**Template Django Project**

Setting Up a Django Project for a Website Clone

**1. Download Templates**

Begin by downloading the templates for the website you want to clone.

**2. Create a New Folder**

Create a new folder for your project and give it a suitable name.

**3. Open in Sublime Text and CMD Prompt**

Open the project folder in Sublime Text and also open a Command Prompt window.

**4. Create a Virtual Environment**

Run the following commands in the Command Prompt to create and activate a virtual environment:

```bash

python -m venv myvenv

myvenv\Scripts\activate

```

**5. Install Django**

Install Django in the virtual environment using pip:

```bash

pip install Django

```

6. Start a Django Project

Use the following commands to start a Django project and app:

```bash

django-admin startproject mysite

django-admin startapp myapp

```

7. Migrate Database

Apply initial database migrations:

```bash

python manage.py migrate

```

8. Create Superuser

Create a superuser for the Django admin panel:

```bash

python manage.py createsuperuser

```

Provide a username and password when prompted:

```plaintext

username = akshay8281

password = Testing@123

```

9. Configure Settings

Open the `settings.py` file in the `mysite` directory and update the `INSTALLED\_APPS` list to include your app:

```python

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'myapp', Add your app here

]

```

Also, set the time zone:

```python

TIME\_ZONE = 'Asia/Kolkata'

```

10. URL Configuration

In the `urls.py` file in the `mysite` directory, include your app's URLs:

```python

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

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

path('', include('myapp.urls')), Add your app URLs

]

```

11. Create App URLs

Create a `urls.py` file in your app directory (`myapp`) and define the URL patterns:

```python

from django.urls import path

from . import views

urlpatterns = [

path('', views.index, name='index'), Define your URL patterns

]

```

12. Create Views

Define the view function for the index page in the `views.py` file:

```python

from django.shortcuts import render

def index(request):

return render(request, 'index.html') Render the index.html template

```

13. Static and Templates

Create `templates` and `static` folders in your app directory. Copy the downloaded templates to the `templates` folder and the static files (CSS, images, JS) to the `static` folder.

14. Update HTML Files

Update the HTML files to load static files using Django template tags:

```html

{% load static %}

<link href="{% static 'css/style.css' %}" rel="stylesheet" />

```

15. Create Base Template

Create a base template (`header.html`) containing common elements like header and footer. Use Jinja templating:

```html

{% block content %}

{% endblock %}

```

16. Extend Base Template

In your `index.html` file, extend the base template and define the block content:

```html

{% extends 'header.html' %}

{% block content %}

<!-- Your index.html content here -->

{% endblock %}

```

---

These steps will guide you through setting up a Django project for your website clone.

***23-05-2024***

**Connect the Pages**

* Connect About  
  **set url in curly braces {% url ‘ ’ %}**
* Give all the path in **urls.py**
* After that make function **views.py** for all the **header links**
* Go to about.html import the **header, static and block content  
  {% extends 'header.html' %}**

**{% load static %}**

**{% block content %}**

* **Remove** the **head ,header and footer** from the **about.html**
* Set the **endblock** of block content at **last of the about.html**
* **Same process to all the html files contact, product, testimonial, blog\_list and index**
* **Remove unwanted links from header.html** like blog and **add another link tag like sign up and login**
* **Create path in urls.py in myapp for signup and login**
* Create **function** in **views.py** for **signup and login**
* Create **html file in templates for sign up and login**
* **Copy** data of **contact.html in signup.html**
* Rename and **set form in url {% url ‘signup’ %}**
* **Add email.mobile,address, password and cpassword to sign up**
* **Add csrf\_token below the form in signup.html  
  {% csrf\_token %}**
* **Set msg syntax above the form of signup.html**
* **After that Add content of SignUp to the Login.html and make changes accordingly**
* Create class for the (User) login.html in models.py
* After that run command  
  **python manage.py makemigrations  
  python manage.py migrate  
  python manage.py runserver**
* After migrate the class then go to **admin.py** add **register** the same **(User) and import the class**.
* Go to **views.py** and **import** the **class** from model.py
* Add **signup logic** in the function of signup and also add usertype in **models.py** for the **class** in **User**  
  **Give command when any changes in CLASS**  
  **python manage.py makemigrations  
  python manage.py migrate  
  Add if else** in sign function  
  **run the server** and check sign up data shows in admin Panel or not?

**Login**

* Check the password of user is match with register by user during signup or in database (admin Panel)
* **Create Session of email and fname**
* Give **conditions** using **try and Except and If Else**
* Check the **Login** is **work or not**
* In **header.html** after login **shows** **logout** button by using Jinjalo method if else

***Logout***

* Create **url of logout in urls.py** and also create **function** for **logout** in **views.py**
* **Delete** the **session** of **email and fname** in **logout** function in **views.py**
* Give **if else** conditions
* Give name of user who sign in logout by using **request.session.fname**

**Profile Picture**

* First to type command below in models.py in the class of User **profile\_picture = models.ImageField(upload\_to = '"profile\_picture/',default = "")**
* Search on google for import Profile Image and paste it in settings.py in the last **MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')**

**MEDIA\_URL = '/media/'**

**Import os in settings.py**

* Add below syntax in **mysite > urls.py  
  from django.contrib import admin**

**from django.urls import path,include**

**from django.conf import settings**

**from django.conf.urls.static import static**

**urlpatterns = [**

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

**path('',include('myapp.urls')),**

**]**

**if settings.DEBUG:**

**urlpatterns += static(settings.MEDIA\_URL,**

**document\_root=settings.MEDIA\_ROOT)**

* Upload the photo from Admin Panel
* Add **Upload input in Signup** and also add below **syntax in form submission  
  enctype="multipart-form-data"**
* Go to **views.py** in function **signup** add below syntax  
  **profile\_picture = request.FILES['profile\_picture'].**
* Go to function **login** add below syntax  
  **request.session['profile\_picture'] = user.profile\_picture.url**
* When **logout** by user **delete the session**  
  **del request.session['profile\_picture']**
* Go to **header.html** add below syntax for the session when logged in and after the session break. Add this below the **logout** button.  
    
  **<li class="nav-item">**

**<a class="nav-link" style="color: darkred;" href="{% url profile %}"><img src="{{request.session.profile\_picture}}" style="width: 50px;height: 50px;"></a>**

**</li>**

* New **url is profile** so it is assign to the **url.py** and then **create** **function** for the **profile in views.py**
* In **profile** **copy** the **data of signup.html** in the **profile.html**
* Replace the required into value and put data  
  **value="{{user.mobile}}"**  
  do it for all
* Also show image using below syntax  
  **<img src="{{user.profile\_picture.url}}" style="width: 100px;height: 100px;align-content: center;border-radius: 50%;">**
* After that we have set is it Buyer or Seller. Below are the syntax to know Seller or Buyer  
    
  **<select name="usertype">**

**{% if user.usertype == "Buyer" %}**

**<option>---Select User Type---</option>**

**<option value="Buyer" selected>Buyer</option>**

**<option value="Seller">Seller</option>**

**{% else %}**

**<option>---Select User Type---</option>**

**<option value="Buyer">Buyer</option>**

**<option value="Seller" selected>Seller</option>**

**{% endif %}**

**</select>**

* Go to **profile.html** in email input set attribute **READONLY**, so no one can change there email during profile update

***28-05-2024***

* Rename the Pages In header section into My Account and in that rename to Cart,Change password and Wishlist.
* Create url of change Password put path in urls and then make function in views.py
* New url in views.py so put that into first urls.py and then goes into views.py and copy the login.html into change-password.html and change according to the change Password
* Give nested if else conditions for change password

**Forgot Password**

* Open login.html add forgot Password Ancor Tag below the Login Button.
* Create url for forgot-password in below the login button I ancor tag in login.html.
* Create path for the forgot-password in urls.py.
* Create function in views.py for the forgot-password.
* From function create forgot-password.html in templates.
* Copy data of login.html into forgot-password.html
* Changes according to Forgot Password.
* After adding in try and except logic then we have to send OTP in mobile
* For mobile OTP we have to use API name called FAST2SMS.Create our account on that and add Rs 100 money in the wallet and also Update KYC in that account.
* We have Developer Option in that we have API KEY option click on it we have link (identity of OTP Sender)
* Go to Fast2SMS and in that go to Dev API For Gmail, For Payment and also shows identity of person to sent a message
* Go to Fast2SMS > Dev API > in that click on READ API DOCS. In that got to OTP SMS API > GET METHOD, in that go to python and import the library.
* Copy **import requests** and paste this library top of the views.py
* After that copy below syntax into the function of forgot\_password  
  **url = "https://www.fast2sms.com/dev/bulkV2"**

**Querystring={"authorization":"YOUR\_API\_KEY","variables\_values":"5599","route":"otp","numbers":"9999999999,8888888888,7777777777"}**

**headers = {'cache-control': "no-cache"}**

**response = requests.request("GET", url, headers=headers, params=querystring)**

**print(response.text)**

* To generate OTP we have to **import random** library in views.py and make logic according it in forgot\_password function.
* **url =** [**https://www.fast2sms.com/dev/bulkV2**](https://www.fast2sms.com/dev/bulkV2) /// This url does not change bcs OTP will receive from this link
* After that go to querystring and changes in Your API KEY  
  **BEFORE** **Changes** = **querystring={"authorization":"YOUR\_API\_KEY","variables\_values":"5599","route":"otp","numbers":"9999999999,8888888888,7777777777"}  
    
  Go to API KEY and copy that link and paste it in YOUR\_API\_KEY  
  API KEY = juGOTH9Krhm0YgJlvsiBRAxn4P7bIaUzf68yVFCSedNqcwpQW3EjY5ta3lfvhmCsocMd0ObFrGDiLPgZ  
    
  Change “Variables\_values” = otp  
  change “numbers” = mobile  
    
  AFTER Changes =   
  querystring = {"authorization":"juGOTH9Krhm0YgJlvsiBRAxn4P7bIaUzf68yVFCSedNqcwpQW3EjY5ta3lfvhmCsocMd0ObFrGDiLPgZ","variables\_values":otp,"route":"otp","numbers":mobile}**
* After OTP sent then what to do ?
* Match the OTP  
  put session otp and mobile
* Return all the content of forgot\_password into new html file called otp.html and copy the data of forgot\_password.html into otp.html
* In otp.html, we have new url name verify-otp so give path in urls.py and then make function for the same in views.py
* Add some logics in verify\_otp function
* After match OTP delete the session
* New url created in verify-otp function name = new-password.html
* Copy the content of change-password.html into new-password.html and changes accordingly.
* New url generated in new-password.html, so make for the update-password in urls.py and make function for the same in views.py
* After adding logics in the function of new\_password then we have to install library for the requests using CMD  
  **pip install requests**
* After that run the server and check whether it is work or not

Dashboard of Buyer and Seller would be different when login or Change Password

Buyer =

* All Product show
* Search, Wishlist, Cart, Payment, Check order

For Different Dashboard

* Go to login function in views.py and write code as per below  
  if user.usertype == "Buyer":

return render(request,'index.html')

else :

return render(request,'seller-index.html')

* Create Seller-index.html and copy of index.html into seller-index.html
* And also create seller-header.html, in that copy header.html into seller-header.html
* **When in seller account we click on change the password the tab open of buyer header. So open header of seller-header nor Buyer header.**
* Go to views.py > change\_password First time request goes in GET METHOD
* New seller-change-password.html for seller side and copy the data of change\_password.html
* When Seller profile login then click on PROFILE PHOTO the header would be change to BUYER
* Go to Profile function in views.py add if else for Profile of Buyer…. Seller….
* **When seller logged in already after that close tab directly and then after that search localhost:8000 then open directly Buyer Account**
* Our root path is index.html so directly access url mentioned in urls.py. So we have to add logic for the Buyer and Seller. Add if else
* Careful while integrating the Template Website because Buyer Logged in so only data show of Buyer only it is know as DATA LEAKAGE

**04-06-2024 – Add Product**

* For add product we have to store our data into database and to store in database we need table for that so we have to create model for that we have to make field.
* In model of Product we have to add product\_name, product\_image, product\_price, product\_desc and also define which seller it is.

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

**product\_name = models.CharField(max\_length=100)**

**product\_price = models.PositiveIntegerField()**

**product\_image = models.ImageField(upload\_to ="product\_image/")**

**product\_desc = models.TextField()**

* In ForeignKey one compulsory Parameters is **(on\_delete = models.CASCADE)**
* We have to make category for the web page (Men, Women and Kids)
* Also Create Sub Category for the same
* After all logic run command makemigrations and then migrate the same after that run the server.
* Add to admin for register and then go to Views.py and import the Product
* Go to seller-header.html and change the url of product to **add product** in **replace** of **product**
* After that **give** **Path** for that **url** in the **urls.py** and go to **views** **create** **function** in the name of **add\_product** and in that we have **created** new html file called **seller-add-product.html.**
* Copy the data of Signup.html in seller-add-product.html
* Changes according to the requirement and then go to views.py in add\_product function create OBJECTS for the product add by the seller
* Add Product in the all category by run the code
* After that add url of view product in seller-header.html called = view-product and add path for the same in urls.py and also make function in views.py
* In the function of view\_product generate new html file called = seller-view-product.html and copy the product.html into seller-view-product.html
* Delete all the static item from the list of items
* Customize the single product using Jinjako to use loop and show all the products.
* After use of for loop then we want get the the data from our database  
  Product Image = **{{ i.product\_image.url }}**Product Price = **{{i.product\_price}}**Product\_Category and Product Sub Category = **{{i.product\_category}}’s {{i.product\_sub\_category}}**
* Change Buy Button and place Details
* When I click any of one shows the details of that item
* We have to set url in ancore tag in details button using Jinjako  
  **(% url ‘seller-product-details’ pk = i.pk %)  
  /// pk means = PRIMARY KEY**
* Set url path in the urls.py as per below code  
  **path('seller-product-details/<int:pk>/',views.seller\_product\_details,name = 'seller-product-details')**
* Go to views.py add logic in that with pk parameter and new html file have to create seller-product-details.html
* Copy the data of seller-view-product.html into seller-product-details.html and changes according to the requirement.
* Remove the FOR LOOP jinjako
* When click on Details then shows single product
* Change details button to EDIT and DELETE

**06-06-2024 – Edit and Delete**

* New url created in seller-product-details.html for EDIT Button
* Go to urls.py and put the new path of seller-product-edit
* Go to views.py and add the function of seller\_product\_edit copy from seller\_product\_details and change according to seller\_product\_edit
* In the views new html created called seller-product-edit.html and copy the data from seller-product-details.html
* After that data will show when edit on particular product item
* Go to product edit and add jinjako using IF ELIF in the select to store existing data in the form of that particular item
* Add other logic access data change data which is editable

**Delete Button**

* Make url in seller-product-details.html in that make url in DELETE button called = seller-product-delete
* Make path in the urls.py and create function in views.py for seller\_product\_delete and add logic for the delete of a particular item
* Uer redirect method to search the product and also import the library for the redirect in the views.py
* Now we look from Buyers point views so al products have to show to the Buyer by using below code  
  in views.py > def product   
  def product(request) :

products = Product.objects.all()

return render(request,'product.html',{ 'products':products })

**Buyer**

* Copy data of seller-view-product.html into product.html and change the header from seller to normal.
* In product.html change in Details of url in **replace** of **seller-product-details** we have create **product-details**
* Copy the data of **seller\_product\_detials** and paste it below that and rename to **product\_detail**
* Create new html file name called product-details
* And copy the data of seller-product-details.html into product-details.html
* In product-details.html we have to change Edit and Delete button into Add to cart and Add to Wishlist
* This show only when user is logged in so change accordingly using jinjako
* What to do when list of items add into Wishlist
* Write function in models.py  
  class Wishlist(models.Model):

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

**product = models.ForeignKey(Product,on\_delete = models.CASCADE)  
 date = models.DateTimeField(default = timezone.now)  
  
def \_\_str\_\_(self):**

**return self.user.fname + " - " + self.product.product\_name**

* Import library for Time and Date  
  **from django.utils import timezone**
* After successfully class created then we have to run below command  
  **python manage.py makemigrations  
  python manage,py migrate  
  python manage.py runserver**
* After we have to registor into the Admin.py  
  **from .models import User,Product,Wishlist  
  admin.site.register(Wishlist)**
* After that we to add on views.py  
  **from .models import User,Product,Wishlist**
* Go to product-details.html create new url in Wishlist called = add-to-wishlist with pk (Primary Key)
* Go to header.html and remove and add wishlist icon and Cart Icon
* Set url wishlist icon called = wishlist and give path for the same in urls.py
* Go to views.py and create function for the wishlist icon
* Fetch the user using email id and session
* Add logic in wishlist function
* Copy data of product.html in wishlist.html
* Changes according to the Wishlist requirement
* When already in Wishlist a particular product then a shows button Remove from Wishlist in the Details of the product
* Each and every user have own Wishlist each one different Wishlist
* In **views.py > product\_details** we got the user  
  **user = User.objects.get(email = request.session['email'])**
* Write below code in views.py > product\_detials to getting the ‘wishlist\_flag, user and product  
  **def product\_details(request,pk):**

**wishlist\_flag = False**

**user = User.objects.get(email = request.session['email'])**

**product = Product.objects.get(pk = pk)**

**try:**

**Wishlist.objects.get(user = user,product = product)**

**wishlist\_flag = True**

**except:**

**pass**

**return render(request,'product-details.html',{'product':product,'wishlist\_flag':wishlist\_flag})**

* Then go to product-details.html and add jinajo in the if else statement for the Add to Wishlist and Remove From The Wishlist  
  **{% if request.session.email %}**

**<a href="{% url 'seller-product-edit' pk=product.pk %}" class="option2">**

**Add to Cart**

**</a>**

**{% if wishlist\_flag == False %}**

**<a href="{% url 'add-to-wishlist' pk=product.pk %}" class="option2">**

**Add to Wishlist**

**</a>**

**{% else %}**

**<a href="{% url 'add-to-wishlist' pk=product.pk %}" class="option2">**

**Remove From Wishlist**

**</a>**

**{% endif %}**

* After that we have create logic for Remove Wishlist
* New url in product\_details.html > Remove From Wishlist name called = {% url ‘remove-from-wishlist’ %}
* Give path in urls.py by copy of add to wishlist and change it into remove-from-wishlist
* Create function in views.py   
  **def remove\_from\_wishlist(request,pk):**

**product = Product.objects.get(pk = pk)**

**user = User.objects.get(email = request.session['email'])**

**wishlist = Wishlist.objects.get(user = user,product = product)**

**wishlist.delete()**

**return redirect('wishlist')**

* When In Wishlists is Empty shows msg No products in wishlists using jinjako IF ELSE in wishlist.html

**11-06-2024**

* Without Login does not show Wishlist,cart and change password and also does not show Wishlist and Cart Logo before login or signup.So we use Jinjako for IF ELSE for the email.
* Click on product details does not shows functionality of Add to cart or Add to wishlist, so that we have to add before login features in product\_details in the function of product\_details
* Require Counting feature in Wishlist for the user friendly
* Go to views.py > login function > add wishlists = Wishlists.objects.filter….
* Session delete when user logged out then add del wishlist\_count session
* When Counting is change in the wishlist ?  
  The counting change product will add or remove from the wishlists
* Go to wishlist function and then add session of wishlist\_count
* Go to header.html add to the span of session wishlist\_count in the wishlists

**Add To Cart**

* Go to models.py and make new class for CART
* Copy of Wishlist Class into Cart Class and changes according to the requirement
* When create model successfully than we have to give below command  
  python manage.py makemigrations  
  python manage.py migrate  
  python manage.py runserver
* After that we have to registor on admin.py of Cart Class
* After go to into views.py and import that class of Cart
* In views.py in function find wishlist code and copy paste and changes according to the requirement of Cart function
* Go to login copy data of wishlist and changes according to Cart
* Logout > copy data of wishlist and changes according to Cart
* Product\_details > copy data of wishlist and changes according to Cart
* add\_to\_cart > copy data of wishlist and changes according to Cart  
  product = product,

**user = user,**

**product\_price = product.product\_price,**

**product\_qty = product.product\_qty,**

**total\_price = product.total\_price**

* cart > copy data of wishlist and changes according to Cart
* remove\_from\_cart > copy data of wishlist and changes according to Cart
* Go to urls.py and add path for the below function  
  **'add-to-cart/<int:pk>/'  
  'cart/'  
  'remove-from-cart/<int:pk>/'**
* Go to header.html and paste jinjako of wishlist to the Cart and also add url in the cart
* Go to product\_details.html and copy the data of wishlist add and remove for the cart add and remove, so changes in btn of add to cart and remove to cart
* After that create cart.html file in templates folder and copy the data of wishlist.html and paste it into cart.html and changes according to the cart.
* Raise error after run the program so that changes in views.py > product\_details in that change below  
  **product\_price = product.product\_price,**

**product\_qty = 1,**

**total\_price = product.product\_price**

* Go to cart.html and remove the details part
* Add product\_price and product\_qty in the card
* When qty change automatically change the Total Price
* Add logic for the change qty change the price new url generated called = **change-qty**
* Go urls.py and give path for the same with pk
* Go to views.py add function of change\_qty
* After that we have to total the amount of item from the cart list
* So we have to take variable net\_price in the cart function views.py after that go to cart.html and add span tag put the data in jinajko format = **net\_price**

**Payment Integrations**

* First we have to register into stripe portal for payment integration by using its API
* There are two types of Payment Integrations  
  **Developer Mode  
  Testing Mode**Make sure we are using Testing Mode for our Project
* We working on testing mode
* Signup in strip to open an account  
  **user name =** [**aks.pitroda.8@gmail.com**](mailto:aks.pitroda.8@gmail.com) **Password = akshay@8281**
* Open cmd command prompt and install STRIPE  
  **pip install strip**
* Search on google Payment integration using stripe and then go to below link  
  <https://testdriven.io/blog/django-stripe-tutorial/>
* Open the above site in chrome find STIPE PUBLISHABLE KEY for setting.py and paste it into last of the setting.py  
  **STRIPE\_PUBLISHABLE\_KEY = '<your test publishable key here>'**/// open stripe API kEY > copy the publishable key and paste it in the settings.py

**/// Stripe does not open so we have to use JIGAR SIR API link for our Project**

**STRIPE\_SECRET\_KEY = '<your test secret key here>'**/// open Stripe API Key > now our secret key is reveal so copy that and paste it in the secret key bottom of the settings.py in secret key tab

**/// Stripe does not open so we have to use JIGAR SIR API link for our Project**

* After that we have create out profile In STRIPE by using below link  
  <https://dashboard.stripe.com/settings/account>

Fill the details in the form Account name,Country,verify Number and save the same

* Now create a product   
  Go to More options > Product Catalogue > Add Product Button  
  name of the Product  
  Product Descriptions  
  Upload Image  
  Amount  
    
  After that click on Next Button and Add the Product

***16-06-2024***

* Go to cart.html and paste the below js script link  
  **<script src="https://js.stripe.com/v3/"></script>**
* MAKE BUTTON FOR Checkout in Cart.html  
   **<form method="post">**

**<input type="hidden" id="net\_price" name="amount" value="{{net\_price}}">**

**<button type="button" class="btn btn-lg btn-primary" id="checkout-button">Checkout</button>**

**</form>**

* Copy the javascript for Checkout Button as per below  
  **<script type="text/javascript">**

**var stripe = Stripe('pk\_test\_51MshH6SJuXwPfjjrrgxJ5tubbOgYAhRrqNcd4RDUSq1peWpIMZrjhIiTncSUWO9u6byeDf4KxdZPkFD4hU55mVOg00aRL5RT8x');**

**var checkoutButton = document.getElementById('checkout-button');**

**checkoutButton.addEventListener('click', function() {**

**fetch('/create-checkout-session/', {**

**method: 'POST',**

**headers:{**

**'Accept': 'application/json',**

**'X-Requested-With': 'XMLHttpRequest', //Necessary to work with request.is\_ajax()**

**},**

**body: JSON.stringify({'post\_data':'{{net\_price}}'})**

**})**

**.then(function(response) {**

**return response.json();**

**})**

**.then(function(session) {**

**return stripe.redirectToCheckout({ sessionId: session.id });**

**})**

**.then(function(result) {**

**// If `redirectToCheckout` fails due to a browser or network**

**// error, you should display the localized error message to your**

**// customer using `error.message`.**

**if (result.error) {**

**alert(result.error.message);**

**}**

**})**

**.catch(function(error) {**

**console.error('Error:', error);**

**});**

**});**

**</script>**

* Now give path for the create-checkout-session,success.html,cancel.html and myOrder as per below  
  **path('create-checkout-session/', views.create\_checkout\_session, name='payment'),**

**path('create-checkout-session/', views.create\_checkout\_session, name='payment'),**

**path('success.html/', views.success,name='success'),**

**path('cancel.html/', views.cancel,name='cancel'),**

**path('myorder/',views.myorder,name='myorder'),**

* Import in views.py as per below at top the of the file  
  **from django.http.response import JsonResponse new**

**from django.views.decorators.csrf import csrf\_exempt new**

**import json**

**import stripe**

**stripe.api\_key = settings.STRIPE\_SECRET\_KEY**

**YOUR\_DOMAIN = 'http://localhost:8000'**

* Copy the functions in views.py for create-checkout-session, success, cancel and myorder as per below:

**@csrf\_exempt**

**def create\_checkout\_session(request):**

**amount = int(json.load(request)['post\_data'])**

**final\_amount=amount\*100**

**session = stripe.checkout.Session.create(**

**payment\_method\_types=['card'],**

**line\_items=[{**

**'price\_data': {**

**'currency': 'inr',**

**'product\_data': {**

**'name': 'Checkout Session Data',**

**},**

**'unit\_amount': final\_amount,**

**},**

**'quantity': 1,**

**}],**

**mode='payment',**

**success\_url=YOUR\_DOMAIN + '/success.html',**

**cancel\_url=YOUR\_DOMAIN + '/cancel.html',)**

**return JsonResponse({'id': session.id})**

**Success**

**def success(request):**

**user=User.objects.get(email=request.session['email'])**

**carts=Cart.objects.filter(user=user,payment\_status=False)**

**for i in carts:**

**i.payment\_status=True**

**i.save()**

**carts=Cart.objects.filter(user=user,payment\_status=False)**

**request.session['cart\_count']=len(carts)**

**return render(request,'success.html')**

**Cancel**

**def cancel(request):**

**return render(request,'cancel.html')**

**MyOrder**

**def myorder(request):**

**user=User.objects.get(email=request.session['email'])**

**carts=Cart.objects.filter(user=user,payment\_status=True)**

**return render(request,'myorder.html',{'carts':carts})**

* After that we have to understand the function create\_checkout\_session in that we have to create **success.html and cancel.html**
* Copy the below data in success.html  
  **{% extends 'header.html' %}**

**{% load static %}**

**{% block content %}**

**<html>**

**<head>**

**<title>Thanks for your order!</title>**

**</head>**

**<body>**

**<div class="container text-center">**

**<p>**

**<b><h3 style="color: darkgreen;">Payment Successful</h3></b>**

**<p>Thank you for your purchase!</p>**

**<p> If you have any questions, please let us know through emails.</p>**

**</p>**

**</div>**

**</body>**

**</html>**

**{% endblock %}**

* After that go to cancel.html nad paste data as per below  
  **{% extends 'header.html' %}**

**{% load static %}**

**{% block content %}**

**<html>**

**<head>**

**<title>Checkout canceled</title>**

**</head>**

**<body>**

**<div class="container text-center">**

**<h3>Forgot to add something to your cart? </h3>**

**<h5>Shop around then come back to pay!</h5>**

**</div>**

**</body>**

**</html>**

**{% endblock %}**

* Go to **cart** function and make **payment\_status = False**
* Go to **login** function and make **payment\_status = False**carts = Cart.objects.filter(user = user,payment\_status = False)
* Create new html file name called = **myorder.html**  in templates folder and remove javascript and Net Price
* Now testing for the payment for that go to cart section and click on checkout button , the window will open for the payment by stripe
* We have some dummy Debit Card as per below  
  **Card Number :** 4242 4242 4242 4242

**Date :** 05/23 **(Set Month and Year by user input)**

CVV : 123 **(It is anything)**

* Go to header.html and add myorder section in Dropdown

**AJAX**

* AJAX **(Asynchronous JavaScript and XML)** in Django lets you update parts of a web page without reloading it. You use JavaScript to send data to the Django server, and the server sends back the needed info to update the page. This makes web pages faster and more interactive.
* Import jQuery library and put that link in the HEAD TAG of the signup.html  
  **<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>**
* Create a script for the email verification using ajax  
  Write JavaSCript for the same at the last of the signupt.html as per below code :  
    
  **<script type="text/javascript">**

**$('email').change(function(){**

**var email = $(this).val();**

**alert(email);**

**$.ajax({**

**url:'/ajax/validate\_email/',**

**data: {**

**'email':email**

**},**

**dataType:'json',**

**success: function(data){**

**if (data.is\_taken){**

**// document.getElementById("email").style.border="5px solid red";**

**document.getElementById("email\_error").innerHTML = "This Email ID Is Already Registered";**

**} else{**

**document.getElementById("email\_error").innerHTML="";**

**}**

**}**

**});**

**});**

**</script>**

* Go to urls.py and give path for the ajax script for the validate\_email as per below :  
  **path('ajax/validate\_email/',views.validate\_signup,name = 'validate\_email'),**
* Go to views.py and create the function for the validate\_signup as per below :  
  **def validate\_signup(request):**

**email = request.GET.get('email')**

**data = {**

**'is\_taken':User.objects.filter(email\_\_iexact = email).exists()**

**}**

**return JsonResponse(data)**

**18-06-2024**

**(Python Django Project Deployment)**

* We have to deploye our project on WORLD WIDE WEB
* First we have to follow the steps of **DJANGO GIRLS TUTORIAL**<https://tutorial.djangogirls.org/en/django_start_project/>
* We have put ROOT STATIC in mysite > settings.py  
  STATIC\_ROOT = BASE\_DIR / 'static'
* After that we have to put Allowed host when we deploy the project  
  **ALLOWED\_HOSTS = ['127.0.0.1', '.pythonanywhere.com','localhost']**
* After that access the folder in CMD Prompt and create txt file using command for LIBRARY INSTALLED IN OUR PROJECT as per below:  
  **(myvenv) E:\Coding World\Python\Others\MyTempProject>pip freeze > requirements.txt**
* This library does not use in Github it is required in data fetch from Github to Python Anywhere.com. It is require to create Environment.
* So requirement.txt is used to deploy the Project in Python Anywhere.com
* First we have to upload this project into Github but make sure upload separate repo for the project to upload required libraries only
* And also we have to create account on Python Anywhere.com to Deploy our Project.

Username = akshay8281

* Upload file in Github as per below commands
  + **create Repo**
  + **(myvenv) E:\Coding World\Python\Others\MyTempProject>git init**
  + **(myvenv) E:\Coding World\Python\Others\MyTempProject> git config --global user.name "akshay8281"**
  + **(myvenv) E:\Coding World\Python\Others\MyTempProject>git config --global user.email** [**akshaypitroda4@gmail.com**](mailto:akshaypitroda4@gmail.com)
* After that we have to create Git ignore root folder in our Main Project Folder  
  .gitignore /// it ignores the files in that
* Put the below content in .gitignore file  
   **Python**

**\*.pyc**

**\*~**

**\_\_pycache\_\_**

**Env**

**.env**

**myvenv/**

**venv/**

* Now check wheather file is ignore or not using below command  
  **(myvenv) E:\Coding World\Python\Others\MyTempProject>git status**
* After that add below command
  + **git add .**
  + **git commit -m “Final Project”**
  + **git remote add origin** [**https://github.com/akshay8281/Django\_Famms\_Project.git**](https://github.com/akshay8281/Django_Famms_Project.git)
  + **git push -u origin main or master**
* Now go to Python Anywhere.com > Dashboard  
  in that click on $ Bash and open the console and paste below link  
  **pip3.10 install --user pythonanywhere**/// It will install the virtual Environment in Python Anywhere.com and also install all the libraries installed in our project
* After that give below command in Python Anywhere.com > $ Bash  
  **pa\_autoconfigure\_django.py --python=3.10 https://github.com/akshay8281/Django\_Famms\_Project.git**

/// red is command and Green is Repository address of Project

* After that go to Python Anywhere.com > Web and click on **akshay8281.pythonanywhere** you will see Reload button click on that the project will reload the content and all
* Open the link mention above the Reload button you will see the Project you created
* When we have to do some changes in the project we cannot change in python IDLE so for that we have to change in Python Anywhere.com.
* Open Python Anywhere.com > Files > Select Project > open file want to edit or change and then reload the same so the changes will refresh and updated the content or data.
* When the user (Buyer) want to purchase the payment done successfully but throws error because user does not redirect to the cart or homepage.In this uses localhost:8000 and we uploaded files in Pythonanywhere.com so we have to change url in the views.py for STRIPE.
* Go to myapp > views.py and remove the url in Domain of stripe  
  **YOUR\_DOMAIN = 'http://localhost:8000'** ///Remove this URL

**YOUR\_DOMAIN = '** [**https://akshay8281.pythonanywhere.com/**](https://akshay8281.pythonanywhere.com/)**'** /// Replaced with new URL