Phase 5: Apex Programming (Developer)

Airline Management System

Salesforce-Based Passenger & Operations Management

Step 1: Log in to Salesforce Developer Org

- 1. Go to Salesforce Developer and sign up if you don't have an org.
- 2. Log in to your **Developer Edition** or **Sandbox**.
- 3. Switch to **Lightning Experience** for easier navigation.

Step 2: Create Custom Objects

Purpose: Define your Airline entities.

Steps:

- 1. Go to Setup \rightarrow Object Manager \rightarrow Create \rightarrow Custom Object.
- 2. Create these objects:
 - o Airline c
 - o Flight c
 - o Passenger c
 - o Booking c

3. Add custom fields:

- Flight_c: Name, Status (Picklist: Scheduled, Completed, Cancelled),
 DepartureDate (DateTime)
- o Airline_c: Name, TotalFlights (Number)
- o Passenger c: Name, Email
- Booking_c: Flight (Lookup to Flight_c), Passenger (Lookup to Passenger_c)

Step 3: Create Apex Classes

Purpose: Write logic to manage your objects.

Steps:

- 1. Go to Setup \rightarrow Apex Classes \rightarrow New.
- 2. Create a class like Airline:

```
Developer Console - Personal - Microsoft Edge
https://orgfarm-a267b8d0e4-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File • Edit • Debug • Test • Workspace • Help • < >
Code Coverage: None • API Version: 64 •
  1 * public class Airline {
! 3 * }public class Airline {
          public String name;
          public Integer totalFlights;
                                                                                                     1 Airline a = new Airline('SkyAir', 100);
         public Airline(String name, Integer flights) {
                                                                                                         a.displayInfo();
               this.totalFlights = flights;
  10
  11
          public void displayInfo() {
              System.debug(name + ' has ' + totalFlights + ' flights');
  13
14
  15 }
                                                                                                                                               Open Log Execute Execute Highlighted
```

- Click Save.
- Test the class using Developer Console → Debug → Open Execute Anonymous Window:

```
Airline a = new Airline('SkyAir', 100);
a.displayInfo();
```

Step 4: Create Apex Triggers

Purpose: Automate actions when records are created/updated/deleted.

Steps:

- 1. Go to Setup \rightarrow Object Manager \rightarrow Flight $c \rightarrow$ Triggers \rightarrow New.
- 2. Create a trigger:

```
Developer Console - Personal - Microsoft Edge

the through the process of the pro
```

3. Click Save.

4. Test: Go to **Flight_c** → **New Record**, add a flight → **Save** → check Status is automatically "Scheduled".

Step 5: Implement Trigger Handler (Design Pattern)

Purpose: Keep triggers clean.

Steps:

1. Create Apex Class → FlightHandler:

```
Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Help < >
| Tile: Edit - Debay - Tiles - Warspare - Warspare - Tiles - Til
```

2. Update trigger to delegate:

```
trigger FlightTrigger on Flight_c (before insert) {
FlightHandler.beforeInsert(Trigger.new);
}
```

Step 6: Use SOQL & SOSL

Purpose: Query records in Salesforce.

Steps:

- 1. Open Developer Console \rightarrow Query Editor.
- 2. Run a SOQL query:

```
SELECT Name, Status c FROM Flight c WHERE Status c='Scheduled'
```

3. Run a SOSL query in **Apex**:

```
List<List<SObject>> results = [FIND 'SkyAir*' IN ALL FIELDS RETURNING Flight_c(Name, Status_c)];
```

Step 7: Use Collections

Purpose: Handle multiple records in code.

Steps:

1. Create a new Apex Class FlightUtils:



2. Execute in **Developer Console** → **Execute Anonymous**:

FlightUtils.exampleCollections();

Step 8: Use Control Statements

Purpose: Handle conditions and loops.

Steps:

1. In your Apex Class, use if-else:

```
for(Flight__c f : [SELECT Name, Status__c FROM Flight__c]) {
    if(f.Status__c == 'Cancelled') System.debug(f.Name + ' is cancelled.');
    else System.debug(f.Name + ' is active.');
}
```

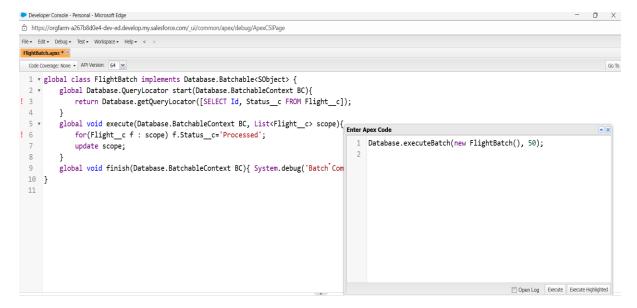
2. Execute in Anonymous Window.

Step 9: Implement Batch Apex

Purpose: Process large datasets.

Steps:

1. Create Apex Class \rightarrow FlightBatch:



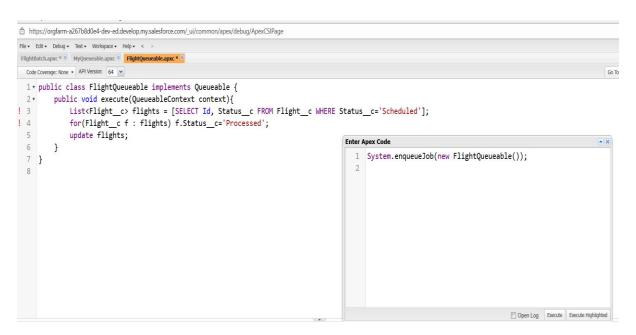
2. Run in Execute Anonymous:

Database.executeBatch(new FlightBatch(), 50);

Step 10: Implement Queueable Apex

Steps:

1. Create Apex Class \rightarrow FlightQueueable:



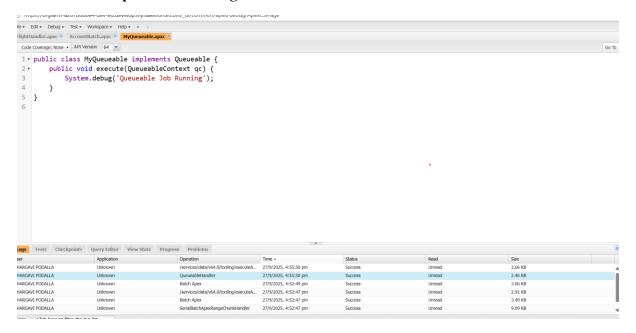
2. Enqueue job:

System.enqueueJob(new FlightQueueable());

Step 11: Implement Scheduled Apex

Steps:

1. Create Apex Class → FlightScheduler:



2. Schedule job in Execute Anonymous:

```
String cronExp = '0 0 12 * * ?'; // Daily 12 PM
System.schedule('DailyFlightJob', cronExp, new FlightScheduler());
```

Step 12: Use Future Methods

Steps:

1. Create Apex Class \rightarrow FlightService:

2. Call asynchronously:

FlightService.updateStatus(new List<Id>{'a0F1t00000123AB'});

Step 13: Exception Handling

Steps:

1. Wrap DML operations in **try-catch**:

```
try {
    Flight__c f = [SELECT Id FROM Flight__c WHERE Name='SkyAir' LIMIT 1];
    f.Status__c='Delayed';
    update f;
} catch(DmlException e) {
    System.debug('Error: ' + e.getMessage());
}
```

Step 14: Write Test Classes

Steps:

1. Create Apex Class \rightarrow FlightTest:

```
https://orgfarm-a267b8d0e4-dev-ed.develop.my.salesforce.com/ui/common/apex/debug/ApexCSIPage
File ▼ Edit ▼ Debug ▼ Test ▼ Workspace ▼ Help ▼ <
FlightBatch.apxc * MyQueueable.apxc * FlightQueueable.apxc * FlightService.apxc * FlightServi
      Code Coverage: None ▼ API Version: 64 ▼
        1 @isTest
      2 ▼ public class FlightTest {
                                      @isTest static void testQueueable(){
                                                          Flight_c f = new Flight_c(Name='TestFlight', Status_c='Scheduled');
4
       5
                                                           insert f;
       6
       7
                                                           Test.startTest();
       8
                                                           System.enqueueJob(new FlightQueueable());
      9
                                                           Test.stopTest();
      10
                                                            f = [SELECT Status_c FROM Flight_c WHERE Id=:f.Id];
11
                                                             System.assertEquals('Processed', f.Status_c);
12
      13
      14 }
      15
```

2. Run tests in Setup \rightarrow Apex Test Execution \rightarrow Run All Tests.

Following this, you will have **fully implemented Apex logic, triggers, asynchronous processing, and test coverage in your Salesforce Org** for the Airline Management System.