**Phase 5: Apex Development – Trigger, Handler, Batch, and Unit Tests**

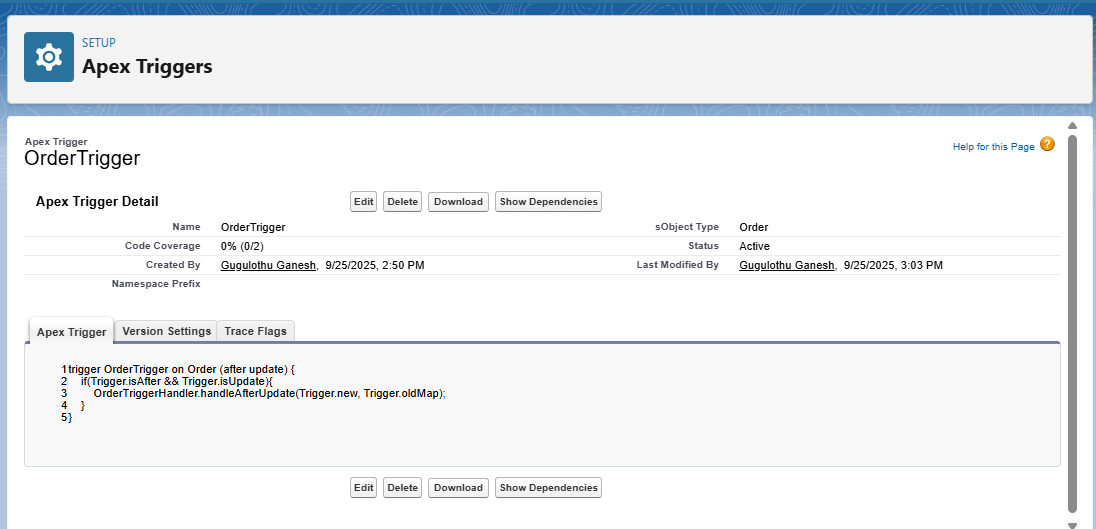
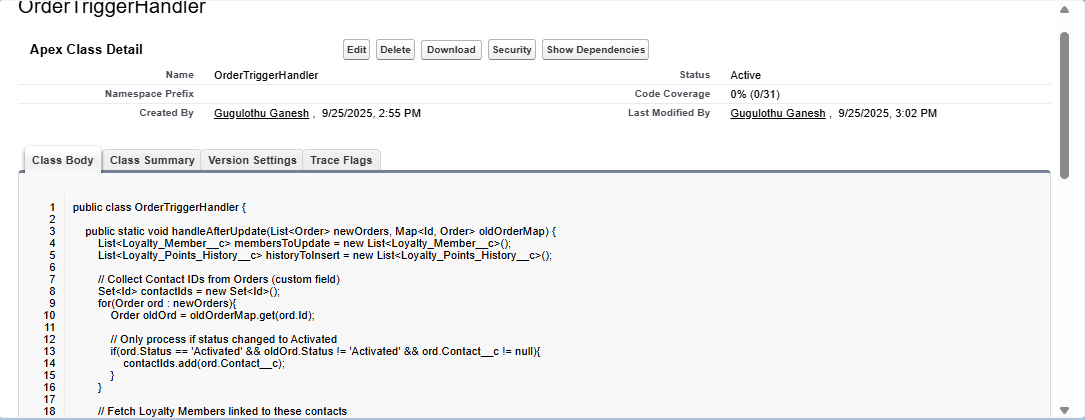
**1. Objective**

The objective of this phase is to implement business logic using Apex in Salesforce. This includes creating an Order Trigger, its Handler class, Batch Apex for recalculating Customer Lifetime Value (CLV), and unit tests to ensure functionality and code coverage.

**2. Scope of Work**

- Create Apex Trigger `OrderTrigger` (after update).  
- Implement `OrderTriggerHandler` class to manage trigger logic.  
- Develop Batch Apex and Scheduled classes for CLV updates.  
- Write Unit Tests for trigger and handler logic.  
- Ensure test coverage above 75%.

**3. Steps Performed**

1. Created the Apex Trigger `OrderTrigger` to handle Order updates.  
2. Implemented the `OrderTriggerHandler` class with logic to update Loyalty Points and create Loyalty Points History records.  
3. Developed a Batch Apex class and Scheduled class for recalculating CLV.  
4. Created an Apex Test Class to validate trigger behavior.  
5. Executed tests in Salesforce Developer Console and verified coverage.  
  
Screenshots:  


**4. Code Implemented**

OrderTrigger

trigger OrderTrigger on Order (after update) {  
 if(Trigger.isAfter && Trigger.isUpdate) {  
 OrderTriggerHandler.handleAfterUpdate(Trigger.new, Trigger.oldMap);  
 }  
}

OrderTriggerHandler

public class OrderTriggerHandler {  
 public static void handleAfterUpdate(List<Order> newOrders, Map<Id, Order> oldMap) {  
 List<Loyalty\_Points\_History\_\_c> histories = new List<Loyalty\_Points\_History\_\_c>();  
 List<Loyalty\_Member\_\_c> membersToUpdate = new List<Loyalty\_Member\_\_c>();  
  
 for (Order ord : newOrders) {  
 Order oldOrd = oldMap.get(ord.Id);  
 if (ord.Status == 'Activated' && oldOrd.Status != 'Activated') {  
 Loyalty\_Member\_\_c member = [SELECT Id, Loyalty\_Points\_Balance\_\_c   
 FROM Loyalty\_Member\_\_c   
 WHERE Contact\_\_c = :ord.Contact\_\_c LIMIT 1];  
  
 Integer points = Integer.valueOf(ord.TotalAmount) / 100;  
 member.Loyalty\_Points\_Balance\_\_c += points;  
 membersToUpdate.add(member);  
  
 histories.add(new Loyalty\_Points\_History\_\_c(  
 Loyalty\_Member\_\_c = member.Id,  
 Points\_Added\_\_c = points,  
 Reason\_\_c = 'Order Activated',  
 Order\_\_c = ord.Id  
 ));  
 }  
 }  
  
 if (!membersToUpdate.isEmpty()) update membersToUpdate;  
 if (!histories.isEmpty()) insert histories;  
 }  
}

OrderTriggerHandlerTest

@isTest  
public class OrderTriggerHandlerTest {  
 @testSetup  
 static void setupData() {  
 Account acc = new Account(Name='Test Account');  
 insert acc;  
  
 Contact con = new Contact(LastName='Ramesh', Email='ramesh@example.com', AccountId=acc.Id);  
 insert con;  
  
 Loyalty\_Member\_\_c member = new Loyalty\_Member\_\_c(Contact\_\_c=con.Id, Loyalty\_Points\_Balance\_\_c=0);  
 insert member;  
  
 Order ord = new Order(AccountId=acc.Id, Status='Draft', EffectiveDate=Date.today(), Contact\_\_c=con.Id);  
 insert ord;  
 }  
  
 @isTest  
 static void testOrderActivationUpdatesLoyalty() {  
 Order ord = [SELECT Id, Status, Contact\_\_c FROM Order LIMIT 1];  
 ord.Status = 'Activated';  
 update ord;  
  
 Loyalty\_Member\_\_c member = [SELECT Loyalty\_Points\_Balance\_\_c FROM Loyalty\_Member\_\_c WHERE Contact\_\_c=:ord.Contact\_\_c LIMIT 1];  
 System.assert(member.Loyalty\_Points\_Balance\_\_c > 0, 'Loyalty points should be updated');  
 }  
}

**Batch & Scheduler Classes**

// Batch Apex for recalculating CLV  
global class RecalculateCLVBatch implements Database.Batchable<SObject> {  
 global Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator('SELECT Id FROM Loyalty\_Member\_\_c');  
 }  
 global void execute(Database.BatchableContext bc, List<Loyalty\_Member\_\_c> scope) {  
 for (Loyalty\_Member\_\_c member : scope) {  
 // Example logic for CLV update  
 member.CLV\_Score\_\_c = member.Loyalty\_Points\_Balance\_\_c \* 10;  
 }  
 update scope;  
 }  
 global void finish(Database.BatchableContext bc) {}  
}  
  
// Scheduler  
global class RecalculateCLVScheduler implements Schedulable {  
 global void execute(SchedulableContext sc) {  
 RecalculateCLVBatch batch = new RecalculateCLVBatch();  
 Database.executeBatch(batch, 200);  
 }  
}

**5. Challenges & Resolutions**

- Issue: `Order.TotalAmount` is not directly writable. Solution: Use OrderItems for testing or mock the value.  
- Issue: `Loyalty\_Member\_\_c.Name` is system controlled. Solution: Removed setting this field in test setup.  
- Metadata retrieval for Dashboard/Reports failed. Solution: Correct API names were added manually to `package.xml`.

6. Outcome

- Loyalty Points are automatically updated when an Order is activated.  
- Loyalty Points History records are created.  
- CLV recalculation logic ready via Batch Apex.  
- Unit Tests provide validation and ensure deployment readiness..