📑 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.

---

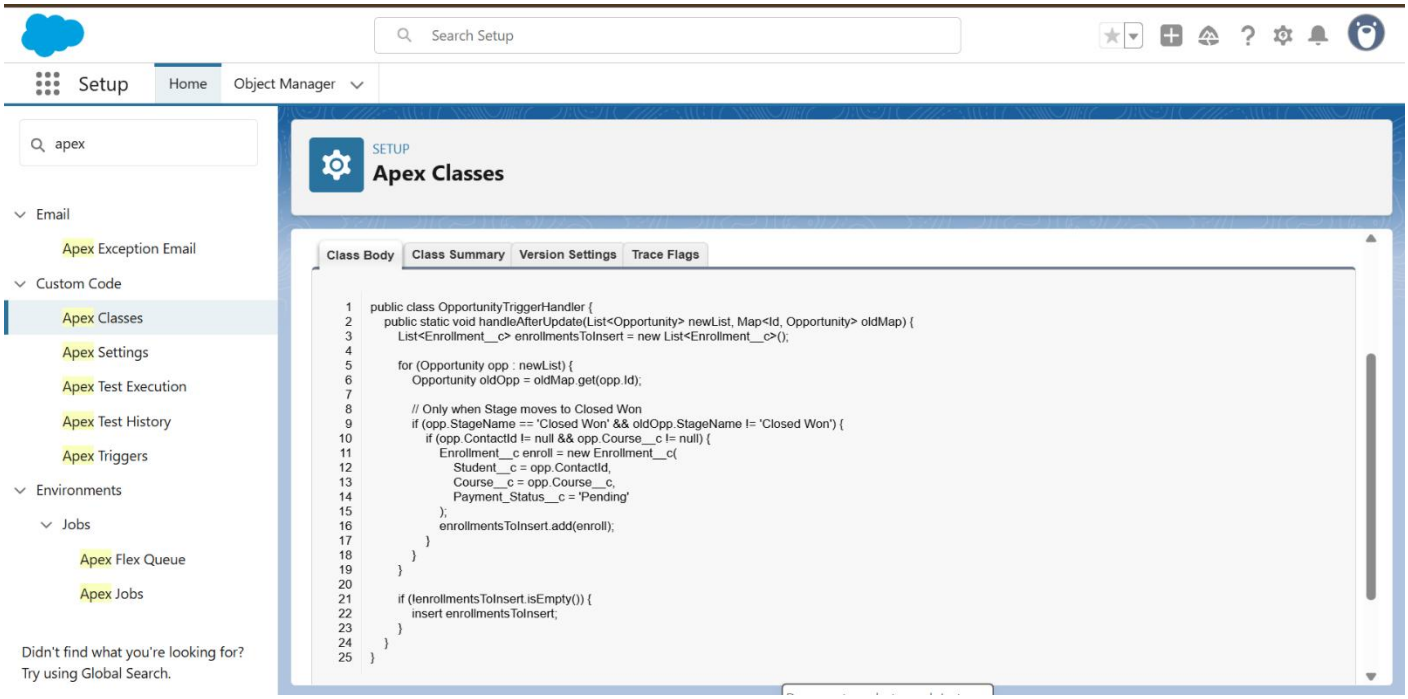
1. Apex Trigger – Auto-Create Enrollment

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;
        }
    }
}
```



Outcome:

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

---

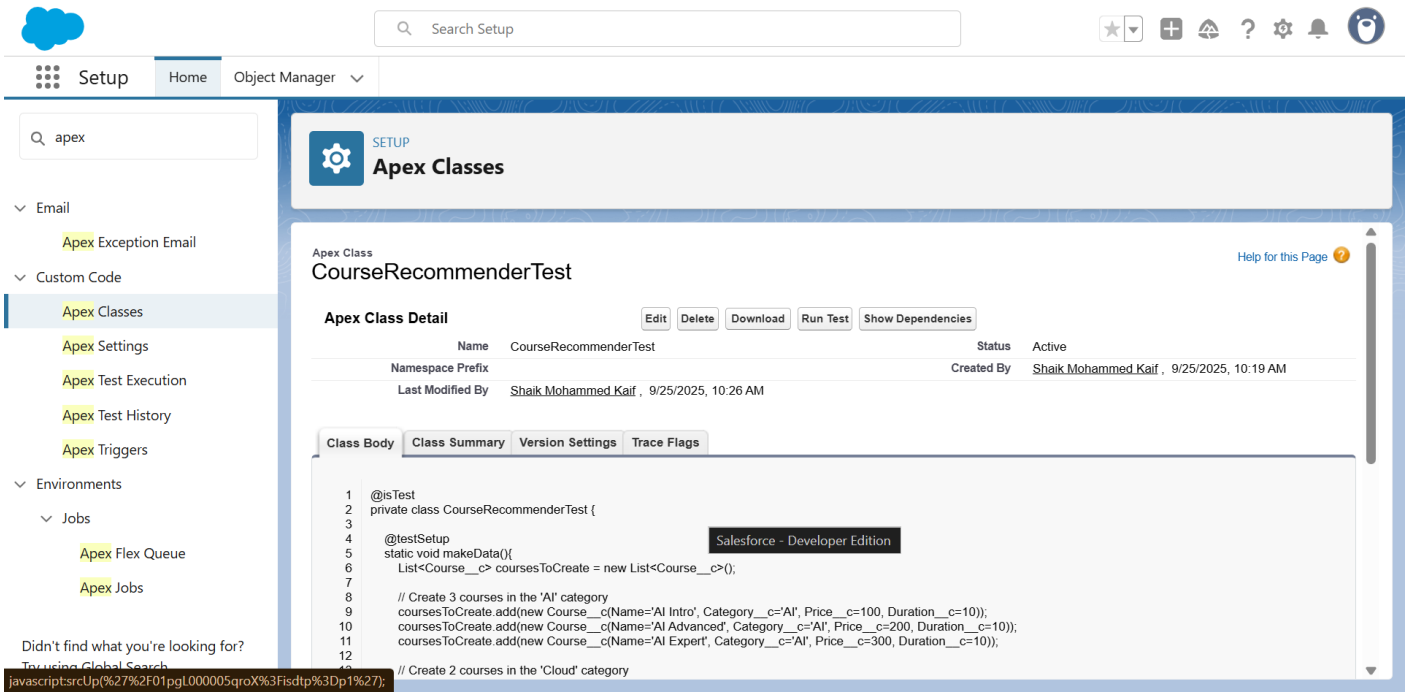
2. Apex Class – Course Recommendation

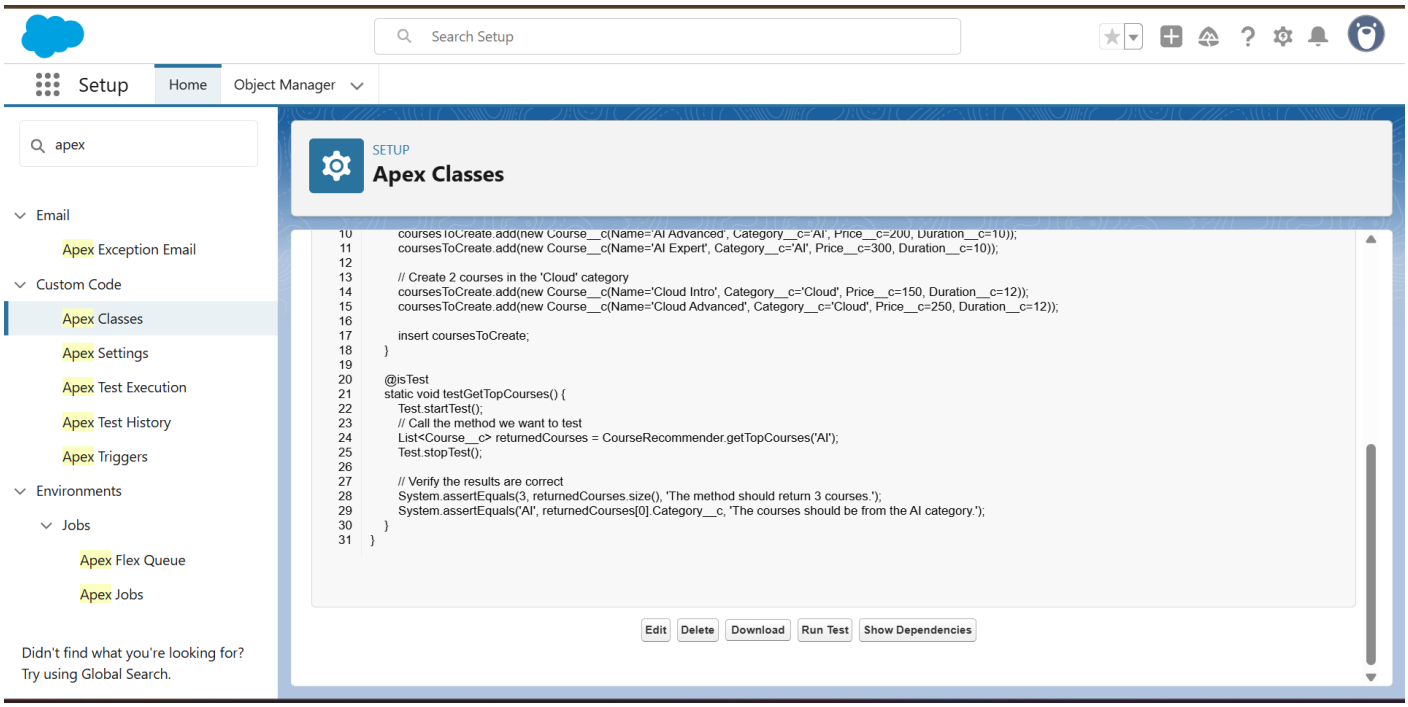
Problem:
Sales reps struggled to suggest the right courses to students during demo sessions.

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
        ];
    }
}
```





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.