Library Management Phase 7 — Integration & External Access (Implemented Parts)

Step 1: Create an External Credential (one time)

Go to Setup → Quick Find → type External Credentials → click External Credentials.

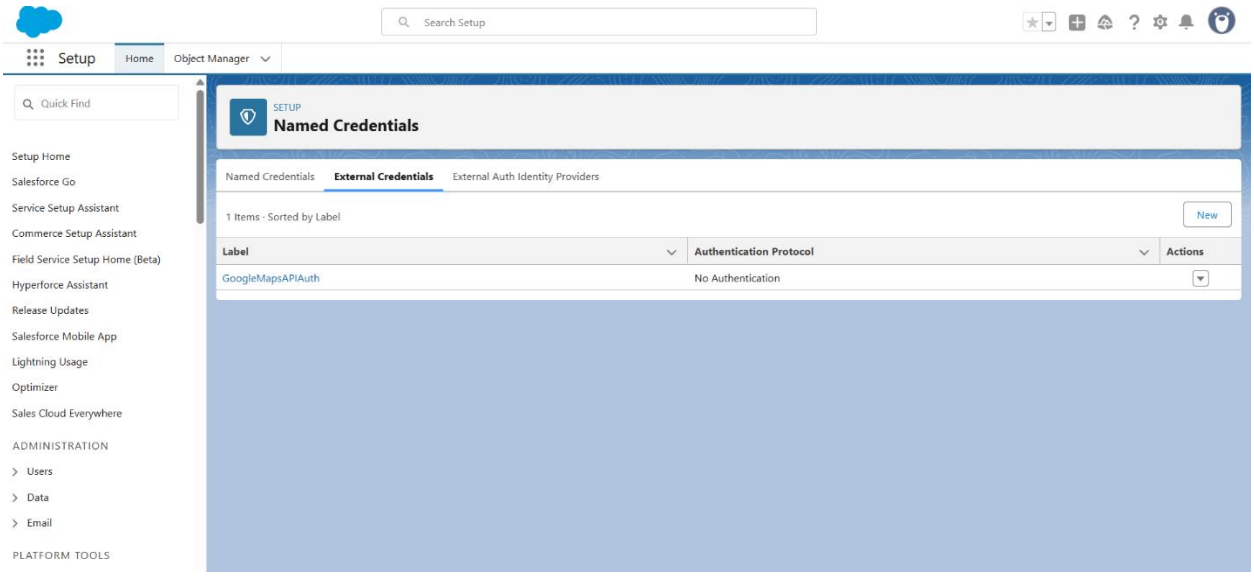
Click New External Credential.

Fill the form:

- Label: GoogleMapsAPIAuth
- Name: GoogleMapsAPIAuth
- Authentication Protocol: No Authentication

Click Save.

You now have an External Credential called GoogleMapsAPIAuth.



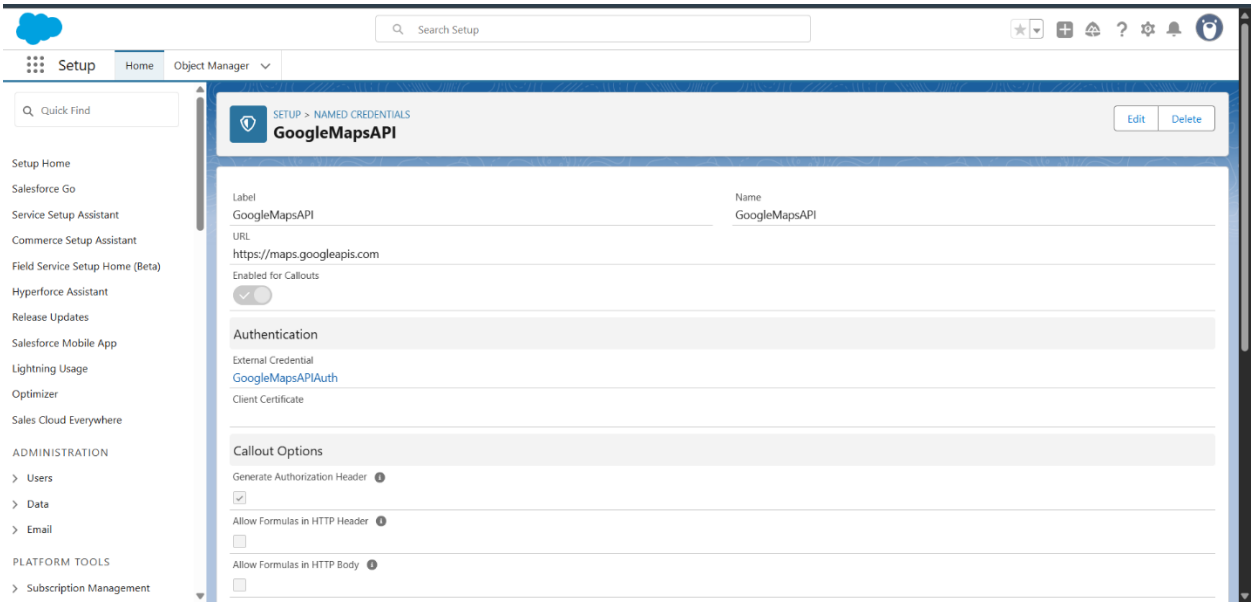
Step 2: Create the Named Credential (link to External Credential)

Go to Setup → Quick Find → Named Credentials → click Named Credentials → click New.

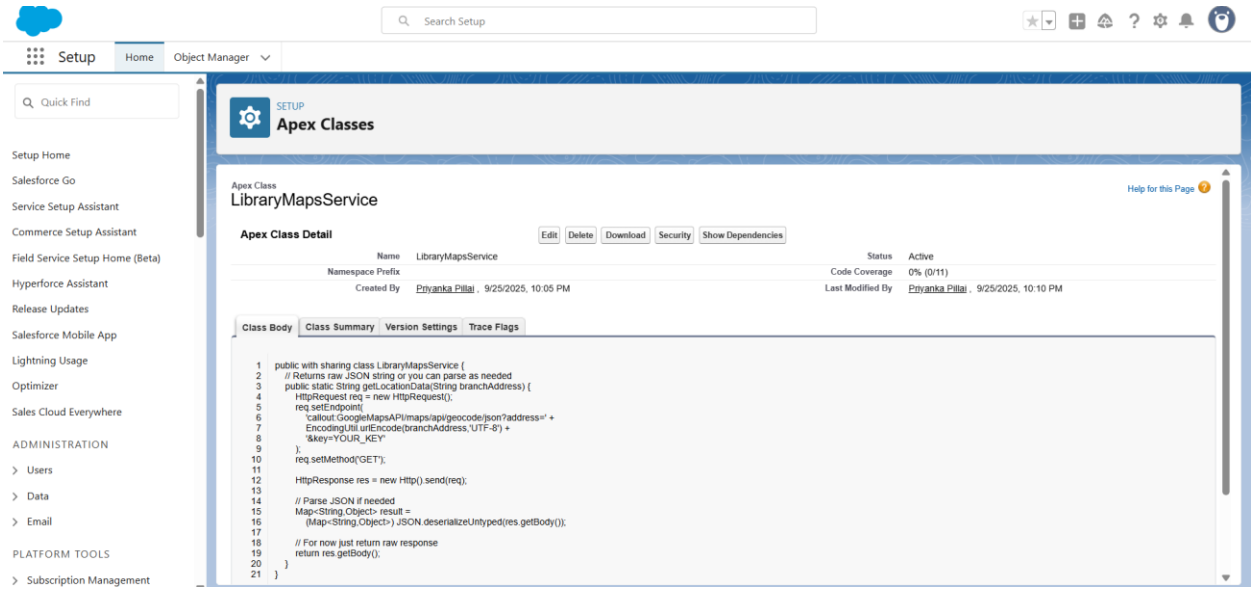
Fill details:

- Label: GoogleMapsAPI
- Name: GoogleMapsAPI
- External Credential: pick GoogleMapsAPIAuth
- URL: https://maps.googleapis.com
- Generate Authorization Header: leave checked
- Authentication: nothing extra (because using API key)

Click Save.


Search Setup
Setup
Home
Object Manager
Quick Find
Setup Home
Salesforce Go
Service Setup Assistant
Commerce Setup Assistant
Field Service Setup Home (Beta)
Hyperforce Assistant
Release Updates
Salesforce Mobile App
Lightning Usage
Optimizer
Sales Cloud Everywhere
ADMINISTRATION
Users
Data
Email
PLATFORM TOOLS
Subscription Management
SETUP > NAMED CREDENTIALS
GoogleMapsAPI
Edit
Delete
Label
GoogleMapsAPI
Name
GoogleMapsAPI
URL
https://maps.googleapis.com
Enabled for Callouts
Authentication
External Credential
GoogleMapsAPIAuth
Client Certificate
Callout Options
Generate Authorization Header
Allow Formulas in HTTP Header
Allow Formulas in HTTP Body


File
Edit
Debug
Test
Workspace
Help
TeacherServiceHandle.apxc *
TeacherServiceHandleTest.apxc *
Teacher_Service.apxc *
TeacherTriggerTest.apxc *
ScheduleTeacherSummary.apxc *
TeacherSummaryBatch.apxc *
LibraryMapsService.apxc *
Code Coverage: None
API Version: 64
Go To
1 public with sharing class LibraryMapsService {
2 // Returns raw JSON string or you can parse as needed
3 public static String getLocationData(String branchAddress) {
4 HttpRequest req = new HttpRequest();
5 req.setEndpoint(
6 'callout:GoogleMapsAPI/maps/api/geocode/json?address=' +
7 EncodingUtil.urlEncode(branchAddress,'UTF-8') +
8 '&key=YOUR_KEY'
9);
10 req.setMethod('GET');
11
12 HttpResponse res = new Http().send(req);
13
14 // Parse JSON if needed
15 Map<String,Object> result =
16 (Map<String,Object>) JSON.deserializeUntyped(res.getBody());
17
18 // For now just return raw response
19 return res.getBody();
20 }
21 }



An LWC is always 3 separate files inside force-app/main/default/lwc/libraryMap/:

- libraryMap.html
- libraryMap.js
- libraryMap.js-meta.xml

Here's the correct code for your libraryMap LWC — all in one answer:

📄 libraryMap.html

```
<template>
  <lightning-card title="Nearest Library Branch">
    <div class="slds-p-around_medium">
      <iframe
        src={mapUrl}
```

```
            width="100%"
            height="300"
            style="border:0;"
            allowfullscreen
            loading="lazy">
          </iframe>
        </div>
    </lightning-card>
</template>
```

libraryMap.js

```
import { LightningElement, api } from 'lwc';

export default class LibraryMap extends LightningElement {
    @api branchAddress;
    mapUrl;

    connectedCallback() {
        // Directly build Google Maps iframe URL
        this.mapUrl =
```

```
'https://www.google.com/maps/embed/v1/place?key=YOUR_KEY&q='
+ this.branchAddress;
    }
}
```

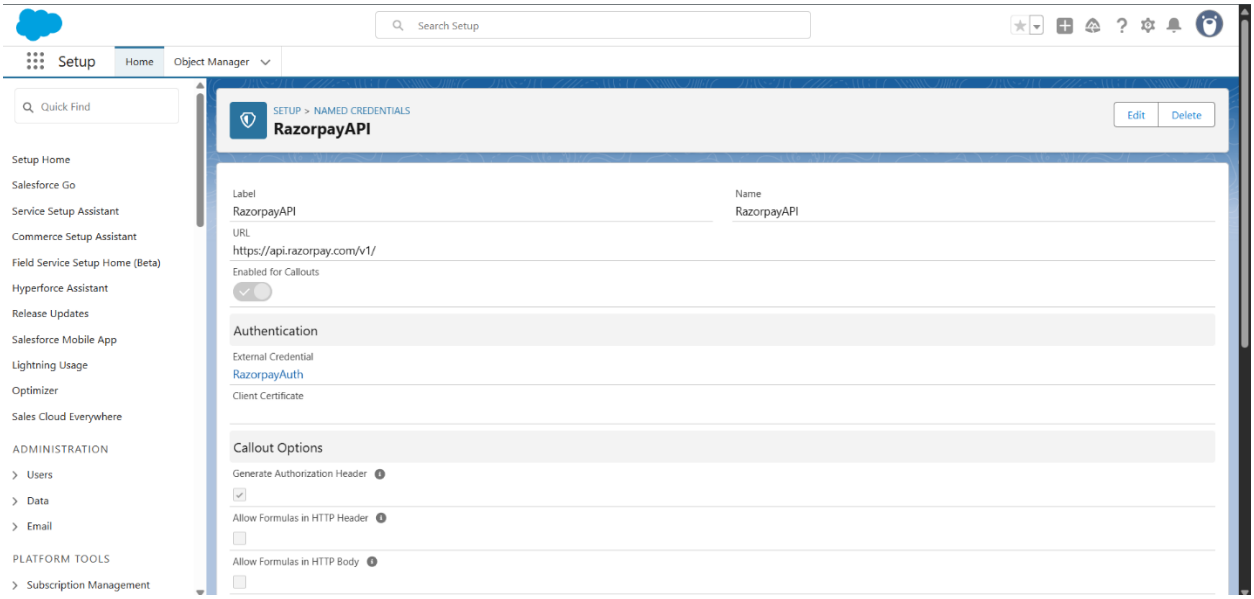
libraryMap.js-meta.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>59.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
    <targetConfigs>
        <targetConfig
targets="lightning_RecordPage,lightningAppPage,lightning_HomePage"
>
```

```
        <property name="branchAddress" type="String" label="Branch Address" description="Address to show on map"/>
    </targetConfig>
  </targetConfigs>
</LightningComponentBundle>
```

Step 3 - Create a Named Credential

- In Setup, search Named Credentials → New.
- Fill:
- Label: RazorpayAPI
- Name: RazorpayAPI
- External Credential: RazorpayAuth (the one you just made)
- URL: https://api.razorpay.com/v1/
- Leave “Generate Authorization Header” checked.
- Save.




```
public with sharing class RazorpayPaymentService {
    public static String createPaymentLink() {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:RazorpayAPI/payment_links'); // note plural endpoint
        req.setMethod('POST');
        req.setHeader('Content-Type','application/json');
        req.setBody('{' +
            '"amount": 1000,' +
            '"currency": "INR",' +
            '"customer": {"email": "test@example.com"}' +
        '}');
```

```
        HttpResponse res = new Http().send(req);
        System.debug('Response: ' + res.getBody());
        return res.getBody();
    }
}
```