Phase 5: Apex Programming (Developer)

1. Introduction

While admin automation (Flows and Validation Rules) handles simple scenarios, Apex programming allows us to implement custom logic that cannot be achieved through standard Salesforce tools. In this project, Apex is used to prevent overlapping leave requests for the same employee, ensuring data integrity and realistic leave tracking.

2. Apex Trigger: Prevent Overlapping Leaves

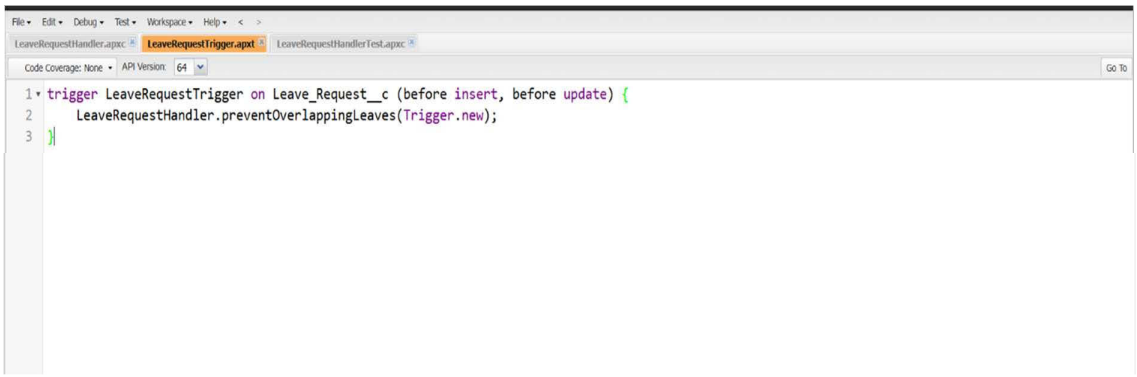
Purpose:

- Ensure that an employee cannot submit a new leave request that overlaps with an existing approved or pending leave.
- Prevents conflicts and errors in leave planning.

Implementation Steps:

1. Go to Setup → Object Manager → Leave_Request__c → Triggers → New
2. Write a trigger that calls a Handler Class for logic separation:

   Trigger (LeaveRequestTrigger.cls)



3. Apex Handler Class

Purpose:

- Encapsulates the logic to check for overlapping leave requests.
- Keeps the trigger clean and maintainable.

LeaveRequestHandler.cls

```
public class LeaveRequestHandler {
    public static void preventOverlappingLeaves(List<Leave_Request__c> leaveRequests) {
        Set<Id> employeeIds = new Set<Id>();
        for(Leave_Request__c lr : leaveRequests){
            if(lr.Employee__c != null) employeeIds.add(lr.Employee__c);
        }

        List<Leave_Request__c> existingLeaves = [
            SELECT Id, Start_Date__c, End_Date__c, Employee__c
            FROM Leave_Request__c
            WHERE Employee__c IN :employeeIds
        ];

        for(Leave_Request__c newLeave : leaveRequests){
            for(Leave_Request__c existing : existingLeaves){
                if(existing.Id != newLeave.Id &&
                   newLeave.Start_Date__c <= existing.End_Date__c &&
                   newLeave.End_Date__c >= existing.Start_Date__c){
                    newLeave.addError('This leave overlaps with an existing leave.');
                }
            }
        }
    }
}
```

4. Test Class

Purpose:

- Ensure the trigger works correctly.
- Covers valid and invalid scenarios to pass deployment requirements.

Test Class (LeaveRequestHandlerTest.cls):

```
@isTest
public class LeaveRequestHandlerTest {
    @isTest static void testOverlappingLeave() {
        Employee__c emp = new Employee__c(Name='Sathvika', E_Mail__c ='sathvika@test.com', Department__c='IT');
        insert emp;

        Leave_Request__c leave1 = new Leave_Request__c(
            Employee__c = emp.Id,
            Start_Date__c = Date.today(),
            End_Date__c = Date.today().addDays(2),
            Leave_Type__c = 'Casual',
            Status__c = 'New'
        );
        insert leave1;

        Leave_Request__c leave2 = new Leave_Request__c(
            Employee__c = emp.Id,
            Start_Date__c = Date.today().addDays(1),
            End_Date__c = Date.today().addDays(3),
            Leave_Type__c = 'Casual',
            Status__c = 'New'
        );

        Test.startTest();
        try {
            insert leave2;
        } catch (DmlException e) {
```


```
        Leave_Request__c leave1 = new Leave_Request__c(
            Employee__c = emp.Id,
            Start_Date__c = Date.today(),
            End_Date__c = Date.today().addDays(2),
            Leave_Type__c = 'Casual',
            Status__c = 'New'
        );
        insert leave1;

        Leave_Request__c leave2 = new Leave_Request__c(
            Employee__c = emp.Id,
            Start_Date__c = Date.today().addDays(1),
            End_Date__c = Date.today().addDays(3),
            Leave_Type__c = 'Casual',
            Status__c = 'New'
        );

        Test.startTest();
        try {
            insert leave2;
        } catch (DmlException e) {
```


```
            System.assert(e.getMessage().contains('overlaps'));
        }
        Test.stopTest();
    }
}
```

5. Outcome of Phase 5

- The Apex trigger and handler prevent overlapping leaves.
- Test class ensures deployment readiness by covering positive and negative cases.
- Employees cannot create conflicting leave requests, maintaining data integrity.

6. Conclusion

- Apex programming in Phase 5 adds **developer-level functionality** that cannot be achieved through standard Salesforce tools. This ensures the Leave Management System is robust, accurate, and aligned with real-world business requirements.