Phase 5 Report

Apex Programming(Developer)

Project: Smart Healthcare Appointment & Compliance Hub
Batch: 4
Program: TCS Last Mile SmartBridge
Prepared by: Palla Bhugarbha

1. Introduction

This phase focuses on developing custom business logic using **Apex programming** for the Smart Healthcare Appointment & Compliance Hub. It extends the system beyond declarative tools by implementing triggers, classes, batch jobs, and scheduled processes to enforce complex healthcare rules such as preventing double bookings, automating compliance checks, and sending appointment reminders. This ensures scalability, reliability, and advanced automation tailored to real-world hospital workflows.

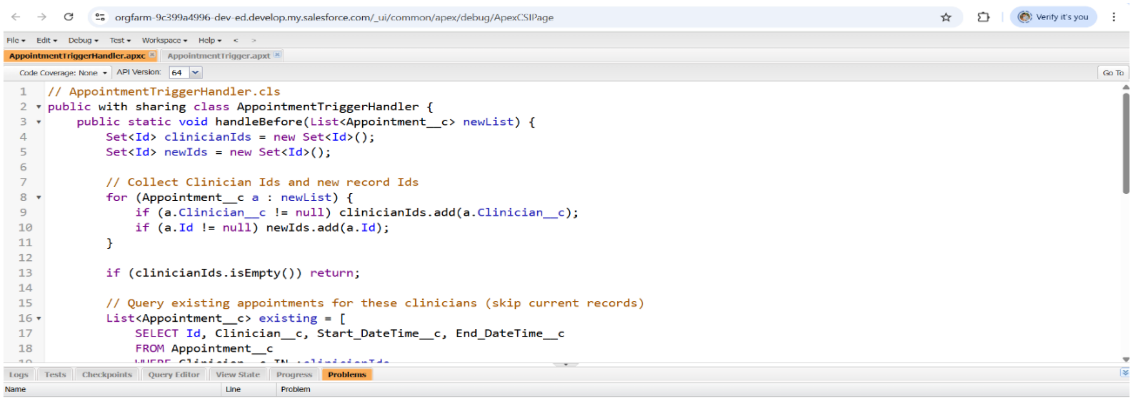
2. Objectives

- Develop **Apex classes and triggers** to enforce healthcare-specific rules such as avoiding clinician double-booking.
- Implement **SOQL queries and asynchronous processing** (Queueable/Batch Apex) to handle large volumes of appointments and compliance records efficiently.
- Configure **Scheduled Apex jobs** to automate daily reminders and compliance expiry checks.
- Ensure system reliability with **Future methods and test classes**, achieving required test coverage for deployment.
- Optimize performance and maintainability using **Trigger design patterns and bulk-safe coding practices**.

3. Steps Performed

3.1 Create Apex Classes

- Created *AppointmentTriggerHandler* class to validate overlapping appointment times for clinicians.
- Built *AppointmentTrigger* on Appointment__c to call the handler before insert/update.
- Implemented *ComplianceTriggerHandler* class to auto-update expired compliance records.
- Verified by creating test appointments and compliance records, capturing error messages and automatic status changes.

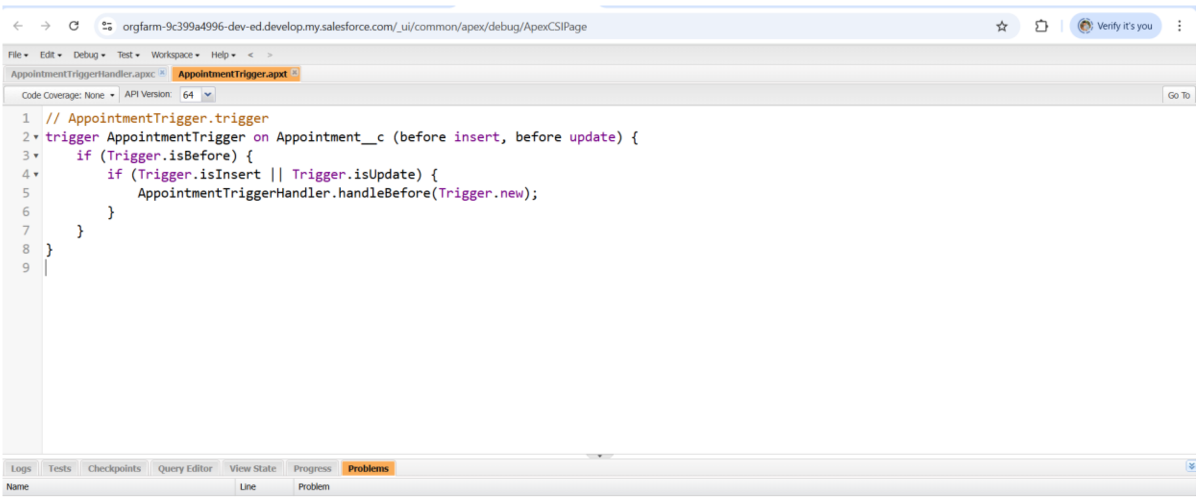


orgfarm-9c399a4996-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
Verify it's you
File Edit Debug Test Workspace Help
AppointmentTriggerHandler.apxc
AppointmentTrigger.apxt
Code Coverage: None
API Version: 64
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
Name Line Problem

```
// AppointmentTriggerHandler.cls
public with sharing class AppointmentTriggerHandler {
    public static void handleBefore(List<Appointment__c> newList) {
        Set<Id> clinicianIds = new Set<Id>();
        Set<Id> newIds = new Set<Id>();

        // Collect Clinician Ids and new record Ids
        for (Appointment__c a : newList) {
            if (a.Clinician__c != null) clinicianIds.add(a.Clinician__c);
            if (a.Id != null) newIds.add(a.Id);
        }

        if (clinicianIds.isEmpty()) return;

        // Query existing appointments for these clinicians (skip current records)
        List<Appointment__c> existing = [
            SELECT Id, Clinician__c, Start_DateTime__c, End_DateTime__c
            FROM Appointment__c
```

3.2 Create Apex Triggers

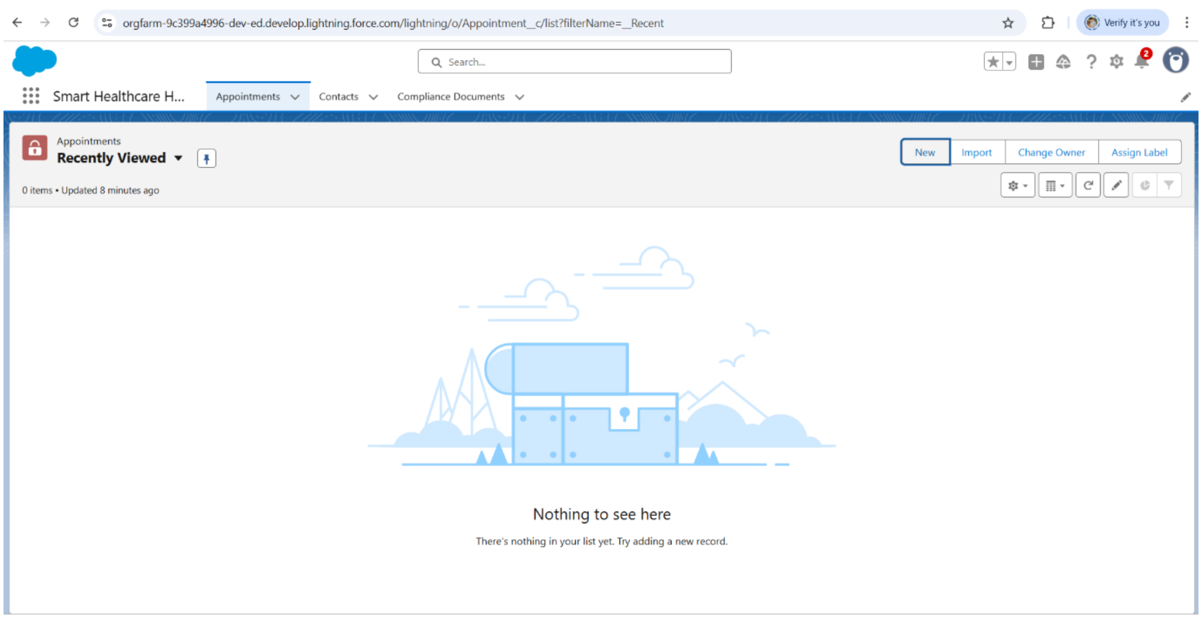
- Used a *Handler Class* instead of writing logic directly in the trigger.
- Improved readability and maintainability of code by separating logic.
- Ensured code is *bulk-safe* by handling multiple records in lists.
- Documented pattern with screenshots of both trigger and handler class.



```
// AppointmentTrigger.trigger
trigger AppointmentTrigger on Appointment__c (before insert, before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {
            AppointmentTriggerHandler.handleBefore(Trigger.new);
        }
    }
}
```

3.3 Make the Custom Object Tab Visible

- Created a **Custom Tab** for the Appointment__c object in Setup → Tabs.
- Built a **new Lightning App** in App Manager and added the Appointment tab.
- Verified the tab became visible in the App Launcher.
- Created sample Appointment records through the UI to test triggers.



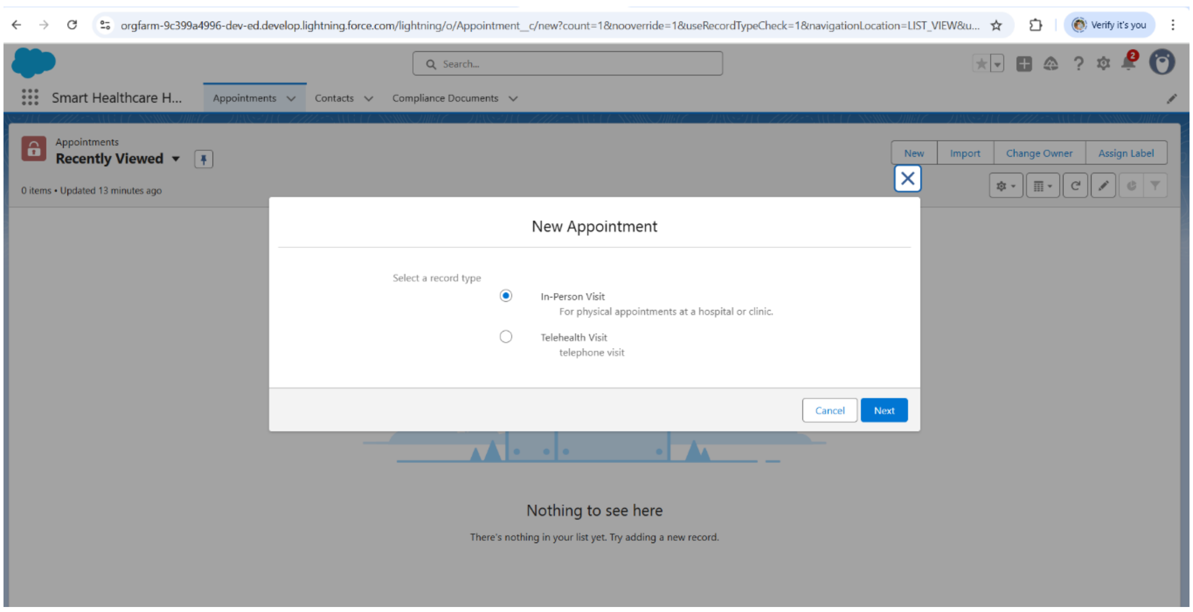
0 items • Updated 8 minutes ago

Nothing to see here

There's nothing in your list yet. Try adding a new record.

3.4 Create Test Data

- Opened the **Smart Healthcare App** and navigated to the *Appointments* tab.
- Clicked **New Appointment**, selected **Record Type = In-Person Visit**.
- Filled required fields: Clinician (doctor user), Patient (contact), Start Time, End Time, and Status = Scheduled.
- Saved the record successfully and confirmed the appointment appeared in the list view.


orgfarm-9c399a4996-dev-ed.develop.lightning.force.com/lightning/o/Appointment__c/new?count=1&nooverride=1&useRecordTypeCheck=1&navigationLocation=LIST_VIEW&u...
Verify it's you
Search...
Smart Healthcare H...
Appointments
Contacts
Compliance Documents
Recently Viewed
0 items • Updated 13 minutes ago
New
Import
Change Owner
Assign Label
Cancel
Next

New Appointment

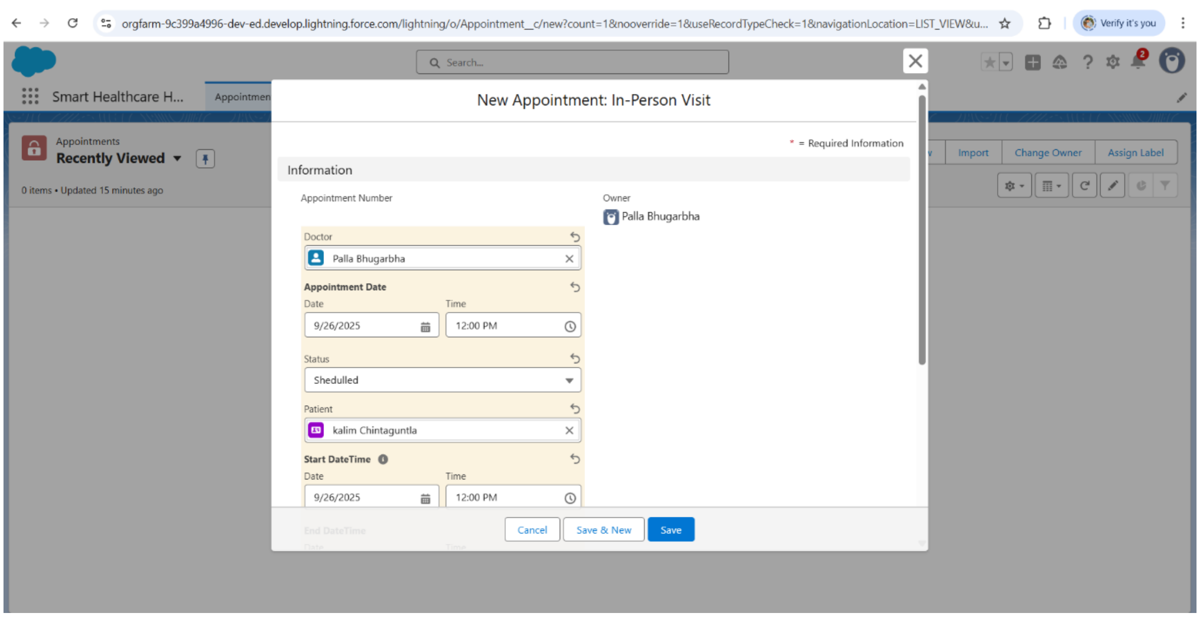
Select a record type

[✓ In-Person Visit]
For physical appointments at a hospital or clinic.

Telehealth Visit
telephone visit

Nothing to see here

There's nothing in your list yet. Try adding a new record.


orgfarm-9c399a4996-dev-ed.develop.lightning.force.com/lightning/o/Appointment__c/new?count=1&nooverride=1&useRecordTypeCheck=1&navigationLocation=LIST_VIEW&u...
Verify it's you
Search...
Smart Healthcare H...
Appointmen
Recently Viewed
0 items • Updated 15 minutes ago
Import
Change Owner
Assign Label
Cancel
Save & New
Save

New Appointment: In-Person Visit

* = Required Information

Information

Appointment Number

Owner: Palla Bhugarbha

Doctor: Palla Bhugarbha

Appointment Date
Date: 9/26/2025 Time: 12:00 PM

Status: Shedulled

Patient: kalim Chintaguntla

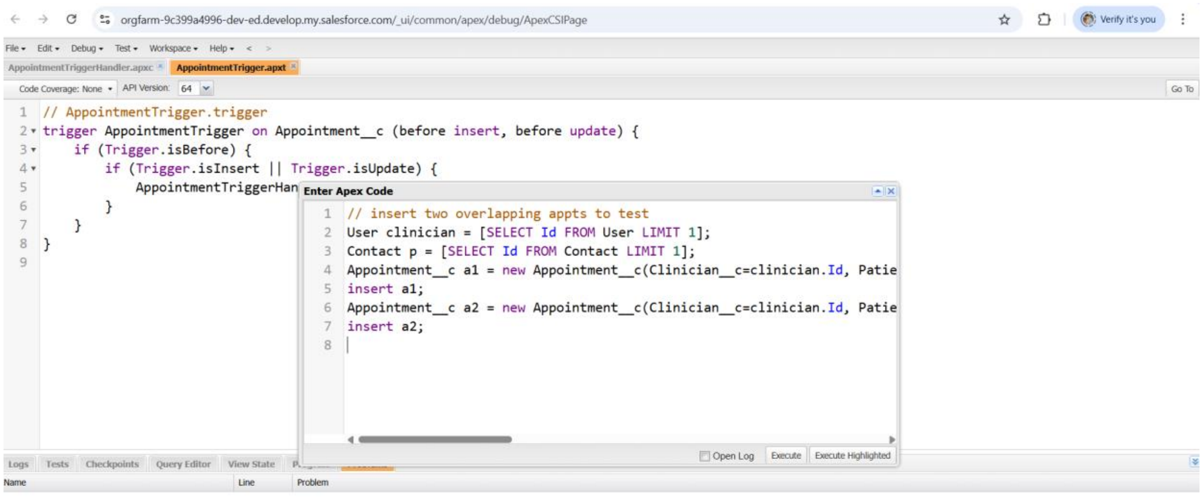
Start DateTime
Date: 9/26/2025 Time: 12:00 PM

End DateTime

3.5 Test the Double-Booking Trigger

- Created the **first appointment** for the same Clinician at 10:00 AM–11:00 AM.
- Tried to create a **second overlapping appointment** (10:30 AM–11:30 AM) with the same Clinician.
- On save, Salesforce displayed a validation error from the **AppointmentTrigger**.

- Verified that the trigger worked correctly by blocking the overlapping booking and captured the error message screenshot.



```
orgfarm-9c399a4996-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
```

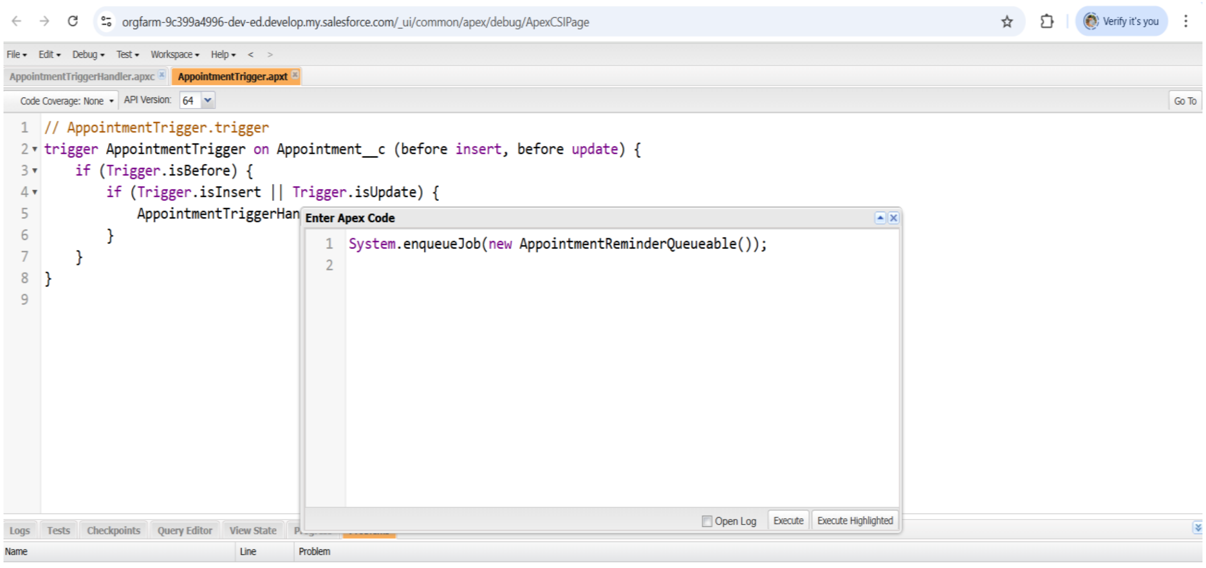
```
// AppointmentTrigger.trigger
trigger AppointmentTrigger on Appointment__c (before insert, before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {
            AppointmentTriggerHan
        }
    }
}
```

Enter Apex Code

```
// insert two overlapping appts to test
User clinician = [SELECT Id FROM User LIMIT 1];
Contact p = [SELECT Id FROM Contact LIMIT 1];
Appointment__c a1 = new Appointment__c(Clinician__c=clinician.Id, Patie
insert a1;
Appointment__c a2 = new Appointment__c(Clinician__c=clinician.Id, Patie
insert a2;
```

3.6 Queueable Apex

- Created *AppointmentReminderQueueable* class implementing *Queueable interface*.
- Queried appointments starting in next 24 hours and created reminder tasks.
- Executed the job with System.enqueueJob() via Execute Anonymous window.
- Verified results in *Apex Jobs* and by viewing created Tasks.



```
orgfarm-9c399a4996-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
```

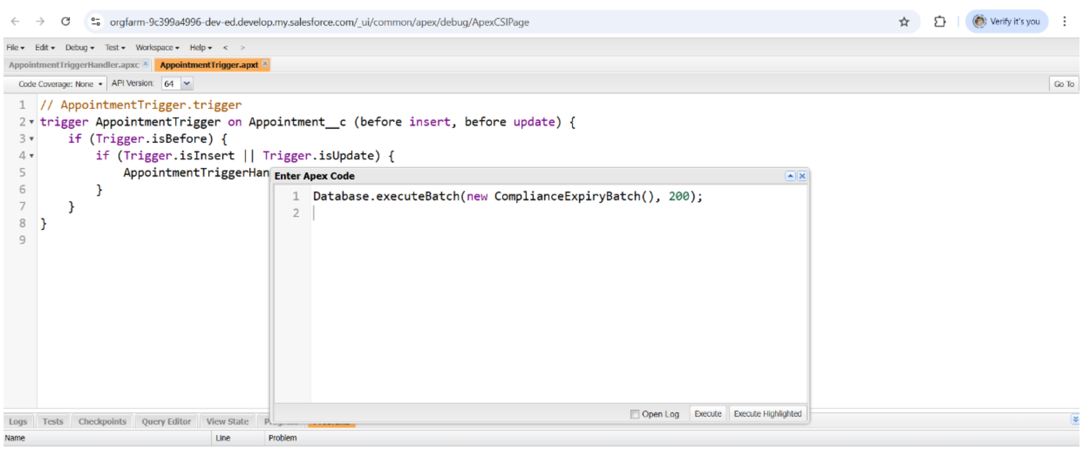
```
// AppointmentTrigger.trigger
trigger AppointmentTrigger on Appointment__c (before insert, before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {
            AppointmentTriggerHan
        }
    }
}
```

Enter Apex Code

```
System.enqueueJob(new AppointmentReminderQueueable());
```

3.7 Batch Apex

- Developed *ComplianceExpiryBatch* class implementing *Database.Batchable*.
- Queried compliance records with past expiry dates.
- Updated their status to **Expired** automatically.
- Ran the batch with Database.executeBatch() and confirmed results in Apex Jobs


orgfarm-9c399a4996-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

AppointmentTriggerHandler.apxc | AppointmentTrigger.apxt

Code Coverage: None   API Version: 64

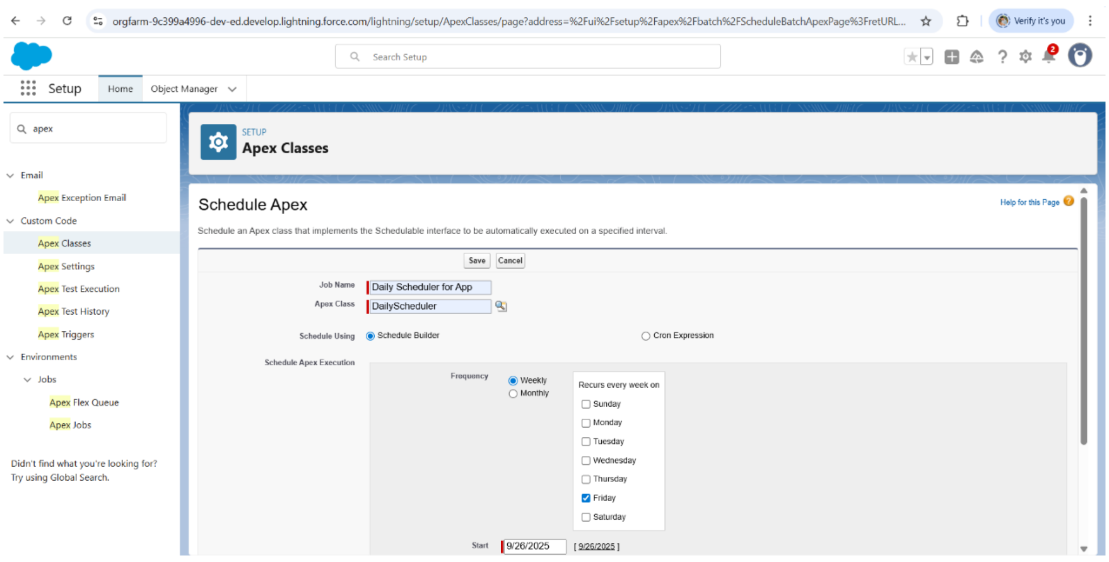
```
// AppointmentTrigger.trigger
trigger AppointmentTrigger on Appointment__c (before insert, before update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert || Trigger.isUpdate) {
            AppointmentTriggerHan
        }
    }
}
```

Enter Apex Code

```
Database.executeBatch(new ComplianceExpiryBatch(), 200);
```

3.8 Schedule the Daily Job

- Created *DailyScheduler* class implementing *Schedulable*.
- Combined *Batch job (compliance)* and *Queueable job (reminders)* in scheduler.
- Scheduled job in *Setup → Apex Classes → Schedule Apex*.
- Verified scheduled jobs in *Setup → Scheduled Jobs*.


orgfarm-9c399a4996-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/page?address=%2Fui%2Fsetup%2Fapex%2Fbatch%2FScheduleBatchApexPage%3FretURL...

SETUP
Apex Classes

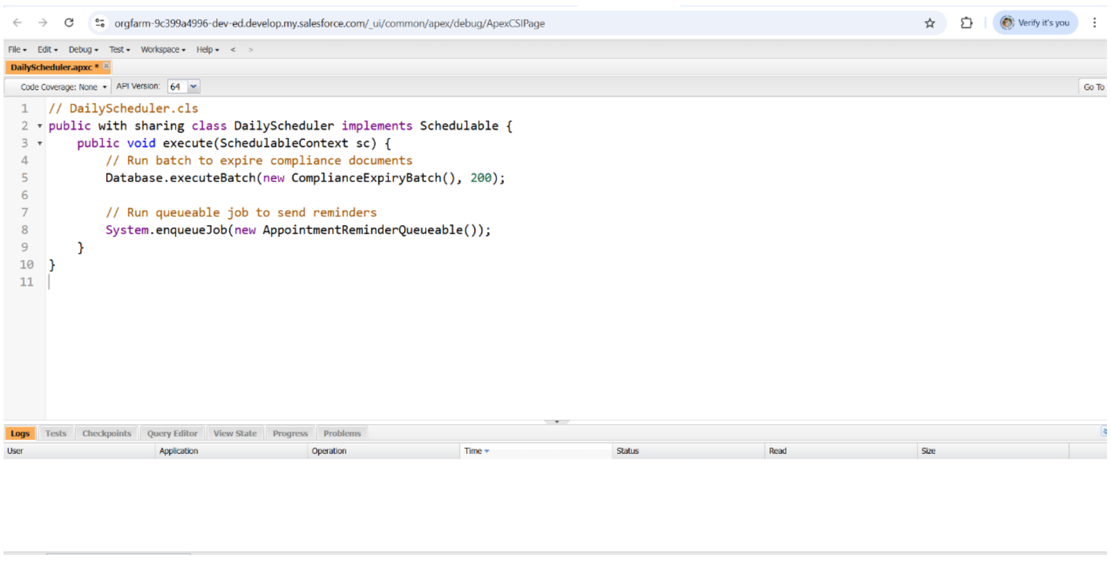
Schedule Apex

Schedule an Apex class that implements the Schedulable interface to be automatically executed on a specified interval.

Job Name	Daily Scheduler for App
Apex Class	DailyScheduler
Schedule Using	[✓ Schedule Builder] Cron Expression
Frequency	[✓ Weekly] Monthly
Recurs every week on	Sunday, Monday, Tuesday, Wednesday, Thursday, [✓ Friday], Saturday
Start	9/26/2025 [9/26/2025]

3.9 Create & Run Test Classes

- Created *ProjectApexTests* class with *@isTest* methods.
- Tested triggers by inserting overlapping appointments (ensuring errors fire).
- Verified batch and queueable execution inside *Test.startTest()/stopTest()*.
- Captured *Apex Test Execution results* with >75% coverage



```
// DailyScheduler.cls
public with sharing class DailyScheduler implements Schedulable {
    public void execute(SchedulableContext sc) {
        // Run batch to expire compliance documents
        Database.executeBatch(new ComplianceExpiryBatch(), 200);

        // Run queueable job to send reminders
        System.enqueueJob(new AppointmentReminderQueueable());
    }
}
```

4. Expected Outcomes

- Apex triggers and classes successfully enforce **core healthcare rules** such as preventing clinician double-booking and automatically marking expired compliance records.
- **Queueable and Batch Apex** jobs handle appointment reminders and compliance checks efficiently, ensuring scalability.
- **Scheduled Apex jobs** run daily to automate reminders and compliance updates without manual intervention.
- Robust **test classes** achieve the required coverage (>75%), enabling safe deployment and reliability.
- The system is now prepared for **integration and final deployment**, with enterprise-level automation established.

5. Conclusion

Phase 5 enhanced the Smart Healthcare Appointment & Compliance Hub with advanced Apex programming capabilities. By implementing triggers, classes, asynchronous jobs, and scheduled automation, the project now enforces complex healthcare processes and handles large volumes of data reliably. With thorough testing and automation in place, the system is ready for integration, ensuring both compliance and patient care efficiency.