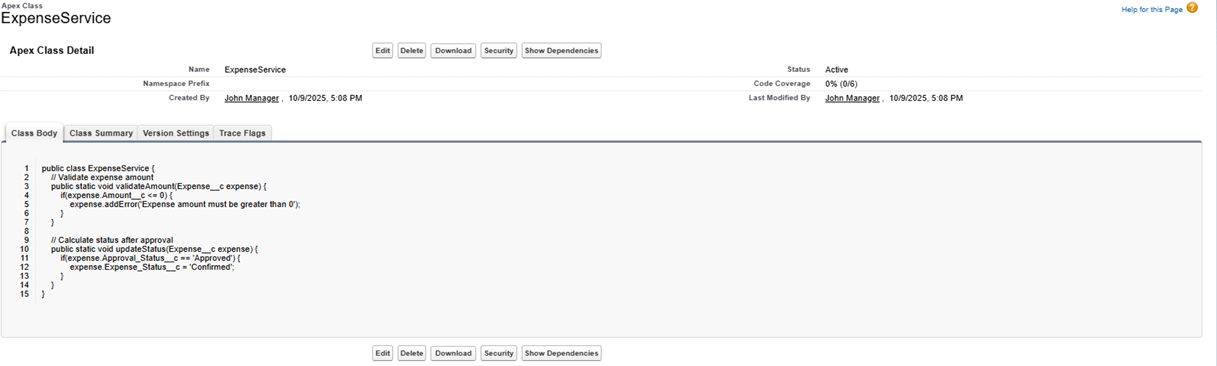
**Expense Tracker Application - Phase 5: Advanced Apex Programming**

**Objective:** To implement advanced business logic, automated batch processing, and maintainability for the Expense Tracker application using Apex, Salesforce's proprietary programming language.

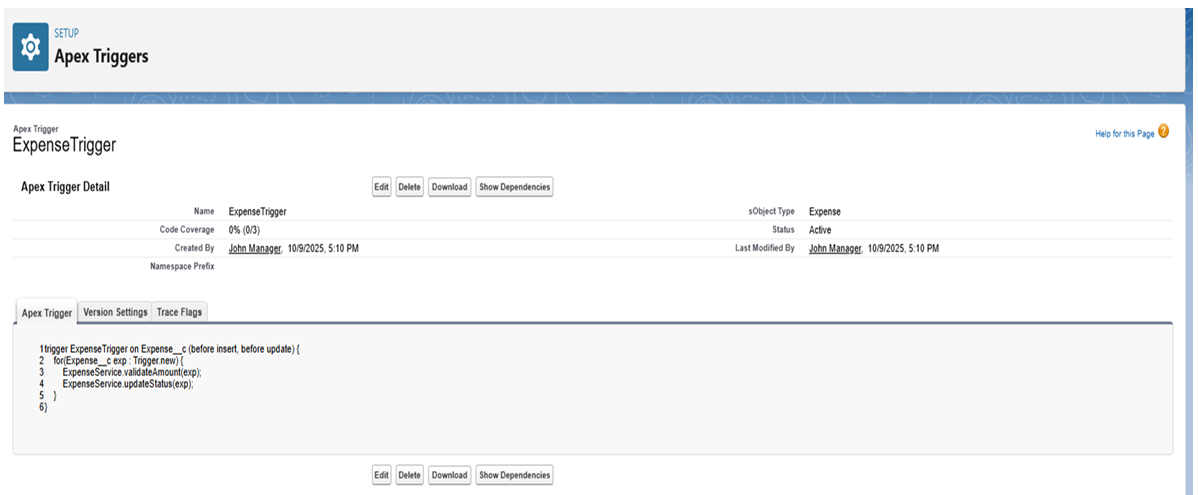
**Step 1: Create an Apex Service Class**

* **Purpose:** To centralize all business logic (such as calculations, status updates, or complex validations) into a single, reusable class. This "Trigger Handler" pattern makes the code more efficient, maintainable, and easier to call from triggers, flows, or other services.
* **Class Name:** ExpenseService
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.



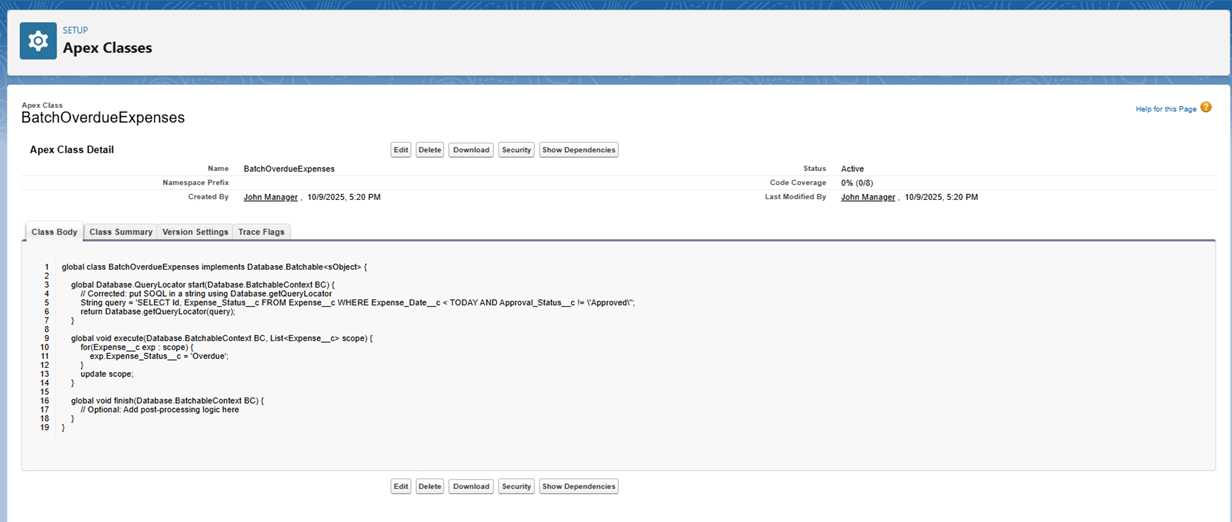
**Step 2: Implement an Apex Trigger**

* **Purpose:** To automatically invoke logic from the ExpenseService class whenever an Expense record is created or updated. The trigger itself remains lightweight, delegating the complex processing to the service class.
* **Trigger Name:** ExpenseTrigger
* **Navigation:** Go to **Setup** → **Object Manager** → **Expense** → **Triggers** → **New**.



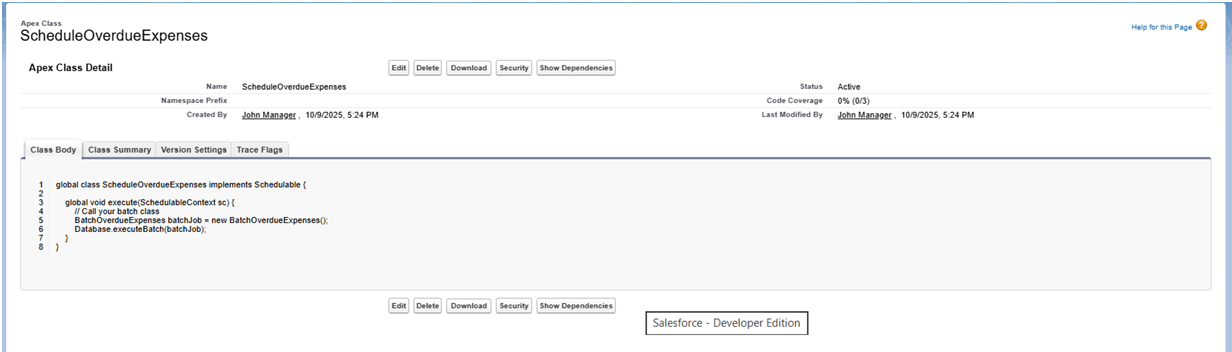
**Step 3: Develop Batch Apex for Overdue Expenses**

* **Purpose:** To process a large number of records asynchronously without hitting governor limits. This batch class is designed to run nightly, find all expense records that are past their due date and still pending approval, and automatically update their status to "Overdue".
* **Class Name:** BatchOverdueExpenses
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.



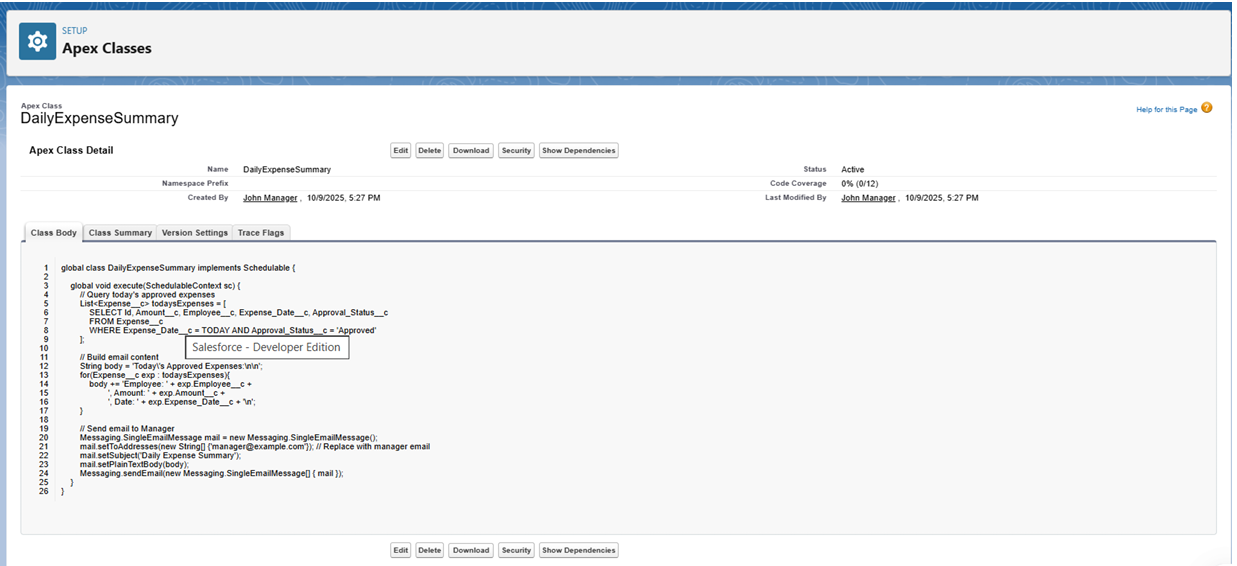
**Step 4: Create a Scheduler Class for the Batch Job**

* **Purpose:** To automatically execute the BatchOverdueExpenses class on a recurring schedule (e.g., daily or weekly). This ensures the "overdue" check runs consistently without manual intervention.
* **Class Name:** ScheduleOverdueExpenses
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.
* **Scheduling:** To activate, go to **Setup** → **Apex Classes** → **Schedule Apex**. Select the ScheduleOverdueExpenses class and set the job frequency, such as **Daily**.



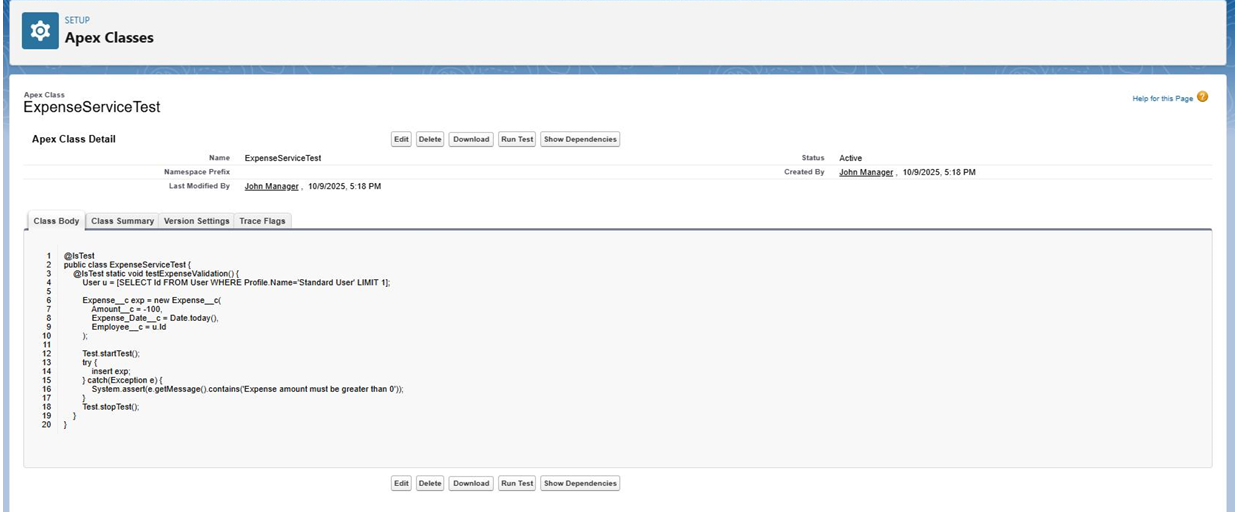
**Step 5: Create Scheduled Apex for a Daily Summary Email**

* **Purpose:** To create a separate scheduled job that queries all expenses approved within the last day and sends a summary email to managers or an finance team.
* **Class Name:** DailyExpenseSummary
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.
* **Scheduling:** This class is also scheduled via **Setup** → **Apex Classes** → **Schedule Apex** , setting the frequency to **Daily**.



**Step 6: Write Apex Test Classes**

* **Purpose:** This is a mandatory step for deploying code to a production environment. Test classes verify that all Apex logic works as expected, handles errors, and meets Salesforce's code coverage requirements.
* **Example Class:** ExpenseServiceTest
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.



**Phase 5 Complete**

The developer-focused tasks for the Expense Tracker application are now complete. The key deliverables for this phase include:

* A centralized ExpenseService class for reusable business logic.
* An ExpenseTrigger to apply logic in real-time during record changes.
* A BatchOverdueExpenses class to handle bulk record updates for overdue items.
* Scheduler classes (ScheduleOverdueExpenses and DailyExpenseSummary) to fully automate batch processing and email notifications.
* Comprehensive Test Classes to ensure code quality and deployment readiness.

