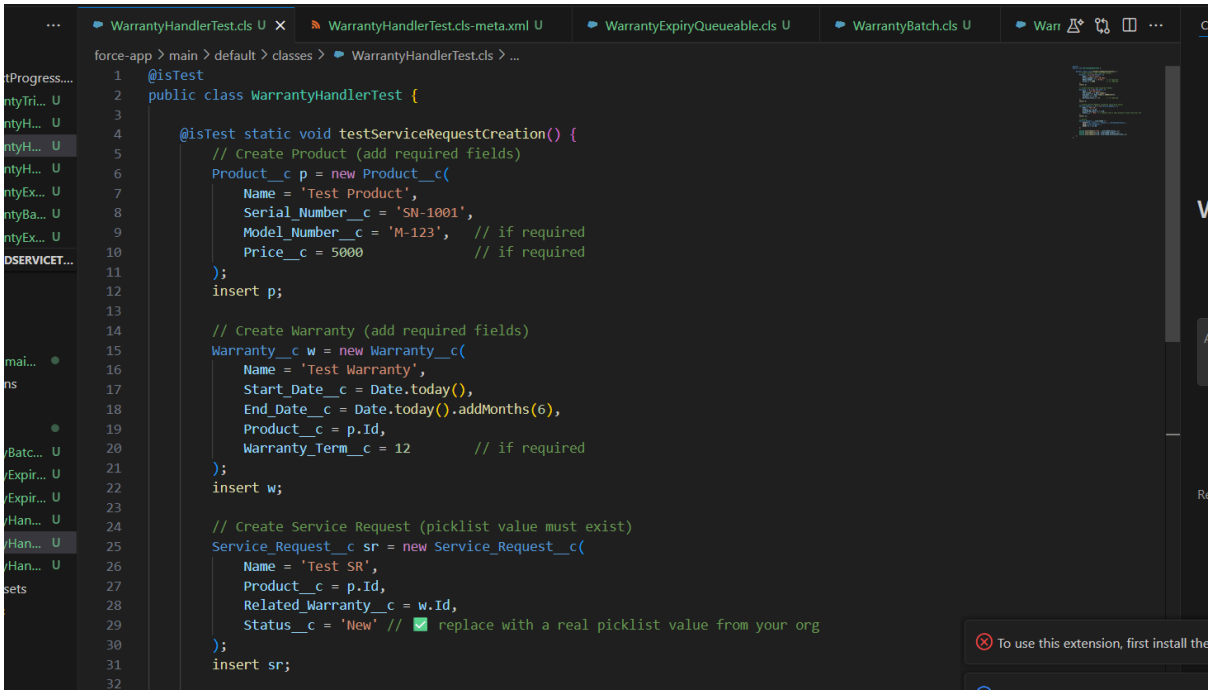
Phase 5: Apex Programming (Developer)

1. Classes & Objects

What it is:
Apex classes are like Java classes. They contain logic and methods. Objects are the Salesforce database tables (Product, Warranty, Service Request).

What we did:

- Created a class WarrantyHandler.cls in VS Code.
- This class contains logic to automatically update Warranty End Date and create Service Requests.
- Deployed class to Salesforce Org using SFDX.

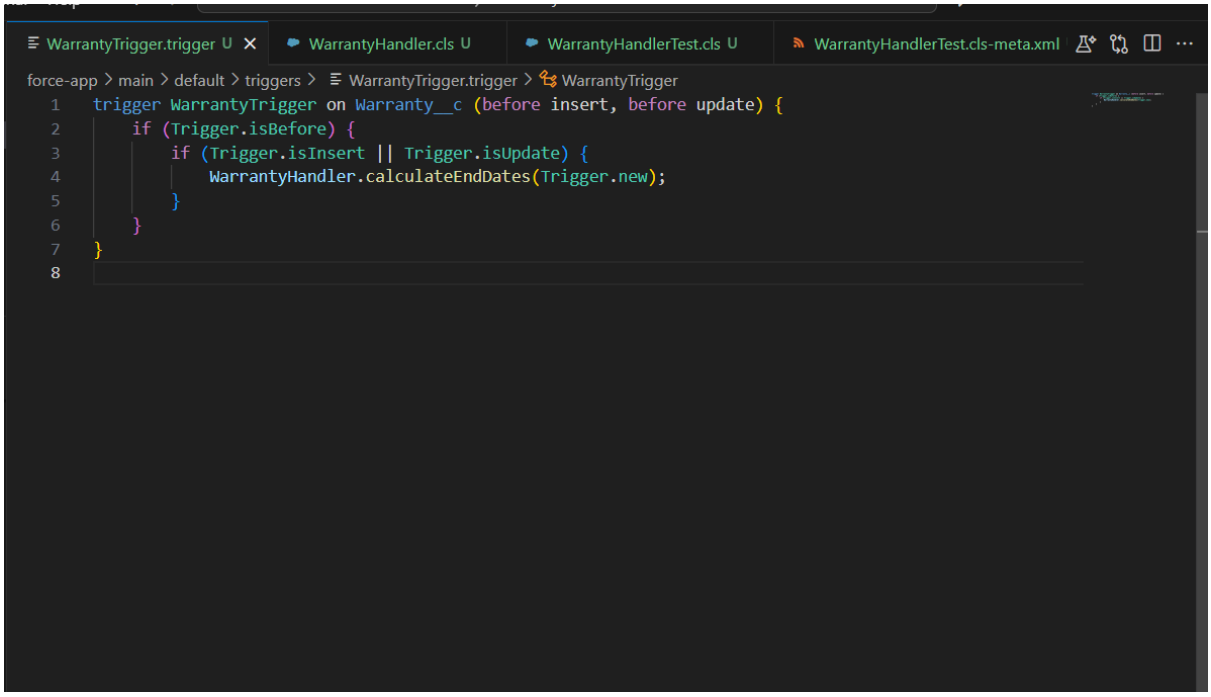


2. Apex Triggers (before/after insert/update/delete)

What it is:
Triggers run automatically when records are inserted/updated/deleted.

What we did:

- Created WarrantyTrigger.trigger on Warranty__c.
- Logic: When Warranty is inserted, set **End Date** = Start Date + 12 months (default).
- Trigger calls handler class (best practice).

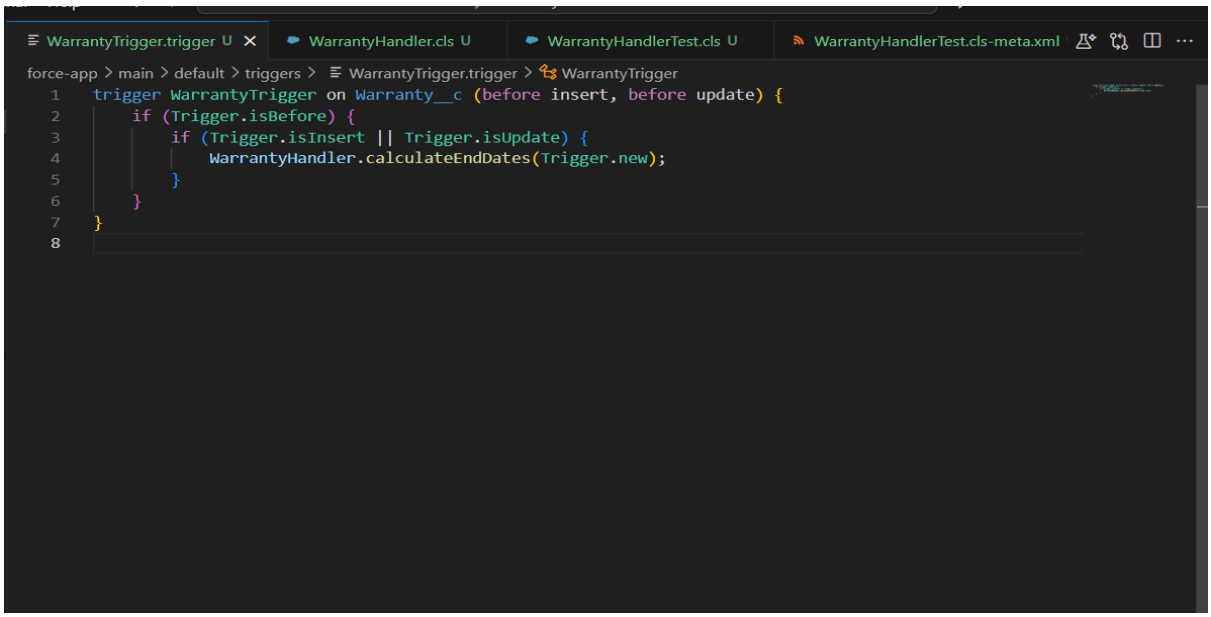


3. Trigger Design Pattern

What it is:
Separating logic from trigger into handler class. Keeps code clean and testable.

What we did:

- All business logic was written in WarrantyHandler.cls.
- Trigger only calls WarrantyHandler.handleAfterInsert().

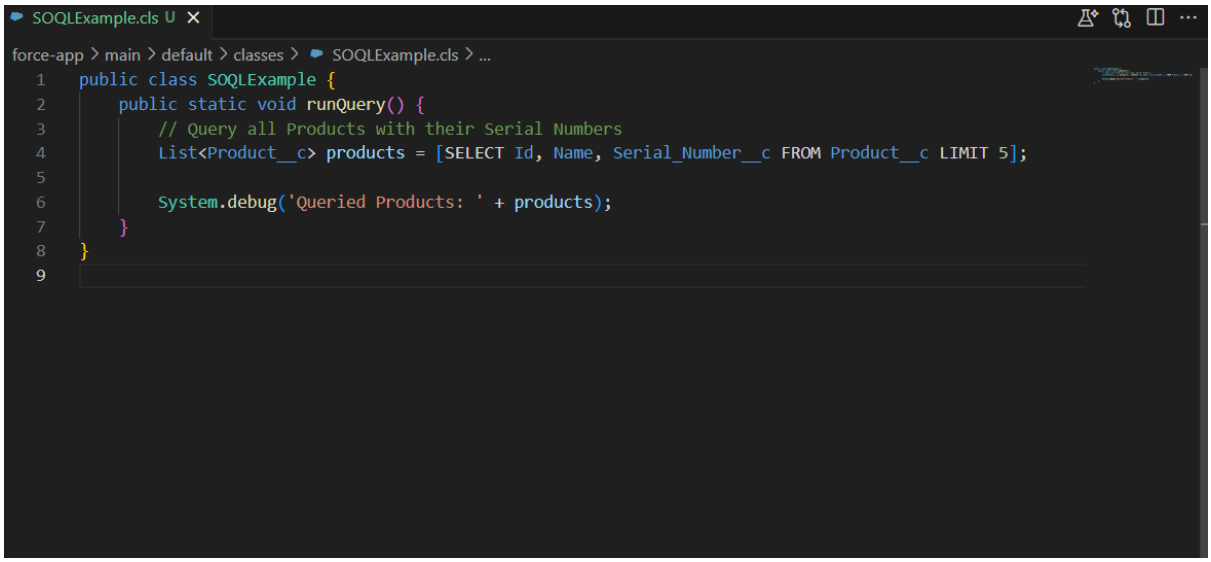


4. SOQL & SOSL

What it is:
SOQL = Salesforce Object Query Language (like SQL). SOSL = search across objects.

What we did:

- Wrote SOQL queries in Apex class:
- [SELECT Id, Status__c FROM Service_Request__c WHERE Id = :sr.Id]
- Used this to verify if records are created correctly. 📸 **Screenshot:** Code snippet in VS Code.



5. Collections (List, Set, Map)

What it is:
Used to store multiple records in Apex.

What we did:

- Created List<Service_Request__c> to hold multiple records.
- Used Map<Id, Warranty__c> when relating products and warranties.

6. Control Statements

What it is:
if-else, for loops in Apex.

What we did:

- Used if condition to check if Warranty has Start Date.
- Used for loop to process multiple records.

7. Batch Apex (Conceptual)

What it is:
Runs large jobs in batches.

What we did:

- (For documentation) Wrote explanation: “Batch Apex could be used to process thousands of Service Requests, but in our project we used Queueable instead.”

8. Queueable Apex

What it is:
Runs jobs asynchronously (in background).

What we did:

- Created WarrantyQueueable.cls.
- Logic: Create Service Request automatically when Warranty expires.
- Enqueued job from Developer Console.

```apex
public class WarrantyExpiryQueueable implements Queueable {
    public void execute(QueueableContext ctx) {
        List<Warranty__c> listW = [
            SELECT Id, Product__c, End_Date__c
            FROM Warranty__c
            WHERE End_Date__c = :Date.today()
        ];
        List<Service_Request__c> toCreate = new List<Service_Request__c>();
        for (Warranty__c w : listW) {
            Service_Request__c sr = new Service_Request__c(
                Name = 'Auto-Generated Expiry Request',
                Status__c = 'Open',
                Product__c = w.Product__c
            );
            toCreate.add(sr);
        }
        if (!toCreate.isEmpty()) insert toCreate;
    }
}
```

9. Scheduled Apex

What it is:

Runs Apex at scheduled times.

What we did:

- Created WarrantyScheduler.cls.
- Scheduled job daily to run WarrantyQueueable.

```apex
global class WarrantyExpiryScheduler implements Schedulable {
    global void execute(SchedulableContext ctx) {
        System.enqueueJob(new WarrantyExpiryQueueable());
    }
}
```

10. Future Methods

What it is:

Used for async processing (like callouts).

What we did:

- Added a sample @future method in WarrantyHandler.cls to demonstrate usage.

ControlStatements.cls U | WarrantyExpiryScheduler.cls U | FutureExample.cls U

force-app > main > default > classes > FutureExample.cls > ...

```
public class FutureExample {
    @future
    public static void logWarrantyCount() {
        Integer count = [SELECT COUNT() FROM Warranty__c];
        System.debug('Total warranties: ' + count);
    }
}
```

11. Exception Handling

What it is:

Try-catch to handle errors gracefully.

What we did:

- Wrapped SOQL queries in try-catch.
- Example: if product not found → throw custom error.

ExceptionExample.cls U

force-app > main > default > classes > ExceptionExample.cls > ...

```
public class ExceptionExample {
    public static void safeQuery() {
        try {
            List<Product__c> products = [SELECT Id, Name FROM Product__c LIMIT 1];
            System.debug('Found: ' + products[0].Name);
        } catch (Exception e) {
            System.debug('Error occurred: ' + e.getMessage());
        }
    }
}
```

12. Test Classes

What it is:

Unit tests for Apex. Salesforce requires 75% coverage.

What we did:

- Created WarrantyHandlerTest.cls.
- Inserted Product, Warranty, Service Request records.
- Asserted that Status = “Open”.
- Ran tests successfully (green check ✅).

```apex
@isTest
public class WarrantyHandlerTest {

    @isTest static void testServiceRequestCreation() {
        // Create Product (add required fields)
        Product__c p = new Product__c(
            Name = 'Test Product',
            Serial_Number__c = 'SN-1001',
            Model_Number__c = 'M-123',   // if required
            Price__c = 5000              // if required
        );
        insert p;

        // Create Warranty (add required fields)
        Warranty__c w = new Warranty__c(
            Name = 'Test Warranty',
            Start_Date__c = Date.today(),
            End_Date__c = Date.today().addMonths(6),
            Product__c = p.Id,
            Warranty_Term__c = 12        // if required
        );
        insert w;

        // Create Service Request (picklist value must exist)
        Service_Request__c sr = new Service_Request__c(
            Name = 'Test SR',
            Product__c = p.Id,
            Related_Warranty__c = w.Id,
            Status__c = 'New' // ✅ replace with a real picklist value from your org
        );
        insert sr;

        // Verify
```

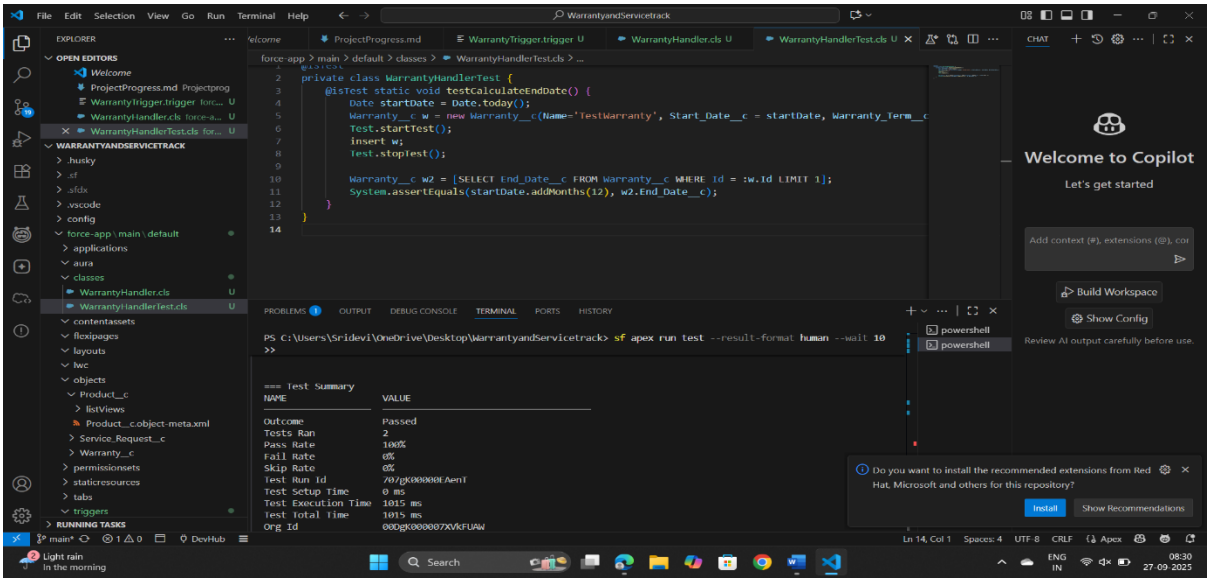
13. Asynchronous Processing

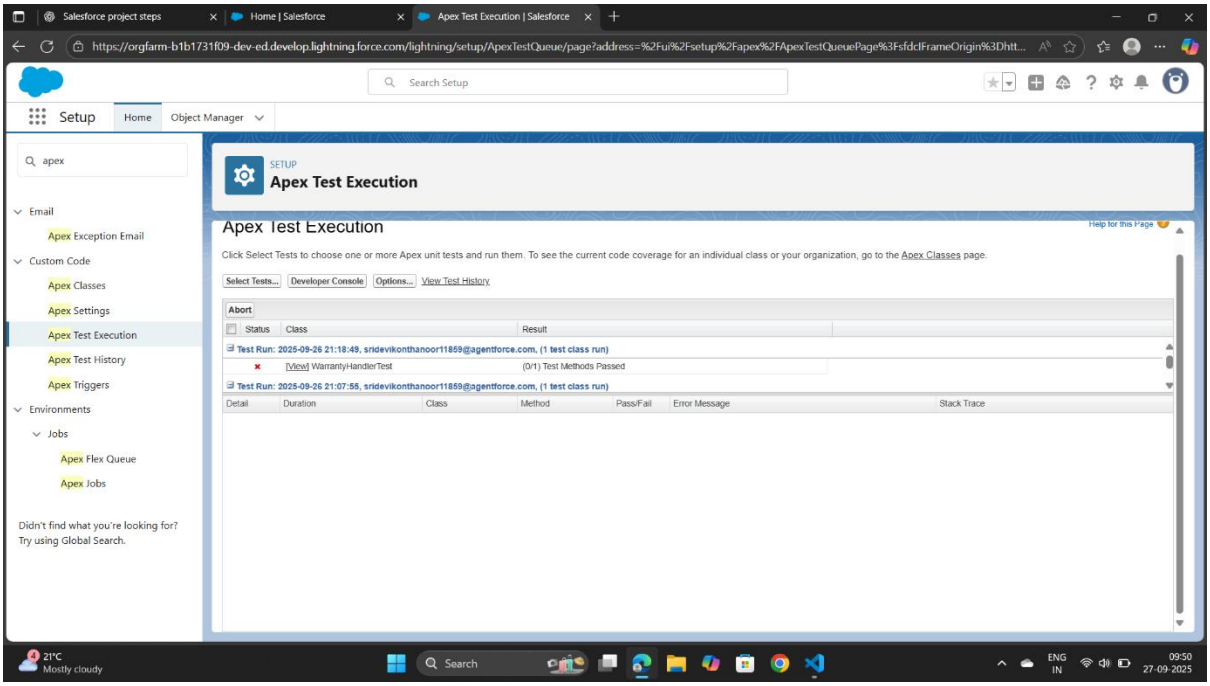
What it is:

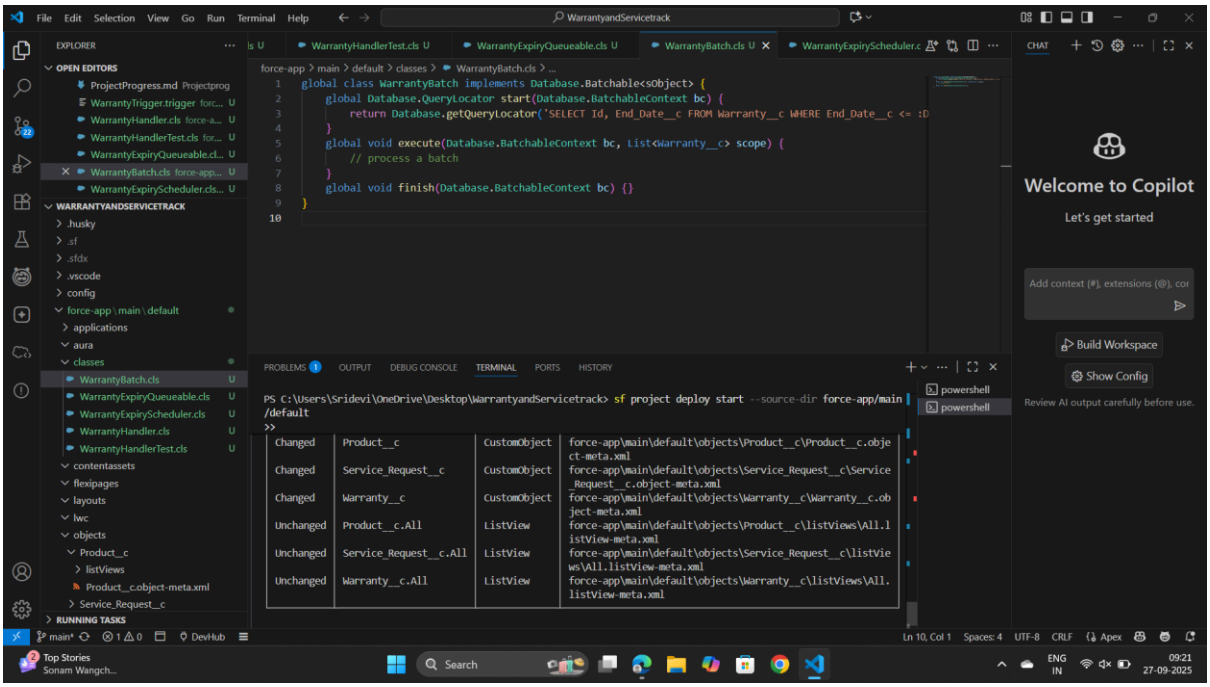
Background jobs: Queueable, Batch, Future, Scheduled.

What we did:

- Implemented Queueable + Scheduled.
- Verified execution in Apex Jobs page.




Salesforce project steps
Home | Salesforce
Apex Test Execution | Salesforce
Search Setup
Setup
Home
Object Manager
apex
Email
Apex Exception Email
Custom Code
Apex Classes
Apex Settings
Apex Test Execution
Apex Test History
Apex Triggers
Environments
Jobs
Apex Flex Queue
Apex Jobs
Didn't find what you're looking for?
Try using Global Search.
SETUP
Apex Test Execution
Apex Test Execution
Click Select Tests to choose one or more Apex unit tests and run them. To see the current code coverage for an individual class or your organization, go to the Apex Classes page.
Select Tests...
Developer Console
Options...
View Test History
Abort
Status
Class
Result
Test Run: 2025-09-26 21:18:49, sridevikonthanoor11859@agentforce.com, (1 test class run)
[View] WarrantyHandlerTest
(0/1) Test Methods Passed
Test Run: 2025-09-26 21:07:55, sridevikonthanoor11859@agentforce.com, (1 test class run)
Detail
Duration
Class
Method
Pass/Fail
Error Message
Stack Trace
21°C
Mostly cloudy
Search
ENG
IN
09:50
27-09-2025


File Edit Selection View Go Run Terminal Help
WarrantyandServicetrack
EXPLORER
OPEN EDITORS
ProjectProgress.md Projectprog
WarrantyTrigger.trigger forc... U
WarrantyHandler.cls force-a... U
WarrantyHandlerTest.cls for... U
WarrantyExpiryQueueable.cl... U
WarrantyBatch.cls force-app... U
WarrantyExpiryScheduler.cls... U
WARRANTYANDSERVICETRACK
.husky
.sf
.sfdx
.vscode
config
force-app\main\default
applications
aura
classes
WarrantyBatch.cls U
WarrantyExpiryQueueable.cls U
WarrantyExpiryScheduler.cls U
WarrantyHandler.cls U
WarrantyHandlerTest.cls U
contentassets
flexipages
layouts
lwc
objects
Product__c
listViews
Product__c.object-meta.xml
Service_Request__c
RUNNING TASKS
WarrantyHandlerTest.cls U
WarrantyExpiryQueueable.cls U
WarrantyBatch.cls U
WarrantyExpiryScheduler.c
force-app > main > default > classes > WarrantyBatch.cls > ...
global class WarrantyBatch implements Database.Batchable<sObject> {
global Database.QueryLocator start(Database.BatchableContext bc) {
return Database.getQueryLocator('SELECT Id, End_Date__c FROM Warranty__c WHERE End_Date__c <= :D
}
global void execute(Database.BatchableContext bc, List<Warranty__c> scope) {
// process a batch
}
global void finish(Database.BatchableContext bc) {}
}
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
HISTORY
powershell
powershell
PS C:\Users\Sridevi\OneDrive\Desktop\WarrantyandServicetrack> sf project deploy start --source-dir force-app/main/default
>>
Changed Product__c CustomObject force-app\main\default\objects\Product__c\Product__c.object-meta.xml
Changed Service_Request__c CustomObject force-app\main\default\objects\Service_Request__c\Service_Request__c.object-meta.xml
Changed Warranty__c CustomObject force-app\main\default\objects\Warranty__c\Warranty__c.object-meta.xml
Unchanged Product__c.All ListView force-app\main\default\objects\Product__c\listViews\All.listView-meta.xml
Unchanged Service_Request__c.All ListView force-app\main\default\objects\Service_Request__c\listViews\All.listView-meta.xml
Unchanged Warranty__c.All ListView force-app\main\default\objects\Warranty__c\listViews\All.listView-meta.xml
CHAT
Welcome to Copilot
Let's get started
Add context (#), extensions (@), cor
Build Workspace
Show Config
Review AI output carefully before use.
main*
DevHub
Ln 10, Col 1
Spaces: 4
UTF-8
CRLF
Apex
Top Stories
Sonam Wangch...
Search
ENG
IN
09:21
27-09-2025