



Phase 5: Apex Programming (Developer)

🎯 Objective

Implement custom Apex code to handle advanced automation that cannot be achieved using point-and-click tools like Flows and Process Builder.

1. Apex Trigger – Auto-Create Enrollment

Problem:

Sales executives were forgetting to manually create **Enrollment records** after closing Opportunities, which led to missing student enrollment data.

Solution:

Created an **Apex Trigger** on the **Opportunity** object to automatically generate an **Enrollment__c** record whenever an Opportunity is marked as *Closed Won*.

Configuration (Trigger Code):

```
trigger CreateEnrollment on Opportunity (after update) {
    for (Opportunity opp : Trigger.new) {
        Opportunity oldOpp = Trigger.oldMap.get(opp.Id);
        if (opp.StageName == 'Closed Won' && oldOpp.StageName != 'Closed Won') {
            Enrollment__c enroll = new Enrollment__c(
                Student__c = opp.ContactId,
                Course__c = opp.Course__c,
                Payment_Status__c = 'Pending'
            );
            insert enroll;
        }
    }
}
```

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar with 'apex' typed in. Under 'Custom Code', 'Apex Classes' is selected. The main area shows the 'Apex Classes' page with a 'SETUP' button and a 'Apex Classes' title. Below is a table with tabs for 'Class Body', 'Class Summary', 'Version Settings', and 'Trace Flags'. The 'Class Body' tab displays the Apex trigger code shown in the previous code block. The code handles updates on the Opportunity object, specifically for opportunities marked as 'Closed Won'. It retrieves the old opportunity from the trigger map, creates a new Enrollment__c record with the student ID, course ID, and payment status, and inserts it into the database.

Line	Code
1	public class OpportunityTriggerHandler {
2	public static void handleAfterUpdate(List<Opportunity> newList, Map<Id, Opportunity> oldMap) {
3	List<Enrollment__c> enrollmentsToInsert = new List<Enrollment__c>();
4	
5	for (Opportunity opp : newList) {
6	Opportunity oldOpp = oldMap.get(opp.Id);
7	// Only when Stage moves to Closed Won
8	if (opp.StageName == 'Closed Won' && oldOpp.StageName != 'Closed Won') {
9	if (opp.ContactId != null && opp.Course__c != null) {
10	Enrollment__c enroll = new Enrollment__c(
11	Student__c = opp.ContactId,
12	Course__c = opp.Course__c,
13	Payment_Status__c = 'Pending'
14);
15	enrollmentsToInsert.add(enroll);
16	}
17	}
18	}
19	if (enrollmentsToInsert.isEmpty()) {
20	insert enrollmentsToInsert;
21	}
22	}
23	}
24	}
25	}

Outcome:

- Every time an Opportunity is closed successfully, the system auto-creates the related Enrollment.
- Prevents missed records and saves manual effort.

2. Apex Class – Course Recommendation

Problem:

Sales reps struggled to suggest the right courses to students during demo sessions.

Solution:

Developed an **Apex Class** to recommend top 3 courses based on the student's interest (AI, ML, Cloud, etc.).

Configuration (Class Code):

```
public with sharing class CourseRecommender {
    public static List<Course__c> suggestCourses(String interest) {
        return [
            SELECT Id, Name, Category__c, Price__c, Duration__c
            FROM Course__c
            WHERE Category__c = :interest
            ORDER BY CreatedDate DESC
            LIMIT 3
        ];
    }
}
```

The screenshot shows the Salesforce Setup interface. In the left sidebar, under 'Custom Code', 'Apex Classes' is selected. The main content area displays the details for the 'CourseRecommenderTest' Apex Class. The 'Apex Class Detail' section shows the class name as 'CourseRecommenderTest', status as 'Active', and last modified by 'Shaik Mohammed Kaif' on 9/25/2025, 10:26 AM. The 'Class Body' tab is selected, showing the following Apex code:

```
1  @isTest
2  private class CourseRecommenderTest {
3
4      @testSetup
5      static void makeData(){
6          List<Course__c> coursesToCreate = new List<Course__c>();
7
8          // Create 3 courses in the 'AI' category
9          coursesToCreate.add(new Course__c(Name='AI Intro', Category__c='AI', Price__c=100, Duration__c=10));
10         coursesToCreate.add(new Course__c(Name='AI Advanced', Category__c='AI', Price__c=200, Duration__c=10));
11         coursesToCreate.add(new Course__c(Name='AI Expert', Category__c='AI', Price__c=300, Duration__c=10));
12
13         // Create 2 courses in the 'Cloud' category
14 }
```

Outcome:

- Provides personalized recommendations.
- Can be called from Lightning Components, Flows, or custom pages.

3. Test Class – Code Coverage

Problem:

Salesforce requires **75% Apex code coverage** before deploying to Production.

Solution:

Built a **Test Class** to validate trigger functionality.

Configuration (Test Class Example):

```
@isTest
public class TestEnrollmentTrigger {
    @isTest static void testClosedWonEnrollment() {
        Account acc = new Account(Name='Test Student');
        insert acc;

        Contact con = new Contact(LastName='Student', AccountId=acc.Id);
        insert con;

        Course__c course = new Course__c(
            Name='AI Basics',
            Category__c='AI',
            Price__c=20000,
            Duration__c='3 Months'
        );
        insert course;
    }
}
```

```

Opportunity opp = new Opportunity(
    Name='Test Opp',
    StageName='Prospecting',
    CloseDate=Date.today().addDays(10),
    ContactId=con.Id,
    Course__c=course.Id,
    Amount=20000
);
insert opp;

opp.StageName = 'Closed Won';
update opp;

List<Enrollment__c> enrollments =
    [SELECT Id FROM Enrollment__c WHERE Student__c=:con.Id];
System.assertEquals(1, enrollments.size());
}
}

```

Outcome:

- Ensures trigger runs correctly.
- Achieves >75% test coverage for deployment.

Phase 5 Outcome

- Apex Trigger automates enrollment creation on Opportunity closure.
- Apex Class provides personalized course recommendations.
- Test Class ensures code reliability and satisfies Salesforce deployment rules.