Phase 5: Apex Programming (Developer)

Objective

To extend Salesforce functionality beyond declarative tools (Flows and Validation Rules) using **Apex Triggers, Classes, and Test Methods**. Apex ensures **scalability, execution of complex logic**, and handling of **bulk data operations** that cannot be fully managed via point-and-click automation.

Key Apex Use Cases in This Project

1. Case Auto-Assignment Trigger

- **Purpose:** Automatically assign newly created cases to the appropriate agent if not manually assigned.
- **Logic:**
    - Trigger executes **before insert** of Case.
    - If **Assigned Agent** is blank → auto-populate with the agent who has the **least workload**.

2. SLA Escalation Handler (Future Enhancement)

- **Purpose:** Escalate high-priority cases automatically if they are approaching SLA deadlines.
- **Logic:**
    - Apex Class evaluates **case priority and created date**.
    - Updates case owner or triggers escalation email if SLA is near breach.

3. Bulk Email Handler

- **Purpose:** Send case notifications to customers in bulk when Flow limits are exceeded.
- **Logic:**
    - Apex Batch Class processes multiple case records at once.
    - Uses Messaging.sendEmail() to send emails without hitting governor limits.

4. Custom Validation via Apex

- **Purpose:** Implement advanced business rules that cannot be handled by standard validation rules.
- **Example:** Prevent duplicate cases for the same customer, same subject, and same date.

Apex Snippets

Trigger Example – Auto-Assign Case to Agent

```
trigger CaseAssignmentTrigger on Case (before insert) {
  List<User> agents = [SELECT Id FROM User WHERE Profile.Name='Support Agent' ORDER
BY Workload__c ASC];
  for(Case c : Trigger.new){
    if(c.OwnerId == null && agents.size() > 0){
      c.OwnerId = agents[0].Id;
    }
  }
}
```

Batch Apex Example – Bulk Case Notification

```
global class CaseNotificationBatch implements Database.Batchable<sObject> {
  global Database.QueryLocator start(Database.BatchableContext bc) {
    return Database.getQueryLocator([SELECT Id, Contact.Email FROM Case WHERE
Status='New']);
  }
  global void execute(Database.BatchableContext bc, List<Case> cases) {
    List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();
    for(Case c : cases){
      Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
      mail.setToAddresses(new String[] {c.Contact.Email});
      mail.setSubject('Your Case is Received');
      mail.setPlainTextBody('Dear Customer, your case has been received and is under review.');
      emails.add(mail);
    }
    Messaging.sendEmail(emails);
  }
  global void finish(Database.BatchableContext bc) {}
}
```

Tabular Summary

Apex Component Type	**Purpose**
CaseAssignmentTrigger (Trigger)	Auto-assigns Case to agent if Owner is missing.
SLAEscalationHandler (Apex Class)	Escalates high-priority cases nearing SLA breach (future enhancement).
CaseNotificationBatch (Batch Apex)	Sends bulk case notifications without Flow limits.
DuplicateCaseValidator (Trigger)	Prevents duplicate cases from being recorded.
Test Classes (Apex Tests)	Ensure 75%+ coverage for deployment readiness.

Benefits of Apex Programming

- Handles **bulk processing** for large case volumes using Batch Apex.
- Implements **complex logic** not possible in Flows (e.g., SLA escalations, duplicate case prevention).
- Prepares the system for **future scalability** and advanced automation.
- Ensures **deployment readiness** by meeting Salesforce test coverage requirements (≥ 75%).