Phase 5: Apex Programming (Developer)

AI-Enabled Hospital & Pharmacy Management System

Goal: The goal of this phase is to implement programmatic logic using Apex to handle complex business requirements that could not be achieved by declarative tools alone. This includes working with Apex classes, triggers, SOQL & SOSL queries, data structures, control statements, and asynchronous processes like Batch Apex, Queueable Apex, Scheduled Apex, and Future methods. The aim is to build a robust backend capable of automating workflows such as updating medicine orders, processing appointments, handling expired inventory, and sending notifications.

Tasks in Phase 5:

- Classes & Objects
- Apex Triggers (before/after insert/update/delete)
- Trigger Design Pattern
- SOQL & SOSL
- Collections: List, Set, Map
- Control Statements
- Batch Apex
- Queueable Apex
- Scheduled Apex
- Future Methods
- Exception Handling
- Test Classes
- Asynchronous Processing

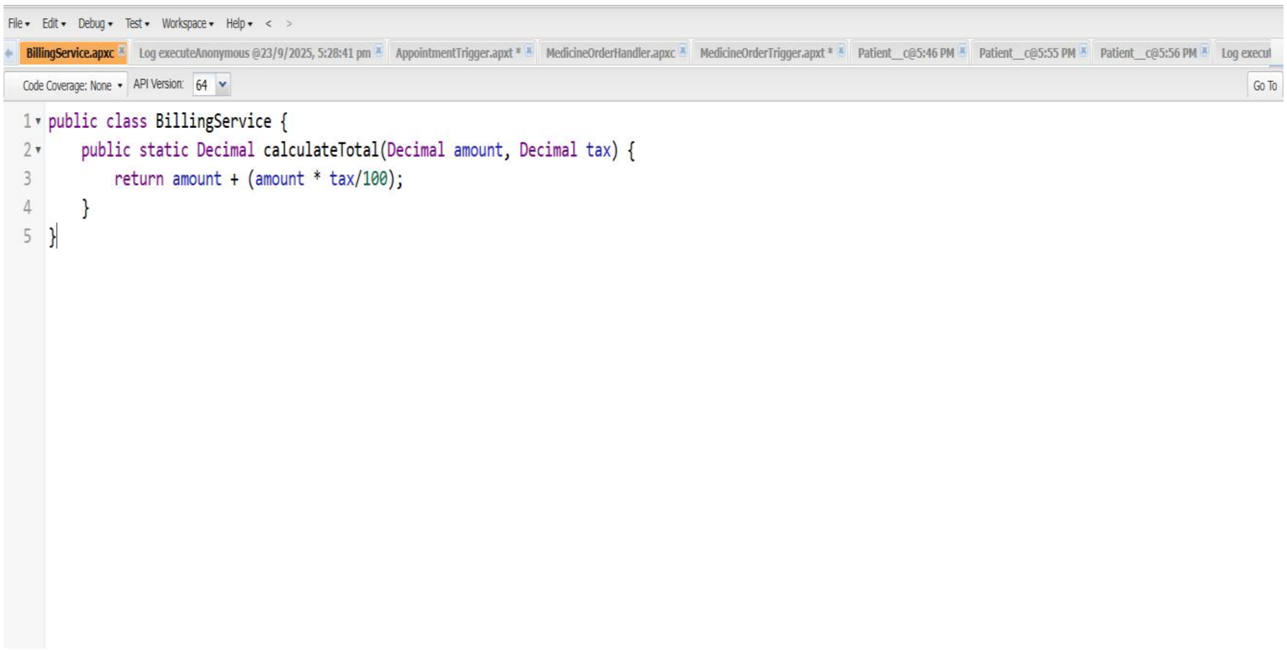
Classes & Object

What I implemented

A small service class to centralize billing logic (BillingService) so business math is reusable and testable.

How & why

Encapsulated the billing formula in a static method to keep logic out of triggers and UI code:

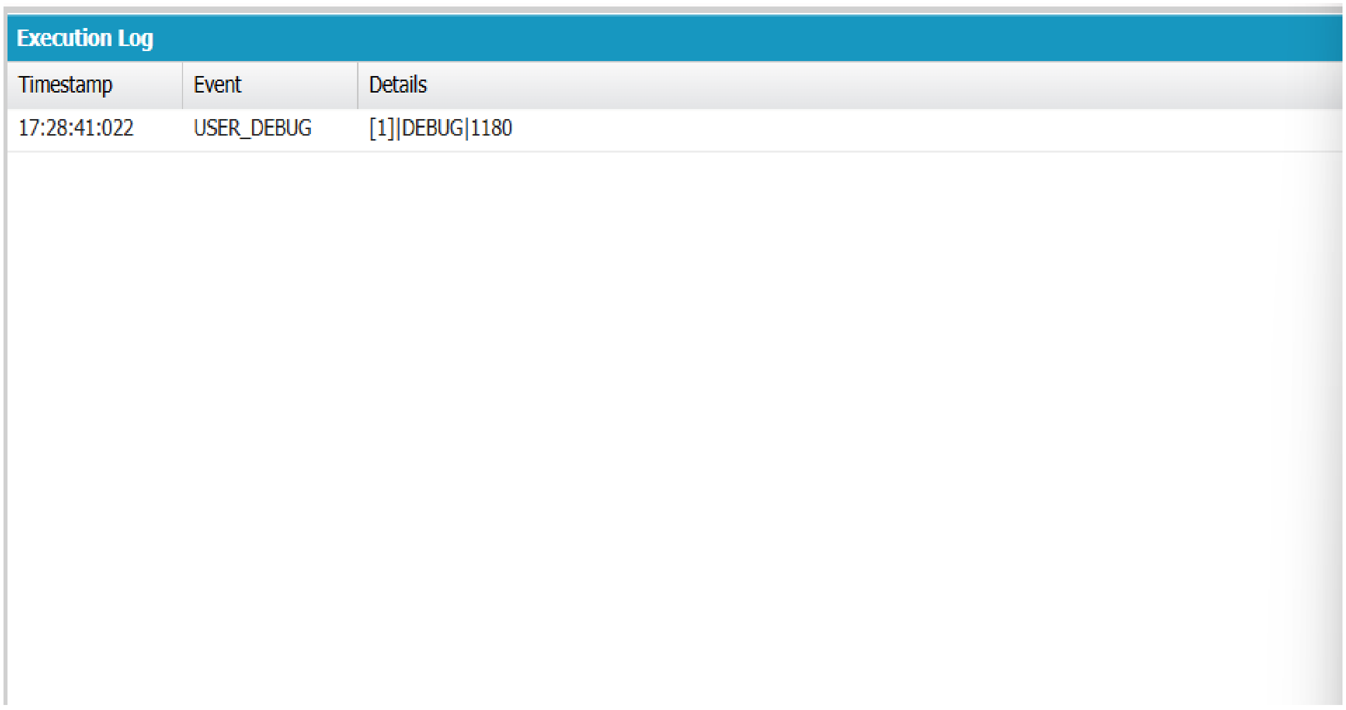

File
Edit
Debug
Test
Workspace
Help
BillingService.apxc
Log executeAnonymous @23/9/2025, 5:28:41 pm
AppointmentTrigger.apxt *
MedicineOrderHandler.apxc
MedicineOrderTrigger.apxt *
Patient__c@5:46 PM
Patient__c@5:55 PM
Patient__c@5:56 PM
Go To

Code Coverage: None API Version: 64

```
public class BillingService {
    public static Decimal calculateTotal(Decimal amount, Decimal tax) {
        return amount + (amount * tax/100);
    }
}
```

Validation

Run in Developer Console (Execute Anonymous):



Execution Log

Timestamp	Event	Details
17:28:41:022	USER_DEBUG	[1]|DEBUG|1180

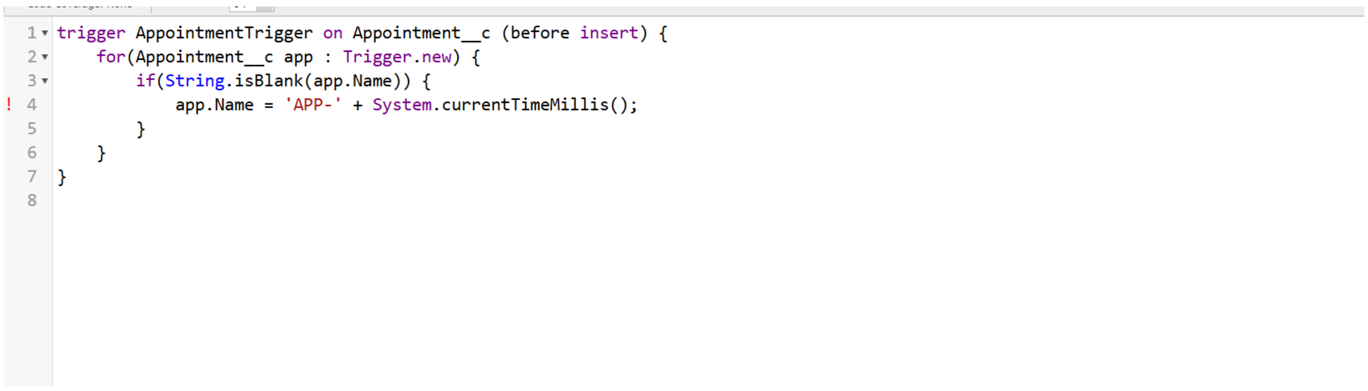
2. Apex Triggers

What I implemented

Triggers to automate record-level behaviour.

How & why

Used a before insert trigger to set a default Name when none provided so that data is consistent and predictable:



```
trigger AppointmentTrigger on Appointment__c (before insert) {
    for(Appointment__c app : Trigger.new) {
        if(String.isBlank(app.Name)) {
            app.Name = 'APP-' + System.currentTimeMillis();
        }
    }
}
```

Validation

Inserted an Appointment__c. Verified the Name field is generated in debug logs / record detail.

---

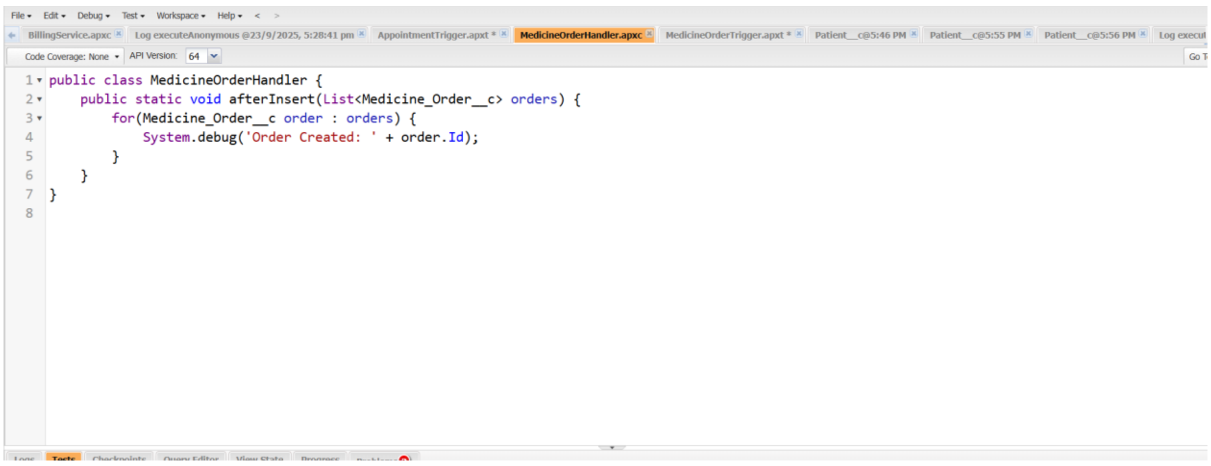
3. Trigger Design Pattern

What I implemented

Separated trigger entry points and business logic into a handler class (MedicineOrderHandler).

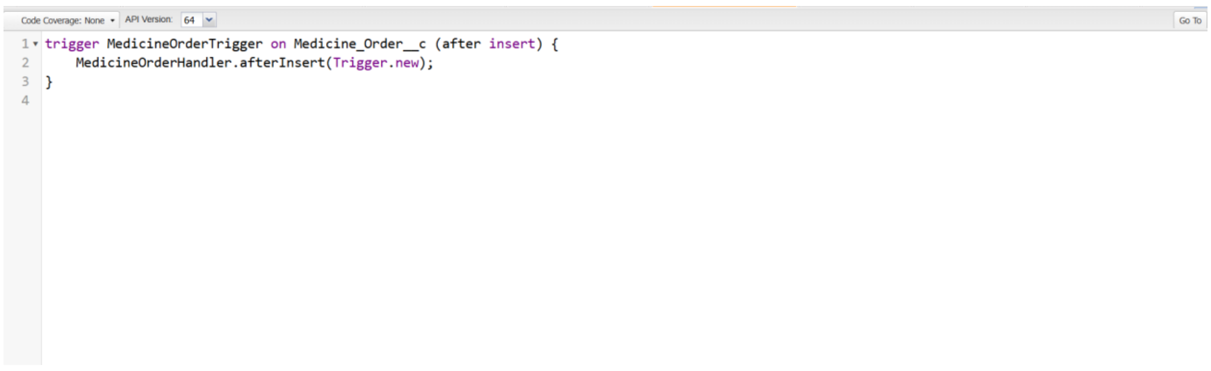
How & why

Trigger delegates to handler methods (keeps triggers lightweight, supports bulkification and unit testing):

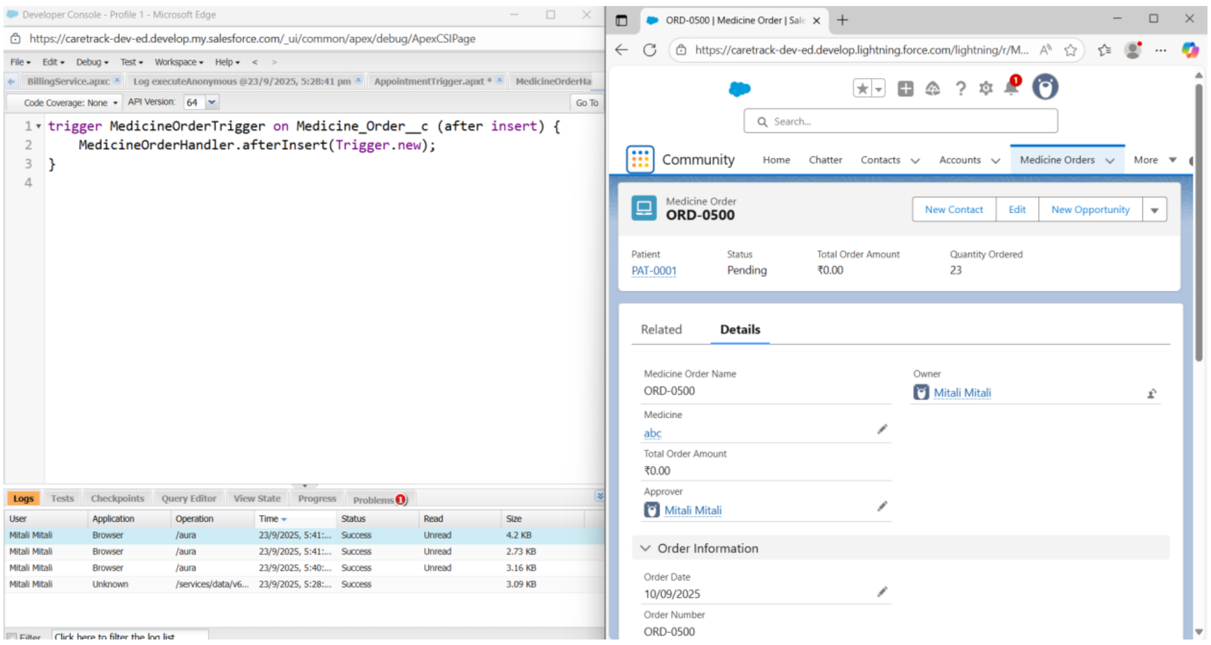


Code Coverage: None API Version: 64

```
public class MedicineOrderHandler {
    public static void afterInsert(List<Medicine_Order__c> orders) {
        for(Medicine_Order__c order : orders) {
            System.debug('Order Created: ' + order.Id);
        }
    }
}
```



```
trigger MedicineOrderTrigger on Medicine_Order__c (after insert) {
    MedicineOrderHandler.afterInsert(Trigger.new);
}
```



Developer Console - Profile 1 - Microsoft Edge
https://caretrack-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
BillingService.apxc
Log executeAnonymous @23/9/2025, 5:28:41 pm
AppointmentTrigger.apxt *
MedicineOrderHa
Code Coverage: None
API Version: 64
Go To
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems
Filter
Click here to filter the log list
ORD-0500 | Medicine Order | Sale
https://caretrack-dev-ed.develop.lightning.force.com/lightning/r/M...
Search...
Community
Home
Chatter
Contacts
Accounts
Medicine Orders
More
New Contact
Edit
New Opportunity
Related
Details

```
trigger MedicineOrderTrigger on Medicine_Order__c (after insert) {
    MedicineOrderHandler.afterInsert(Trigger.new);
}
```

User	Application	Operation	Time	Status	Read	Size
Mitali Mitali	Browser	/aura	23/9/2025, 5:41:...	Success	Unread	4.2 KB
Mitali Mitali	Browser	/aura	23/9/2025, 5:41:...	Success	Unread	2.73 KB
Mitali Mitali	Browser	/aura	23/9/2025, 5:40:...	Success	Unread	3.16 KB
Mitali Mitali	Unknown	/services/data/v6...	23/9/2025, 5:28:...	Success		3.09 KB

Medicine Order
ORD-0500

Patient	Status	Total Order Amount	Quantity Ordered
PAT-0001	Pending	₹0.00	23

Medicine Order Name: ORD-0500
Owner: Mitali Mitali
Medicine: abc
Total Order Amount: ₹0.00
Approver: Mitali Mitali

Order Information
Order Date: 10/09/2025
Order Number: ORD-0500

Validation
Inserted sample Medicine_Order__c records and reviewed debug logs confirming handler execution.

---

4. SOQL & SOSL

What I implemented
Used SOQL for targeted queries and SOSL for multi-field searching.

How & why
SOQL:

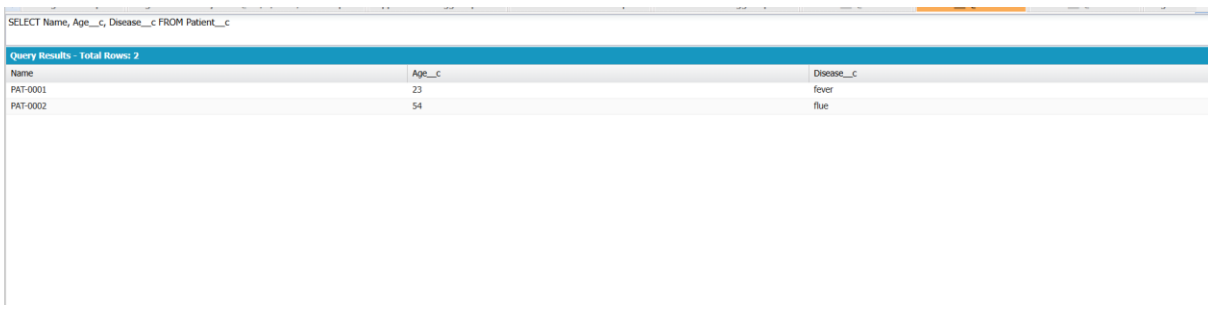
List<Patient__c> fluPatients =

  [SELECT Name, Age__c FROM Patient__c WHERE Disease__c = 'Flu'];

SOSL:

List<List<sObject>> results = [FIND 'flue' IN PHONE FIELDS RETURNING Patient__c(Name, Phone__c)];

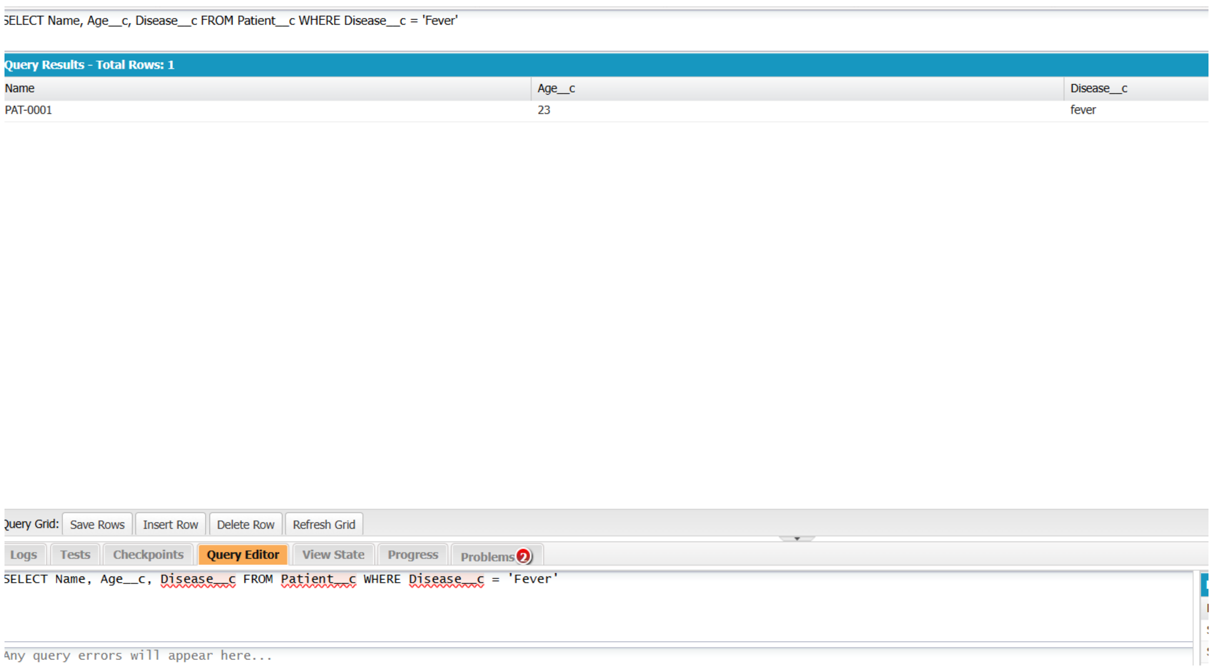
Validation
Ran queries in Developer Console → Query Editor; verified returned rows and field values.



```
SELECT Name, Age__c, Disease__c FROM Patient__c
```

Query Results - Total Rows: 2

Name	Age__c	Disease__c
PAT-0001	23	fever
PAT-0002	54	flue



```
SELECT Name, Age__c, Disease__c FROM Patient__c WHERE Disease__c = 'Fever'
```

Query Results - Total Rows: 1

Name	Age__c	Disease__c
PAT-0001	23	fever

```
SELECT Name, Age__c, Disease__c FROM Patient__c WHERE Disease__c = 'Fever'
```

```
Any query errors will appear here...
```

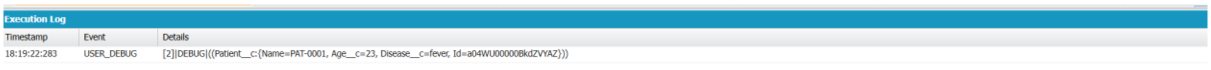
5. Collections: List, Set, Map

Used Lists, Sets and Maps for efficient data handling and deduplication.

- List<String> for ordered doctor names.
- Set<String> to keep unique disease names (no duplicates).
- Map<Id, Patient__c> to look up patients by Id (O(1) lookups).
- Executed in Execute Anonymous

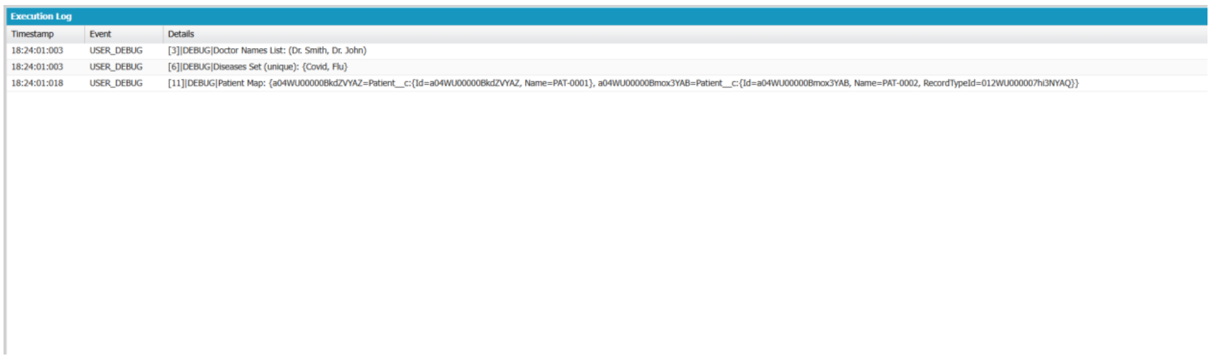
```
List<String> doctorNames = new List<String>{'Dr. Smith','Dr. John'};

Set<String> diseases = new Set<String>{'Flu','Covid','Flu'};

Map<Id, Patient__c> patientMap = new Map<Id, Patient__c>([SELECT Id, Name
FROM Patient__c]);


for(Patient__c p : [SELECT Name, Age__c FROM Patient__c]) {

   if(p.Age__c < 18) {
```

```
                System.debug(p.Name + ' is a minor');
            }
        }
```



Execution Log

```
Timestamp      Event        Details
18:19:22:283   USER_DEBUG   [2]|DEBUG|((Patient__c:{Name=PAT-0001, Age__c=23, Disease__c=fever, Id=a04WU00000BkdZVYAZ}))
```



Execution Log

```
Timestamp      Event        Details
18:24:01:003   USER_DEBUG   [3]|DEBUG|Doctor Names List: (Dr. Smith, Dr. John)
18:24:01:003   USER_DEBUG   [6]|DEBUG|Diseases Set (unique): {Covid, Flu}
18:24:01:018   USER_DEBUG   [11]|DEBUG|Patient Map: {a04WU00000BkdZVYAZ=Patient__c:{Id=a04WU00000BkdZVYAZ, Name=PAT-0001}, a04WU00000Bmox3YAB=Patient__c:{Id=a04WU00000Bmox3YAB, Name=PAT-0002, RecordTypeId=012WU000007hi3NYAQ}}
```

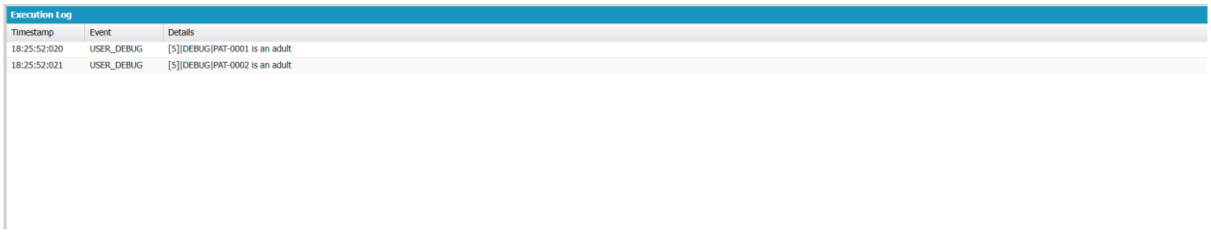
---

6. Control Statements

Applied if/else and loops in Apex logic to make decisions per record.

```
for(Patient__c p : [SELECT Name, Age__c FROM Patient__c]) {
    if(p.Age__c < 18) {
        System.debug(p.Name + ' is a minor');
    } else {
        System.debug(p.Name + ' is an adult');
    }
}
```

Checked debug output for expected branches executing in sample data.

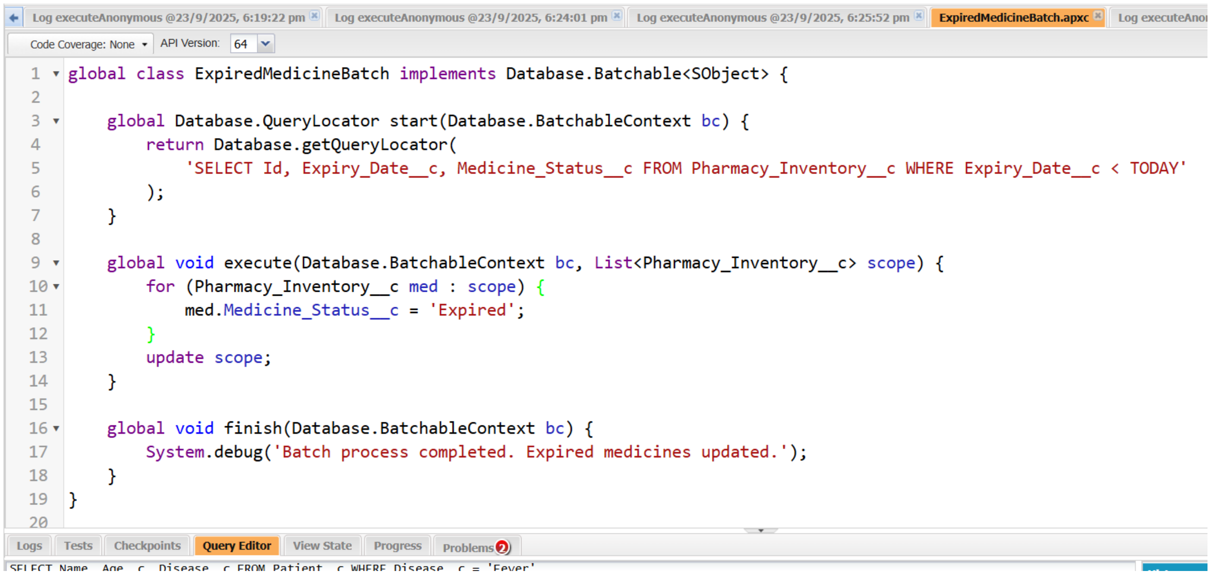


```
Timestamp      Event        Details
18:25:52:020   USER_DEBUG   [5]|DEBUG|PAT-0001 is an adult
18:25:52:021   USER_DEBUG   [5]|DEBUG|PAT-0002 is an adult
```

7. Batch Apex

ExpiredMedicineBatch to bulk-update inventory records whose expiry date has passed.

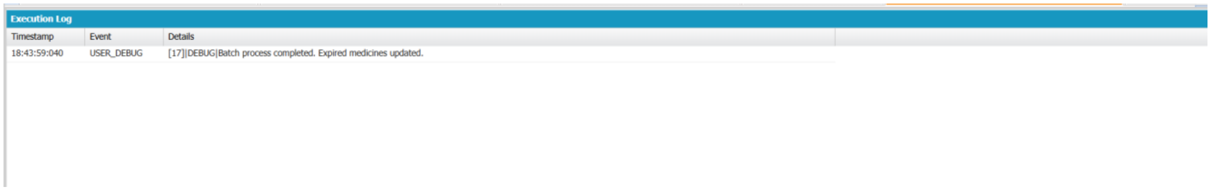
Batchable handles millions of records in chunks to avoid governor limits:



Log executeAnonymous @23/9/2025, 6:19:22 pm
Log executeAnonymous @23/9/2025, 6:24:01 pm
Log executeAnonymous @23/9/2025, 6:25:52 pm
ExpiredMedicineBatch.apxc
Code Coverage: None
API Version: 64
Logs
Tests
Checkpoints
Query Editor
View State
Progress
Problems

```
global class ExpiredMedicineBatch implements Database.Batchable<SObject> {

    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator(
            'SELECT Id, Expiry_Date__c, Medicine_Status__c FROM Pharmacy_Inventory__c WHERE Expiry_Date__c < TODAY'
        );
    }

    global void execute(Database.BatchableContext bc, List<Pharmacy_Inventory__c> scope) {
        for (Pharmacy_Inventory__c med : scope) {
            med.Medicine_Status__c = 'Expired';
        }
        update scope;
    }

    global void finish(Database.BatchableContext bc) {
        System.debug('Batch process completed. Expired medicines updated.');
    }
}
```

Executed: Database.executeBatch(new ExpiredMedicineBatch(), 100);
Validated records’ Status__c updated to Expired and reviewed Execution Log.



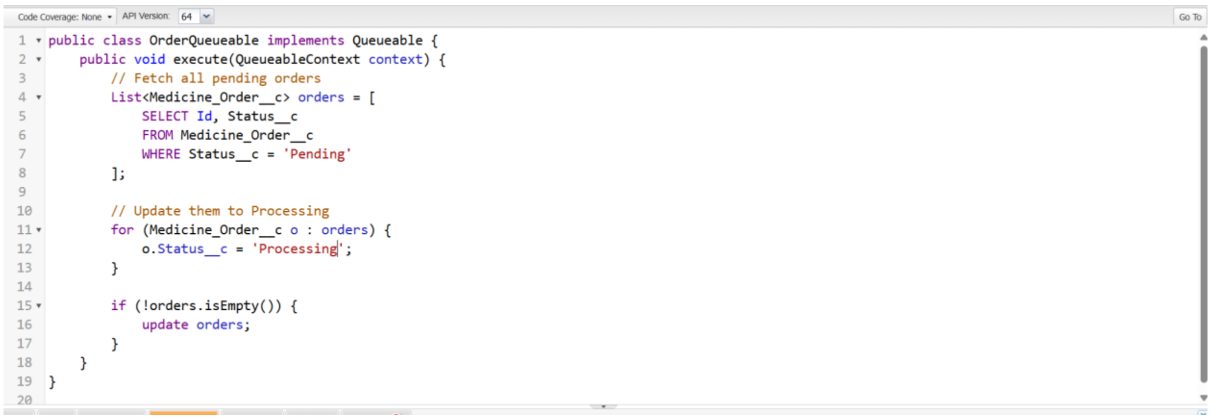
```
Timestamp      Event        Details
18:43:59:040   USER_DEBUG   [17]|DEBUG|Batch process completed. Expired medicines updated.
```

8. Queueable Apex

OrderQueueable to asynchronously process pending medicine orders.
Queueable is ideal for straightforward asynchronous tasks and can be chained:

System.enqueueJob(new OrderQueueable());

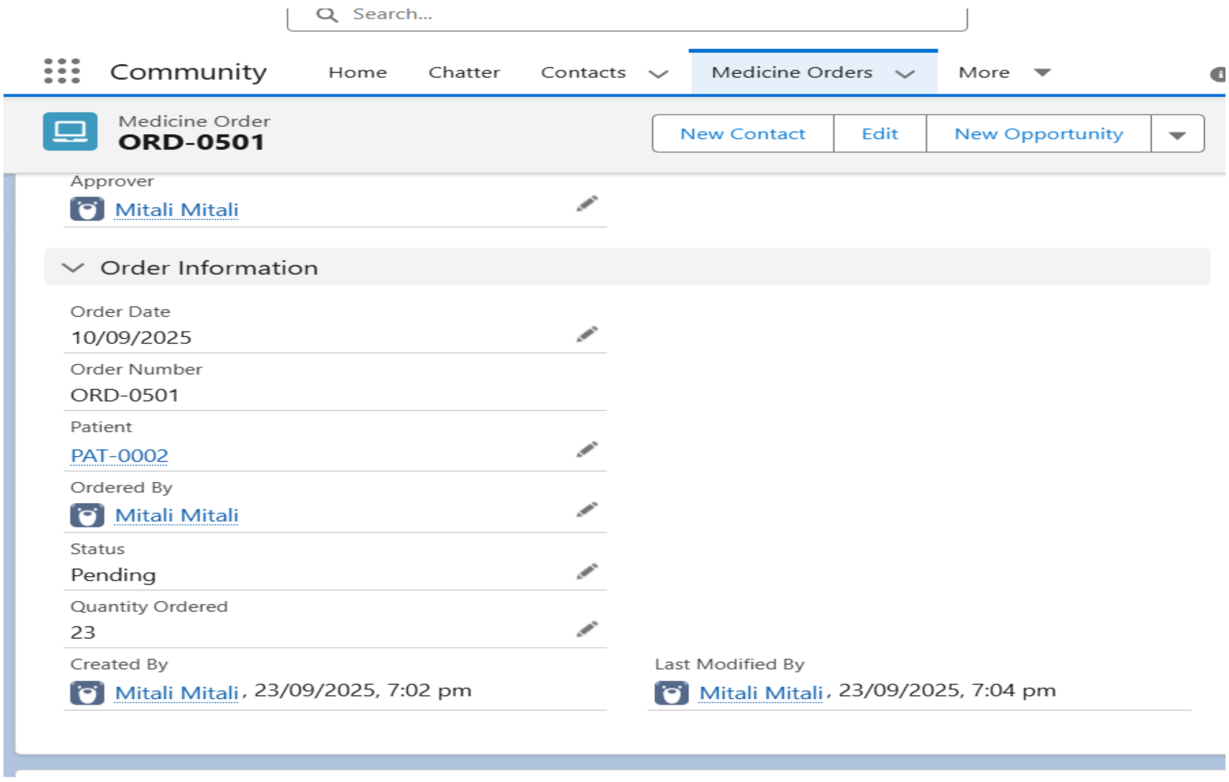
all those with Status = Pending should now be updated to Processing.



```
public class OrderQueueable implements Queueable {
    public void execute(QueueableContext context) {
        // Fetch all pending orders
        List<Medicine_Order__c> orders = [
            SELECT Id, Status__c
            FROM Medicine_Order__c
            WHERE Status__c = 'Pending'
        ];

        // Update them to Processing
        for (Medicine_Order__c o : orders) {
            o.Status__c = 'Processing';
        }

        if (!orders.isEmpty()) {
            update orders;
        }
    }
}
```

Ran System.enqueueJob from Execute Anonymous; inspected async job status in Setup → Apex Jobs and validated order records updated.

Before=



Medicine Order
ORD-0501

Approver
Mitali Mitali

Order Information

Order Date
10/09/2025

Order Number
ORD-0501

Patient
PAT-0002

Ordered By
Mitali Mitali

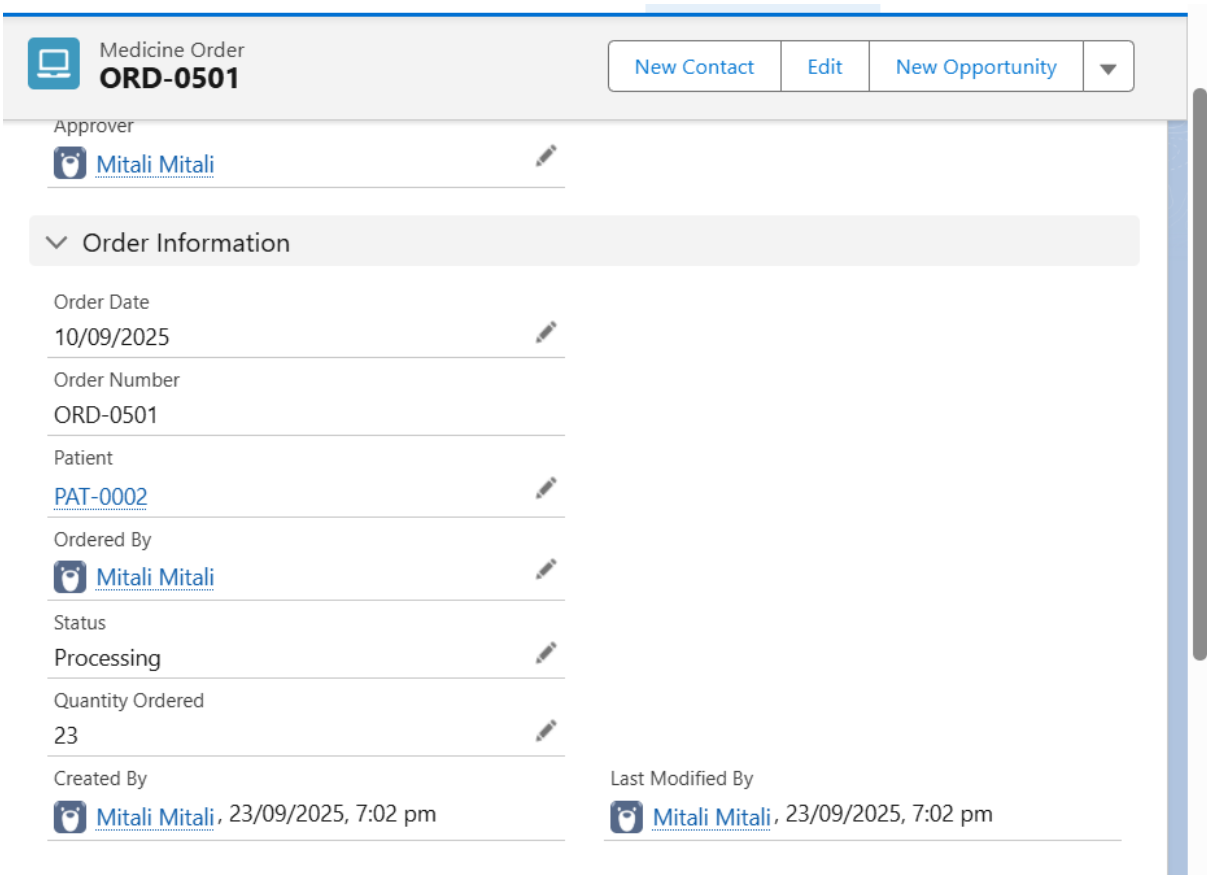
Status
Pending

Quantity Ordered
23

Created By
Mitali Mitali, 23/09/2025, 7:02 pm

Last Modified By
Mitali Mitali, 23/09/2025, 7:04 pm

After=


Medicine Order
ORD-0501

Approver
Mitali Mitali

Order Information

Order Date
10/09/2025

Order Number
ORD-0501

Patient
PAT-0002

Ordered By
Mitali Mitali

Status
Processing

Quantity Ordered
23

Created By
Mitali Mitali, 23/09/2025, 7:02 pm

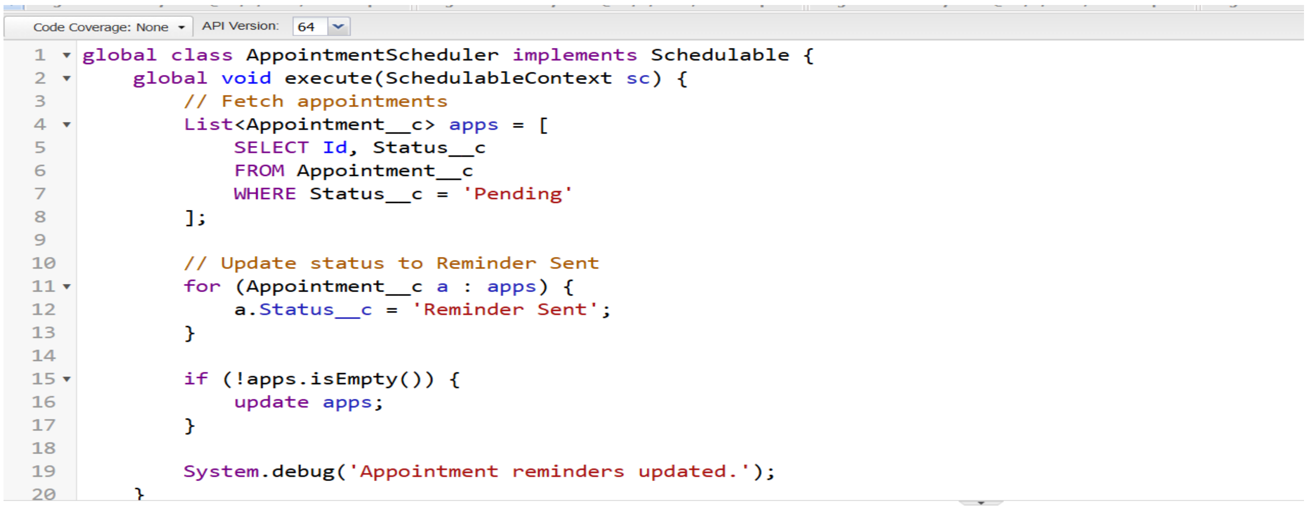
Last Modified By
Mitali Mitali, 23/09/2025, 7:02 pm

9. Scheduled Apex

AppointmentScheduler to send daily reminders and update appointment statuses. Scheduled Apex runs on a CRON expression; automates time-based processes:

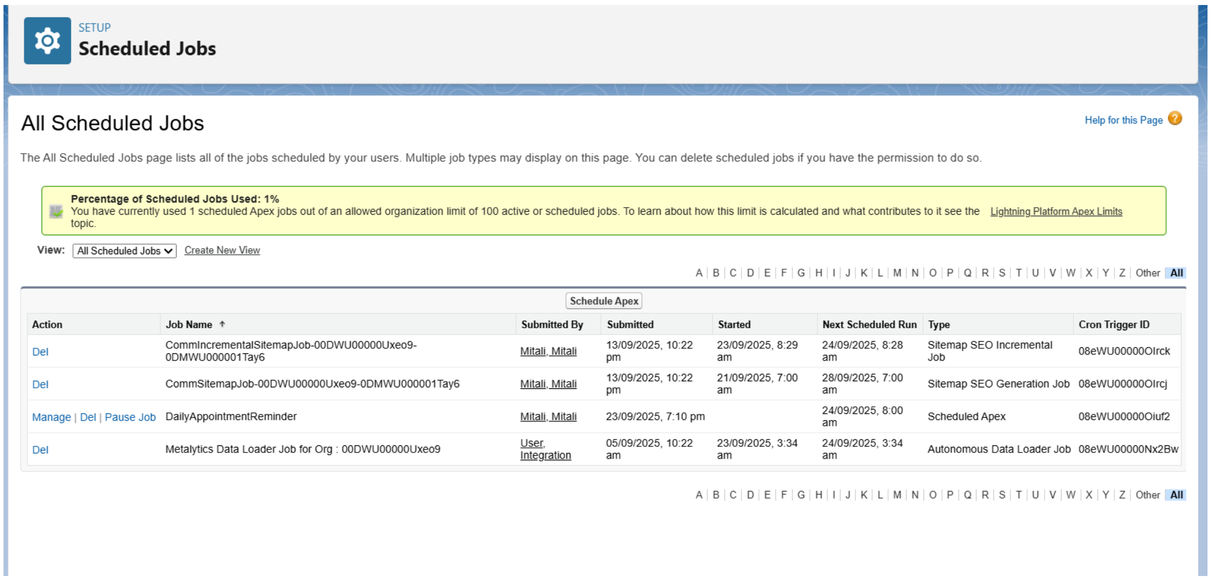
String cron = '0 0 9 * * ?'; // daily at 9 AM

System.schedule('Daily Appointment Reminder', cron, new AppointmentScheduler());


Code Coverage: None API Version: 64

```
global class AppointmentScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        // Fetch appointments
        List<Appointment__c> apps = [
            SELECT Id, Status__c
            FROM Appointment__c
            WHERE Status__c = 'Pending'
        ];

        // Update status to Reminder Sent
        for (Appointment__c a : apps) {
            a.Status__c = 'Reminder Sent';
        }

        if (!apps.isEmpty()) {
            update apps;
        }

        System.debug('Appointment reminders updated.');
    }
```

Scheduled job appeared in Setup → Scheduled Jobs; logs showed daily runs and updated appointment statuses.



SETUP

Scheduled Jobs

All Scheduled Jobs

Help for this Page

The All Scheduled Jobs page lists all of the jobs scheduled by your users. Multiple job types may display on this page. You can delete scheduled jobs if you have the permission to do so.

Percentage of Scheduled Jobs Used: 1%
You have currently used 1 scheduled Apex jobs out of an allowed organization limit of 100 active or scheduled jobs. To learn about how this limit is calculated and what contributes to it see the Lightning Platform Apex Limits topic.

View: All Scheduled Jobs | 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

Schedule Apex

Action	Job Name ↑	Submitted By	Submitted	Started	Next Scheduled Run	Type	Cron Trigger ID
Del	CommIncrementalSitemapJob-00DWU00000Uxeo9-0DMWU000001Tay6	Mitali, Mitali	13/09/2025, 10:22 pm	23/09/2025, 8:29 am	24/09/2025, 8:28 am	Sitemap SEO Incremental Job	08eWU00000OIrck
Del	CommSitemapJob-00DWU00000Uxeo9-0DMWU000001Tay6	Mitali, Mitali	13/09/2025, 10:22 pm	21/09/2025, 7:00 am	28/09/2025, 7:00 am	Sitemap SEO Generation Job	08eWU00000OIrcj
Manage | Del | Pause Job	DailyAppointmentReminder	Mitali, Mitali	23/09/2025, 7:10 pm		24/09/2025, 8:00 am	Scheduled Apex	08eWU00000Oiuf2
Del	Metalytics Data Loader Job for Org : 00DWU00000Uxeo9	User, Integration	05/09/2025, 10:22 am	23/09/2025, 3:34 am	24/09/2025, 3:34 am	Autonomous Data Loader Job	08eWU00000Nx2Bw

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

---

10. Future Methods

NotificationService.sendNotification(Id appointmentId) as @future for non-blocking notifications.
Used to offload external calls or lightweight background tasks to avoid slowing the user transaction.
Called via Execute Anonymous (sample ID) and verified the job in Apex Jobs and debug logs.

---

11. Exception Handling

What I implemented

Added try / catch blocks around risky operations and logged exceptions.

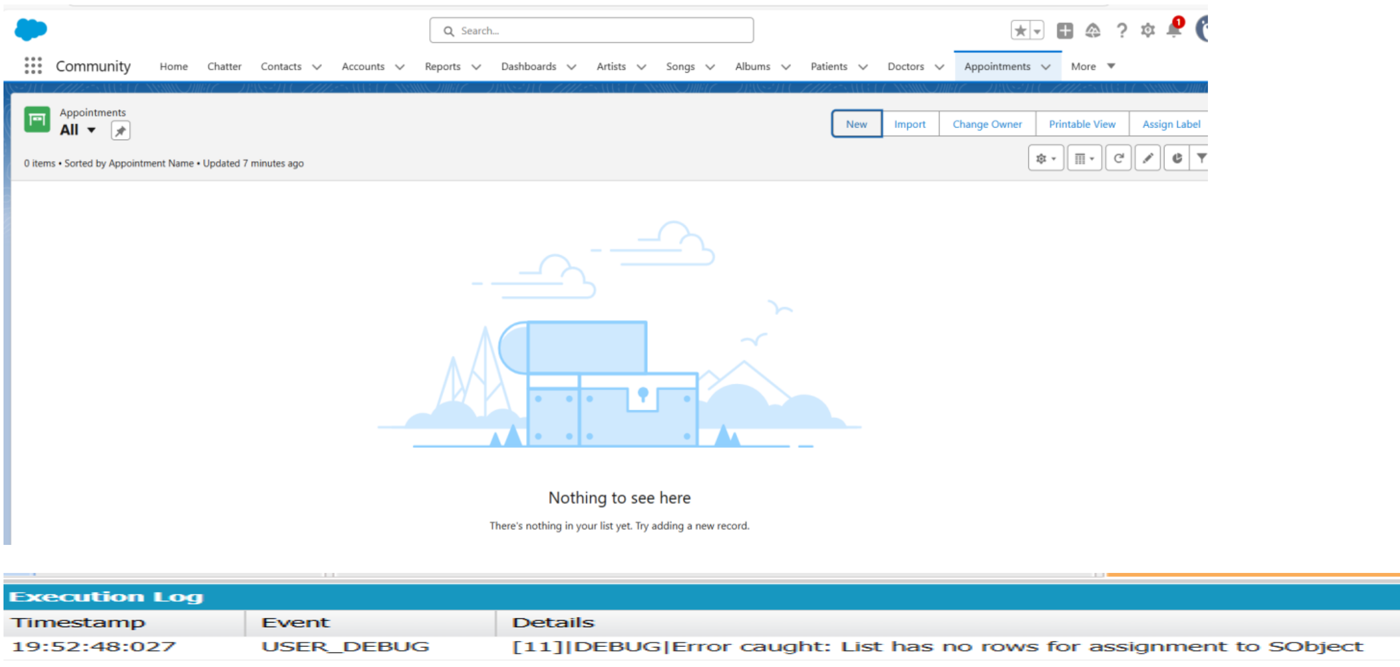
How & why

Prevents unhandled exceptions, logs meaningful messages for troubleshooting:

```
try {

  // risky operation

} catch(QueryException e) {

  System.debug('Error: ' + e.getMessage());

}
```

Validation
Introduced a controlled error in a test and verified that the catch block executed, with a descriptive log entry.

Since we don't have any Appointment named "InvalidAppointment123", Salesforce will throw a QueryException.



```
Timestamp       Event        Details
19:52:48:027    USER_DEBUG   [11]|DEBUG|Error caught: List has no rows for assignment to SObject
```

---

12. Test Classes

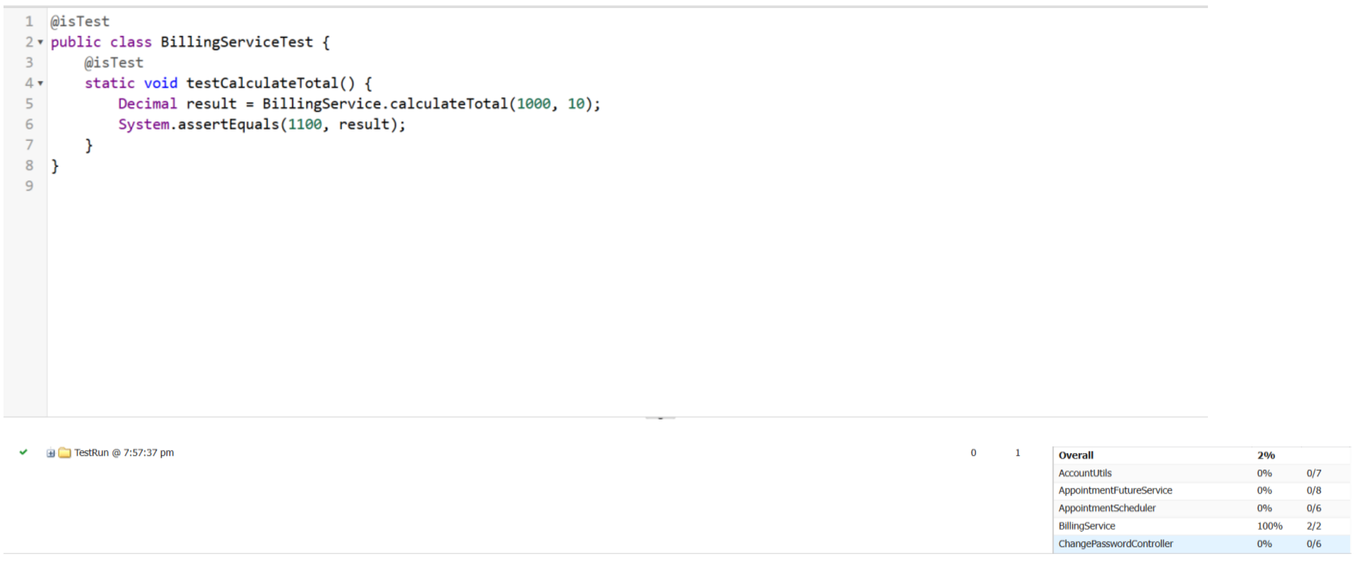
What I implemented
Unit tests for key classes to ensure correctness and satisfy deployment coverage.

How & why
Test for billing logic:

```
@isTest

public class BillingServiceTest {

  @isTest
  static void testCalculateTotal() {
    Decimal result = BillingService.calculateTotal(1000, 10);
    System.assertEquals(1100, result);
  }
}
```

Validation

Ran test suite in Developer Console → Test → New Run; test passed and provided coverage % for the related classes.



```
@isTest
public class BillingServiceTest {
    @isTest
    static void testCalculateTotal() {
        Decimal result = BillingService.calculateTotal(1000, 10);
        System.assertEquals(1100, result);
    }
}
```

```
✓ TestRun @ 7:57:37 pm    0    1
```

Overall	2%	
AccountUtils	0%	0/7
AppointmentFutureService	0%	0/8
AppointmentScheduler	0%	0/6
BillingService	100%	2/2
ChangePasswordController	0%	0/6

---

13. Asynchronous Processing (summary)

What I implemented

A combined solution using Batch Apex (large data), Queueable (chained background jobs), Scheduled (time-driven), and Future (simple async calls).

Why

Together these support scalability and performance: background execution prevents blocking user transactions and respects governor limits.

Validation

Reviewed Apex Jobs, Scheduled Jobs, and Execution Logs; confirmed jobs completed and results were applied to records.

Conclusion

Phase 5 implements production-grade Apex that complements declarative automation. The solution uses standard design patterns (trigger handlers), efficient queries (SOQL/SOSL), collection types for performance, and asynchronous constructs to scale. Each component was validated via Execute Anonymous, Developer Console queries, Apex job monitoring, and unit tests — ensuring the system is reliable, maintainable, and ready for deployment.