FinSmart CRM Project – Phase 5:

Apex Programming (Step by Step)

1: Create a Simple Trigger (Loan Application Auto Status)

- Purpose: Auto-set Loan_Status__c when a loan is created.
- What I did: Created an Apex Trigger on Loan_Application__c object to automatically set status based on Loan Amount.
- Reason: Ensures loan applications are automatically categorized as 'Draft' or 'Under_Review' depending on loan amount, reducing manual work and enforcing business rules.

Code:

```
trigger LoanApplicationTrigger on Loan_Application__c (before insert) {

   for (Loan_Application__c loan : Trigger.new) {

      if (loan.Loan_Amount__c != null) {

         if (loan.Loan_Amount__c >= 500000) {

            loan.Loan_Status__c = 'Under_Review';

         } else {

            loan.Loan_Status__c = 'Draft';

         }

      }

   }

}
```

File Edit Debug Test Workspace Help

anHelperTest.apxc | CreditBureauServiceTest.apxc | OverdueRepaymentReminder.apxc | LoanApplicationTrigger.apxt

Code Coverage: None | API Version: 64 | Go

```
trigger LoanApplicationTrigger on Loan_Application__c (before insert)
    for (Loan_Application__c loan : Trigger.new) {
        if (loan.Loan_Amount__c != null) {
            if (loan.Loan_Amount__c >= 500000) {
                loan.Loan_Status__c = 'Under_Review';
            } else {
                loan.Loan_Status__c = 'Draft';
            }
        }
    }
}
```

2: Create a Simple Apex Class (Helper for EMI)

- Purpose: Calculate EMI (basic formula).
- What I did: Created an Apex Class LoanHelper to calculate EMI values.
- Reason: Provides a reusable utility to compute EMI, useful for loan simulations and repayment planning.

Code:

```
public with sharing class LoanHelper {

    public static Decimal calculateEMI(Decimal amount, Decimal annualRate, Integer months) {

        if (amount == null || annualRate == null || months == null || months <= 0) {

            return 0;

        }

        // Convert to Double for pow()

        Double principal = amount.doubleValue();

        Double monthlyRate = (annualRate.doubleValue() / 12) / 100;

        Double n = Double.valueOf(months);
```

```
        // Formula: EMI = P * r * (1+r)^n / ((1+r)^n - 1)
        Double powVal = Math.pow(1 + monthlyRate, n);
        Double emiDouble = (principal * monthlyRate * powVal) / (powVal - 1);

        // Cast back to Decimal
        Decimal emi = Decimal.valueOf(emiDouble);
        return emi.setScale(2);  // round to 2 decimal places
    }
}
```

3: Simulate Credit Bureau Integration

- Purpose: Show external API call for Credit Score.
- What I did: Created a Named Credential, Apex Service class, and Mock class for integration testing.
- Reason: Simulates integration with external systems like Credit Bureau, demonstrating how Salesforce can handle callouts, responses, and use them to update loan application status.

3.1 Apex Class to Call Credit Bureau:

```
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;
        }
    }
}
```

3.2 Mock Class for Testing:

```
@IsTest
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;
    }
}
```

4: Create Test Classes

- Purpose: Ensure >75% coverage and validate logic.
- What I did: Wrote test classes for LoanHelper and CreditBureauService to validate functionality and meet Salesforce coverage requirements.
- Reason: Test classes are mandatory in Salesforce to deploy Apex code and ensure system reliability.

LoanHelperTest.cls:

```
@isTest
```

```
private class LoanHelperTest {
    @isTest
    static void testLoanApplicationTriggerAndEMI() {
        Bank_Customer__c cust = new Bank_Customer__c(Name='Test Cust',
Email__c='test@test.com');
        insert cust;

        Loan_Application__c loan = new Loan_Application__c(
            Bank_Customer__c = cust.Id,
            Loan_Amount__c = 600000,
            Interest_Rate__c = 10,
            Tenure_Months__c = 12
        );
        insert loan; // trigger runs

        Loan_Application__c inserted = [SELECT Loan_Status__c FROM Loan_Application__c
WHERE Id=:loan.Id];
        System.assertEquals('Under_Review', inserted.Loan_Status__c);

        Decimal emi = LoanHelper.calculateEMI(100000, 12, 12);
        System.assert(emi > 0, 'EMI should be positive');
    }
}
```

CreditBureauServiceTest.cls:

```
@isTest
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);
    }
}
```

5: (Optional) Simple Scheduled Apex

- Purpose: Simple scheduled check for overdue repayments.
- What I did: Created a schedulable class that fetches overdue repayments and logs them.
- Reason: Demonstrates ability to schedule background jobs in Salesforce to monitor repayment compliance.

Code:

```
public with sharing class OverdueRepaymentReminder implements Schedulable {

   public void execute(SchedulableContext sc) {

      List<Repayment__c> overdues = [

         SELECT Id, Loan__c, Due_Amount__c

         FROM Repayment__c

         WHERE Payment_Status__c = 'Overdue'

];

      for (Repayment__c r : overdues) {

         System.debug('Overdue repayment: ' + r.Id + ' Amount: ' + r.Due_Amount__c);

      }

   }

}
```

To schedule: Setup → Apex Classes → Schedule Apex → Job Name = OverdueReminder, Class = OverdueRepaymentReminder, set time.