Phase 5: Apex Programming (Developer)

Project: Customer Complaint Management System

1. Apex Classes & Objects

Apex is Salesforce's **object-oriented programming language**, similar to Java, used to implement business logic on the Salesforce platform.

Example:

```
public class ComplaintHandler {
    public void assignAgent(Complaint__c comp) {
        if(comp.Issue_Type__c == 'Technical') {
            comp.Assigned_Agent__c = '0055g00000ABCDEF'; // Tech Agent User ID
        } else if(comp.Issue_Type__c == 'Billing') {
            comp.Assigned_Agent__c = '0055g00000XYZABC'; // Billing Agent
        }
    }
}
```

✓ **Use:** This class contains the logic for assigning agents based on complaint type.

2. Apex Triggers (Before/After Insert, Update, Delete)

Triggers execute **automatically** before or after data changes occur in Salesforce objects.

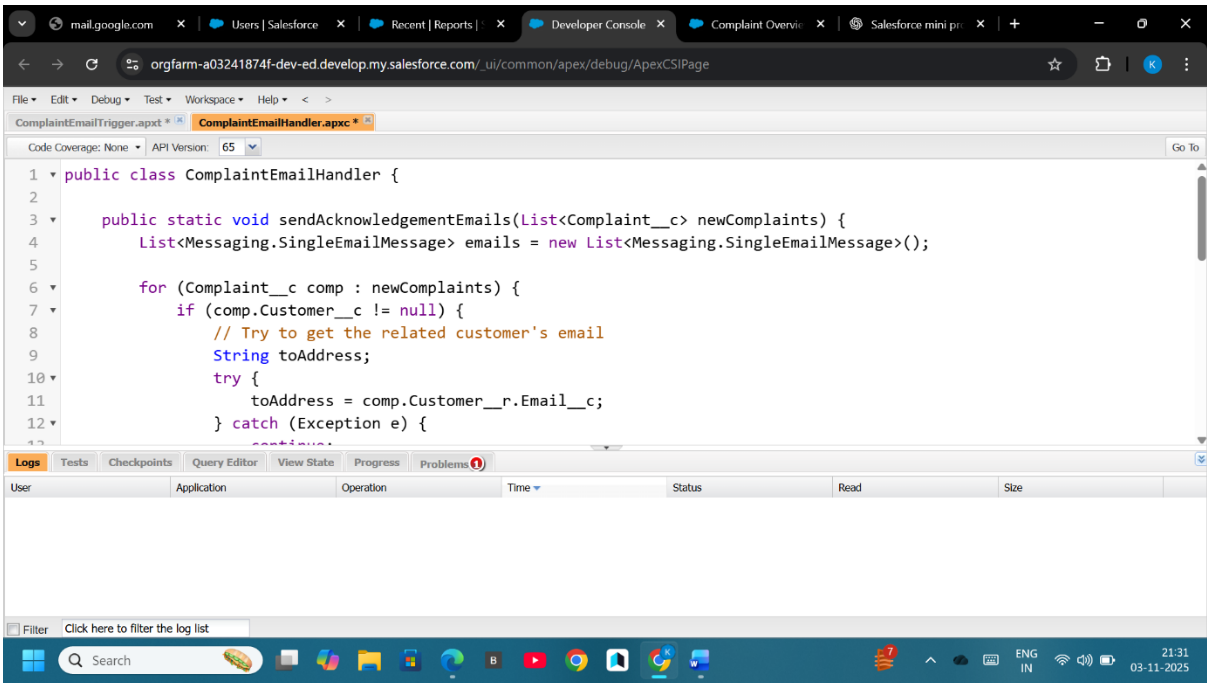
Trigger Types:

- **Before Insert / Before Update:** Modify field values before saving.
- **After Insert / After Update:** Perform actions after record is saved.
- **Before Delete / After Delete:** Clean up related data when deleted.

Example Trigger – Auto Assign Agent

```
trigger ComplaintTrigger on Complaint__c (before insert) {
    for(Complaint__c c : Trigger.new) {
        if(c.Issue_Type__c == 'Technical')
            c.Assigned_Agent__c = '0055g00000ABCDEF';
        else if(c.Issue_Type__c == 'Billing')
            c.Assigned_Agent__c = '0055g00000XYZABC';
        c.Status__c = 'Assigned';
    }
}
```

☑ **Use:** Automatically assigns an agent when a complaint is created.


mail.google.com
Users | Salesforce
Recent | Reports |
Developer Console
Complaint Overvie
Salesforce mini pr
orgfarm-a03241874f-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
File Edit Debug Test Workspace Help
ComplaintEmailTrigger.apxt *
ComplaintEmailHandler.apxc *
Code Coverage: None
API Version: 65
Go To
Logs Tests Checkpoints Query Editor View State Progress Problems
User Application Operation Time Status Read Size
Filter Click here to filter the log list
Search
ENG IN
21:31
03-11-2025

```
public class ComplaintEmailHandler {

    public static void sendAcknowledgementEmails(List<Complaint__c> newComplaints) {
        List<Messaging.SingleEmailMessage> emails = new List<Messaging.SingleEmailMessage>();

        for (Complaint__c comp : newComplaints) {
            if (comp.Customer__c != null) {
                // Try to get the related customer's email
                String toAddress;
                try {
                    toAddress = comp.Customer__r.Email__c;
                } catch (Exception e) {
```

3. Trigger Design Pattern

To avoid logic duplication and errors, follow the **Trigger Framework pattern**:

- Create **one trigger per object**
- Call logic from **Handler Class**

Example:

```
// Trigger
trigger ComplaintTrigger on Complaint__c (before insert, before update) {
    ComplaintTriggerHandler.run(trigger.new);
}


// Handler
public class ComplaintTriggerHandler {
    public static void run(List<Complaint__c> compList) {
        for(Complaint__c c : compList) {
            if(c.Issue_Type__c == 'Technical') {
                c.Assigned_Agent__c = '0055g00000ABCDEF';
            }
        }
    }
```

}

✅ **Use:** Keeps triggers clean and easier to maintain.

4. SOQL & SOSL

SOQL (Salesforce Object Query Language)

Used to query records from Salesforce.

Example:

```
List<Complaint__c> openComplaints = [SELECT Id, Status__c FROM Complaint__c WHERE
Status__c != 'Closed'];
```

SOSL (Salesforce Object Search Language)

Used to search text across multiple objects.

Example:

```
List<List<SObject>> results = [FIND 'Network Issue' IN ALL FIELDS RETURNING
Complaint__c(Id, Name)];
```

✅ **Use:** Fetch and search complaint data efficiently.

5. Collections (List, Set, Map)

List

Stores ordered records.

```
List<String> issueTypes = new List<String>{'Technical','Billing','Service'};
```

Set

Stores unique values (no duplicates).

```
Set<String> priorities = new Set<String>{'High','Medium','Low'};
```

Map

Stores key-value pairs.

```
Map<Id, Complaint__c> complaintMap = new Map<Id, Complaint__c>([SELECT Id, Name FROM
Complaint__c]);
```

✅ **Use:** For bulk processing and data organization.

6. Control Statements

Used to define logic and flow in Apex.

Example:

```
if(c.Priority__c == 'High') {
   sendEscalationEmail(c.Id);
} else {
   System.debug('Normal priority complaint');
}
```

✅ **Use:** Implement conditional logic for complaint escalation.

7. Batch Apex

Used to handle **large data volumes** (above governor limits).

Example:

```
global class CloseOldComplaintsBatch implements Database.Batchable<sObject> {
    global Database.QueryLocator start(Database.BatchableContext bc) {
        return Database.getQueryLocator('SELECT Id FROM Complaint__c WHERE Status__c = 
\'Resolved\'');
    }
    global void execute(Database.BatchableContext bc, List<Complaint__c> complaints) {
        for(Complaint__c c : complaints) c.Status__c = 'Closed';
        update complaints;
    }
    global void finish(Database.BatchableContext bc) {
        System.debug('Batch process complete');
    }
}
```

✅ **Use:** Automatically closes resolved complaints after a period.

8. Queueable Apex

Used for **asynchronous processing** with chaining support.

Example:

```
public class ComplaintFollowUpQueueable implements Queueable {
    public void execute(QueueableContext context) {
        // Send follow-up emails to customers
        System.debug('Follow-up task executed');
    }
}
```

✅ **Use:** To send follow-up emails in the background without slowing user operations.

9. Scheduled Apex

Runs code automatically at specified times.

Example:

```
global class ComplaintScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        CloseOldComplaintsBatch batch = new CloseOldComplaintsBatch();
        Database.executeBatch(batch);
    }
}
```

To schedule:

- Go to **Setup → Apex Classes → Schedule Apex**
- Choose frequency (daily/weekly)

✅ **Use:** Daily complaint maintenance or SLA checks.

10. Future Methods

Used for asynchronous calls, especially for callouts or delayed actions.

Example:

```
public class ComplaintNotifier {
    @future
    public static void sendEmail(Id complaintId) {
        System.debug('Email sent for complaint: ' + complaintId);
    }
}
```

✅ **Use:** To send emails after record save without slowing the transaction.

11. Exception Handling

Used to prevent runtime errors and show meaningful messages.

Example:

```
try {
    update complaintList;
} catch(DmlException e) {
    System.debug('Error updating complaints: ' + e.getMessage());
}
```

✅ **Use:** Ensures graceful error handling.

12. Test Classes

Used to test Apex logic and ensure 75% or more code coverage (mandatory for deployment).

Example:

```
@isTest
public class ComplaintTriggerTest {
   @isTest
   static void testComplaintInsert() {
      Complaint__c c = new Complaint__c(Issue_Type__c='Technical', Priority__c='High');
      insert c;
      System.assertEquals('Assigned', c.Status__c);
   }
}
```

✅ **Use:** Ensures that triggers and classes work correctly before deployment.

13. Asynchronous Processing

Apex provides several asynchronous options:

Method	**Use Case**
@future	Send emails or perform small background tasks
Batch Apex	Handle large data operations
Queueable Apex	Flexible background jobs with chaining
Scheduled Apex	Run automation at specific intervals

✅ **Use:** Keeps system performance fast and avoids hitting limits.

Conclusion

In Phase 5, we implemented **custom logic using Apex programming**, enhancing the CRM's intelligence and automation.

✅ **Key Outcomes:**

- Automated complaint assignments using triggers
- Optimized data handling using SOQL/SOSL
- Used Batch, Queueable, and Scheduled Apex for performance
- Ensured quality through exception handling and test classes

This phase brings **developer-level customization** and efficiency to the **Customer Complaint Management System**.