

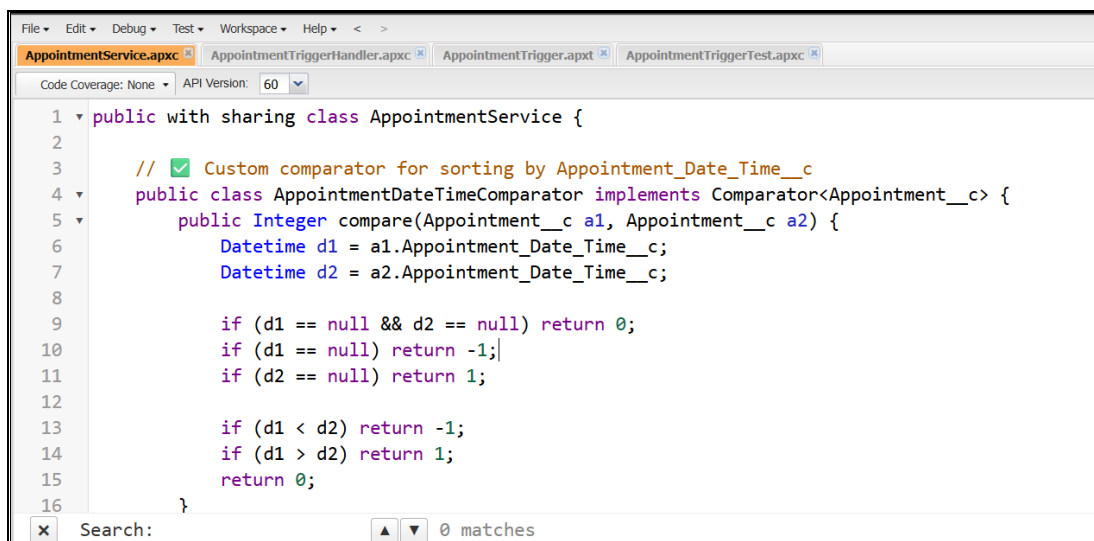
MediCare Connect — Phase 5: Apex Programming (Developer)

Goal: Add server-side automation and validations that cannot be done by declarative tools. Implement custom Apex classes, triggers, test classes and a batch job to enforce appointment rules and send daily reminders.

1. Classes & Objects

Created an Apex utility class **AppointmentService**:

- **Inner Comparator:** **AppointmentDateTimeComparator** sorts appointments by **Appointment__Date_Time__c**.
- **Static Method:** **preventOverlap(List<Appointment__c> incoming, Map<Id,Appointment__c> oldMap)** checks new/updated appointments against existing appointments for each doctor and throws **addError()** if overlaps are found.

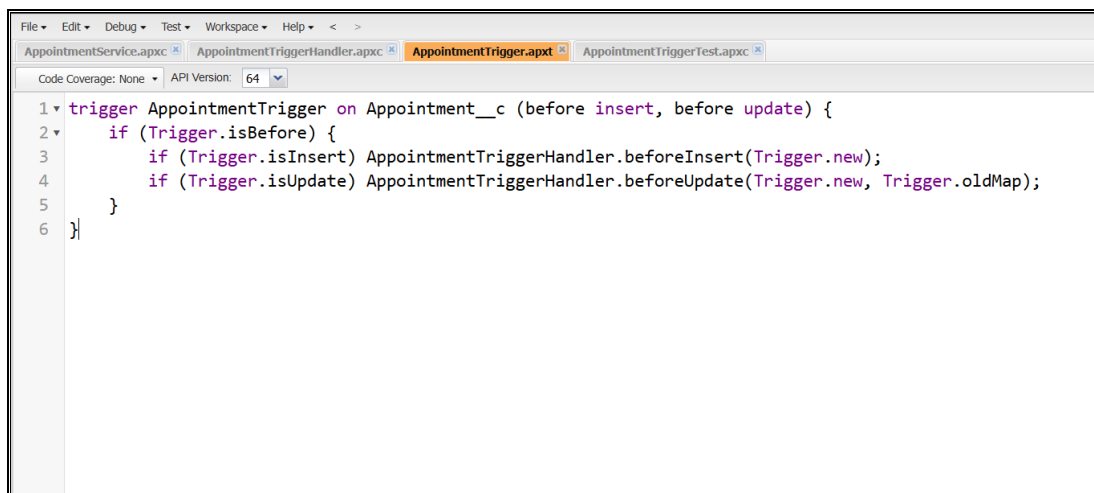


```
1 public with sharing class AppointmentService {
2
3     // Custom comparator for sorting by Appointment__Date_Time__c
4     public class AppointmentDateTimeComparator implements Comparator<Appointment__c> {
5         public Integer compare(Appointment__c a1, Appointment__c a2) {
6             Datetime d1 = a1.Appointment__Date_Time__c;
7             Datetime d2 = a2.Appointment__Date_Time__c;
8
9             if (d1 == null && d2 == null) return 0;
10            if (d1 == null) return -1;
11            if (d2 == null) return 1;
12
13            if (d1 < d2) return -1;
14            if (d1 > d2) return 1;
15            return 0;
16        }
17    }
18 }
```

Fig.1 AppointmentService

2. Apex Triggers

Trigger **AppointmentTrigger** on Appointment__c:



```
1 trigger AppointmentTrigger on Appointment__c (before insert, before update) {
2     if (Trigger.isBefore) {
3         if (Trigger.isInsert) AppointmentTriggerHandler.beforeInsert(Trigger.new);
4         if (Trigger.isUpdate) AppointmentTriggerHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
5     }
6 }
```

Fig.2 AppointmentTrigger

3. Trigger Design Pattern

Business logic kept in **AppointmentService** class; trigger simply calls the method.

4. SOQL & Collections

Inside preventOverlap:

- **SOQL** queries existing appointments for relevant doctors.
- **Collections** (**Set<Id>**, **Map<Id, List<Appointment__c>>**) used to group and compare appointments.



Fig.3 SOQL_Collections

5. Control Statements

for loops, if conditions and **addError()** to prevent overlaps.

6. Test Classes

A test class **AppointmentTriggerTest** was created to:

- Insert a valid appointment (no overlap) and verify it saves successfully.
- Attempt to insert an overlapping appointment and verify a **DmlException** with “overlaps” in the message is thrown.

This test indirectly executes the logic in **AppointmentService.preventOverlap** and gives code coverage for both the trigger and the service class.

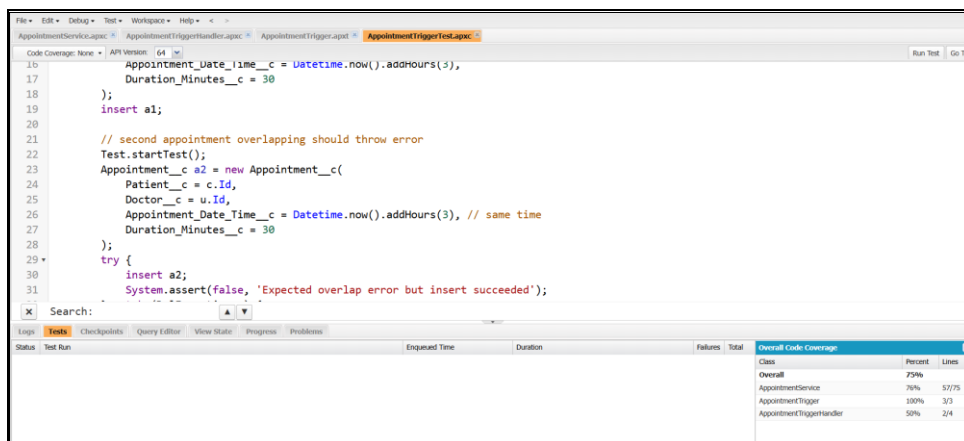


Fig.4 AppointmentTriggerTest

7. Batch Apex and Scheduled Jobs

A Batch Apex class **DailyAppointmentSummaryBatch** was created to query upcoming appointments for each doctor and send reminder notifications/emails every morning.

To automate this, the class was scheduled using **Schedule Apex** in Salesforce Setup. The job **Notify Doctor** runs the batch daily at **6:00 AM** on all days of the week.

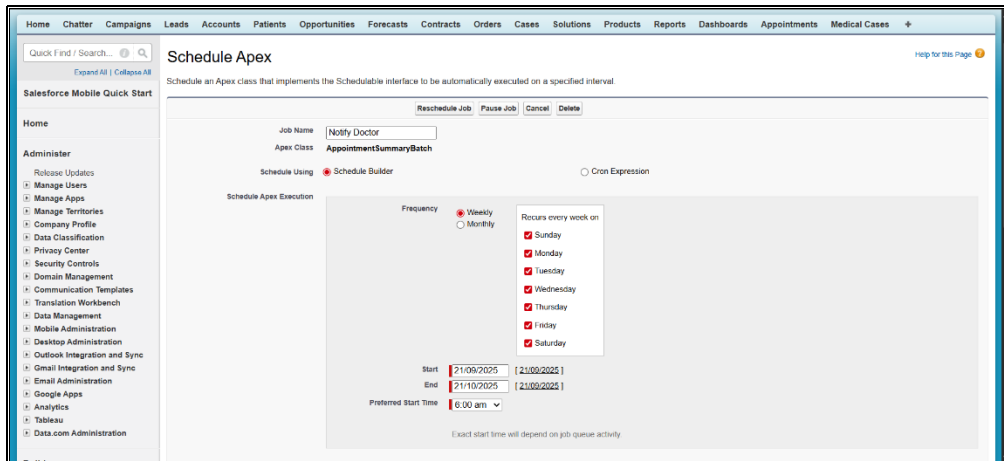


Fig.5 Schedule Apex

8. Exception Handling

Used **addError()** on SObjects to stop DML with a user-friendly message.

9. Skipped / Not Implemented

Queueable Apex, Future Methods, Asynchronous Processing beyond the daily batch job were not implemented at this phase.

10. Testing & Results

- Overlapping appointments throw error.
- Non-overlapping appointments save successfully.
- Daily batch job runs in the morning and sends reminders to doctors.

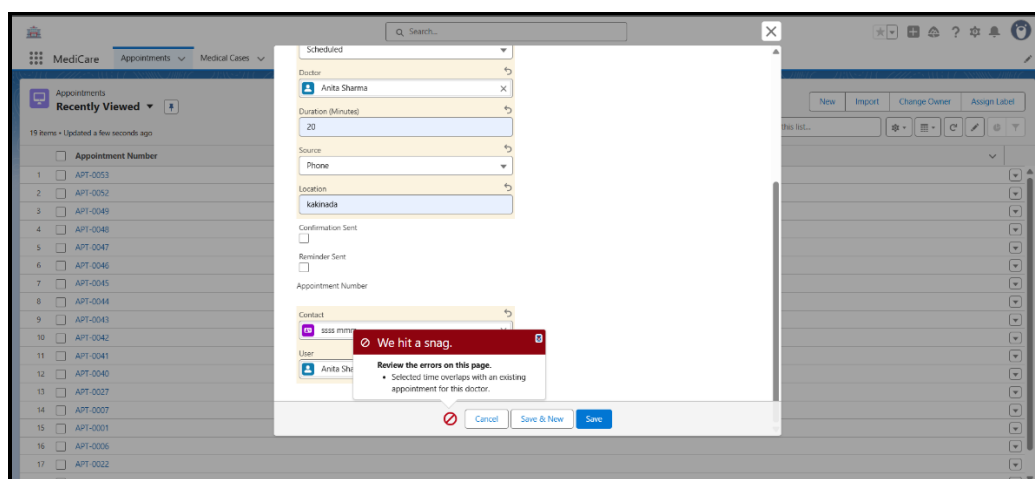


Fig.6 Appointment Overlapping Result