Phase 5 — Apex Programming (Developer) — Step-by-Step

Step 1 — Plan & Scope

1.1 Identify scenarios requiring Apex (complex routing decisions, external callouts, bulk recalculation of routing rules, audit logging).
1.2 Map which behaviors remain declarative (Flows, Assignment Rules, Omni-Channel) and which need Apex (bulk callouts, heavy processing, retries).
1.3 Design data contracts: what Apex will read/write (Case fields, RoutingRule__c, CaseAssignment__c, Feedback__c).

Step 2 — Developer Environment & Version Control

2.1 Install Salesforce CLI + VS Code + Salesforce Extension Pack.
2.2 Create a scratch org or sandbox for development.
2.3 Initialize a Git repo (feature branches) and use SFDX for metadata tracking.

Step 3 — Trigger Framework & Design Pattern

3.1 Implement “one trigger per object” + separate handler class pattern.
3.2 Create a generic trigger template for Case with delegated handler calls (before/after insert/update/delete).
3.3 Keep trigger bodies minimal — only orchestration.

Step 4 — Implement Case Trigger & Handler (basic)

4.1 **Before insert / update**: data validation (required Category__c, Severity__c rules).
4.2 **After insert / after update**: evaluate routing (call RoutingService), persist CaseAssignment__c entries for audit, set/ change OwnerId via DML when needed.
4.3 Ensure handler methods accept Lists and Maps (bulkified signatures).

Step 5 — Bulkification & Collections Best Practices

5.1 Collect IDs and aggregate data outside loops (use Sets for IDs).
5.2 Use Maps for lookup maps (Map<Id, RoutingRule__c>, Map<Id, User>).
5.3 Do not perform SOQL/DML inside loops — perform single queries and batched DML.

Step 6 — SOQL & SOSL Hygiene

6.1 Query only needed fields.
6.2 Use FOR loops with sub-selects only when efficient.
6.3 Use aggregate queries for counts/metrics where appropriate.

Step 7 — Asynchronous Patterns

7.1 **Queueable Apex**: for post-assignment processing and callouts (supports chaining).
7.2 **Batch Apex**: for nightly re-evaluation of routing rules across large case sets (use Database.Batchable).
7.3 **Scheduled Apex**: schedule batch or maintenance jobs (SLA health checks, rebalancing workloads).
7.4 Prefer Queueable over @future; use @future only for very small, legacy needs.

Step 8 — External Callouts & Integrations

8.1 Use Named Credentials and Auth Providers for secure callouts.
8.2 Implement Database.AllowsCallouts in Queueable/Batch if making HTTP requests.
8.3 Use HttpCalloutMock in tests for deterministic behavior.

Step 9 — Platform Events / Event-Driven Decoupling (optional but recommended)

9.1 Publish a Platform Event when assignment decisions are made (for analytics, downstream sync).
9.2 Create subscribers (Apex Trigger on Platform Event or external system).
9.3 Use events to decouple heavy integrations from synchronous case creation.

Step 10 — Logging, Exception Handling & Retries

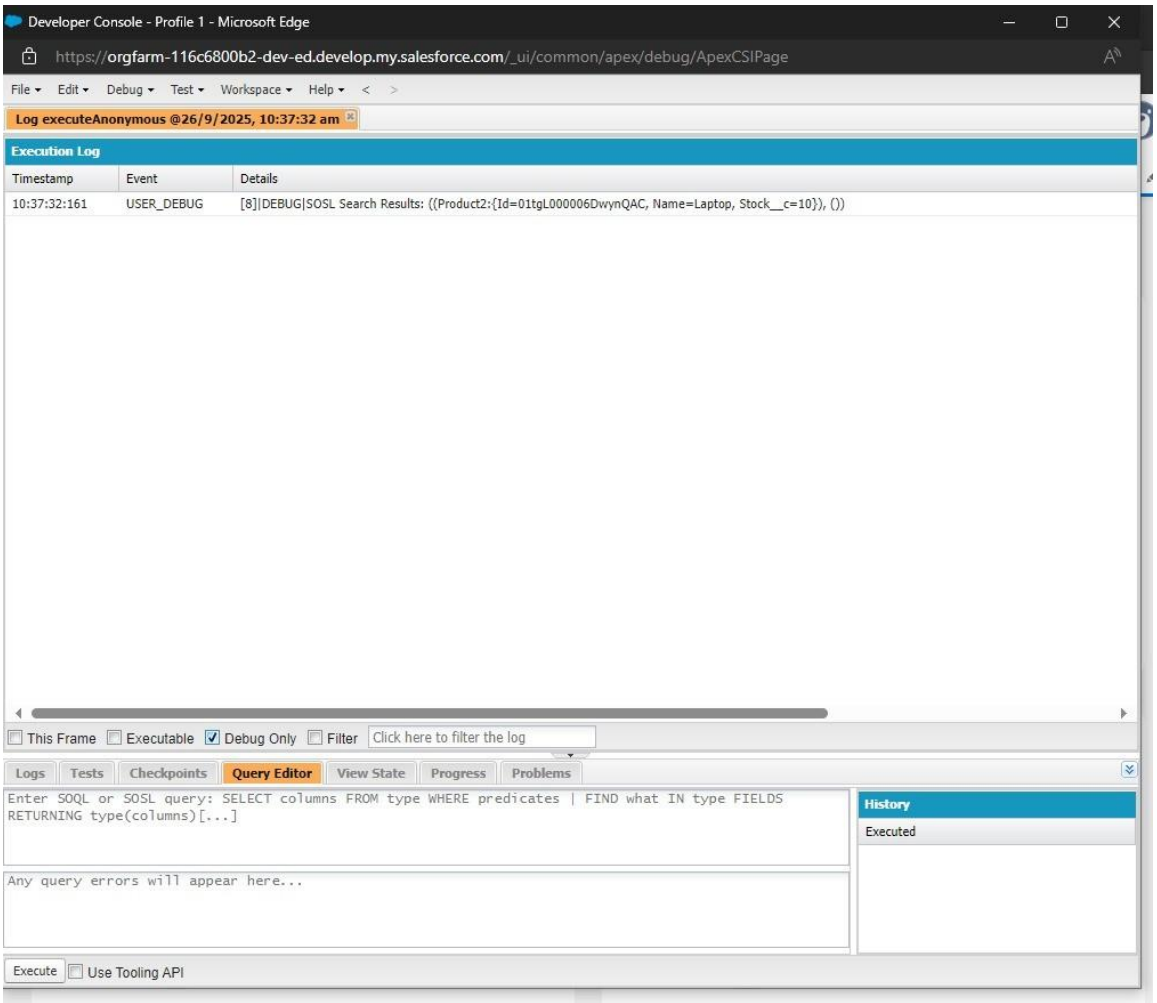
10.1 Wrap callouts and complex logic in try/catch and create Audit__c or Apex_Error__c records for failed flows.
10.2 For transient failures, enqueue a Queueable retry with exponential backoff (store retry count).
10.3 Avoid surfacing raw exception messages to end users — log details and show friendly messages.

Step 11 — Security & Sharing

11.1 Use with sharing / without sharing intentionally; prefer with sharing for data-sensitive operations.
11.2 Respect CRUD/FLS — use Schema.sObjectType checks or Security.stripInaccessible as needed.
11.3 Ensure Apex runs with the appropriate user context for assignments.

Step 12 — Test Strategy & Quality Gates

12.1 Create @IsTest classes for every class/trigger. Cover happy path, bulk path, negative path, and callout scenarios.
12.2 Use Test.startTest() / Test.stopTest() to simulate async jobs and execute scheduled/batch jobs.
12.3 Mock HTTP callouts with HttpCalloutMock; assert logs and CaseAssignment__c created.
12.4 Maintain code coverage > 75% and assert functional correctness, not only lines covered.

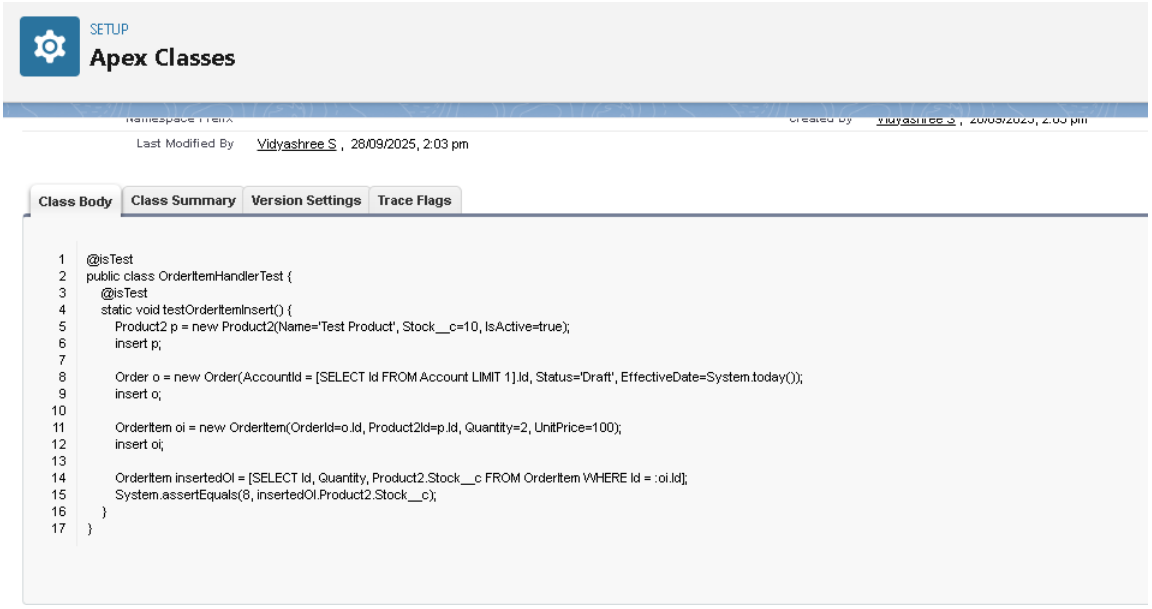


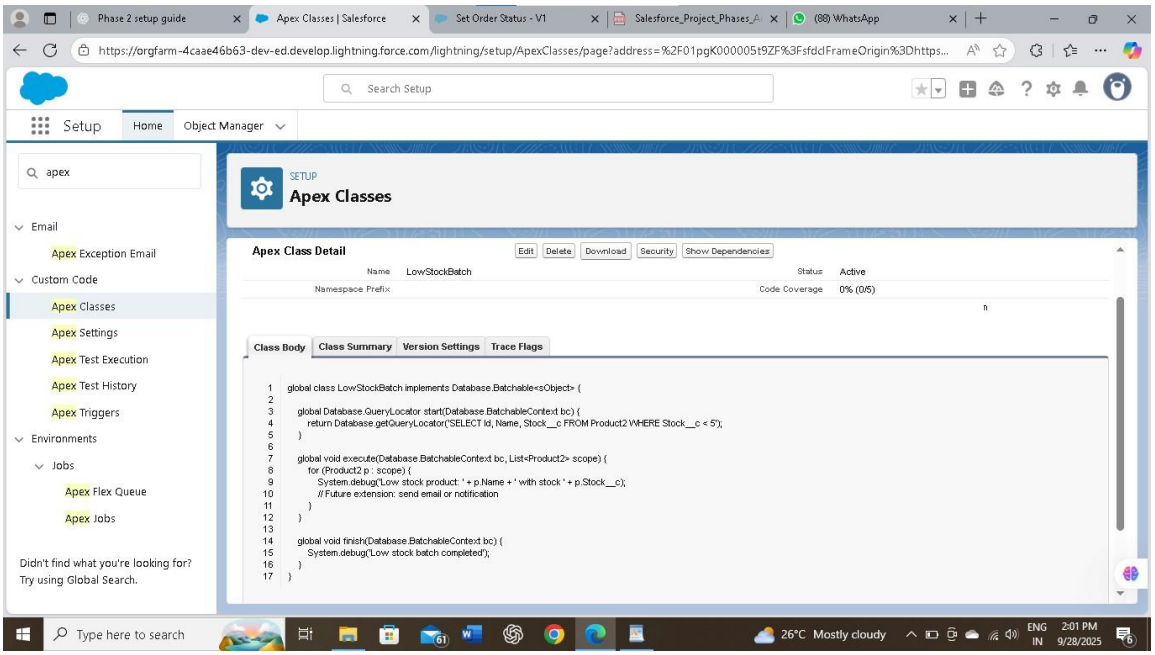
Step 13 — Deployment & CI

13.1 Use SFDX or CI (GitHub Actions / Jenkins) to run apex tests on each pull request.
13.2 Deploy only validated change sets or SFDX packages; include test run results.
13.3 Maintain release rollback steps (deactivate flows, revert apex versions).

Step 14 — Monitoring & Alerts

14.1 Use custom objects (CaseAssignment__c) + reports to monitor assignment rates, failed assignments, and SLA breaches.
14.2 Enable Apex Exception Email alerts and set up logging dashboards.
14.3 Schedule regular batch jobs to produce summary reports for managers.





Quick, copy-ready code examples

A. Case Trigger (one trigger only)

```
trigger CaseTrigger on Case (before insert, before update, after insert, after update) {
    if (Trigger.isBefore) {
        if (Trigger.isInsert) CaseTriggerHandler.beforeInsert(Trigger.new);
        if (Trigger.isUpdate) CaseTriggerHandler.beforeUpdate(Trigger.new, Trigger.oldMap);
    }
    if (Trigger.isAfter) {
        if (Trigger.isInsert) CaseTriggerHandler.afterInsert(Trigger.new);
        if (Trigger.isUpdate) CaseTriggerHandler.afterUpdate(Trigger.newMap);
    }
}
```

B. Minimal Handler skeleton

```
public with sharing class CaseTriggerHandler {
    public static void beforeInsert(List<Case> newCases) {
        for (Case c : newCases) {
            if (String.isBlank(c.Category__c)) {
                c.addError('Please select a Category for this case.');
            }
        }
    }

    public static void afterInsert(List<Case> newCases) {
        // collect ids for async processing or immediate assignment
        List<Id> ids = new List<Id>();
        for (Case c : newCases) ids.add(c.Id);
        // enqueue assignment job to keep insert fast and avoid long transactions
        System.enqueueJob(new CaseAssignmentQueueable(ids));
    }
```

```
    // Implement other lifecycle methods similarly, always bulk-safe
}
```

C. Queueable assignment job (example)

```
public class CaseAssignmentQueueable implements Queueable, Database.AllowsCallouts {
    private List<Id> caseIds;
    public CaseAssignmentQueueable(List<Id> ids) { this.caseIds = ids; }

    public void execute(QueueableContext ctx) {
        List<Case> cases = [SELECT Id, Category__c, Severity__c FROM Case WHERE Id IN :caseIds];
        List<Case> updates = new List<Case>();
        List<CaseAssignment__c> audit = new List<CaseAssignment__c>();

        for (Case c : cases) {
            Id assignee = RoutingService.findAssignee(c); // implement lookup logic in RoutingService
            if (assignee != null) {
                updates.add(new Case(Id = c.Id, OwnerId = assignee));
                audit.add(new CaseAssignment__c(Case__c = c.Id, AssignedTo__c = assignee));
            }
        }
        if (!updates.isEmpty()) update updates;
        if (!audit.isEmpty()) insert audit;
    }
}
```

D. Outline of a simple test

```
@IsTest
private class CaseAssignmentTest {
```

```
    @IsTest static void testQueueableAssignment() {
        // Setup test data
        Account acc = new Account(Name='Tst'); insert acc;
        Case c = new Case(Subject='T', Status='New', AccountId=acc.Id, Category__c='Billing');
        Test.startTest();
            insert c;
            // execute queued jobs
        Test.stopTest();

        // Assert assignment audit record created or Owner changed
        Integer auditCount = [SELECT COUNT() FROM CaseAssignment__c WHERE Case__c =
:c.Id];
        System.assertEquals(1, auditCount);
    }
}
```