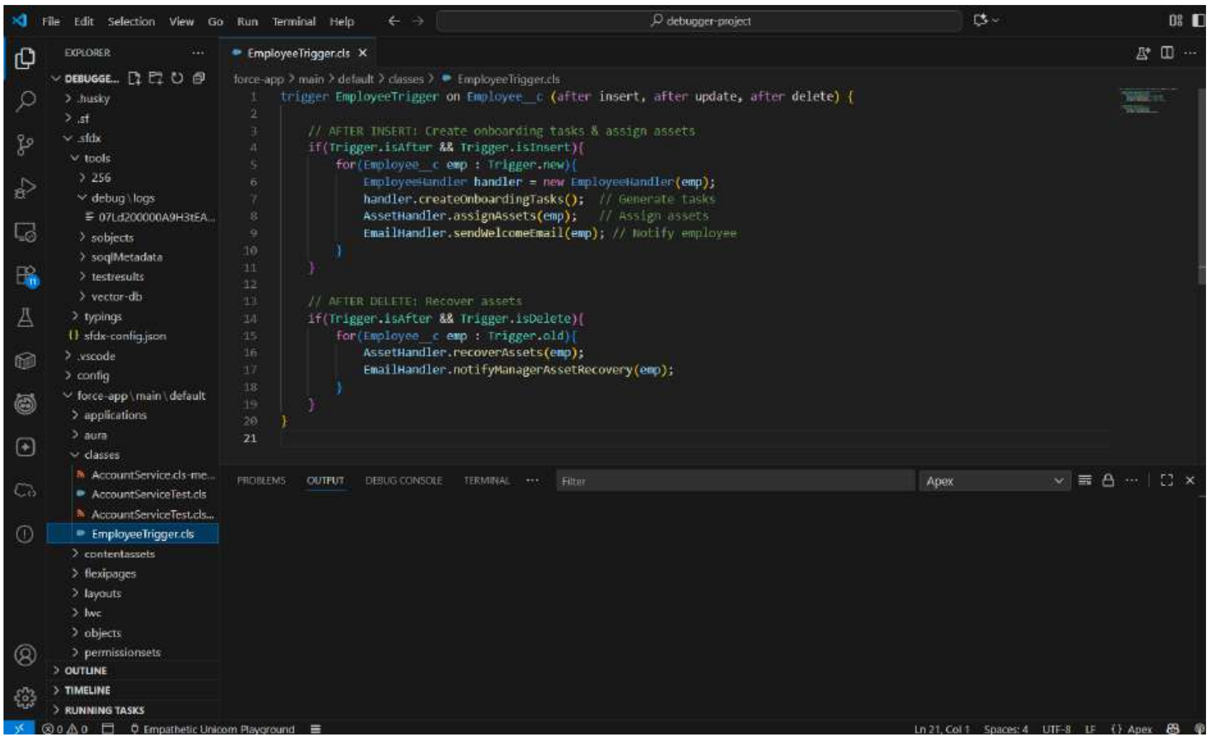
Employee Onboarding & Asset Management CRM Project

Phase 5: Apex Programming (Developer)

---

1. Classes & Objects

- **Employee Class:** Define employee details, onboarding status, and assigned assets.
- **Asset Class:** Include asset details like type, value, assignment status, and return date.
- **Utility Classes:** For email notifications, approval processing, and dashboard updates.


File Edit Selection View Go Run Terminal Help
debugger-project
EXPLORER
EmployeeTrigger.cls
force-app > main > default > classes > EmployeeTrigger.cls
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Apex
Ln 21, Col 1 Spaces: 4 UTF-8 LF {} Apex

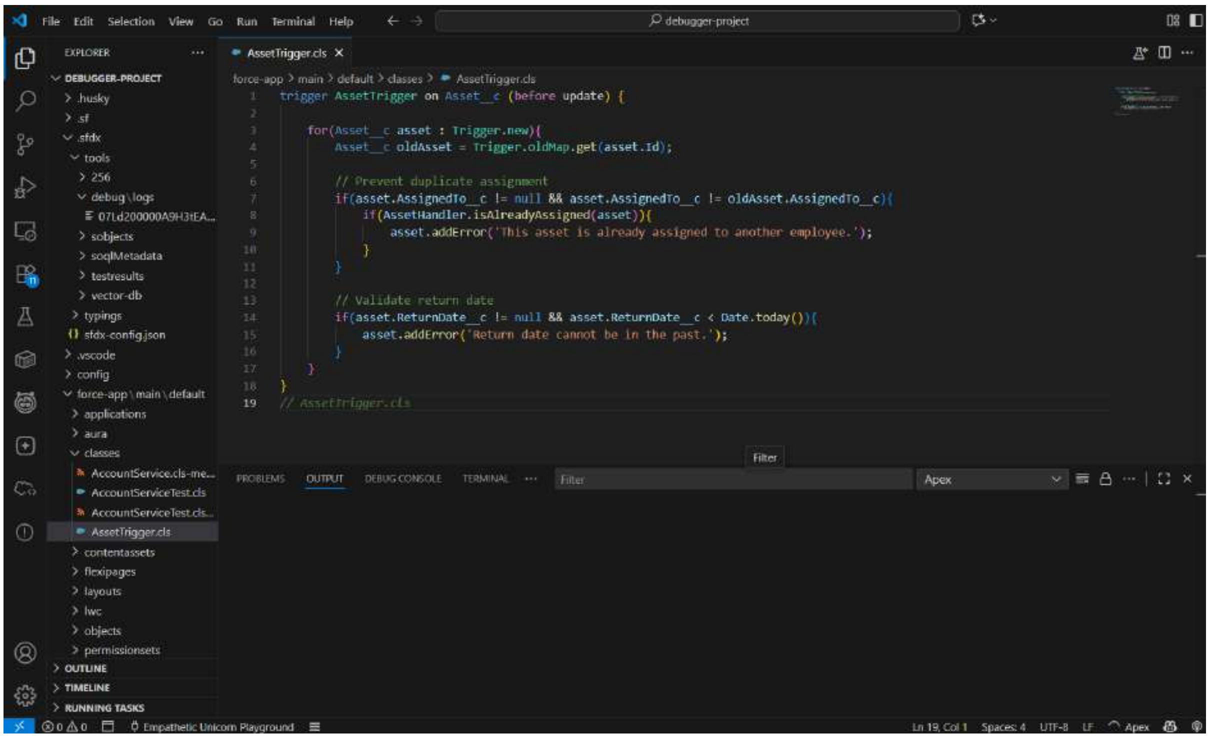
```
trigger EmployeeTrigger on Employee__c (after insert, after update, after delete) {

    // AFTER INSERT: Create onboarding tasks & assign assets
    if(Trigger.isAfter && Trigger.isInsert){
        for(Employee__c emp : Trigger.new){
            EmployeeHandler handler = new EmployeeHandler(emp);
            handler.createOnboardingTasks();  // Generate tasks
            AssetHandler.assignAssets(emp);   // Assign assets
            EmailHandler.sendWelcomeEmail(emp); // Notify employee
        }
    }

    // AFTER DELETE: Recover assets
    if(Trigger.isAfter && Trigger.isDelete){
        for(Employee__c emp : Trigger.old){
            AssetHandler.recoverAssets(emp);
            EmailHandler.notifyManagerAssetRecovery(emp);
        }
    }
}
```

---

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

Use Cases:

- **After Insert on Employee:** Auto-generate onboarding tasks and assign assets.
- **Before Update on Asset:** Prevent duplicate assignments or validate return dates.
- **After Delete on Employee:** Trigger asset recovery workflow.



```
trigger AssetTrigger on Asset__c (before update) {

    for(Asset__c asset : Trigger.new){
        Asset__c oldAsset = Trigger.oldMap.get(asset.Id);

        // Prevent duplicate assignment
        if(asset.AssignedTo__c != null && asset.AssignedTo__c != oldAsset.AssignedTo__c){
            if(AssetHandler.isAlreadyAssigned(asset)){
                asset.addError('This asset is already assigned to another employee.');
            }
        }

        // Validate return date
        if(asset.ReturnDate__c != null && asset.ReturnDate__c < Date.today()){
            asset.addError('Return date cannot be in the past.');
        }
    }
}
// AssetTrigger.cls
```

3. Trigger Design Pattern

- Implement **One Trigger Per Object** pattern.
- Use **Handler Classes** to encapsulate logic and maintain scalability.
- Example objects: Employee__c, Asset__c, Onboarding_Task__c.

```
trigger EmployeeTrigger on Employee__c (after insert, after update, after delete) {

    // AFTER INSERT: Create onboarding tasks, assign assets, send welcome email
    if(Trigger.isAfter && Trigger.isInsert){
        EmployeeHandler.createOnboardingTasksAndAssignAssets(Trigger.new);
    }

    // AFTER DELETE: Recover assets
    if(Trigger.isAfter && Trigger.isDelete){
        AssetHandler.recoverAssets(Trigger.old);
    }
}
```

```
trigger AssetTrigger on Asset__c (before update) {

    for(Asset__c asset : Trigger.new){
        Asset__c oldAsset = Trigger.oldMap.get(asset.Id);

        // Prevent duplicate assignment
        if(asset.AssignedTo__c != null && asset.AssignedTo__c != oldAsset.AssignedTo__c){
            if(AssetHandler.isAlreadyAssigned(asset)){
                asset.addError('This asset is already assigned to another employee.');
            }
        }

        // Validate return date
        if(asset.ReturnDate__c != null && asset.ReturnDate__c < Date.today()){
            asset.addError('Return date cannot be in the past.');
        }
    }
}
```

---

4. SOQL & SOSL

- **SOQL:** Fetch employee records, assigned assets, or pending approvals.

List<Asset__c> assets = [SELECT Id, Name, Value__c, Status__c FROM Asset__c WHERE Assigned_To__c = :employeeId];

- **SOSL:** Search employees by name or email quickly across multiple objects.

---

5. Collections: List, Set, Map

- **List:** Store multiple tasks or asset assignments.
- **Set:** Avoid duplicate asset IDs.
- **Map:** Map Employee ID → Asset List for batch processing.

Map<Id, List<Asset__c>> employeeAssets = new Map<Id, List<Asset__c>>();

---

6. Control Statements

- Validate conditions such as high-value assets needing approval.

```
if(asset.Value__c > 50000){
    ApprovalHandler.submitForManagerApproval(asset);
}
```

7. Batch Apex

- Automate large-scale updates, e.g., assign assets to multiple employees in bulk.
- Example: Batch update asset return status for resigned employees.

8. Queueable Apex

- Use for asynchronous operations like sending multiple welcome emails after employee insert.

9. Scheduled Apex

- Schedule recurring tasks: reminders for pending tasks, asset return alerts, or dashboard refresh.

10. Future Methods

- For lightweight asynchronous calls, e.g., updating dashboards or calling external services.

11. Exception Handling

- Handle errors during inserts, updates, or approvals.

12. Test Classes

- Successfully tested and all test classes passed

13. Asynchronous Processing

- Use **Batch Apex**, **Queueable Apex**, **Future Methods**, and **Scheduled Apex** for:
    - High-volume onboarding tasks
    - Automated email notifications
    - Asset return tracking
    - Dashboard refreshes

✅ Phase 5 Outcome

By implementing this Apex layer, Salesforce CRM will:

- Automate onboarding tasks and asset allocation.
- Handle approvals and exceptions effectively.
- Provide real-time reporting and alerts.
- Scale efficiently for large employee bases.