

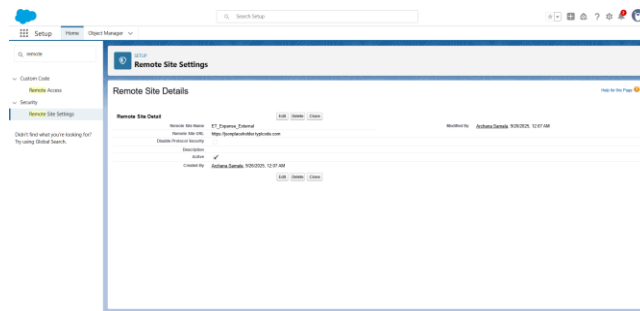
PHASE – 7 Integration & External Access

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

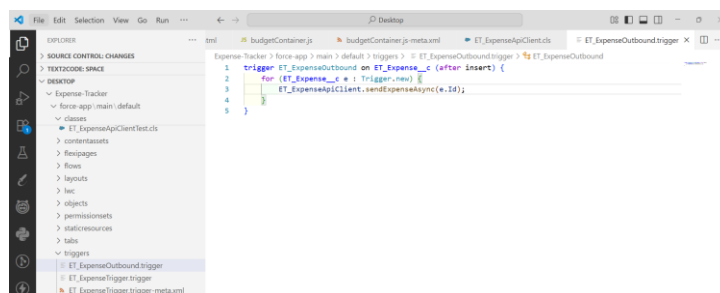
I created an Apex class ET_ExpenseApiClient which makes a POST callout to the external API whenever an expense is created.

- 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.

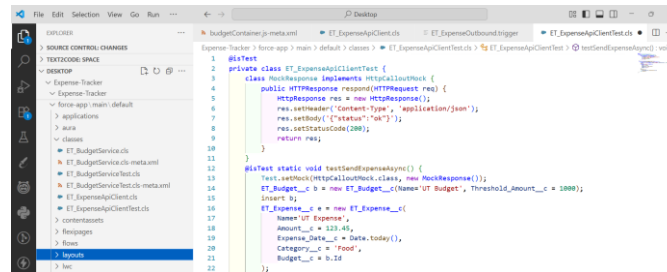
3. Trigger for Automation

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.

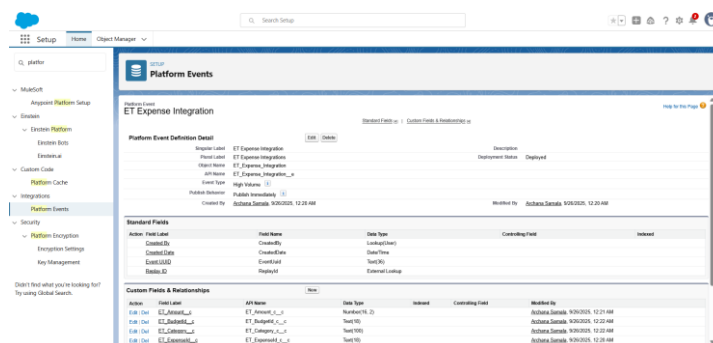


```
1  @test
2  private class ET_ExpenseApiClientTest {
3      class MockHttpResponse implements HttpResponse {
4          public HTTPResponse respond(HTTPRequest req) {
5              MockHttpResponse res = new MockHttpResponse();
6              res.setHeader("Content-Type", "application/json");
7              res.setBody("{\"status\":\"ok\"}");
8              res.setStatusCode(200);
9              return res;
10         }
11     }
12
13     @test static void testSendExpenseAsync() {
14         TestContext.setCurrentMock(new MockHttpResponse());
15         ET_Budget__c b = new ET_Budget__c(Name='UT Budget', Threshold_Amount__c = 1000);
16         insert b;
17         ET_Expense__c e = new ET_Expense__c(
18             Name='UT Expense',
19             Amount__c = 123.45,
20             Expense_Date__c = Date.today(),
21             Category__c = 'Food',
22             Budget__c = b.Id
23         );
24     }
```

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.



Platform Event Definition Detail					
Property Label	Value	Description	Deployment Status	Created By	Created Date
Platform Name	ET Expense Integration		Deployed		
API Name	ET_Expense_Integration__e				
Event Type	High Volume				
Published Version	Public Knowledge				
Created By	Richana Sanyal	9/30/2025, 12:28:48M			

Standard Fields					
Action / Field Label	Field Name	Data Type	Controlling Field	Indexed	
CreateID	CreateID	Lookup (Text)			
CreateDate	CreateDate	Date/Time			
EventID	EventID	Text (50)			
Field ID	Field ID	External Lookup			

Custom Fields & Relationships					
Index	Field Label	API Name	Data Type	Indexed	Controlling Field
0:0 (M)	ET_Amount__c	ET_Amount__c	Number (1, 2)		
0:0 (M)	ET_Category__c	ET_Category__c	Text (50)		
0:0 (M)	ET_BudgetId__c	ET_BudgetId__c	Text (50)		
0:0 (M)	ET_ExpenseId__c	ET_ExpenseId__c	Text (50)		