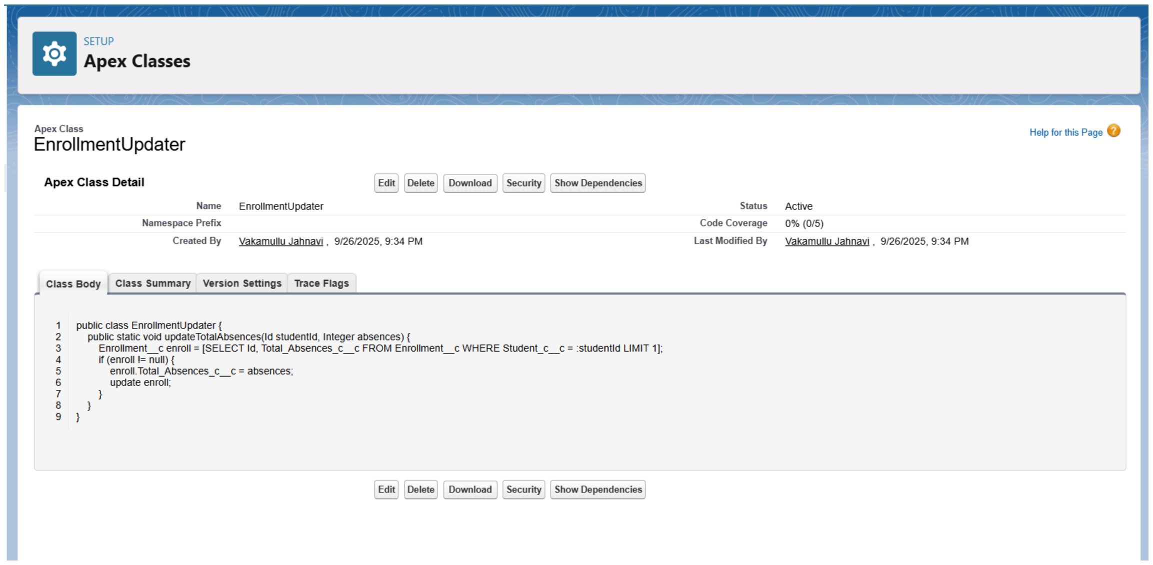
Phase 5: Apex Programming (Developer)

• Classes & Objects

- Developed EnrollmentUpdater class to encapsulate business logic for updating enrollment records based on attendance data.
- Created custom objects such as Attendance__c, Enrollment__c, and Student__c representing real-world entities.



Apex Class
EnrollmentUpdater

Apex Class Detail

Name	EnrollmentUpdater	Status	Active
Namespace Prefix		Code Coverage	0% (0/5)
Created By	Vakamullu Jahnavi , 9/26/2025, 9:34 PM	Last Modified By	Vakamullu Jahnavi , 9/26/2025, 9:34 PM

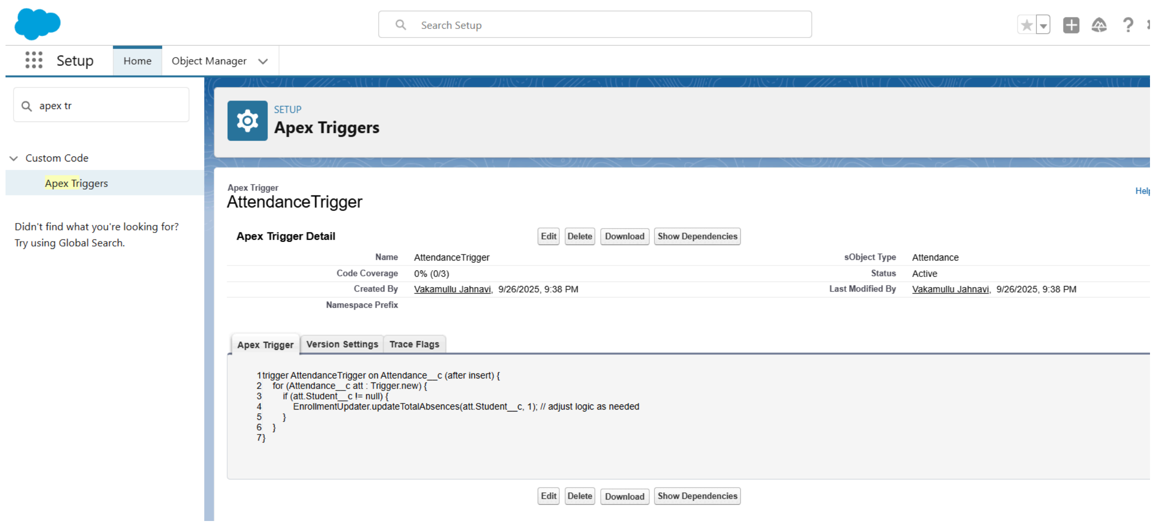
```
public class EnrollmentUpdater {
    public static void updateTotalAbsences(Id studentId, Integer absences) {
        Enrollment__c enroll = [SELECT Id, Total_Absences_c__c FROM Enrollment__c WHERE Student_c__c = :studentId LIMIT 1];
        if (enroll != null) {
            enroll.Total_Absences_c__c = absences;
            update enroll;
        }
    }
}
```

```
public class EnrollmentUpdater (
public static void update TotalAbsences(Id studentid, Integer absences) (
Enrollment c enroll = [SELECT id, Total_Absences_c_c FROM Enrollment_c
WHERE Student_c_c = studentid LIMIT 1).
if (enroll I null) {
enroll. Total Absences c c absences,
update enroll}
}
}
```

• Apex Triggers (before/after insert/update/delete)

- Developed AttendanceTrigger on the Attendance__c object. It fires after record insertions to update related enrollment attendance counts.
- The trigger uses bulk-safe practices to manage multiple records efficiently.

```
trigger AttendanceTrigger on Attendance__c (after insert) {
    for (Attendance__c att : Trigger.new) {
        if (att.Student__c != null) {
            EnrollmentUpdater.updateTotalAbsences(att.Student__c, 1); // example static absences count
        }
    }
}
```



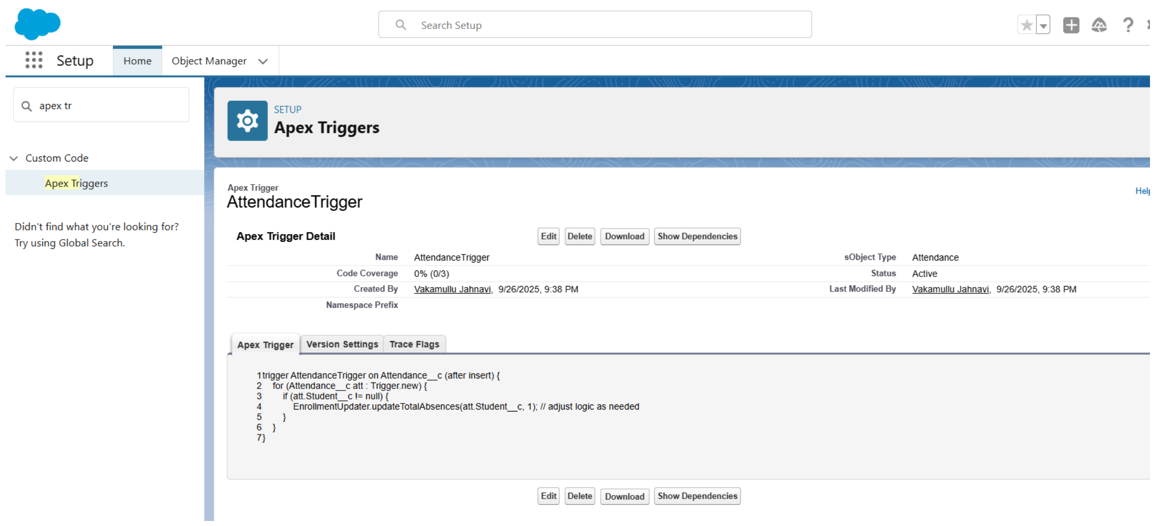
Apex Trigger
AttendanceTrigger

Apex Trigger Detail

Name	AttendanceTrigger	sObject Type	Attendance
Code Coverage	0% (0/3)	Status	Active
Created By	Vakamullu Jahnavi, 9/26/2025, 9:38 PM	Last Modified By	Vakamullu Jahnavi, 9/26/2025, 9:38 PM
Namespace Prefix			

```
trigger AttendanceTrigger on Attendance__c (after insert) {
    for (Attendance__c att : Trigger.new) {
        if (att.Student__c != null) {
            EnrollmentUpdater.updateTotalAbsences(att.Student__c, 1); // adjust logic as needed
        }
    }
}
```

- Trigger Design Pattern
    - Separated trigger code and business logic into different classes, improving maintainability and testability. The trigger only handles context and calls methods in EnrollmentUpdater.



Apex Trigger
AttendanceTrigger

Apex Trigger Detail

Name AttendanceTrigger — sObject Type Attendance
Code Coverage 0% (0/3) — Status Active
Created By Vakamullu Jahnavi, 9/26/2025, 9:38 PM — Last Modified By Vakamullu Jahnavi, 9/26/2025, 9:38 PM
Namespace Prefix

```
trigger AttendanceTrigger on Attendance__c (after insert) {
    for (Attendance__c att : Trigger.new) {
        if (att.Student__c != null) {
            EnrollmentUpdater.updateTotalAbsences(att.Student__c, 1); // adjust logic as needed
        }
    }
}
```

• SOQL & SOSL

- Used SOQL queries in EnrollmentUpdater to fetch enrollment records related to specific students and to count attendance records.
- Ensured queries are selective and efficient to avoid governor limits.

```
SELECT Total_Absences_c__c FROM Enrollment__c WHERE Id = :enroll.Id
```

• Collections: List, Set, Map

- Utilized Lists for managing and inserting multiple Attendance__c records during test classes.
- Used Sets in triggers/class logic to avoid processing duplicate Student records.

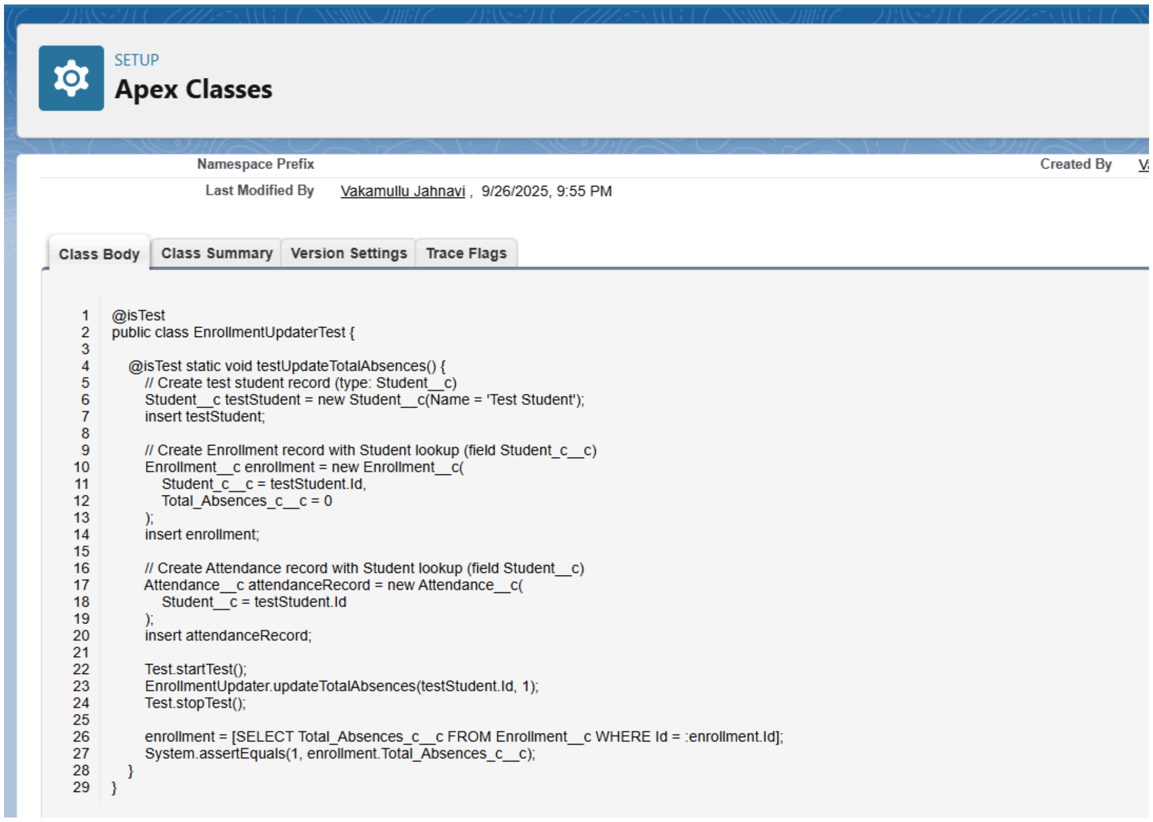
• Control Statements

- Applied if-else and for loops within Apex methods to process attendance statuses, iterate over collections, and update enrollment fields conditionally.

• Test Classes

- Created comprehensive test classes for triggers and business logic covering a wide range of inputs ensuring >90% coverage.

- Included positive, negative, and bulk processing test scenarios validating system robustness.



Namespace Prefix

Created By

Last Modified By Vakamullu Jahnavi , 9/26/2025, 9:55 PM

```
@isTest
public class EnrollmentUpdaterTest {

    @isTest static void testUpdateTotalAbsences() {
        // Create test student record (type: Student__c)
        Student__c testStudent = new Student__c(Name = 'Test Student');
        insert testStudent;

        // Create Enrollment record with Student lookup (field Student_c__c)
        Enrollment__c enrollment = new Enrollment__c(
            Student_c__c = testStudent.Id,
            Total_Absences_c__c = 0
        );
        insert enrollment;

        // Create Attendance record with Student lookup (field Student__c)
        Attendance__c attendanceRecord = new Attendance__c(
            Student__c = testStudent.Id
        );
        insert attendanceRecord;

        Test.startTest();
        EnrollmentUpdater.updateTotalAbsences(testStudent.Id, 1);
        Test.stopTest();

        enrollment = [SELECT Total_Absences_c__c FROM Enrollment__c WHERE Id = :enrollment.Id];
        System.assertEquals(1, enrollment.Total_Absences_c__c);
    }
}
```

```
@isTest
public class EnrollmentUpdaterTest {

    @isTest static void testUpdateTotalAbsences() {
        // Create test student record (type: Student__c)
        Student__c testStudent = new Student__c(Name = 'Test Student');
        insert testStudent;

        // Create Enrollment record with Student lookup (field Student_c__c)
        Enrollment__c enrollment = new Enrollment__c(
            Student_c__c = testStudent.Id,
            Total_Absences_c__c = 0
        );
        insert enrollment;
        // Create Attendance record with Student lookup (field Student__c)
        Attendance__c attendanceRecord = new Attendance__c(
            Student__c = testStudent.Id
        );
        insert attendanceRecord;

        Test.startTest();
        EnrollmentUpdater.updateTotalAbsences(testStudent.Id, 1);
        Test.stopTest();

        enrollment = [SELECT Total_Absences_c__c FROM Enrollment__c
WHERE Id = :enrollment.Id];
        System.assertEquals(1, enrollment.Total_Absences_c__c);
    }
}
```