Phase 5 Report
Apex Programming (Developer)

1. Classes & Objects

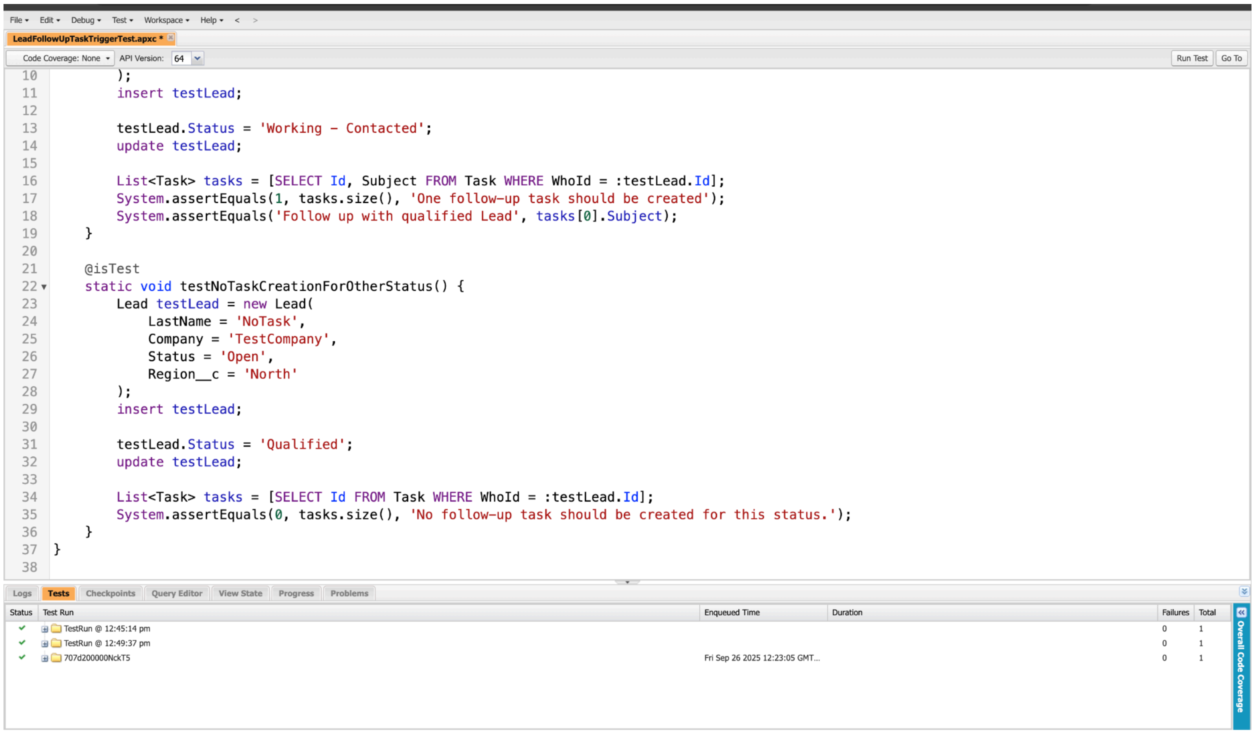
- Used Apex classes indirectly via triggers to organize Lead follow-up logic.
- Lead and Task objects were primary Salesforce objects manipulated.

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

- Implemented an **after update** trigger on the Lead object.
- The trigger automates task creation upon Lead status changes, specifically when status becomes "Working - Contacted".

```
@isTest
public class LeadFollowUpTaskTriggerTest {
  @isTest
  static void testFollowUpTaskCreation() {
    Lead testLead = new Lead(
      LastName = 'Test',
      Company = 'TestCompany',
      Status = 'Open',
      Region__c = 'North'
    );
    insert testLead;

    testLead.Status = 'Working - Contacted';
    update testLead;

    List<Task> tasks = [SELECT Id, Subject FROM Task WHERE WhoId = :testLead.Id];
```

```
        System.assertEquals(1, tasks.size(), 'One follow-up task should be created');
        System.assertEquals('Follow up with qualified Lead', tasks[0].Subject);
    }

    @isTest
    static void testNoTaskCreationForOtherStatus() {
        Lead testLead = new Lead(
            LastName = 'NoTask',
            Company = 'TestCompany',
            Status = 'Open',
            Region__c = 'North'
        );
        insert testLead;

        testLead.Status = 'Qualified';
        update testLead;

        List<Task> tasks = [SELECT Id FROM Task WHERE WhoId = :testLead.Id];
        System.assertEquals(0, tasks.size(), 'No follow-up task should be created for this status.');
    }
}
```



LeadFollowUpTaskTriggerTest.apxc *

Code Coverage: None   API Version: 64

```
        );
        insert testLead;

        testLead.Status = 'Working - Contacted';
        update testLead;

        List<Task> tasks = [SELECT Id, Subject FROM Task WHERE WhoId = :testLead.Id];
        System.assertEquals(1, tasks.size(), 'One follow-up task should be created');
        System.assertEquals('Follow up with qualified Lead', tasks[0].Subject);
    }

    @isTest
    static void testNoTaskCreationForOtherStatus() {
        Lead testLead = new Lead(
            LastName = 'NoTask',
            Company = 'TestCompany',
            Status = 'Open',
            Region__c = 'North'
        );
        insert testLead;

        testLead.Status = 'Qualified';
        update testLead;

        List<Task> tasks = [SELECT Id FROM Task WHERE WhoId = :testLead.Id];
        System.assertEquals(0, tasks.size(), 'No follow-up task should be created for this status.');
    }
}
```

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 12:45:14 pm			0	1
✓	TestRun @ 12:49:37 pm			0	1
✓	707d200000NckT5	Fri Sep 26 2025 12:23:05 GMT...		0	1

3. Trigger Design Pattern

- Followed basic bulkification patterns in the trigger.
- Used collections and guard clauses to ensure efficient processing of multiple records.
- Scope for advanced design patterns like handler classes exists but not implemented.

4. SOQL & SOSL

- Utilized SOQL queries in the test classes to validate task creation.
- SOSL was not needed as there was no requirement for multi-object searching.

5. Collections: List, Set, Map

- Used Lists to batch task records for bulk insertion.
- Used Map for old versus new Lead record comparison in the trigger to detect changes.

6. Control Statements

- Employed if-else, for-loops, and conditional checks to execute logic selectively.

7. Batch Apex

- **Not used** because the project’s data volume and complexity did not require asynchronous batch processing.
- The bulkified trigger handles the expected record volumes efficiently within limits.

8. Queueable Apex

- **Not used** due to absence of complex operations needing asynchronous queued jobs.

9. Scheduled Apex

- **Not used** as there are no requirements for periodic or scheduled automated jobs in this phase.

10. Future Methods

- **Not implemented** since no asynchronous or future processing was demanded.

11. Exception Handling

- Used try-catch blocks to handle exceptions during task insertions inside triggers.
- Ensures graceful failure without affecting overall transactions.

```
if (!tasksToCreate.isEmpty()) {
    try {
      insert tasksToCreate;
    } catch (Exception e) {
      System.debug('Error inserting tasks: ' + e.getMessage());
    }
  }
```

12. Test Classes

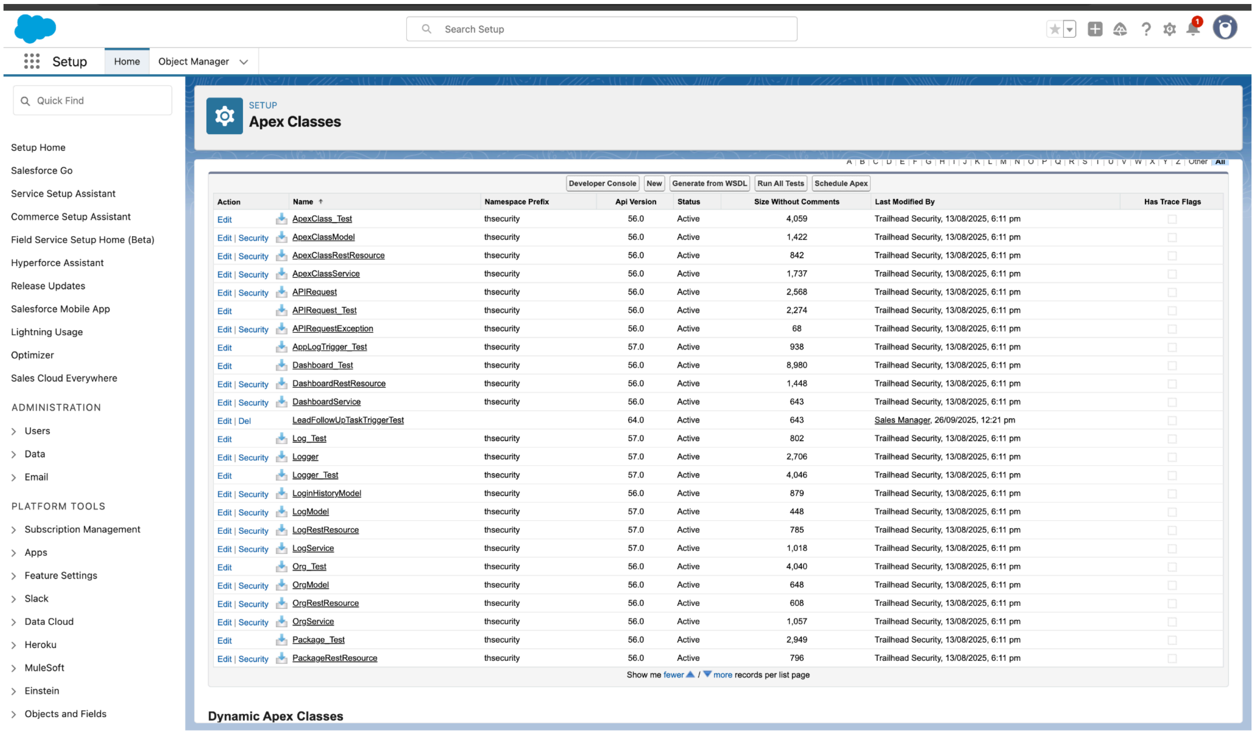
- Developed comprehensive test classes covering:
  - Task creation when Lead status changes to "Working - Contacted".
  - Prevention of task creation for irrelevant status changes.
  - Bulk testing scenario with multiple Lead updates.

13. Asynchronous Processing

- Deferred for future enhancements as current synchronous process suffices.

Conclusion

In this phase, robust automation functionality was developed using Apex triggers and test classes, ensuring automatic and reliable task creation upon key Lead status changes. Exception handling was incorporated to improve resilience, while comprehensive test coverage validated all critical scenarios, meeting Salesforce deployment standards. Non-essential advanced features such as Batch Apex, Queueable Apex, Scheduled Apex, and Future Methods were consciously excluded, as the project scope and data volume did not necessitate these asynchronous or scheduled processing techniques. Overall, the Apex programming in this phase significantly streamlines sales follow-up processes, increases efficiency, and lays a scalable foundation for future enhancements in the system.



Apex Classes

Action	Name ↑	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit	ApexClass_Test	thsecurity	56.0	Active	4,059	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	ApexClassModel	thsecurity	56.0	Active	1,422	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	ApexClassRestResource	thsecurity	56.0	Active	842	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	ApexClassService	thsecurity	56.0	Active	1,737	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	APIRequest	thsecurity	56.0	Active	2,568	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	APIRequest_Test	thsecurity	56.0	Active	2,274	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	APIRequestException	thsecurity	56.0	Active	68	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	AppLogTrigger_Test	thsecurity	57.0	Active	938	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	Dashboard_Test	thsecurity	56.0	Active	8,980	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	DashboardRestResource	thsecurity	56.0	Active	1,448	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	DashboardService	thsecurity	56.0	Active	643	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Del	LeadFollowUpTaskTriggerTest		64.0	Active	643	Sales Manager, 26/09/2025, 12:21 pm	□
Edit	Log_Test	thsecurity	57.0	Active	802	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	Logger	thsecurity	57.0	Active	2,706	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	Logger_Test	thsecurity	57.0	Active	4,046	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	LoginHistoryModel	thsecurity	56.0	Active	879	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	LogModel	thsecurity	57.0	Active	448	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	LogRestResource	thsecurity	57.0	Active	785	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	LogService	thsecurity	57.0	Active	1,018	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	Org_Test	thsecurity	56.0	Active	4,040	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	OrgModel	thsecurity	56.0	Active	648	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	OrgRestResource	thsecurity	56.0	Active	608	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	OrgService	thsecurity	56.0	Active	1,057	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit	Package_Test	thsecurity	56.0	Active	2,949	Trailhead Security, 13/08/2025, 6:11 pm	□
Edit | Security	PackageRestResource	thsecurity	56.0	Active	796	Trailhead Security, 13/08/2025, 6:11 pm	□

Show me fewer ▲ / ▼ more records per list page

Dynamic Apex Classes