

Student Attendance & Performance Tracker

PHASE 5 – APEX PROGRAMMING (Developer)

Introduction

Phase 5 focuses on implementing **server-side automation using Apex programming**.

Apex is used to handle complex business logic that cannot be achieved using declarative tools alone. In this phase, Apex classes, triggers, SOQL queries, collections, and test classes were developed to **automatically track student attendance and update enrollment records accurately**.

1. Classes & Objects

Use Case

Whenever a student is marked **Absent**, the system should automatically increase the **Total Absences** count in the corresponding **Enrollment** record for that specific course.

To achieve this, a dedicated Apex class was created to handle enrollment updates in a reusable and scalable manner.

Implementation

- Created an Apex class named **EnrollmentUpdater**
- The class encapsulates business logic to:
 - Fetch Enrollment records based on **Student + Course**
 - Update or create Enrollment records if required
 - Increment or decrement total absences correctly
- Mandatory fields like **Course__c** are handled to avoid runtime errors



SETUP

Apex Classes

Apex Classes

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

**Percent of Apex Used: 0.08%**

You are currently using 4,723 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex C

Estimate your organization's code coverage [i](#)

Compile all classes [i](#)

View: [All](#) [Create New View](#)

A B C D E F G H I J K L M N O P Q R S T

		Developer Console	New	Generate from WSDL	Run All Tests	Schedule Apex
Action	Name ↑	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By
Edit Del Security	AttendanceController		65.0	Active	375	Siri varshini Pentakota , 18/12/2025, 1:09 pm
Edit Del Security	EnrollmentUpdater		65.0	Active	2,008	Siri varshini Pentakota , 20/12/2025, 8:33 pm
Edit Del	EnrollmentUpdaterTest		65.0	Active	891	Siri varshini Pentakota , 13/12/2025, 1:51 pm

```
List<Enrollment__c> toUpdate = new List<Enrollment__c>();
```

```
for (Id stuId : studentAbsenceIncrements.keySet()) {
```

```
    Integer increment = studentAbsenceIncrements.get(stuId);
```

```
    if (studentToEnrollment.containsKey(stuId)) {
```

```
        Enrollment__c en = studentToEnrollment.get(stuId);
```

```
        Integer current = (en.Total_Absences__c == null) ? 0 : Integer.valueOf(en.Total_Absences__c);
```

```
        en.Total_Absences__c = current + (increment == null ? 0 : increment);
```

```
        toUpdate.add(en);
```

```
    } else {
```

```
        Enrollment__c newEn = new Enrollment__c(
```

```
            Student__c = stuId,
```

```
            Course__c = courseId,
```

```
            Total_Absences__c = (increment == null ? 0 : increment)
```

```
        );
```

```
        toUpdate.add(newEn);
```

```
    }
```

```
}
```

2. Apex Triggers (After Insert & After Update)

Use Case

Whenever an Attendance record is:

- Inserted with status **Absent**
- Updated from **Present** → **Absent** or **Absent** → **Present**

The system should automatically recalculate the total absences in Enrollment records.

Implementation

- Created **AttendanceTrigger** on Attendance__c object
- Trigger executes **after insert and after update**
- Collects student-wise and course-wise absence changes
- Sends data to the EnrollmentUpdater class
- Fully bulk-safe and optimized

```
1 trigger AttendanceTrigger on Attendance__c (after insert, after update) {
2
3     Map<Id, Map<Id, Integer>> courseStudentAbsenceMap = new Map<Id, Map<Id, Integer>>();
4
5     if (Trigger.isAfter) {
6
7         if (Trigger.isInsert) {
8             for (Attendance__c a : Trigger.new) {
9                 if (
10                     a.Student__c != null &&
11                     a.Course__c != null &&
12                     a.Status__c == 'Absent'
13                 ) {
14                     if (!courseStudentAbsenceMap.containsKey(a.Course__c)) {
15                         courseStudentAbsenceMap.put(a.Course__c, new Map<Id, Integer>());
16                     }
17
18                     Map<Id, Integer> studentMap = courseStudentAbsenceMap.get(a.Course__c);
19                     Integer curr = studentMap.containsKey(a.Student__c) ? studentMap.get(a.Stu
```

User	Application	Operation	Time	Status	Read	Size
Siri varshini Pentakota	Unknown	ApexTestHandler	20/12/2025, 21:13:37	Success	Unread	1020 bytes

SETUP

Apex Triggers

Apex Triggers

This page allows you to view and modify all the triggers in your organization. To create a new trigger, navigate to the appropriate sObject triggers page.

Percent of Apex Used: 0.08%

You are currently using 4,723 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

Compile all Triggers

View: All Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M

Developer Console

Action	Name	Namespace Prefix	sObject Type	Api Version	Status	Size Without Comments	Last Modified By
Edit Del	AttendanceTrigger		Attendance	65.0	Active	2,340	Siri vanshini Pentakota, 20/12/2025, 8:38 pm

3. Trigger Design Pattern

Use Case

To ensure maintainability and scalability, trigger logic should be minimal and reusable.

Implementation

- Trigger only handles:
 - Context (insert/update)
 - Data collection
- Business logic is moved to **EnrollmentUpdater class**
- This separation improves:
 - Code readability
 - Debugging
 - Test coverage

4. SOQL & SOSL

Use Case

Enrollment records must be fetched efficiently based on **Student and Course** to avoid duplicate enrollments and governor limit issues.

Implementation

- Used selective SOQL queries with WHERE conditions

- Queried Enrollment records using:
 - Student__c
 - Course__c
- Avoided SOQL inside loops

```
SELECT Id, Student__c, Course__c, Total_Absences__c
FROM Enrollment__c
WHERE Student__c IN :studentIds
AND Course__c = :courseId
```

The screenshot shows the Salesforce Apex Class Editor for the **EnrollmentUpdater** class. The interface includes a header with the 'Apex Classes' setup icon and title. Below the class name, there are 'Apex Class Edit' tabs for 'Apex Class' and 'Version Settings', along with 'Save', 'Quick Save', and 'Cancel' buttons. The main editor area displays the following Apex code:

```

11  ) {
12      if (studentAbsenceIncrements == null || studentAbsenceIncrements.isEmpty()) {
13          if (courseId == null) return;
14      }
15      Set<Id> studentIds = studentAbsenceIncrements.keySet();
16
17      List<Enrollment__c> enrollments = [
18          SELECT Id, Student__c, Course__c, Total_Absences__c
19          FROM Enrollment__c
20          WHERE Student__c IN :studentIds
21          AND Course__c = :courseId
22      ];
23  }
  
```

5. Collections (List, Set, Map)

Use Case

Attendance records can be created or updated in bulk.
Collections ensure efficient processing without duplicate updates.

Implementation

- **Set<Id>** → Used to store unique Student IDs
- **Map<Id, Integer>** → Tracks absence count per student
- **List<Enrollment_c>** → Used for bulk DML operations

Benefits

- Bulk-safe
 - Prevents duplicate processing
 - Improves performance
-

6. Control Statements

Use Case

Attendance status changes need conditional handling.

Implementation

- Used **if-else conditions** to:
 - Detect Absent → increment
 - Detect Absent → Present → decrement
 - Used **for loops** to iterate over Trigger.new records
-

9. Test Classes

Use Case

Salesforce requires test coverage to deploy Apex code.

Implementation

- Created **EnrollmentUpdaterTest** class
 - Covered:
 - Single attendance insert
 - Bulk attendance insert
 - Status change scenarios
 - Achieved **>90% code coverage**
-

10. Final Outcome of Phase 5

- Attendance automation fully functional
- Enrollment absences updated accurately
- Apex logic bulk-safe and reusable
- Foundation ready for email alerts, reports, and dashboards