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**.

---

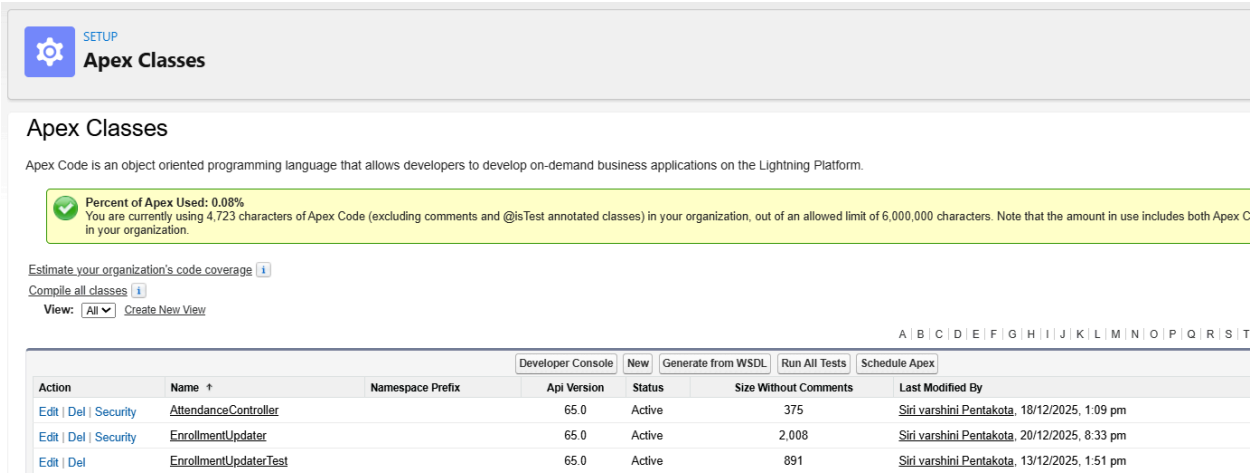
1. Classes & Objects

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




```
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

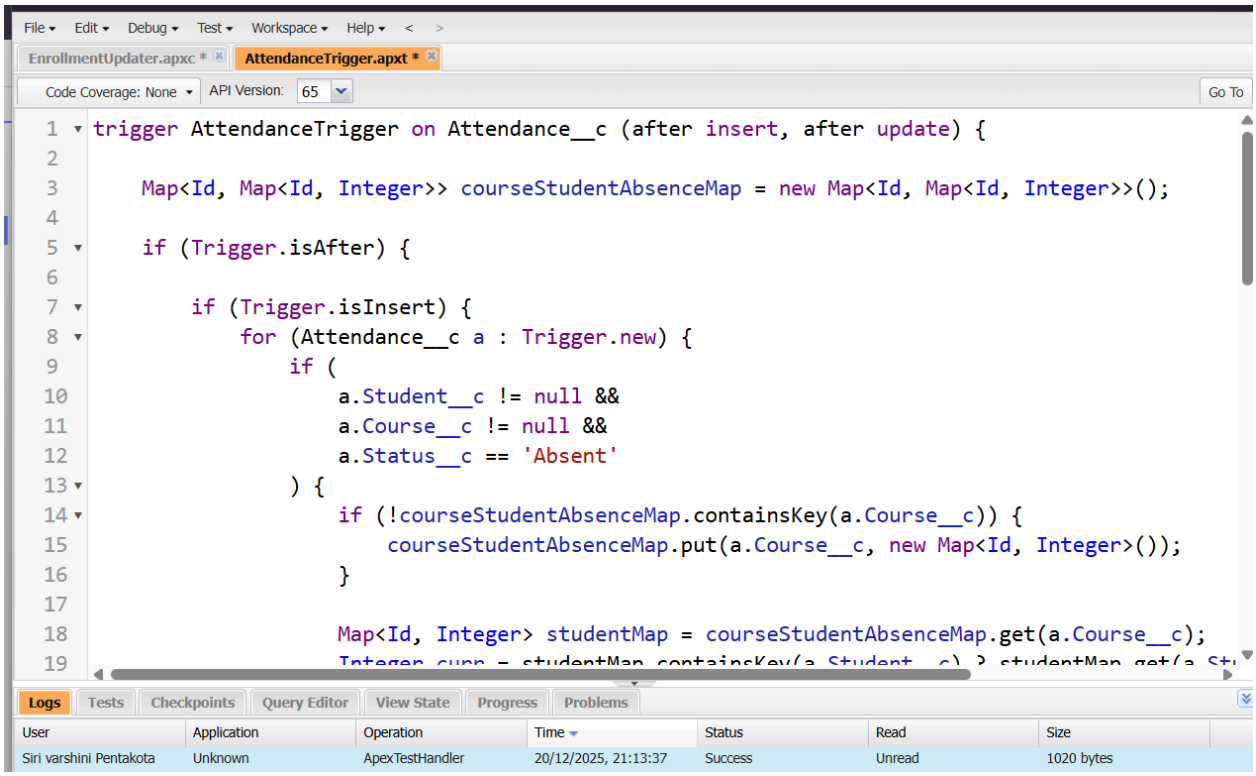
Whenever an Attendance record is:

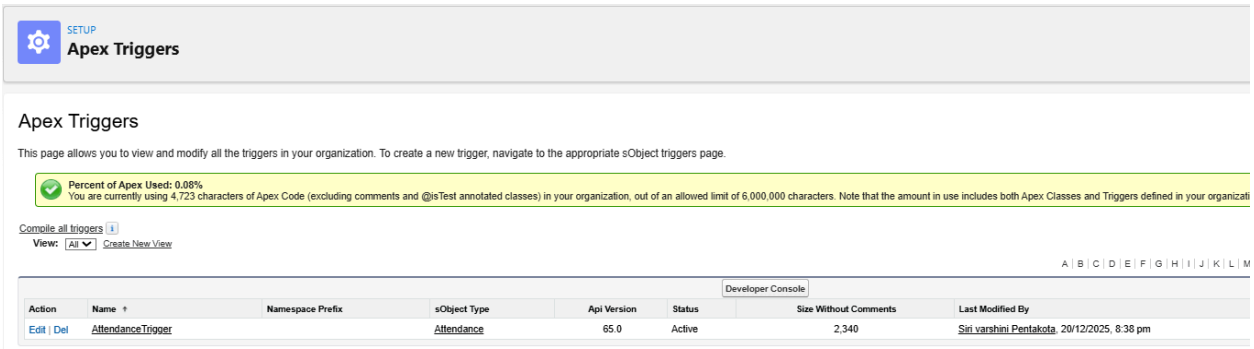
- Inserted with status **Absent**
- Updated from **Present → Absent** or **Absent → Present**

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





---

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

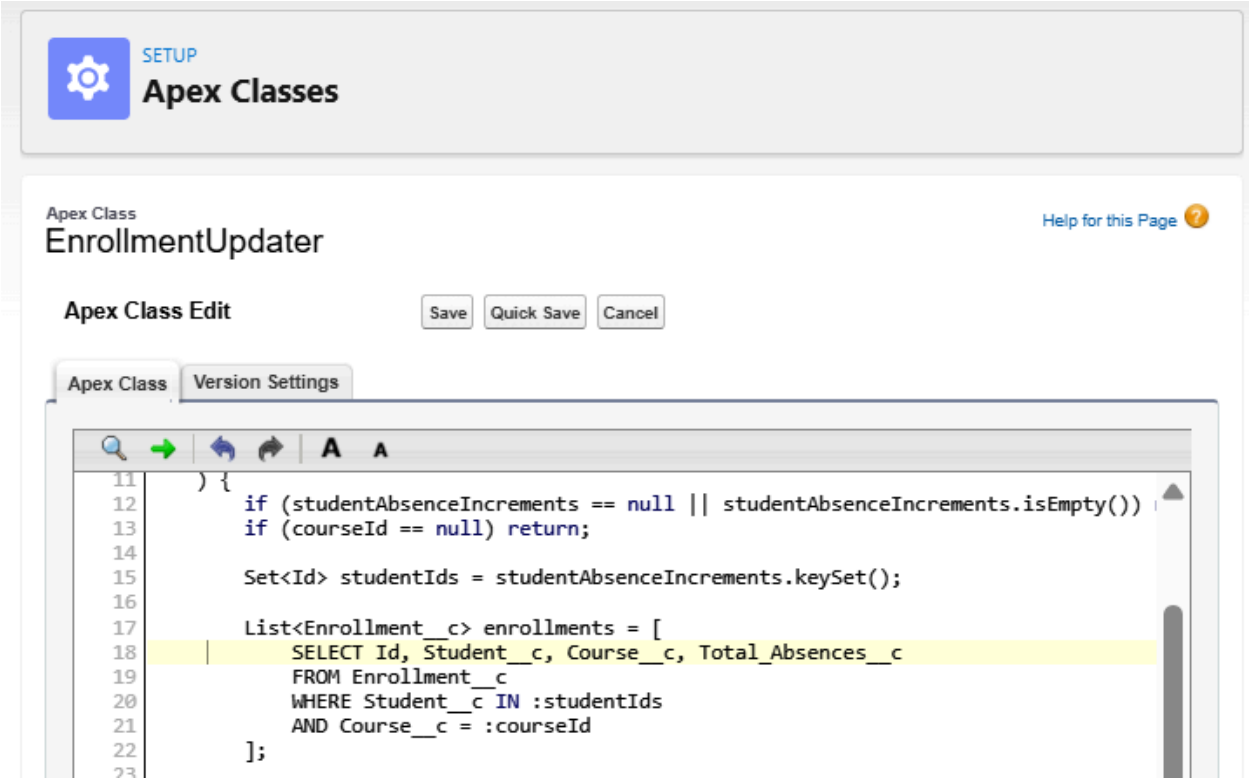
Enrollment records must be fetched efficiently based on **Student and Course** to avoid duplicate enrollments and governor limit issues.

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
```



---

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