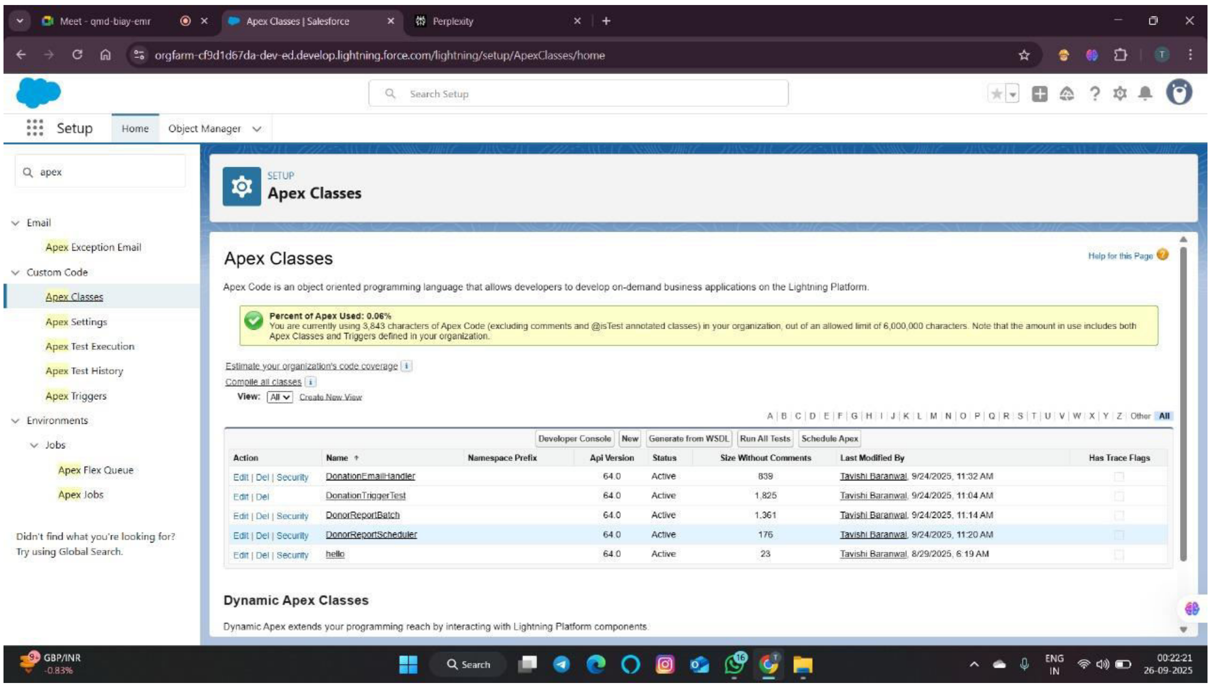
Phase 5: Apex Programming (Developer)

Classes & Objects:

Healthcare: Apex classes manage reusable logic like patient billing or appointment scheduling.
Donation: Classes handle donor segmentation or campaign budget calculations. Objects represent real-world entities (patients, donations), while classes encapsulate behaviors, ensuring structured, maintainable code across both systems.



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: 0.06%
You are currently using 3,843 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.

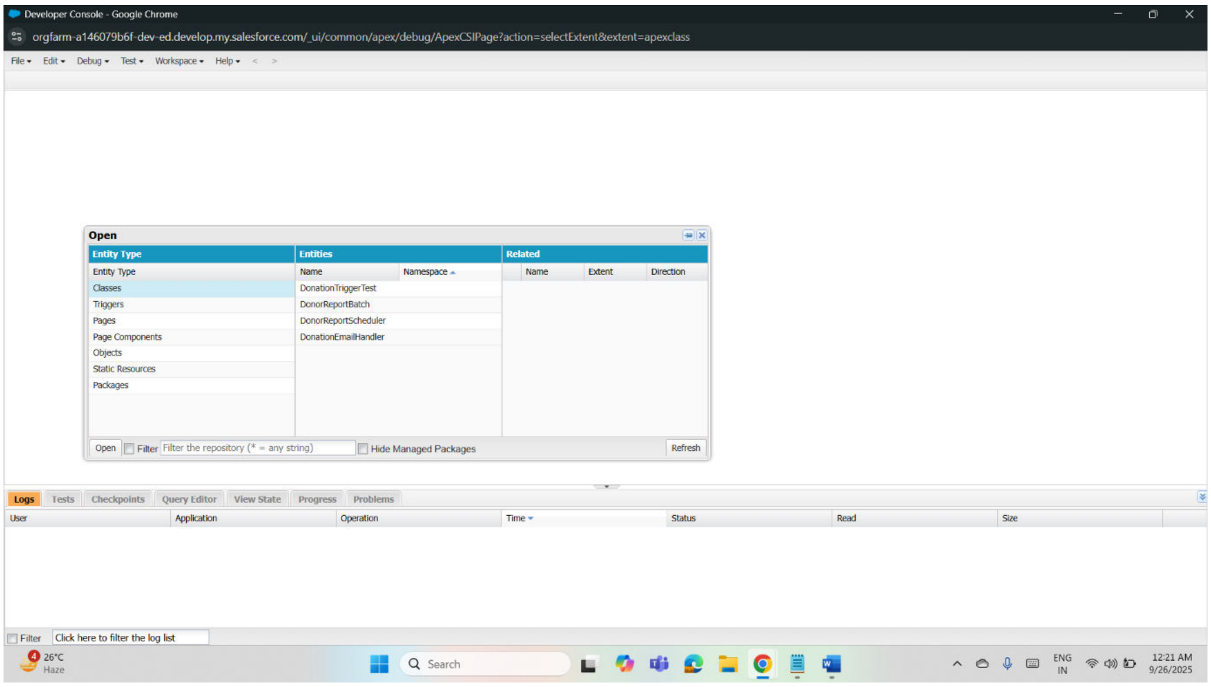
Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit | Del | Security	DonationEmailHandler		64.0	Active	839	Tavishi Baranwal, 9/24/2025, 11:32 AM	
Edit | Del	DonationTriggerTest		64.0	Active	1,825	Tavishi Baranwal, 9/24/2025, 11:04 AM	
Edit | Del | Security	DonorReportBatch		64.0	Active	1,361	Tavishi Baranwal, 9/24/2025, 11:14 AM	
Edit | Del | Security	DonorReportScheduler		64.0	Active	176	Tavishi Baranwal, 9/24/2025, 11:20 AM	
Edit | Del | Security	hello		64.0	Active	23	Tavishi Baranwal, 8/29/2025, 6:19 AM	

Dynamic Apex Classes

Dynamic Apex extends your programming reach by interacting with Lightning Platform components.

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

Healthcare: Trigger alerts before inserting duplicate patient records, or after updating treatment completion.
Donation: Before-insert prevents duplicate donors, after-insert creates donor acknowledgment records. Triggers enforce data integrity and automate complex logic at database events.


Developer Console - Google Chrome

orgfarm-a146079b6f-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apexclass

Open

Entity Type	Entities	
Entity Type	Name	Namespace
Classes	DonationTriggerTest	
Triggers	DonorReportBatch	
Pages	DonorReportScheduler	
Page Components	DonationEmailHandler	
Objects		
Static Resources		
Packages		

Related: Name, Extent, Direction

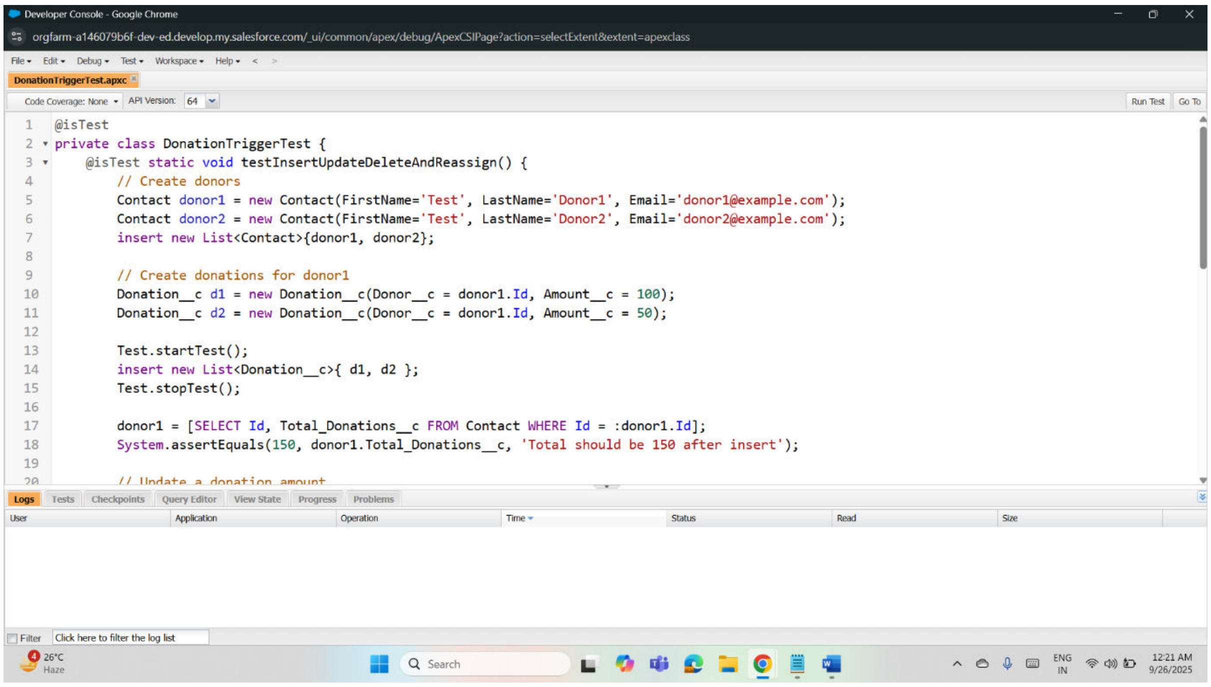
Filter: Filter the repository (* = any string) · Hide Managed Packages · Refresh

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size
Filter Click here to filter the log list
26°C Haze
Search
ENG IN
12:21 AM 9/26/2025

Trigger Design Pattern:

Healthcare: Organizes triggers to manage patient workflows without recursion or redundancy.

Donation: Manages campaign-related updates systematically. The design pattern separates logic into handler classes, ensuring scalability, reusability, and compliance with Salesforce best practices.


Developer Console - Google Chrome

orgfarm-a146079b6f-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apexclass

DonationTriggerTest.apxc

Code Coverage: None · API Version: 64

```
@isTest
private class DonationTriggerTest {
    @isTest static void testInsertUpdateDeleteAndReassign() {
        // Create donors
        Contact donor1 = new Contact(FirstName='Test', LastName='Donor1', Email='donor1@example.com');
        Contact donor2 = new Contact(FirstName='Test', LastName='Donor2', Email='donor2@example.com');
        insert new List<Contact>{donor1, donor2};

        // Create donations for donor1
        Donation__c d1 = new Donation__c(Donor__c = donor1.Id, Amount__c = 100);
        Donation__c d2 = new Donation__c(Donor__c = donor1.Id, Amount__c = 50);

        Test.startTest();
        insert new List<Donation__c>{ d1, d2 };
        Test.stopTest();

        donor1 = [SELECT Id, Total_Donations__c FROM Contact WHERE Id = :donor1.Id];
        System.assertEquals(150, donor1.Total_Donations__c, 'Total should be 150 after insert');

        // Update a donation amount
```

File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾
Run Test Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size
Filter Click here to filter the log list
26°C Haze
Search
ENG IN
12:21 AM 9/26/2025

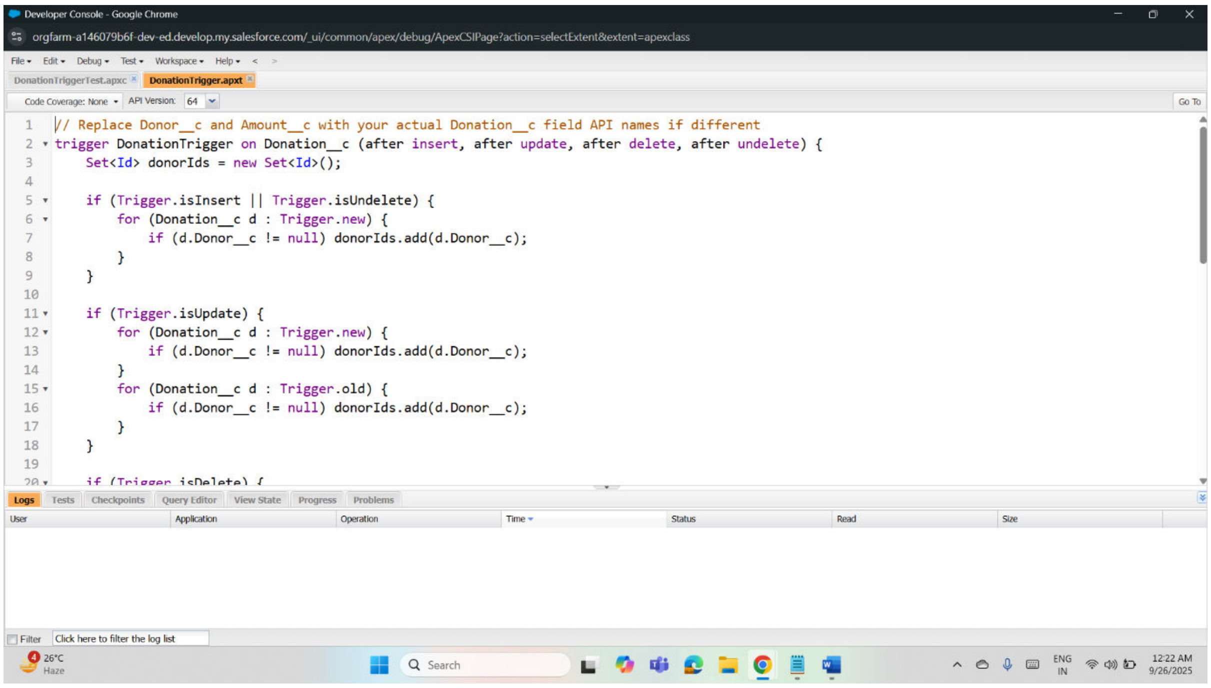
SOQL & SOSL:

Healthcare: SOQL queries fetch patient medical history; SOSL retrieves

emergency contacts across objects.
Donation: SOQL retrieves donation records by campaign; SOSL searches donor names across multiple fields. Both optimize data retrieval and reporting.

Collections (List, Set, Map):

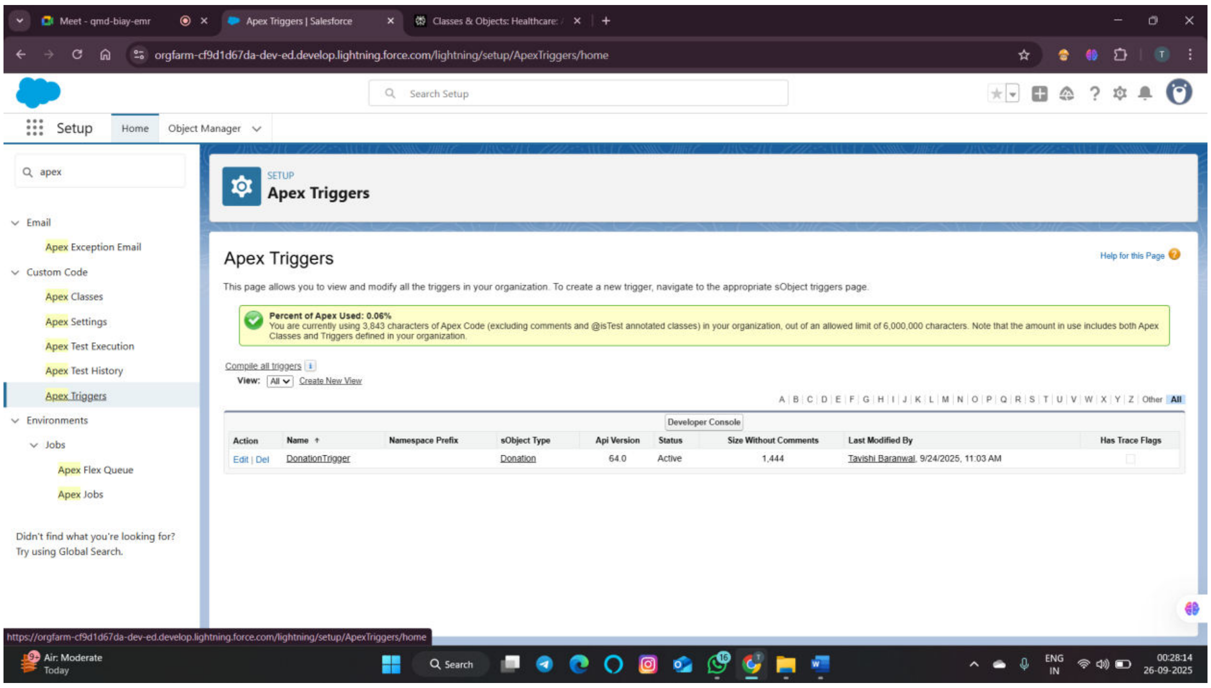
Healthcare: Lists store multiple appointments; Sets prevent duplicate patient IDs; Maps link doctor IDs with patients.
Donation: Lists store donor records; Sets ensure unique campaign names; Maps connect donors with their pledges. Collections manage data efficiently in bulk operations.


Developer Console - Google Chrome
orgfarm-a146079b6f-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage?action=selectExtent&extent=apexclass
File Edit Debug Test Workspace Help
DonationTriggerTest.apxc
DonationTrigger.apxt
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size
Filter Click here to filter the log list

```
// Replace Donor__c and Amount__c with your actual Donation__c field API names if different
trigger DonationTrigger on Donation__c (after insert, after update, after delete, after undelete) {
    Set<Id> donorIds = new Set<Id>();

    if (Trigger.isInsert || Trigger.isUndelete) {
        for (Donation__c d : Trigger.new) {
            if (d.Donor__c != null) donorIds.add(d.Donor__c);
        }
    }

    if (Trigger.isUpdate) {
        for (Donation__c d : Trigger.new) {
            if (d.Donor__c != null) donorIds.add(d.Donor__c);
        }
        for (Donation__c d : Trigger.old) {
            if (d.Donor__c != null) donorIds.add(d.Donor__c);
        }
    }

    if (Trigger.isDelete) {
```

Control Statements:

Healthcare: Conditional logic routes critical lab results to doctors immediately.
Donation: Loops process multiple donations; IF-ELSE statements apply discounts for recurring pledges. Control statements guide code flow, making automation intelligent.



SETUP
Apex Triggers

Apex Triggers

Help for this Page

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: 0.06%
You are currently using 3,843 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 | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

Developer Console

Action	Name ↑	Namespace Prefix	sObject Type	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit | Del	DonationTrigger		Donation	64.0	Active	1,444	Tavishi Baranwal, 9/24/2025, 11:03 AM	☐

Batch Apex:

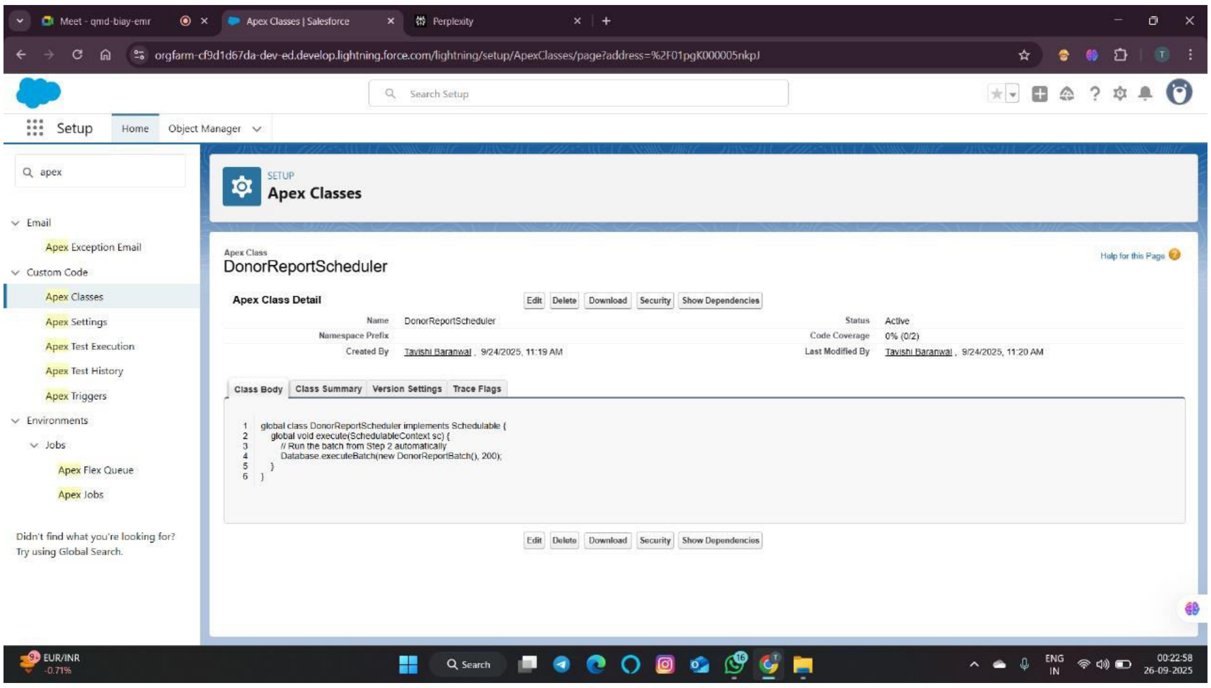
Healthcare: Processes bulk patient records for billing or insurance claims. Donation: Updates thousands of donor records during annual campaign reconciliation. Batch Apex handles large datasets asynchronously without hitting governor limits.

Queueable Apex:

Healthcare: Chain background jobs like scheduling patient reminders. Donation: Run complex donor segmentation logic asynchronously. Queueable Apex supports job chaining and structured asynchronous processing beyond Batch Apex.

Scheduled Apex:

Healthcare: Automates daily patient checkup reminders or medication reports. Donation: Schedules monthly donor pledge reminders or quarterly fundraising summaries. Scheduled Apex executes recurring tasks at defined intervals.


Apex Class
DonorReportScheduler

Apex Class Detail

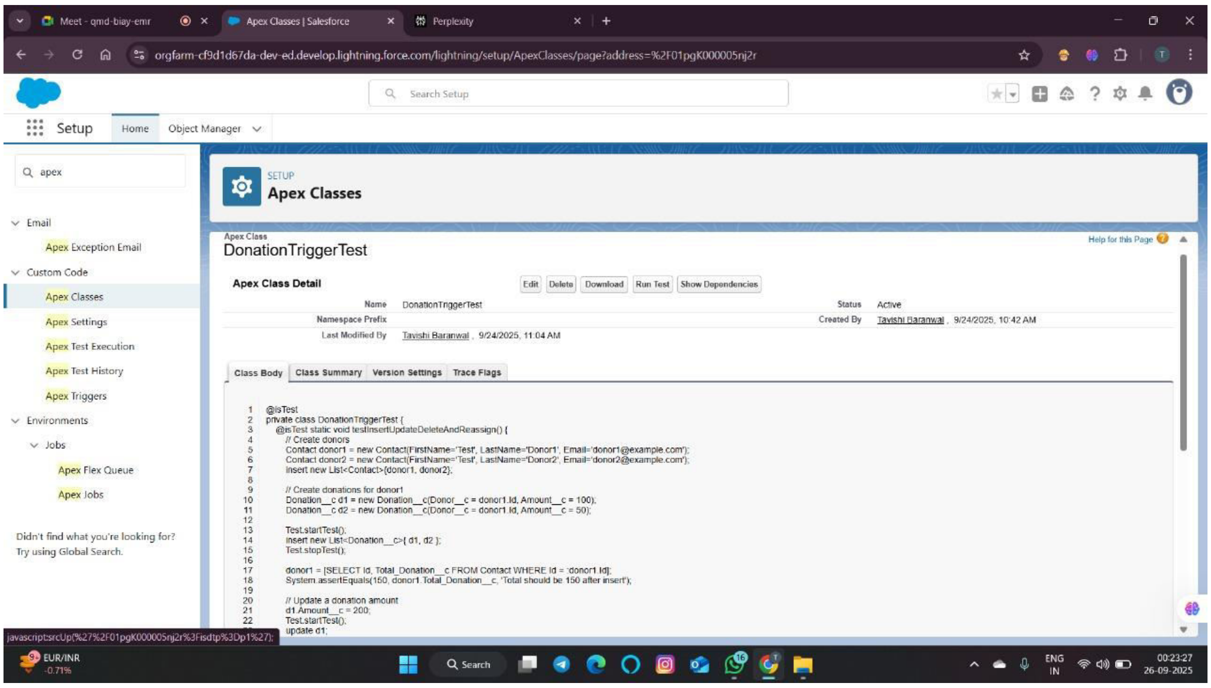
Name: DonorReportScheduler
Namespace Prefix:
Created By: Tavishi Baranwal , 9/24/2025, 11:19 AM
Status: Active
Code Coverage: 0% (0/2)
Last Modified By: Tavishi Baranwal , 9/24/2025, 11:20 AM

```
global class DonorReportScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        // Run the batch from Step 2 automatically
        Database.executeBatch(new DonorReportBatch(), 200);
    }
}
```

Future Methods:

Healthcare: Send patient lab result notifications asynchronously.

Donation: Call external payment gateways in the background. Future methods free up synchronous transactions by running time-consuming operations later.


Apex Class
DonationTriggerTest

Apex Class Detail

Name: DonationTriggerTest
Namespace Prefix:
Last Modified By: Tavishi Baranwal , 9/24/2025, 11:04 AM
Status: Active
Created By: Tavishi Baranwal , 9/24/2025, 10:42 AM

```
@isTest
private class DonationTriggerTest {
    @isTest static void testInsertUpdateDeleteAndReassign() {
        // Create donors
        Contact donor1 = new Contact(FirstName='Test', LastName='Donor1', Email='donor1@example.com');
        Contact donor2 = new Contact(FirstName='Test', LastName='Donor2', Email='donor2@example.com');
        insert new List<Contact>{donor1, donor2};

        // Create donations for donor1
        Donation__c d1 = new Donation__c(Donor__c = donor1.Id, Amount__c = 100);
        Donation__c d2 = new Donation__c(Donor__c = donor1.Id, Amount__c = 50);

        Test.startTest();
        insert new List<Donation__c>{ d1, d2 };
        Test.stopTest();

        donor1 = [SELECT Id, Total_Donation__c FROM Contact WHERE Id = :donor1.Id];
        System.assertEquals(150, donor1.Total_Donation__c, 'Total should be 150 after insert');

        // Update a donation amount
        d1.Amount__c = 200;
        Test.startTest();
        update d1;
```

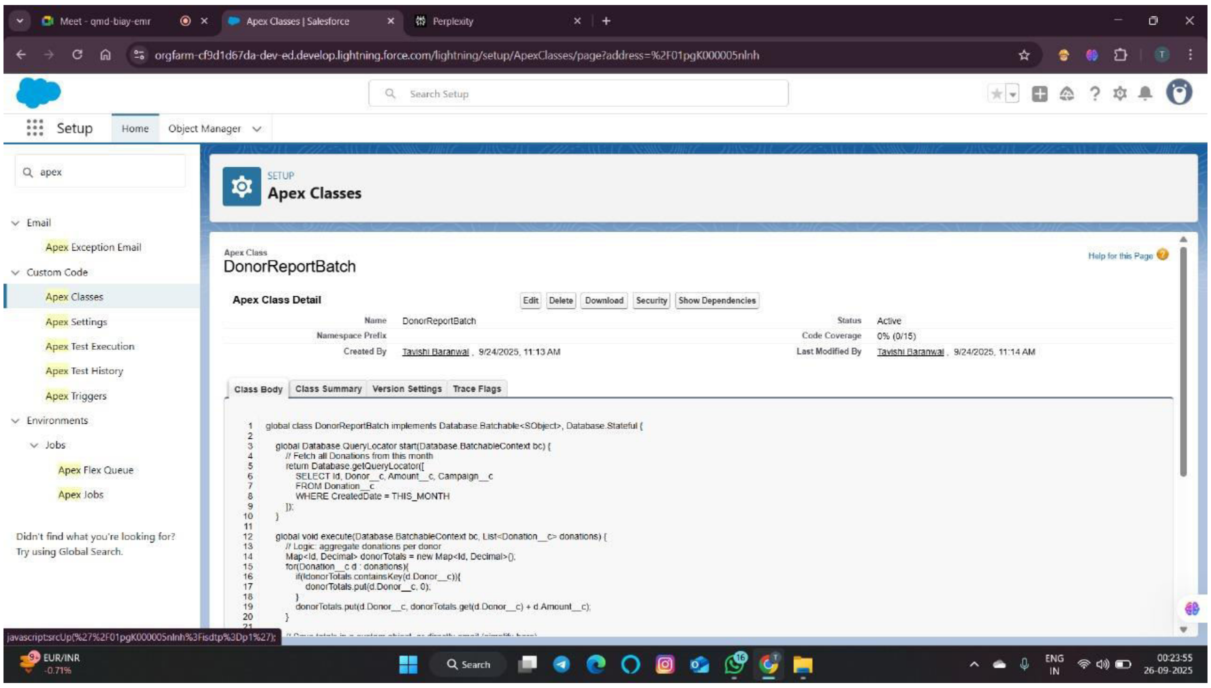
Exception Handling:

Healthcare: Catch errors in prescription workflows to prevent wrong medicine entries.

Donation: Handle failed donation transactions gracefully with user-friendly messages. Exception handling improves system reliability and user trust.

Test Classes:

Healthcare: Validate patient workflows, ensuring triggers and classes work correctly.
Donation: Test donor creation, campaign automation, and batch processes. Test classes ensure code quality, 75% coverage, and compliance with Salesforce deployment requirements.



Apex Class
DonorReportBatch

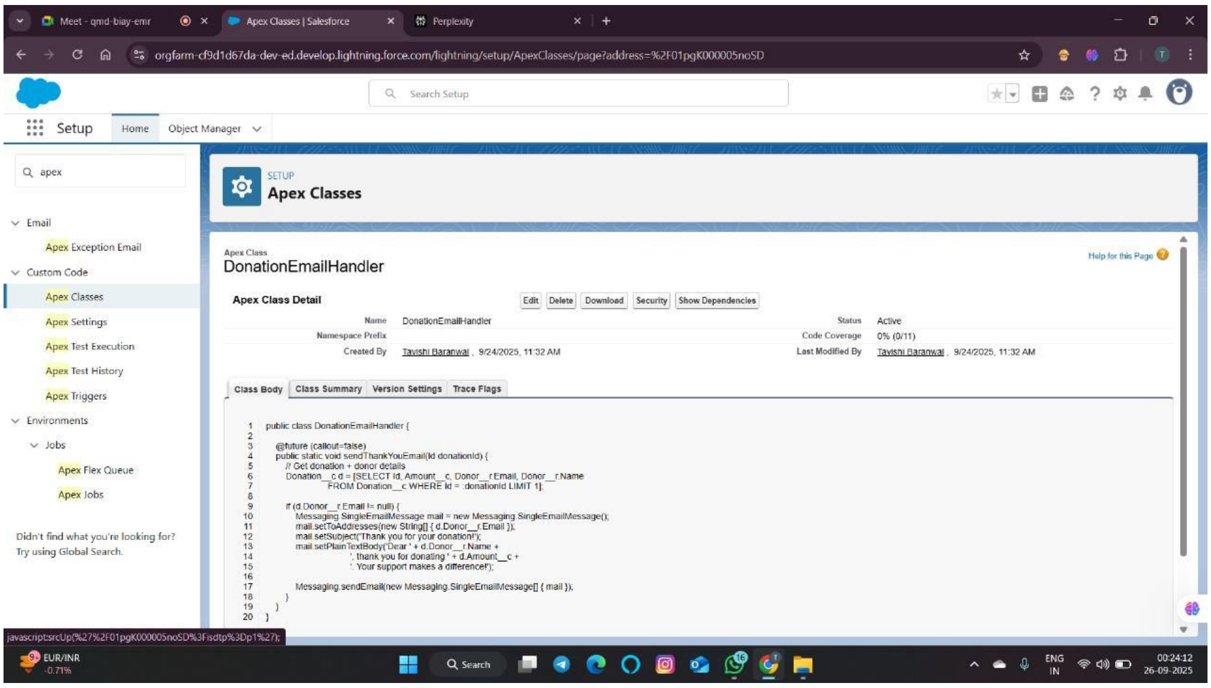
Apex Class Detail

Name	DonorReportBatch	Status	Active
Namespace Prefix		Code Coverage	0% (0/15)
Created By	Tavishi Baranwal , 9/24/2025, 11:13 AM	Last Modified By	Tavishi Baranwal , 9/24/2025, 11:14 AM

```
global class DonorReportBatch implements Database.Batchable<SObject>, Database.Stateful {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        // Fetch all Donations from this month
        return Database.getQueryLocator([
            SELECT Id, Donor__c, Amount__c, Campaign__c
            FROM Donation__c
            WHERE CreatedDate = THIS_MONTH
        ]);
    }

    global void execute(Database.BatchableContext bc, List<Donation__c> donations) {
        // Logic: aggregate donations per donor
        Map<Id, Decimal> donorTotals = new Map<Id, Decimal>();
        for(Donation__c d : donations){
            if(!donorTotals.containsKey(d.Donor__c)){
                donorTotals.put(d.Donor__c, 0);
            }
            donorTotals.put(d.Donor__c, donorTotals.get(d.Donor__c) + d.Amount__c);
        }
```

Asynchronous Processing:

Healthcare: Run intensive data jobs like medical history analysis asynchronously.
Donation: Process thousands of donations or generate pledge reports in the background. Asynchronous processing improves performance, avoids limits, and maintains user experience.



Apex Class
DonationEmailHandler

Apex Class Detail

Name	DonationEmailHandler	Status	Active
Namespace Prefix		Code Coverage	0% (0/11)
Created By	Tavishi Baranwal , 9/24/2025, 11:32 AM	Last Modified By	Tavishi Baranwal , 9/24/2025, 11:32 AM

```
public class DonationEmailHandler {

    @future (callout=false)
    public static void sendThankYouEmail(Id donationId) {
        // Get donation + donor details
        Donation__c d = [SELECT Id, Amount__c, Donor__r.Email, Donor__r.Name
                    FROM Donation__c WHERE Id = :donationId LIMIT 1];

        if (d.Donor__r.Email != null) {
            Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
            mail.setToAddresses(new String[] { d.Donor__r.Email });
            mail.setSubject('Thank you for your donation!');
            mail.setPlainTextBody('Dear ' + d.Donor__r.Name +
                          ', thank you for donating ' + d.Amount__c +
                          '. Your support makes a difference!');

            Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
        }
    }
}
```