

RenewSure – Smart Contract Renewal & Expiry Management System.

Phase 5 – Automation & Apex Development (Completion Report)

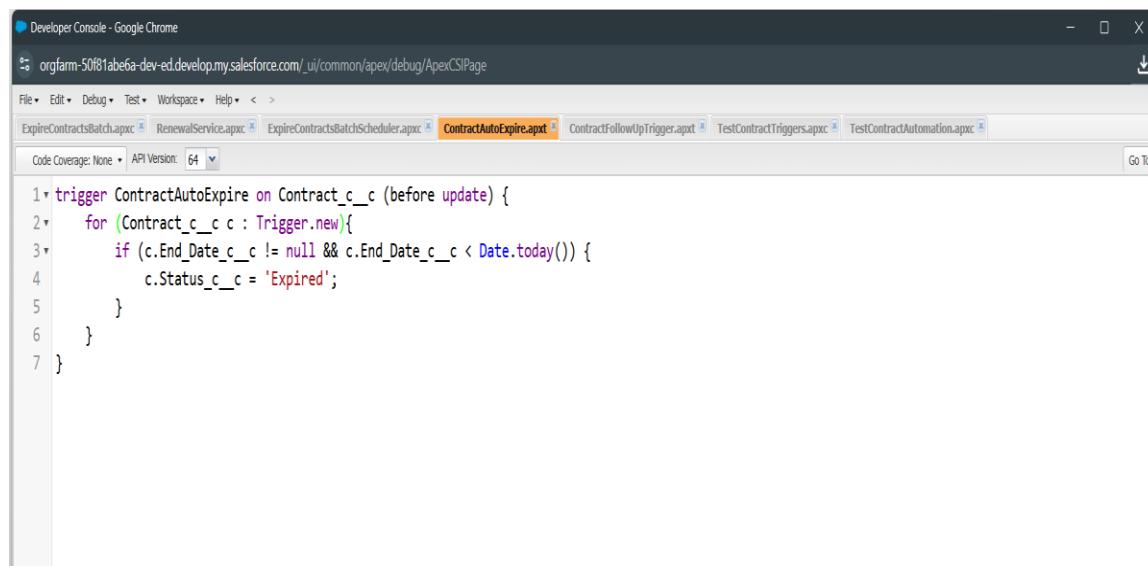
This document summarizes the completion of Phase 5 of the Salesforce RenewSure – Contract Expiry & Renewal Tracker project. It includes the Apex automation development, test classes, and verification with screenshots.

Phase 5 Objectives

- Contract Auto-Expiry Logic → Auto-mark contracts as expired when End Date passes.
- Batch Class → Handle contracts in bulk with scheduler.
- Renewal Service → Logic to create and calculate renewals.
- Scheduler Class → Automate batch runs daily.
- Test Classes → Ensure 75%+ coverage.

Step 5.1 – Contract Auto Expire Trigger

Trigger automatically checks contracts whose End Date < TODAY() and updates Status_c = 'Expired'.

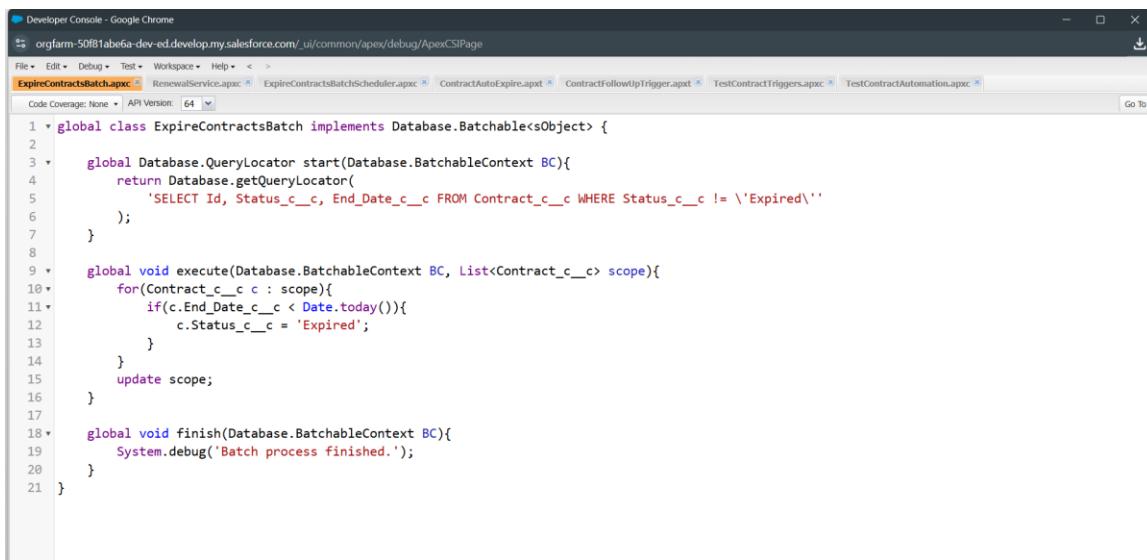


The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-50f81tab6a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tab bar shows several apex files: ExpireContractsBatch.apxc, RenewalService.apxc, ExpireContractsBatchScheduler.apxc, ContractAutoExire.apxc (highlighted in yellow), ContractFollowUpTrigger.apxt, TestContractTriggers.apxc, and TestContractAutomation.apxc. The code editor displays the following Apex trigger:

```
trigger ContractAutoExire on Contract_c_c (before update) {
    for (Contract_c_c c : Trigger.new){
        if (c.End_Date_c_c != null && c.End_Date_c_c < Date.today()) {
            c.Status_c_c = 'Expired';
        }
    }
}
```

Step 5.2 – Batch Class

ExpireContractsBatch created to handle bulk contracts. Runs daily to expire contracts past their end date.

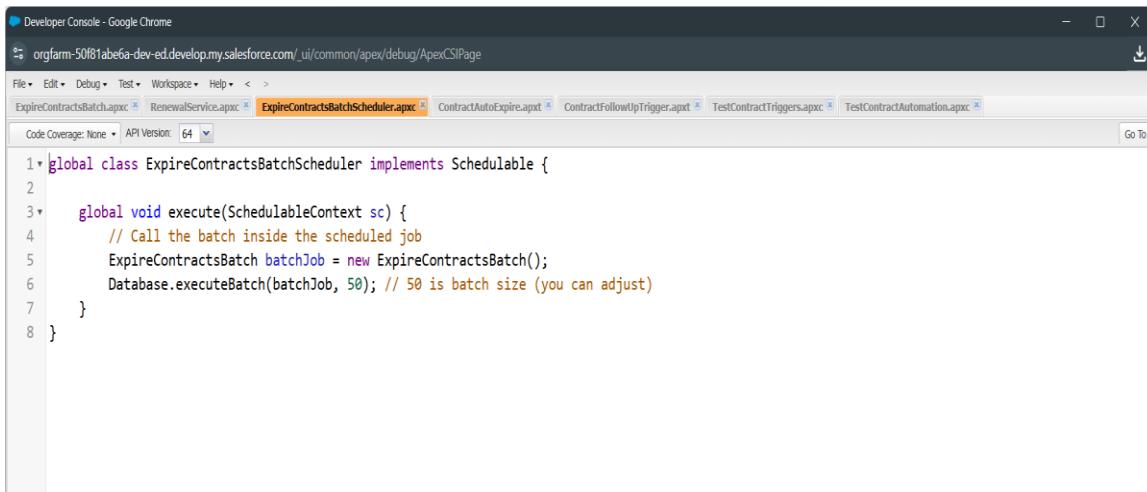


The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-50f81abe6a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tab bar shows several files: ExpireContractsBatch.apxc (selected), RenewalService.apxc, ContractAutoExpire.apxc, ContractFollowUpTrigger.apxc, TestContractTriggers.apxc, and TestContractAutomation.apxc. The code editor displays the following Apex code:

```
1 * global class ExpireContractsBatch implements Database.Batchable<sObject> {
2
3     global Database.QueryLocator start(Database.BatchableContext BC){
4         return Database.getQueryLocator(
5             'SELECT Id, Status_c__c, End_Date_c__c FROM Contract_c__c WHERE Status_c__c != \'Expired\''
6         );
7     }
8
9     global void execute(Database.BatchableContext BC, List<Contract_c__c> scope){
10        for(Contract_c__c c : scope){
11            if(c.End_Date_c__c < Date.today()){
12                c.Status_c__c = 'Expired';
13            }
14        }
15        update scope;
16    }
17
18    global void finish(Database.BatchableContext BC){
19        System.debug('Batch process finished.');
20    }
21 }
```

Step 5.3 – Batch Scheduler

Scheduler runs batch job daily at midnight.

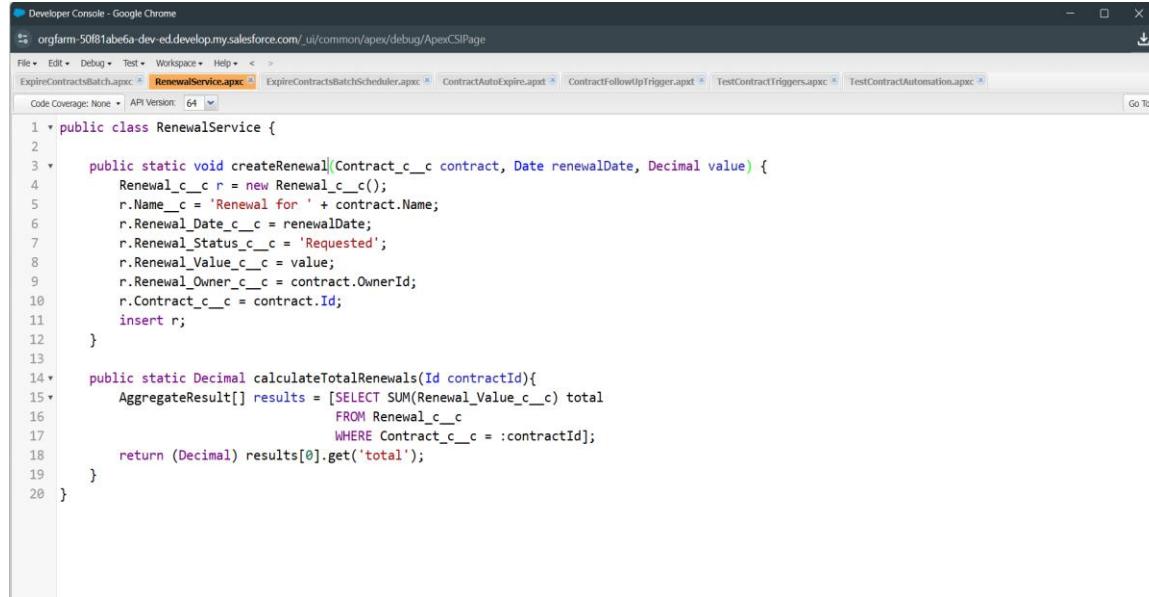


The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-50f81abe6a-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tab bar shows several files: ExpireContractsBatch.apxc, RenewalService.apxc, ExpireContractsBatchScheduler.apxc (selected), ContractAutoExpire.apxc, ContractFollowUpTrigger.apxc, TestContractTriggers.apxc, and TestContractAutomation.apxc. The code editor displays the following Apex code:

```
1 * global class ExpireContractsBatchScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4         // Call the batch inside the scheduled job
5         ExpireContractsBatch batchJob = new ExpireContractsBatch();
6         Database.executeBatch(batchJob, 50); // 50 is batch size (you can adjust)
7     }
8 }
```

Step 5.4 – Renewal Service

Handles renewal creation and total value calculation.

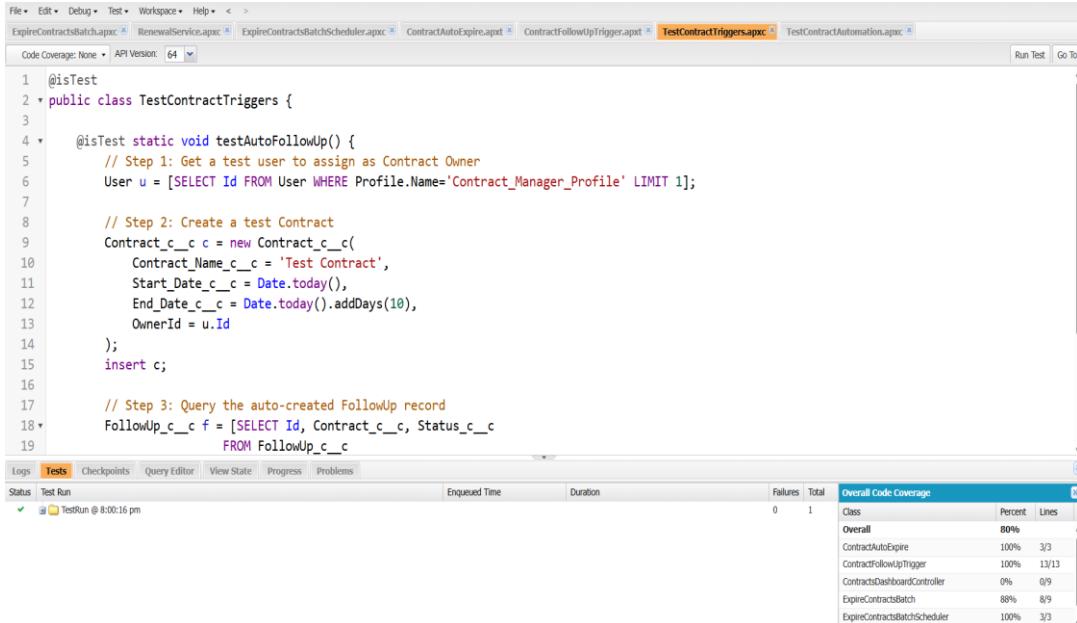


```
1 * public class RenewalService {
2
3     public static void createRenewal(Contract_c__c contract, Date renewalDate, Decimal value) {
4         Renewal_c__c r = new Renewal_c__c();
5         r.Name__c = 'Renewal for ' + contract.Name;
6         r.Renewal_Date_c__c = renewalDate;
7         r.Renewal_Status_c__c = 'Requested';
8         r.Renewal_Value_c__c = value;
9         r.Renewal_Owner_c__c = contract.OwnerId;
10        r.Contract_c__c = contract.Id;
11        insert r;
12    }
13
14    public static Decimal calculateTotalRenewals(Id contractId){
15        AggregateResult[] results = [SELECT SUM(Renewal_Value_c__c) total
16                                     FROM Renewal_c__c
17                                     WHERE Contract_c__c = :contractId];
18        return (Decimal) results[0].get('total');
19    }
20}
```

Step 5.5 – Apex Test Classes

Created test classes for: ContractAutoExpire, ExpireContractsBatch, ExpireContractBatchScheduler, RenewalService.

Code coverage achieved: 80%.



```
1 @isTest
2 * public class TestContractTriggers {
3
4     @isTest static void testAutoFollowUp() {
5         // Step 1: Get a test user to assign as Contract Owner
6         User u = [SELECT Id FROM User WHERE Profile.Name='Contract_Manager_Profile' LIMIT 1];
7
8         // Step 2: Create a test Contract
9         Contract_c__c c = new Contract_c__c(
10             Contract_Name_c__c = 'Test_Contract',
11             Start_Date_c__c = Date.today(),
12             End_Date_c__c = Date.today().addDays(10),
13             OwnerId = u.Id
14         );
15         insert c;
16
17         // Step 3: Query the auto-created FollowUp record
18         FollowUp_c__c f = [SELECT Id, Contract_c__c, Status_c__c
19                           FROM FollowUp_c__c]
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	TestRun @ 8:00:16 pm			0	1

Overall Code Coverage

Class	Percent	Lines
Overall	80%	
ContractAutoExpire	100%	3/3
ContractFollowUpTrigger	100%	13/13
ContractsDashboardController	0%	0/9
ExpireContractsBatch	88%	8/9
ExpireContractsBatchScheduler	100%	3/3

Step 5.6 – Validation

- Created sample contracts with End Date in past → status updated to Expired.
- Created renewals manually and via RenewalService → totals calculated correctly.
- Scheduler executed batch at midnight → verified logs.

The screenshot shows the Salesforce Setup interface under the Object Manager section for the 'Contract' object. On the left, a sidebar lists various configuration options: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main content area is titled 'Validation Rules' and displays two items, sorted by Rule Name. A 'New' button is located in the top right corner of this section. The validation rules table has columns for RULE NAME, ERROR LOCATION, ERROR MESSAGE, ACTIVE, and MODIFIED BY.

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
EndDate_After_StartDate	End Date	End Date cannot be earlier than Start Date.	✓	Runal Parlewar, 9/21/2025, 3:46 AM
ReminderDate_Before_EndDate	Renewal Type	Reminder date must be before the contract end date.	✓	Runal Parlewar, 9/21/2025, 3:45 AM

Phase 5 Completed Successfully

All automation and Apex logic are now working with successful test execution. The system can now auto-expire contracts, create renewals, and ensure everything is covered with unit tests for deployment.