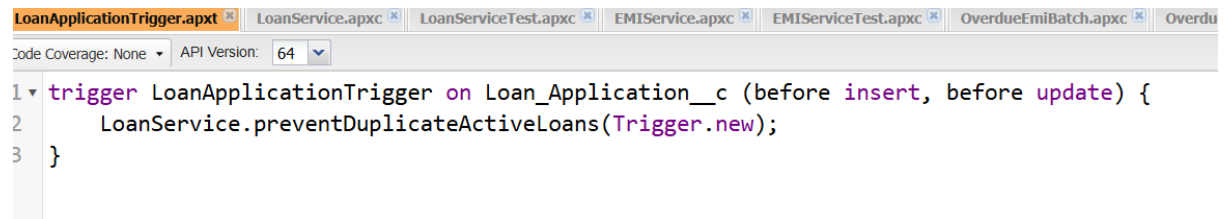# Phase 5 Apex Programming

A) Trigger & Handler – Prevent Duplicate Active Loans

- Prevents creating/updating multiple active loans for the same customer.

- Bulk-safe; handles multiple loan applications at once.

Trigger (LoanApplicationTrigger.trigger)

```
trigger LoanApplicationTrigger on Loan_Application__c (before insert, before update) {
    LoanService.preventDuplicateActiveLoans(Trigger.new);
}
```

Handler (LoanApplicationHandler.cls)

Prevents creating/updating multiple active loans for the same customer.

Bulk-safe; handles multiple loan applications at once.

```
public with sharing class LoanService {
    public static void preventDuplicateActiveLoans(List<Loan_Application__c> loans) {
        Set<Id> contactIds = new Set<Id>();
        for (Loan_Application__c l : loans) if (l.Contact__c != null) contactIds.add(l.Contact__c);

        if (contactIds.isEmpty()) return;

        // Define statuses considered "active"
        List<String> activeStatuses = new List<String>{'Pending','Approved','Disbursed'};
        Map<Id, List<Loan_Application__c>> existingMap = new Map<Id, List<Loan_Application__c>>();

        for (Loan_Application__c ex : [
            SELECT Id, Contact__c, Loan_Type__c, Status__c
            FROM Loan_Application__c
            WHERE Contact__c IN :contactIds AND Status__c IN :activeStatuses
        ]) {
            Id key = ex.Contact__c;
            if (!existingMap.containsKey(key)) existingMap.put(key, new List<Loan_Application__c>());
            existingMap.get(key).add(ex);
        }

        for (Loan_Application__c l : loans) {
            if (l.Contact__c != null && existingMap.containsKey(l.Contact__c)) {
                // Optional: Check Loan Type if needed
                for (Loan_Application__c existing : existingMap.get(l.Contact__c)) {
                    if (existing.Loan_Type__c == l.Loan_Type__c) {
                        l.addError('This contact already has an active loan of this type.');
                    }
                }
            }
        }
    }
}
```

Output:

B) EMI Generation Service

- Generates amortized EMI schedules for approved loans.

- Can be invoked via Flow or directly in Apex.

EMIService.cls


```
1   public with sharing class EMIService {
2
3       // Invocable method to be called from Flow or Apex
4       @InvocableMethod(label='Generate EMI Schedule')
5       public static void generateEMISchedules(List<Id> loanIds) {
6
7           // Step 1: Fetch Loan Applications based on the IDs
8           List<Loan_Application__c> loans = [
9               SELECT Id, Loan_Amount__c, Tenure_Months__c, Interest_Rate__c, Disbursement_Date__c
10              FROM Loan_Application__c
11              WHERE Id IN :loanIds
12          ];
13
14          // Step 2: Prepare EMI schedule records
15          List<EMI_Schedule__c> emisToInsert = new List<EMI_Schedule__c>();
16
17          for (Loan_Application__c loan : loans) {
18              if (loan.Tenure_Months__c == null || loan.Tenure_Months__c == 0) continue;
19
20              Decimal principal = loan.Loan_Amount__c;
21              Integer months = Integer.valueOf(loan.Tenure_Months__c);
22              Decimal monthlyRate = (loan.Interest_Rate__c == null) ? 0 : (loan.Interest_Rate__c / 100) / 12;
23              Decimal emiAmount;
24
25              // Step 3: EMI Calculation
26              if (monthlyRate == 0) {
27                  // Simple division if interest rate is 0
28                  emiAmount = (principal / months).setScale(2);
29              } else {
30                  // Standard EMI formula for amortized loan
31                  Double P = Double.valueOf(principal);
32                  Double r = Double.valueOf(monthlyRate);
33                  Double pow = Math.pow(1 + r, months);
34                  Double emiDouble = P * r * pow / (pow - 1);
35                  emiAmount = Decimal.valueOf(emiDouble).setScale(2);
36              }
37
38              // Step 4: Determine starting date
39              Date startDate = (loan.Disbursement_Date__c == null) ? Date.today() : loan.Disbursement_Date__c;
40
41              // Step 5: Generate EMI schedule for each month
```

Timestamp	Event	Details
16:33:38:000	CODE_UNIT_ST...	[EXTERNAL]execute_anonymous_apex
16:33:38:001	STATEMENT_EX...	[1]|
16:33:38:001	STATEMENT_EX...	[1]|
16:33:38:001	VARIABLE_ASSI...	[1]|sch|"0 0 2 * * ?"
16:33:38:001	STATEMENT_EX...	[2]|
16:33:38:003	METHOD_ENTRY	[1]|01pQy0000009w0D|OverdueEmiBatchScheduler.OverdueEmiBatchScheduler()
16:33:38:003	STATEMENT_EX...	[1]|
16:33:38:003	STATEMENT_EX...	[1]|
16:33:38:003	METHOD_EXIT	[1]|OverdueEmiBatchScheduler

EMIServiceTest.cls

```apex
@isTest
private class EMIServiceTest {

    @isTest static void testGenerateEMISchedules() {
        // Step 1: Create a sample Contact
        Contact c = new Contact(LastName='Test User', Email='testuser@example.com');
        insert c;

        // Step 2: Create a Loan Application
        Loan_Application__c loan = new Loan_Application__c(
            Contact__c = c.Id,
            Loan_Amount__c = 120000,
            Tenure_Months__c = 12,
            Interest_Rate__c = 12,
            Disbursement_Date__c = Date.today()
        );
        insert loan;

        // Step 3: Call the EMI Service
        Test.startTest();
        EMIService.generateEMISchedules(new List<Id>{loan.Id});
        Test.stopTest();

        // Step 4: Verify EMI schedule records
        List<EMI_Schedule__c> schedules = [
            SELECT Loan_Application__c, Sequence__c, EMI_Amount__c, Due_Date__c
            FROM EMI_Schedule__c
            WHERE Loan_Application__c = :loan.Id
        ];
        System.assertEquals(12, schedules.size(), 'There should be 12 EMI records.');
    }
}
```

Output:

User	Application	Operation	Time ▼	Status	Read	Size
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:10:06 PM	Success	Unread	2.53 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:09:57 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:09:17 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:08:13 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:08:10 PM	Success	Unread	1.74 KB

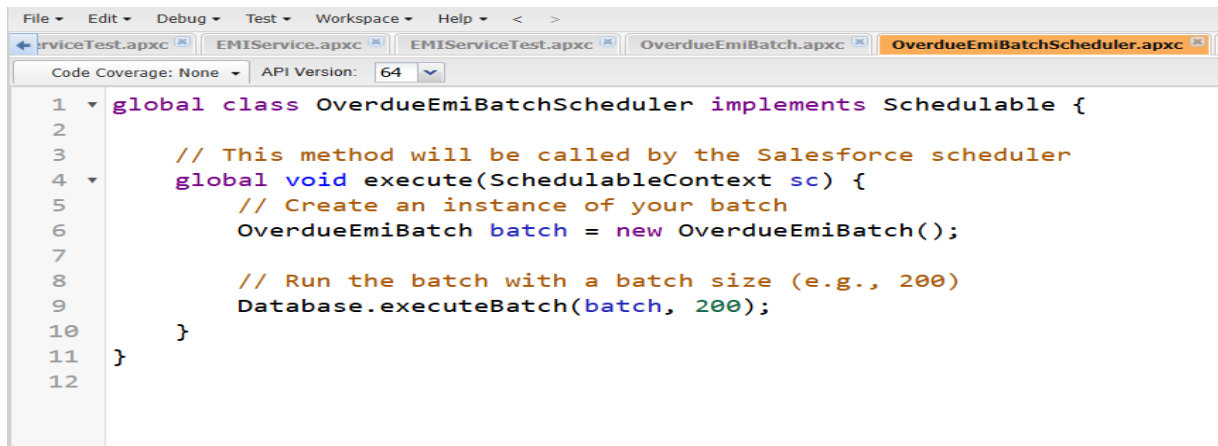| Timestamp	Event	Details
16:33:38:000	CODE_UNIT_ST...	[EXTERNAL]|execute_anonymous_apex
16:33:38:001	STATEMENT_EX...	[1]|
16:33:38:001	STATEMENT_EX...	[1]|
16:33:38:001	VARIABLE_ASSI...	[1]|schi|"0 0 2 * * ?"
16:33:38:001	STATEMENT_EX...	[2]|
16:33:38:003	METHOD_ENTRY	[1]|01pQy000009w0D|OverdueEmiBatchScheduler.OverdueEmiBatchScheduler()
16:33:38:003	STATEMENT_EX...	[1]|
16:33:38:003	STATEMENT_EX...	[1]|
16:33:38:003	METHOD_EXIT	[1]|OverdueEmiBatchScheduler
16:33:38:004	CONSTRUCTOR...	[2]|01pQy000009w0D|<init>()|OverdueEmiBatchScheduler
16:33:38:004	VARIABLE_ASSI...	[1]|this|{}|0x4b4f35
16:33:38:005	STATEMENT_EX...	[1]|
16:33:38:005	CONSTRUCTOR...	[2]|01pQy000009w0D|<init>()|OverdueEmiBatchScheduler
16:33:38:011	EXCEPTION_TH...	[2]|System.AsyncException: The Apex job named "Overdue EMIs Job" is already scheduled for execution.
16:33:38:011	FATAL_ERROR	System.AsyncException: The Apex job named "Overdue EMIs Job" is already scheduled for execution.
16:33:38:011	CODE_UNIT_FI...	execute_anonymous_apex

c) Overdue Batch – Scheduled Nightly

- Automatically marks EMIs overdue nightly.
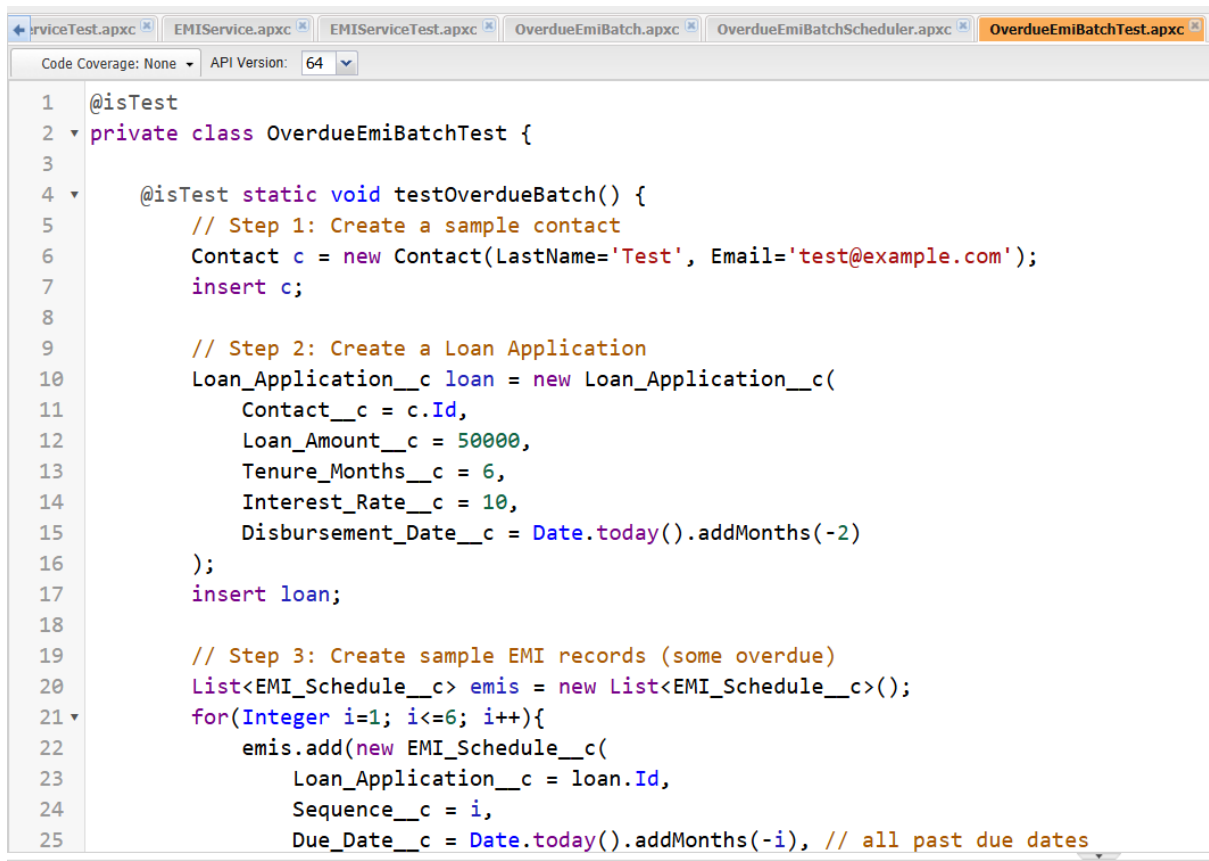- Bulk-safe and efficient.

OverdueEmiBatch.cls

```
global class OverdueEmiBatch implements Database.Batchable<sObject>, Database.Stateful {

    // Step 1: Start method - defines which records to process
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator([
            SELECT Id, Loan_Application__c, Due_Date__c, Paid_Status__c
            FROM EMI_Schedule__c
            WHERE Due_Date__c < :Date.today()
            AND Paid_Status__c = 'Pending'
            AND Overdue__c = false
        ]);
    }

    // Step 2: Execute method - process each batch of records
    global void execute(Database.BatchableContext bc, List<EMI_Schedule__c> scope) {
        List<EMI_Schedule__c> updates = new List<EMI_Schedule__c>();

        for (EMI_Schedule__c emi : scope) {
            emi.Overdue__c = true; // mark as overdue
            updates.add(emi);
        }

        if (!updates.isEmpty()) {
            update updates; // bulk update
```

Scheduler.cls

Code Coverage: None ▾   API Version: 64 ▾

```apex
1  ▾ global class OverdueEmiBatchScheduler implements Schedulable {
2
3        // This method will be called by the Salesforce scheduler
4  ▾     global void execute(SchedulableContext sc) {
5            // Create an instance of your batch
6            OverdueEmiBatch batch = new OverdueEmiBatch();
7
8            // Run the batch with a batch size (e.g., 200)
9            Database.executeBatch(batch, 200);
10       }
11 }
12
```

Test class

Code Coverage: None ▾   API Version: 64 ▾

```apex
1    @isTest
2  ▾ private class OverdueEmiBatchTest {
3
4  ▾     @isTest static void testOverdueBatch() {
5            // Step 1: Create a sample contact
6            Contact c = new Contact(LastName='Test', Email='test@example.com');
7            insert c;
8
9            // Step 2: Create a Loan Application
10           Loan_Application__c loan = new Loan_Application__c(
11               Contact__c = c.Id,
12               Loan_Amount__c = 50000,
13               Tenure_Months__c = 6,
14               Interest_Rate__c = 10,
15               Disbursement_Date__c = Date.today().addMonths(-2)
16           );
17           insert loan;
18
19           // Step 3: Create sample EMI records (some overdue)
20           List<EMI_Schedule__c> emis = new List<EMI_Schedule__c>();
21  ▾        for(Integer i=1; i<=6; i++){
22               emis.add(new EMI_Schedule__c(
23                   Loan_Application__c = loan.Id,
24                   Sequence__c = i,
25                   Due_Date__c = Date.today().addMonths(-i), // all past due dates
```

Output

User	Application	Operation	Time ▼	Status	Read	Size
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:10:06 PM	Success	Unread	2.53 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:09:57 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:09:17 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:09:13 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Browser	/aura	9/27/2025, 5:08:10 PM	Success	Unread	1.74 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first...	Unread	16.21 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first...	Unread	23.17 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first...	Unread	61.04 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first...	Unread	16.37 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first...	Unread	16.23 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first...	Unread	23.17 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first...	Unread	61 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first...	Unread	16.36 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:55:50 PM	Insert failed. First exception on row 0; first...	Unread	23.16 KB
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:55:41 PM	Insert failed. First exception on row 0; first...	Unread	16.23 KB

Test Classes

- **Covers duplicate loan prevention.**
- **Verifies correct EMI generation.**
- **Ensures bulk-safe operations.**

Code Coverage: None ▾   API Version: 64 ▾

```
1    @isTest
2  ▾ private class LoanAppTests {
3
4  ▾     @isTest static void testDuplicatePreventionAndEMIGeneration() {
5            // Create a test contact
6            Contact c = new Contact(LastName='User', Email='user@test.com');
7            insert c;
8
9            // Create a loan application
10           Loan_Application__c l1 = new Loan_Application__c(
11               Contact__c = c.Id,
12               Loan_Amount__c = 100000,
13               Tenure_Months__c = 12,
14               Interest_Rate__c = 8,
15               Status__c = 'Pending',
16               Application_Date__c = Date.today()
17           );
18           insert l1;
19
20           // Attempt a duplicate loan application for the same contact
21           Loan_Application__c l2 = new Loan_Application__c(
22               Contact__c = c.Id,
23               Loan_Amount__c = 50000,
24               Tenure_Months__c = 6,
25               Interest_Rate__c = 8,
26               Status__c = 'Pending',
27               Application_Date__c = Date.today()
28           );
29  ▾        try {
30               insert l2;
31               System.assert(false, 'Should have failed due to duplicate active loan');
32  ▾        } catch (DmlException ex) {
33               System.assert(ex.getMessage().contains('Customer already has an active loan application'));
34           }
35
36           // Approve first loan and generate EMIs
37           Test.startTest();
38           l1.Status__c = 'Approved';
```

Output:

User	Application	Operation	Time ▾	Status	Read	Size	
Kotha Sree Jyothirmai	Browser	laura	9/27/2025, 5:10:06 PM	Success	Unread	2.53 KB	
Kotha Sree Jyothirmai	Browser	laura	9/27/2025, 5:09:57 PM	Success	Unread	1.74 KB	
Kotha Sree Jyothirmai	Browser	laura	9/27/2025, 5:09:17 PM	Success	Unread	1.74 KB	
Kotha Sree Jyothirmai	Browser	laura	9/27/2025, 5:09:13 PM	Success	Unread	1.74 KB	
Kotha Sree Jyothirmai	Browser	laura	9/27/2025, 5:08:10 PM	Success	Unread	1.74 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	16.21 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	23.17 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first error: INVALID_...	Unread	61.04 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 5:05:46 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	16.37 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	16.23 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	23.17 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first error: INVALID_...	Unread	61 KB	
Kotha Sree Jyothirmai	Unknown	ApexTestHandler	9/27/2025, 4:58:04 PM	Insert failed. First exception on row 0; first error: REQUIRE...	Unread	16.36 KB	

Collections:

In Apex, collections are lists, sets, and maps. In this test class, you're using:

List<Id>

EMIService.generateEMISchedules(new List<Id>{l1.Id});

- A List collection is created with a single Loan Id (l1.Id).
- Passed to EMIService so it can bulk-generate EMI schedules for one or more loans.
- Shows bulkification best practice.

Control Statements Used

Control statements control flow of logic in your code. In this test class, you're using:

1. try-catch block

try {

   insert l2;

   System.assert(false, 'Should have failed due to duplicate active loan');

} catch (DmlException ex) {

   System.assert(ex.getMessage().contains('Customer already has an active loan application'));

}

- Purpose: Handle the expected DML exception.
- try → attempts to insert a duplicate loan.
- catch → catches the DmlException when trigger blocks it.
- Ensures negative testing is validated.