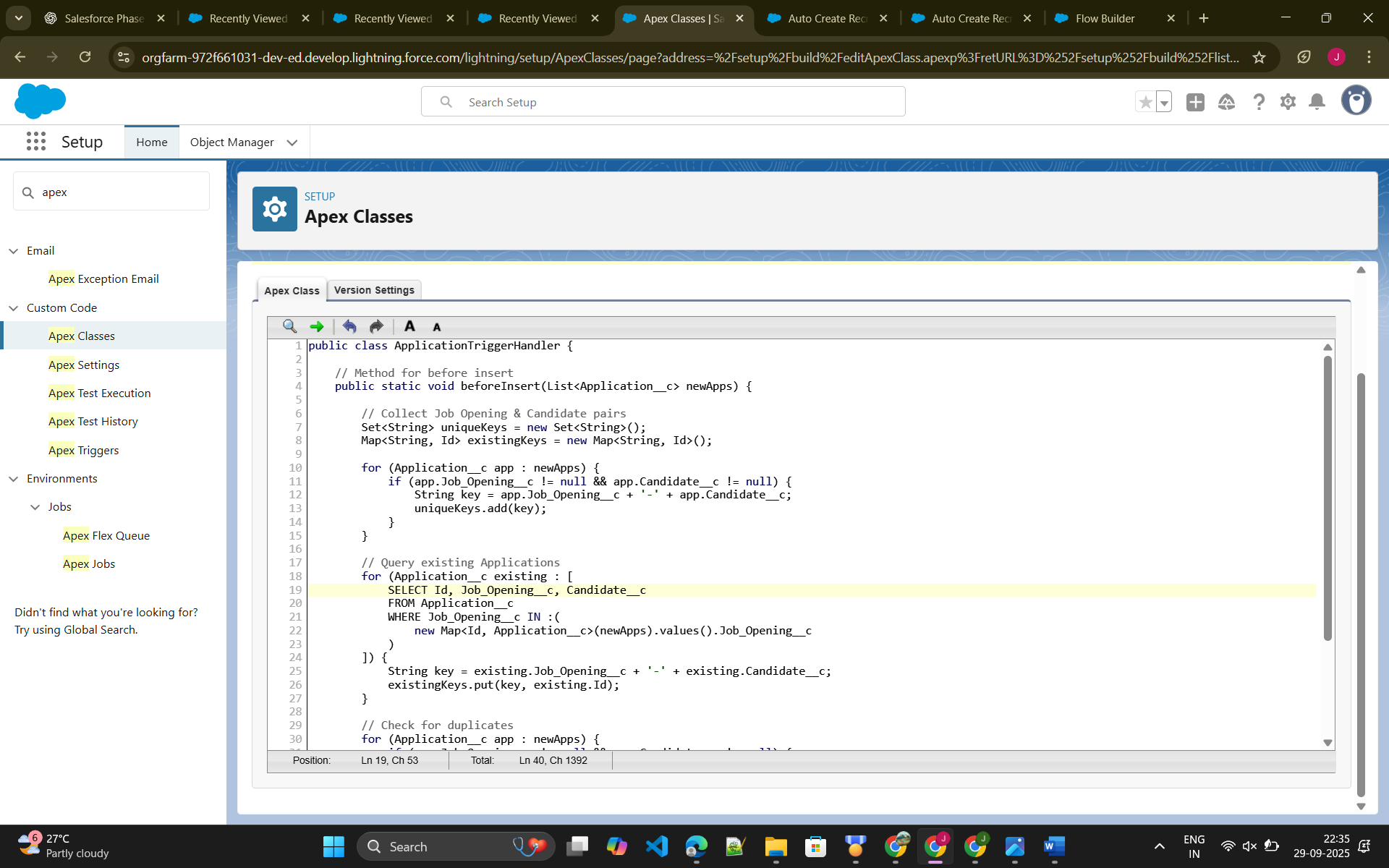
# Phase 5: Apex Programming (Developer) - Recruitment App

Goal: Add advanced logic and automation to the Recruitment App.

## 1. Classes & Objects

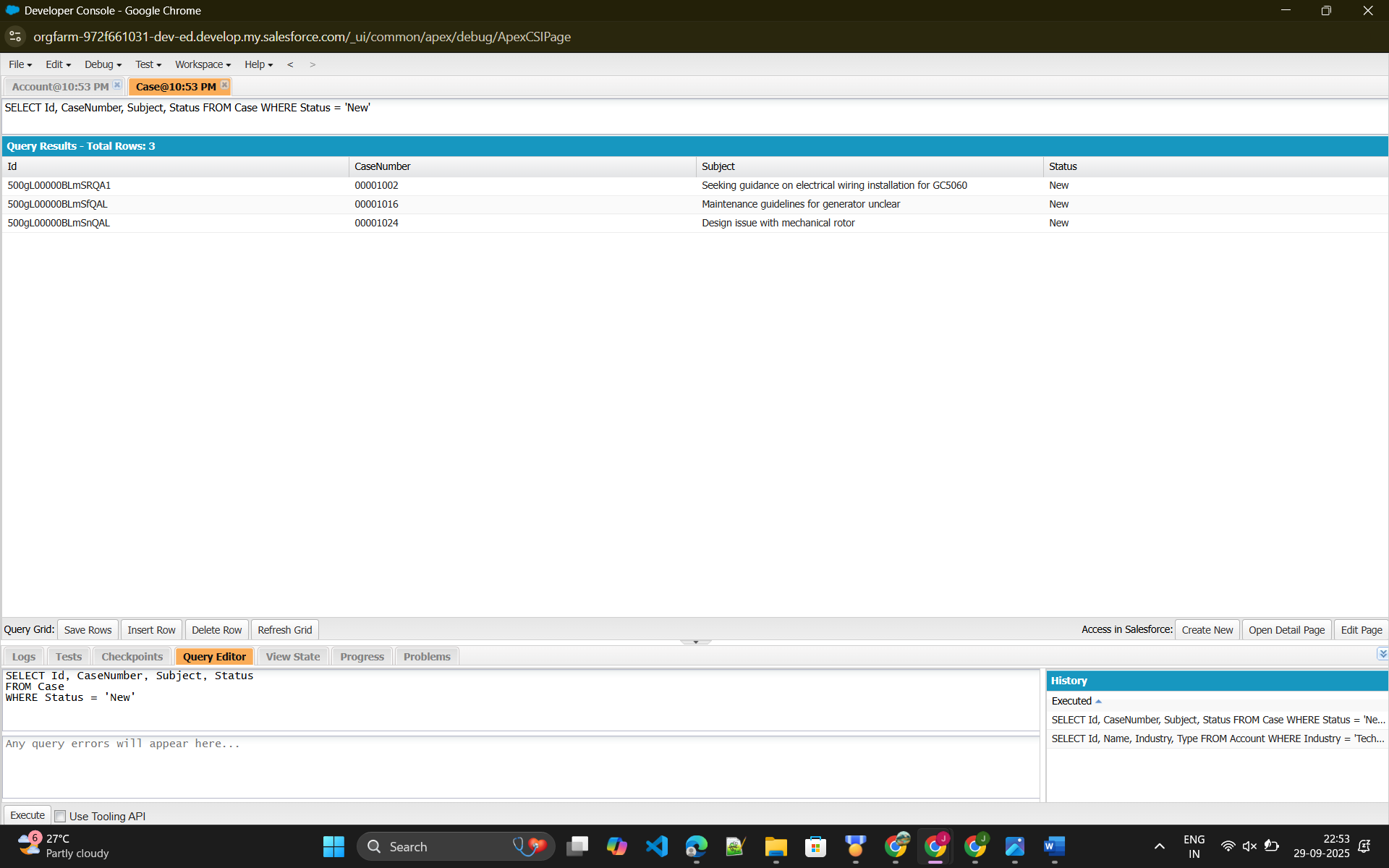
RecruitmentService class centralizes common logic.  
  
Example:  
public class RecruitmentService {  
 public static Boolean checkEligibility(Contact c) {  
 return (c.Experience\_\_c >= 2);  
 }  
}

## 2. Apex Triggers

Trigger on Application insert to prevent duplicate applications.  
  
trigger ApplicationTrigger on Application\_\_c (before insert) {  
 Set<String> appKeys = new Set<String>();  
 for (Application\_\_c a : Trigger.new) {  
 appKeys.add(a.Candidate\_\_c + '-' + a.Job\_Opening\_\_c);  
 }  
  
 List<Application\_\_c> existingApps = [  
 SELECT Candidate\_\_c, Job\_Opening\_\_c FROM Application\_\_c  
 WHERE Candidate\_\_c IN :Trigger.newMap.keySet()  
 ];  
  
 for (Application\_\_c e : existingApps) {  
 for (Application\_\_c n : Trigger.new) {  
 if (e.Candidate\_\_c == n.Candidate\_\_c && e.Job\_Opening\_\_c == n.Job\_Opening\_\_c) {  
 n.addError('Duplicate application detected for this candidate and job opening.');  
 }  
 }  
 }  
}

## 3. Trigger Design Pattern

Handler class to keep trigger logic clean.  
  
public class ApplicationHandler {  
 public static void beforeInsert(List<Application\_\_c> newApps) {  
 // logic from trigger moved here  
 }  
}

**4. SOQL & SOSL** Example query:  
List<Contact> candidates = [SELECT Id, Name FROM Contact WHERE Status\_\_c ='Available']

## 5. Collections

Used Set to avoid duplicates and List for bulk operations.

## 6. Control Statements

Used IF statements to validate overlapping applications and throw errors.

## 7. Batch Apex

Nightly batch job to flag applications pending for >30 days as 'Stale'.  
  
global class MarkStaleApplications implements Database.Batchable<sObject> {  
 global Database.QueryLocator start(Database.BatchableContext bc) {  
 return Database.getQueryLocator(  
 'SELECT Id, Status\_\_c FROM Application\_\_c WHERE Status\_\_c = \'Pending\' AND CreatedDate < LAST\_N\_DAYS:30'  
 );  
 }  
  
 global void execute(Database.BatchableContext bc, List<Application\_\_c> scope) {  
 for (Application\_\_c app : scope) app.Status\_\_c = 'Stale';  
 update scope;  
 }  
  
 global void finish(Database.BatchableContext bc) {}  
}

## 8. Queueable Apex

Used for bulk notifications.  
  
public class NotifyCandidatesQueueable implements Queueable {  
 public void execute(QueueableContext context) {  
 // logic to send notifications  
 }  
}

## 9. Scheduled Apex

Daily morning job to email recruiters with list of interviews.  
  
global class DailyInterviewEmail implements Schedulable {  
 global void execute(SchedulableContext sc) {  
 // logic to query interviews and send emails  
 }  
}

## 10. Future Methods

Integrated with external background-check API asynchronously.  
  
@future  
public static void callBackgroundCheck(Id candidateId) {  
 // API call logic  
}

## 11. Exception Handling

Used try-catch blocks to handle runtime errors gracefully.

## 12. Test Classes

Created test classes with >75% coverage.  
  
@isTest  
private class ApplicationTriggerTest {  
 @isTest  
 static void testDuplicatePrevention() {  
 // create test data and assert errors  
 }  
}

## 13. Asynchronous Processing

Combined Batch Apex, Queueable, and Future Methods for high-volume tasks.