**Expense Tracker Application: Final Project Report**

**Author:** Kothapalli Guna Sekhar Achari **Date:** October 24, 2025 **Project:** A comprehensive, 10-phase project to build, automate, and deploy an enterprise-grade expense management application on the Salesforce platform.

**Table of Contents**

1. **Executive Summary**
2. **Phase 1: Project Foundation & Domain Analysis**
3. **Phase 2: Salesforce Org Setup and Configuration**
4. **Phase 3: Data Modeling and UI Configuration**
5. **Phase 4: Business Process Automation**
6. **Phase 5: Advanced Apex Programming**
7. **Phase 6: Lightning UI and App Development**
8. **Phase 7: Integration and External Access**
9. **Phase 8: Data Management and Deployment**
10. **Phase 9: Reporting, Dashboards, and Security Review**
11. **Phase 10: Go-Live, Training, and Project Handoff**

**Executive Summary**

This document details the complete lifecycle of the **Automated Expense Reporter** project. The project's mission was to solve the inefficiencies of manual expense reporting by building a unified, automated, and secure application on the Salesforce platform.

The project began with foundational analysis, defining stakeholders, and mapping business processes. This was followed by the core build, which included establishing a secure Salesforce org , architecting the custom Expense\_\_c data model , and building automations using **Validation Rules**, **Flows**, and **Approval Processes** to enforce data integrity and business logic.

For advanced functionality, the project leveraged **Apex programming** to create triggers, service classes, and scheduled batch jobs for nightly processing. A modern user experience was constructed using a **Lightning App**, custom **Lightning Web Components (LWCs)**, and intuitive record pages.

The application's capabilities were extended through integrations with external APIs using **Named Credentials**. The project also established robust data management procedures, including data loading, duplicate prevention, and a full deployment plan using **Change Sets**.

Finally, the project delivered actionable insights through real-time **reports and dashboards** and was successfully deployed to end-users with comprehensive training. The result is a complete, end-to-end solution that reduces data entry errors, shortens reimbursement cycles, and provides full visibility into corporate spending.

**Phase 1: Project Foundation & Domain Analysis**

**1. Project Mission Statement**

Modern organizations require a dynamic and unified platform to streamline the submission, tracking, and reconciliation of employee-initiated business expenses. Traditional, often manual, expense reporting methods are prone to inefficiencies, including data entry errors, protracted reimbursement cycles, and a critical lack of real-time visibility into corporate expenditure.

This Automated Expense Reporter application is designed to:

* Empower employees with an intuitive interface for quick and accurate expense logging against predefined cost centers.
* Equip managers with a streamlined workflow for reviewing, approving, or rejecting expense claims in a timely manner.
* Deliver actionable insights to leadership through a real-time analytics dashboard that visualizes spending patterns and budget adherence.
* Enforce corporate spending policies automatically through configurable validation and approval rules.
* Enhance overall fiscal governance by improving budget tracking, financial transparency, and strategic cost management.

**2. Functional & Technical Requirements**

* Data Model: The system will be built upon three core custom objects: Employee, Expense Report, and Expense Category.
* Submission Process: Employees can create an Expense Report record, attaching line items with details such as amount, date, category, and a digital copy of the receipt.
* Approval Automation: A multi-tiered approval process will be triggered upon submission, routing the expense report to the appropriate manager for action.
* Data Integrity: The system will use validation rules to maintain data accuracy, such as preventing the submission of expenses with future dates or negative monetary values.
* Security Framework: Access will be governed by a role-based security model:
  + Employee Profile: Can create, edit, and submit their own expense reports.
  + Manager Profile: Can view and act upon (approve/reject) reports submitted by their direct reports.
  + Administrator Profile: Has comprehensive access to configure and manage the system.
* Analytics & Reporting: The application will feature a comprehensive dashboard visualizing key metrics, including monthly expenditure by category and departmental spending trends. Custom reports will be available for the finance team to conduct in-depth analysis.

**3. Stakeholder Matrix**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| | Stakeholder Group | Primary Interest/Role | Key Requirements | | --- | --- | --- | | General Employees | Submitting expense reports for reimbursement. | A simple, mobile-friendly interface for creating reports and uploading receipts effortlessly. | | Line Managers | Reviewing and validating team expenditures. | Automated notifications and a centralized queue for efficient approvals or rejections. | | Finance Department | Overseeing financial compliance and budget allocation. | Granular reports, data export capabilities, and dashboards to monitor spending trends. | | IT/System Admins | Ensuring system integrity, security, and user management. | Tools for user provisioning, permission set configuration, and system maintenance. | | Executive Leadership | Strategic oversight of organizational spending and cost control. | High-level dashboards providing insights into quarterly and annual expense data. | |

**4. Business Process Flow**

The end-to-end workflow is as follows:

1. Initiation: An employee creates a new expense report, adds expense line items with required details (category, amount, etc.), and submits it for review.
2. Managerial Review: The system automatically assigns the report to the employee's designated manager for approval.
3. Decision Point:
   * If Approved: The expense data is integrated into the financial dashboards and reports for tracking.
   * If Rejected: The report is returned to the employee's queue with comments, allowing for correction and resubmission.
4. Data Aggregation: Approved expenses are compiled in real-time to update dashboards that summarize spending by department, category, and time period.

**5. Strategic Use Case Analysis**

* Corporate T&E Management: Efficiently track and manage travel, meals, and other corporate expenses.
* Policy Enforcement & Compliance: Automate compliance with internal spending policies and external regulations.
* Audit Trail & Readiness: Maintain a clear, immutable record of all expenses, submissions, and approvals for audit purposes.
* Dynamic Budget Management: Provide teams with real-time visibility into their spending against allocated budgets.
* Enterprise Scalability: Design the system to accommodate a growing workforce and increasing transaction volume without a proportional increase in manual administration.

**6. AppExchange Synergy Exploration**

To enhance functionality and accelerate development, the following categories of AppExchange solutions will be evaluated:

* Expense Automation Solutions: Pre-built packages that handle core expense submission and approval logic.
* Advanced Workflow Tools: Applications that can enhance the standard approval process with features like parallel approvals and dynamic reminders.
* OCR & Document Management: Solutions for automatically scanning receipts and handling secure file storage.
* Business Intelligence Platforms: Advanced analytics tools for creating more sophisticated and interactive expense dashboards.
* Governance & Compliance Frameworks: Apps designed to help log, monitor, and report on system activity to ensure compliance.

**Expense Tracker Project - Phase 2: Salesforce Org Setup and Configuration**

**Objective:** This phase details the essential setup steps required to prepare a Salesforce developer environment for the Expense Tracker project.

**Step 1: Access Your Developer Organization**

* **Action:** Log in to your Salesforce Developer Edition org using your System Administrator credentials. If you do not have one, you will need to sign up for one.
* **Purpose:** This provides a free, sandboxed environment to build and test the application without impacting a live system.

**Step 2: Configure Company Information**

1. **Navigate:** Go to Setup, use the Quick Find box to search for "Company Information," and click Edit.
2. **Settings:**
   * Set the **Company Name** to your project's name.
   * Adjust the **Default Time Zone** (e.g., GMT+05:30 India Standard Time).
   * Select the **Default Currency** (e.g., INR or USD).
3. **Save** your changes.

* **Purpose:** This configuration ensures that all date, time, and currency fields display correctly throughout the org.

**Step 3: Set Fiscal Year**

1. **Navigate:** Go to Setup, then use Quick Find to search for "Fiscal Year".
2. **Action:** Confirm that the "Standard Fiscal Year" (January-December) is selected.
3. **Save** if you made any modifications.

* **Purpose:** Establishes standard reporting periods, which is crucial for future expense and revenue reporting.

**Step 4: Establish Role Hierarchy**

1. **Navigate:** Go to Setup, use Quick Find to search for "Roles," and select "Set Up Roles".
2. **Create Roles:**
   * Define a top-level role, such as "Manager".
   * Create a child role under "Manager," such as "Employee/Agent".
3. **Save** the new hierarchy.

* **Purpose:** The role hierarchy is essential for record visibility. It allows managers to view records owned by their subordinates when Organization-Wide Defaults (OWD) are set to Private.

**Step 5: Create Project Users**

1. **Navigate:** Go to Setup, use Quick Find to search for "Users," and click "New User".
2. **Create Manager User:**
   * **Profile:** System Administrator (This is the fastest for demonstration purposes).
   * **Role:** Manager.
   * Complete all required fields: Name, Email, Username, Alias, Time Zone, and Locale.
   * Click **Save**.
3. **Create Employee User:**
   * **Profile:** Standard User.
   * **Role:** Employee.
   * Complete all required fields: Name, Email, Username, Alias, Time Zone, and Locale.
   * Click **Save**.
4. **Update Employee Record:** After saving the Employee user, you must **edit their record** and set the "Manager" lookup field to point to the Manager user you created. Save the record again.

* **Purpose:** This structure is critical for role-based visibility and approval processes, which rely on the Manager link and the defined role hierarchy.

**Step 6: Clone the Standard Profile**

1. **Navigate:** Go to Setup, use Quick Find for "Profiles." Select the "Standard User" profile and click "Clone".
2. **New Profile Name:** Enter "Expense Employee Profile".
3. **Save** the new profile.

* **Purpose:** This creates a custom profile for employees, allowing you to set minimal permissions without altering the default Standard User profile.

**Step 7: Configure Org-Wide Defaults (OWD)**

1. **Navigate:** Go to Setup, use Quick Find for "Sharing Settings," and click "Edit".
2. **Set:** Find the "Expense\_\_c" object and set its default access to **Private**. (Note: This object must be created first).
3. **Save** the changes.

* **Purpose:** Setting the OWD to Private ensures that, by default, only the record owner and their manager (via the role hierarchy) can view an expense record.

**Step 8: Set Profile and Field-Level Security**

*(This step should be performed after the Expense custom object and its fields have been created)*

* **Step 8A: Field-Level Security (FLS)**
  + **Navigate:** Go to Setup → Object Manager → Expense → Fields & Relationships.
  + **Action:** For each field, click its name, select "Set Field-Level Security," and ensure the "Visible" checkbox is checked for the "Expense Employee Profile" and the Manager/Admin profile.
* **Step 8B: Profile Object Access**
  + **Navigate:** Go to Setup → Profiles → Expense Employee Profile.
  + **Action:** Find "Object Settings" → "Expense" → "Edit".
  + **Permissions:** Grant Read, Create, and Edit access (Delete is optional). Click **Save**.
* **Purpose:** This configuration ensures that employees have the necessary permissions to create and manage their own expense records , while managers can review them.

**Step 9: Quick Validation Test**

1. Log in to Salesforce as the "Employee User" and create a new "Expense" record.
2. Log out, then log in as the "Manager User." Verify that you can see the expense record created by the employee.
3. **Confirm:** This test validates that the role hierarchy and OWD settings are working correctly.

* **Purpose:** This validation confirms the security and sharing setup before proceeding to Phase 3 (object creation and relationships).

**Phase 2 Completion**

* The Salesforce organization is now minimally configured for the Expense Tracker project.
* You are now ready to proceed to Phase 3: Creating objects, fields, and relationships.

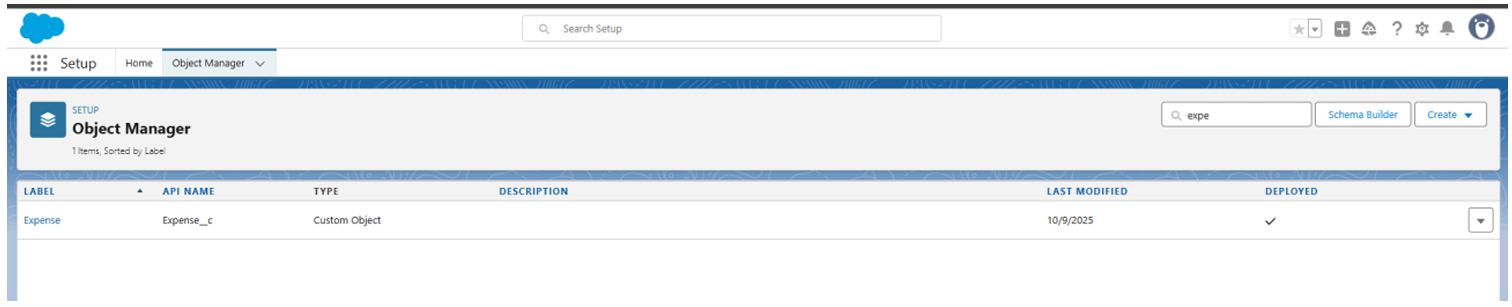
**Expense Tracker Application Phase - 3: Data Modeling and UI Configuration**

**Objective:** To construct the core data architecture for the Expense Tracker application. This phase covers the creation of custom objects, fields, and relationships, as well as the configuration of user interface elements like page layouts and record types.

**Step 1: Create the 'Expense' Custom Object**

First, we will create the central object that will store all expense records.

1. **Navigate:** Go to **Setup → Object Manager**, then click **Create → Custom Object**.
2. **Enter Details:**
   * **Label:** Expense
   * **Plural Label:** Expenses
   * **Object Name:** Expense (The API name will automatically become Expense\_\_c)
3. **Configure Record Name:**
   * **Data Type:** Select **Auto Number**.
   * **Display Format:** EXP-{0000}
   * **Starting Number:** 1
   * **Purpose:** This creates a unique, user-friendly identifier for every expense record (e.g., EXP-0001).
4. **Save** the new object.

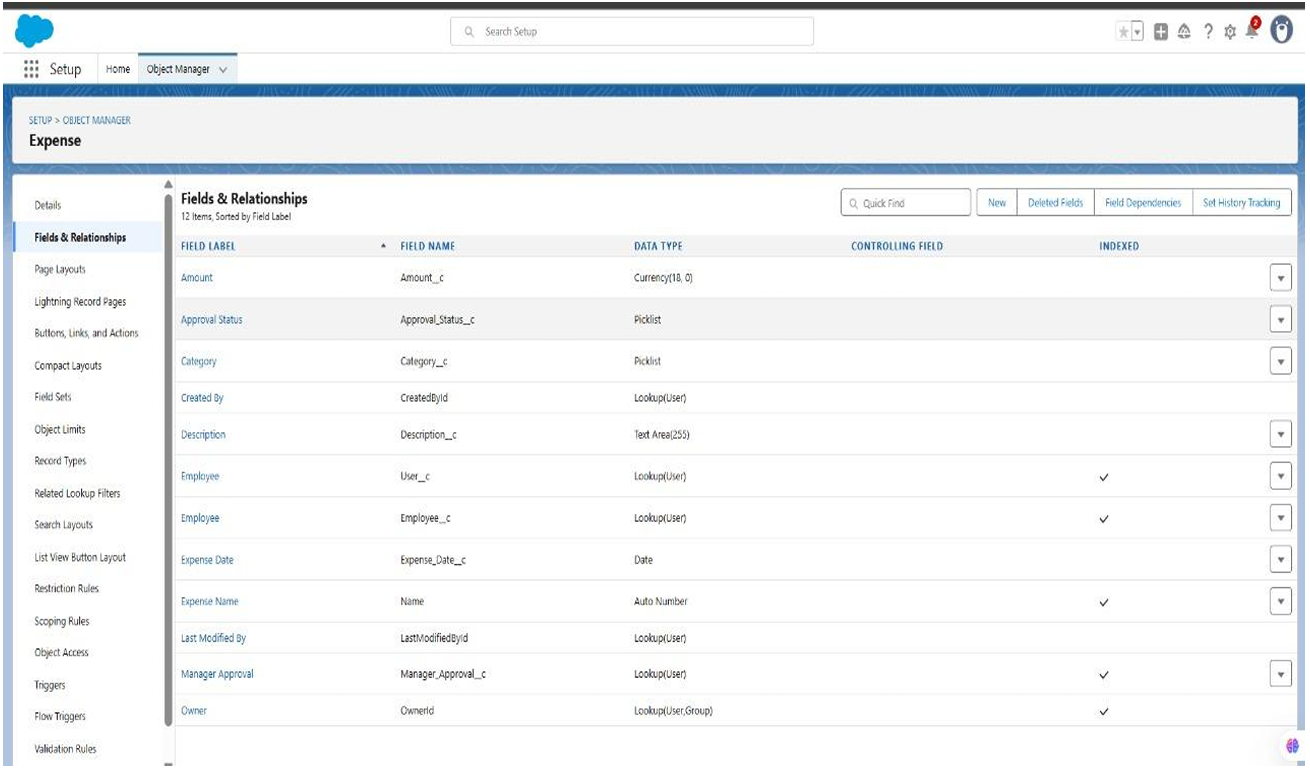


**Step 2: Add Custom Fields to the 'Expense' Object**

Next, we will add the necessary fields to capture expense details.

1. **Navigate:** From the **Object Manager**, select **Expense**, then go to **Fields & Relationships** and click **New**.
2. **Create the following fields:**

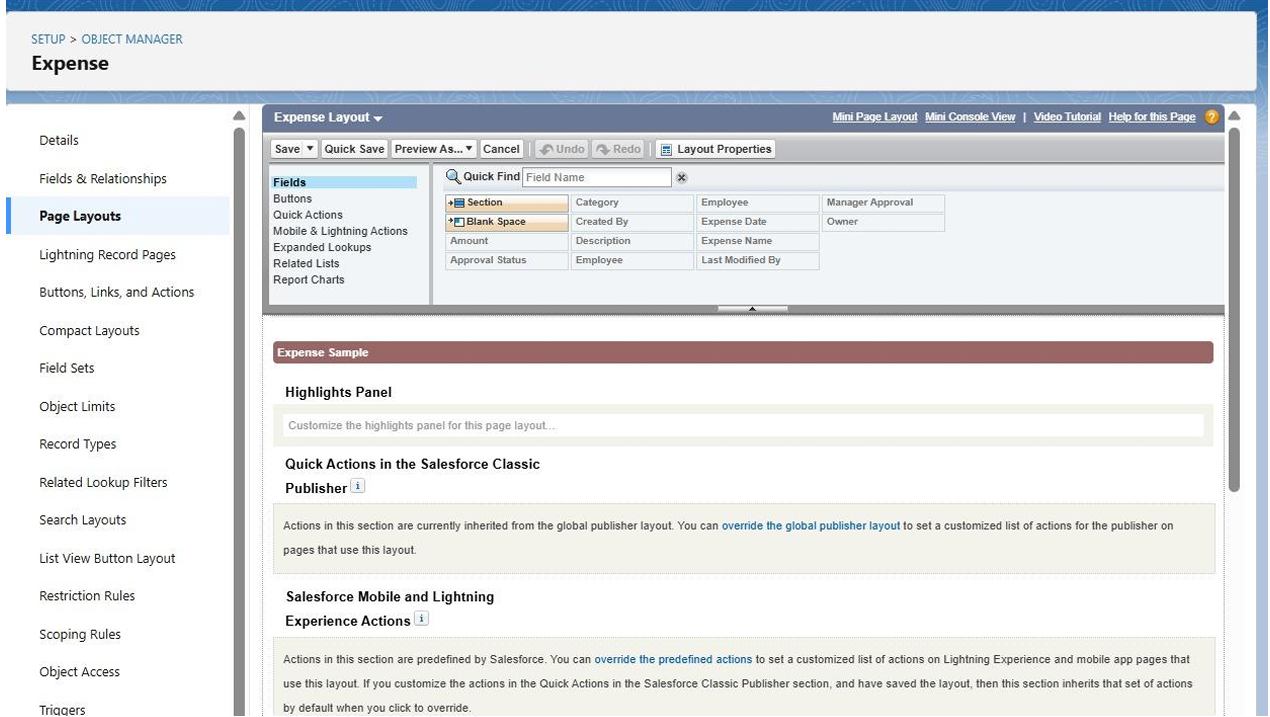
|  |  |  |
| --- | --- | --- |
| Field Label | Data Type | Details |
| **Amount** | Currency | Mark this field as **Required**. |
| **Expense Date** | Date | Mark this field as **Required**. |
| **Category** | Picklist | Enter the values: Travel, Food, Other. |
| **Description** | Text Area | This field is optional for additional notes. |
| **Approval Status** | Picklist | Enter the values: Pending, Approved, Rejected. |
| **Employee** | Lookup(User) | Mark this field as **Required**. This links the expense to a user record. |
| **Manager Approval** | Lookup(User) | This optional field can be used later for approval processes. |



**Step 3: Configure Page and Compact Layouts**

Now, we will organize the fields on the user interface for optimal usability.

1. **Page Layout:**
   * **Navigate:** From the **Expense** object page, go to **Page Layouts** and **Edit** the default layout.
   * **Action:** Drag and drop the following fields onto the layout in a logical order: Expense Number, Amount, Expense Date, Category, Employee, Description, and Approval Status.
   * **Save** the layout.
2. **Compact Layout:**
   * **Navigate:** From the **Expense** object page, go to **Compact Layouts** and click **New**.
   * **Action:** Add the following key fields: Expense Number, Amount, Expense Date, and Approval Status.
   * **Save** the layout and then use **Compact Layout Assignment** to assign it to all profiles.
   * **Purpose:** The compact layout controls which fields are displayed in highlights, mobile views, and list views for quick reference.



**Step 4: (Optional) Implement Record Types**

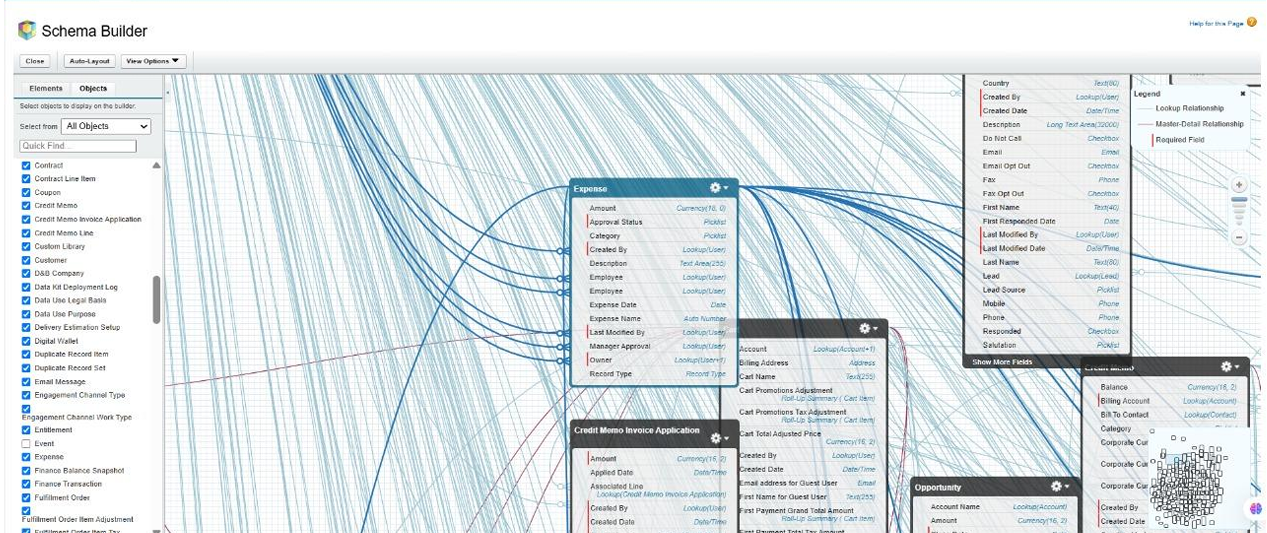
Record types allow for different business processes and page layouts for various kinds of expenses.

1. **Navigate:** From the **Expense** object page, go to **Record Types** and click **New**.
2. **Create Record Types:** Set up distinct record types based on the Category picklist, such as Travel Expense, Food Expense, and Other Expense.
3. **Assign:** Make these record types available to the Expense Employee Profile and Manager/Admin profiles. You can assign the same page layout to all for now.

**Step 5: Verify the Data Model with Schema Builder**

Use the Schema Builder to visually confirm the relationship you created.

1. **Navigate:** Go to **Setup → Schema Builder**.
2. **Action:** In the object list, find and select the Expense and User objects to display them on the canvas.
3. **Verify:** Confirm that a lookup relationship line connects the Expense object (via the Employee field) to the User object.



**Step 6: Finalize Security and Access Settings**

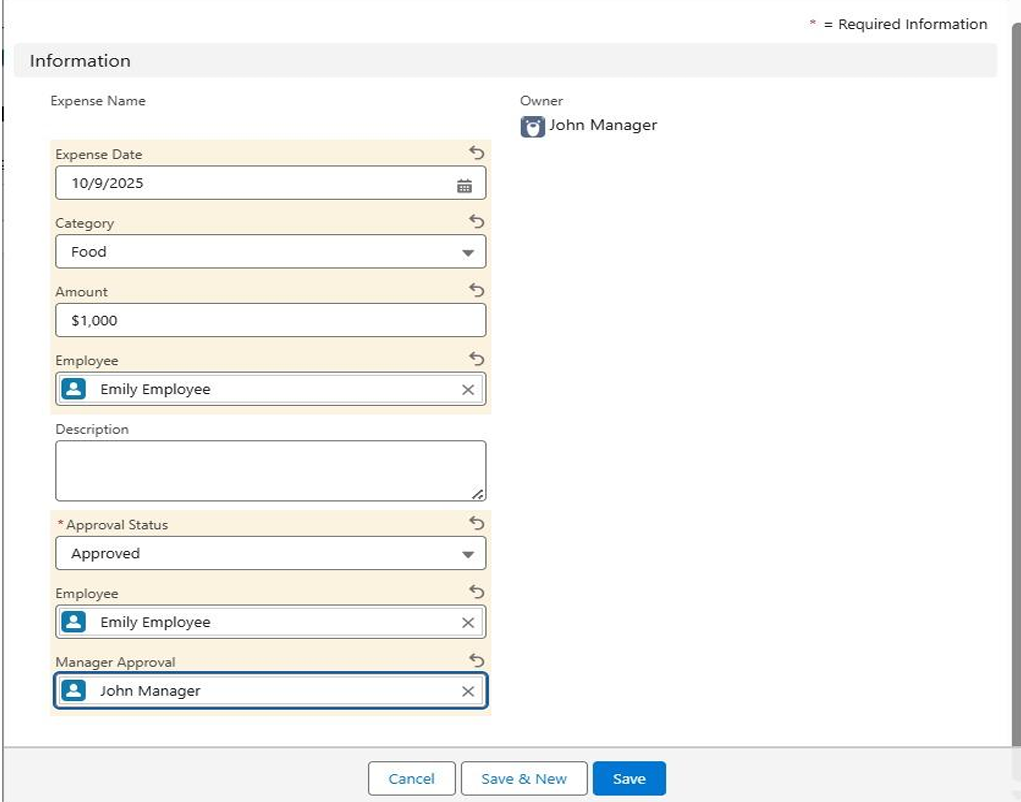
Ensure the correct profiles can see and edit the new fields and object.

1. **Field-Level Security (FLS):**
   * **Navigate:** Go to **Setup → Profiles → Expense Employee Profile**.
   * **Action:** Under **Object Settings → Expense**, ensure users have **Read**, **Create**, and **Edit** access. Verify that all the new fields are visible. Repeat this process for the Manager/Admin profile, granting them appropriate access.
2. **Save** any changes to the profiles.

**Step 7: Conduct a Validation Test**

Finally, test the configuration to ensure everything works as expected.

1. **Log in as the Employee user.** Navigate to the Expenses tab and create a new expense record. Confirm you can fill out all fields and save the record.
2. **Log in as the Manager user.** Verify that you can view the record created by the employee.
3. **Negative Test:** Log in as a different employee user (if one exists) and confirm that they **cannot** see the expense record created by the first employee, validating your sharing settings.



**Phase 3 Completion:** The data model for the Expense object is now established, user interfaces are configured, and security settings have been verified. The application is ready for the next phase of development, such as building automation and reports.

**Expense Tracker Application - Phase 4: Business Process Automation**

**Objective:** To implement the business logic and automation required for a functional expense management process. This phase focuses on ensuring data integrity with validation rules, automating the manager approval process, and creating a guided user experience with flows.

**Step 1: Implement Validation Rules**

First, we will create rules to ensure high-quality data is entered *before* a record is saved.

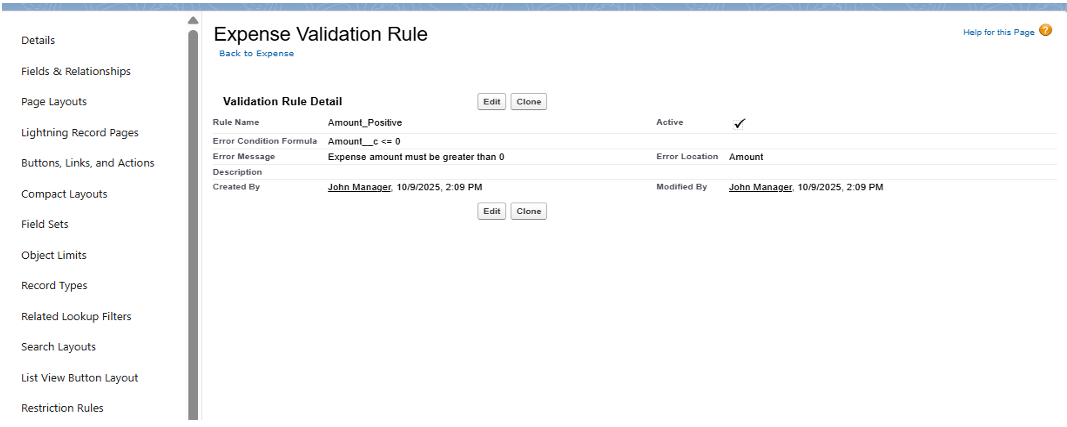
* **Purpose:** To enforce data quality standards, such as ensuring all expense amounts are positive and dates are not set in the future.
* **Navigation:** Go to **Setup** → **Object Manager** → **Expense** → **Validation Rules** → **New**.

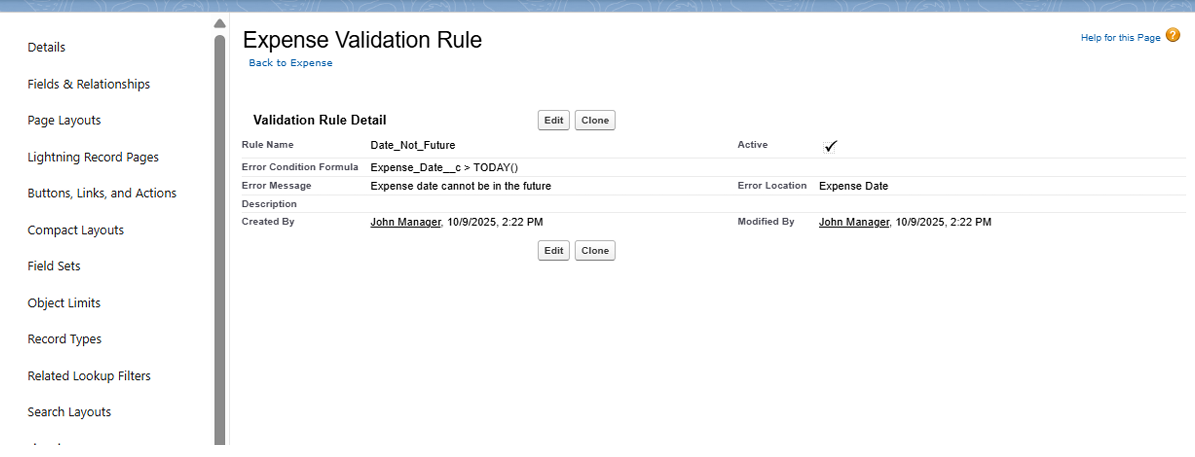
**Rule A: Ensure Positive Amount**

* **Rule Name:** Amount\_Positive
* **Error Condition Formula:** Amount\_\_c <= 0
* **Error Message:** "Expense amount must be greater than 0"
* **Error Location:** Select the Amount field.

**Rule B: Prevent Future Expense Dates**

* **Rule Name:** Date\_Not\_Future
* **Error Condition Formula:** Expense\_Date\_\_c > TODAY()
* **Error Message:** "Expense date cannot be in the future"
* **Error Location:** Select the Expense Date field.



****

**Step 2: Create a Guided Submission with a Screen Flow**

We will create a user-friendly wizard to guide employees through submitting a new expense.

* **Purpose:** To provide a simple, guided form for employees, ensuring all required information is captured in a sequential flow.
* **Navigation:** Go to **Setup** → **Flow** → **New Flow** → **Screen Flow**.

**Flow Structure:**

1. **Screen Element:** Create a screen titled "Submit Expense Form".
   * Add input components for Amount (Currency, Required) , Expense Date (Date, Required) , Category (Picklist) , and Description (Text Area).
2. **Create Records Element:** Add this element after the screen.
   * **Label:** "Create Expense Record".
   * **How to Set Record Fields:** Select "Use separate resources and literal values."
   * **Map Fields:** Map the screen components to the fields on your Expense\_\_c object.
   * **Set Employee:** Critically, map the Employee\_\_c field to the global variable \$User.Id. This automatically stamps the record with the current user's ID.
3. **Connections:** Connect the **Start** element to the **Screen**, and the **Screen** to the **Create Records** element.
4. **Save and Activate** the flow.

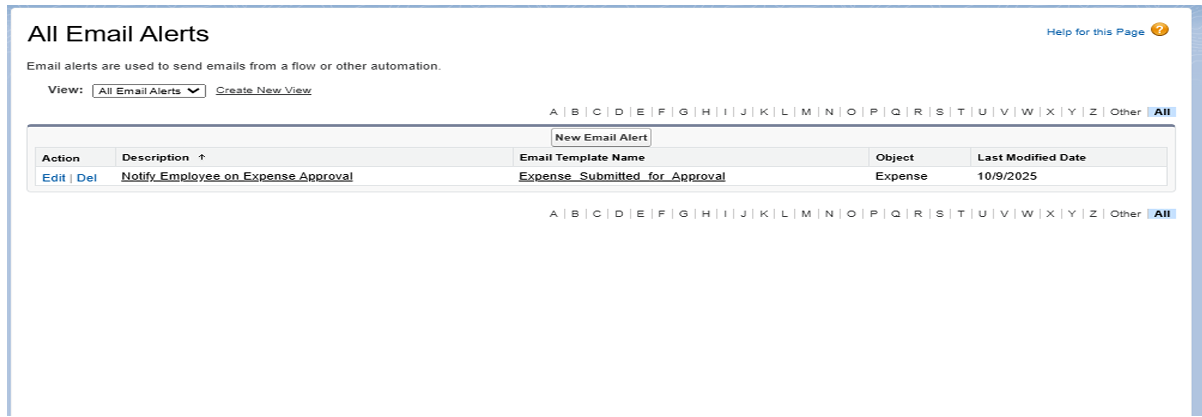
**Step 3: Build the Manager Approval Process**

This process will automatically route expenses for approval based on the amount.

* **Purpose:** To ensure that expenses exceeding a certain monetary threshold receive manager approval before being processed.
* **Navigation:** Go to **Setup** → **Approval Processes** → Select the **Expense** object → **Create New Approval Process** → **Use Standard Setup Wizard**.

**Configuration:**

* **Name:** Expense Manager Approval.
* **Entry Criteria:** Set the criteria to trigger this process, for example: Amount\_\_c > 5000.
* **Approver:** Choose "Manager" as the approver. The system will automatically use the Manager field on the submitter's User record.
* **Approval Actions:** Add a **Field Update** action to change the Approval\_Status\_\_c field to 'Approved'.
* **Rejection Actions:** Add a **Field Update** action to change the Approval\_Status\_\_c field to 'Rejected'.
* **Email Alerts:** (Optional) You can configure email alerts here to notify the manager of a new request and the submitter of the outcome.
* **Activate:** Once all steps are configured, **Activate** the approval process.



**Step 4: Automate Status Updates with a Record-Triggered Flow**

This flow will run in the background to update a record *after* it has been approved.

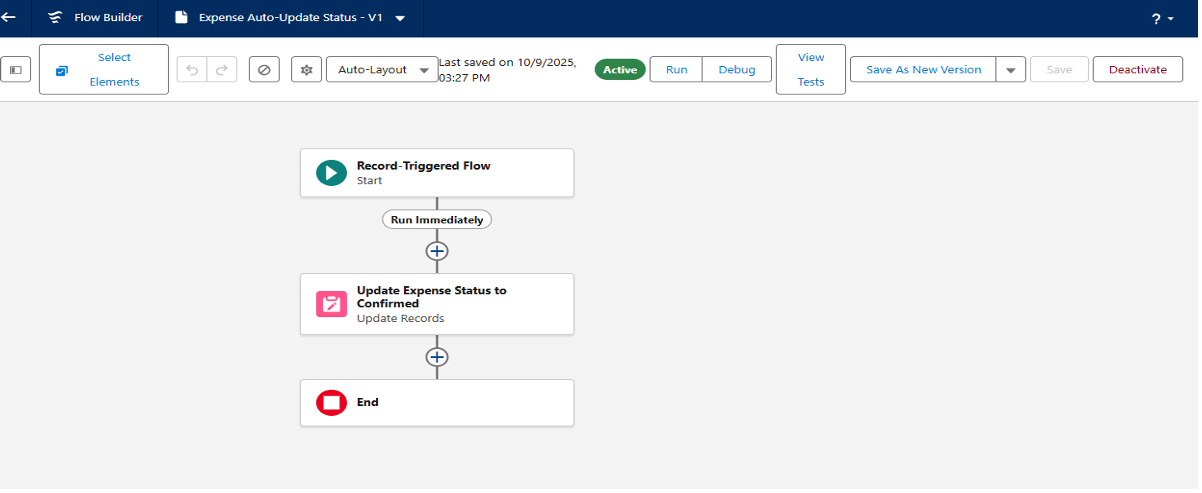
* **Purpose:** To automatically perform a final action, such as updating a "final" status to 'Confirmed' once the approval process is complete.
* **Navigation:** Go to **Setup** → **Flow** → **New Flow** → **Record-Triggered Flow**.

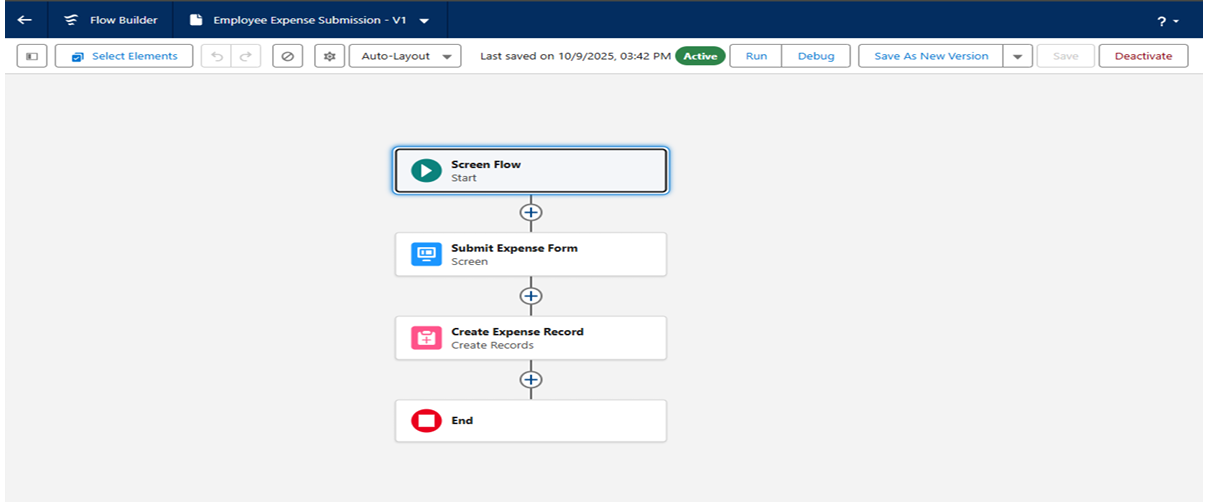
**Trigger Configuration:**

* **Object:** Expense.
* **Trigger:** Select "A record is created or updated".
* **Condition Requirements:** Set the condition for the flow to run: Approval\_Status\_\_c equals 'Approved'.
* **Optimize for:** Select "Actions and Related Records."

**Flow Element:**

1. Add an **Update Records** element to the flow.
2. **Label:** "Update Expense Status to Confirmed".
3. **How to Find Records:** Select "Use the expense record that triggered the flow".
4. **Set Field Values:** Set a field (e.g., a new field named Expense\_Status\_\_c) to 'Confirmed'.
5. **Save and Activate** the flow.

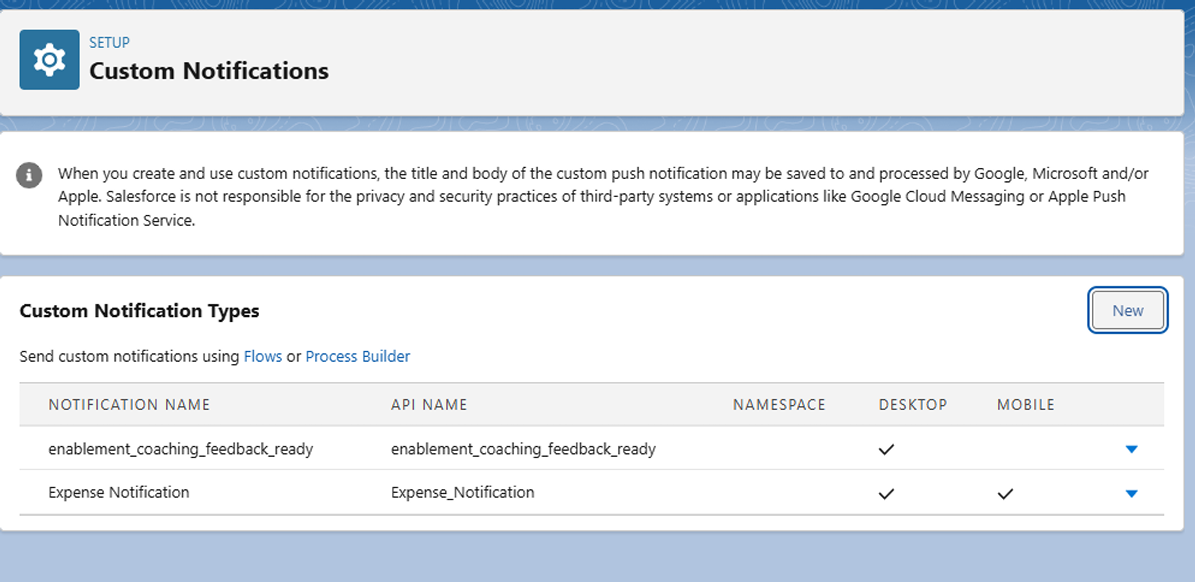




**Step 5: Configure In-App Notifications**

Finally, we will set up in-app (bell icon) notifications to alert users instantly.

* **Purpose:** To notify employees or managers directly within the Salesforce app when an action occurs, such as an approval.
* **Navigation (Part 1):** Go to **Setup** → **Notification Builder** → **Custom Notifications** → **New**.
  + **Name:** Expense Notification.
  + **Supported Channels:** Check **Desktop** and **Mobile**.
  + Save.
* **Navigation (Part 2):** Add this action to your **Record-Triggered Flow** (from Step 4).
  + Add an **Action** element to your flow.
  + **Action:** Search for "Send Custom Notification".
  + **Notification Type:** Select the Expense Notification you just created.
  + **Recipient:** Set the recipient (e.g., the record's owner \$Record.OwnerId or manager).
  + **Message:** Create a dynamic message, such as "Your expense {$Record.Name} has been approved".



**Phase 4 Complete**

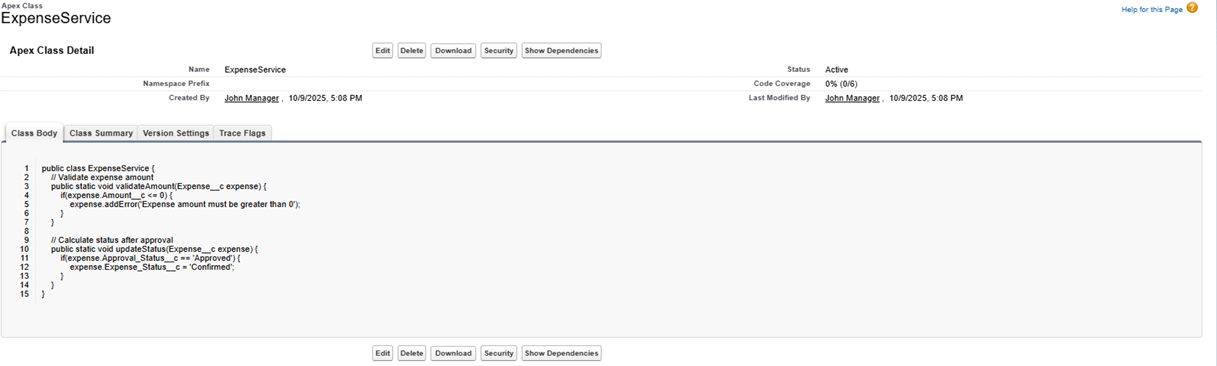
With these automations, the Expense Tracker application is now intelligent. Employees are guided through a Screen Flow , data is validated, approvals are automatically routed, statuses are updated, and all relevant users are notified of the changes.

**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**.



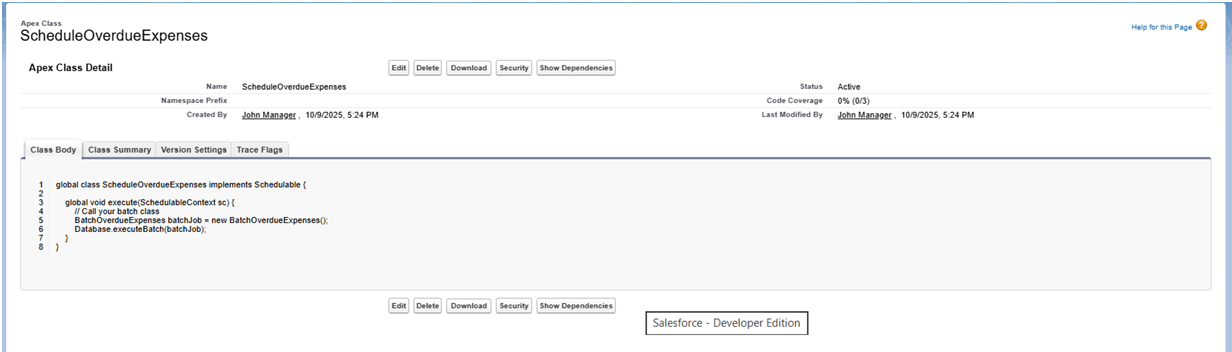
**Step 3: Develop Batch Apex for Overdue Expenses**

* **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**.



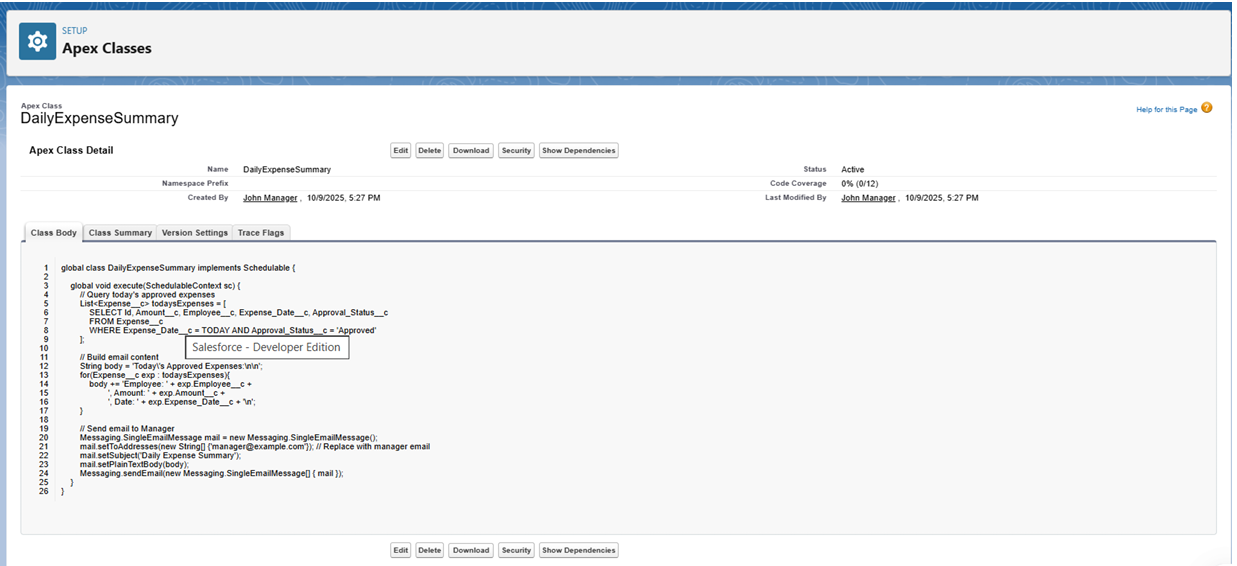
**Step 4: Create a Scheduler Class for the Batch Job**

* **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**.



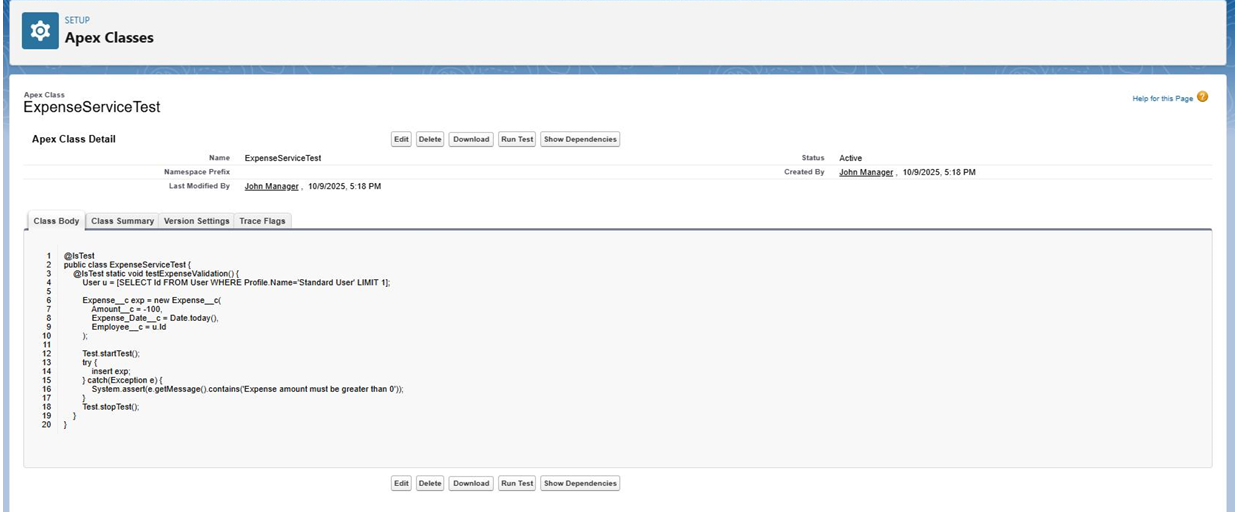
**Step 5: Create Scheduled Apex for a Daily Summary Email**

* **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.



**Expense Tracker Application - Phase 6: Lightning UI and App Development**

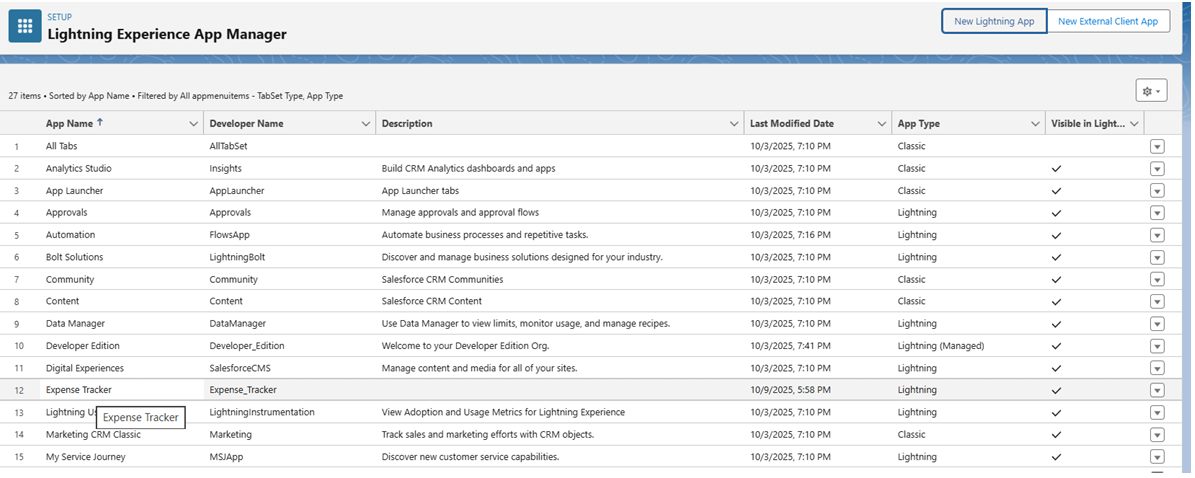
**Objective:** To construct a modern, user-friendly, and cohesive user interface for the Expense Tracker. This phase involves creating a dedicated Lightning App, designing custom record and home pages, and building interactive Lightning Web Components (LWC) for enhanced functionality.

**Step 1: Create the 'Expense Tracker' Lightning App**

* **Purpose:** To group all project-related tabs, pages, and tools into a single, branded application for easy user access.
* **Navigation:** Go to **Setup** → **App Manager** → **New Lightning App**.

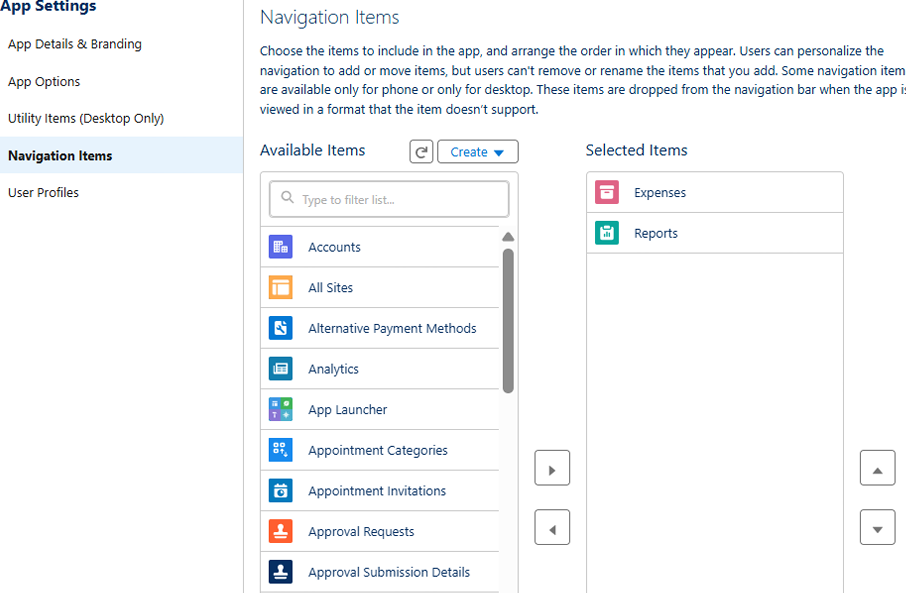
**Configuration:**

1. **App Details:** Enter the **App Name** as Expense Tracker and upload a logo (optional).
2. **Navigation Items:** Add the Expenses tab and the Reports tab to the "Selected Items" list.
3. **Utility Bar (Optional):** Add a utility item, such as the New Expense Quick Action, to allow users to create records from anywhere in the app.
4. **User Profiles:** Assign the app to the relevant user profiles (e.g., Expense Employee, Manager, System Administrator).
5. **Save & Finish**.



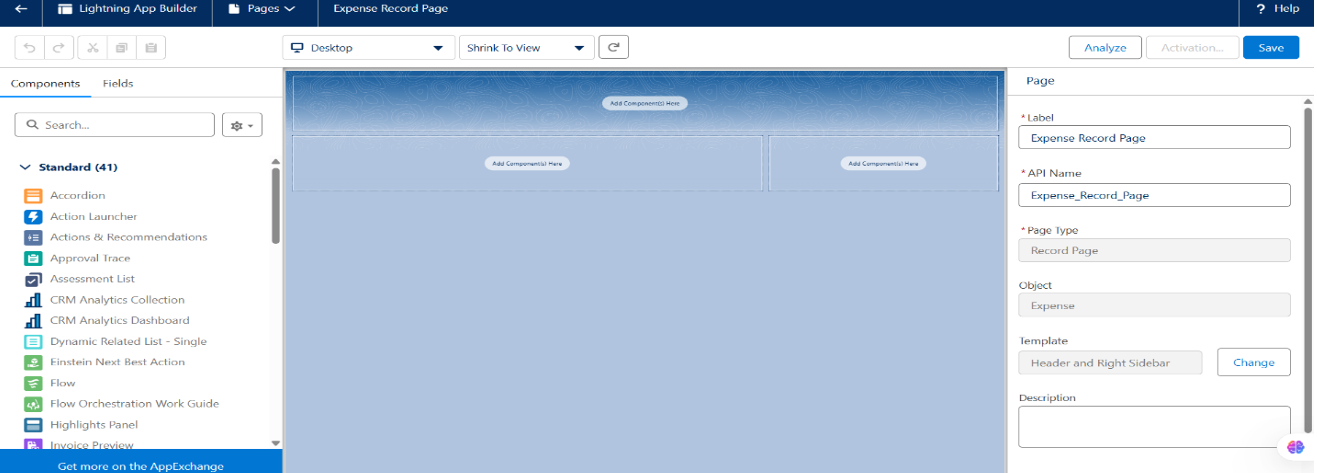
**Step 2: Design the 'Expense' Lightning Record Page**

* **Purpose:** To create an efficient and informative layout for viewing a single expense record, combining key details, related lists, and actions.
* **Navigation:** Go to **Setup** → **Object Manager** → **Expense** → **Lightning Record Pages** → **New** (or Edit existing).



**Layout Configuration:**

1. **Template:** Select a template, such as Header + Right Sidebar.
2. **Main Region:** Drag the **Record Detail** component into the main section. This will display the fields arranged in the page layout, such as:
   * Expense Number
   * Amount
   * Expense Date
   * Category
   * Employee
   * Approval Status
3. **Sidebar:** Drag the **Related Lists** component into the sidebar. Include lists like Approval History and Tasks (if applicable).
4. **Highlights Panel:** Configure the header to show key actions like Submit for Approval and Edit.
5. **Activation:** Click **Activate** and assign the page as the **Org Default** for the Expense object.

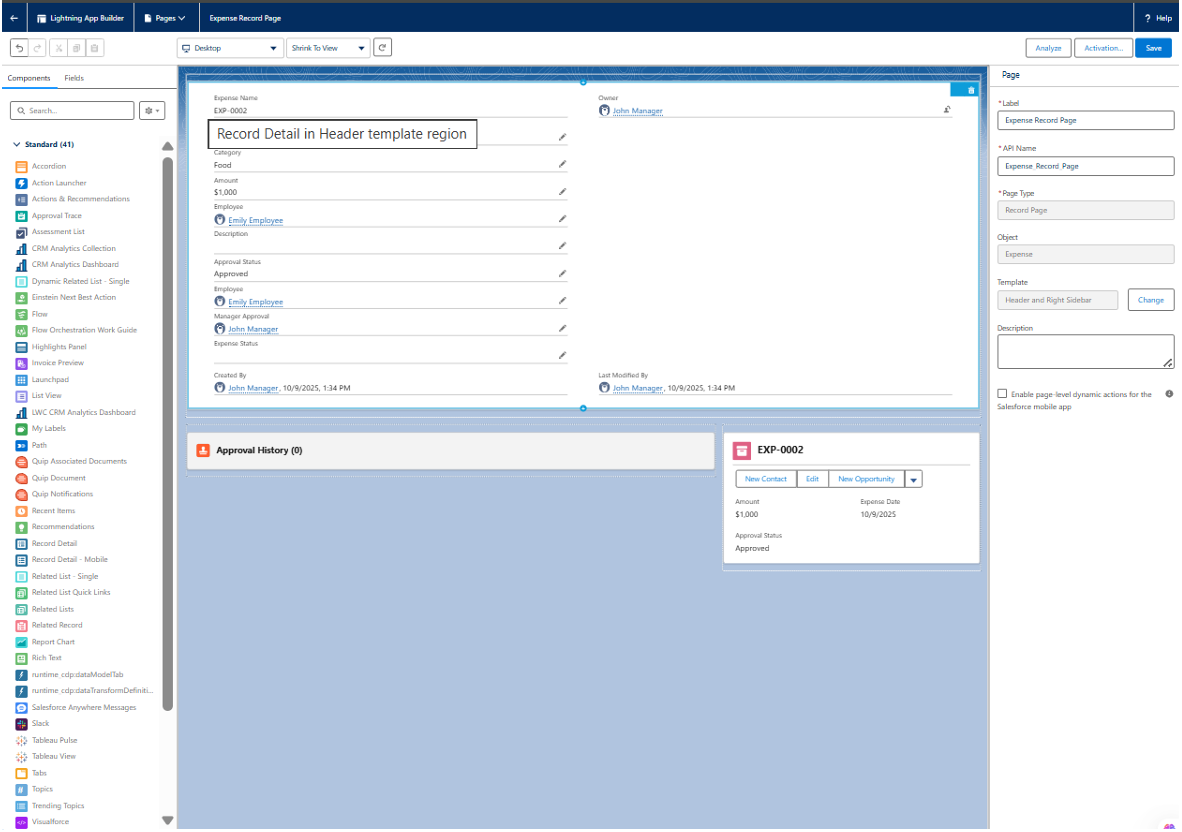


**Step 3: Customize the App Home Page**

* **Purpose:** To provide users with a high-level dashboard summarizing key expense data immediately upon opening the app.
* **Navigation:** Go to **Setup** → **Lightning App Builder** → **New** → **Home Page** (or Edit existing).

**Component Configuration:**

1. Add **Dashboard** or **Report Chart** components to visualize data like Pending Approvals or Total Expenses.
2. Add a **List View** component configured to show Recent Expenses.
3. **Save** and **Activate** the page, assigning it as the default Home Page for the Expense Tracker app and its user profiles.

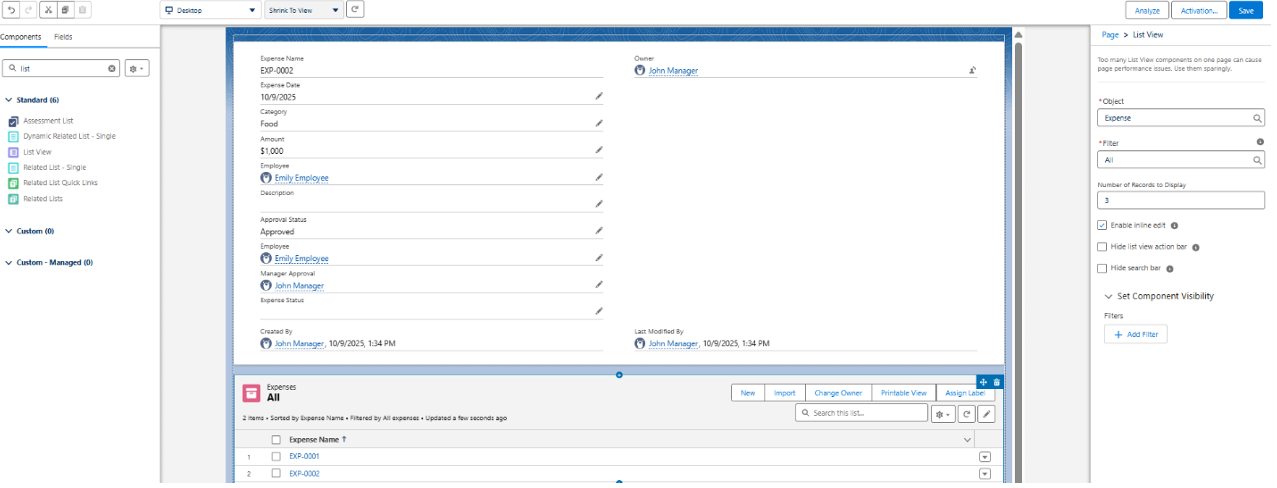


**Step 4: Develop Custom Lightning Web Components (LWC)**

* **Purpose:** To build interactive components that provide functionality beyond standard Salesforce elements, such as custom data tables or one-click approval buttons.
* **Tools:** Use **Visual Studio Code** with the Salesforce Extensions.

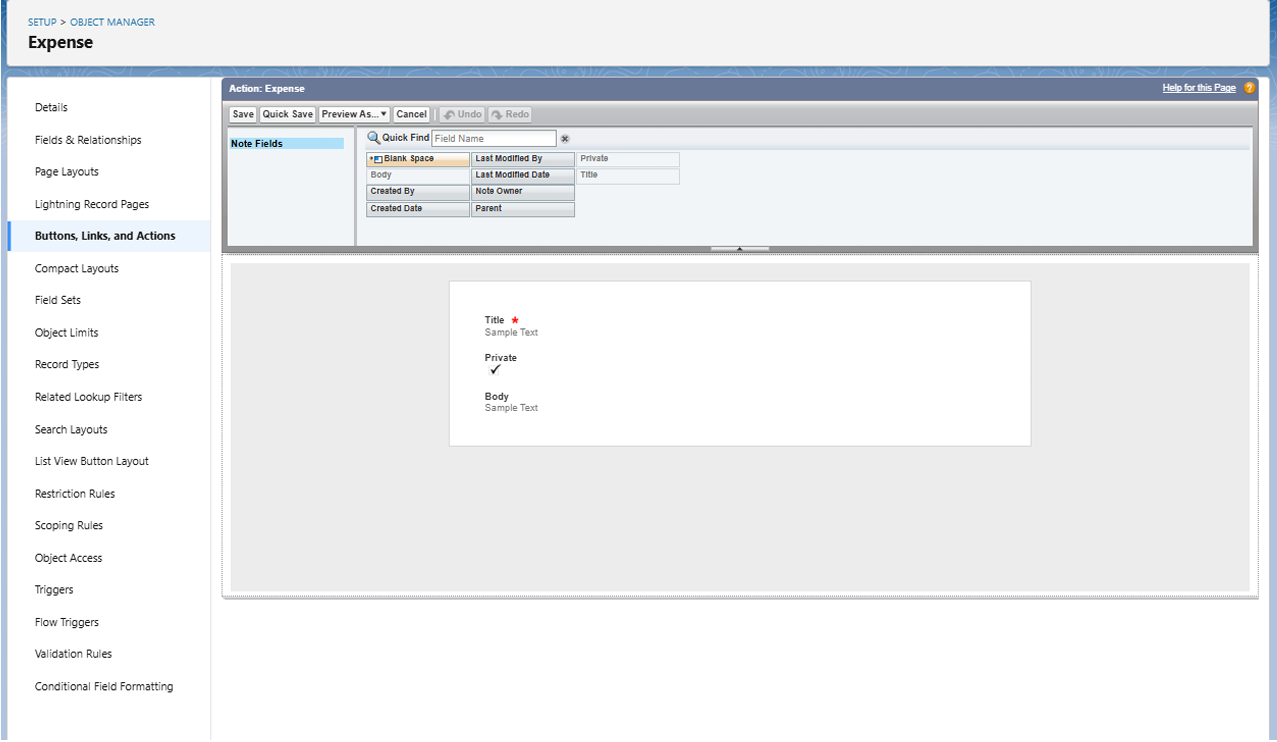
**Part A: Create the Apex Controller**

1. Create an Apex class (e.g., ExpenseController).
2. Add @AuraEnabled methods to make the code callable from an LWC.
   * Create a method to fetch expenses (e.g., getExpenses).
   * Create a method to update an expense (e.g., approveExpense).



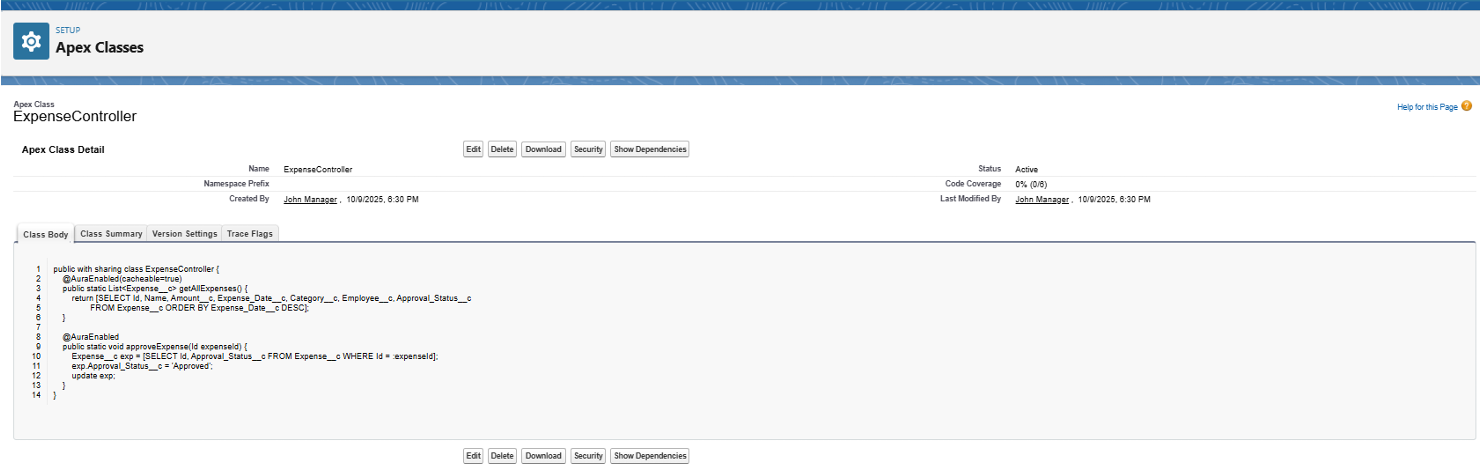
**Part B: Build the LWC (e.g., expenseList)**

1. Create an LWC to display expenses in a table (e.g., expenseList).
2. Use the @wire service to call the getExpenses Apex method and retrieve data dynamically.
3. Display the results in a lightning-datatable.



**Part C: Build an Imperative LWC (e.g., expenseApproveButton)**

1. Create a separate LWC for a custom "Approve" button.
2. When the button is clicked, use an **imperative Apex call** to the approveExpense method, passing the record ID to update the Approval Status.
3. Embed these new LWCs onto the Expense Record Page or Home Page using the Lightning App Builder.

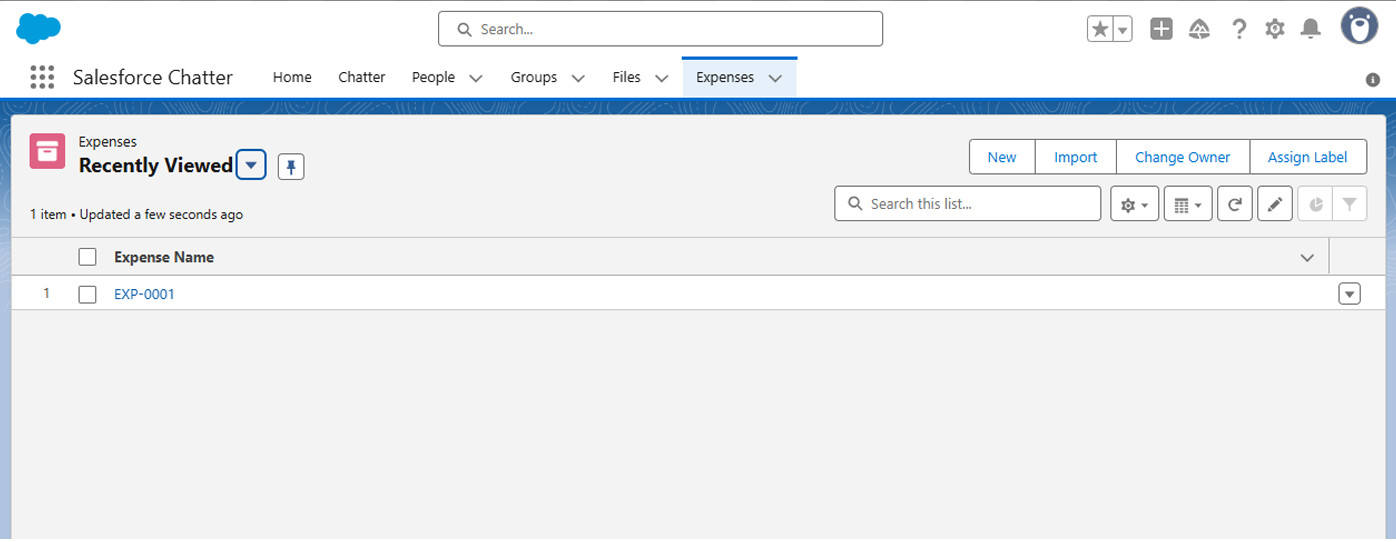


**Step 5: (Optional) Implement Navigation Service**

* **Purpose:** To programmatically redirect users to another page after a custom action is completed, such as navigating back to the home page after an approval.
* **Implementation:** In your LWC JavaScript file, import the NavigationMixin and call the Maps function to redirect the user.

**Step 6: Final User Acceptance Testing**

* **Purpose:** To verify the complete end-to-end user experience from the perspective of different users.
* **Test Cases:**
  1. **Employee Persona:** Log in as an employee. Can you open the Expense Tracker app, create a new expense, and submit it for approval?
  2. **Manager Persona:** Log in as the manager. Can you see the submitted expense? Are the custom approval buttons or actions visible? Can you successfully approve or reject the expense?
  3. **Data-Sync:** Confirm that after an action is taken, the custom LWC (if built) dynamically updates to show the new status. Verify all quick actions and navigation work as expected.



**Phase 6 Complete**

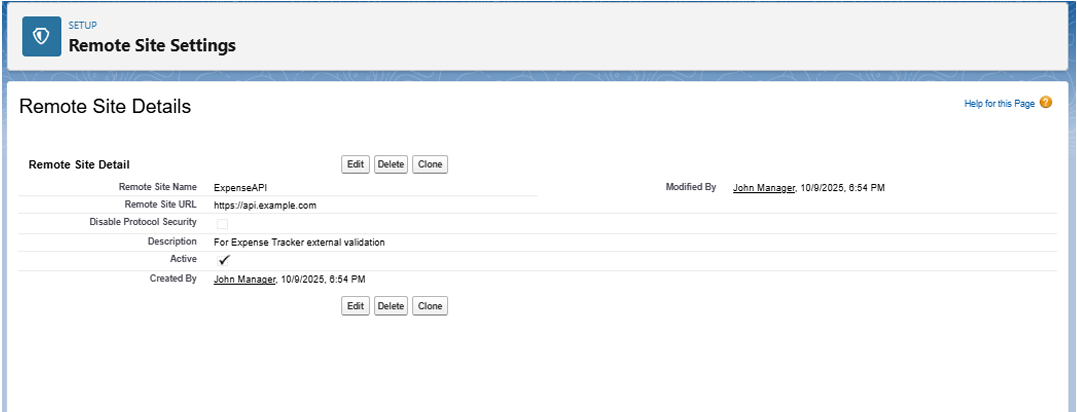
The Expense Tracker application is now fully encapsulated in a user-friendly Lightning App, complete with custom record pages, a dashboard-style home page, and interactive custom components.

**Expense Tracker Application - Phase 7: Integration and External Access**

**Objective:** To securely connect the Expense Tracker application with external third-party systems. This phase covers configuring Salesforce to both *request* data from external APIs (e.g., for validation) and *send* notifications *to* external systems (e.g., when an expense is approved).

**Step 1: Configure Remote Site Settings**

* **Purpose:** To authorize Salesforce to send outbound API requests (callouts) to a specific external domain. By default, Salesforce blocks all outbound calls to unknown URLs for security.
  + *Note: This step is often a prerequisite, but the modern* ***Named Credentials*** *(Step 2) approach is preferred as it handles both the URL and authentication, automatically bypassing this requirement.*
* **Navigation:** Go to **Setup** → **Security** → **Remote Site Settings** → **New Remote Site**.
* **Configuration:**
  + **Remote Site Name:** ExpenseAPI
  + **Remote Site URL:** https://api.example.com (Enter the root URL of the external service)
  + **Active:** Ensure this is checked.



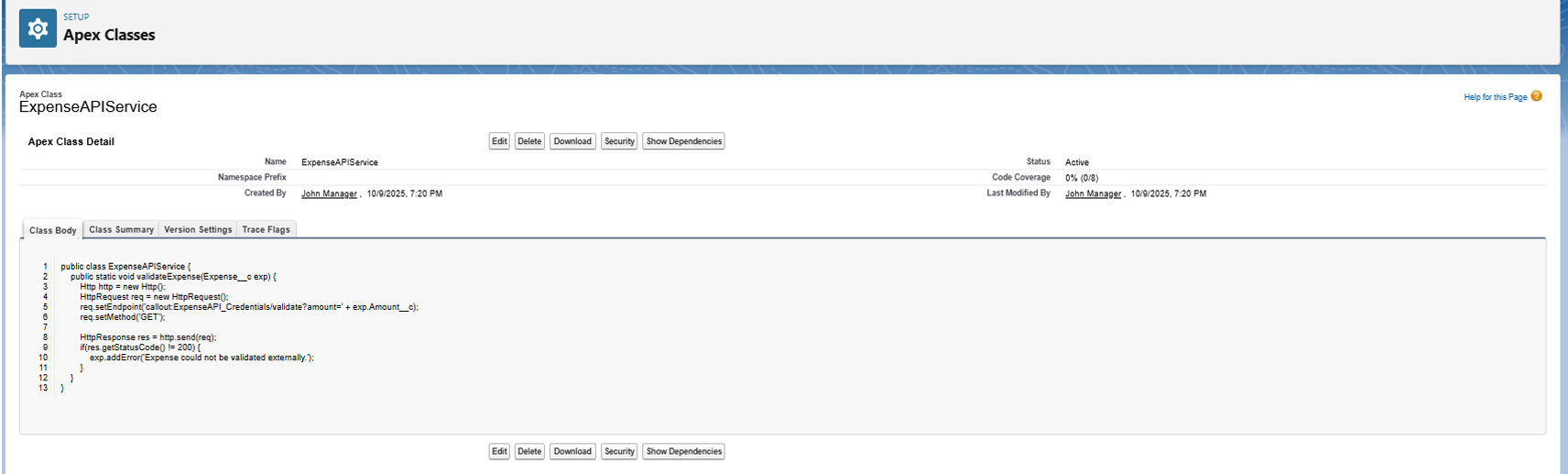
**Step 2: Create Named Credentials**

* **Purpose:** To securely store the API endpoint URL and its authentication credentials (like a password or OAuth token). This is the modern best practice as it separates sensitive data from your Apex code, making it more secure and easier to maintain.
* **Navigation:** Go to **Setup** → **Security** → **Named Credentials** → **New Named Credential**.
* **Configuration:**
  + **Label:** Expense\_API\_Credentials
  + **URL:** https://api.example.com (This is the base endpoint for the API)
  + **Identity Type:** Select Named Principal if all users share one integration credential.
  + **Authentication Protocol:** Choose the method required by the external API, such as Password Authentication or OAuth 2.0.



**Step 3: Implement Apex Callout Service**

* **Purpose:** To write the Apex code that makes the actual API request (callout) to the external service. This code will reference the Named Credential created in Step 2 to handle the endpoint and authentication seamlessly.
* **Navigation:** Go to **Setup** → **Apex Classes** → **New**.
* **Example Class:** ExpenseAPIService
  + This class would contain a method (e.g., validateExpense) that uses the HttpRequest and HttpResponse classes to send data to and receive a response from the external API.



**Step 4: (Optional) Configure Outbound Messaging with Events**

* **Purpose:** To notify external systems *from* Salesforce in real-time when an expense record changes (e.g., "Expense Approved" or "Expense Submitted").
* **Method A: Platform Events**
  + **Navigation:** Go to **Setup** → **Platform Events** → **New Platform Event**.
  + **Use Case:** Define a custom event (e.g., Expense\_Approved\_\_e). Your Apex trigger or Flow can then "publish" this event. An external system can "subscribe" to this event channel to receive the notification.
* **Method B: Change Data Capture (CDC)**
  + **Navigation:** Go to **Setup** → **Change Data Capture**.
  + **Use Case:** Enable CDC for the Expense object. Salesforce will automatically publish change events for any creation, update, or deletion of expense records. External systems can subscribe to this data stream.

**Step 5: Review Security & API Limits**

* **Purpose:** To ensure the integration is stable, secure, and operates within Salesforce's governor limits.
* **Key Actions:**
  + **Monitor Limits:** Regularly check API callout usage under **Setup** → **System Overview** → **API Usage**.
  + **Scope Permissions:** Ensure that only the necessary profiles or permission sets have access to the Named Credentials.
  + **Trust Domains:** Only add trusted, secure (HTTPS) domains to your integration settings.

**Phase 7 Complete**

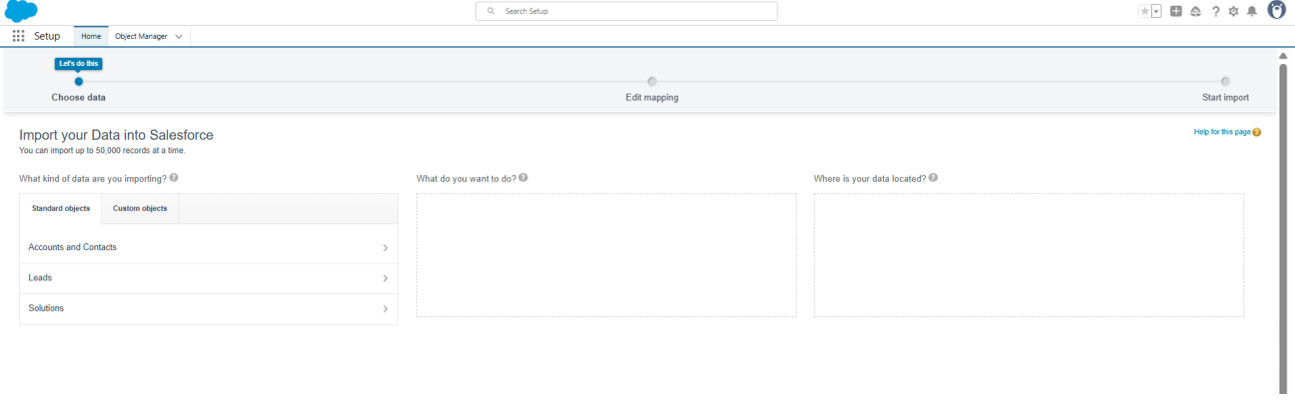
The Expense Tracker application is now capable of secure, two-way communication with external systems. Salesforce is configured to trust external endpoints, store credentials securely, and call external APIs. The system can also proactively notify external services of record changes.

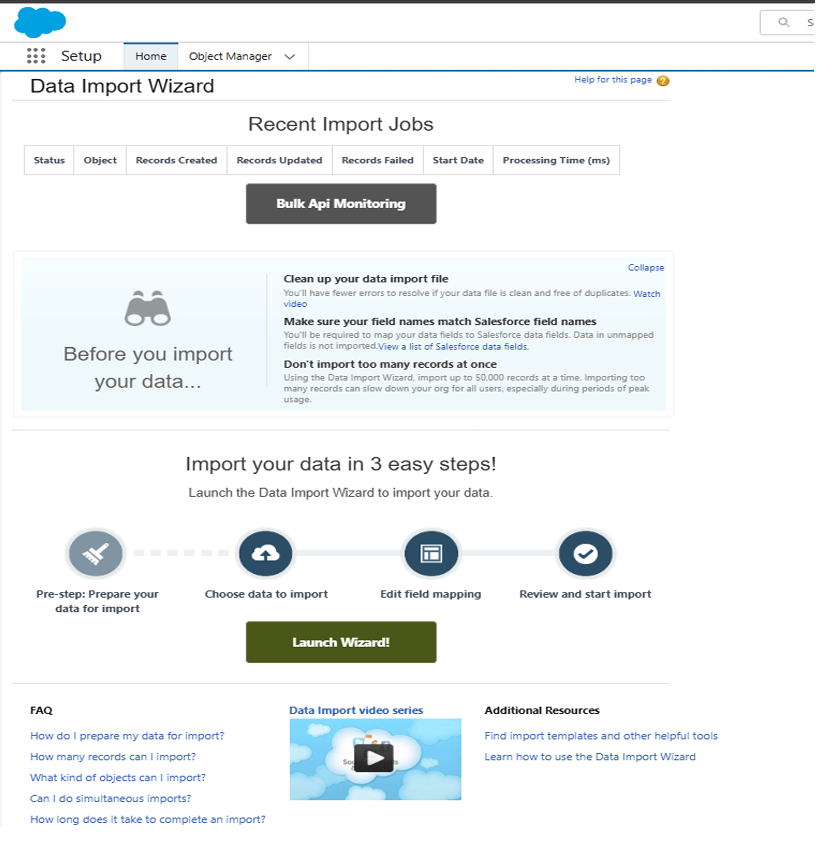
**Expense Tracker Application - Phase 8: Data Management and Deployment**

**Objective:** To establish procedures for managing application data, ensuring data integrity, and executing the successful migration of the Expense Tracker application from a sandbox environment to production.

**Step 1: Import Test Data with the Data Import Wizard**

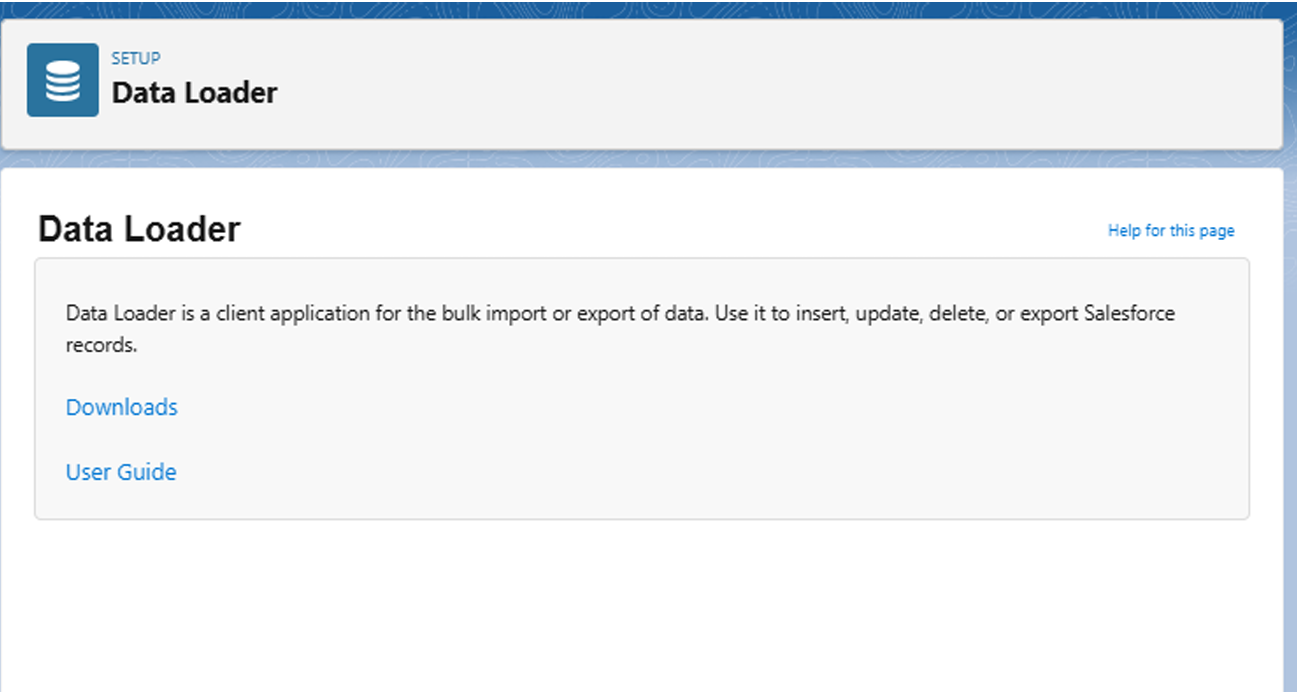
* **Purpose:** To quickly import small datasets (up to 50,000 records) directly through the Salesforce UI. This tool is simple, intuitive, and ideal for loading initial test data or demo expense records without needing an external tool.
* **Navigation:** Go to **Setup** → **Data** → **Data Import Wizard**.
* **Process:**
  1. Launch the wizard.
  2. Select the **Expense** custom object.
  3. Choose the operation, such as **Add New Records**.
  4. Upload your CSV file containing test data (e.g., Amount, Expense Date, Category, Employee) .
  5. Map your CSV column headers to the corresponding Salesforce fields.
  6. Start the import and review the results log for any successes or errors.





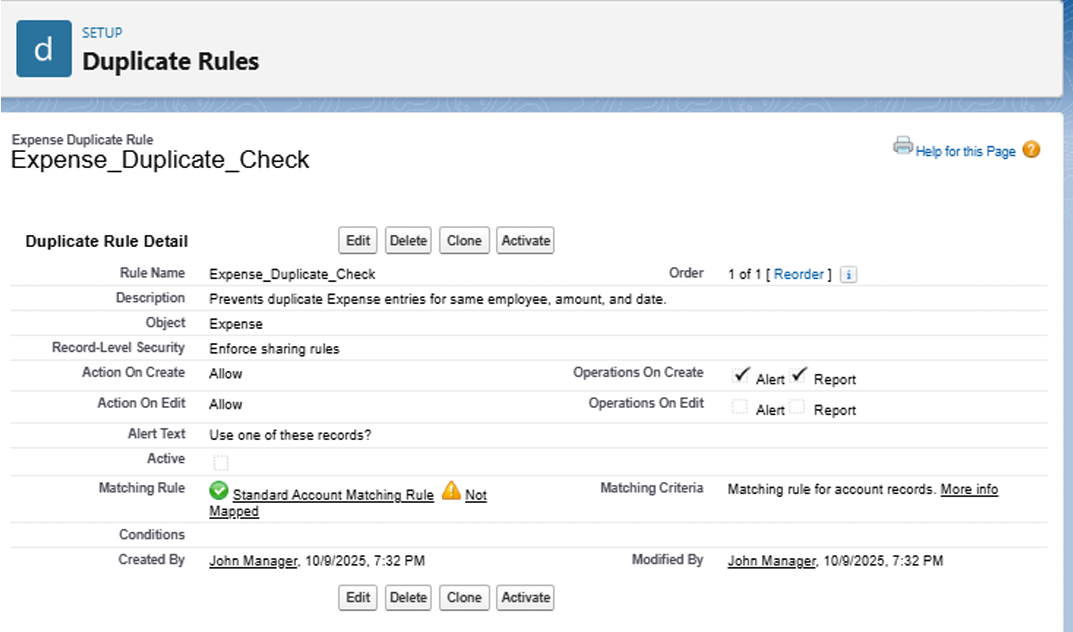
**Step 2: Manage Large Datasets with Data Loader**

* **Purpose:** For bulk data operations involving hundreds or thousands of records, such as mass imports, updates, or exports.
* **Navigation:** This is a separate client application that you must download and install.
* **Process:**
  1. Open the Data Loader application and log in to your Salesforce org.
  2. Select an operation: **Insert**, **Update**, or **Export**.
  3. Choose the **Expense\_\_c** object and provide your CSV file.
  4. Map the fields and run the operation, reviewing the success and error logs afterward.
  5. **Why it's important:** This tool is essential for exporting data for backups or migrating large volumes of data that the UI wizard can't handle.



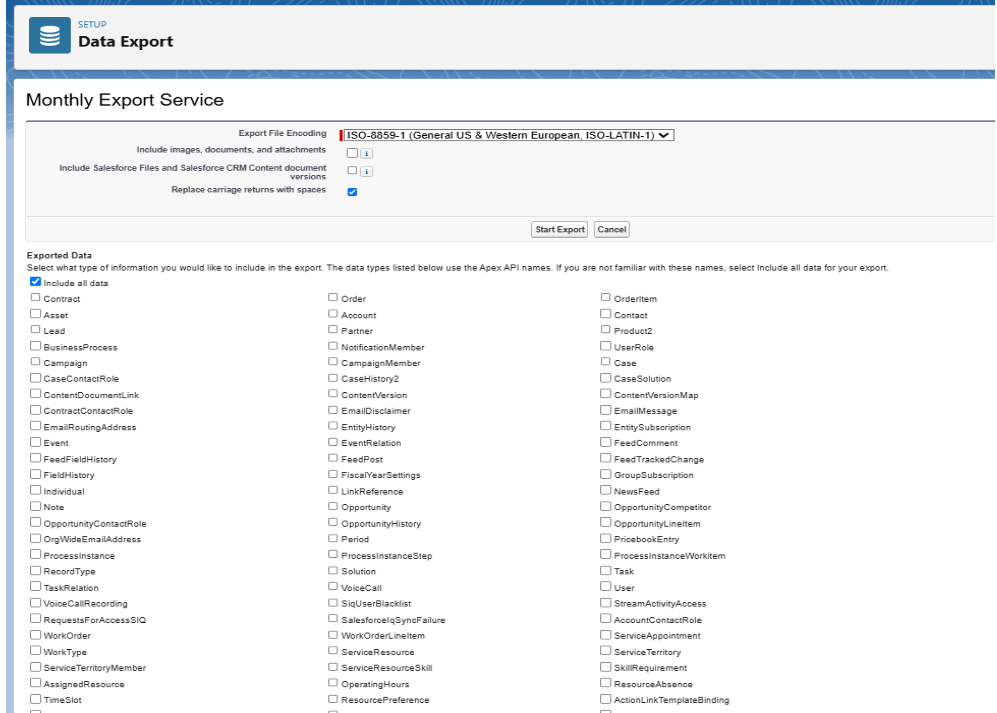
**Step 3: Prevent Duplicates with Duplicate Rules**

* **Purpose:** To maintain clean, reliable data by preventing users from creating duplicate expense records. This stops duplicate entries from appearing in reports or approval queues.
* **Navigation:** Go to **Setup** → **Duplicate Management** → **Duplicate Rules** → **New Rule**.
* **Configuration:**
  1. Select the **Expense\_\_c** object.
  2. Name the rule (e.g., Expense\_Duplicate\_Check).
  3. Define the matching criteria. For example, a record is a duplicate if it has an exact match on **Amount**, **Expense Date**, and **Employee** .
  4. Set the **Action on Create** to either **Block** the user from saving or **Allow** it but show an **Alert**.
  5. **Activate** the rule.



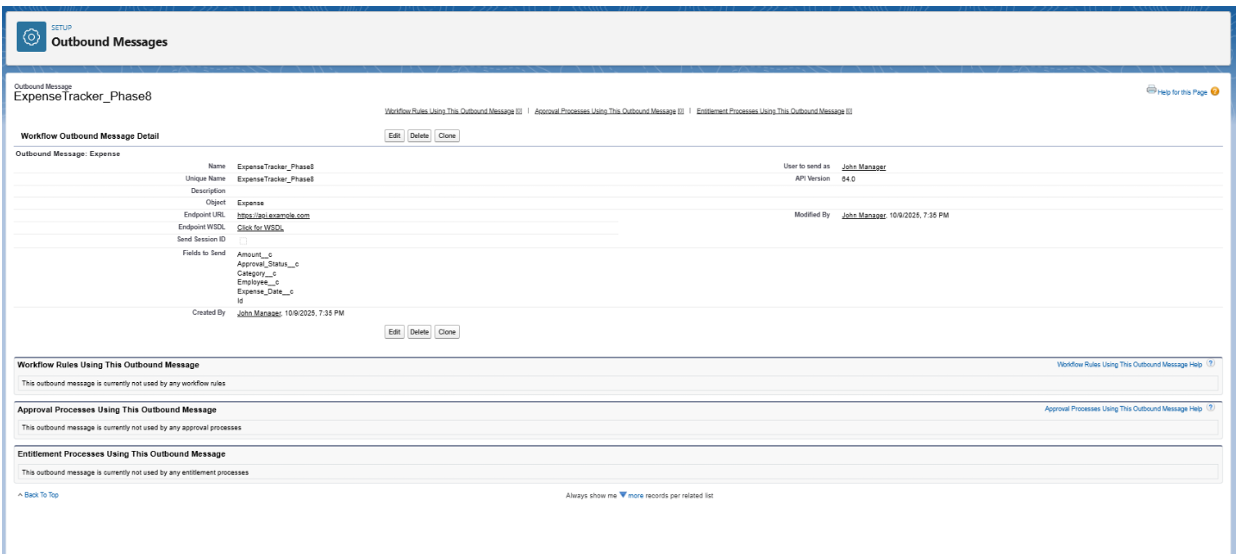
**Step 4: Schedule Regular Data Backups**

* **Purpose:** To create a secure backup of your application's data for data recovery or audit compliance. This protects against accidental data loss.
* **Navigation:** Go to **Setup** → **Data** → **Data Export**.
* **Process:**
  1. You can either **Export Now** for an immediate backup or **Schedule Export** for a recurring weekly backup.
  2. Select the objects to include, such as **Expense\_\_c** and **User**.
  3. Start the export. Salesforce will email you a link to download a .zip file containing your data, which you should store securely.



**Step 5: Deploy to Production Using Change Sets**

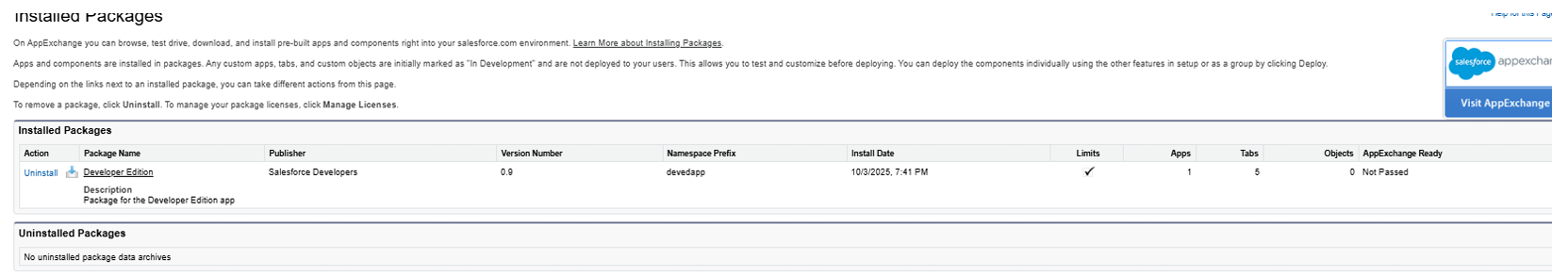
* **Purpose:** This is the standard, UI-based method for safely migrating all your configured components (objects, automation, code) from a Sandbox environment to the live Production org.
* **Process (In Sandbox):**
  1. Navigate to **Setup** → **Outbound Change Sets** → **New**.
  2. Name your change set (e.g., Expense\_Tracker\_Deployment).
  3. Click **Add** to include all your components. This includes the Expense\_\_c custom object, all its fields, validation rules, flows, approval processes, Apex classes, and Lightning pages .
  4. Once all components are added, **Upload** the change set to your Production org.
* **Process (In Production):**
  1. Navigate to **Setup** → **Inbound Change Sets**.
  2. Find the uploaded change set, **Validate** it to check for errors, and then **Deploy** it.



**Step 6: Advanced Deployment Options (Optional)**

For more complex or reusable applications, you can also use:

* **Packages:** You can bundle all components into a **Package** (either unmanaged or managed) using the **Package Manager**. This is ideal for distributing your app, sharing it, or for version control.
* **SFDX and VS Code:** Developers can use the **Salesforce CLI (SFDX)** and VS Code to retrieve metadata from one org (sfdx force:source:retrieve) and deploy it to another (sfdx force:source:deploy). This is a code-driven approach favored for developer-centric workflows.



**Expense Tracker Application - Phase 9: Reporting, Dashboards, and Security Review**

**Objective:** To analyze application data by building insightful reports and dashboards, and to conduct a final security review to ensure the application is secure, compliant, and ready for end-users.

**1. Build Key Reports**

**Purpose:** To create a series of reports to monitor spending, track the status of approvals, and identify expense patterns across the organization.

**Key Reports to Create:**

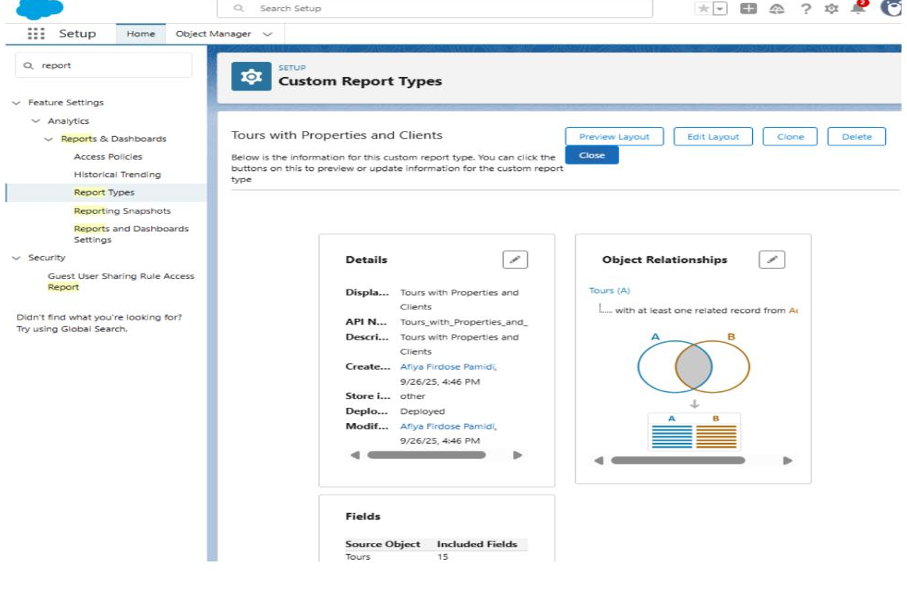
* **Expenses by Category:** Shows the total amount spent, grouped by category (e.g., Food, Travel).
* **Pending Approvals:** A list of all expense records currently awaiting manager approval.
* **Monthly Expense Trend:** A line report tracking the total volume of expenses submitted over time.
* **Employee Expense Summary:** Shows the total spending submitted by each employee.

**How to Create the 'Expenses by Category' Report:**

1. **Navigate:** Go to the **App Launcher** → **Reports** → **New Report**.
2. **Report Type:** Select the **Expenses** report type. (Note: If this is not visible, you may need to go to Setup → Object Manager → Expense, edit the object, and check "Allow Reports" ).
3. **Add Columns:** From the fields panel, add key columns to your report, such as Expense Number , Employee , Expense Date , Category , Amount , and Approval Status.
4. **Group Data:** Drag the Category field to the **Group Rows** section.
5. **Summarize Data:** On the Amount column, click the arrow and select **Summarize** → **Sum** .
6. **Set Filters:** In the **Filters** tab, set the Expense Date range to All Time or This Fiscal Year as needed.
7. **Save:** Click **Save & Run**. Name the report Expenses by Category and save it in a shared folder.

**Create a Custom Report Type (Optional):**

* **Purpose:** To create a report that links the Expense object to the User object, allowing you to include employee details (like their department, role, or manager) in the same report.
* **Navigation:** Go to **Setup** → **Report Types** → **New Custom Report Type**.
* **Configuration:** Set Expense as the Primary Object and relate it to the User (Employee) object.



**2. Create the Expense Tracker Dashboard**

**Purpose:** To provide managers and executives with a quick, visual overview of key expense metrics, all in one place.

**Dashboard Creation Steps:**

1. **Navigate:** Go to the **Dashboards** tab and click **New Dashboard**.
2. **Name:** Give your dashboard a name, such as Expense Tracker Dashboard.
3. **Add Components:** Click **+ Component** and use the reports you built in the previous step:
   * Use the Expenses by Category report to create a **Pie Chart**.
   * Use the Pending Approvals report to create a **Table**.
   * Use the Monthly Expense Trend report to create a **Line Chart**.
4. **Arrange and Save:** Drag and drop the components to arrange your layout, then click **Save**.

**Advanced Feature: Dynamic Dashboards**

* You can set the dashboard to run as "The logged-in user". This ensures that managers only see data for their own teams, and employees only see their own expense data, all from the same dashboard.

**3. Conduct Final Security Review**

**Purpose:** To review and harden the security settings for the application, ensuring data is only visible to the correct users.

* **Organization-Wide Defaults (OWD):**
  + **Action:** Go to **Setup** → **Sharing Settings** and confirm that the Expense object's internal and external access is set to **Private**. This is the foundation of your security, ensuring users can only see their own records.
* **Field-Level Security (FLS):**
  + **Action:** Go to **Object Manager** → **Expense** → **Fields & Relationships**. For any sensitive fields (e.g., Manager\_Comments ), click the field and **Set Field-Level Security** to hide it from profiles that should not see it, like a standard employee profile.
* **Login IP Ranges:**
  + **Action:** For high-security environments, go to **Setup** → **Profiles** and select a profile. Under **Login IP Ranges** , you can restrict users to only log in from trusted networks, like an office IP address.
* **Session Settings:**
  + **Action:** Go to **Setup** → **Session Settings**. Set the **Timeout Value** to a shorter duration (e.g., 30 minutes ) to automatically log out inactive users and reduce security risks.
* **Setup Audit Trail:**
  + **Action:** Go to **Setup** → **View Setup Audit Trail**. Regularly review this log to see who has made changes to your Salesforce configuration.

**Phase 9 Complete**

The Expense Tracker application is now fully equipped with analytical tools, allowing management to track spending and approvals. The security settings have been hardened to protect sensitive financial data, ensuring the application is compliant and secure. The project is now ready for end-user training and go-live.

**Expense Tracker Application - Phase 10: Go-Live, Training, and Project Handoff**

**Objective:** To successfully launch the Expense Tracker application to all end-users, conduct role-specific training, demonstrate the final product's value to stakeholders, and deliver all necessary documentation for long-term maintenance and future development.

**Step 1: Conduct End-User Training and Go-Live**

* **Purpose:** To ensure all stakeholder groups are proficient in using the new application before it becomes the official system of record.
* **Actions:**
  + **Role-Specific Training:**
    - **Employee Session:** Train the "Expense Employee Profile" users on:
      * Accessing the new **'Expense Tracker' Lightning App**.
      * Submitting new expenses using the **"Submit Expense Form" Screen Flow**.
      * Understanding the **Approval Status** picklist (Pending, Approved, Rejected) on their records.
    - **Manager Session:** Train the "Manager" role users on:
      * Monitoring the **'Expense Tracker Dashboard'** , focusing on the 'Pending Approvals' component.
      * Reviewing, approving, and rejecting submissions from the **Approval History** related list or custom LWC buttons.
  + **Go-Live Activation:**
    - Formally activate the **Lightning App** and **Lightning Record Pages** as the default for all target profiles.
    - In Production, schedule and activate the automated Apex jobs: **ScheduleOverdueExpenses** and **DailyExpenseSummary**.
    - Send the official company-wide launch communication.

**Step 2: Final Stakeholder Presentation and Demo**

* **Purpose:** To demonstrate the full, end-to-end business process and showcase the value delivered to executive and finance stakeholders.
* **Actions:**
  + **Pitch Presentation:**
    - Begin by restating the core problem from Phase 1: inefficient, manual, and error-prone reporting.
    - Present the solution: a unified, automated, and secure platform.
    - Emphasize key value propositions:
      * **Data Integrity:** Showcase the Validation Rules (Amount\_Positive, Date\_Not\_Future) that enforce policy automatically.
      * **Process Efficiency:** Highlight the automated **Expense Manager Approval** process that routes claims to the correct manager.
      * **Actionable Insights:** Feature the **Expense Tracker Dashboard** as the tool for real-time leadership visibility.
  + **Live Demo Walkthrough:**
    - **Employee Persona:** Log in as the "Employee User". Navigate to the **Expense Tracker** app and submit a new expense using the Screen Flow.
    - **Manager Persona:** Log in as the "Manager User". Show the new expense appearing on their **Home Page** dashboard and/or in their approval queue. Approve the record.
    - **Finance/Admin Persona:** Refresh the **Expense Tracker Dashboard** to show the newly approved data reflected in the 'Expenses by Category' pie chart and other reports.

**Step 3: Collect Feedback and Establish Support**

* **Purpose:** To gather immediate feedback from the live-user experience and establish a "hypercare" period for rapid support.
* **Actions:**
  + **Feedback Mechanism:** Distribute a simple survey or feedback form to all trained users, asking for specific feedback on the submission flow and dashboard usability.
  + **Monitor and Support:**
    - Establish a clear support channel (e.g., a dedicated email or Chatter group) for the first 30 days.
    - Admins actively monitor for any automated error notifications from Flows , Apex , or batch jobs.
    - Review the **Setup Audit Trail** and API limits to ensure system health.

**Step 4: Deliver Final Handoff Documentation**

* **Purpose:** To equip the System Administrator and IT department with all necessary documentation to own, maintain, and enhance the application.
* **Actions:**
  + **Configuration Guide:**
    - Provide a full data dictionary for the **Expense\_\_c** object and all its custom fields.
    - Document all automation logic, including Validation Rules , the Approval Process , and all Record-Triggered Flows.
  + **Technical (Apex/LWC) Guide:**
    - Hand off all Apex classes, clearly explaining the logic in:
      * ExpenseService (Trigger Handler)
      * ExpenseTrigger
      * BatchOverdueExpenses (and its scheduler)
      * ExpenseController (for LWCs)
    - Provide the source code and purpose for all custom Lightning Web Components (e.g., expenseList, expenseApproveButton).
  + **Integration & Data Guide:**
    - Document all integration points, including **Named Credentials** and the purpose of the **ExpenseAPIService**.
    - Explain the **Duplicate Rule** logic and the **Data Export** backup schedule.
  + **Deployment Package:**
    - Provide the final, consolidated **Outbound Change Set** or SFDX Package used for the production deployment as a final backup of the metadata.

**Step 5: Create Portfolio and Project Showcase**

* **Purpose:** To translate the completed project into a professional asset that clearly communicates your skills and the project's success.
* **Actions:**
  + **Build Case Study:**
    - Create a portfolio write-up using the STAR method.
    - **Situation:** Companies use inefficient, manual processes for expense reporting.
    - **Task:** To build a secure, scalable, and automated Expense Tracker application on the Salesforce platform.
    - **Action:** "I designed and built the full solution, including:
      * Architecting the core **Expense\_\_c** data model.
      * Building a modern **Lightning App** with custom **LWCs** for a streamlined user experience.
      * Implementing complex business logic via **Flows** , **Apex Triggers** , and **Batch Apex** to automate approvals and nightly processing.
      * Creating real-time **reports and dashboards** for executive insight.
      * Managing the full data and deployment lifecycle using **Change Sets**."
    - **Result:** A successful, end-to-end application that reduces data entry errors, shortens reimbursement cycles , and provides real-time visibility into corporate spending.

Link: https://github.com/Gunasekharachari/Expense-Tracker-Project.git

**Phase 10 Complete**

The Expense Tracker application is now fully deployed, trained, and adopted by its end-users. All technical and administrative documentation has been handed off to the system owner, and the project is formally closed, marking a successful transition from development to a live, value-driving business solution.