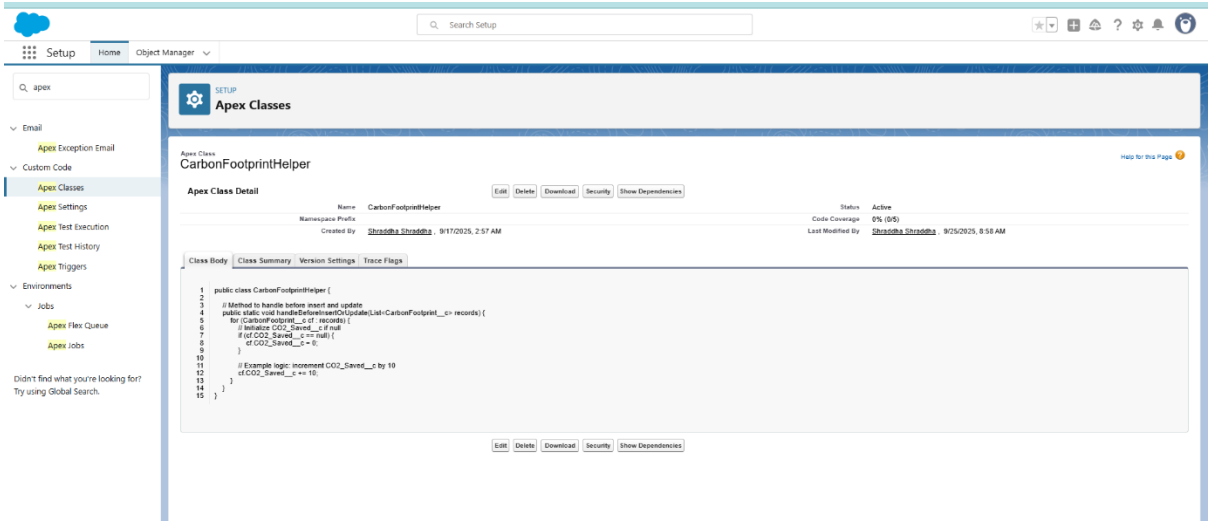
Phase 5: Apex Programming (Developer)

Step 1: Classes & Objects

- **Apex Classes** are reusable units of code to implement business logic.

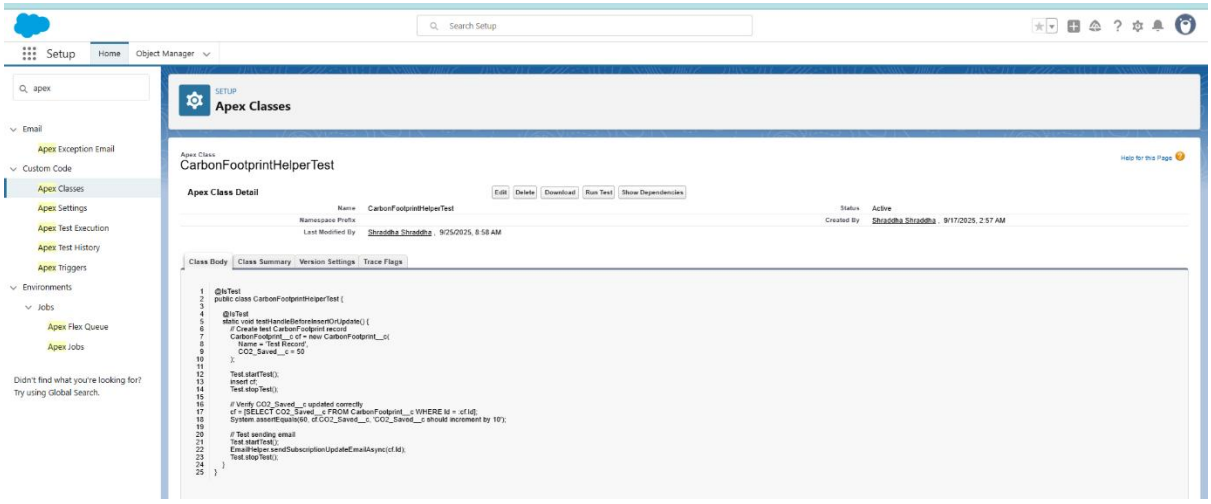
1.CarbonFootprintHelper:

- **Purpose:** Encapsulates all operations related to Carbon_Footprint__c, e.g., calculating total CO_2, updating related subscriptions, or performing bulk-safe operations.
- **Usage in Project:** Called from triggers or flows to keep Subscription__c.Total_CO2__c accurate.



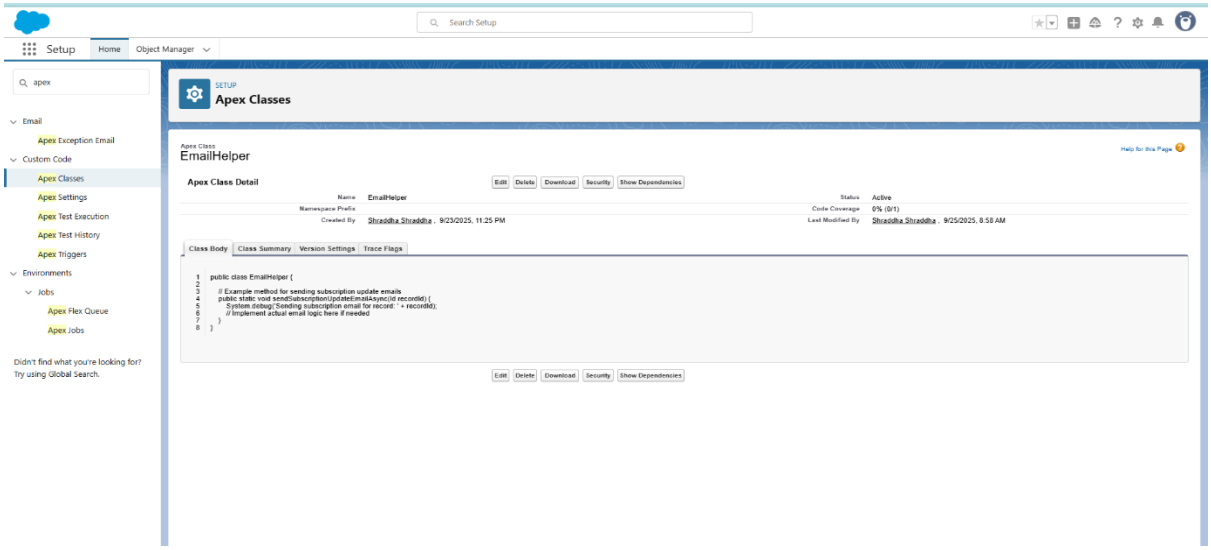
2. CarbonFootprintHelperTest

- **Purpose:** Unit tests for CarbonFootprintHelper ensuring proper logic, bulk operations, and test coverage for deployment.



3. EmailHelper

- **Purpose:** Handles sending emails for notifications like subscription CO_2 updates or expiry alerts.



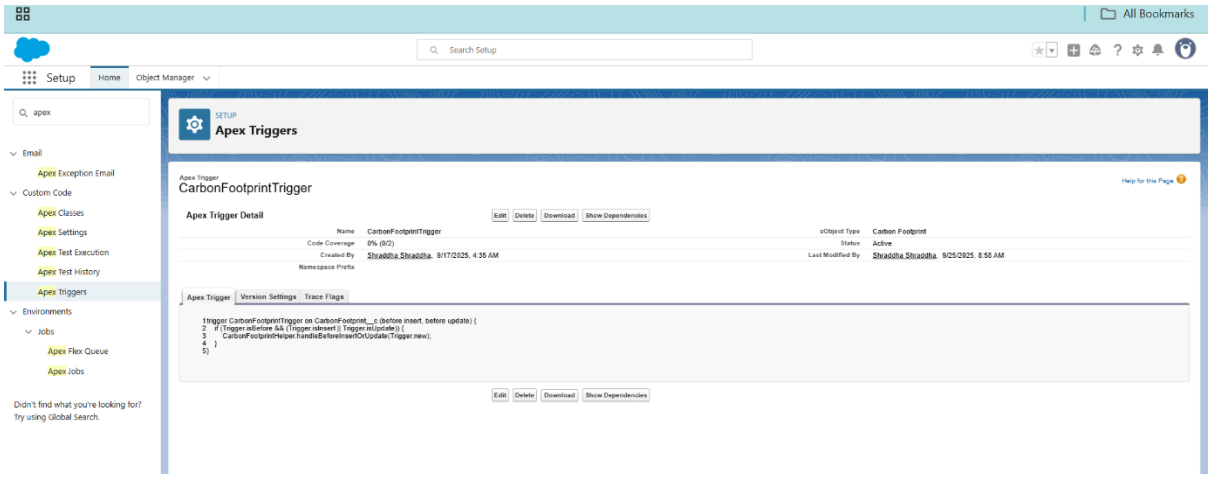
Step 2: Apex Trigger:

Automate actions when records change.

Example: CarbonFootprintTrigger

Purpose:

- Automatically updates the Total_CO2__c field on the related Subscription__c whenever a Carbon_Footprint__c record is inserted, updated, or deleted.
- Ensures real-time aggregation of CO_2 for each subscription.
- Can also trigger email notifications via EmailHelper if needed.



Step 3: SOQL & SOSL

SOQL: Query Salesforce objects.

Code: List<Subscription__c> subs = [SELECT Id, Name, Total_CO2__c

FROM Subscription__c

WHERE Total_CO2__c > 100];

SOSL: Search across multiple objects.

Code: List<List<SObject>> results = [FIND 'GreenEnergy*' IN ALL FIELDS

RETURNING Subscription__c(Name)];

Step 4: Collections: List, Set, Map

- **Location:** Used inside Apex classes, triggers, or anonymous Apex.

Example:

List<Subscription__c> subsList = new List<Subscription__c>();

Set<Id> subIds = new Set<Id>();

Map<Id, Decimal> subCO2Map = new Map<Id, Decimal>();

Step 5: Batch Apex

- **Location:** Setup → Apex Classes → New.
- **Purpose:** Process large datasets asynchronously in batches.

```
global class BatchUpdateCO2 implements Database.Batchable<SObject> {
    global Database.QueryLocator start(Database.BatchableContext BC){
        return Database.getQueryLocator('SELECT Id FROM Subscription__c');
    }
    global void execute(Database.BatchableContext BC, List<Subscription__c> scope){
        for(Subscription__c sub : scope){
            sub.Total_CO2__c = CarbonFootprintHelper.calculateTotalCO2(sub.Id);
        }
        update scope;
    }
    global void finish(Database.BatchableContext BC){}
}
```

Step 6: Queueable Apex

- **Purpose:** Chain asynchronous jobs, more flexible than future methods.

```
public class UpdateCO2Queueable implements Queueable {
    private Id subscriptionId;
    public UpdateCO2Queueable(Id subId){ this.subscriptionId = subId; }
    public void execute(QueueableContext context){
        Decimal total = CarbonFootprintHelper.calculateTotalCO2(subscriptionId);
        update new Subscription__c(Id=subscriptionId, Total_CO2__c=total);
    }
}
```

Step 7: Scheduled Apex

- **Purpose:** Run batch/queueable jobs on a schedule.

```
global class DailyCO2Scheduler implements Schedulable {
    global void execute(SchedulableContext sc){
        Database.executeBatch(new BatchUpdateCO2());
    }
}
// Schedule in Developer Console: System.schedule('Daily CO2 Job','0 0 0 * * ?', new DailyCO2Sched
```

Step 8: Future Methods

- **Purpose:** Run callouts or asynchronous operations.

```
@future
public static void notifyOwner(Id subscriptionId){
    Subscription__c sub = [SELECT Id, Owner.Email FROM Subscription__c WHERE Id=:subscriptionId];
    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
    mail.setToAddresses(new String[]{sub.Owner.Email});
    mail.setSubject('CO₂ Update Notification');
    Messaging.sendEmail(new Messaging.SingleEmailMessage[]{mail});
}
```

Step 9: Exception Handling

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

Step 10: Asynchronous Processing

- Use **Batch, Queueable, Scheduled, or Future** Apex to handle large datasets or time-consuming operations without hitting governor limits.
- Examples: updating total CO_2 across thousands of subscriptions, sending notifications, generating reports.