📓 Phase 5: Apex Programming (Developer)

🔷 Objective

To implement custom business logic using **Apex Triggers and Classes** in Salesforce. The goal is to automate updates in the **Production Batch** whenever related **Bag** records are created or deleted. This ensures real-time accuracy of production data.

🔷 Implementation

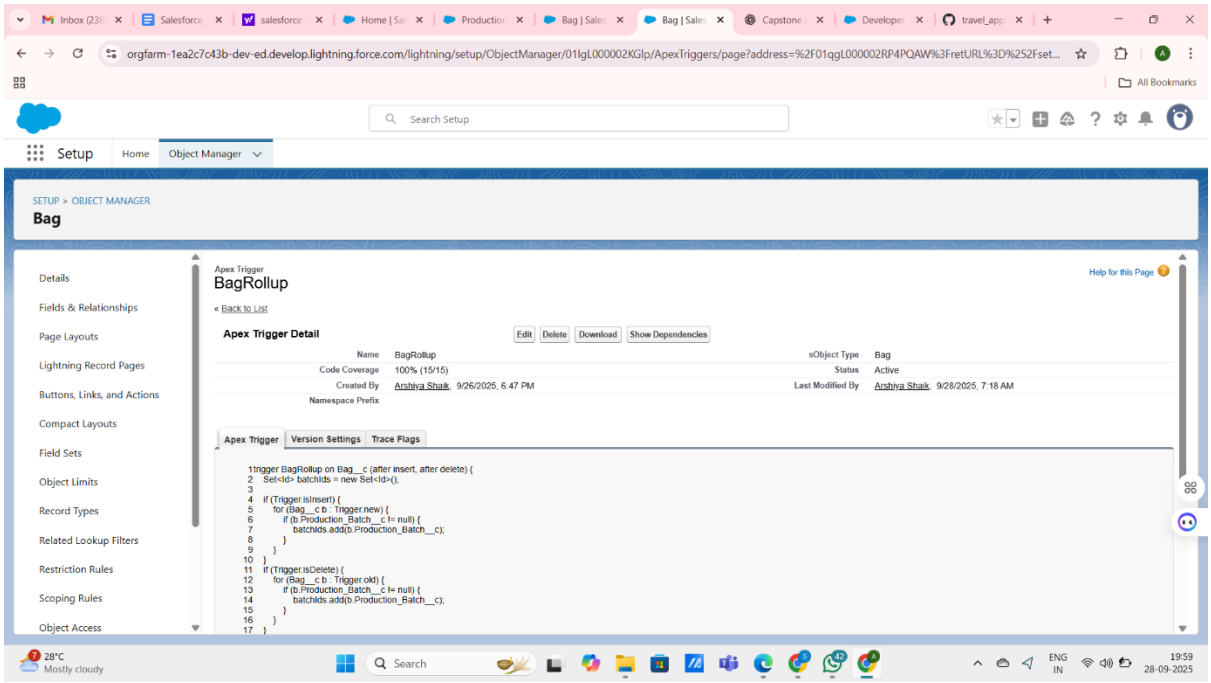
1. Apex Trigger – BagRollup

- **Purpose**: Automatically updates Total_Bags_Produced__c on Production_Batch__c whenever Bag records are inserted or deleted.
- **Key Concepts Used**:
    - **SOQL**: Query related Bag records.
    - **Collections**: Set<Id> and Map<Id, Integer>.
    - **Control Statements**: if, for, Trigger.isInsert, Trigger.isDelete.

Code (Example):

```
trigger BagRollup on Bag__c (after insert, after delete) {
  Set<Id> batchIds = new Set<Id>();

  if (Trigger.isInsert) {
    for (Bag__c b : Trigger.new) {
      if (b.Production_Batch__c != null) {
        batchIds.add(b.Production_Batch__c);
      }
    }
  }

  if (Trigger.isDelete) {
    for (Bag__c b : Trigger.old) {
      if (b.Production_Batch__c != null) {
```

```
            batchIds.add(b.Production_Batch__c);
        }
    }
}

Map<Id, Integer> batchCounts = new Map<Id, Integer>();
for (AggregateResult ar : [
    SELECT Production_Batch__c batchId, COUNT(Id) cnt
    FROM Bag__c
    WHERE Production_Batch__c IN :batchIds
    GROUP BY Production_Batch__c
]) {
    batchCounts.put((Id)ar.get('batchId'), (Integer)ar.get('cnt'));
}

List<Production_Batch__c> updates = new List<Production_Batch__c>();
for (Id batchId : batchCounts.keySet()) {
    updates.add(new Production_Batch__c(
        Id = batchId,
        Total_Bags_Produced__c = batchCounts.get(batchId)
    ));
}

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



---

2. Apex Test Class – BagRollupTest

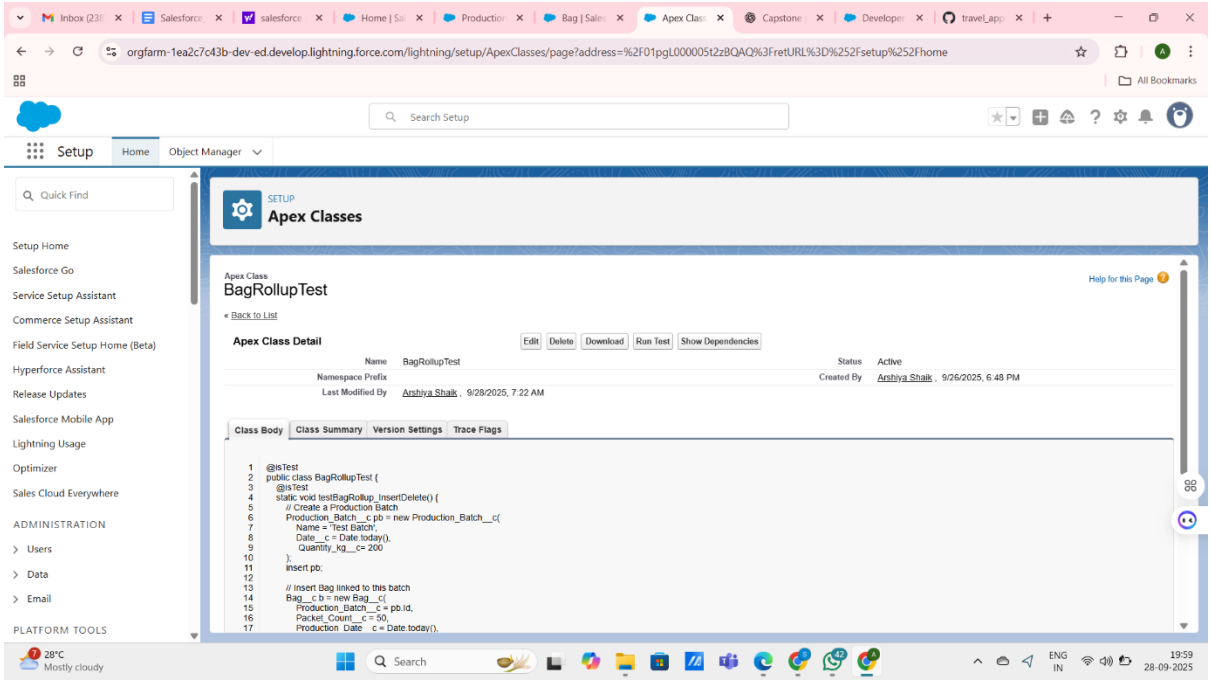
- **Purpose**: Ensures trigger works correctly by testing insert and delete of Bag records.
- **Key Concepts Used**:
    - **SOQL** inside test.
    - **DML** operations: insert, delete.
    - **Assertions** to check correctness.

Code (Example):

```
@isTest
public class BagRollupTest {
    @isTest
    static void testBagRollup_Insert() {
        Production_Batch__c batch = new Production_Batch__c(
            Name = 'Batch Test',
            Quantity__c = 200,
            Supervisor__c = 'Test Supervisor'
        );
```

```
        insert batch;

        Bag__c bag1 = new Bag__c(
            Name = 'Bag 1',
            Packet_Count__c = 50,
            Production_Batch__c = batch.Id,
            Production_Date__c = Date.today()
        );

        Bag__c bag2 = new Bag__c(
            Name = 'Bag 2',
            Packet_Count__c = 50,
            Production_Batch__c = batch.Id,
            Production_Date__c = Date.today()
        );

        insert new List<Bag__c>{ bag1, bag2 };

        batch = [SELECT Id, Total_Bags_Produced__c FROM Production_Batch__c WHERE Id =
:batch.Id];
        System.assertEquals(2, batch.Total_Bags_Produced__c, 'Bag count should be 2');

        delete bag1;

        batch = [SELECT Id, Total_Bags_Produced__c FROM Production_Batch__c WHERE Id =
:batch.Id];
        System.assertEquals(1, batch.Total_Bags_Produced__c, 'Bag count should be updated to
1');
    }
```

}



---

3. Test Execution

- Ran the test in **Developer Console → Test → New Run → BagRollupTest**.
- Result: ✅ Test passed successfully.
- Code coverage achieved (must be 75% or higher for deployment).