**Expense Tracker Application: Final Project Report**

**Author:** Kothapalli Guna Sekhar Achari **Date:** October 22, 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
* Advanced Workflow Tools
* OCR & Document Management
* Business Intelligence Platforms
* Governance & Compliance Frameworks

**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.
* **Purpose:** This provides a free, sandboxed environment to build and test the application without impacting a live system.

**Step 2: Configure Company Information**

* **Navigate:** Go to Setup, use the Quick Find box to search for "Company Information," and click Edit.
* **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).
* **Purpose:** This configuration ensures that all date, time, and currency fields display correctly throughout the org.

**Step 3: Set Fiscal Year**

* **Navigate:** Go to Setup, then use Quick Find to search for "Fiscal Year".
* **Action:** Confirm that the "Standard Fiscal Year" (January-December) is selected.
* **Purpose:** Establishes standard reporting periods, which is crucial for future expense and revenue reporting.

**Step 4: Establish Role Hierarchy**

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

* **Navigate:** Go to Setup, use Quick Find to search for "Users," and click "New User".
* **Create Manager User:**
  + **Profile:** System Administrator
  + **Role:** Manager
* **Create Employee User:**
  + **Profile:** Standard User
  + **Role:** Employee
* **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.
* **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**

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

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

* **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).
* **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**

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

**Phase 2 Completion:** The Salesforce organization is now minimally configured for the Expense Tracker project.

**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.

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

* **Navigate:** Go to Setup → Object Manager, then click Create → Custom Object.
* **Enter Details:**
  + **Label:** Expense
  + **Plural Label:** Expenses
  + **Object Name:** Expense (API name Expense\_\_c)
* **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).
* **Save** the new object.

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

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

* **Page Layout:**
  + **Navigate:** From the Expense object page, go to Page Layouts and Edit the default layout.
  + **Action:** Drag and drop the fields onto the layout in a logical order: Expense Number, Amount, Expense Date, Category, Employee, Description, and Approval Status.
* **Compact Layout:**
  + **Navigate:** From the Expense object page, go to Compact Layouts and click New.
  + **Action:** Add key fields: Expense Number, Amount, Expense Date, and Approval Status.
  + **Save** and 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**

* **Purpose:** Record types allow for different business processes and page layouts for various kinds of expenses.
* **Navigate:** From the Expense object page, go to Record Types and click New.
* **Create Record Types:** Set up distinct record types such as Travel Expense, Food Expense, and Other Expense.
* **Assign:** Make these record types available to the Expense Employee Profile and Manager/Admin profiles.

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

* **Navigate:** Go to Setup → Schema Builder.
* **Action:** In the object list, find and select the Expense and User objects.
* **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**

* **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 and repeat for the Manager/Admin profile.

**Step 7: Conduct a Validation Test**

* **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.
* **Log in as the Manager user.** Verify that you can view the record created by the employee.
* **Negative Test:** Log in as a different employee user 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.

**Phase 4: Business Process Automation**

**Objective:** To implement the business logic and automation required for a functional expense management process. This phase focuses on data integrity, approval automation, and guided user experiences.

**Step 1: Implement Validation Rules**

* **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"
* **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"

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

* **Purpose:** To provide a simple, guided form for employees, ensuring all required information is captured.
* **Navigation:** Go to Setup → Flow → New Flow → Screen Flow.
* **Flow Structure:**
  + **Screen Element:** Create a screen titled "Submit Expense Form". Add input components for Amount, Expense Date, Category, and Description.
  + **Create Records Element:** Add this element after the screen.
    - **Label:** "Create Expense Record".
    - **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.
* **Save and Activate** the flow.

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

* **Purpose:** To ensure that expenses exceeding a certain monetary threshold receive manager approval.
* **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'.
* **Activate:** Once all steps are configured, **Activate** the approval process.

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

* **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: Approval\_Status\_\_c equals 'Approved'.
  + **Optimize for:** Select "Actions and Related Records."
* **Flow Element:**
  + Add an **Update Records** element.
  + **Label:** "Update Expense Status to Confirmed".
  + **Set Field Values:** Set a field (e.g., Expense\_Status\_\_c) to 'Confirmed'.
* **Save and Activate** the flow.

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

* **Purpose:** To notify employees or managers directly within the Salesforce app (bell icon) when an action occurs.
* **Navigation (Part 1):** Go to Setup → Notification Builder → Custom Notifications → New.
  + **Name:** Expense Notification.
  + **Supported Channels:** Check Desktop and Mobile.
* **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.
  + **Recipient:** Set the recipient (e.g., the record's owner $Record.OwnerId).
  + **Message:** Create a dynamic message, such as "Your expense {$Record.Name} has been approved".

**Phase 4 Complete:** With these automations, the Expense Tracker application is now intelligent. Employees are guided, data is validated, approvals are routed, statuses are updated, and users are notified.

**Phase 5: Advanced Apex Programming**

**Objective:** To implement advanced business logic, automated batch processing, and maintainability for the Expense Tracker application using Apex.

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

* **Purpose:** To centralize all business logic (such as calculations or complex validations) into a single, reusable class. This "Trigger Handler" pattern makes the code more efficient and maintainable.
* **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 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 runs nightly, finds all expenses that are past their due date and still pending, and updates 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). This ensures the "overdue" check runs consistently.
* **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 frequency to 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 a 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 production. Test classes verify that all Apex logic works as expected and meets Salesforce's code coverage requirements.
* **Example Class:** ExpenseServiceTest
* **Navigation:** Go to Setup → Apex Classes → New.

**Phase 5 Complete:** The developer-focused tasks are now complete. The deliverables include a centralized ExpenseService class , an ExpenseTrigger , a BatchOverdueExpenses class , scheduler classes for automation , and comprehensive Test Classes.

**Phase 6: Lightning UI and App Development**

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

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

* **Purpose:** To group all project-related tabs, pages, and tools into a single, branded application.
* **Navigation:** Go to Setup → App Manager → New Lightning App.
* **Configuration:**
  + **App Details:** Enter the App Name as Expense Tracker.
  + **Navigation Items:** Add the Expenses tab and the Reports tab.
  + **Utility Bar (Optional):** Add a utility item, such as the New Expense Quick Action.
  + **User Profiles:** Assign the app to relevant user profiles (e.g., Expense Employee, Manager).
* **Save & Finish**.

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

* **Purpose:** To create an efficient and informative layout for viewing a single expense record.
* **Navigation:** Go to Setup → Object Manager → Expense → Lightning Record Pages → New.
* **Layout Configuration:**
  + **Template:** Select a template, such as Header + Right Sidebar.
  + **Main Region:** Drag the Record Detail component into the main section. This will display fields like Amount, Expense Date, Category, and Approval Status.
  + **Sidebar:** Drag the Related Lists component into the sidebar. Include Approval History.
  + **Highlights Panel:** Configure the header to show key actions like Submit for Approval.
* **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.
* **Navigation:** Go to Setup → Lightning App Builder → New → Home Page.
* **Component Configuration:**
  + Add Dashboard or Report Chart components to visualize data like Pending Approvals.
  + Add a List View component configured to show Recent Expenses.
* **Save** and **Activate** the page, assigning it as the default Home Page for the Expense Tracker app.

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

* **Purpose:** To build interactive components that provide functionality beyond standard elements.
* **Tools:** Use Visual Studio Code with the Salesforce Extensions.
* **Part A: Create the Apex Controller**
  + Create an Apex class (e.g., ExpenseController).
  + Add @AuraEnabled methods to make the code callable from an LWC.
  + Create methods to fetch expenses (e.g., getExpenses) and update an expense (e.g., approveExpense).
* **Part B: Build the LWC (e.g., expenseList)**
  + Create an LWC to display expenses in a table.
  + Use the @wire service to call the getExpenses Apex method.
  + Display the results in a lightning-datatable.
* **Part C: Build an Imperative LWC (e.g., expenseApproveButton)**
  + Create a separate LWC for a custom "Approve" button.
  + When the button is clicked, use an imperative Apex call to the approveExpense method.
* **Embed** these new LWCs onto the Expense Record 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.
* **Implementation:** In your LWC JavaScript file, import the NavigationMixin.

**Step 6: Final User Acceptance Testing**

* **Purpose:** To verify the complete end-to-end user experience.
* **Test Cases:**
  + **Employee Persona:** Log in as an employee. Can you open the Expense Tracker app, create a new expense, and submit it?
  + **Manager Persona:** Log in as the manager. Can you see the submitted expense? Can you successfully approve or reject it?
  + **Data-Sync:** Confirm that after an action, the custom LWC dynamically updates.

**Phase 6 Complete:** The Expense Tracker application is now fully encapsulated in a user-friendly Lightning App, complete with custom pages and interactive components.

**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 request data from and send notifications to external systems.

**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.
* **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.
* **Navigation:** Go to Setup → Security → Remote Site Settings → New Remote Site.
* **Configuration:**
  + **Remote Site Name:** ExpenseAPI
  + **Remote Site URL:** https://api.example.com
  + **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.
* **Navigation:** Go to Setup → Security → Named Credentials → New Named Credential.
* **Configuration:**
  + **Label:** Expense\_API\_Credentials
  + **URL:** https://api.example.com
  + **Identity Type:** Select Named Principal.
  + **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 to handle the endpoint and authentication.
* **Navigation:** Go to Setup → Apex Classes → New.
* **Example Class:** ExpenseAPIService
* This class would contain a method that uses the HttpRequest and HttpResponse classes to send and receive data.

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

* **Purpose:** To notify external systems *from* Salesforce in real-time when an expense record changes.
* **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.
* **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 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 only necessary profiles have access to the Named Credentials.
  + **Trust Domains:** Only add trusted, secure (HTTPS) domains.

**Phase 7 Complete:** The Expense Tracker application is now capable of secure, two-way communication with external systems. Salesforce is configured to trust endpoints, store credentials securely, call external APIs , and proactively notify external services.

**Phase 8: Data Management and Deployment**

**Objective:** To establish procedures for managing application data, ensuring data integrity, and executing the successful migration of the application from sandbox 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 ideal for loading initial test data.
* **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.
  5. Map your CSV column headers to the Salesforce fields.
  6. Start the import and review the results.

**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 Data Loader and log in.
  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.
* **Why it's important:** This tool is essential for backups or migrating large volumes of data.

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

* **Purpose:** To maintain clean, reliable data by preventing users from creating duplicate expense records.
* **Navigation:** Go to Setup → Duplicate Management → Duplicate Rules → New Rule.
* **Configuration:**
  + Select the Expense\_\_c object.
  + **Name the rule** (e.g., Expense\_Duplicate\_Check).
  + **Define the matching criteria:** For example, a record is a duplicate if it has an exact match on Amount, Expense Date, and Employee.
  + **Set the Action on Create** to either Block the user or Allow with an Alert.
  + **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.
* **Navigation:** Go to Setup → Data → Data Export.
* **Process:**
  + You can either Export Now or Schedule Export for a recurring weekly backup.
  + Select the objects to include, such as Expense\_\_c and User.
  + Salesforce will email you a link to download a .zip file.

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

* **Purpose:** The standard, UI-based method for safely migrating all configured components (objects, automation, code) from a Sandbox to Production.
* **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: the Expense\_\_c custom object, all its fields, validation rules, flows, approval processes, Apex classes, and Lightning pages.
  4. 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, and then Deploy it.

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

* **Packages:** You can bundle all components into a Package using the Package Manager. This is ideal for distributing your app.
* **SFDX and VS Code:** Developers can use the Salesforce CLI (SFDX) to retrieve and deploy metadata. This is a code-driven approach favored for developer-centric workflows.

**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 and compliant.

**1.** Build Key Reports

* **Purpose:** To create reports to monitor spending, track approval status, and identify patterns.
* **Key Reports to Create:**
  + **Expenses by Category:** Shows total amount spent, grouped by category.
  + **Pending Approvals:** A list of all expense records awaiting approval.
  + **Monthly Expense Trend:** A line report tracking expense volume over time.
  + **Employee Expense Summary:** Shows total spending by employee.
* **How to Create the 'Expenses by Category' Report:**
  1. **Navigate:** Go to the Reports tab → New Report.
  2. **Report Type:** Select the Expenses report type.
  3. **Add Columns:** Add 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. **Save & Run** the report.
* **Create a Custom Report Type (Optional):**
  1. **Purpose:** To link the Expense object to the User object, allowing you to include employee details (like their department or role) in the report.
  2. **Navigation:** Go to Setup → Report Types → New Custom Report Type.

**2.** Create the Expense Tracker Dashboard

* **Purpose:** To provide managers and executives with a quick, visual overview of key expense metrics.
* **Dashboard Creation Steps:**
  1. **Navigate:** Go to the Dashboards tab and click New Dashboard.
  2. **Name:** Expense Tracker Dashboard.
  3. **Add Components:** Click + Component and use the reports you built:
     + Use Expenses by Category to create a **Pie Chart**.
     + Use Pending Approvals to create a **Table**.
     + Use Monthly Expense Trend to create a **Line Chart**.
  4. **Arrange and Save**.
* **Advanced Feature: Dynamic Dashboards**
  1. You can set the dashboard to run as "The logged-in user". This ensures that managers only see data for their own teams.

**3.** Conduct Final Security Review

* **Purpose:** To review and harden the security settings, ensuring data is only visible to the correct users.
* **Organization-Wide Defaults (OWD):**
  + **Action:** Go to Setup → Sharing Settings and confirm the Expense object's access is set to Private. This is the foundation of your security.
* **Field-Level Security (FLS):**
  + **Action:** Go to Object Manager → Expense → Fields & Relationships. For sensitive fields, Set Field-Level Security to hide them from profiles that should not see them.
* **Login IP Ranges:**
  + **Action:** Go to Setup → Profiles. Under Login IP Ranges, you can restrict users to only log in from trusted networks.
* **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.
* **Setup Audit Trail:**
  + **Action:** Go to Setup → View Setup Audit Trail. Regularly review this log to see who has made configuration changes.

**Phase 9 Complete:** The application is now fully equipped with analytical tools , and security settings have been hardened to protect sensitive financial data.

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

**Objective:** To successfully launch the Expense Tracker application, conduct role-specific training, demonstrate the final product's value, and deliver all necessary documentation.

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

* **Purpose:** To ensure all stakeholder groups are proficient in using the new application.
* **Actions:**
  + **Role-Specific Training:**
    - **Employee Session:** Train users on accessing the Expense Tracker' Lightning App , submitting expenses using the Screen Flow , and understanding the Approval Status picklist.
    - **Manager Session:** Train managers on monitoring the Expense Tracker Dashboard , and approving/rejecting submissions.
  + **Go-Live Activation:**
    - Formally activate the Lightning App and Lightning Record Pages as the default.
    - In Production, schedule the automated Apex jobs: ScheduleOverdueExpenses and DailyExpenseSummary.
    - Send the official launch communication.

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

* **Purpose:** To demonstrate the full, end-to-end business process and showcase its value.
* **Actions:**
  + **Pitch Presentation:**
    - Restate the core problem from Phase 1: inefficient, manual reporting.
    - Present the solution: a unified, automated platform.
    - Emphasize key value propositions:
      * **Data Integrity:** Showcase Validation Rules.
      * **Process Efficiency:** Highlight the automated Approval Process.
      * **Actionable Insights:** Feature the Expense Tracker Dashboard.
  + **Live Demo Walkthrough:**
    - **Employee Persona:** Log in, navigate to the app, and submit a new expense.
    - **Manager Persona:** Log in, show the expense on the dashboard, and approve it.
    - **Finance/Admin Persona:** Refresh the dashboard to show the newly approved data.

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

* **Purpose:** To gather immediate feedback and establish a "hypercare" period for support.
* **Actions:**
  + **Feedback Mechanism:** Distribute a simple survey to all trained users.
  + **Monitor and Support:**
    - Establish a clear support channel for the first 30 days.
    - Admins actively monitor for any automated error notifications from Flows or Apex.
    - Review the Setup Audit Trail to ensure system health.

**Step 4: Deliver Final Handoff Documentation**

* **Purpose:** To equip the System Administrator with all documentation to maintain the app.
* **Actions:**
  + **Configuration Guide:**
    - Provide a full data dictionary for the Expense\_\_c object.
    - Document all automation logic (Validation Rules, Flows, etc.).
  + **Technical (Apex/LWC) Guide:**
    - Hand off all Apex classes, explaining the logic in ExpenseService , ExpenseTrigger , BatchOverdueExpenses , and ExpenseController.
    - Provide source code for all custom LWCs.
  + **Integration & Data Guide:**
    - Document all Named Credentials and the ExpenseAPIService.
    - Explain the Duplicate Rule and Data Export schedule.
  + **Deployment Package:**
    - Provide the final Outbound Change Set or SFDX Package.

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

* **Purpose:** To translate the completed project into a professional asset.
* **Actions:**
  + **Build Case Study (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.
      * Implementing complex business logic via Flows, Apex Triggers, and Batch Apex.
      * Creating real-time reports and dashboards.
      * Managing the full data and deployment lifecycle."
    - **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. All documentation has been handed off, and the project is formally closed.