SkillSync – Phase 5: Apex Programming (Developer)

Project Title: SkillSync- Knowledge and Expertise Management

Phase 5: Apex Programming

Date: 25 September 2025

Prepared By: SHARANYA LAKSHMI S N

Introduction

In this phase, Apex programming is used to extend Salesforce beyond declarative (point-and-click) automation. While Process Builder and Flow are powerful, complex business logic often requires **Apex triggers, classes, and asynchronous processing**. This ensures **SkillSync** can handle bulk operations, complex calculations, and integrations efficiently.

---

1. Classes & Objects

- Apex Classes define reusable logic, similar to Java classes.
- Objects represent Salesforce records or custom objects.
- **Example in SkillSync:**
  - SkillPointsManager.cls class calculates points when employees complete tasks or mentorship programs.
  - A method assignPoints(EmployeeId, Points) updates employee engagement scores.

SkillPointsManager.cls :

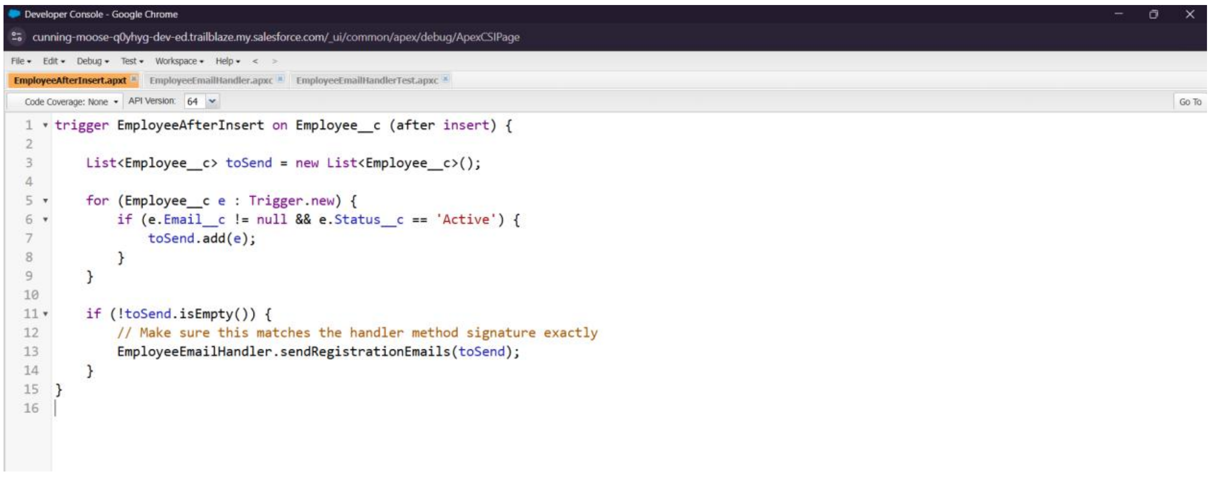
```
public class SkillPointsManager {

  // Method to award points to an employee
  public static void assignPoints(Id employeeId, Decimal points) {
    if(employeeId == null || points == null) {
      return; // avoid null pointer
    }
```

```
        KEM_Employee__c emp = [SELECT Id, Points__c, Engagement_Score__c
                               FROM KEM_Employee__c
                               WHERE Id = :employeeId
                               LIMIT 1];

        emp.Points__c = (emp.Points__c == null ? 0 : emp.Points__c) + points;
        emp.Engagement_Score__c = (emp.Engagement_Score__c == null ? 0 :
emp.Engagement_Score__c) + points;

        update emp;
    }
}
```

---

2. Apex Triggers (before/after insert/update/delete)

- Triggers execute custom logic when records are created, updated, or deleted.
- **Example in SkillSync:**
    - Trigger on Engagement_Log__c after insert → updates Employee__c's Engagement_Score__c.
    - Before insert trigger ensures no duplicate mentorship request between the same mentor and mentee.


Developer Console - Google Chrome
cunning-moose-q0yhyg-dev-ed.trailblaze.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File
Edit
Debug
Test
Workspace
Help
EmployeeAfterInsert.apxt
EmployeeEmailHandler.apxc
EmployeeEmailHandlerTest.apxc
Code Coverage: None
API Version: 64
Go To

```
trigger EmployeeAfterInsert on Employee__c (after insert) {

    List<Employee__c> toSend = new List<Employee__c>();

    for (Employee__c e : Trigger.new) {
        if (e.Email__c != null && e.Status__c == 'Active') {
            toSend.add(e);
        }
    }

    if (!toSend.isEmpty()) {
        // Make sure this matches the handler method signature exactly
        EmployeeEmailHandler.sendRegistrationEmails(toSend);
    }
}
```

---

3. Trigger Design Pattern

- Instead of writing multiple triggers on the same object, a **handler class** is used.
- **Example in SkillSync:**
  - EmployeeTriggerHandler.cls manages logic for different Employee events (onboarding, points, validations).
  - Ensures scalability and prevents recursive calls.

EmployeeTriggerHandler.cls:

```
public class EngagementTriggerHandler {


  public static void afterInsert(List<KEM_Engagement__c> newEngagements) {
    Set<Id> empIds = new Set<Id>();


    for(KEM_Engagement__c eng : newEngagements) {
      if(eng.Employee__c != null && eng.Points_Awarded__c != null) {
        SkillPointsManager.assignPoints(eng.Employee__c, eng.Points_Awarded__c);
        empIds.add(eng.Employee__c);
      }
    }
  }
}
```

---

4. SOQL & SOSL

- **SOQL (Salesforce Object Query Language):** Used to query records.
- **SOSL (Salesforce Object Search Language):** Used for searching across multiple objects.
- **Example in SkillSync:**
  - SOQL: Get all projects assigned to an employee.

    List<Project__c> projects = [SELECT Name, Status__c FROM Project__c WHERE Employee__c = :empId];

- SOSL: Search employees by skill keywords.

  List<List<SObject>> results = [FIND 'AI' IN ALL FIELDS RETURNING Employee__c(Name, Skills__c)];

---

5. Collections: List, Set, Map

- **List:** Ordered collection of records.

  List<Employee__c> toSend = new List<Employee__c>();

- **Set:** Unique values, prevents duplicates.
- **Map:** Key-value pairs.
- **Example in SkillSync:**
  - List of all mentors for a program.
  - Set of unique skills employees possess.
  - Map<EmployeeId, EngagementScore> for leaderboard calculation.

---

6. Control Statements

- Apex supports **if-else, loops, switch, break, continue**.
- **Example in SkillSync:**
  - Loop through all employees in a project and send notifications.
  - Conditional check: If Employee Engagement_Score__c > 1000 → upgrade level to *Expert*.

For Loop:

```
for (Employee__c e : Trigger.new) {

    if (e.Email__c != null && e.Status__c == 'Active') {

      toSend.add(e);

    }
```

---

7. Batch Apex (Optional)

- Used for processing large volumes of records asynchronously.
- **Example in SkillSync:**

    - Batch job recalculates Engagement_Score__c for all employees at month-end.
    - Efficiently processes thousands of records in chunks.

LeaderBoardBatch.cls:

```
global class LeaderboardBatch implements Database.Batchable<sObject> {

  global Database.QueryLocator start(Database.BatchableContext bc) {
    return Database.getQueryLocator('SELECT Id, Points__c, Engagement_Score__c 
FROM KEM_Employee__c');
  }

  global void execute(Database.BatchableContext bc, List<KEM_Employee__c> scope) {
    for(KEM_Employee__c emp : scope) {
      emp.Engagement_Score__c = emp.Points__c; // recalculating engagement
    }
    update scope;
  }

  global void finish(Database.BatchableContext bc) {
    System.debug('Leaderboard scores updated.');
  }
}
```

---

8. Queueable Apex

- Similar to Batch Apex but more flexible and allows chaining.
- **Example in SkillSync:**
    - When a new project is created, a queueable job assigns default tasks to employees.
    - Jobs can be chained to handle multi-step onboarding.

---

9. Scheduled Apex

- Runs Apex at a specific time or interval.
- **Example in SkillSync:**
    - Weekly job sends reminders for incomplete learning paths.
    - Monthly job publishes leaderboard updates to employees.

---

10. Future Methods

- Runs asynchronous operations that don't need immediate execution.
- **Example in SkillSync:**
    - Send email alerts to mentors when a mentee joins → handled asynchronously to avoid delay in record save.

---

11. Exception Handling

- Try-Catch-Finally ensures graceful error handling.
- **Example in SkillSync:**
    - If points calculation fails due to missing Employee record, catch exception and log error instead of crashing the transaction.

---

12. Test Classes

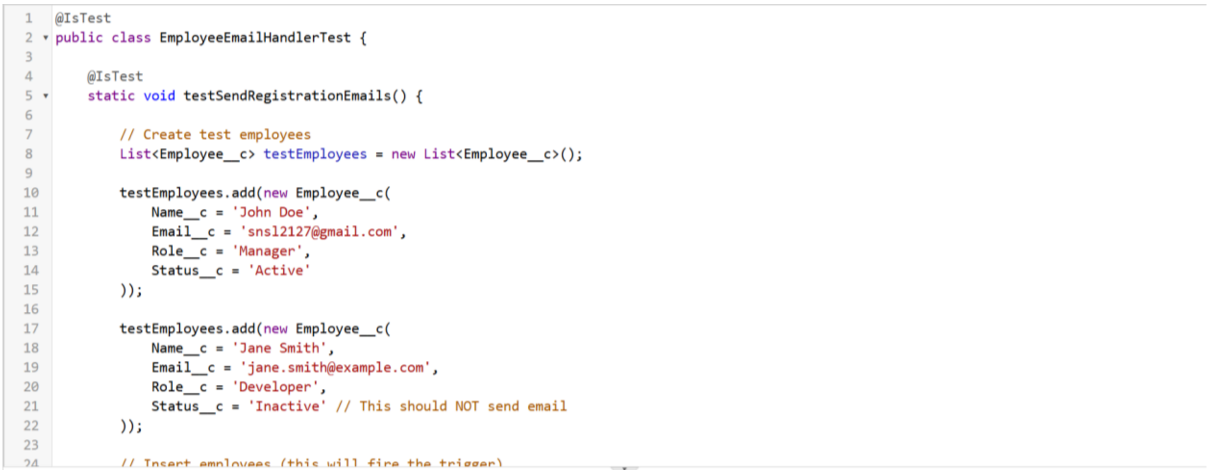
- Apex requires at least **75% code coverage** for deployment.
- Test classes validate logic with test data.
- **Example in SkillSync:**
    - SkillPointsManagerTest.cls creates test employees, simulates mentorship completions, and verifies engagement score updates correctly.

SkillPointsManagerTest.cls:

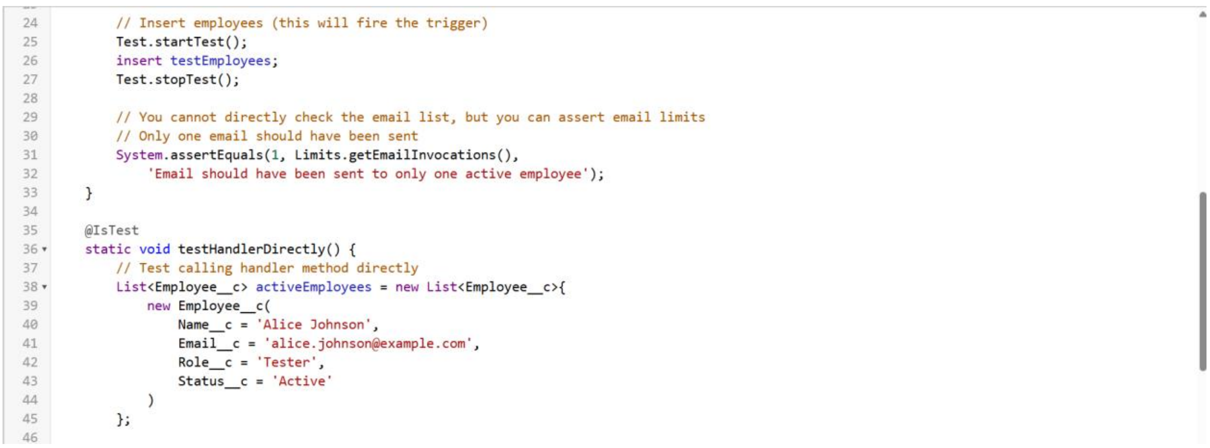
@isTest

public class SkillPointsManagerTest {

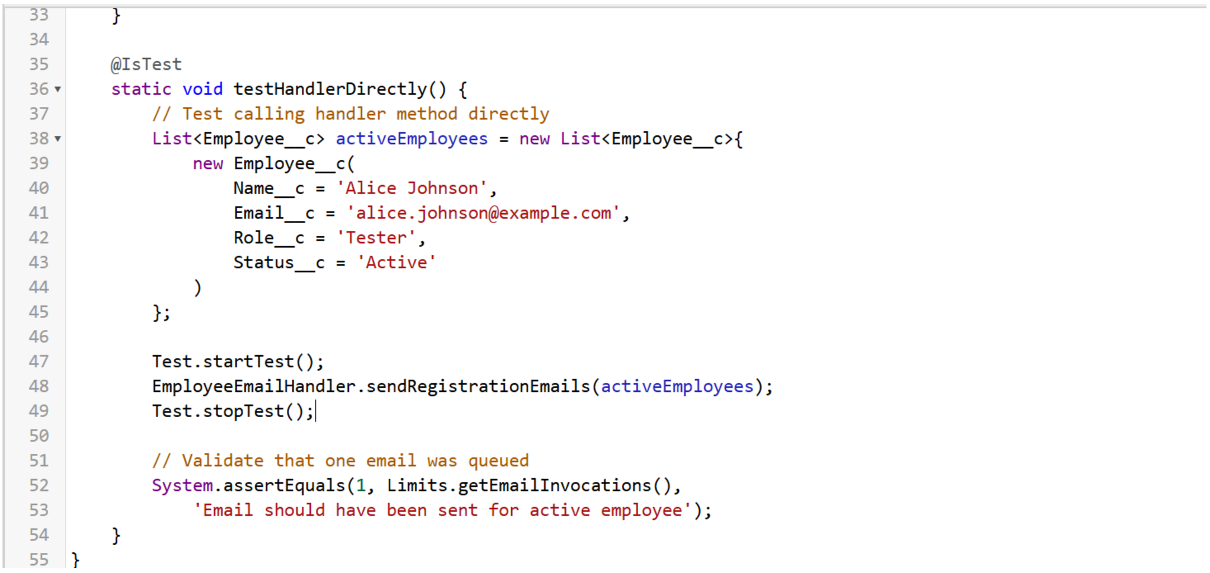
```
  @isTest
  static void testAssignPoints() {
    KEM_Employee__c emp = new KEM_Employee__c(
      Name = 'Test User',
      Email__c = 'test@example.com',
      Points__c = 0,
      Engagement_Score__c = 0
    );
    insert emp;

    Test.startTest();
    SkillPointsManager.assignPoints(emp.Id, 50);
    Test.stopTest();

    emp = [SELECT Points__c, Engagement_Score__c FROM KEM_Employee__c WHERE Id = :emp.Id];
    System.assertEquals(50, emp.Points__c);
    System.assertEquals(50, emp.Engagement_Score__c);
  }
}
```



```
@IsTest
public class EmployeeEmailHandlerTest {

    @IsTest
    static void testSendRegistrationEmails() {

        // Create test employees
        List<Employee__c> testEmployees = new List<Employee__c>();

        testEmployees.add(new Employee__c(
            Name__c = 'John Doe',
            Email__c = 'sns12127@gmail.com',
            Role__c = 'Manager',
            Status__c = 'Active'
        ));

        testEmployees.add(new Employee__c(
            Name__c = 'Jane Smith',
            Email__c = 'jane.smith@example.com',
            Role__c = 'Developer',
            Status__c = 'Inactive' // This should NOT send email
        ));

        // Insert employees (this will fire the trigger)
```



```
        // Insert employees (this will fire the trigger)
        Test.startTest();
        insert testEmployees;
        Test.stopTest();

        // You cannot directly check the email list, but you can assert email limits
        // Only one email should have been sent
        System.assertEquals(1, Limits.getEmailInvocations(),
            'Email should have been sent to only one active employee');
    }

    @IsTest
    static void testHandlerDirectly() {
        // Test calling handler method directly
        List<Employee__c> activeEmployees = new List<Employee__c>{
            new Employee__c(
                Name__c = 'Alice Johnson',
                Email__c = 'alice.johnson@example.com',
                Role__c = 'Tester',
                Status__c = 'Active'
            )
        };

```



```
    }

    @IsTest
    static void testHandlerDirectly() {
        // Test calling handler method directly
        List<Employee__c> activeEmployees = new List<Employee__c>{
            new Employee__c(
                Name__c = 'Alice Johnson',
                Email__c = 'alice.johnson@example.com',
                Role__c = 'Tester',
                Status__c = 'Active'
            )
        };

        Test.startTest();
        EmployeeEmailHandler.sendRegistrationEmails(activeEmployees);
        Test.stopTest();

        // Validate that one email was queued
        System.assertEquals(1, Limits.getEmailInvocations(),
            'Email should have been sent for active employee');
    }
}
```

---

13. Asynchronous Processing

- Includes **Batch Apex, Queueable, Future Methods, Scheduled Apex**.
- Ensures long-running operations don't block users.
- **Example in SkillSync:**
    - Bulk recalculation of engagement scores runs asynchronously at night to avoid slowing the system.