

EXPENSE TRACKER APP

Phase 5: Apex Programming (Developer)

Goal: Add advanced logic and automation for Expense Tracker using Apex.

1. Apex Service Class

Purpose: Centralize business logic for reusability in triggers, flows, and batch classes.

Navigation:

Setup → Apex Classes → New

The screenshot shows the Apex Class Detail page for the ExpenseService class. The page includes the class name, namespace prefix, creation date, status, and code coverage information. The code body contains two static methods: validateAmount and updateStatus. The validateAmount method checks if the expense amount is greater than 0. The updateStatus method updates the expense status to 'Confirmed' if it was previously 'Approved'.

```
public class ExpenseService {
    public static void validateAmount(Expense__c expense) {
        if(expense.Amount__c < 0) {
            expense.addError('Expense amount must be greater than 0');
        }
    }
    // Calculate status after approval
    public static void updateStatus(Expense__c expense) {
        if(expense.Approval_Status__c == 'Approved') {
            expense.Expense_Status__c = 'Confirmed';
        }
    }
}
```

2. Apex Trigger

Purpose: Apply logic automatically when records are inserted or updated.

Navigation:

Setup → Object Manager → Expense → Triggers → New

The screenshot shows the Apex Trigger Detail page for the ExpenseTrigger. The page includes the trigger name, object type, namespace prefix, and code coverage information. The trigger code is defined as a before insert and before update trigger on the Expense__c object, which calls the validateAmount and updateStatus methods from the ExpenseService class.

```
trigger ExpenseTrigger on Expense__c (before insert, before update) {
    for(Expense__c exp : Trigger.new) {
        ExpenseService.validateAmount(exp);
        ExpenseService.updateStatus(exp);
    }
}
```

3. Batch Apex — Overdue Expenses

Purpose: Automatically mark expenses as overdue if not approved and past the expense date.

Navigation:

Setup → Apex Classes → New

The screenshot shows the Apex Class Detail page for 'BatchOverdueExpenses'. The class implements Database.Batchable<Object>. The code queries for expenses where the approval status is not 'Approved' and the expense date is today or earlier. It then updates the expense status to 'Overdue'. The class also includes a finish method. The page includes tabs for Class Body, Class Summary, Version Settings, and Trace Flags. Buttons for Edit, Delete, Download, Security, and Show Dependencies are at the top right. Status and Active indicators are shown on the right.

```
1 global class BatchOverdueExpenses implements Database.Batchable<Object> {
2     global Database.QueryLocator start(Database.BatchableContext BC) {
3         // Corrected put SOQL in a string using Database.getQueryLocator
4         String query = 'SELECT Id, Expense_Status__c FROM Expense__c WHERE Expense_Date__c < TODAY AND Approval_Status__c != \'Approved\'';
5         return Database.getQueryLocator(query);
6     }
7     global void execute(Database.BatchableContext BC, List<Expense__c> scope) {
8         for(Expense__c exp : scope) {
9             exp.Expense_Status__c = 'Overdue';
10        }
11    update scope;
12    }
13    global void finish(Database.BatchableContext BC) {
14        // Optional: Add post-processing logic here
15    }
16 }
17
18 }
```

4. Scheduler Class — Run Batch Automatically

Purpose: Schedule the batch to run daily/weekly.

Navigation:

Setup → Apex Classes → New

The screenshot shows the Apex Class Detail page for 'ScheduleOverdueExpenses'. The class implements Schedulable. It executes a batch job named 'BatchOverdueExpenses'. The page includes tabs for Class Body, Class Summary, Version Settings, and Trace Flags. Buttons for Edit, Delete, Download, Security, and Show Dependencies are at the top right. Status and Active indicators are shown on the right.

```
1 global class ScheduleOverdueExpenses implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         // Create batch job
4         BatchOverdueExpenses batchJob = new BatchOverdueExpenses();
5         Database.executeBatch(batchJob);
6     }
7 }
8 }
```

Steps to Schedule:

1. Setup → Apex Classes → Schedule Apex
2. Job Name: Overdue_Expenses_Batch
3. Apex Class: ScheduleOverdueExpenses

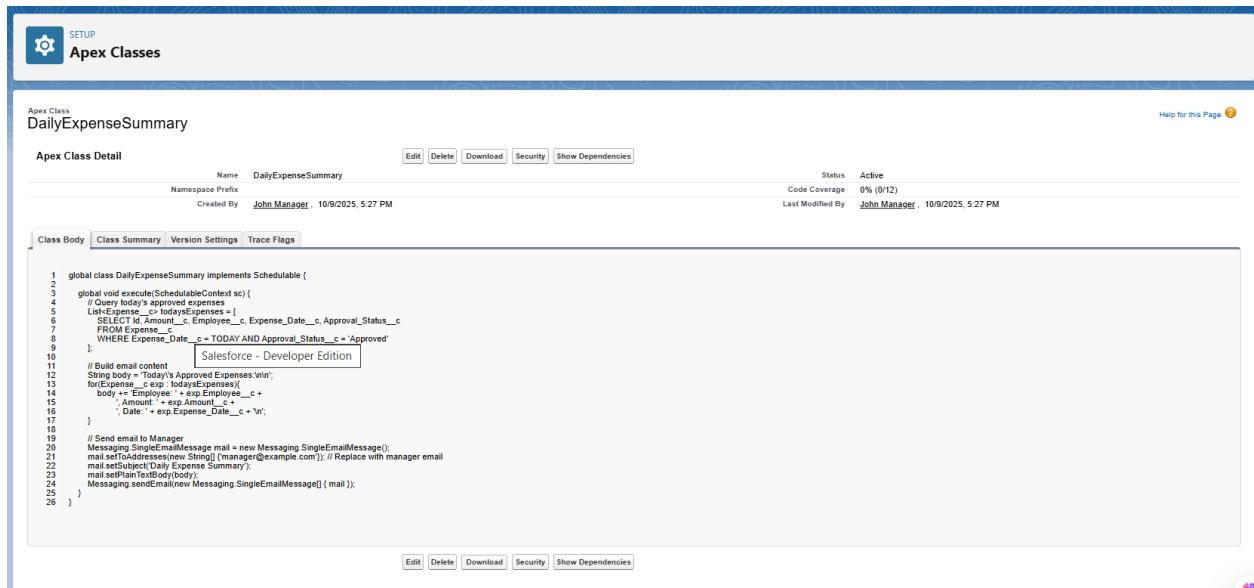
4. Frequency: Daily / Weekly
5. Set Start/End Dates and Time → Save

5. Scheduled Apex — Daily Expense Summary Email

Purpose: Send daily approved expense summary to manager.

Navigation:

Setup → Apex Classes → New



The screenshot shows the Apex Classes page in Salesforce. The top navigation bar has 'SETUP' and 'Apex Classes'. Below it, the page title is 'Apex Class DailyExpenseSummary'. The main content area shows the class details:

- Name:** DailyExpenseSummary
- Namespace Prefix:** (empty)
- Created By:** John Manager, 10/9/2025, 5:27 PM
- Status:** Active
- Code Coverage:** 0% (0/12)
- Last Modified By:** John Manager, 10/9/2025, 5:27 PM

The 'Class Body' tab is selected, displaying the Apex code:

```

1 global class DailyExpenseSummary implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         // Query today's approved expenses
4         List<Expense__c> todayExpenses = [
5             SELECT Id, Amount__c, Employee__c, Expense_Date__c, Approval_Status__c
6             FROM Expense__c
7             WHERE Expense_Date__c = TODAY AND Approval_Status__c = 'Approved'
8         ];
9     }
10    // Build email content
11    String body = 'Today's Approved Expenses:\n';
12    for(Expense__c exp : todayExpenses){
13        body += 'Employee: ' + exp.Employee__c +
14            ', Amount: ' + exp.Amount__c +
15            ', Date: ' + exp.Expense_Date__c + '\n';
16    }
17
18    // Send email to Manager
19    Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
20    mail.setToAddresses(new String[]{manager@example.com}); // Replace with manager email
21    mail.setSubject('Daily Expense Summary');
22    mail.setPlainTextBody(body);
23    Messaging.sendEmail(new Messaging.SingleEmailMessage[] { mail });
24
25 }
26

```

Steps to Schedule:

- Setup → Apex Classes → Schedule Apex → Select DailyExpenseSummary
- Frequency: Daily
- Preferred Start Time → Save

6. Test Classes (Mandatory for Deployment)

Purpose: Ensure code coverage and correctness before deployment.

Navigation:

Setup → Apex Classes → New

The screenshot shows the Apex Classes page for the ExpenseServiceTest class. The class is annotated with @IsTest. It contains a test method that inserts a new expense record with a negative amount and asserts that an exception is thrown with the message "Expense amount must be greater than 0". The class has 20 lines of code.

```

1  @IsTest
2  public class ExpenseServiceTest {
3      @TestSetup
4          void setup() {
5              User u = [SELECT Id FROM User WHERE Profile.Name='Standard User' LIMIT 1];
6              Expense__c exp = new Expense__c();
7              exp.Amount__c = -100;
8              exp.Expense_Date__c = Date.today();
9              exp.Employee__c = u.Id;
10         }
11     Test.startTest();
12     try {
13         insert exp;
14     } catch(Exception e) {
15         System.assert(e.getMessage().contains('Expense amount must be greater than 0'));
16     }
17     Test.stopTest();
18 }
19
20

```

Phase 5 Complete

- Apex Service Class for logic reuse
- Trigger for real-time validation and status updates
- Batch Apex to mark overdue expenses
- Scheduler class to automate batch execution
- Scheduled Apex for daily manager email
- Exception handling for data integrity
- Test Classes for deployment compliance

The screenshot shows a list of Apex classes in the devedapp namespace. The classes listed are BatchOverdueExpenses, DailyExpenseSummary, DeveloperEditionUtils, DeveloperEditionUtilsTest, ExpenseExternalAPI, ExpenseService, ExpenseServiceTest, PostInstallScript, PostInstallScriptTest, QueueableExpenseSum, and ScheduleOverdueExpenses. The table includes columns for Action, Name, Namespace Prefix, API Version, Status, Size Without Comments, Last Modified By, and Has Trace Flags.

| Action | Name | Namespace Prefix | API Version | Status | Size Without Comments | Last Modified By | Has Trace Flags |
|-----------------------|---------------------------|------------------|-------------|--------|-----------------------|------------------|--------------------|
| Edit Del Security | BatchOverdueExpenses | | 64.0 | Active | 606 | John Manager | 10/9/2025, 5:20 PM |
| Edit Del Security | DailyExpenseSummary | | 64.0 | Active | 948 | John Manager | 10/9/2025, 5:27 PM |
| Edit Security | DeveloperEditionUtils | devedapp | 64.0 | Active | 164 | OrgFarm EPIC | 10/3/2025, 7:41 PM |
| Edit Security | DeveloperEditionUtilsTest | devedapp | 64.0 | Active | 261 | OrgFarm EPIC | 10/3/2025, 7:41 PM |
| Edit Del Security | ExpenseExternalAPI | | 64.0 | Active | 124 | John Manager | 10/9/2025, 5:29 PM |
| Edit Del Security | ExpenseService | | 64.0 | Active | 398 | John Manager | 10/9/2025, 5:08 PM |
| Edit Del | ExpenseServiceTest | | 64.0 | Active | 583 | John Manager | 10/9/2025, 5:18 PM |
| Edit Security | PostInstallScript | devedapp | 64.0 | Active | 2,175 | OrgFarm EPIC | 10/3/2025, 7:41 PM |
| Edit | PostInstallScriptTest | devedapp | 64.0 | Active | 781 | OrgFarm EPIC | 10/3/2025, 7:41 PM |
| Edit Del Security | QueueableExpenseSum | | 64.0 | Active | 112 | John Manager | 10/9/2025, 5:26 PM |
| Edit Del Security | ScheduleOverdueExpenses | | 64.0 | Active | 228 | John Manager | 10/9/2025, 5:24 PM |