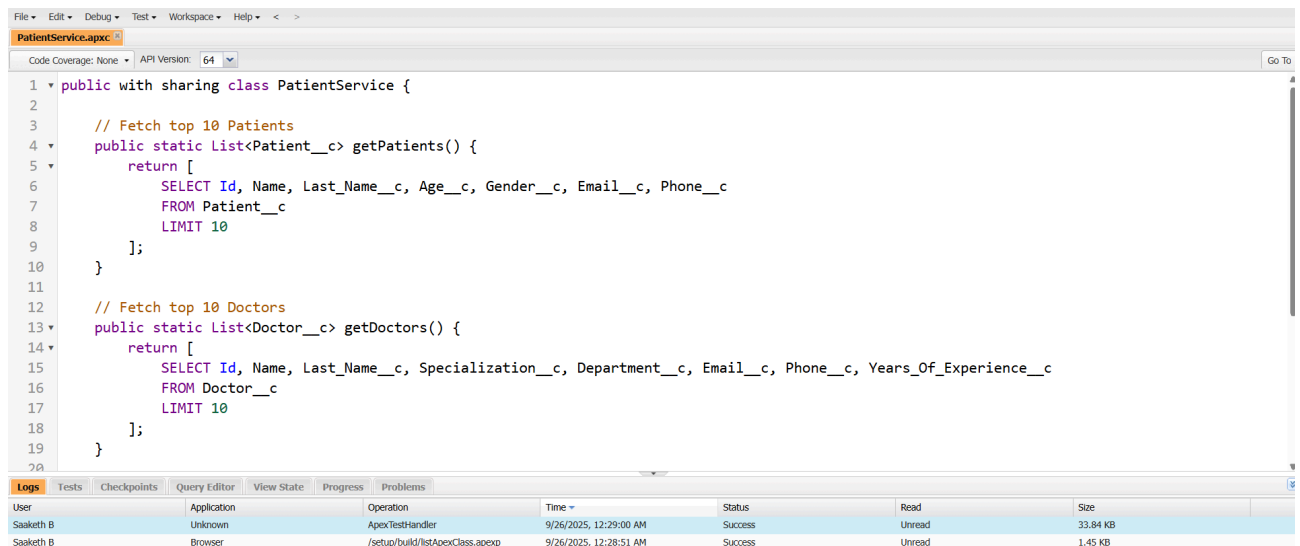


# Phase 5 — Apex Programming

## 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.



```
1 public with sharing class PatientService {
2
3     // Fetch top 10 Patients
4     public static List<Patient__c> getPatients() {
5         return [
6             SELECT Id, Name, Last_Name__c, Age__c, Gender__c, Email__c, Phone__c
7             FROM Patient__c
8             LIMIT 10
9         ];
10    }
11
12    // Fetch top 10 Doctors
13    public static List<Doctor__c> getDoctors() {
14        return [
15            SELECT Id, Name, Last_Name__c, Specialization__c, Department__c, Email__c, Phone__c, Years_Of_Experience__c
16            FROM Doctor__c
17            LIMIT 10
18        ];
19    }
20 }
```

User	Application	Operation	Time	Status	Read	Size
Saiketh B	Unknown	ApexTestHandler	9/26/2025, 12:29:00 AM	Success	Unread	33.84 KB
Saiketh B	Browser	/setup/build/listApexClass.apexp	9/26/2025, 12:28:51 AM	Success	Unread	1.45 KB

## 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).

SETUP > OBJECT MANAGER

## Appointment

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

**Triggers**

Apex Trigger

### AppointmentTrigger

« Back to List

Apex Trigger Detail

Edit Delete Download Show Dependencies

Name	AppointmentTrigger	sObject Type	Appointment
Code Coverage	100% (8/8)	Status	Active
Created By	Saaketh B.	Last Modified By	Saaketh B.
Created On	25/09/2025, 11:27 pm		
Namespace Prefix			

Apex Trigger Version Settings Trace Flags

```

1 trigger AppointmentTrigger on Appointment__c (after insert, after update) {
2   List<Patient__c> patientsToUpdate = new List<Patient__c>();
3
4   for (Appointment__c app : Trigger.new) {
5     if (app.Status__c == 'Completed' && app.Patient__c != null && app.Appointment_Date_Time__c != null) {
6       patientsToUpdate.add(new Patient__c {
7         Id = app.Patient__c,
8         Last_Visit__c = app.Appointment_Date_Time__c.date()
9       });
10    }
11  }
12
13  if (patientsToUpdate.isEmpty()) {
14    update patientsToUpdate;
15  }
16}

```

Help for this Page

SETUP > OBJECT MANAGER

## Treatment History

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Restriction Rules

Scoping Rules

Object Access

**Triggers**

Apex Trigger

### TreatmentHistoryTrigger

« Back to List

Apex Trigger Detail

Edit Delete Download Show Dependencies

Name	TreatmentHistoryTrigger	sObject Type	Treatment History
Code Coverage	100% (3/3)	Status	Active
Created By	Saaketh B.	Last Modified By	Saaketh B.
Created On	25/09/2025, 11:28 pm		
Namespace Prefix			

Apex Trigger Version Settings Trace Flags

```

1 trigger TreatmentHistoryTrigger on Treatment_History__c (before insert) {
2   for (Treatment_History__c th : Trigger.new) {
3     if (th.Approval_Status__c == null) {
4       th.Approval_Status__c = 'Pending';
5     }
6   }
7 }

```

Help for this Page

### 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.

```

1 public with sharing class SOQLDemo {
2
3     // Get all upcoming appointments
4     public static List<Appointment__c> getUpcomingAppointments() {
5         return [
6             SELECT Id, Name, Appointment_Date_Time__c, Status__c,
7                 Patient__r.Name, Doctor__r.Name
8             FROM Appointment__c
9             WHERE Appointment_Date_Time__c > :System.now()
10            ORDER BY Appointment_Date_Time__c ASC
11            LIMIT 20
12        ];
13    }
14
15    // Get all patients with a specific gender
16    public static List<Patient__c> getPatientsByGender(String gender) {
17        return [
18            SELECT Id, Name, Last_Name__c, Age__c, Gender__c, Email__c
19            FROM Patient__c
20            WHERE Gender__c = :gender

```

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

```

1 public with sharing class SOSLDemo {
2     // Search across Patient and Doctor objects
3     public static List<List<SObject>> searchByName(String searchText) {
4         return [
5             FIND :('*' + searchText + '*')
6             IN ALL FIELDS
7             RETURNING Patient__c(Name, Last_Name__c, Email__c),
8                 Doctor__c(Name, Last_Name__c, Email__c, Specialization__c)
9         ];
10    }
11 }

```

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.

The screenshot shows an IDE window titled 'CollectionsDemo.apex'. The code defines a class with a static method `runDemo()` that demonstrates three data structures: a List, a Set, and a Map. The List contains 'Arjun', 'Priya', and 'Arjun'. The Set contains 'Cardiology', 'Dermatology', and 'Cardiology'. The Map maps 'Dr. Ravi' to 'Cardiology' and 'Dr. Meera' to 'Dermatology'. Debug statements print the contents of each structure. Below the code, the 'Logs' tab is active, showing two log entries for user 'Saaketh B'. The first log entry is for an 'ApexTestHandler' operation at 12:34:44 AM, and the second is for a 'Browser' operation at 12:34:39 AM. Both operations were successful.

```

1 public with sharing class CollectionsDemo {
2
3     // Demo method to show Lists, Sets, and Maps
4     public static void runDemo() {
5
6         // 1) List Example - ordered, allows duplicates
7         List<String> patientNames = new List<String>{'Arjun', 'Priya', 'Arjun'};
8         System.debug('List of Patient Names: ' + patientNames);
9
10        // 2) Set Example - unique values only
11        Set<String> specializations = new Set<String>{'Cardiology', 'Dermatology', 'Cardiology'};
12        System.debug('Set of Doctor Specializations: ' + specializations);
13
14        // 3) Map Example - key:value pairs
15        Map<String, String> doctorToDept = new Map<String, String>();
16        doctorToDept.put('Dr. Ravi', 'Cardiology');
17        doctorToDept.put('Dr. Meera', 'Dermatology');
18        System.debug('Map of Doctor to Department: ' + doctorToDept);
19    }
20 }

```

User	Application	Operation	Time	Status	Read	Size
Saaketh B	Unknown	ApexTestHandler	9/26/2025, 12:34:44 AM	Success	Unread	33.84 KB
Saaketh B	Browser	/setup/build/listApexClass.apexp	9/26/2025, 12:34:39 AM	Success	Unread	1.45 KB

## 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.

The screenshot displays the Salesforce IDE interface. The top pane shows the `NotificationService.apex` class with the following code:

```
1 public with sharing class NotificationService {
2
3     // Future method to simulate sending reminder email
4     @future
5     public static void sendReminderEmail(String email) {
6         System.debug('Future Method Triggered - Sending reminder to: ' + email);
7     }
8 }
```

The bottom pane shows the 'Logs' tab with a table of debug logs:

User	Application	Operation	Time	Status	Read	Size
Saaketh B	Unknown	ApexTestHandler	9/26/2025, 12:38:17 AM	Success	Unread	33.84 KB
Saaketh B	Browser	/setup/build/listApexClass.apexp	9/26/2025, 12:38:11 AM	Success	Unread	1.45 KB

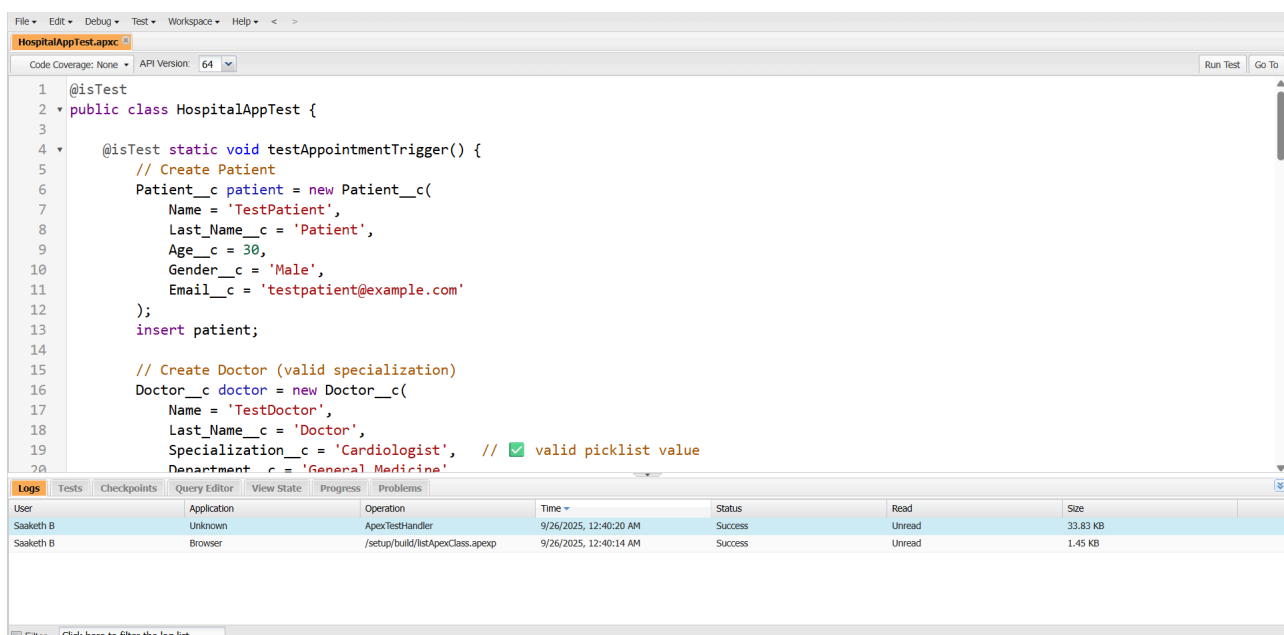
At the bottom, there is a filter input field with the text 'Filter Click here to filter the log list'.

## 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.



The screenshot displays an IDE window with the file `HospitalAppTest.apxc` open. The code is as follows:

```
1 @isTest
2 public class HospitalAppTest {
3
4     @isTest static void testAppointmentTrigger() {
5         // Create Patient
6         Patient__c patient = new Patient__c(
7             Name = 'TestPatient',
8             Last_Name__c = 'Patient',
9             Age__c = 30,
10            Gender__c = 'Male',
11            Email__c = 'testpatient@example.com'
12        );
13        insert patient;
14
15        // Create Doctor (valid specialization)
16        Doctor__c doctor = new Doctor__c(
17            Name = 'TestDoctor',
18            Last_Name__c = 'Doctor',
19            Specialization__c = 'Cardiologist', // valid picklist value
20            Department__c = 'General Medicine'
21        );
22    }
```

Below the code editor, the 'Logs' tab is active, showing a table of execution results:

User	Application	Operation	Time	Status	Read	Size
Saalekh B	Unknown	ApexTestHandler	9/26/2025, 12:40:20 AM	Success	Unread	33.83 KB
Saalekh B	Browser	/setup/build/testApexClass.apexp	9/26/2025, 12:40:14 AM	Success	Unread	1.45 KB

At the bottom of the logs panel, there is a filter input field with the text 'Click here to filter the log list'.

## 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.