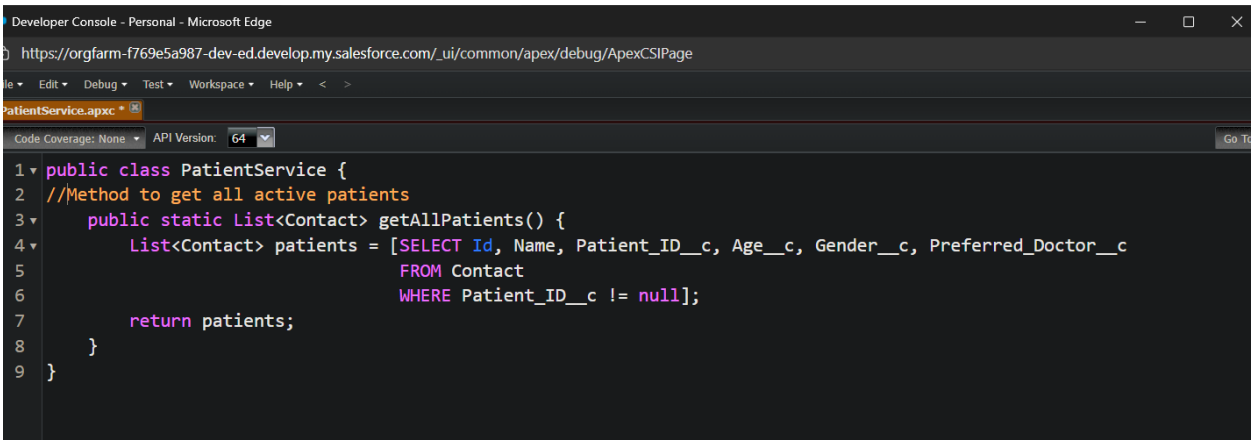
Phase 5: Apex Programming (Developer)

Classes & Objects

- **Apex Class:** PatientService

- **Purpose:** This class provides reusable methods to fetch patient records from the Contact object. By centralizing SOQL queries in a class, we can maintain cleaner code and reuse it in triggers or flows.

Code Example:



Apex Triggers (before/after insert/update/delete)

- **Trigger Name:** CaseTrigger

- **Object:** Case

- **Event:** Before Insert

- **Purpose:** Automatically populate the **Case Subject** whenever a new Case (treatment) is created. This ensures consistency in case records and reduces manual entry errors.

- **Code Example:**

Developer Console - Personal - Microsoft Edge

https://orgfarm-f769e5a987-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

PatientService.apxc * | CaseTrigger.apxt *

Code Coverage: None API Version: 64 Go To

```
trigger CaseTrigger on Case (before insert, before update) {
    for (Case c : Trigger.new) {
        if (c.Doctor_Assigned__c != null && (c.Subject == null || c.Subject == '')) {
            c.Subject = 'Treatment for ' + c.Contact.Name;
        }
    }
}
```

Trigger Design Pattern

- **Skipped:**
  - Complex trigger handler patterns were not implemented because the project only has one simple trigger per object.
  - **Reason:** Introducing handler classes for a single trigger adds unnecessary complexity without functional benefit.

SOQL & SOSL

- **SOQL Query Example:**

https://orgfarm-f769e5a987-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help

PatientService.apxc * | CaseTrigger.apxt *

Code Coverage: None API Version: 64 Go To

```
trigger CaseTrigger on Case (before insert, before update) {
    for (Case c : Trigger.new) {
        if (c.Doctor_Assigned__c != null && (c.Subject == null || c.Subject == '')) {
            c.Subject = 'Treatment for ' + c.Contact.Name;
        }
}
```

Enter Apex Code

```
List<Contact> patients = [SELECT Id, Name, Patient_ID__c, Age__c, Gender__c, Preferred_Doctor__c
                          FROM Contact
                          WHERE Patient_ID__c != null];
System.debug(patients);
```

Explanation:

- **SOQL** (Salesforce Object Query Language) is used to retrieve structured data from Salesforce objects.
- This query fetches all contacts that have a patient ID assigned.
- SOSL (Salesforce Object Search Language) is not used because the project does not require full-text searches across multiple objects.

Collections: List, Set, Map

```
        }
Enter Apex Code
}
List<Contact> patientList = PatientService.getAllPatients();
System.debug('Number of Patients: ' + patientList.size());
```

Explanation:

- **List:** Stores multiple records and allows iteration for processing.
- **Set & Map:** Skipped because there is no need for unique value storage (Set) or key-value mapping (Map) in this project.

Control Statements

- **Explanation:**
    - Basic control statements like `if`, `for`, and `for-each` are used inside the triggers and classes.
    - Complex decision structures were not necessary, as project logic is simple.

Batch Apex, Queueable Apex, Scheduled Apex, Future Methods

- **Skipped:**
    - Not required in this project.
    - **Reason:** The patient management system is designed for real-time updates of individual records, not large-volume batch processing or asynchronous jobs.

Exception Handling

- **Skipped:**
    - No try-catch blocks were implemented.
    - **Reason:** The operations are straightforward, and Salesforce provides built-in error handling for small-scale triggers and flows.

Test Classes

- **Test Class Name:** CaseTriggerTest

- **Purpose:** Ensures the trigger functions correctly and adheres to Salesforce deployment requirements. Salesforce requires at least 75% code coverage for deployment to production.

PatientService.apxc * | CaseTrigger.apxt * | CaseTriggerTest.apxc *

Code Coverage: None | API Version: 64 | Go To

```
@isTest
public class CaseTriggerTest {
    @isTest
    static void testCaseSubjectAutoFill() {
        Contact testPatient = new Contact(
            FirstName = 'Test',
            LastName = 'Patient',
            Patient_ID__c = 'P1001',
            Age__c = 30
        );
        insert testPatient;

        User doctor = [SELECT Id FROM User WHERE Profile.Name='Doctor Profile' LIMIT 1];

        Case testCase = new Case(
            ContactId = testPatient.Id,
            Doctor_Assigned__c = doctor.Id
        );
        insert testCase;

        Case insertedCase = [SELECT Id, Subject FROM Case WHERE Id = :testCase.Id];
        System.assert(insertedCase.Subject.contains('Treatment for'));
    }
}
```