

PHASE 5: APEX PROGRAMMING (DEVELOPER)

Goal: Extend the Event Management System app with custom Apex logic for validations, automation, asynchronous processing, and unit testing.

Step 1: Apex Classes & Objects

1. Fee Payment Processing

- Developed FeePaymentService Apex class to securely process fee payments.
- Enabled storage of payment transactions, status updates, and fee types.
- Created corresponding test classes to validate payment methods and ensure code coverage.

Apex Class

FeePaymentService

Apex Class Detail

Edit

Delete

Download

Security

Show Dependencies

Name	FeePaymentService	Status	Active
Namespace Prefix		Code Coverage	0% (0/13)
Created By	Rasika Bharsakle , 10/3/2025, 2:18 AM	Last Modified By	Rasika Bharsakle , 10/3/2025, 2:18 AM

Class Body

Class Summary

Version Settings

Trace Flags

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

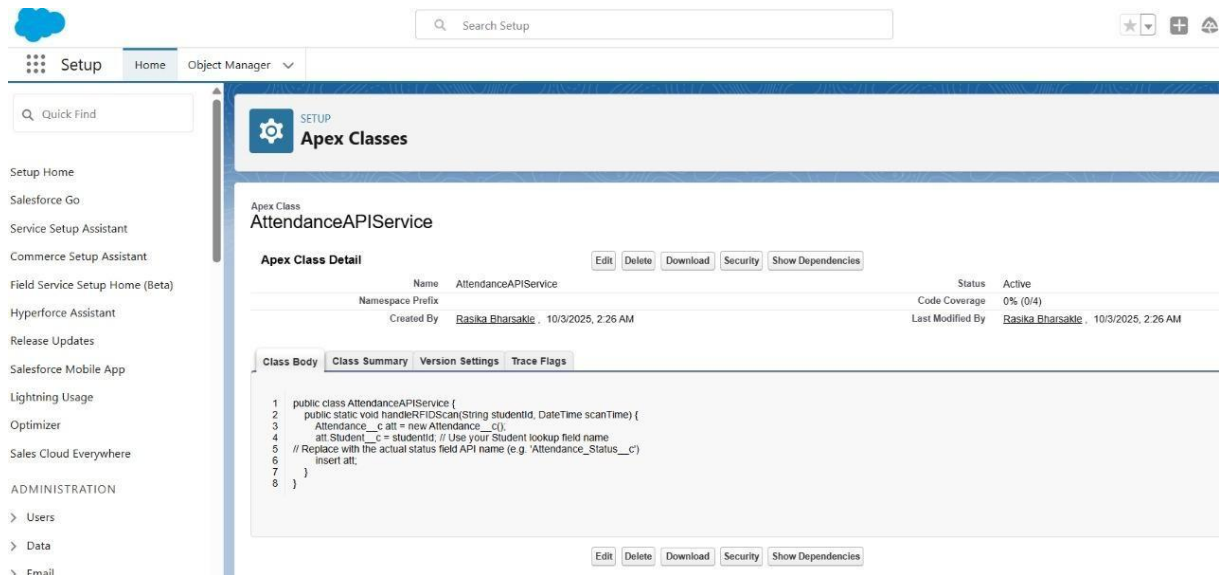
20

```
public class FeePaymentService {
    @AuraEnabled
    public static String processFeePayment(String studentId, Decimal feeAmount, String feeName, String examId) {
        try {
            // Create Fee record
            Fee__c fee = new Fee__c();
            fee.Student__c = studentId;
            fee.Fees_Amount__c = feeAmount;
            fee.Fees_Status__c = 'Paid';
            fee.Fees_Status__c = 'Paid';
            fee.Name = feeName;
            fee.Exam__c = examId;
            insert fee;
            return 'Payment successful';
        } catch (Exception ex) {
            return 'Payment failed: ' + ex.getMessage();
        }
    }
}
```

2. Attendance Integration

- Implemented AttendanceAPIService to process Biometric/RFID scan data.
- Automatically updated attendance records linked to student profiles.

- Developed test methods to verify attendance logging works correctly.



3. SMS Gateway Integration Preparation

- Designed SMSGatewayService Apex class framework to send SMS alerts using external SMS APIs.
- Setup remote site settings for HTTP callouts.
- Prepared test mocks for SMS sending validation without external dependencies.

The screenshot shows the Salesforce Setup interface. On the left is a navigation menu with categories like Email, Custom Code, and Environments. The 'Apex Classes' link is highlighted. The main content area displays the details for the 'SMSGatewayService' Apex Class. It includes a table with class information and a tabbed interface for viewing the class body, summary, settings, and flags. The 'Class Body' tab is active, showing the following Apex code:

```

1 public class SMSGatewayService {
2     public static void sendSMS(String toPhone, String messageBody) {
3         String accountId = 'YOUR_TWILIO_ACCOUNT_SID';
4         String authToken = 'YOUR_TWILIO_AUTH_TOKEN';
5         String fromPhone = 'YOUR_TWILIO_PHONE_NUMBER';
6
7         String endpoint = 'https://api.twilio.com/2010-04-01/Accounts/' + accountId + '/Messages.json';
8         HttpRequest req = new HttpRequest();
9         req.setEndpoint(endpoint);
10        req.setMethod('POST');
11
12        String body = 'From=' + EncodingUtil.urlEncode(fromPhone, 'UTF-8') +
13                    '&To=' + EncodingUtil.urlEncode(toPhone, 'UTF-8') +
14                    '&Body=' + EncodingUtil.urlEncode(messageBody, 'UTF-8');
15
16        req.setBody(body);
17        Blob headerValue = Blob.valueOf(accountId + ':' + authToken);
18        String authorizationHeader = 'Basic: ' + EncodingUtil.base64Encode(headerValue);
19        req.setHeader('Authorization', authorizationHeader);
20        req.setHeader('Content-Type', 'application/x-www-form-urlencoded');
21    }
22 }

```

Step 2: Apex Triggers

- A trigger Attendance & Fee Trigger was implemented on Attendance_c & Fee_c.
- It executes before insert and before update, ensuring invalid registrations are blocked before being committed to the database.
- The trigger itself is lean, simply delegating work to the service class.
- This follows the Trigger Handler Design Pattern, which separates trigger context from business logic.

SETUP > OBJECT MANAGER

Attendance

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Object Access

Triggers

1 Items, Sorted by Label

Q, Quick Find [New](#)

LABEL	API VERSION	SIZE WITHOUT COMMENTS	MODIFIED BY
AttendanceTrigger	64.0	63	Rasika Bharsakle, 10/3/2025, 9:56 AM

SETUP > OBJECT MANAGER

Fee

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

Triggers

1 Items, Sorted by Label

Q, Quick Find [New](#)

LABEL	API VERSION	SIZE WITHOUT COMMENTS	MODIFIED BY
FeePaymentTrigger	64.0	56	Rasika Bharsakle, 10/3/2025, 9:55 AM

Step 1: Test Cases

- This test class also ensures code coverage requirements are met for deployment.
- An initial failure occurred due to a conflicting Flow; this was resolved by bypassing/deactivating it during test runs.

```
profammegheinstituteofte37-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apextrigger
File Edit Debug Test Workspace Help < >
FeePaymentTriggerTest.apxc *
Code Coverage: None API Version: 64
1 @isTest
2 public class FeePaymentTriggerTest {
3
4     @testSetup
5     static void setupData() {
6         // Create a student record
7         Student__c student = new Student__c(Name = 'John Doe', Fee_Status__c = 'Pending');
8         insert student;
9
10        // Create a pending fee payment
11        Fee_Payment__c payment = new Fee_Payment__c(
12            Student__c = student.Id,
13            Amount__c = 1000,
14            Status__c = 'Pending'
15        );
16        insert payment;
17    }
```