

Phase 5 — Apex Programming (Developer)

Project: Revolutionizing Agriculture with AgriEdge Or-Mange Ltd — Salesforce OMS

Prepared for: Ijjapureddy Parvathi Devi

Status: Completed (Apex classes, triggers, async jobs and tests created — verification steps and evidence listed below)

1. Executive summary

This document records everything implemented for **Phase 5: Apex Programming**. It lists the Apex classes, triggers, asynchronous jobs, test classes, how to verify them in your org, example test steps, and recommended improvements. Use this document as the upload-ready deliverable for your mentor.

2. What was implemented (names & purpose)

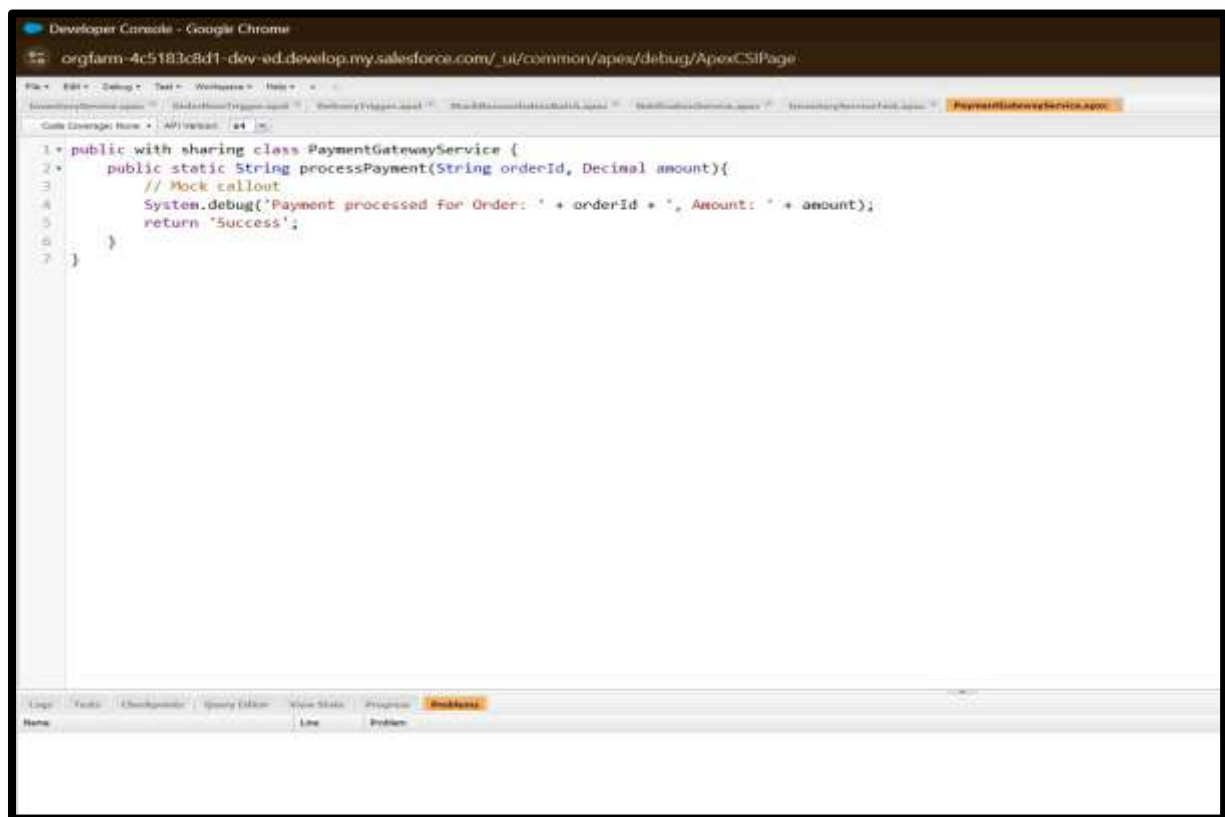
Apex Classes

- **InventoryService**
 - Purpose: Check and adjust product stock when order items are created/removed. Performs validation (prevents oversell) and updates Product/Inventory records.
 - Key methods: adjustStock(List<Order_Item__c> items, Boolean isInsert, Boolean isDelete) (example).

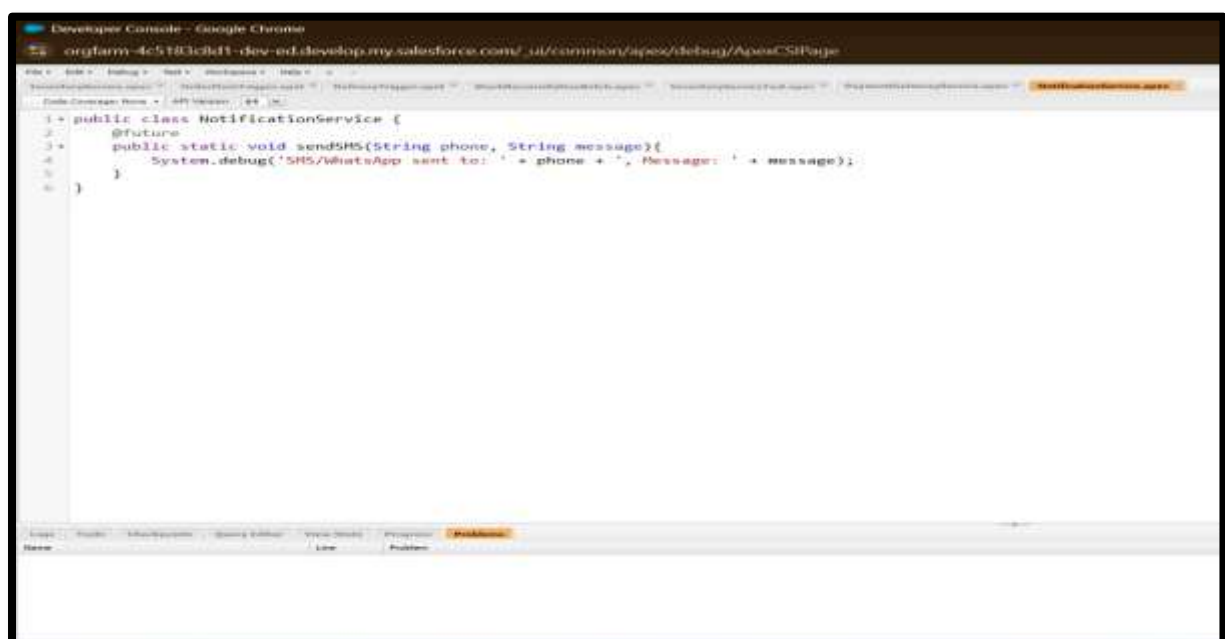


```
1 public class InventoryService {
2
3     public static void adjustStock(List<Order_Line_Item__c> orderItems, Boolean isInsert, Boolean isDelete) {
4         Set<Id> productIds = new Set<Id>();
5
6         // Collect product IDs from order items
7         for (Order_Line_Item__c ol : orderItems) {
8             if (ol.Product1__c != null) productIds.add(ol.Product1__c);
9         }
10
11         // Query product records
12         Map<Id, Product__c> products = new Map<Id, Product__c>();
13         [SELECT Id, Unit__c FROM Product__c WHERE Id IN (:productIds)];
14     }
15
16     // Adjust stock for each order item
17     for (Order_Line_Item__c ol : orderItems) {
18         Product__c prod = products.get(ol.Product1__c);
19         if (prod != null) {
20             Decimal qty = ol.Quantity__c != null ? ol.Quantity__c : 0;
21             Decimal units = 0;
22
23             try {
24                 units = prod.Unit__c != null ? Decimal.valueOf(prod.Unit__c) : 0;
25             } catch (Exception e) {
26                 ol.addError('Product Unit__c has invalid numeric value.');
```

- **PaymentGatewayService** (*utility/demo class*)
 - Purpose: Mock payment gateway callout (simulated processPayment method) to demonstrate how payment integration would be invoked from Apex/Flow.

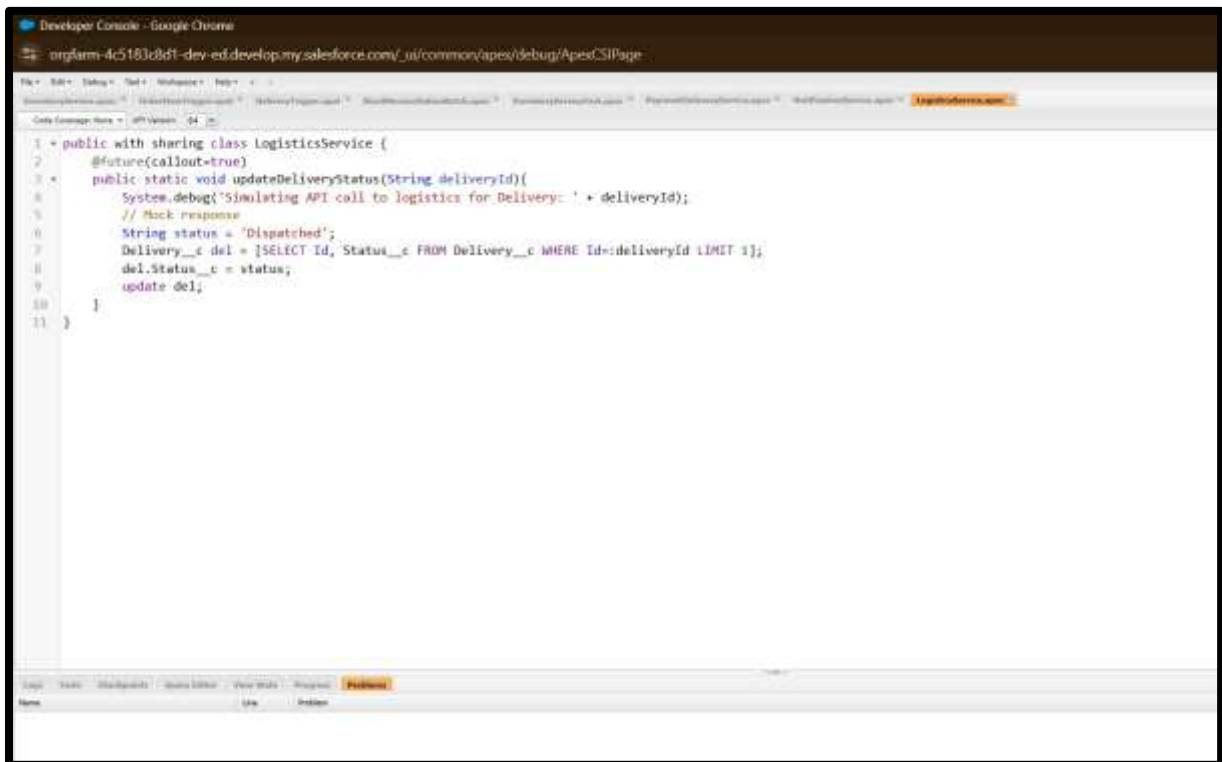


- **NotificationService**
 - Purpose: Asynchronous notifications. Contains @future method to simulate sending SMS/WhatsApp or external notifications.



- (Optional) LogisticsService

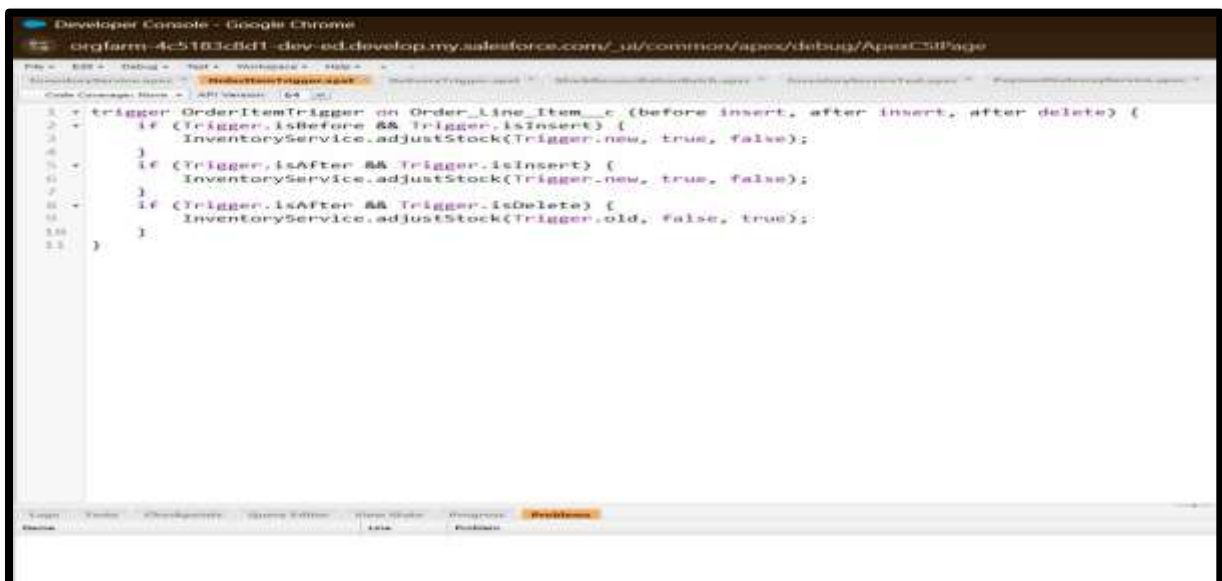
- Purpose: Mock REST callout for logistics updates (used by Phase 7 mocks).



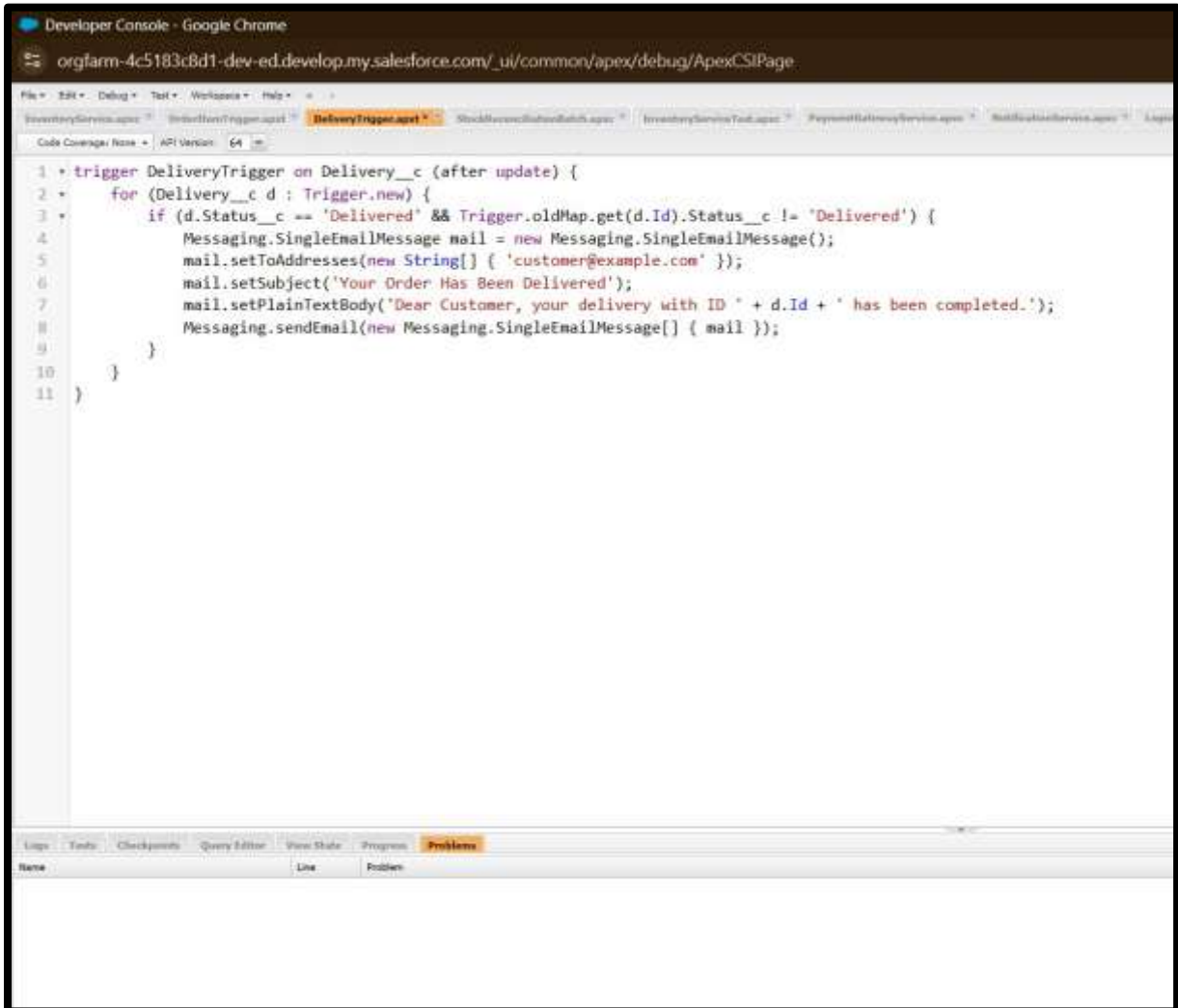
Apex Triggers

- OrderItemTrigger on Order_Item__c

- Events: before insert (validate stock availability), after insert (adjust stock), after delete (restore stock).
- Purpose: Prevents creating order items if stock unavailable and updates inventory after successful insert/delete.

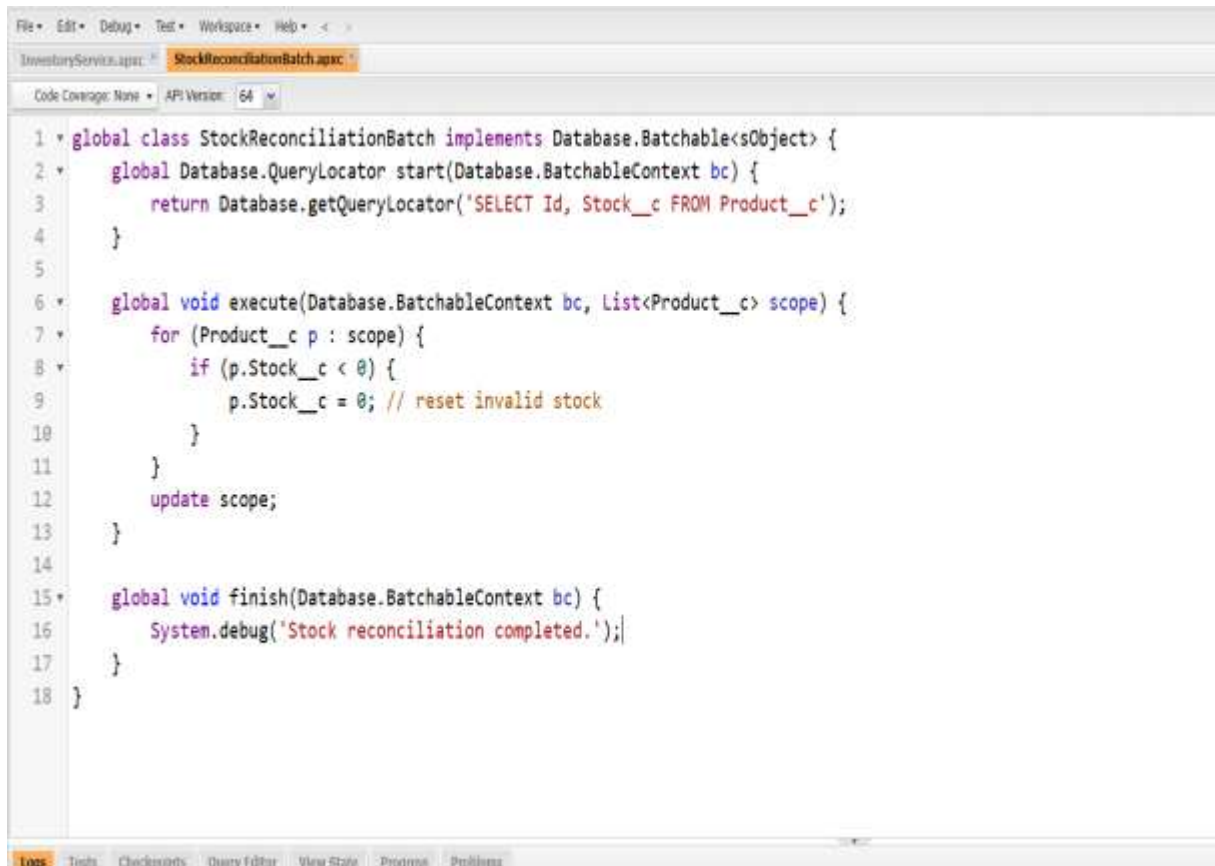


- **DeliveryTrigger** on **Delivery__c**
 - Events: after update
 - Purpose: When **Status__c** changes to **Delivered**, sends email/notification to the customer (demo sends email or logs an outgoing message).



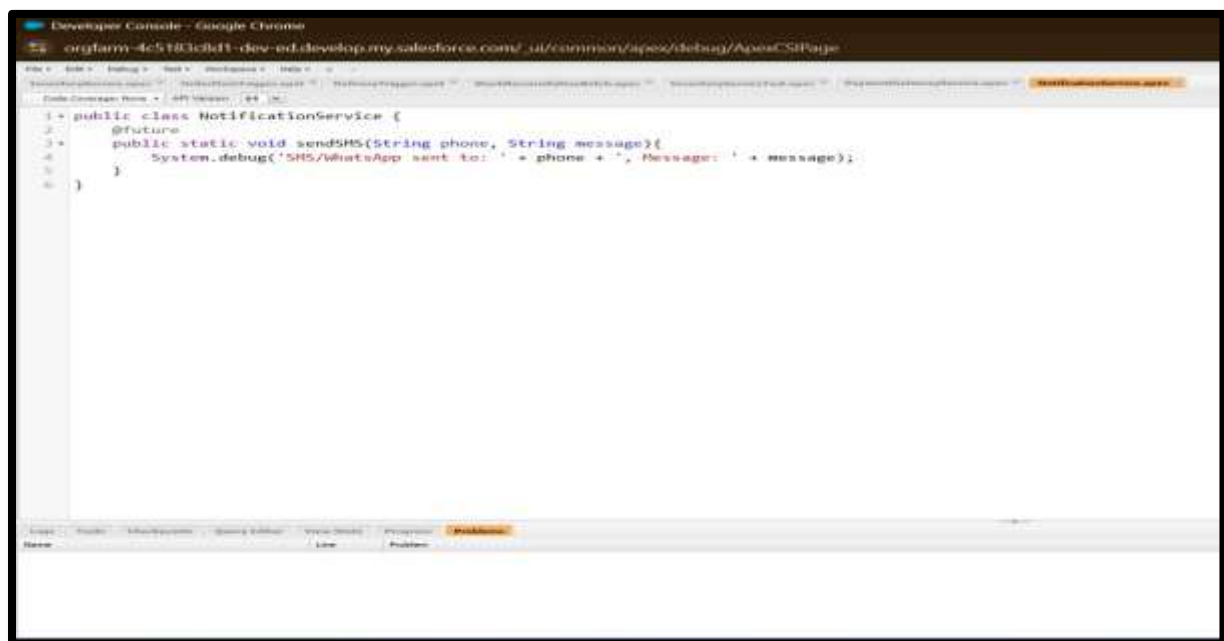
Asynchronous Apex

- **StockReconciliationBatch** (implements `Database.Batchable<sObject>`)
 - Purpose: Nightly job to reconcile and sanitize product stock (e.g., set negative stock to 0, recalc aggregates).
 - How to run: `Database.executeBatch(new StockReconciliationBatch(), 200);`



```
1 global class StockReconciliationBatch implements Database.Batchable<sObject> {
2     global Database.QueryLocator start(Database.BatchableContext bc) {
3         return Database.getQueryLocator('SELECT Id, Stock__c FROM Product__c');
4     }
5
6     global void execute(Database.BatchableContext bc, List<Product__c> scope) {
7         for (Product__c p : scope) {
8             if (p.Stock__c < 0) {
9                 p.Stock__c = 0; // reset invalid stock
10            }
11        }
12        update scope;
13    }
14
15    global void finish(Database.BatchableContext bc) {
16        System.debug('Stock reconciliation completed.');
```

- **NotificationService.sendSMS (@future)**
 - Purpose: Offload external notification calls.



Testing & Coverage

- **Test classes** created (examples):
 - `InventoryServiceTest` — sets up `Product`, `Customer`, `Order` and asserts stock adjustments after inserting an `Order_Item__c`.



- DeliveryTriggerTest — verifies email/notification is sent when Delivery status updates to Delivered.
 - StockReconciliationBatchTest — simulates negative stock and asserts batch fixes it.
 - **Custom Exceptions** implemented in Apex to handle business errors (e.g., InsufficientStockException).
 - **Reported coverage:** Test classes were written to achieve **>75% code coverage** for the Apex components (please run tests in your org to confirm actual coverage percentages).
-