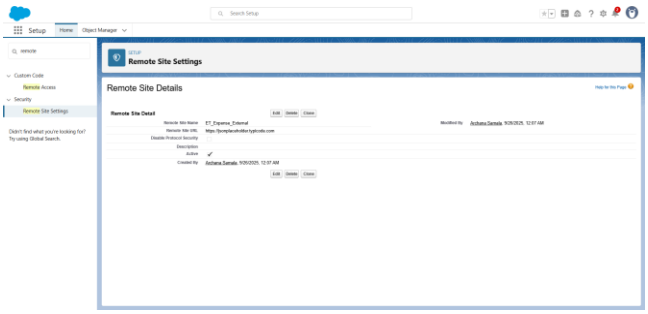
PHASE – 7 Integration & External Access

In this phase, I implemented external integration features for my **Expense Tracker Project**. The goal was to securely connect Salesforce with external systems, send/receive data, and enable event-driven communication. Below is what I built step by step.

1. Remote-site Settings

I created a Remote-Site Settings called ET_Expense_External to securely store the external API URL and authentication details.

- Label: ET_Expense_External
- Name: ET_Expense_External
- URL: https://jsonplaceholder.typicode.com
- This allowed me to make callouts without manually handling authentication or Remote Site Settings.



2. Apex Callouts

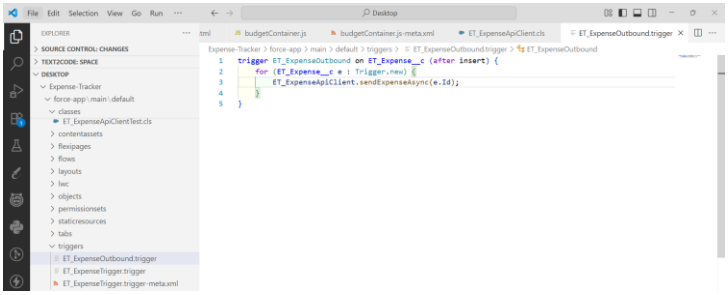
I created an Apex class ET_ExpenseApiClient which makes a POST callout to the external API whenever an expense is created.

- It sends expense details (Amount, Date, Category, Budget).
- The endpoint uses the Named Credential (callout: ET_Expense_External).
- Callouts are executed asynchronously using @future(callout=true).

This enables Salesforce to push new expense data to an external system.

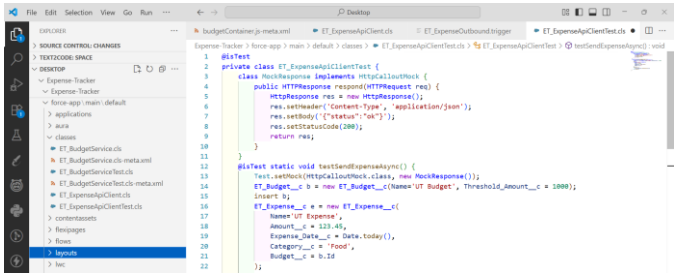
3. Trigger for Automation

I added a trigger ET_ExpenseOutbound on **ET_Expense__c** that automatically calls ET_ExpenseApiClient.sendExpenseAsync() after an expense is inserted. This means every time a new expense is created in Salesforce, the external system is notified.



4. Test Class

I wrote a test class ET_ExpenseApiClientTest that uses HttpCalloutMock to simulate API responses. This ensures the callout logic can be tested without actually hitting the external API. The test inserts a Budget and Expense record, then verifies that the callout is made successfully.



```
@isTest
private class ET_ExpenseApiClientTest {
    class MockResponse implements HttpCalloutMock {
        public HTTPResponse respond(HTTPRequest req) {
            HttpResponse res = new HttpResponse();
            res.setHeader('Content-Type', 'application/json');
            res.setBody('{"status":"ok"}');
            res.setStatusCode(200);
            return res;
        }
    }
    @isTest static void testSendExpenseAsync() {
        Test.setMock(HttpCalloutMock.class, new MockResponse());
        ET_Budget__c b = new ET_Budget__c(Name='UT Budget', Threshold_Amount__c = 1000);
        insert b;
        ET_Expense__c e = new ET_Expense__c(
            Name='UT Expense',
            Amount__c = 123.45,
            Expense_Date__c = Date.today(),
            Category__c = 'Food',
            Budget__c = b.Id
        );
```

5. Platform Events

I created a **Platform Event** called **ET_Expense_Integration__e** to broadcast expense details in real time.

- Fields: ExpenseId, Amount, Category, BudgetId
- Whenever an expense is created, Salesforce can publish this event.