INSTITUTE MANAGEMENT SYSTEM

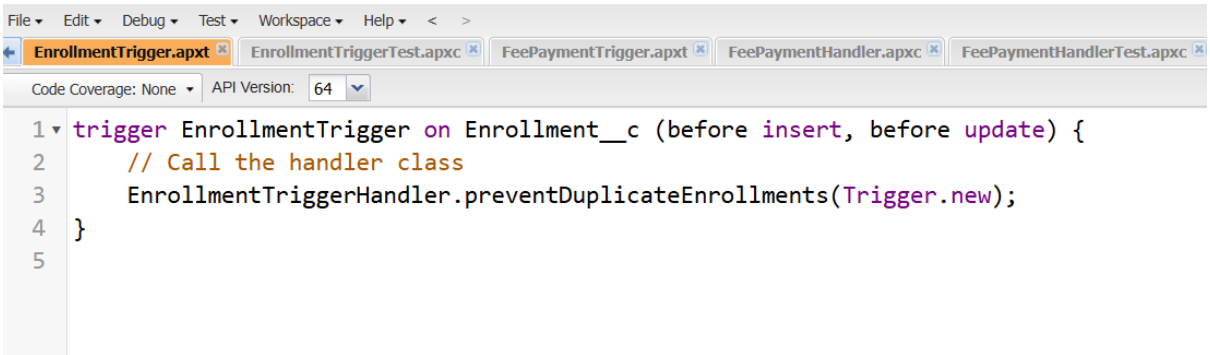
Phase 5: Apex Programming

Apex classes and Triggers:

📌 Enrollment Management

1. EnrollmentTrigger (Trigger)

Purpose:
Fires whenever a student enrollment record (Enrollment__c) is created or updated. Its only job is to call the EnrollmentTriggerHandler so the logic is not written directly in the trigger (best practice).



2. EnrollmentTriggerHandler (Handler Class)

Purpose:
Ensures that a student cannot be enrolled in the same course twice.

- Collects Student–Course combinations.
- Checks database for existing enrollments.
- Throws an error (addError) if a duplicate is found.

```
public class EnrollmentTriggerHandler {

    public static void preventDuplicateEnrollments(List<Enrollment__c> enrollments) {
        // Set to store new Student-Course combinations
        Set<String> newEnrollmentsSet = new Set<String>();

        // Sets to collect unique Student and Course IDs from the incoming records
        Set<Id> studentIds = new Set<Id>();
        Set<Id> courseIds = new Set<Id>();

        for (Enrollment__c e : enrollments) {
            if(e.Student__c != null && e.Course__c != null){
                studentIds.add(e.Student__c);
                courseIds.add(e.Course__c);

                // Also track combination of student-course in new batch
                newEnrollmentsSet.add(e.Student__c + '-' + e.Course__c);
            }
        }

        // Query existing enrollments in database
        Set<String> existingEnrollmentsSet = new Set<String>();
        for (Enrollment__c e : [
            SELECT Id, Student__c, Course__c
            FROM Enrollment__c
            WHERE Student__c IN :studentIds
              AND Course__c IN :courseIds
        ]) {
            existingEnrollmentsSet.add(e.Student__c + '-' + e.Course__c);
        }
```

```
        // Compare and throw error if duplicate found
        for (Enrollment__c e : enrollments) {
            if(e.Student__c != null && e.Course__c != null){
                String key = e.Student__c + '-' + e.Course__c;
                if(existingEnrollmentsSet.contains(key)){
                    e.addError('This student is already enrolled in the selected course.');
                }
            }
        }
    }
}
```

---

3. EnrollmentTriggerTest (Test Class)

Purpose:

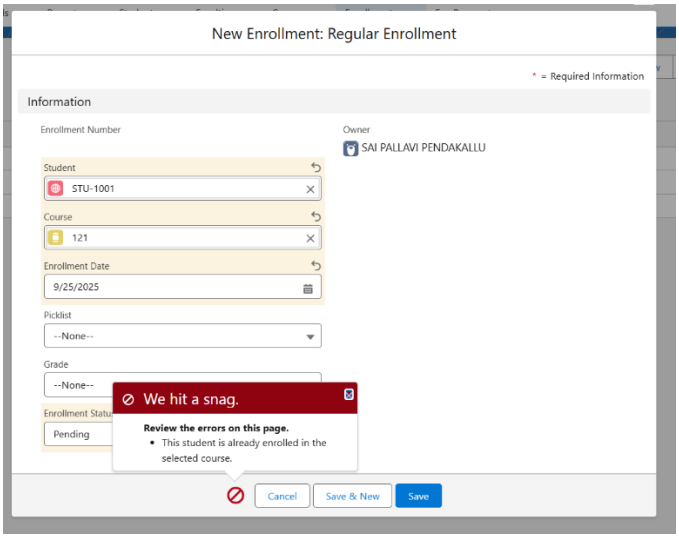
Verifies that the duplicate enrollment prevention logic works correctly.

- Inserts a valid enrollment (success).
- Tries inserting a duplicate (fails with error).
- Confirms the trigger behavior with assertions.

```apex
@IsTest
public class EnrollmentTriggerTest {
    @IsTest
    static void testPreventDuplicateEnrollments() {
        // Create sample student and course
        Student__c student = new Student__c(
            Name__c = 'Test Student',
            Email__c = 'test@student.com'
        );
        insert student;
        Course__c course = new Course__c(
            Course_Name__c = 'Test Course',
            Duration__c = 6
        );
        insert course;
        // Insert first enrollment (should succeed)
        Enrollment__c e1 = new Enrollment__c(
            Student__c = student.Id,
            Course__c = course.Id
        );
        insert e1;
        // Attempt duplicate enrollment (should fail)
        Enrollment__c e2 = new Enrollment__c(
            Student__c = student.Id,
            Course__c = course.Id
        );
        try {
            insert e2;
            System.assert(false, 'Expected duplicate enrollment error.');
        } catch (DmlException ex) {
            System.assert(ex.getMessage().contains('This student is already enrolled'));
```

OUTCOME:

- **Prevents Duplicate Enrollments**
  - A student cannot be enrolled in the same course more than once.
  - If attempted, Salesforce shows an error message: *"This student is already enrolled in the selected course."*

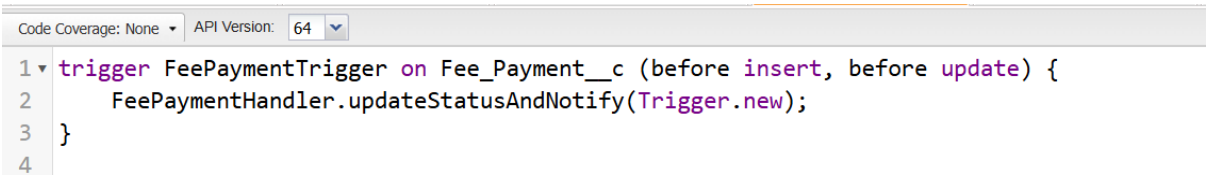


📌 Fee Payment Management

1. FeePaymentTrigger (Trigger)

Purpose:
Runs before insert or update of a Fee_Payment__c record.

- Calls FeePaymentHandler to update fee status and notify students.



---

2. FeePaymentHandler (Handler Class)

Purpose:
Automatically updates the fee payment status and sends confirmation.

- If Amount > 0 and Status != "Paid", update Status → "Paid".
- Sends an email to the student confirming payment.
- Bulk-safe for multiple payments at once.

---

3. FeePaymentHandlerTest (Test Class)

Purpose:
Confirms fee payments are marked as "Paid" correctly.

- Inserts a fee record with Pending status.
- Checks if trigger updates it to Paid.

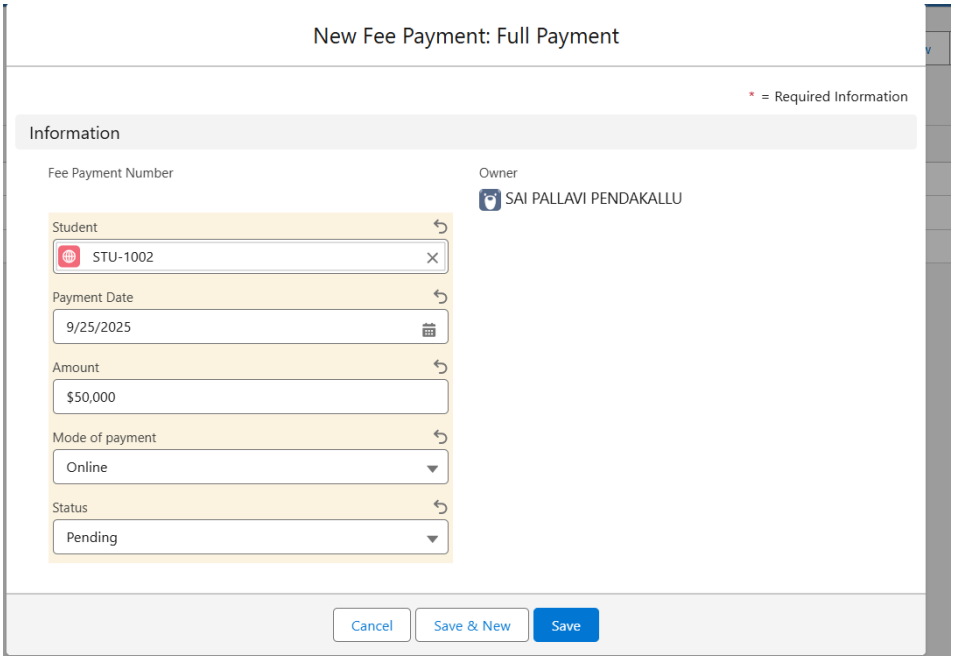
- Covers email notification logic in a safe way (emails not actually sent in test).

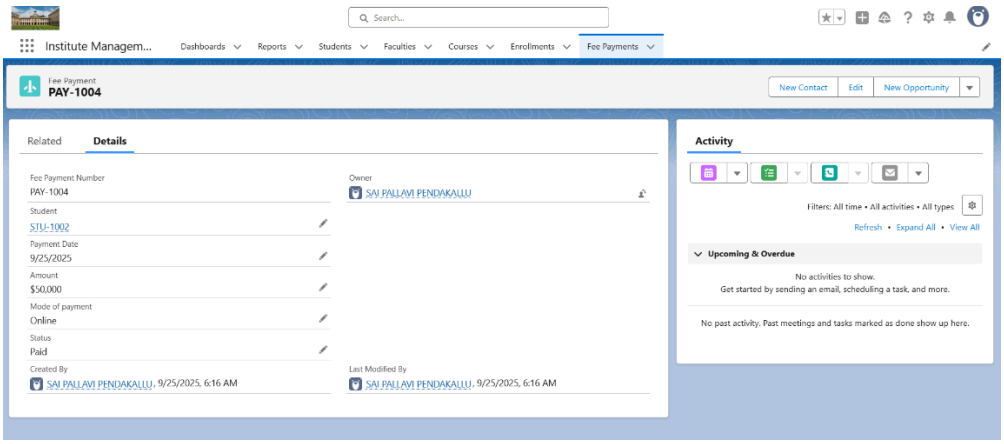
```
@IsTest
public class FeePaymentHandlerTest {
    @IsTest
    static void testFeePaymentStatusUpdate() {
        // Create sample student
        Student__c stu = new Student__c(Name__c='Test Student', Email__c='test@student.com');
        insert stu;

        // Insert payment
        Fee_Payment__c payment = new Fee_Payment__c(
            Student__c = stu.Id,
            Amount__c = 500,
            Status__c = 'Pending'
        );
        insert payment;

        // Verify status updated
        Fee_Payment__c fp = [SELECT Status__c FROM Fee_Payment__c WHERE Id = :payment.Id];
        System.assertEquals('Paid', fp.Status__c);
    }
}
```

Outcome:

□ Tracks Student Payments Accurately

- Each payment is linked to a student record.
- Captures amount, status (Pending, Paid, Overdue), and payment date.





 Prevents Inconsistencies

- Triggers and validation rules ensure no negative amounts or invalid payment entries.
- Stops duplicate or incomplete payment records.

📌 Enrollment Status Update

1. EnrollmentStatusTrigger (Trigger)

Purpose:
Fires when enrollment records are inserted or updated.

- Calls EnrollmentStatusHandler to check if a grade is entered.

```
trigger EnrollmentStatusTrigger on Enrollment__c (before insert, before update) {
    EnrollmentStatusHandler.updateStatusOnGrade(Trigger.new, Trigger.oldMap);
}
```

---

2. EnrollmentStatusHandler (Handler Class)

Purpose:
Automatically updates enrollment status based on grades.

- If Grade__c is filled in, update Enrollment_Status__c → "Completed".
- Ensures enrollments are closed out automatically when results are entered.

```
public class EnrollmentStatusHandler {
    public static void updateStatusOnGrade(List<Enrollment__c> newEnrollments, Map<Id, Enrollment__c> oldMap) {
        for(Enrollment__c e : newEnrollments) {
            // Check if grade is entered or updated
            if(e.Grade__c != null && (oldMap == null || e.Grade__c != oldMap.get(e.Id).Grade__c)) {
                e.Enrollment_Status__c = 'Completed';
            }
        }
    }
}
```

3. EnrollmentStatusHandlerTest (Test Class)

Purpose:
Tests that enrollment status auto-updates.

- Creates an Active enrollment.
- Updates with a grade.
- **Asserts status becomes "Completed".**

```
@IsTest
public class EnrollmentStatusHandlerTest {
    @IsTest
    static void testStatusUpdate() {
        // Create student and course
        Student__c stu = new Student__c(Name__c='Test Student', Email__c='test@student.com');
        insert stu;
        Course__c course = new Course__c(Course_Name__c='Test Course');
        insert course;

        Enrollment__c enrollment = new Enrollment__c(
            Student__c = stu.Id,
            Course__c = course.Id,
            Enrollment_Status__c = 'Active'
        );
        insert enrollment;

        // Update grade
        enrollment.Grade__c = 'A';
        update enrollment;

        Enrollment__c updated = [SELECT Enrollment_Status__c FROM Enrollment__c WHERE Id = :enrollment.Id];
        System.assertEquals('Completed', updated.Enrollment_Status__c);
    }
}
```