Tops Technology

Module 16) Python DB and Framework

Presented By : Nandni Vala

Payment Integration Using Paytm

1.• Introduction to integrating payment gateways (like Paytm) in Django projects.

Integrating a payment gateway in a Django project allows you to accept online payments securely. Payment gateways like Paytm provide APIs and SDKs for handling transactions.

Overview of Payment Gateway Integration

Payment gateways process payments made by users via credit/debit cards, UPI, wallets, or net banking. Integration typically involves the following steps:

- **User Interaction**: A user initiates a payment from the website or application.
- **Server Request**: The application sends a request to the payment gateway with transaction details.
- **Gateway Interaction**: The payment gateway handles user authentication and payment processing.
- **Response Handling**: The gateway returns a response (success or failure) to the application.

- **Steps for Integrating Paytm Payment Gateway**
- **Step 1: Register with Paytm**
- Sign up for a Paytm Business account.
- Obtain the **Merchant ID**, **Merchant Key**, and **Website Name** from the Paytm dashboard.
- **Step 2: Install Required Libraries**
- Install the required SDK or libraries for Paytm. For Python, use the Paytm payment library or make HTTP requests to the Paytm API.
- pip install django-paytm
- **Step 3: Set Up Configuration in settings.py**
- Add the Paytm credentials in the settings.
- PAYTM_MERCHANT_ID = 'your_merchant_id'
- PAYTM_MERCHANT_KEY = 'your_merchant_key'
- PAYTM_WEBSITE = 'your_website_name'
- PAYTM_CALLBACK_URL = 'http://your-domain.com/payment/response/'
- PAYTM_TRANSACTION_URL = 'https://securegw-stage.paytm.in/theia/processTransaction' # Use production URL in production

- **Create Payment Views**
- Handle the initiation of the payment and response handling.
- **Initiate Payment**:
- import requests
- from django.shortcuts import render
- from django.http import JsonResponse
- from .utils import generate_checksum # Utility to generate Paytm checksum
- def initiate_payment(request):
-     if request.method == "POST":
-         amount = request.POST.get("amount") # Payment amount
-         order_id = "ORDER" + str(int(time.time())) # Unique order ID

```
➢  # Paytm parameters
➢      paytm_params = {
➢          'MID': settings.PAYTM_MERCHANT_ID,
➢          'ORDER_ID': order_id,
➢          'CUST_ID': 'customer123',
➢          'TXN_AMOUNT': str(amount),
➢          'CHANNEL_ID': 'WEB',
➢          'WEBSITE': settings.PAYTM_WEBSITE,
➢          'INDUSTRY_TYPE_ID': 'Retail',
➢          'CALLBACK_URL': settings.PAYTM_CALLBACK_URL,
➢      }
➢      checksum = generate_checksum(paytm_params,
   settings.PAYTM_MERCHANT_KEY)
➢      paytm_params['CHECKSUMHASH'] = checksum
➢      return render(request, 'paytm_payment.html', {'paytm_params':
   paytm_params, 'paytm_url': settings.PAYTM_TRANSACTION_URL})
```

- **HTML Form** (paytm_payment.html):
- <form method="post" action="{{ paytm_url }}">
-     {% for key, value in paytm_params.items %}
-         <input type="hidden" name="{{ key }}" value="{{ value }}">
-     {% endfor %}
-     <button type="submit">Pay Now</button>
- </form>
- **Handle Payment Response**
- Capture the response from Paytm after the payment.
- from django.views.decorators.csrf import csrf_exempt
- from .utils import verify_checksum # Utility to verify Paytm checksum
- @csrf_exempt
- def payment_response(request):
-     if request.method == "POST":
-         received_data = dict(request.POST)
-         paytm_checksum = received_data.pop('CHECKSUMHASH', [None])[0]

- is_valid_checksum = verify_checksum(received_data, settings.PAYTM_MERCHANT_KEY, paytm_checksum)
-         if is_valid_checksum and received_data['RESPCODE'] == '01':
-             return JsonResponse({"message": "Payment successful"})
-         else:
-             return JsonResponse({"message": "Payment failed"})
- **Utilities for Paytm Checksum**
- Paytm requires checksum generation and verification for secure transactions. Use the Paytm Python SDK or write custom utility functions.
- **Testing the Integration**
- Use the **Staging Environment** provided by Paytm for testing.
- Verify transaction logs in the Paytm dashboard.
- **Deploy to Production**
- Switch to the production API endpoint in the configuration.
- Use secure HTTPS for your website.