

# Phase 7 — Integration & External Access (NGO Management System)

**Objective:** Connect the NGO Management System with external services such as online payment gateways, donor verification services, and external NGO databases to provide a seamless donor experience and real-time campaign updates.

---

Purpose: Allow donors to contribute online securely and track donations in Salesforce.

Steps Implemented (Click by Click):

1. Go to Setup → Named Credentials → New.

2. Enter details:

Label: NGO\_PaymentAPI

Name: NGO\_PaymentAPI

URL: <https://api.razorpay.com/v1/> (or PayPal API endpoint)

Authentication: Basic Auth (with API Key & Secret)

**3. Click Save.**

4. Create an Apex Class for sending payment requests:

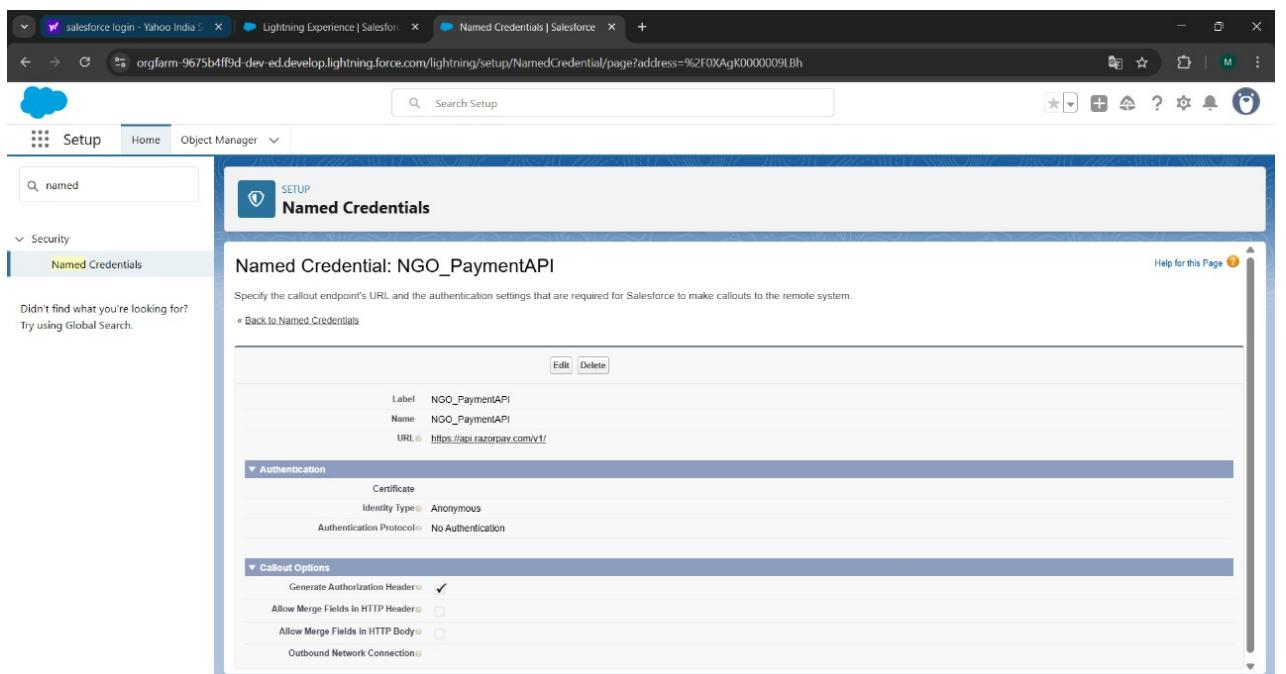
```
HttpRequest req = new HttpRequest();
req.setEndpoint('callout:NGO_PaymentAPI/payment_link');
req.setMethod('POST');
req.setHeader('Content-Type','application/json');
req.setBody('{ "amount": 50000, "currency": "INR", "customer": {"email":
```

```
"donor@example.com"} }');
```

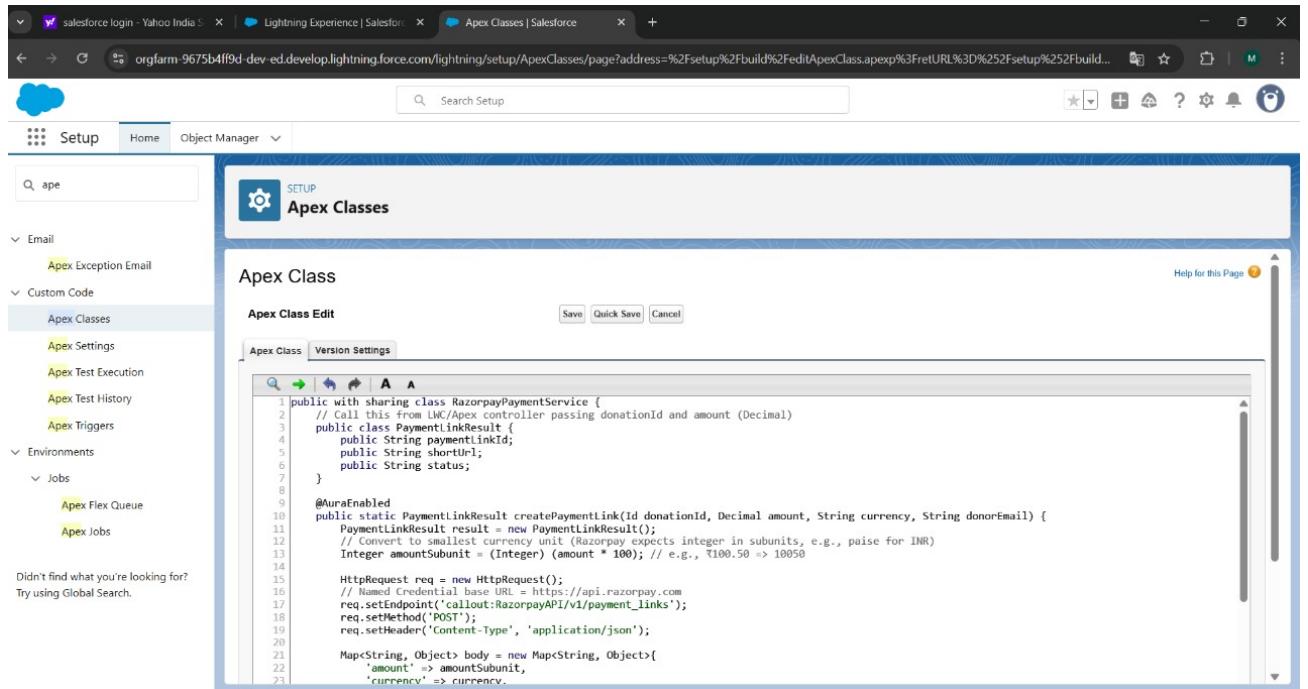
```
HttpResponse res = new Http().send(req);
```

5. Capture Payment ID / Status in Donation\_\_c record after response
6. Update logic: On successful payment → update Donation Status = Confirmed.
7. Expose the Payment button in Donation LWC component → Add to Donor Portal page.

 Now donors can make online contributions directly, and all transactions are stored in Salesforce.



The screenshot shows the 'Named Credentials' setup page in Salesforce. A named credential named 'NGO\_PaymentAPI' is being edited. The URL is set to 'https://api.razorpay.com/v1/'. The authentication section is configured with 'Anonymous' identity type and 'No Authentication' protocol. Callout options include generating an authorization header and allowing merge fields in headers and body. The page includes a search bar, a back button, and a help link.



The screenshot shows the 'Apex Class' setup page in Salesforce. An apex class named 'RazorpayPaymentService' is being edited. The code defines a public sharing class with a static method 'createPaymentLink'. This method takes parameters: donationId (Id), amount (Decimal), currency (String), and donorEmail (String). It converts the amount to a subunit (amountSubunit) and creates an HttpRequest object to call the Razorpay API at 'https://api.razorpay.com/v1/payment\_links'. The request is set to POST method and application/json content type. The page includes a search bar, a back button, and a help link.

```
public with sharing class RazorpayPaymentService {
    // Call this from LWC/Apex controller passing donationId and amount (Decimal)
    public class PaymentLinkResult {
        public String paymentLinkId;
        public String shortUrl;
        public String status;
    }
    @AuraEnabled
    public static PaymentLinkResult createPaymentLink(Id donationId, Decimal amount, String currency, String donorEmail) {
        PaymentLinkResult result = new PaymentLinkResult();
        // Convert to smallest currency unit (Razorpay expects integer in subunits, e.g., pause for INR)
        Integer amountSubunit = (Integer) (amount * 100); // e.g., ₹100.50 => 10050
        HttpRequest req = new HttpRequest();
        // Named Credential base URL = https://api.razorpay.com
        req.setEndpoint('callout:RazorpayAPI/v1/payment_links');
        req.setMethod('POST');
        req.setHeader('Content-Type', 'application/json');
        Map<String, Object> body = new Map<String, Object>{
            'amount' => amountSubunit,
            'currency' => currency,
        };
    }
}
```

## B. REST API Integration for Donor Verification

Purpose: Validate donor identity via external verification service (e.g., Aadhaar/Email verification API).

Steps Implemented (Click by Click):

1. Go to Setup → Named Credentials → New.

2. Enter details:

Label: DonorVerifyAPI

Name: DonorVerifyAPI

URL: <https://api.donorverify.com/v1>

Authentication: API Key Header

3. Create an Apex Class for donor verification:

```
HttpRequest req = new HttpRequest();
req.setEndpoint('callout:DonorVerifyAPI/verify');
req.setMethod('POST');
req.setHeader('Authorization','Bearer YOUR_API_KEY');
req.setHeader('Content-Type','application/json');
req.setBody('{ "email": "donor@example.com" }');
HttpResponse res = new Http().send(req);
```

4. Parse JSON response → Update `Donor__c.Verified__c` = TRUE/FALSE.

5. Display verification result on Donor Detail Page using an LWC badge.

Donor authenticity is verified and stored in Salesforce.

## C. Platform Events & Change Data Capture (CDC)

Purpose: Sync NGO campaign updates across Salesforce and external websites/mobile

apps.

Steps Implemented (Click by Click):

1. Go to Setup → Platform Events → New Platform Event.

Label: Campaign\_Update

API Name: Campaign\_Update\_e

Fields: CampaignId, Title, Status, GoalAmount, RaisedAmount

2. Save & Deploy Platform Event.

3. Create Apex Trigger on Campaign\_c to publish event when a record is updated.

4. External apps subscribe via CometD/WebSocket to get updates in real time.

- Now external NGO portals/mobile apps get live updates whenever campaigns change.

#### D. Salesforce Connect (External NGO Database)

Purpose: View NGO partner database (e.g., Volunteers, Beneficiaries) inside Salesforce without storing data locally.

Steps Implemented (Click by Click):

1. Go to Setup → External Data Sources → New External Data Source.

2. Enter details:

**Name: NGO\_ExternalDB**

**Type: OData 4.0**

**URL: https://ngo-external-db.com/odata**

**Authentication: Named Credential (OAuth)**

3. Validate & Sync tables.

4. Create External Objects like Volunteer\_x, Beneficiary\_x

5. Add related list of Volunteers/Beneficiaries to NGO Campaign record page.

- Users can see external NGO data inside Salesforce seamlessly

#### E. OAuth & Remote Site Settings

Purpose: Enable secure API access for all integrations.

Steps Implemented (Click by Click):

1. Go to Setup → Security → Remote Site Settings → New

Add URLs for Payment Gateway, Donor Verification API, NGO Database.

2. Go to Setup → App Manager → New Connected App.

Provide OAuth Scopes (Full Access / API / Refresh Token).

Save and copy Consumer Key & Secret.

3. Update Named Credentials to use OAuth 2.0

External integrations are now secured with OAuth and remote site settings.

#### Summary of Implemented Integrations

Payment Gateway (Razorpay/PayPal): Enables secure online donations.

Donor Verification API: Ensures donor identity authenticity.

Platform Events & CDC: Keeps campaigns synced with external portals.

Salesforce Connect: Displays live external NGO data.

OAuth & Remote Site Settings: Secures all external API communication.