser and register the model in **admin.py**
* Login to the admin with email and password

## User Register System

* Create new form class UserRegisterForm(UserCreationForm): in **forms.py**
* Write a class meta that specifies the model and the fields to be shown on the form
* Write the view to register users: def RegisterView(request):
* Import the form in the view
* Create a variable context with the form as value
* Then pass the context variable to the render function as 3rd parameter
* Configure template to show the form
* Login to the website from the frontend

## User Login System

* Write the view to login users: def LoginView(request):
* Configure template to grab input fields
* Login to the website from the frontend

## User Logout System

* Write the view to login users: def LogoutView(request):
* Configure URL
* Test the feature

## Alerts In Django

* Grab alert snippet from Bootstrap(version4)
* Copy and paste CDN
* Write alert conditional statements

## Product Model structure

* Install UUID to assign your own id and not use the default id from django
* Create new model classes and add fields for products

## List View for Products

* Create logic to Display only featured products in homepage
* Create new view to list all the active products from the database
* Configure urls.py and templates

## Category list view

* Create new view to list all the active categories
* Configure urls.py and templates

## Product Category list view

* Create new view to list all the active products from the database depending on the category selected
* Configure urls.py and templates

## Django Context Processor For Template

Context processors help us access particular objects in our template without necessarily passing it through a view. It is used to make the base.html to access data from database since there is no particular view rendering it

* Create a new file context processor.py in the core app
* Install in settings.py TEMPLATES Section List as ‘core.context\_processor.default’
* Now add the code for the context processor

## Vendor List View

* Create new view to list all active vendors
* Configure urls.py and templates

## Vendor List Detail

* Create new view to show details of the active vendor
* Configure urls.py and templates

## Product Detail View

* Create new view to showcase the details of a selected product using **pid**
* Configure urls.py and templates

## Product Rasting and Review

* Get all reviews in Product Details and List them out in the template
* Calculate the rating\_average for a product
* Create new forms for adding reviews
* Write the view to add reviews
* Create ajax function to create review
* Check if a user have made a review before then restrict them from making reviews

## Searching for products

* Create a view to search for products
* Create template
* Configure URL.py

## Filtering Products using Ajax Jquery

* Create view to filter products
* Configure urls.py
* Create another dir **async** and new file **product-lists.html** in the new folder
* Iterate over the brands and categories in the product-lists-filter section in templates
* Start writing function in filter.js