

## FinSmart CRM Project – Phase 7:

Credit Bureau Integration (Minimal Implementation)

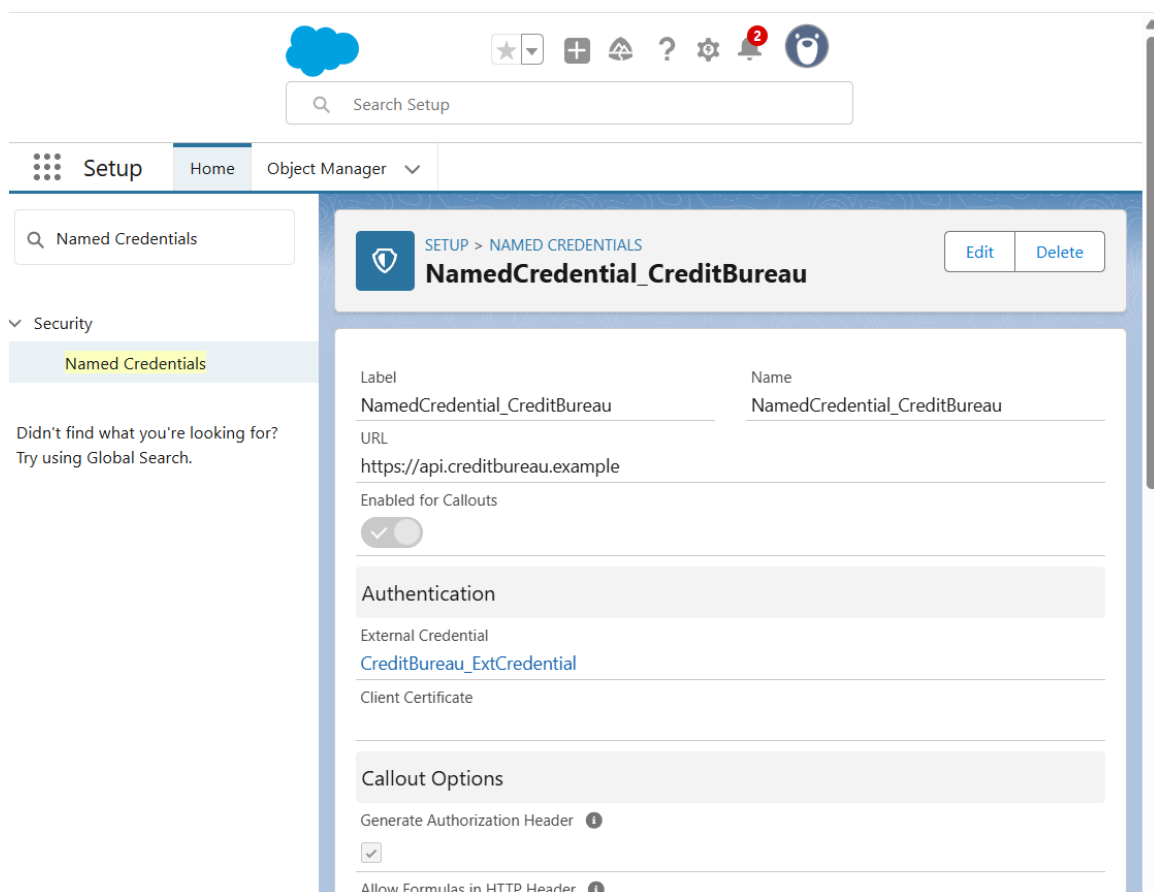
### Step 1: Named Credential

What I did: I created a Named Credential in Salesforce for connecting to the Credit Bureau API.

Reason: Named Credentials simplify API callouts by securely storing the endpoint URL and authentication details. This avoids hardcoding sensitive information in Apex classes and makes future updates easier.

Steps:

1. Go to Setup → Named Credentials → New.



The screenshot shows the Salesforce Setup interface. At the top, there's a search bar labeled "Search Setup". Below it, the navigation menu includes "Setup", "Home", and "Object Manager". The "Setup" menu is expanded, showing "Named Credentials" under the "Security" section. The main content area displays the configuration for a Named Credential named "NamedCredential\_CreditBureau".

**Named Credential Configuration:**

- Label:** NamedCredential\_CreditBureau
- Name:** NamedCredential\_CreditBureau
- URL:** https://api.creditbureau.example
- Enabled for Callouts:** ☒
- Authentication:**
  - External Credential:** [CreditBureau\\_ExtCredential](#)
  - Client Certificate:**
- Callout Options:**
  - Generate Authorization Header:** ☒
  - Allow Formulas in HTTP Header:** ☐

**Apex Callout:**

```
public with sharing class CreditBureauService {  
  
    @future(callout=true)
```

```

public static void getCreditScore(Id loanId) {

    Loan_Application__c loan = [SELECT Id, Credit_Score__c FROM Loan_Application__c
WHERE Id = :loanId LIMIT 1];

    HttpRequest req = new HttpRequest();

    req.setEndpoint('callout:NamedCredential__CreditBureau/score?cust=' + loan.Id);

    req.setMethod('GET');


    Http http = new Http();

    HttpResponse res = http.send(req);

    if (res.getStatusCode() == 200) {

        Integer score = Integer.valueOf(res.getBody());

        loan.Credit_Score__c = score;

        if (score >= 650) loan.Loan_Status__c = 'Under_Review';

        else loan.Loan_Status__c = 'Rejected';

        update loan;

    }

}
}

```

## Step 2: Test Callout (Using HttpCalloutMock)

What I did: I created a Mock class and a Test class to simulate the Credit Bureau API response without calling a real service.

Reason: Salesforce requires test classes to mock callouts. This ensures code can be deployed while avoiding dependencies on external systems during testing.

### Mock Class Code:

```

@Test
global class CreditBureauMock implements HttpCalloutMock {

```

```

global HttpResponse respond(HttpRequest req) {

    HttpResponse res = new HttpResponse();

    res.setStatusCode(200);

    res.setBody('720'); // fake credit score

    return res;

}
}

```

### **Test Class Code:**

```

@Test

private class CreditBureauServiceTest {

    @isTest

    static void testCreditScoreUpdate() {

        Bank_Customer__c cust = new Bank_Customer__c(Name='Cust B',
Email__c='cust@test.com');

        insert cust;

        Loan_Application__c loan = new Loan_Application__c(

            Bank_Customer__c = cust.Id,

            Loan_Amount__c = 400000,

            Loan_Status__c = 'Submitted'

        );

        insert loan;

        // mock callout

        Test.setMock(HttpCalloutMock.class, new CreditBureauMock());

        Test.startTest();
    }
}

```

```
CreditBureauService.getCreditScore(loan.Id);
```

```
Test.stopTest();
```

```
Loan_Application__c updated = [SELECT Credit_Score__c, Loan_Status__c FROM  
Loan_Application__c WHERE Id = :loan.Id];
```

```
System.assertEquals(720, updated.Credit_Score__c);
```

```
System.assertEquals('Under_Review', updated.Loan_Status__c);
```

```
}
```

```
}
```

## Summary

- What I did: Configured a Named Credential and implemented a minimal Apex callout setup with test classes.
- Reason: This ensures Salesforce can simulate Credit Bureau integration securely and pass deployment requirements without relying on a real API.

Next step: Add screenshots of Named Credential setup and test execution for documentation.