

V-Electronics: Revolutionizing Smart Device Management

Phase 5: Developer Automation

While declarative automation tools covered a majority of processes in Phase 4, there were still advanced business requirements in the **V-Electronics Project** that required custom logic. Salesforce provides **Apex**, a strongly typed, object-oriented programming language similar to Java, to extend platform functionality.

This phase focused on developing custom Apex code to handle complex logic, bulk data operations, asynchronous processes, and advanced system integrations.

1. Introduction

V-Electronics required features such as:

- Updating thousands of records at once (bulk order processing).
- Scheduled data updates and background jobs.
- Integrations with external systems.
- Maintaining modular, reusable code for scalability.

Apex development allowed the project to go beyond the limits of point-and-click tools, ensuring full customization to meet business needs.

2. Objectives of this Phase

- To design and implement Apex classes for reusable and modular business logic.
 - To create Apex triggers following best practices (handler pattern).
 - To perform efficient data queries using SOQL and SOSL.
 - To handle large datasets using collections.
 - To implement asynchronous processes such as Batch Apex, Queueable Apex, and Scheduled Apex.
 - To ensure robust exception handling.
 - To achieve high code coverage with Apex test classes.
-

3. Detailed Description of Contents with Examples

Classes & Objects

- Created reusable classes to encapsulate logic.
- Example: `OrderUtility.cls` with a method `calculateOrderTotal(List<OrderItem> items)` that calculates total value of an order.
- Example: `CustomerService.cls` with a method `updateLoyaltyPoints(Id customerId, Decimal orderValue)` that increases loyalty points after a successful order.




```

1 public static void validateOrderQuantity(List<VOrder__c> orderList) {
2
3     for (VOrder__c order : orderList) {
4
5         if (order.Order_Status__c == 'Confirmed') {
6
7             if (order.Quantity__c == null || order.Quantity__c <= 500) {
8
9                 order.Quantity__c.addError('For Status "Confirmed", Quantity must be more than 500.');

```

Apex Triggers

- Implemented before/after triggers for business rules.
- Example: **Before Insert on VOrders** → Auto-populate Order Total field using `OrderUtility`.
- Example: **After Update on VCustomers** → If loyalty points cross 1000, automatically update the loyalty tier field.



```

1 global class InventoryBatchJob implements Database.Batchable<SObject>, Schedulable {
2
3     global Database.QueryLocator start(Database.BatchableContext BC) {
4
5         return Database.getQueryLocator(
6
7             'SELECT Id, Stock_Quantity__c FROM Product__c WHERE Stock_Quantity__c < 10'
8
9         );
10
11     }
12
13     global void execute(Database.BatchableContext BC, List<SObject> records) {
14
15         List<VProduct__c> productsToUpdate = new List<VProduct__c>();
16
17         // Cast SObject list to Product__c list
18
19         for (SObject record : records) {
20
21             VProduct__c product = (VProduct__c) record;
22
23         }
24
25     }
26 }

```

Trigger Design Pattern

- Used handler classes for trigger logic.
- Example:

- **Trigger:** OrderTrigger → only calls OrderTriggerHandler.beforeInsert() method.
 - **Handler:** OrderTriggerHandler → contains reusable methods for insert, update, delete operations.
-

SOQL & SOSL

- **Example SOQL:**
 - `List<VOrder__c> recentOrders = [SELECT Id, Total_Amount__c FROM VOrder__c WHERE CreatedDate = LAST_N_DAYS:30];`
 - **Example SOSL:**
 - `List<List<SObject>> searchResults = [FIND 'john*' IN ALL FIELDS RETURNING VCustomer__c(Id, Name, Email__c)];`
-

Collections (List, Set, Map)

- **List Example:**
`List<VProduct__c> products = [SELECT Id, Name FROM VProduct__c LIMIT 10];`
 - **Set Example:**
`Set<String> emails = new Set<String>{'a@test.com', 'b@test.com', 'a@test.com'}; // removes duplicates`
 - **Map Example:**
`Map<Id, VCustomer__c> custMap = new Map<Id, VCustomer__c>([SELECT Id, Name FROM VCustomer__c]);`
-

Control Statements

- **Example:**
 - ```
for (VOrder__c order : orders) {
 if (order.Total_Amount__c > 50000) {
 order.VIP__c = true;
 }
}
```
- 

## Batch Apex

- **Example:** Recalculate loyalty points for all customers monthly.
- ```
global class LoyaltyBatch implements Database.Batchable<SObject> {  
    global Database.QueryLocator start(Database.BatchableContext bc)  
    {  
        return Database.getQueryLocator('SELECT Id, Total_Spent__c FROM VCustomer__c');  
    }  
}
```

- `global void execute(Database.BatchableContext bc, List<VCustomer__c> scope) {`
 - `for (VCustomer__c c : scope) {`
 - `c.Loyalty_Points__c = c.Total_Spent__c / 10;`
 - `}`
 - `update scope;`
 - `}`
 - `global void finish(Database.BatchableContext bc) {}`
 - `}`
-

Queueable Apex

- Example: Push customer data to ERP system.
 - `public class ERPIntegrationQueue implements Queueable {`
 - `public void execute(QueueableContext context) {`
 - `// Callout logic here`
 - `}`
 - `}`
-

Scheduled Apex

- Example: Run every night to refresh product inventory.
 - `global class InventoryScheduler implements Schedulable {`
 - `global void execute(SchedulableContext sc) {`
 - `// call InventoryBatch`
 - `}`
 - `}`
-

Future Methods

- Example: Send async callout for order confirmation.
 - `@future(callout=true)`
 - `public static void sendOrderConfirmation(Id orderId) {`
 - `// Callout logic`
 - `}`
-

Exception Handling

- Example:
 - `try {`
 - `update orderList;`
 - `} catch (DmlException e) {`
 - `System.debug('Error: ' + e.getMessage());`
 - `}`
-

Test Classes

- Example:
 - `@isTest`
 - `private class OrderUtilityTest {`
 - `@isTest static void testCalculateOrderTotal() {`
 - `VOrder__c order = new VOrder__c(Name='Test Order');`
 - `insert order;`
 - `Test.startTest();`
 - `// logic`
 - `Test.stopTest();`
 - `System.assertEquals(100, order.Total_Amount__c);`
 - `}`
 - `}`
-

Asynchronous Processing

- Example: For bulk order processing, batch recalculates totals while queueable updates ERP system simultaneously.
-

☒ These examples show **realistic use cases** that you can include in your Phase 5 documentation.

4. Deliverables/Outcomes of Phase 5

- Developed modular Apex classes for reusable logic.
 - Built Apex triggers with handler design pattern.
 - Implemented SOQL/SOSL queries and optimized with collections.
 - Built batch, queueable, scheduled, and future methods for asynchronous operations.
 - Established exception handling and error logging.
 - Created test classes with >85% coverage.
-

5. Conclusion

Phase 5 extended Salesforce beyond declarative capabilities, delivering **scalable and advanced custom logic** for V-Electronics. Apex programming ensured the system could handle bulk data, run background jobs, and integrate smoothly with external applications.

With development complete, the project progressed to **Phase 6 (User Interface Development)**, where these backend processes were integrated into a seamless, user-friendly Salesforce UI.

