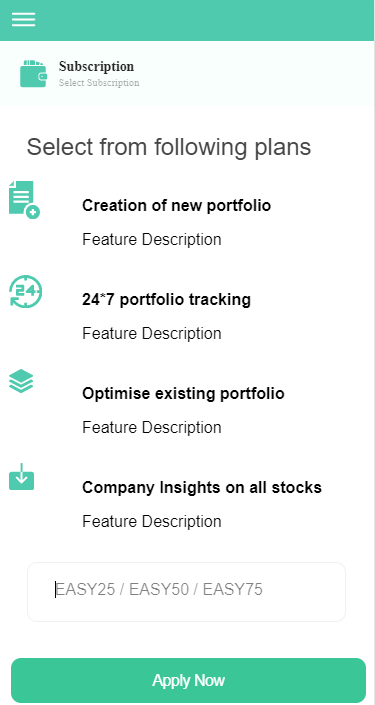
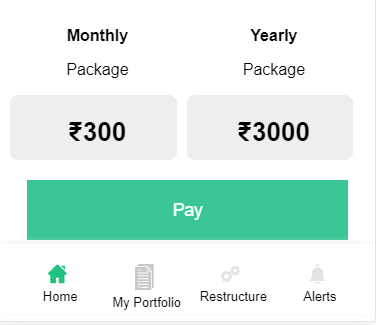
**FUNCTIONAL AND TECHNICAL COMBINED DOCUMENTATION**

**Page (49): Subscription Page and PayTM Gateway (Front End Url :**http://localhost:8100/page49/Page49)

**Functional Picture:**

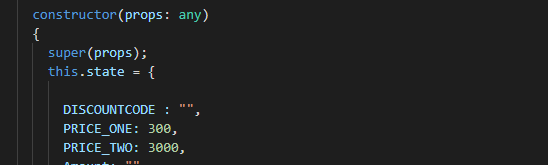




**This is a subscription page with the promo code of EASY25, EASY50 and EASY75 on monthly and yearly package.**

**Technical Explanation (Front End and Back End):**

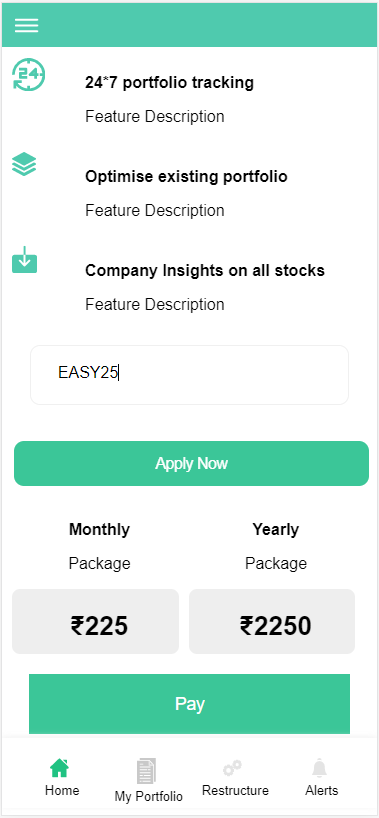
**In Page49.tsx, for No discount:**



**We have Initialized the monthly and yearly package. PRICE\_ONE and PRICE\_TWO.**

**For No Discount the Original Price will be displayed.**

**In Page49.tsx, for 25% discount:**



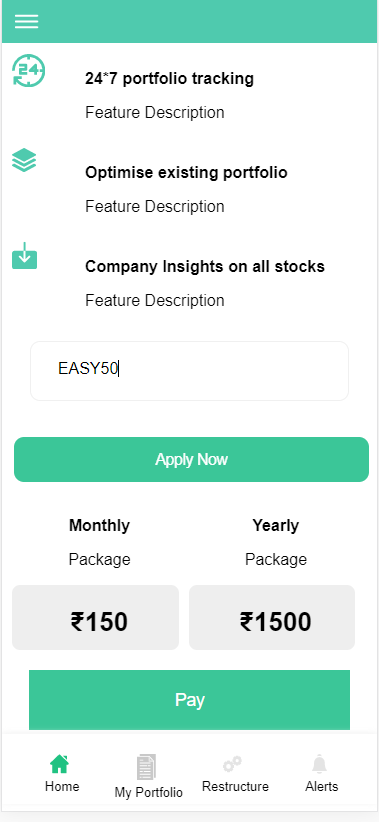


**We have Initialized the monthly and yearly package. PRICE\_ONE and PRICE\_TWO.**

**New Monthly Price = PRICE\_ONE – (PRICE\_ONE \* 0.25)**

**New Yearly Price = PRICE\_TWO – (PRICE\_TWO \* 0.25)**

**In Page49.tsx, for 50% discount:**



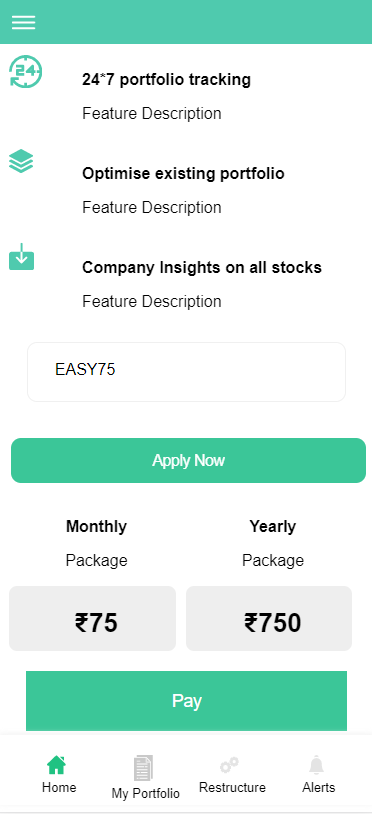


**We have Initialized the monthly and yearly package. PRICE\_ONE and PRICE\_TWO.**

**New Monthly Price = PRICE\_ONE – (PRICE\_ONE \* 0.50)**

**New Yearly Price = PRICE\_TWO – (PRICE\_TWO \* 0.50)**

**In Page49.tsx, for 75% discount:**

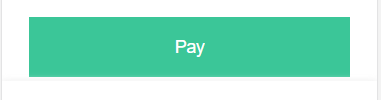




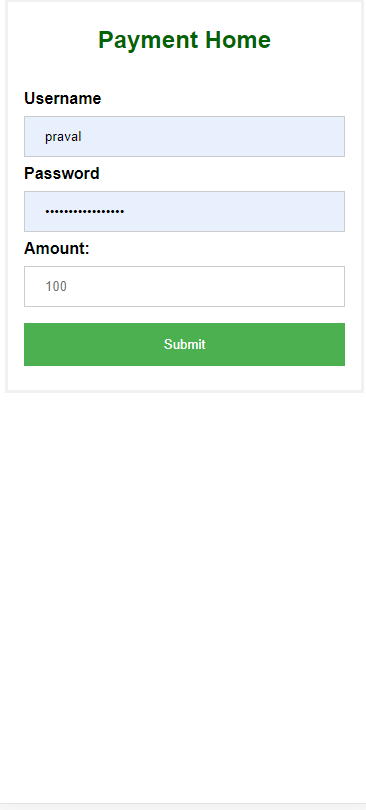
**We have Initialized the monthly and yearly package. PRICE\_ONE and PRICE\_TWO.**

**New Monthly Price = PRICE\_ONE – (PRICE\_ONE \* 0.75)**

**New Yearly Price = PRICE\_TWO – (PRICE\_TWO \* 0.75)**



**After Clicking on Pay button, where the user will give the amount that was shown on the price card after getting discount in the input box along with its credentials.**



**Now after this the backend code for PayTM payment gateway starts. Kindly pay attention.**

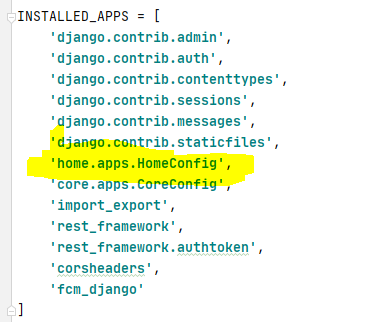
Step 1: Making the UI

Since we are keeping this concise we will only have two pages:

* **Payment page** : Where we will enter the amount we have to pay
* **Callback Page** : The response received from Paytm carrying payment status

Lets start by making a new app to organize these features.

First of all let's add the **home** app to the project. To do that we will add 'home.apps.HomeConfig' in the INSTALLED\_APPS list in settings.py .

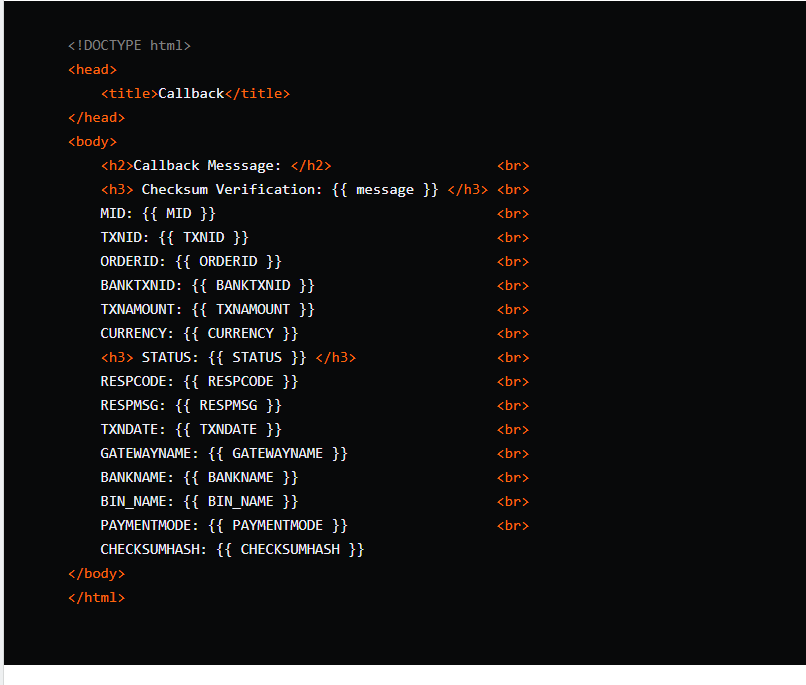


Now lets add the template files pay.html and callback.html in templates/home in the **home** app directory.

##### pay.html



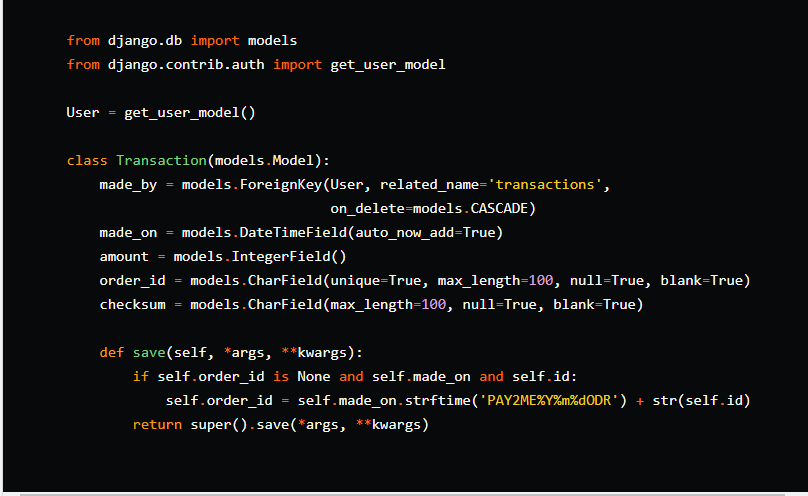
##### callback.html



The **Payment page** asks for user login information and payment amount, whereas **Callback page** shows the values of a lot of parameters which would be provided by Paytm upon completion of payment.

## Step 2: Make Transaction model

Since any payment related application would require transactions we would create a Transaction model in home/models.py as follows:



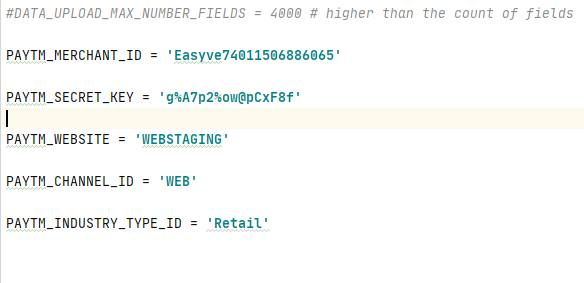
The transaction model is defined in such a way that the **order\_id** is unique and generated based on date of transaction. We can see there is a **checksum** which will store the checksum generated by python file.

We overrode the save method to automatically generate order\_id from the date and time of the transaction.

Let's run the migrations again to add the Transaction model in the database.

## Step 3: Adding Paytm Settings

Adding the following settings to home/settings.py :

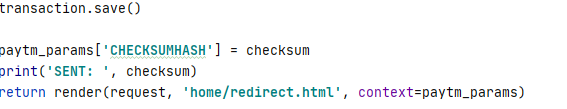


## Step 4: Create Views for Payments

Paytm has provided a repository for the checksum generation code [here](https://github.com/Paytm-Payments/Paytm_Web_Sample_Kit_Python) but the code is using a deprecated library pycrypto, the following code contains the modified file compatible with the latest pycryptodome library. So [download the file](https://gist.github.com/masterashu/f90b8ba6d4324d0a7d65ab0f2d8306c0) and save it in your payments app with file name paytm.py.

Now lets begin the create the initiate\_payment view that would receive the username, password and amount. Open payments/views.py and add the following code.





Lets understand the view part by part:

def initiate\_payment(request):

if request.method == "GET":

return render(request, 'payments/pay.html')

The first step is to check the method of the request. When the request method is **GET** then we will just return the payment page.

try:

username = request.POST['username']

password = request.POST['password']

amount = int(request.POST['amount'])

user = authenticate(request, username=username, password=password)

if user is None:

raise ValueError

auth\_login(request=request, user=user)

except:

return render(request, 'payments/pay.html', context={'error': 'Wrong Account Details or amount'})

Then we verify the username and password entered by the user in the form and tries to log in the user. If the login details are invalid then the view returns back the login page with an error message.

transaction = Transaction.objects.create(made\_by=user, amount=amount)

transaction.save()

merchant\_key = settings.PAYTM\_SECRET\_KEY

Next we create a Transaction object for our user and get our merchant key from settings.py.

Next we create a Transaction object for our user and get our merchant key from settings.py.

params = (

('MID', settings.PAYTM\_MERCHANT\_ID),

('ORDER\_ID', str(transaction.order\_id)),

('CUST\_ID', str(transaction.made\_by.email)),

('TXN\_AMOUNT', str(transaction.amount)),

('CHANNEL\_ID', settings.PAYTM\_CHANNEL\_ID),

('WEBSITE', settings.PAYTM\_WEBSITE),

# ('EMAIL', request.user.email),

# ('MOBILE\_N0', '9911223388'),

('INDUSTRY\_TYPE\_ID', settings.PAYTM\_INDUSTRY\_TYPE\_ID),

('CALLBACK\_URL', 'http://127.0.0.1:8000/callback/'),

# ('PAYMENT\_MODE\_ONLY', 'NO'),

)

paytm\_params = dict(params)

Then we create a dictionary for all the settings for the **Paytm Checksum Generator** to create a checksum.

checksum = generate\_checksum(paytm\_params, merchant\_key)

transaction.checksum = checksum

transaction.save()

paytm\_params['CHECKSUMHASH'] = checksum

Next after generating the checksum we add the checksum to the paytm\_params dictionary as well as the Transaction object.

return render(request, 'payment/redirect.html', context=paytm\_params)

At the end we return the redirect page.

## Step 5: Create a Redirect Page

Create a redirect page in home/templates/home/redirect.html:

<html>

<head>

<title>Merchant Check Out Page</title>

</head>

<body>

<h1>Please do not refresh this page...</h1>

<form method="post" action="https://securegw-stage.paytm.in/order/process/" name="f1">

<table>

<tbody>

<input type="hidden" name="MID" value="{{ MID }}">

<input type="hidden" name="WEBSITE" value="{{ WEBSITE }}">

<input type="hidden" name="ORDER\_ID" value="{{ ORDER\_ID }}">

<input type="hidden" name="CUST\_ID" value="{{ CUST\_ID }}">

<input type="hidden" name="INDUSTRY\_TYPE\_ID" value="{{ INDUSTRY\_TYPE\_ID }}">

<input type="hidden" name="CHANNEL\_ID" value="{{ CHANNEL\_ID }}">

<input type="hidden" name="TXN\_AMOUNT" value="{{ TXN\_AMOUNT }}">

<input type="hidden" name="CALLBACK\_URL" value="{{ CALLBACK\_URL }}">

<input type="hidden" name="CHECKSUMHASH" value="{{ CHECKSUMHASH }}">

</tbody>

</table>

<script type="text/javascript">

document.f1.submit();

</script>

</form>

</body>

</html>

The redirect page contains a simple form that contains the fields from the dictionary. We used the javascript code to automatically post the form to the paytm gateway.

## Step 6: Create the Callback View

The callback view is the view which would be called when paytm will respond with the status of the Transaction.

Add the following callback view in the home/views.py

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

@csrf\_exempt

def callback(request):

if request.method == 'POST':

received\_data = dict(request.POST)

paytm\_params = {}

paytm\_checksum = received\_data['CHECKSUMHASH'][0]

for key, value in received\_data.items():

if key == 'CHECKSUMHASH':

paytm\_checksum = value[0]

else:

paytm\_params[key] = str(value[0])

# Verify checksum

is\_valid\_checksum = verify\_checksum(paytm\_params, settings.PAYTM\_SECRET\_KEY, str(paytm\_checksum))

if is\_valid\_checksum:

received\_data['message'] = "Checksum Matched"

else:

received\_data['message'] = "Checksum Mismatched"

return render(request, 'payments/callback.html', context=received\_data)

The callback view receives a post request from the paytm server. Then we retrieve the received data in a dictionary and and verify the checksum sent from paytm as to prevent forged requests. We then return the page with the details of the transaction and the message.

## Step 7: Creating the routes.

As the final step, Let's make the URLs for the views and get this long tutorial over. Create/open urls.py in the payments app. Add the following routes.

from django.urls import path

from .views import initiate\_payment, callback

urlpatterns = [

path('pay/', initiate\_payment, name='pay'),

path('callback/', callback, name='callback'),

]

Now let's include these URLs in the pay2me/urls.py

from django.urls import include

urlpatterns = [

# include

path('', include('payments.urls'))

]

**DONE!!!**