

# Phase 5 : Apex Programming(Developer)

## 1. Classes & Objects

The screenshot shows the Salesforce Setup Apex Classes page. The left sidebar contains links like Setup Home, Salesforce Go, Service Setup Assistant, Commerce Setup Assistant, Field Service Setup Home (Beta), Hyperforce Assistant, Release Updates, Salesforce Mobile App, Lighting Usage, Optimizer, Sales Cloud Everywhere, Administration, Users, Data, and Email. The main content area is titled "Apex Classes" and shows a table of classes. The columns include Action, Name, Developer Console, New, Generate from WSDL, Run All Tests, Schedule Apex, Namespace Prefix, Api Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags. The table lists various classes such as CarbonCreditManager, CarbonCreditQueries, CarbonPriceService, ChangePasswordController, ChangePasswordControllerTest, CommunitiesLandingController, CommunitiesLandingControllerTest, CommunitiesLoginController, CommunitiesLoginControllerTest, CommunitiesSelfRegConfirmController, CommunitiesSelfRegConfirmControllerTest, CommunitiesSelfRegController, and CommunitiesSelfRegControllerTest. The last modified by user is Eshwari Santosh Lade for most entries, with dates ranging from 27/09/2025 to 25/09/2025.

The screenshot shows the VS Code editor with the file "CarbonCreditManager.cls" open. The code defines a public class "CarbonCreditManager" with a static method "createCreditsForCompletedProjects". The method takes a list of "SustainabilityProject" objects and creates a list of "CarbonCredit" objects to insert. It iterates through the projects, checks if they are completed, and creates a new "CarbonCredit" object for each. The "CreditAmount\_c" is set to 100 if no budget is present or calculated based on the project budget. The "Type\_c" is set to "Earned", and other fields like "DateRecorded\_c" and "Status\_c" are set. Finally, it adds the credits to the insert list. The code uses annotations like `with sharing` and `public with sharing class`.

```
public with sharing class CarbonCreditManager {
    public static void createCreditsForCompletedProjects(List<SustainabilityProject__c> projects) {
        if (projects == null || projects.isEmpty()) return;
        List<CarbonCredit__c> creditsToInsert = new List<CarbonCredit__c>();
        for (SustainabilityProject__c p : projects) {
            if (p.Status__c == 'Completed') {
                CarbonCredit__c c = new CarbonCredit__c();
                c.Name = p.Name + ' Credit';
                // Default Logic: if budget present, estimate credits, otherwise default 100
                Decimal creditVal = 100;
                if (p.ProjectBudget__c != null && p.ProjectBudget__c > 0) {
                    creditVal = Math.max(1, (p.ProjectBudget__c / 1000).setScale(0));
                }
                c.CreditAmount__c = creditVal;
                c.Type__c = 'Earned';
                c.DateRecorded__c = Date.today();
                c.Project__c = p.Id;
                c.Status__c = 'Active';
                c.Source__c = 'Internal';
                creditsToInsert.add(c);
            }
        }
        if (!creditsToInsert.isEmpty()) {
            try {
                insert creditsToInsert;
            } catch (DmlException ex) {
                System.debug('Error inserting CarbonCredits: ' + ex.getMessage());
            }
        }
    }
}
```

The screenshot shows the Salesforce Setup Apex Classes page. The left sidebar includes links for Setup Home, Salesforce Go, Service Setup Assistant, Commerce Setup Assistant, Field Service Setup Home (Beta), Hyperforce Assistant, Release Updates, Salesforce Mobile App, Lighting Usage, Optimizer, Sales Cloud Everywhere, Administration (Users, Data, Email), and Platform Tools. The main content area displays the Apex Classes section with a table listing various Apex classes. The table columns include Action, Name, Namespace Prefix, API Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags. A message at the top indicates 0.38% of apex code is used.

Action	Name	Namespace Prefix	API Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit   Del   Security	SiteLoginController		64.0	Active	352	Eshwar Santosh Lade	25/09/2025, 12:31 am
Edit   Del	Site>LoginControllerTest		64.0	Active	393	Eshwar Santosh Lade	25/09/2025, 12:31 am
Edit   Del   Security	SiteRegisterController		64.0	Active	1,524	Eshwar Santosh Lade	25/09/2025, 12:31 am
Edit   Del	Site/RegisterControllerTest		64.0	Active	599	Eshwar Santosh Lade	25/09/2025, 12:31 am
Edit   Del   Security	SustainabilityProjectTriggerHandler		58.0	Active	1,005	Eshwar Santosh Lade	27/09/2025, 3:47 am

The screenshot shows the GreenFutureProject code editor. The left sidebar lists files and folders: .gitignore, force-app/main/default/classes (CarbonCreditManager, CarbonCreditManagerTest, CarbonCreditQueries, CarbonCreditQueriesTest, ExpireCreditsBatch, ExpireCreditsBatchTest, NotifyQueueable, NotifyQueueableTest), force-app/main/default/layouts, force-app/main/default/resource, project-scratch-def.json, config, .vscode, .sfdx, and .gitignore. The right pane displays the code for CarbonCreditQueries.cls:

```

force-app > main > default > classes > ● CarbonCreditQueries.cls ...
1  public with sharing class CarbonCreditQueries {
2    public static List<CarbonCredit__c> getActiveCreditsByProjectIds(Set<Id> projectIds) {
3      if (projectIds == null || projectIds.isEmpty()) return new List<CarbonCredit__c>();
4      return [
5        SELECT Id, Name, CreditAmount__c, Status__c, DateRecorded__c, Project__c
6        FROM CarbonCredit__c
7        WHERE Project__c IN :projectIds AND Status__c = 'Active'
8      ];
9    }
10
11   public static List<Partner__c> findPartnerByNameOrEmail(String searchTerm) {
12     if (String.isBlank(searchTerm)) return new List<Partner__c>();
13     List<Object> results = [FIND :searchTerm IN ALL FIELDS RETURNING Partner__c(Id, Name, Email__c LIMIT 10)];
14     return (results.size() > 0 && results[0] != null) ? (List<Partner__c>) results[0] : new List<Partner__c>();
15   }
16 }
17
18

```

The status bar at the bottom shows: Line 17, Col 1, Spaces: 4, UTF-8, LF, Apex, Go Live, Prettier, ENG IN, 03:55, 27-09-2025.

```

force-app > main > default > classes > SustainabilityProjectTriggerHandler.cls ...
1  public with sharing class SustainabilityProjectTriggerHandler {
2      private static Boolean executed = false;
3
4      public static void afterUpdate(List<SustainabilityProject__c> newList, Map<Id, SustainabilityProject__c> oldMap) {
5          if (executed) return;
6          executed = true;
7
8          List<SustainabilityProject__c> completedProjects = new List<SustainabilityProject__c>();
9          for (SustainabilityProject__c p : newList) {
10              SustainabilityProject__c oldP = oldMap.get(p.id);
11              if (oldP != null && oldP.Status__c != 'Completed' && p.Status__c == 'Completed') {
12                  completedProjects.add(p);
13              }
14          }
15
16          if (!completedProjects.isEmpty()) {
17              CarbonCreditManager.createCreditsForCompletedProjects(completedProjects);
18
19              // Enqueue notifications/tasks for reviewing created credits (non-blocking)
20              List<Id> projectIds = new List<Id>();
21              for (SustainabilityProject__c p : completedProjects) projectIds.add(p.id);
22              System.enqueueJob(new NotifyQueueable(projectIds));
23          }
24      }
25  }
26

```

Fig.5.1. Apex Classes List and Code

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

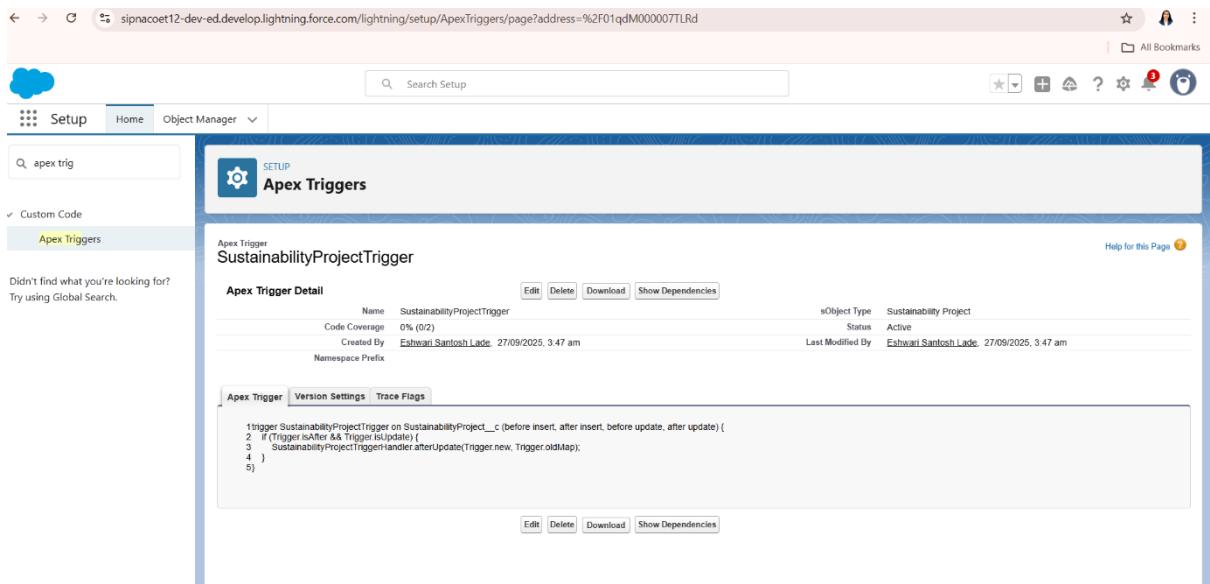


Fig.5.2. Apex Triggers List

## 3. Trigger Design Pattern

```

force-app > main > default > classes > SustainabilityProjectTriggerHandler.cls ...
1  public with sharing class SustainabilityProjectTriggerHandler {
2      private static Boolean executed = false;
3
4      public static void afterUpdate(List<SustainabilityProject__c> newList, Map<Id, SustainabilityProject__c> oldMap) {
5          if (executed) return;
6          executed = true;
7
8          List<SustainabilityProject__c> completedProjects = new List<SustainabilityProject__c>();
9          for (SustainabilityProject__c p : newList) {
10              SustainabilityProject__c oldP = oldMap.get(p.Id);
11              if (oldP != null && oldP.Status__c != 'Completed' && p.Status__c == 'Completed') {
12                  completedProjects.add(p);
13              }
14          }
15
16          if (!completedProjects.isEmpty()) {
17              CarbonCreditManager.createCreditsForCompletedProjects(completedProjects);
18
19              // Enqueue notifications/tasks for reviewing created credits (non-blocking)
20              List<Id> projectIds = new List<Id>();
21              for (SustainabilityProject__c p : completedProjects) projectIds.add(p.Id);
22              System.enqueueJob(new NotifyQueueable(projectIds));
23          }
24      }
25  }
26

```

Fig.5.3. Handler Pattern

## 4. SOQL & SOSL

```

force-app > main > default > classes > CarbonCreditQueries.cls ...
1  public with sharing class CarbonCreditQueries {
2      public static List<CarbonCredit__c> getActiveCreditsByProjectIds(Set<Id> projectIds) {
3          if (projectIds == null || projectIds.isEmpty()) return new List<CarbonCredit__c>();
4          return [
5              SELECT Id, Name, CreditAmount__c, Status__c, DateRecorded__c, Project__c
6              FROM CarbonCredit__c
7              WHERE Project__c IN :projectIds AND Status__c = 'Active'
8          ];
9      }
10
11      public static List<Partner__c> findPartnerByNameOrEmail(String searchTerm) {
12          if (String.isBlank(searchTerm)) return new List<Partner__c>();
13          List<Object> results = [FIND :searchTerm IN ALL FIELDS RETURNING Partner__c(Id, Name, Email__c LIMIT 10)];
14          return (results.size() > 0 && results[0] != null) ? (List<Partner__c>) results[0] : new List<Partner__c>();
15      }
16  }
17
18

```

Fig.5.4. CarbonCreditQueries Code

## 5. Collections: List, Set, Map

The screenshot shows the Salesforce IDE interface with the file `CarbonCreditManager.cls` open. The code implements a static method `createCreditsForCompletedProjects` that iterates through a list of completed projects, creates a new `CarbonCredit` record for each, and inserts them into a database. The code uses several loops and conditions to handle different project statuses and budget logic.

```
force-app > main > default > classes > CarbonCreditManager.cls > ...
1 public with sharing class CarbonCreditManager {
2     public static void createCreditsForCompletedProjects(List<sustainabilityProject__c> projects) {
3         if (projects == null || projects.isEmpty()) return;
4
5         List<CarbonCredit__c> creditsToInsert = new List<Carboncredit__c>();
6
7         for (SustainabilityProject__c p : projects) {
8             if (p.Status__c == 'Completed') {
9                 CarbonCredit__c c = new CarbonCredit__c();
10                c.Name = p.Name + ' Credit';
11
12                // Default logic: if budget present, estimate credits, otherwise default 100
13                Decimal creditVal = 100;
14                if (p.ProjectBudget__c != null && p.ProjectBudget__c > 0) {
15                    creditVal = Math.max(1, (p.ProjectBudget__c / 1000).setScale(0));
16                }
17                c.CreditAmount__c = creditVal;
18
19                c.Type__c = 'Earned';
20                c.DateRecorded__c = Date.today();
21                c.Project__c = p.Id;
22                c.Status__c = 'Active';
23                c.Source__c = 'Internal';
24
25                creditsToInsert.add(c);
26            }
27
28            if (!creditsToInsert.isEmpty()) {
29                try {
30                    insert creditsToInsert;
31                } catch (DmlException ex) {
32                    System.debug('Error inserting Carboncredits: ' + ex.getMessage());
33                }
34            }
35        }
36    }
37
38    > contentassets
39    > flexipages
40    > layouts
41 }
```

Fig.5.5. CarbonCreditManager Code

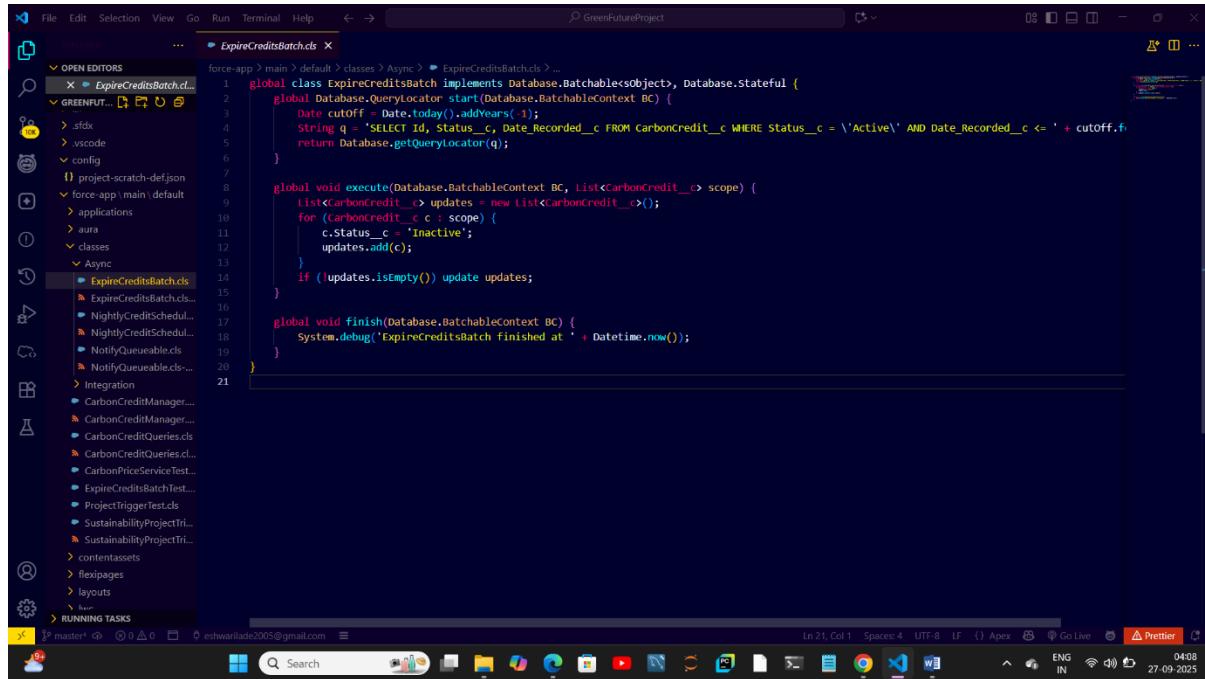
## 6. Control Statements

This screenshot is identical to Fig.5.5, but it highlights the `for` loop and the nested `try` block in the code editor. These sections demonstrate the use of loops for iteration and conditions for error handling.

```
force-app > main > default > classes > CarbonCreditManager.cls > ...
1 public with sharing class CarbonCreditManager {
2     public static void createCreditsForCompletedProjects(List<sustainabilityProject__c> projects) {
3         if (projects == null || projects.isEmpty()) return;
4
5         List<CarbonCredit__c> creditsToInsert = new List<Carboncredit__c>();
6
7         for (SustainabilityProject__c p : projects) {
8             if (p.Status__c == 'Completed') {
9                 CarbonCredit__c c = new CarbonCredit__c();
10                c.Name = p.Name + ' Credit';
11
12                // Default logic: if budget present, estimate credits, otherwise default 100
13                Decimal creditVal = 100;
14                if (p.ProjectBudget__c != null && p.ProjectBudget__c > 0) {
15                    creditVal = Math.max(1, (p.ProjectBudget__c / 1000).setScale(0));
16                }
17                c.CreditAmount__c = creditVal;
18
19                c.Type__c = 'Earned';
20                c.DateRecorded__c = Date.today();
21                c.Project__c = p.Id;
22                c.Status__c = 'Active';
23                c.Source__c = 'Internal';
24
25                creditsToInsert.add(c);
26            }
27
28            if (!creditsToInsert.isEmpty()) {
29                try {
30                    insert creditsToInsert;
31                } catch (DmlException ex) {
32                    System.debug('Error inserting Carboncredits: ' + ex.getMessage());
33                }
34            }
35        }
36    }
37
38    > contentassets
39    > flexipages
40    > layouts
41 }
```

Fig.5.6. CarbonCreditManager Loops & Conditions

## 7. Batch Apex

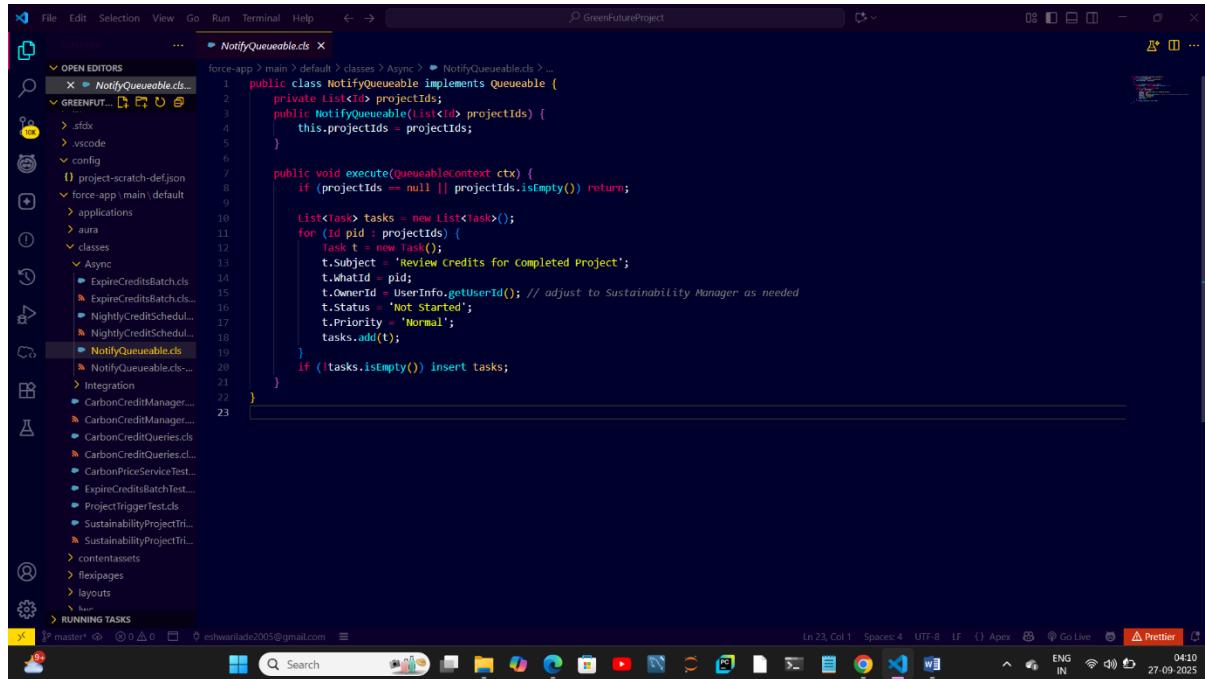


The screenshot shows the VS Code interface with the file `ExpireCreditsBatch.cls` open in the editor. The code implements a batch apex class named `ExpireCreditsBatch`. It includes methods for starting the batch, executing updates, and finishing the batch. The code uses SOQL to query records and update their status to 'Inactive'.

```
force-app > main > default > classes > Async > ExpireCreditsBatch.cls > ...
1 global class ExpireCreditsBatch implements Database.Batchable<Object>, Database.Stateful {
2     global Database.QueryLocator start(Database.BatchableContext BC) {
3         date cutoff = Date.today().addYears(-1);
4         String q = 'SELECT Id, Status__c, Date_Recorded__c FROM CarbonCredit__c WHERE Status__c = \'Active\' AND Date_Recorded__c <= ' + cutoff;
5         return Database.getQueryLocator(q);
6     }
7
8     global void execute(Database.BatchableContext BC, List<CarbonCredit__c> scope) {
9         List<CarbonCredit__c> updates = new List<CarbonCredit__c>();
10        for (CarbonCredit__c c : scope) {
11            c.Status__c = 'Inactive';
12            updates.add(c);
13        }
14        if (!updates.isEmpty()) update updates;
15    }
16
17    global void finish(Database.BatchableContext BC) {
18        System.debug('ExpireCreditsBatch finished at ' + Datetime.now());
19    }
20}
```

Fig.5.7. ExpireCreditsBatch Code

## 8. Queueable Apex

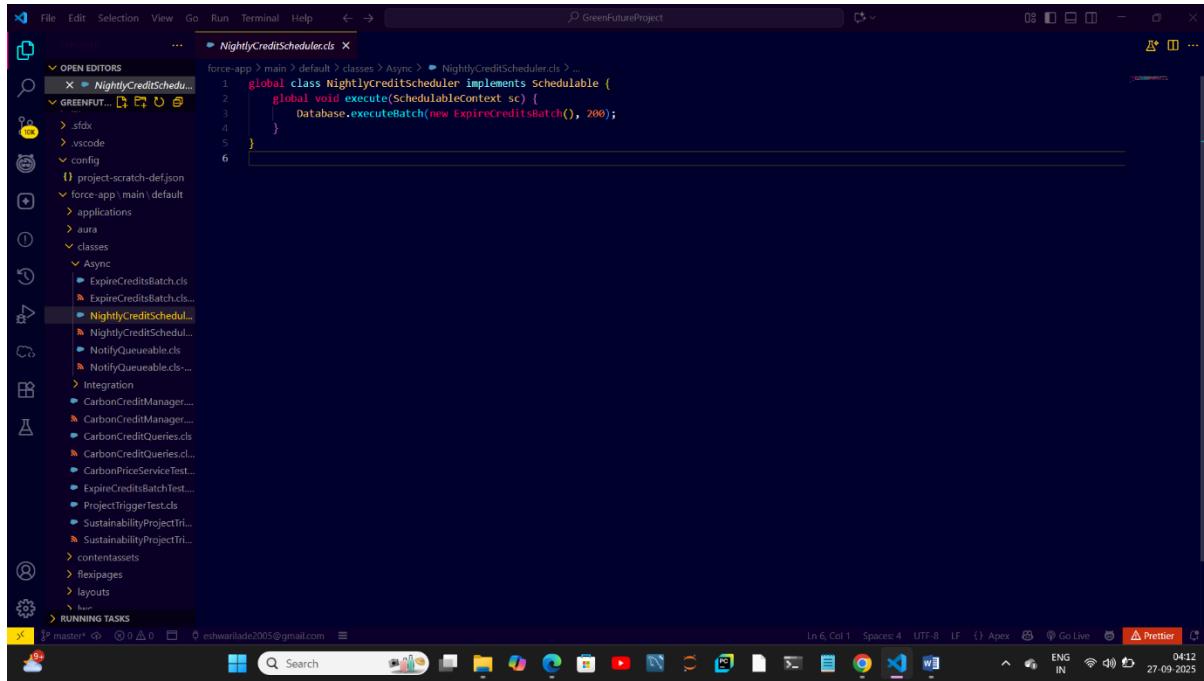


The screenshot shows the VS Code interface with the file `NotifyQueueable.cls` open in the editor. The code implements a queueable apex class named `NotifyQueueable`. It takes a list of project IDs and creates tasks for each ID, setting the subject to 'Review Credits for Completed Project'. The code uses the `UserInfo.getUserId()` method to get the user ID.

```
force-app > main > default > classes > Async > NotifyQueueable.cls > ...
1 public class NotifyQueueable implements Queueable {
2     private List<Id> projectIds;
3     public NotifyQueueable(List<Id> projectIds) {
4         this.projectIds = projectIds;
5     }
6
7     public void execute(ExecutionContext ctx) {
8         if (projectIds == null || projectIds.isEmpty()) return;
9
10        List<Task> tasks = new List<Task>();
11        for (Id pid : projectIds) {
12            Task t = new Task();
13            t.Subject = 'Review Credits for Completed Project';
14            t.WhatId = pid;
15            t.OwnerId = UserInfo.getUserId(); // adjust to sustainability Manager as needed
16            t.Status = 'Not Started';
17            t.Priority = 'Normal';
18            tasks.add(t);
19        }
20        if (!tasks.isEmpty()) insert tasks;
21    }
22}
```

Fig.5.8. NotifyQueueable Code

## 9. Scheduled Apex

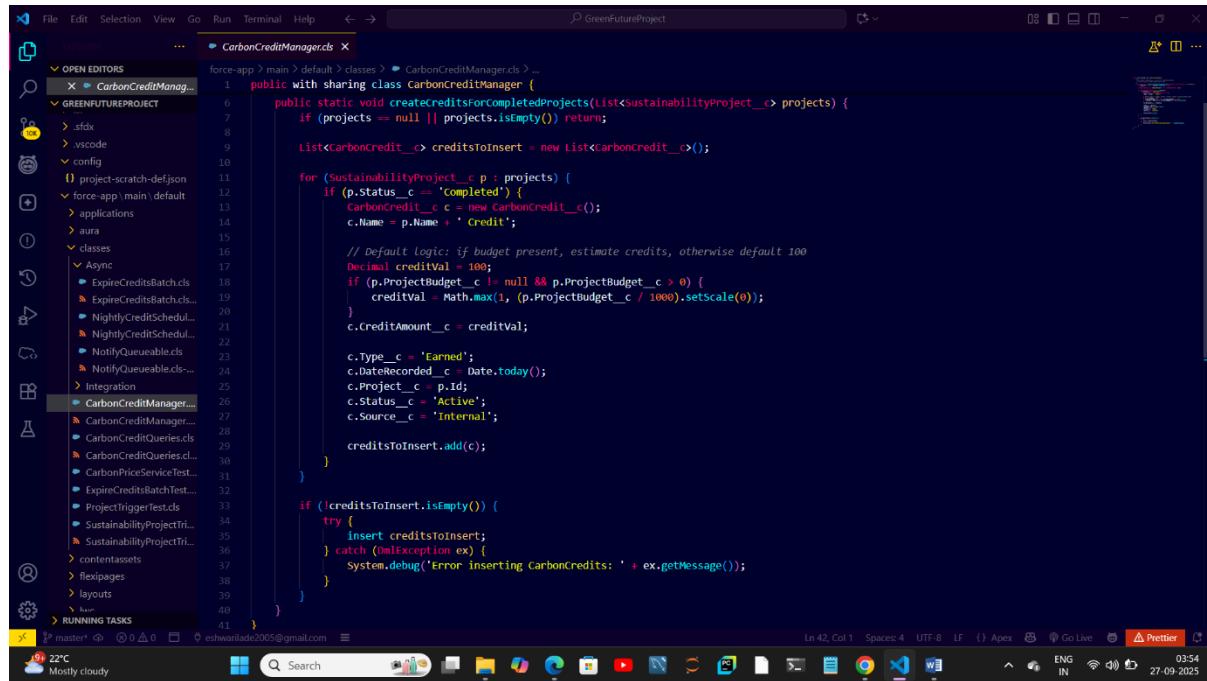


The screenshot shows the VS Code interface with the file 'NightlyCreditScheduler.cls' open in the editor. The code implements the 'Scheduled' interface with a single method 'execute'. The code is as follows:

```
global class nightlyCreditscheduler implements Scheduledable {
    global void execute(SchedulableContext sc) {
        Database.executeBatch(new ExpireCreditsBatch(), 200);
    }
}
```

Fig.5.9. NightlyCreditScheduler Code

## 10. Exception Handling



The screenshot shows the VS Code interface with the file 'CarbonCreditManager.cls' open in the editor. The code defines a class with a static method 'createCreditsForCompletedProjects'. The code handles exceptions and inserts records into a 'CarbonCredit' object. The code is as follows:

```
public static void createCreditsForCompletedProjects(List<SustainabilityProject__c> projects) {
    if (projects == null || projects.isEmpty()) return;

    List<CarbonCredit__c> creditsToInsert = new List<CarbonCredit__c>();

    for (SustainabilityProject__c p : projects) {
        if (p.Status__c == 'Completed') {
            CarbonCredit__c c = new CarbonCredit__c();
            c.Name = p.Name + ' Credit';

            // Default logic: if budget present, estimate credits, otherwise default 100
            Decimal creditVal = 100;
            if (p.ProjectBudget__c != null && p.ProjectBudget__c > 0) {
                creditVal = Math.max(1, (p.ProjectBudget__c / 1000).setScale(0));
            }
            c.CreditAmount__c = creditVal;

            c.Type__c = 'Earned';
            c.DateRecorded__c = Date.today();
            c.Project__c = p.Id;
            c.Status__c = 'Active';
            c.Source__c = 'Internal';

            creditsToInsert.add(c);
        }
    }

    if (!creditsToInsert.isEmpty()) {
        try {
            insert creditsToInsert;
        } catch (OmException ex) {
            System.debug('Error inserting CarbonCredits: ' + ex.getMessage());
        }
    }
}
```

Fig.5.10. CarbonCreditManager Code

## 11. Test Classes

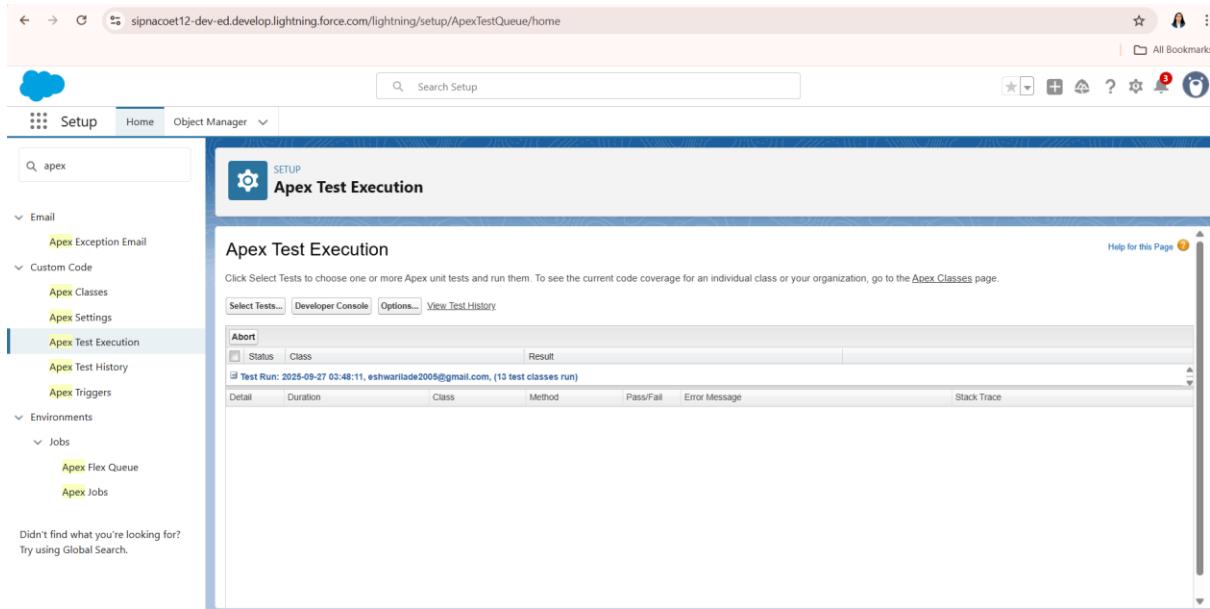


Fig.5.11. Apex Test Execution

The screenshot shows the VS Code interface with the terminal tab active. The terminal output shows the command 'sf project deploy start --source-dir force-app --target-org ProductionOrgAlias' being run, followed by deployment logs. The logs indicate a successful deployment with Deploy ID: 0afdf000000PzubSAJ, Target Org: eshwarilade2005@gmail.com, and Elapsed Time: 8.52s. Below the terminal, a table titled 'Deployed Source' lists the deployed classes with their state, name, type, and path. The table includes rows for ExpireCreditsBatch, ExpireCreditsBatch, NightlyCreditscheduler, NightlyCreditscheduler, NotifyQueueable, NotifyQueueable, CarboncreditManager, CarboncreditManager, CarboncreditQueries, CarboncreditQueries, and CarbonPriceService.

State	Name	Type	Path
Created	ExpireCreditsBatch	ApexClass	force-app/main/default/classes/Async/ExpireCreditsBatch.cls
Created	ExpireCreditsBatch	ApexClass	force-app/main/default/classes/Async/ExpireCreditsBatch.cls-meta.xml
Created	NightlyCreditscheduler	ApexClass	force-app/main/default/classes/Async/NightlyCreditscheduler.cls
Created	NightlyCreditscheduler	ApexClass	force-app/main/default/classes/Async/NightlyCreditscheduler.cls-meta.xml
Created	NotifyQueueable	ApexClass	force-app/main/default/classes/Async/NotifyQueueable.cls
Created	NotifyQueueable	ApexClass	force-app/main/default/classes/Async/NotifyQueueable.cls-meta.xml
Created	CarboncreditManager	ApexClass	force-app/main/default/classes/CarbonCreditManager.cls
Created	CarboncreditManager	ApexClass	force-app/main/default/classes/CarbonCreditManager.cls-meta.xml
Created	CarboncreditQueries	ApexClass	force-app/main/default/classes/CarbonCreditQueries.cls
Created	CarboncreditQueries	ApexClass	force-app/main/default/classes/CarbonCreditQueries.cls-meta.xml
Created	CarbonPriceService	ApexClass	force-app/main/default/classes/integration/CarbonPriceService.cls

The screenshot shows the VS Code interface with the following details:

- File Explorer:** On the left, it lists "OPEN EDITORS" and "GREENFUTUREPROJECT". Inside "GREENFUTUREPROJECT", there are several files and folders: ".husky", ".sfdx", ".vscode", "config", "force-app", "scripts", ".forceignore", ".gitignore", ".prettierignore", ".eslint.config.js", ".jest.config.js", "package.json", "README.md", and "sfdx-project.json".
- Terminal:** The active terminal tab is "TERMINAL". It shows the command "powershell" running.
- Deployed Source:** This section displays a table of deployed source code elements. The columns are "State", "Name", "Type", and "Path". The table contains numerous rows, mostly starting with "Created" or "Changed", representing various Apex classes, custom fields, and objects from the "force-app/main/default/classes", "force-app/main/default/classes/Asynchronous", and "force-app/main/default/classes/Integration" paths.
- Bottom Status Bar:** Shows the status "master" and the date "27-09-2025".

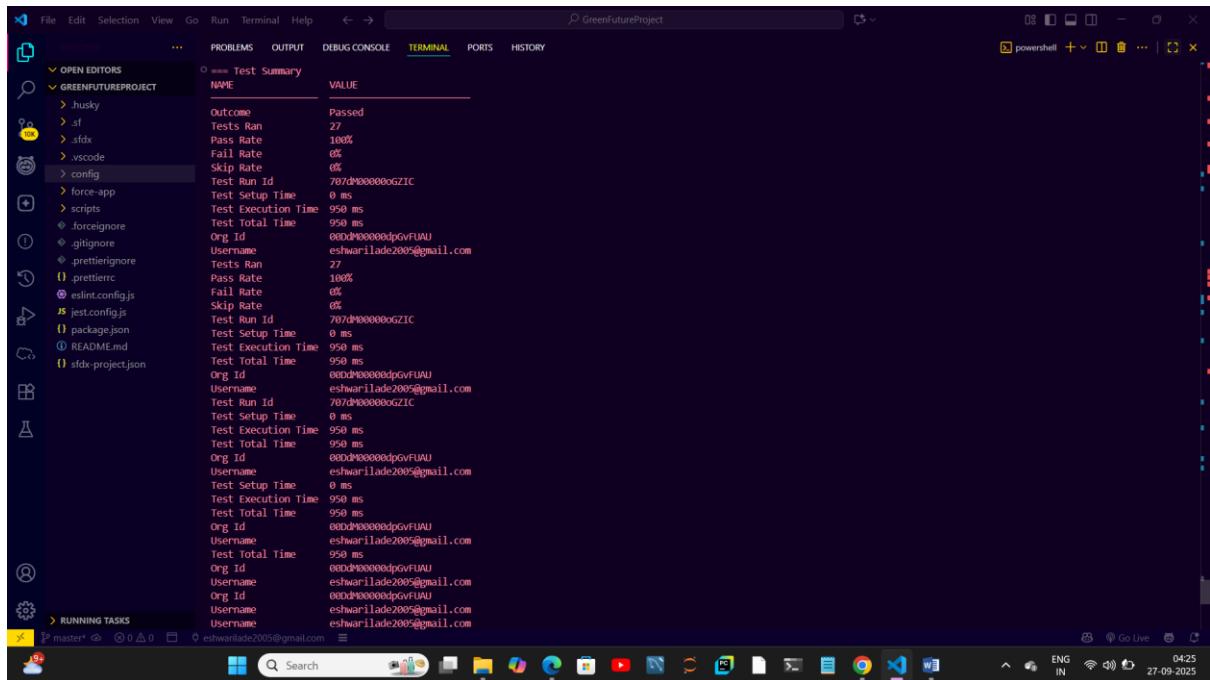


Fig.5.12. Test Results

## 12. Asynchronous Processing

- Covered with **Batch, Queueable, Scheduled**