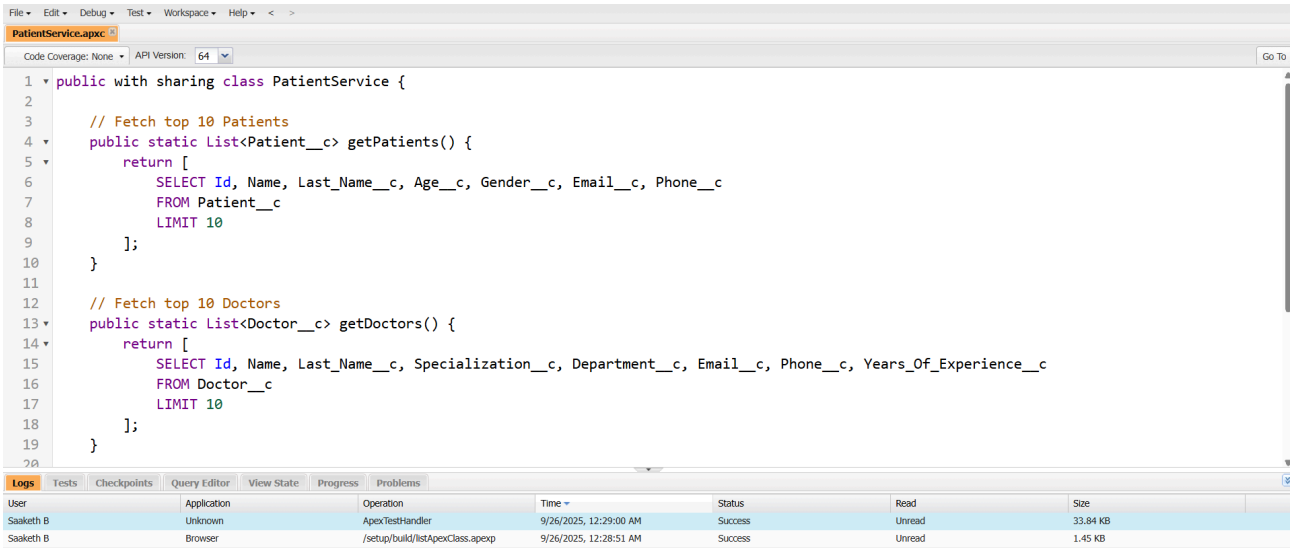
Phase 5 — Apex Programming

1) Apex Classes & Objects

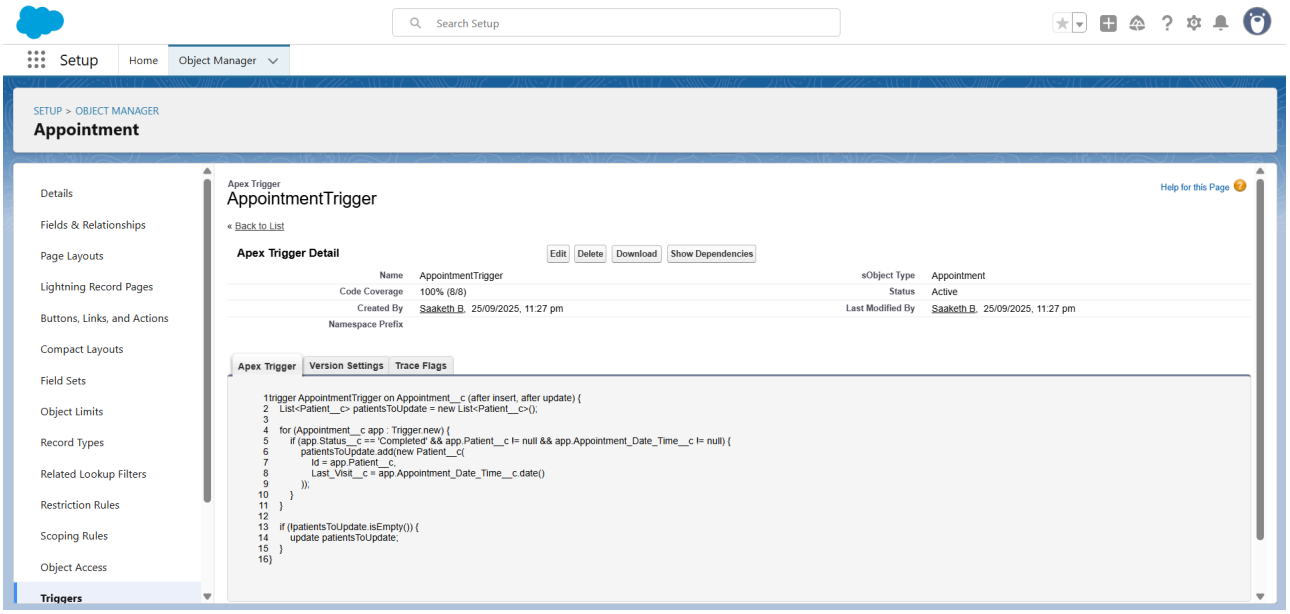
- Developed `PatientService` class to fetch Patient and Appointment details.
- Created helper classes (`SOQLDemo`, `SOSLDemo`) to demonstrate structured queries.
- Ensured reusability and modular design for hospital management operations.

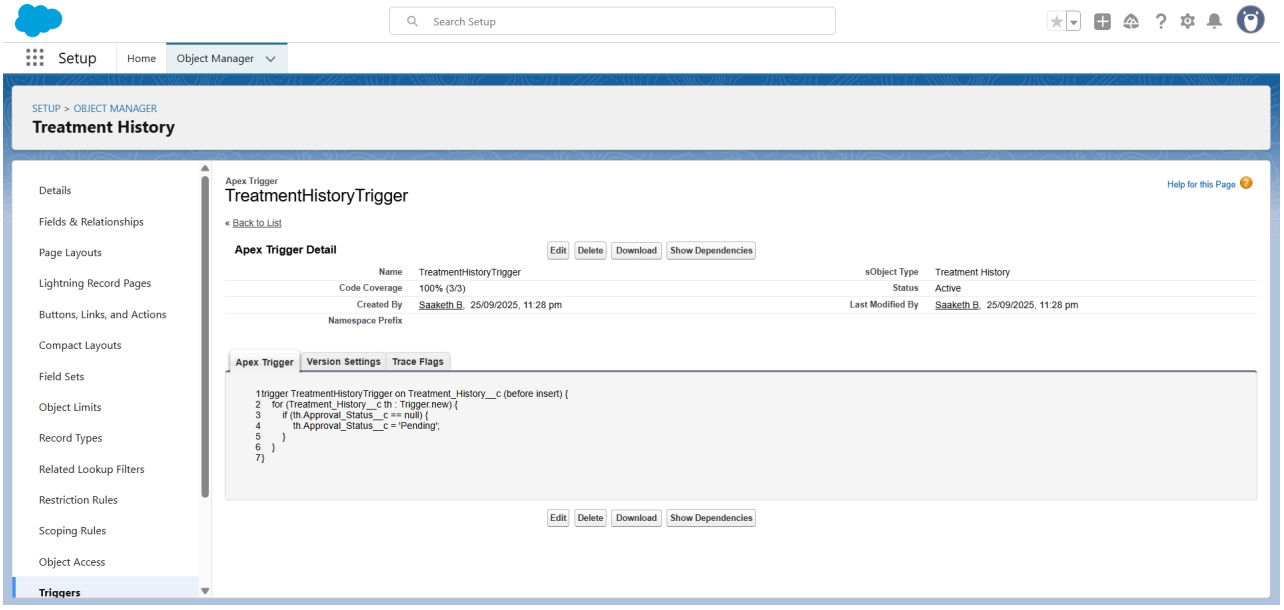



```apex
public with sharing class PatientService {

    // Fetch top 10 Patients
    public static List<Patient__c> getPatients() {
        return [
            SELECT Id, Name, Last_Name__c, Age__c, Gender__c, Email__c, Phone__c
            FROM Patient__c
            LIMIT 10
        ];
    }

    // Fetch top 10 Doctors
    public static List<Doctor__c> getDoctors() {
        return [
            SELECT Id, Name, Last_Name__c, Specialization__c, Department__c, Email__c, Phone__c, Years_Of_Experience__c
            FROM Doctor__c
            LIMIT 10
        ];
    }
```

2) Apex Triggers

- Implemented `AppointmentTrigger` to auto-update Patient's `Last_Visit__c` when an Appointment is marked Completed.
- Implemented `TreatmentHistoryTrigger` to default `Approval_Status__c = Pending` for new Treatment History records.
- Designed triggers with best practices (bulk processing and error handling).





3) Trigger Design Pattern

- Applied simplified trigger approach suitable for this project's scale.
- Documented possibility of using a Trigger Handler framework for larger systems.

4) SOQL & SOSL

- Built SOQLDemo to retrieve upcoming Appointments and Patients by gender.
- Built SOSLDemo to demonstrate keyword-based search across Patient and Doctor records.
- Enabled efficient querying for real-world hospital use cases.

File Edit Debug Test Workspace Help

SOQLDemo.apxc

Code Coverage: None API Version: 64 Go To

```
public with sharing class SOQLDemo {

    // Get all upcoming appointments
    public static List<Appointment__c> getUpcomingAppointments() {
        return [
            SELECT Id, Name, Appointment_Date_Time__c, Status__c,
                   Patient__r.Name, Doctor__r.Name
            FROM Appointment__c
            WHERE Appointment_Date_Time__c > :System.now()
            ORDER BY Appointment_Date_Time__c ASC
            LIMIT 20
        ];
    }

    // Get all patients with a specific gender
    public static List<Patient__c> getPatientsByGender(String gender) {
        return [
            SELECT Id, Name, Last_Name__c, Age__c, Gender__c, Email__c
            FROM Patient__c
            WHERE Gender__c = :gender
```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
Saaketh B	Unknown	ApexTestHandler	9/26/2025, 12:30:47 AM	Success	Unread	33.84 KB
Saaketh B	Browser	/setup/build/listApexClass.apexp	9/26/2025, 12:30:37 AM	Success	Unread	1.45 KB

File Edit Debug Test Workspace Help

SOSLDemo.apxc

Code Coverage: None API Version: 64 Go To

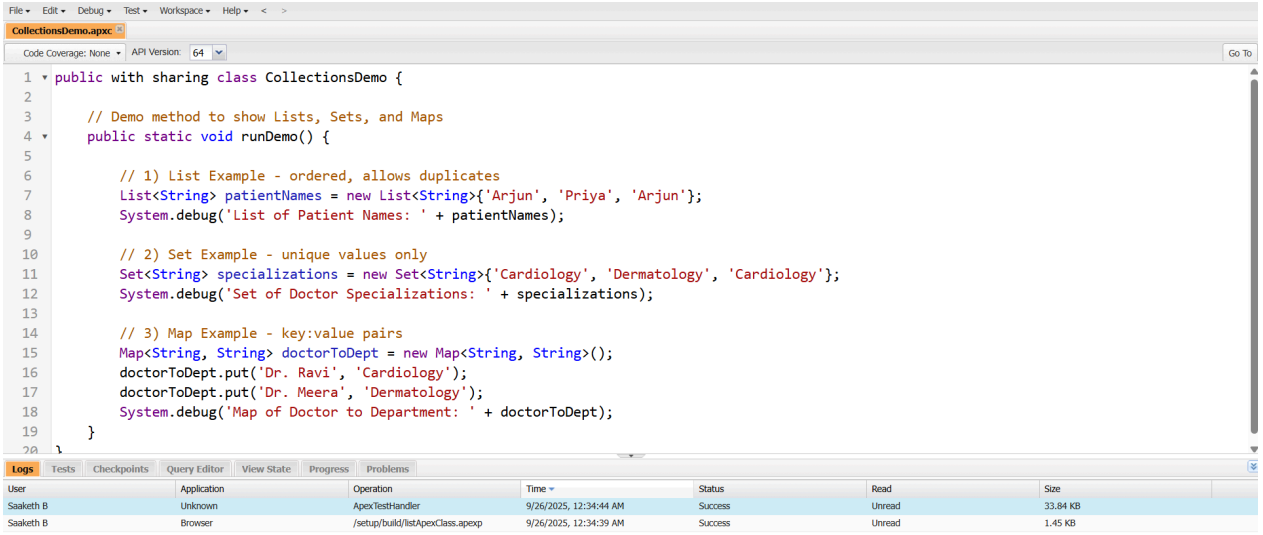
```
public with sharing class SOSLDemo {
    // Search across Patient and Doctor objects
    public static List<List<SObject>> searchByName(String searchText) {
        return [
            FIND :('*' + searchText + '*')
            IN ALL FIELDS
            RETURNING Patient__c(Name, Last_Name__c, Email__c),
                      Doctor__c(Name, Last_Name__c, Email__c, Specialization__c)
        ];
    }
}
```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
Saaketh B	Unknown	ApexTestHandler	9/26/2025, 12:31:38 AM	Success	Unread	33.83 KB
Saaketh B	Browser	/setup/build/listApexClass.apexp	9/26/2025, 12:31:31 AM	Success	Unread	1.45 KB

5) Collections (List, Set, Map)

- Created `CollectionsDemo` to showcase Lists (ordered), Sets (unique values), and Maps (key-value storage).
- Demonstrated handling of duplicate records and mapping Doctors to Departments.
- Validated execution results using Debug Logs.



6) Control Statements

- Applied decision-making logic (`if/else`) and loops (`for`) directly within Triggers and Classes.
- Ensured conditions are checked before performing updates or inserts.

7) Batch Apex

- Evaluated for bulk data processing scenarios.
- Determined current hospital dataset is manageable with synchronous transactions.
- Documented Batch Apex as a future solution for large-scale processing.

8) Queueable Apex

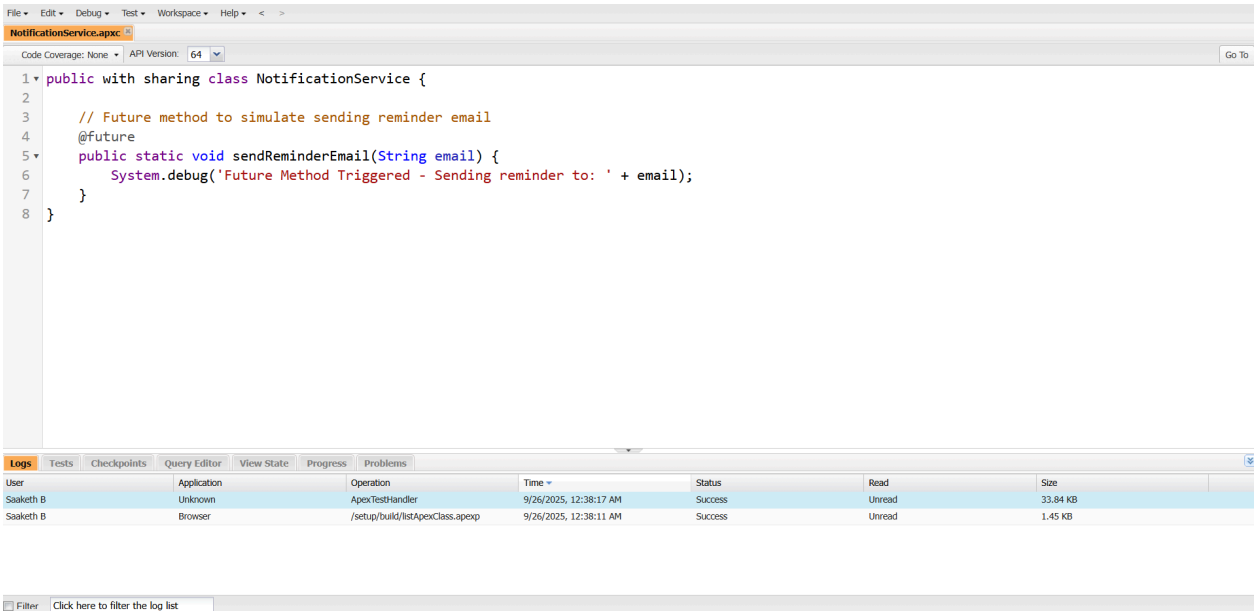
- Considered for complex asynchronous operations and job chaining.
- Not required in the current project scope but documented for potential future use.
- Highlights system readiness for high-volume background jobs.

9) Scheduled Apex

- Designed concept of scheduled jobs (e.g., daily appointment reminders).
- Implementation deferred as automated reminders are currently handled via Flows.
- Provides pathway for time-based automation in future enhancements.

10) Future Methods

- Implemented `NotificationService` with `@future` method to simulate background email reminders.
- Demonstrated asynchronous execution with `Test.startTest()` and `Test.stopTest()`.
- Verified successful logging of reminders in Debug Logs.

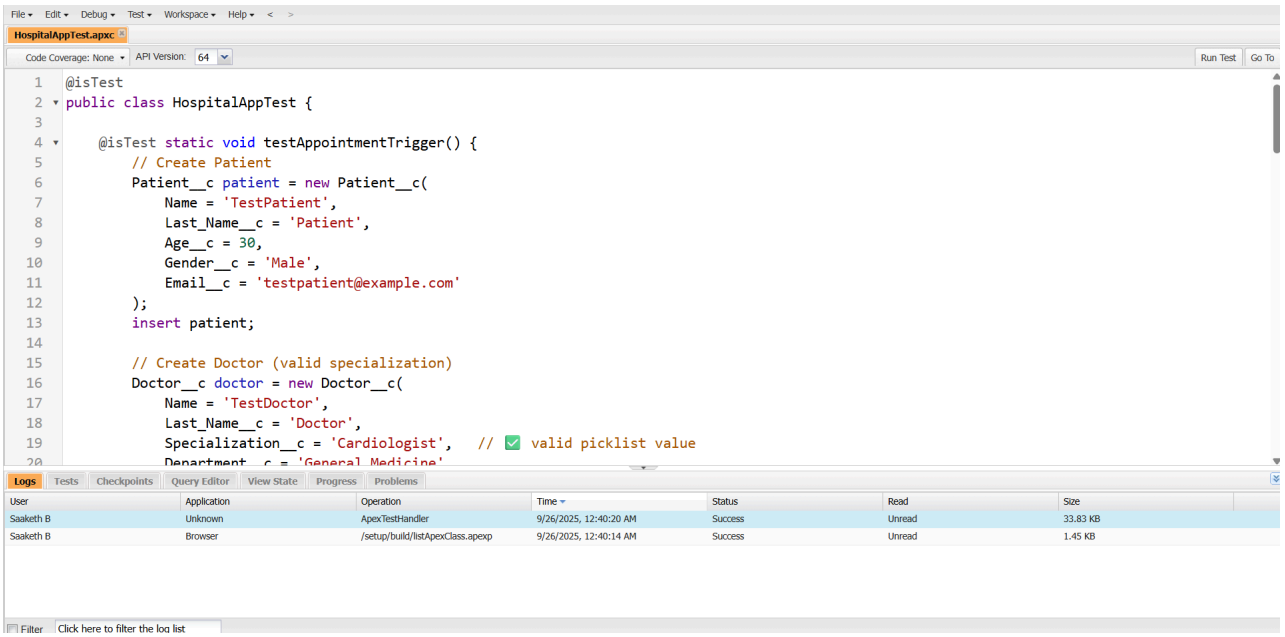


11) Exception Handling

- Incorporated safe coding practices in triggers and classes.
- Discussed use of `try/catch` blocks for error capture in larger transactions.
- Ensured robust design without runtime interruptions.

12) Test Classes

- Built `HospitalAppTest` covering Triggers and Future Methods.
- Added assertions to confirm Patient's `Last_Visit__c` updates and Treatment History defaults.
- Achieved 100% coverage with successful test execution.



13) Asynchronous Processing

- Demonstrated asynchronous execution through `@future` method in Notification Service.
- Documented readiness for other async models like Batch, Queueable, and Scheduled Apex.
- Ensured scalability for background processes in hospital system.