Salesforce Project Implementation Phases with Concepts (Admin + Developer)

TITLE: EMPLOYEE RECOGNITION AND REWARDS TRACKER

Problem Statement  
Many organizations rely on manual or spreadsheet-based processes to track employee achievements and reward points, leading to delays, errors, and lack of transparency. Managers and HR struggle to monitor performance effectively, and employees often miss timely recognition. The proposed Employee Recognition & Rewards Tracker in Salesforce will centralize achievements, automate notifications, calculate reward points, and provide real-time dashboards to enhance engagement and motivation.

# Phase 1: Problem Understanding & Industry Analysis

* Objective:

To understand the current employee recognition process, identify challenges, and define requirements for a Salesforce system that tracks achievements, manages reward points, and improves engagement.

* Requirement Gathering

1. Identify what the organization wants to achieve with the recognition system.
2. Track employee achievements and milestones.
3. Award and manage reward points.
4. Send notifications for recognitions and milestones.
5. Generate performance reports and dashboards.

* Stakeholder Analysis

1. HR Managers: Define recognition criteria and manage the reward system.
2. Team Leads / Managers: Nominate employees for recognition, approve rewards.
3. Employees: View own achievements, points, and leaderboard rankings.

* Business Process Mapping

1. Document the current manual process (if any) for employee recognition.
2. Identify pain points such as:

\*Delayed or missed recognition.

\*Manual calculations of points.

\*Lack of visibility for employees and managers.

* Industry-specific Use Case Analysis

1. Research how other organizations implement recognition and reward programs.
2. Identify best practices: automated notifications, point-based rewards, dashboards, and leaderboards.

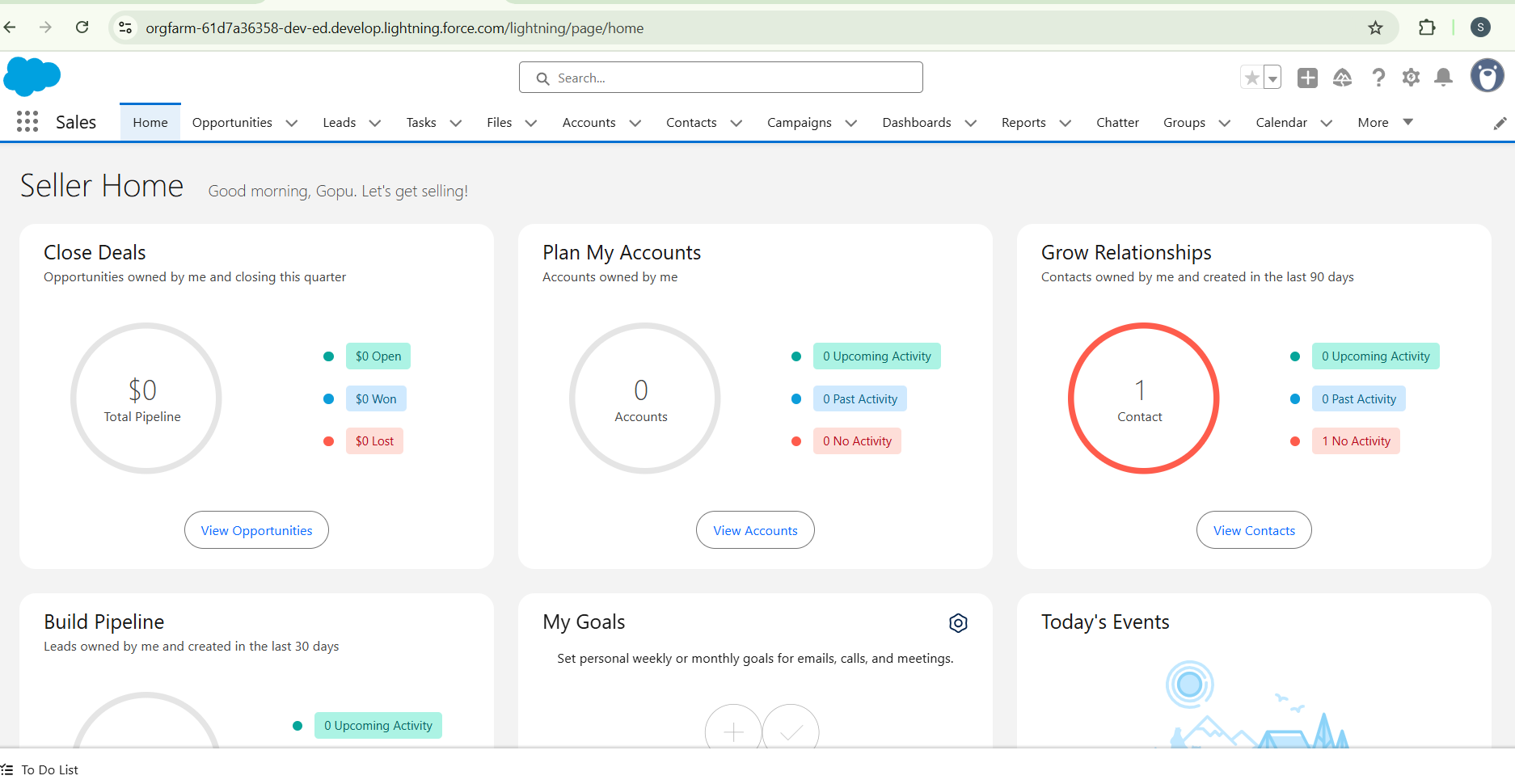
* AppExchange Exploration

1. Explore existing Salesforce apps for employee recognition.
2. Identify features that can be included or improved in your custom solution.

# Phase 2: Org Setup & Configuration

* Salesforce Editions

1. **Edition Used:** Salesforce **Developer Edition**
2. Provides free access to: Custom Objects, Apex, Flows, Reports, Dashboards, and security features.
3. Suitable for testing and academic/demo projects.

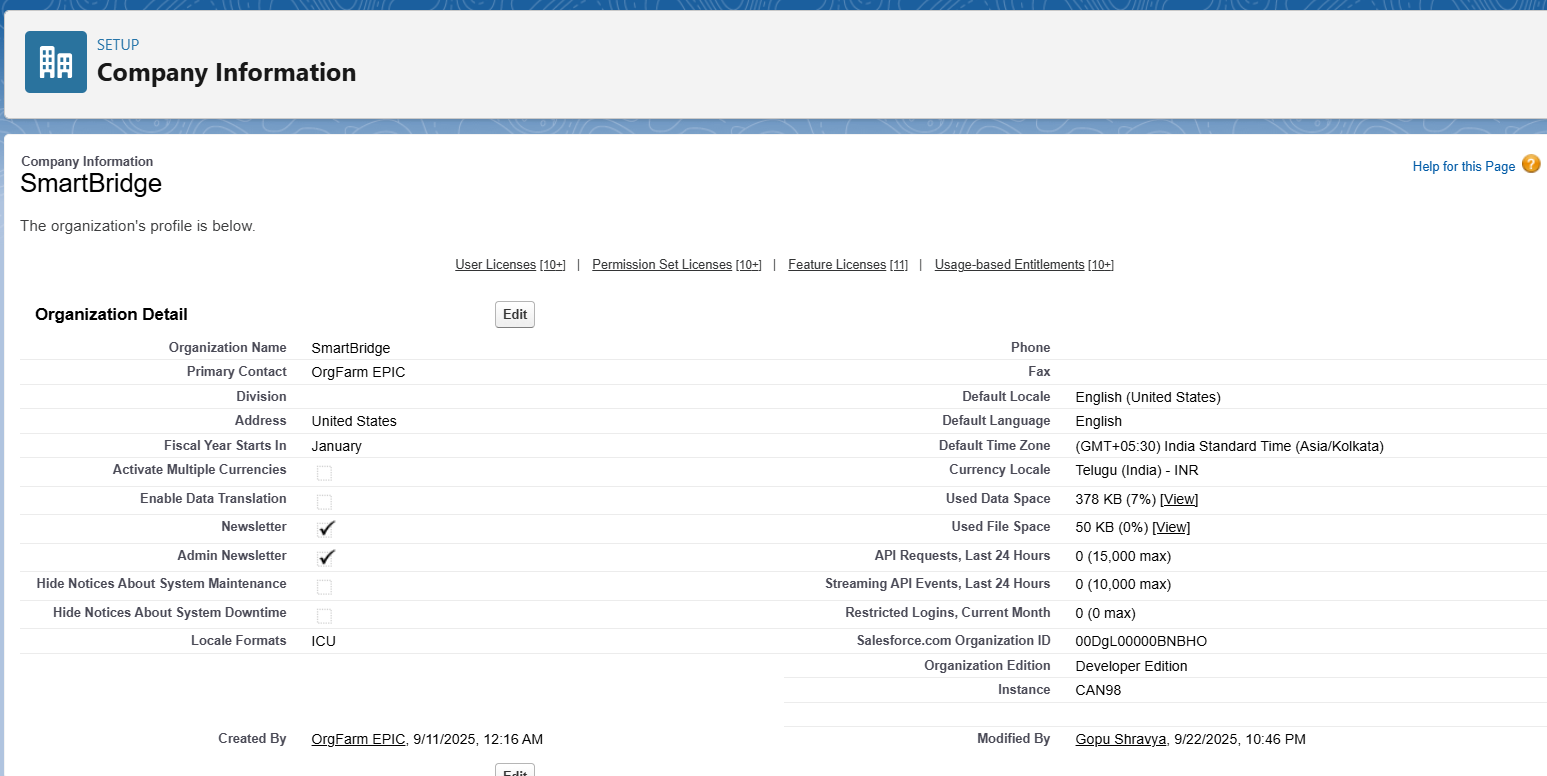


* Company Profile Setup

1. Configured company information:

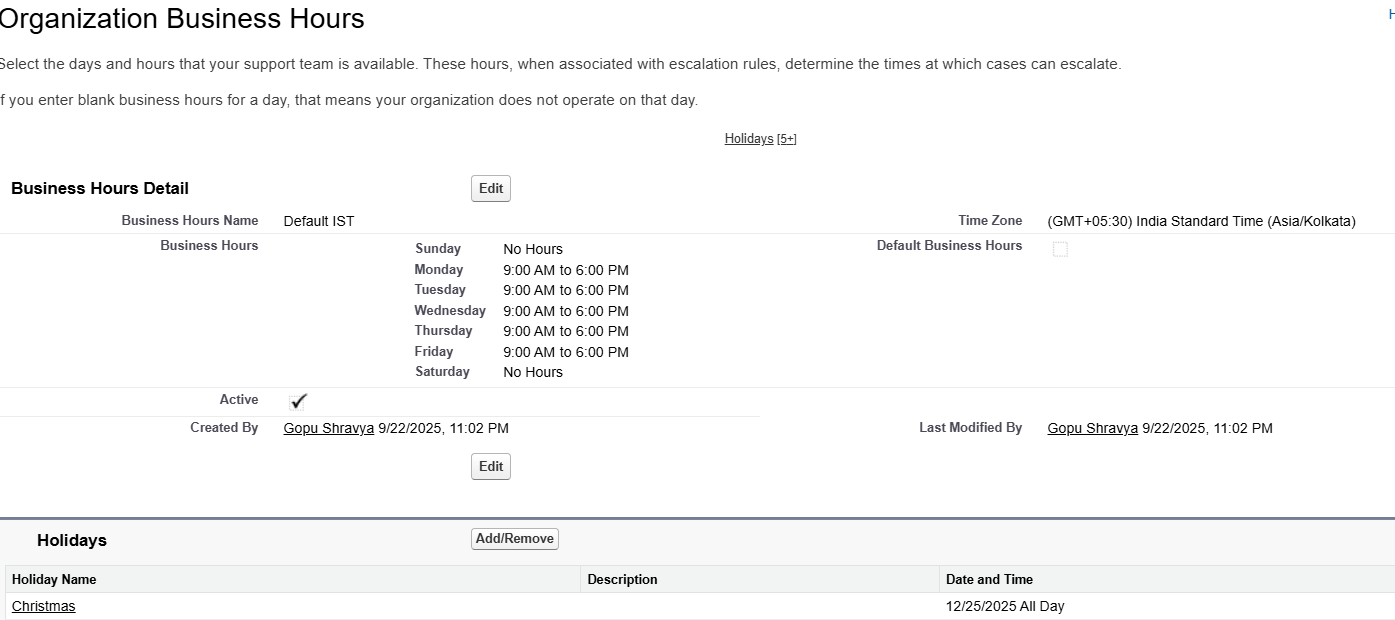
* **Company Name**
* **Default Time Zone**
* **Locale**
* **Currency**

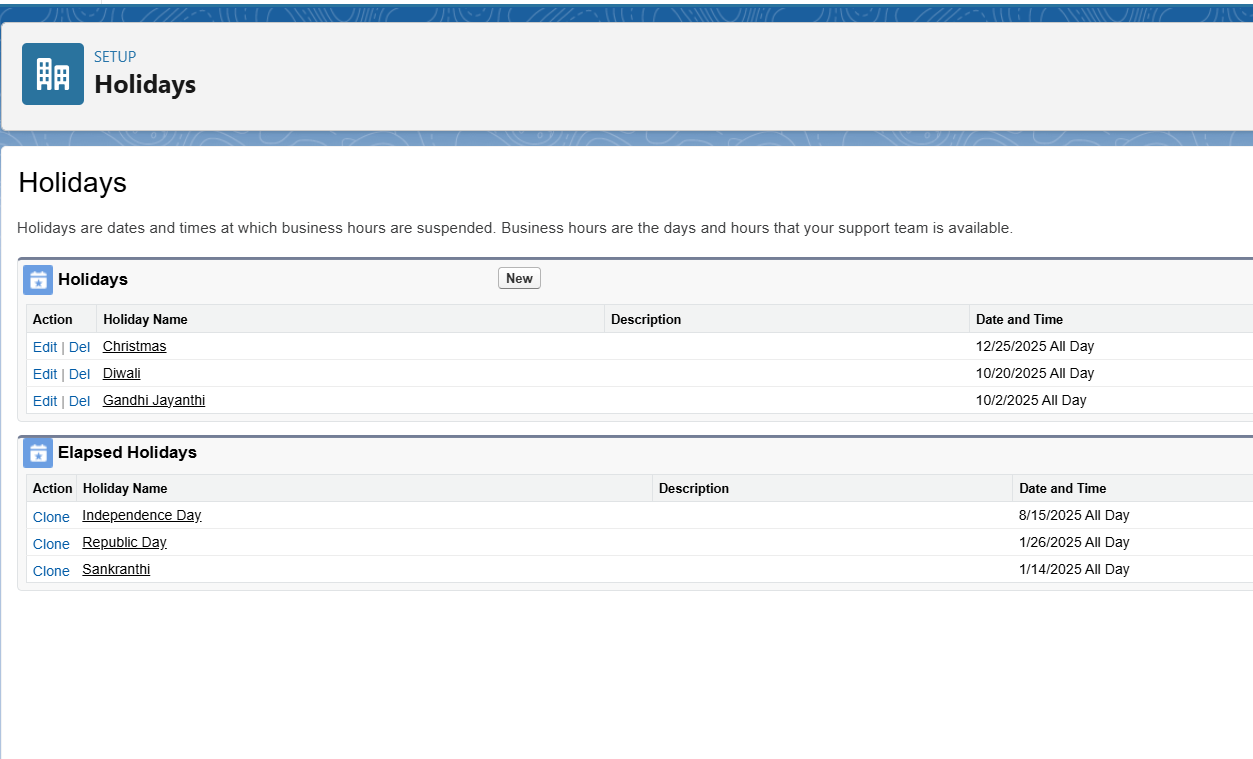
1. This ensures consistency across the organization.



* Business Hours & Holidays

1. Business Hours: **9:00 AM – 6:00 PM**
2. Holidays configured: Republic Day, Diwali, etc.
3. Helps in escalation rules and SLA calculations.





* Fiscal Year Settings

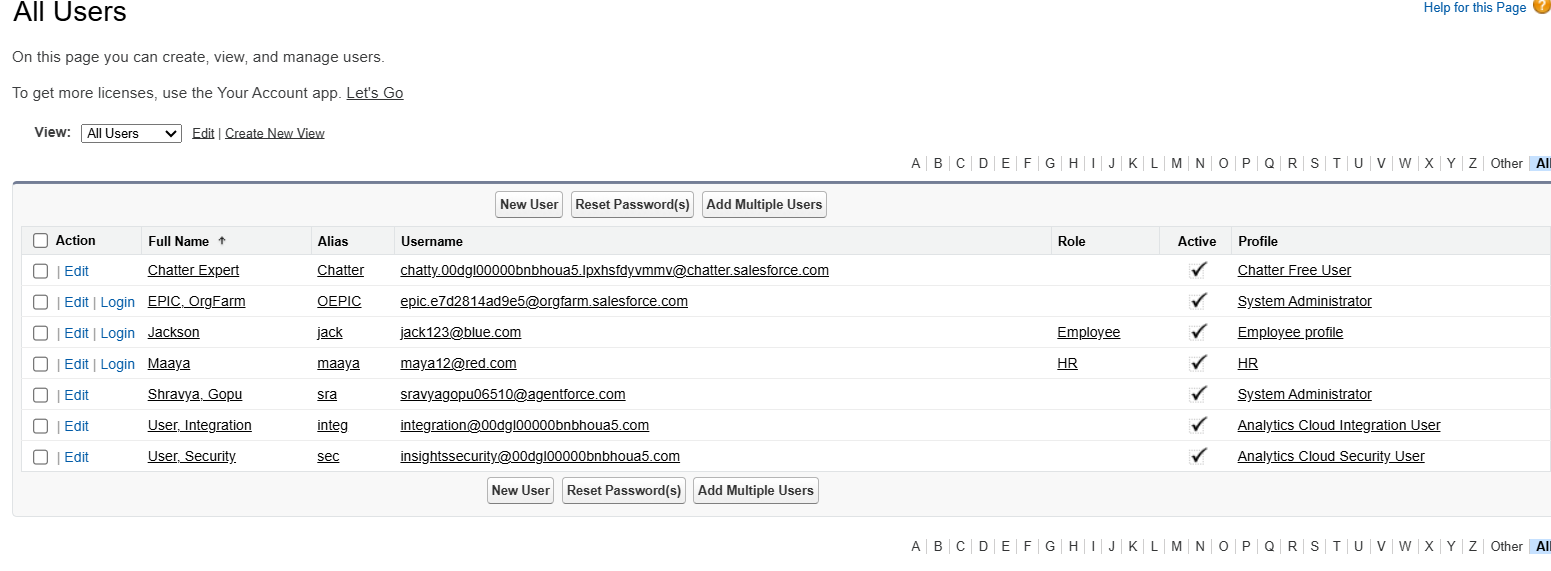
1. We are using Standard Fiscal Year (Jan–Dec). If the company requires April–March, this can be switched to Custom Fiscal Year.

* User Setup & Licenses

1. Users created with different roles:

* **Employee User** – Employee Profile
* **HR User** – HR Profile

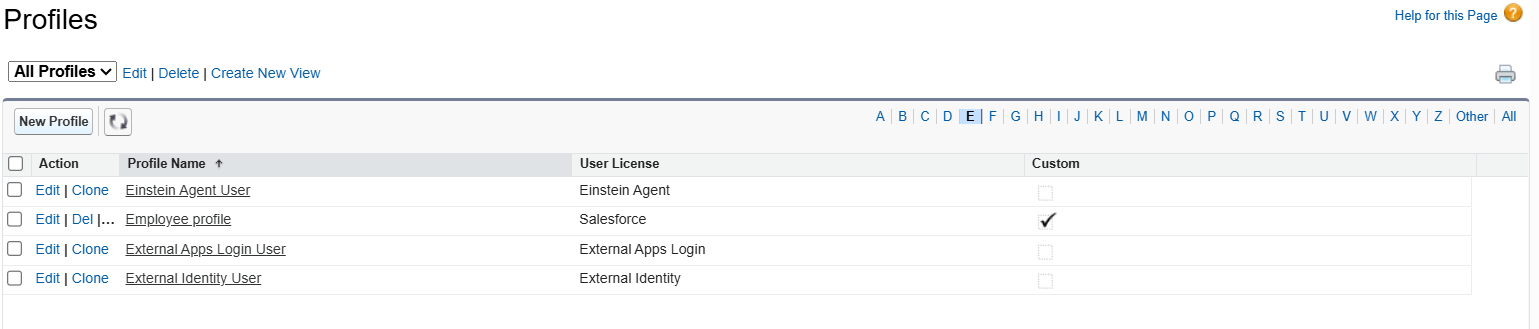
1. Licenses assigned: **Salesforce License**

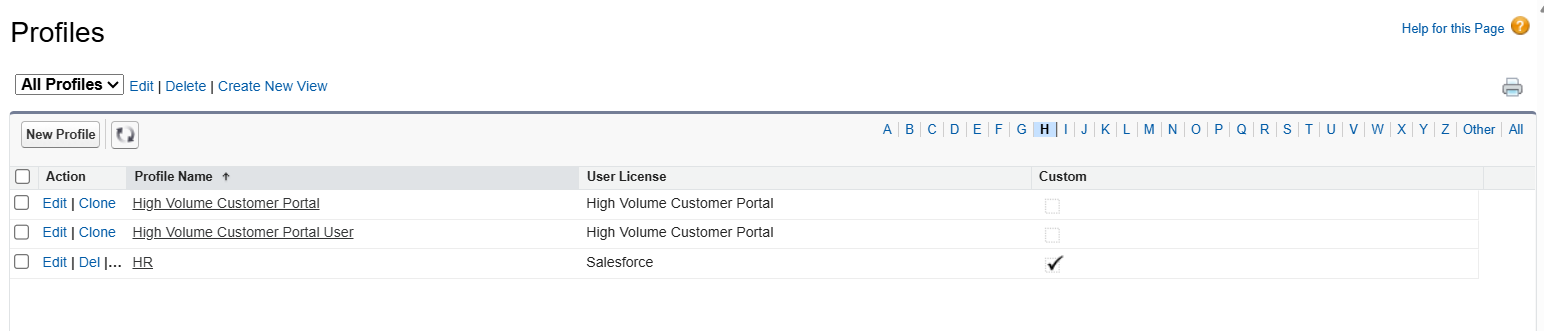


* Profiles

1. Profiles define object-level permissions.
2. Created:

* **Employee Profile** – Read-only on own records
* **HR Profile** – Full access to Employee & Recognition



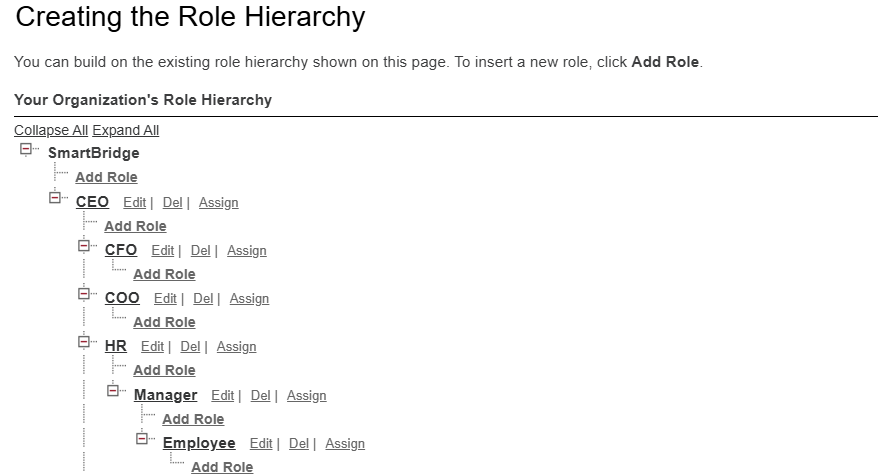


* Roles

1. Role Hierarchy ensures record visibility:

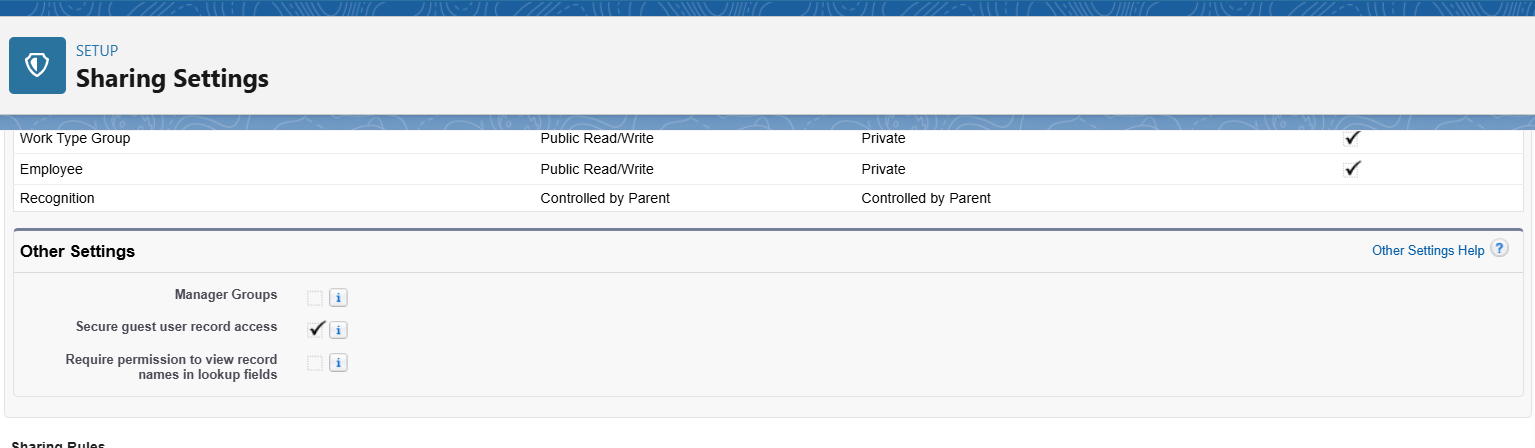
* CEO → HR → Manager → Employee

1. Managers and HR can view team employees’ recognitions.



* OWD

1. **Employee Object** → Public Read/Write
2. **Recognition Object** → Controlled by Parent
3. Provides secure record visibility.



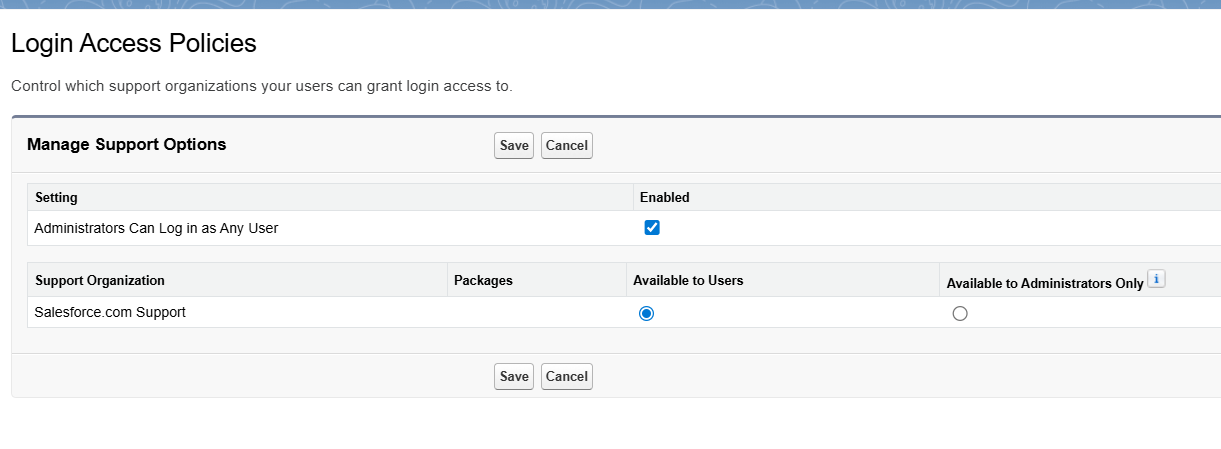
* Sharing Rules

1. Configured **sharing rules** so:

* Employees only see their own recognitions
* HR/Managers can see records of their subordinates
* **Implementation:** Achieved using **Org-Wide Defaults (Private)** on the Recognition object + **Role Hierarchy**. (No custom rule is created).

* Login Access Policies

1. Enabled **Grant Login Access** so Admin can log in as Employee/HR for troubleshooting.



* Dev Org Setup

1. Project is built on **Salesforce Developer Org**.
2. All configurations, Apex triggers, and flows tested here.

* Sandbox Usage

1. For enterprise deployment, a **Sandbox** is used for testing before moving to production.
2. In this project, we worked directly on Developer Org (acts as sandbox).

* Deployment Basics

1. Deployment methods available in Salesforce:
   1. Change Sets
   2. ANT Migration Tool
   3. Salesforce CLI (SFDX)
2. For this project: Direct configuration in Developer Org.

# Phase 3: Data Modeling & Relationships

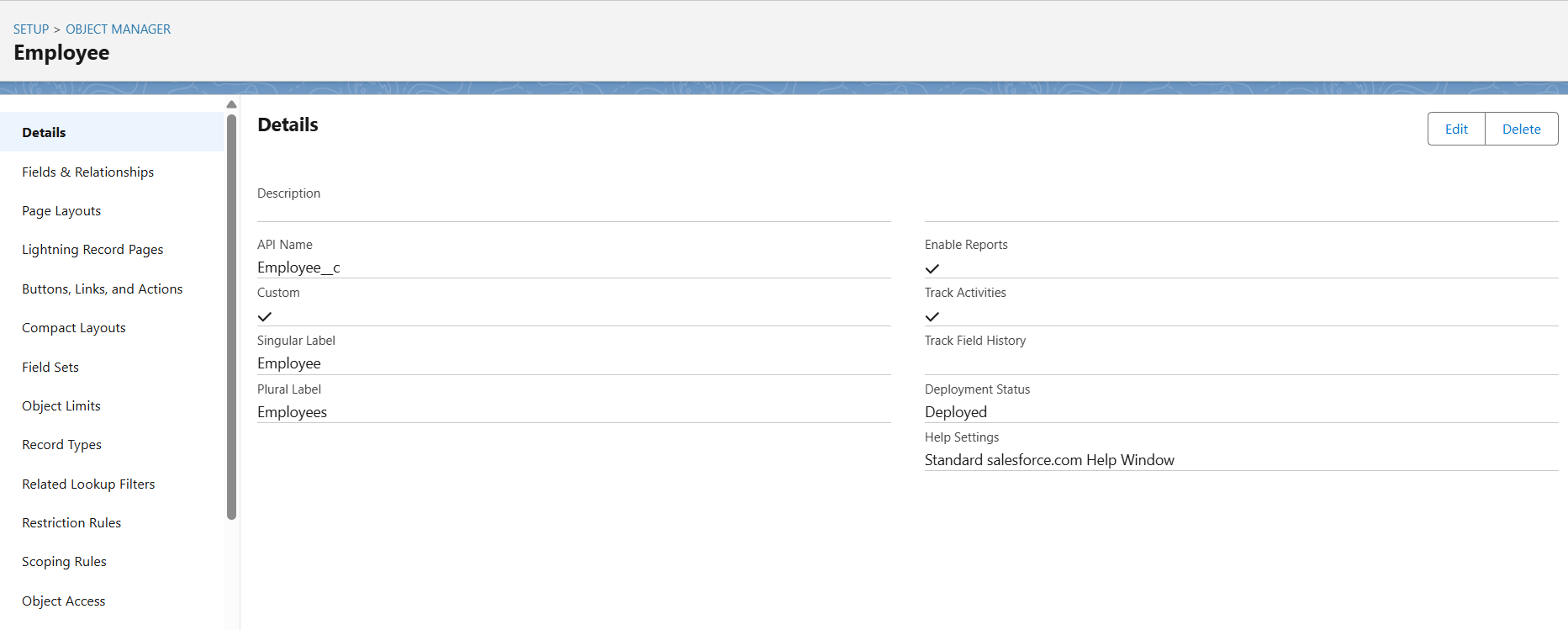
* Standard & Custom Objects

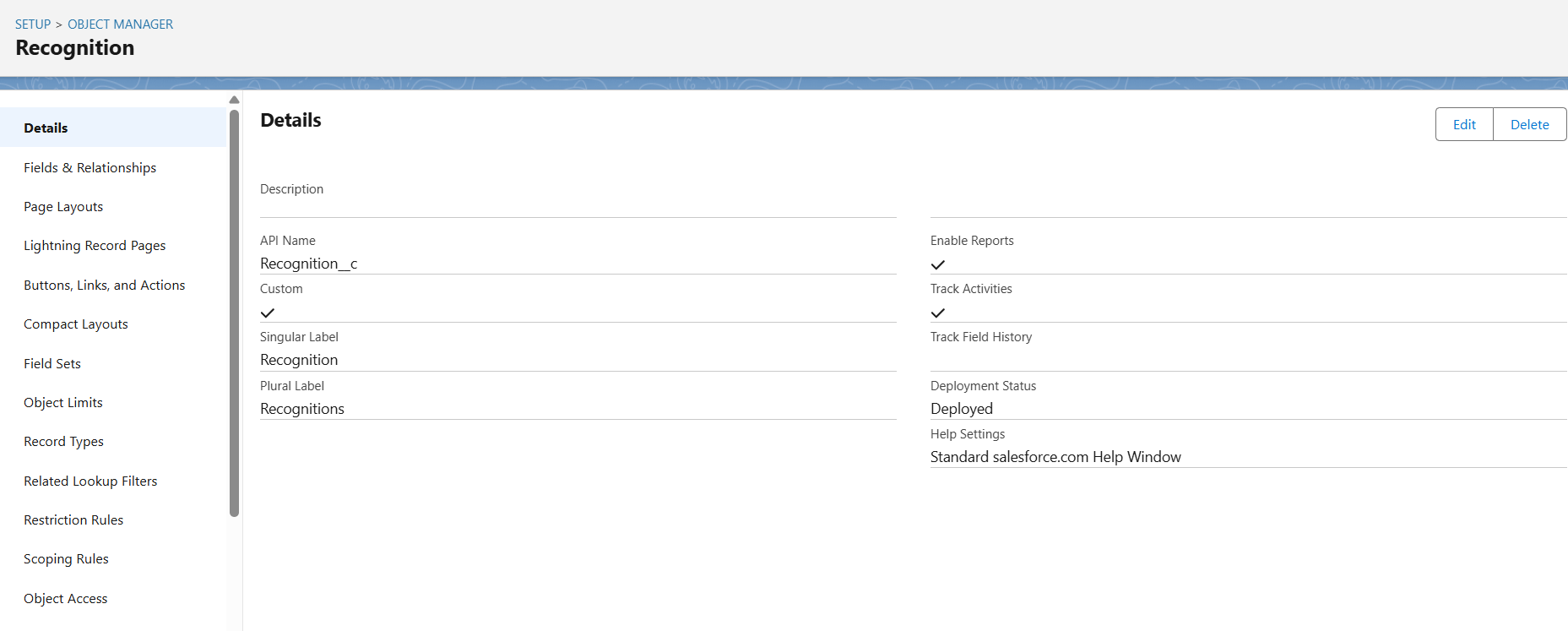
**Standard Objects:** Pre-built objects like User, Profile, and Role.

* Used to manage HR users, Managers, and their roles.

**Custom Objects:** Created specifically for this project:

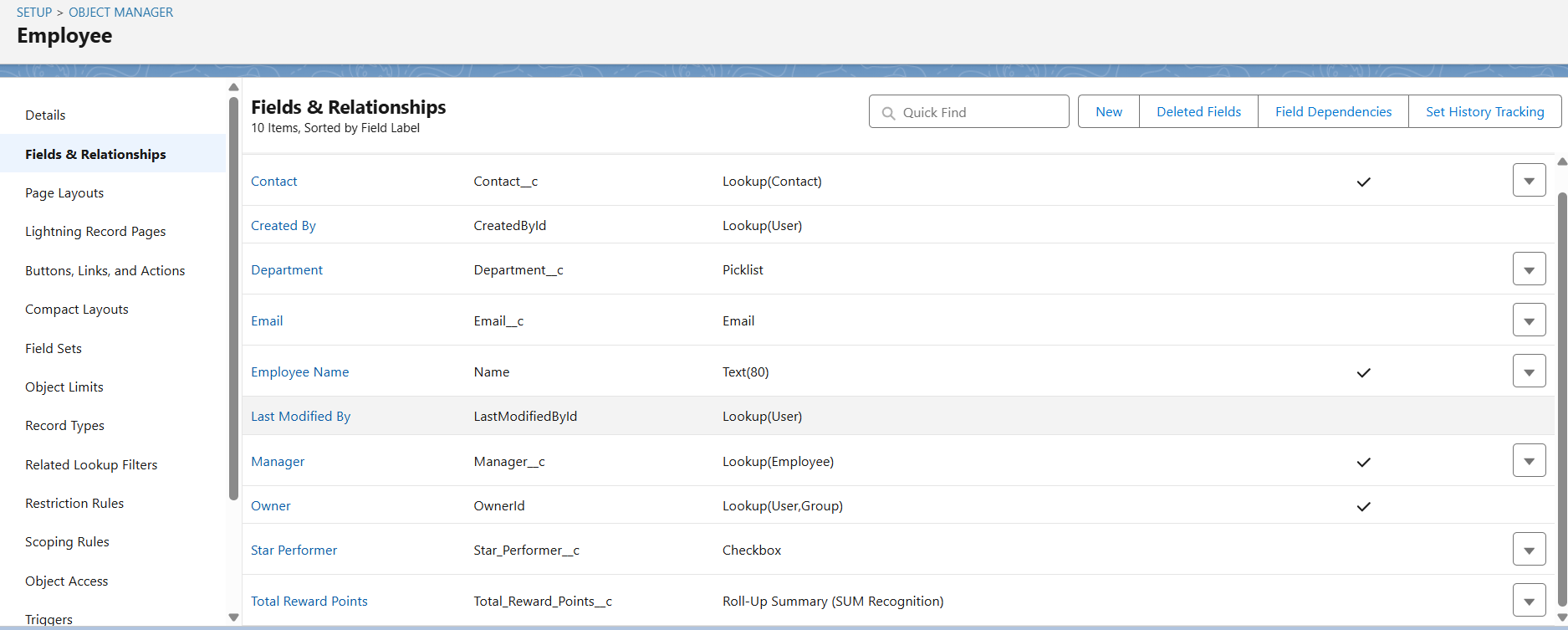
* Employee\_\_c – Stores employee details.
* Recognition\_\_c – Stores recognition records linked to employees.

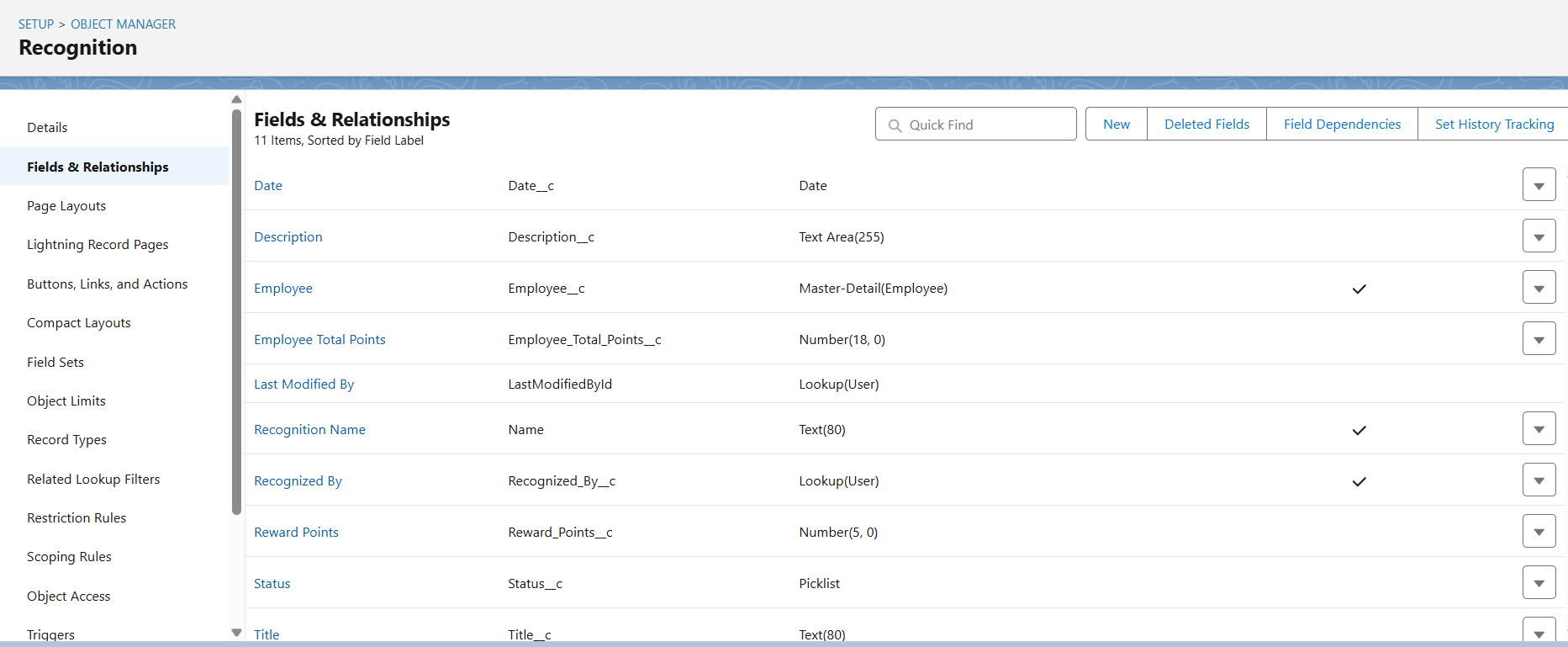




* Fields

1. Custom fields on **Employee\_\_c**: Department, Star Performer, Total Reward Points.
2. Custom fields on **Recognition\_\_c**: Title, Description, Reward Points, Employee Lookup.
3. Fields are of types **Text, Number, Lookup** depending on the requirement.



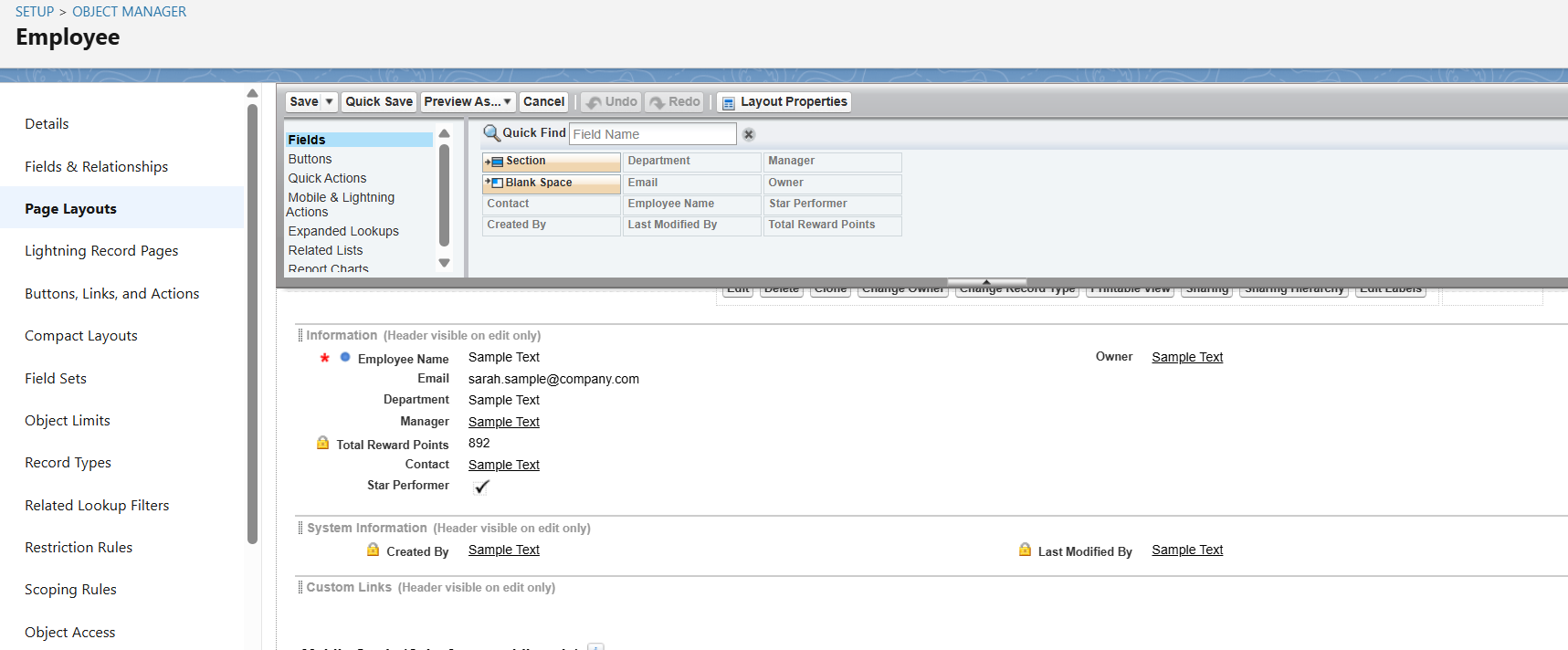


* Record Types

1. Not strictly needed unless different types of recognitions are tracked (e.g., “Monthly”, “Spot Award”).
2. Helps in controlling **page layouts** for different recognition types.

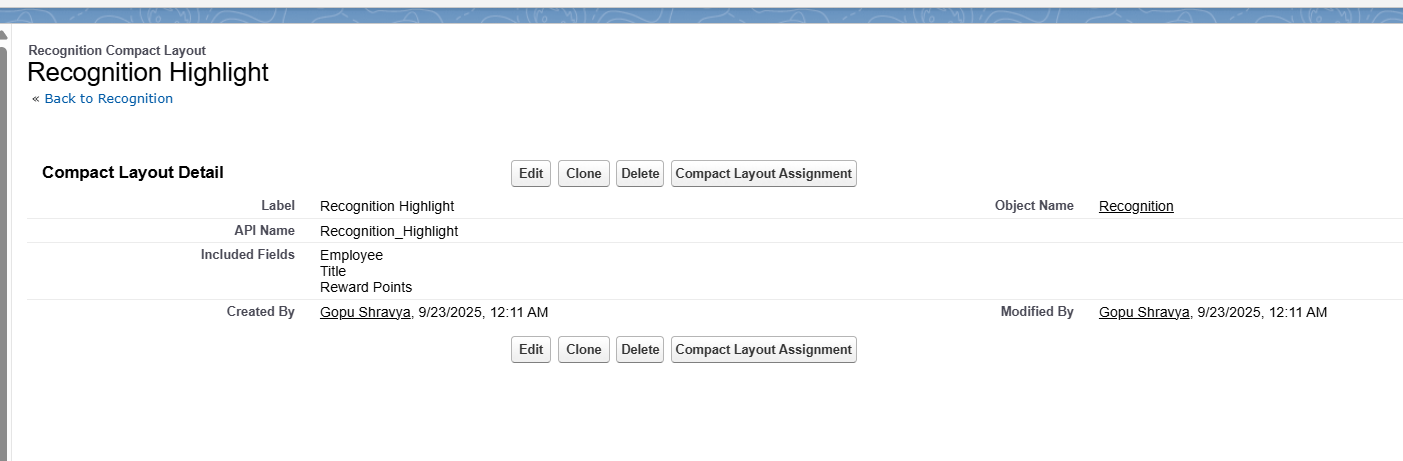
* Page Layouts

1. Page layouts control which fields appear on the record page.
2. Example: Employee layout shows Name, Department, Total Reward Points.
3. Recognition layout shows Title, Description, Employee Lookup, Reward Points.



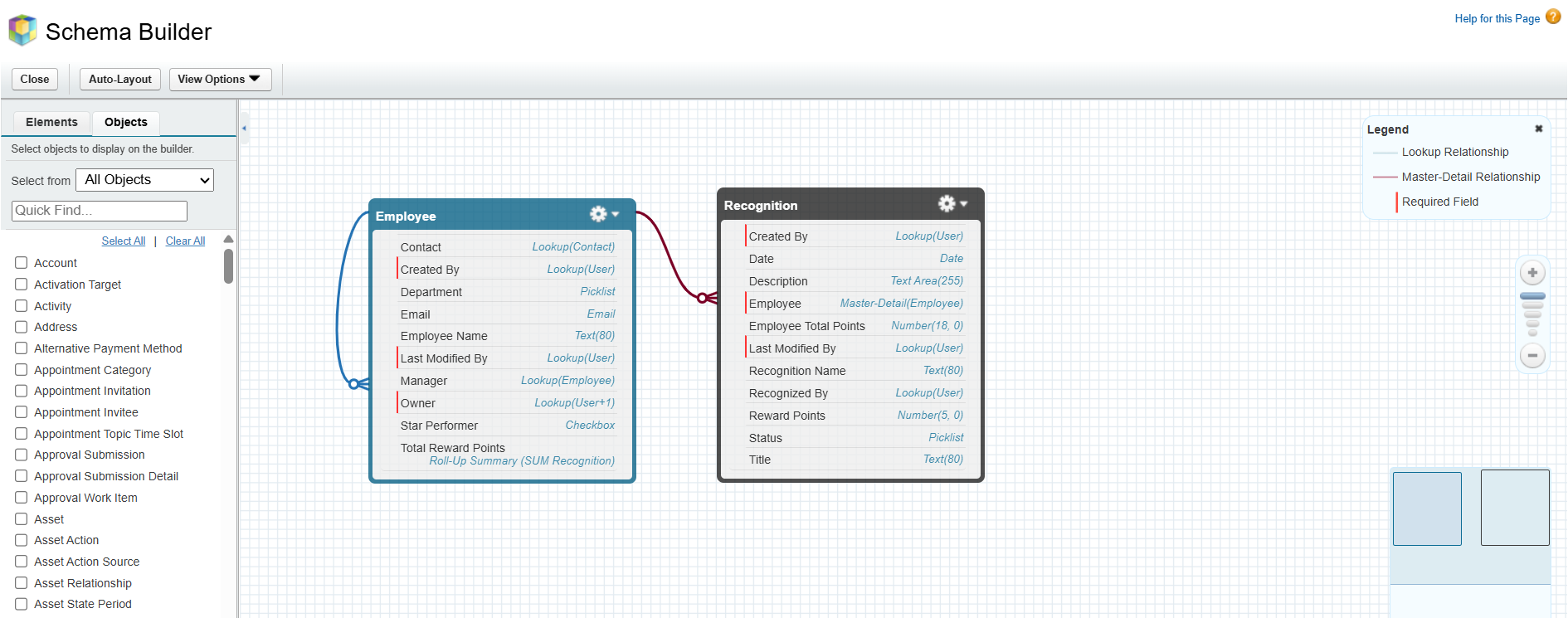
* Compact Layouts

1. Compact layouts determine the **key fields visible in record highlights or mobile**.
2. Example: Recognition compact layout shows Title,employee and Reward Points.



* Schema Builder

1. Visual tool to **see objects and relationships**.
2. Shows how Recognition\_\_c is linked to Employee\_\_c via a lookup.



* Lookup vs Master-Detail vs Hierarchical Relationships

1. **Employee → Recognition:** Lookup Relationship (not Master-Detail)

* Each recognition belongs to an employee.
* Allows HR/Managers to see subordinates’ recognitions without deleting employee when a recognition is deleted.

1. **Master-Detail:** Not used here, since deletion of employee should not necessarily delete recognition records automatically.

* Junction Objects

1. Junction objects are used for **many-to-many relationships**.
2. In this project, not required as recognition is **one-to-many** (Employee → multiple Recognition records).

* External Objects

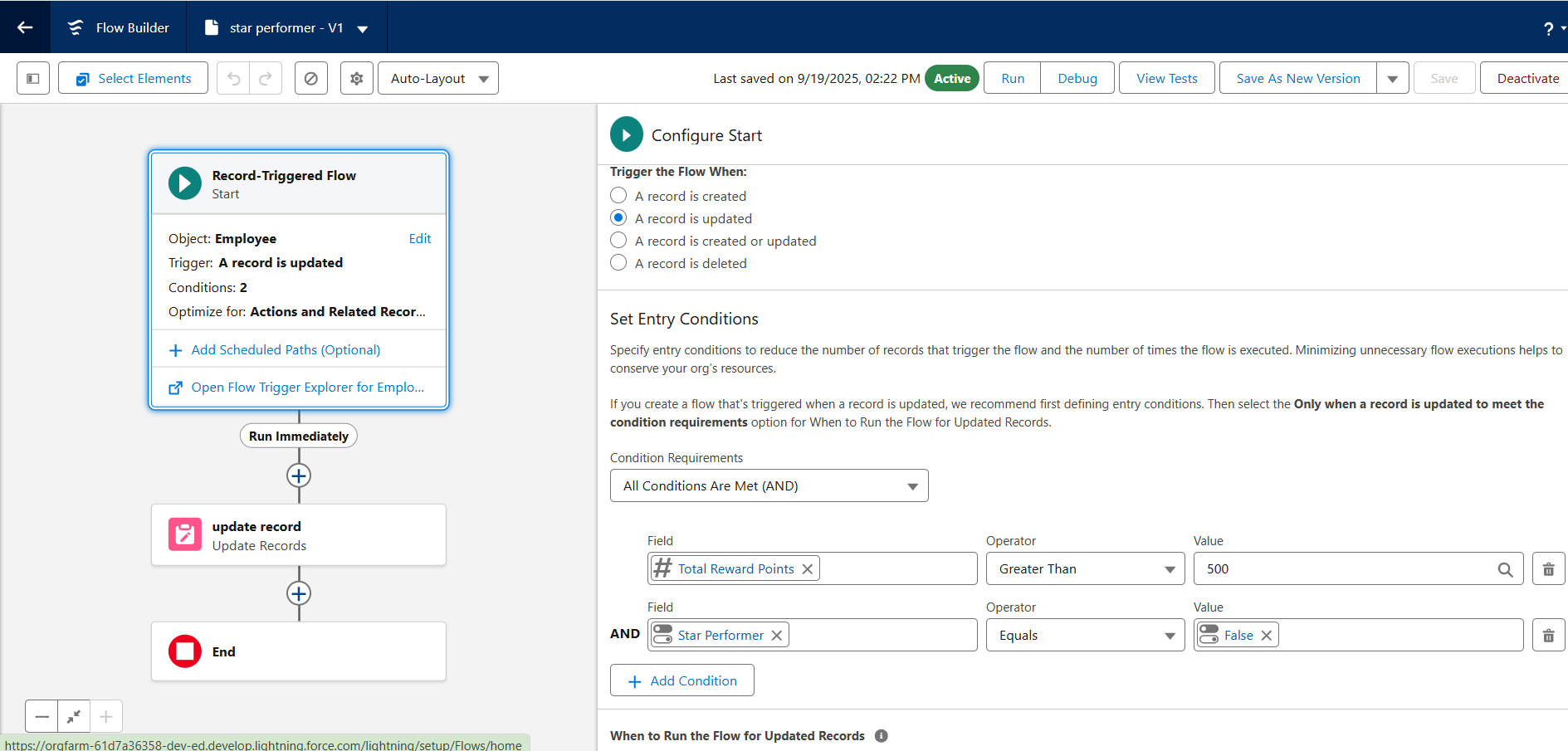
1. Used to integrate **external data sources**.
2. Not used in this project; all data stored in Salesforce objects.

# Phase 4: Process Automation (Admin)

* **Autolaunched Flow – Star Performer Flag**

1. **Purpose:** Automatically updates the Star Performer status based on Employee’s total reward points. **Project Implementation:**

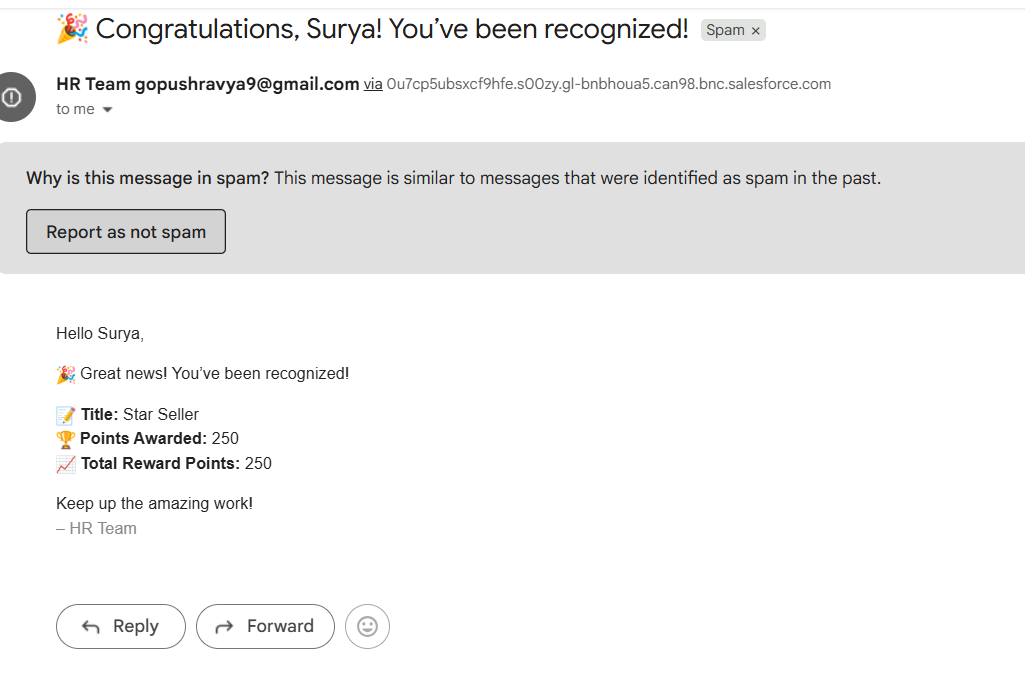
* Invoked by the Record-Triggered Flow.



* **Apex Email Notifications**

1. **Purpose:** Automatically notify users about key events.  
   **Project Implementation:**

* Sends email notifications to **Employees** whenever a new Recognition is created.
* Ensures that all stakeholders are informed in real-time.

****

* **Profiles, Roles & Sharing**

1. **Purpose:** Control access to records and Lightning pages without creating additional sharing rules.  
   **Project Implementation:**

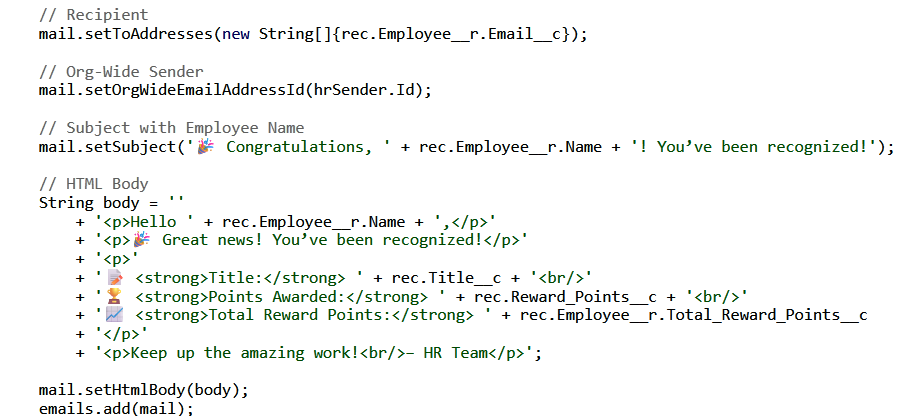
* **Role Hierarchy:** HR and Managers can view recognition records of their subordinates.
* **Employee Access:** Employees can view only their own recognition records.
* Tabs and Lightning pages are assigned appropriately to each profile.

# Phase 5: Apex Programming (Developer)

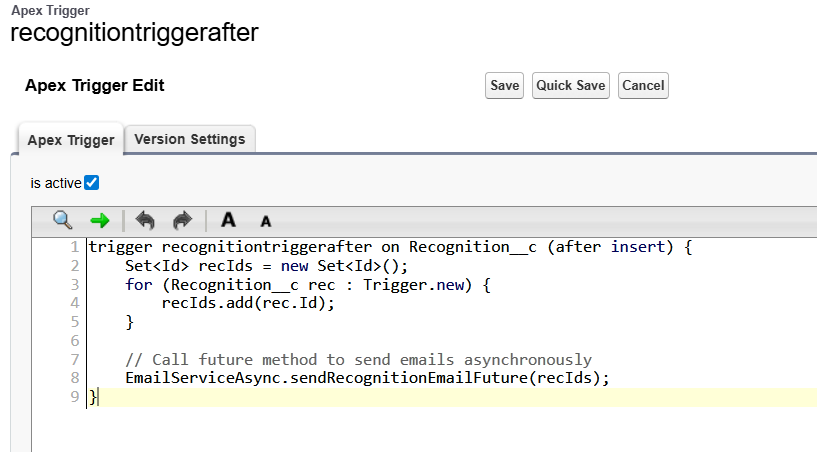
* Classes & Objects

**1. Apex Classes**

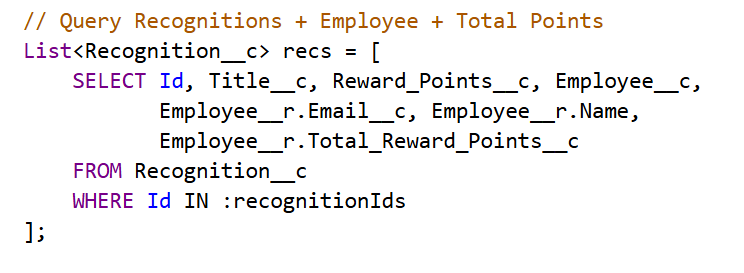
* **Purpose:** Encapsulate business logic for recognition emails.  
  **Project Implementation:**
* **EmailService** → Sends recognition emails to employees with their Name, Title, Reward Points, and Total Reward Points.
* **EmailServiceAsync** → @future method to send emails asynchronously to avoid governor limits.



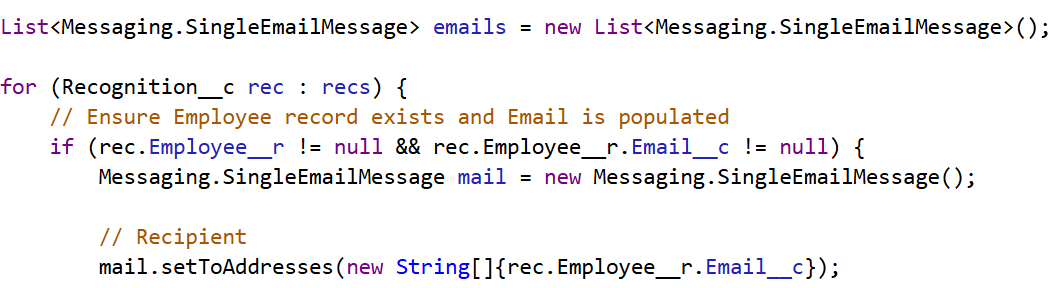
* Apex Triggers (after insert)
* **Purpose:** Automate updates and email notifications when Recognition records are created.  
  **Project Implementation:**
* **Trigger Name:** recognitiontriggerafter
* **Event:** after insert on Recognition\_\_c
* **Functionality:**
  + Collect Recognition IDs in a Set
  + Call **EmailServiceAsync.sendRecognitionEmailFuture**
  + Updates **Total Reward Points** and **Star Performer Flag**



* Trigger Design Pattern
* **Purpose:** Separate trigger logic from business logic for scalability and maintainability.  
  **Project Implementation:**
* One trigger per object (Recognition\_\_c)
* All business logic delegated to Apex classes (EmailService / EmailServiceAsync)
* SOQL & SOSL
* **Purpose:** Retrieve records from Salesforce objects.  
  **Project Implementation:**
* Query Recognition\_\_c and related Employee\_\_c records for email, name, and total points in EmailServiceAsync.



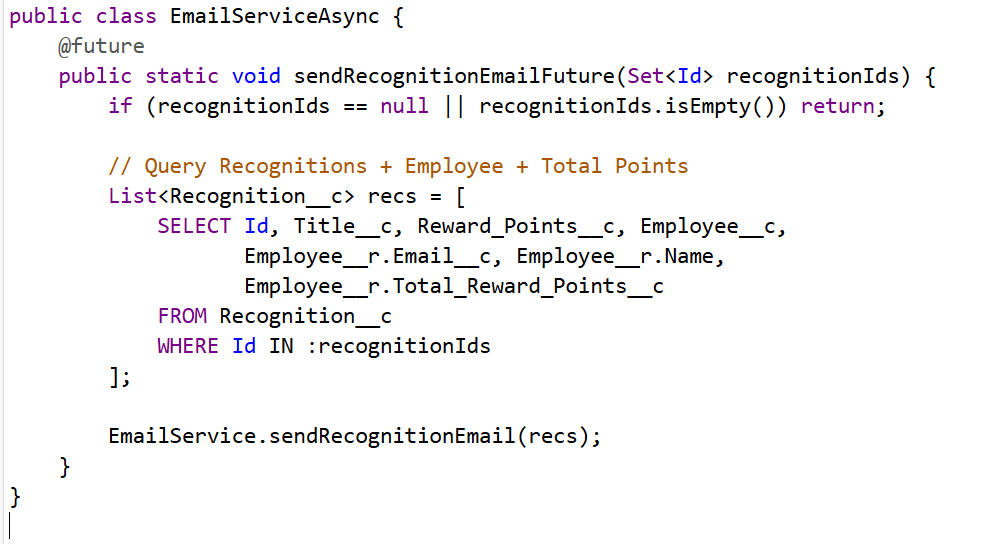
* Collections: List, Set, Map
* **Purpose:** Efficiently handle multiple records.  
  **Project Implementation:**
* **Set<Id>** → Recognition IDs in trigger
* **List<Messaging.SingleEmailMessage>** → Sending emails



* Exception Handling
* **Purpose:** Avoid runtime errors during email sending.  
  **Project Implementation:**
* Null checks for recognition list and employee email before sending emails:



* Asynchronous Processing
* **Purpose:** Perform email sending out-of-transaction to avoid limits.  
  **Project Implementation:**
* **@future method** in EmailServiceAsync
* Trigger calls future method after insert

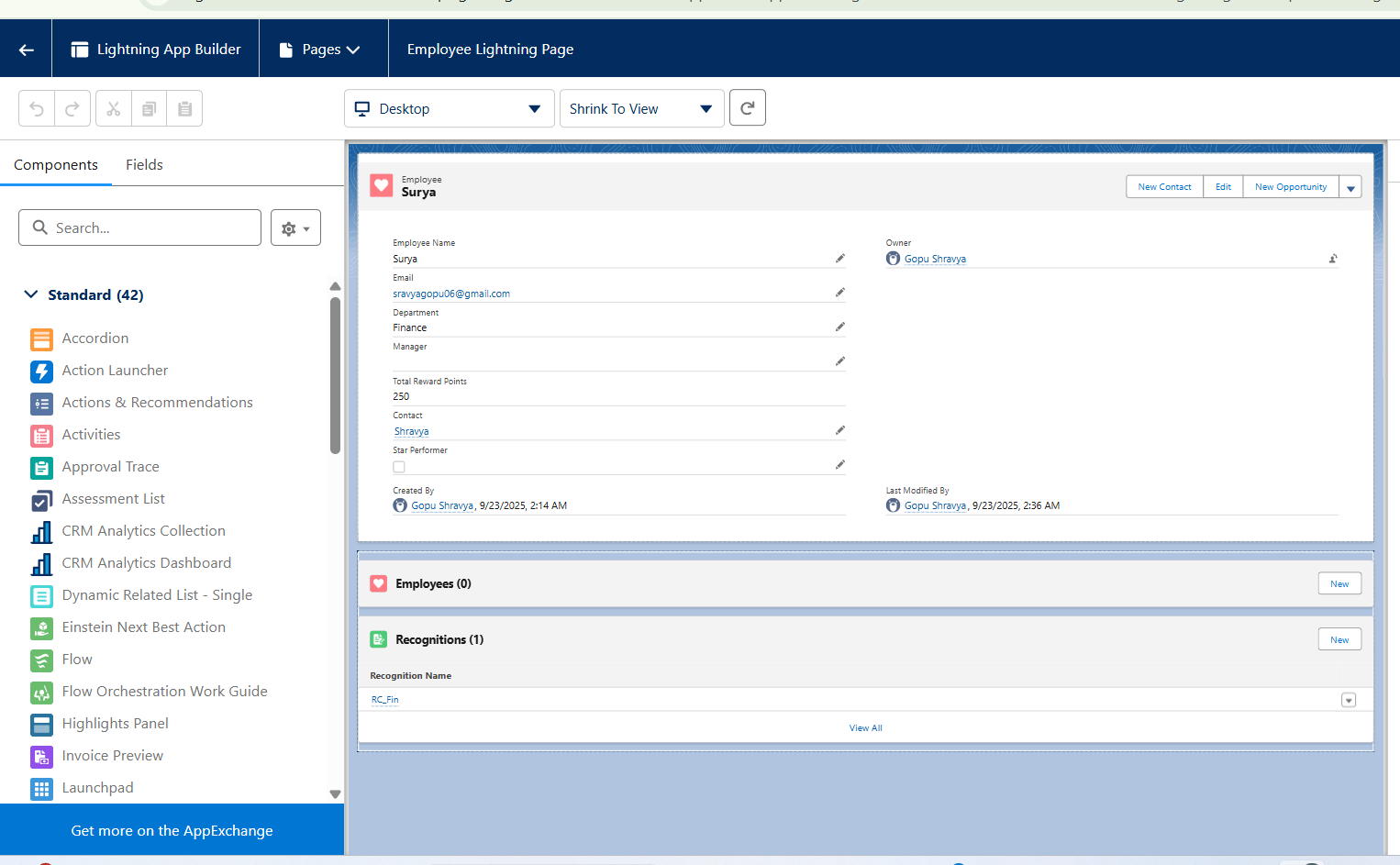


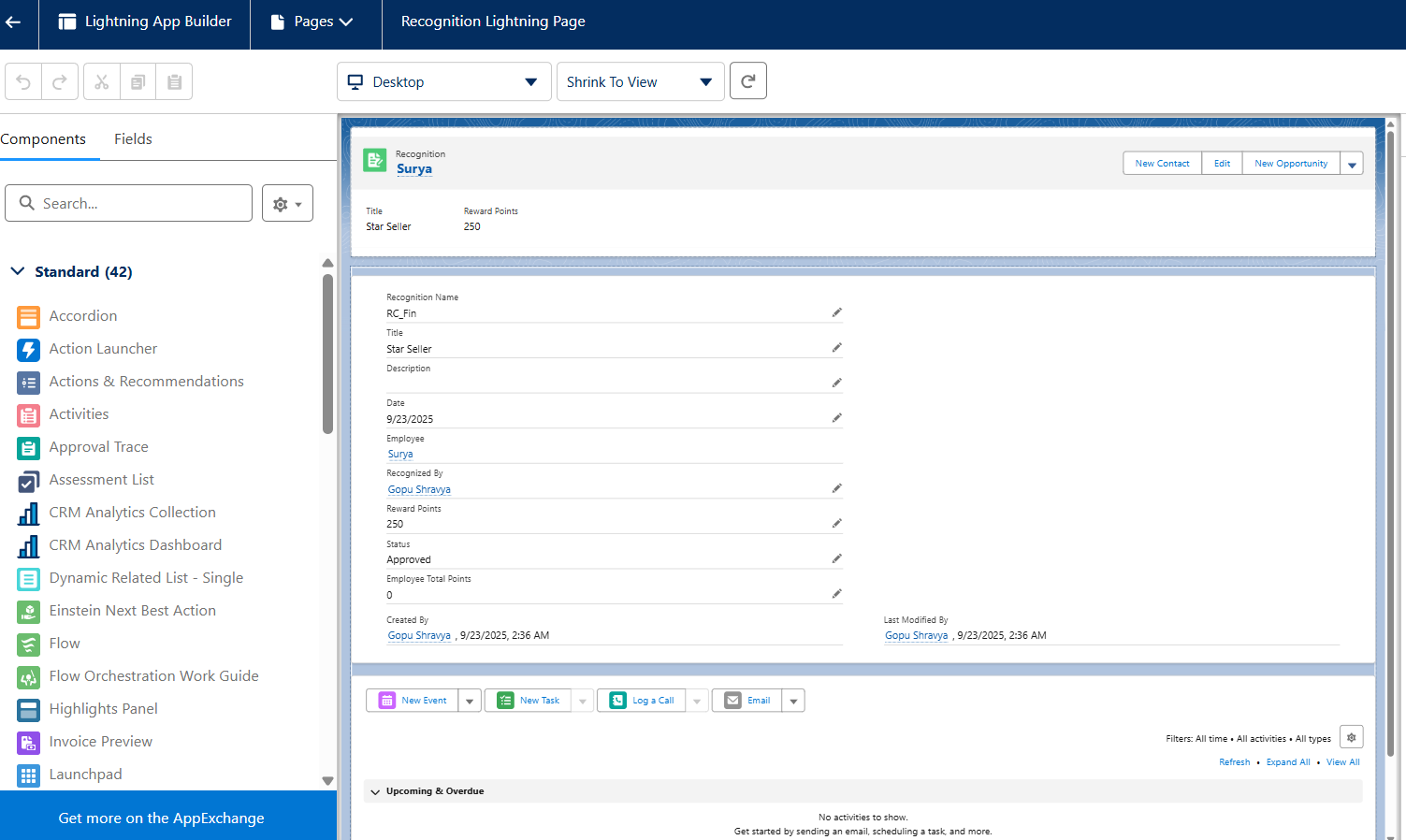
# Phase 6: User Interface Development

* Lightning App Builder

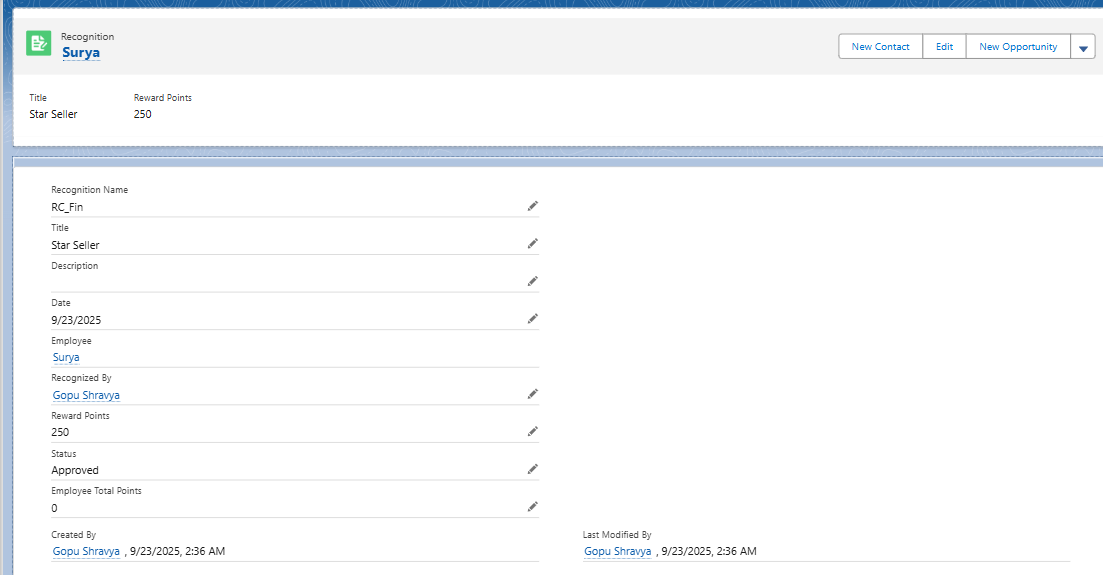
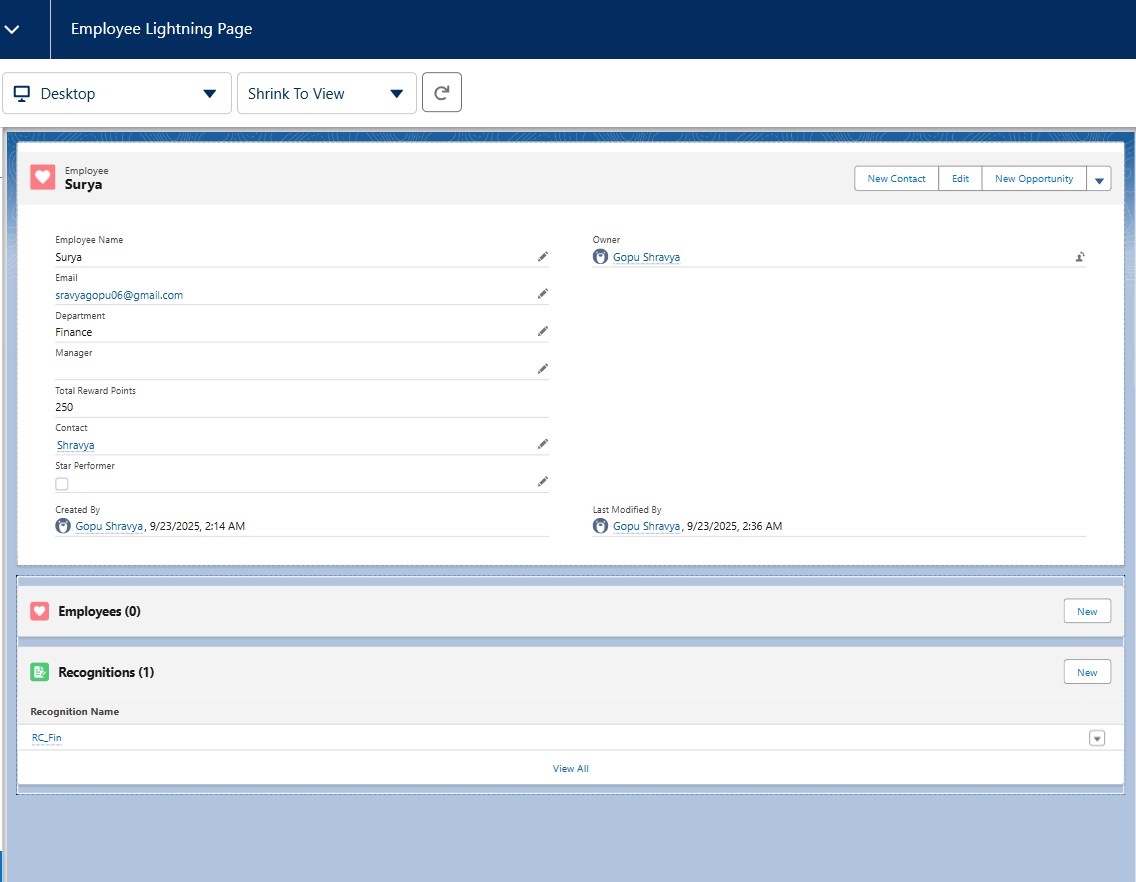
1. **Purpose:** Customize how users interact with records in Lightning.  
   **Project Implementation:**

* Created **custom record pages** for **Employee** and **Recognition** objects.
* Included important fields: Total Reward Points, Star Performer Flag, Recognition Title, Points, etc.
* Ensured **HR, Managers, and Employees** see only what’s relevant.

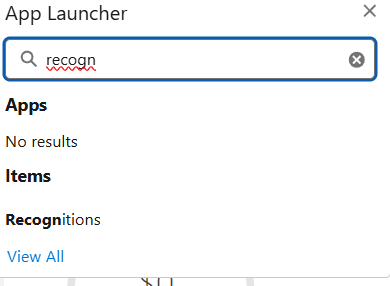
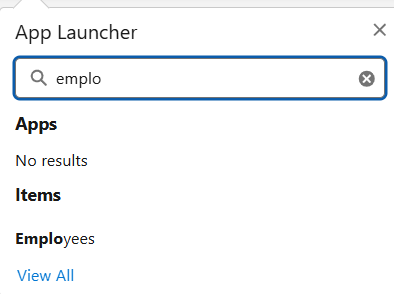




* Record Pages
* **Purpose:** Organize and display object records effectively.  
  **Project Implementation:**
* Employee Page: Shows employee details, total reward points, and list of recognitions.
* Recognition Page: Shows recognition title, points, and associated employee.



* Tabs
* **Purpose:** Access objects quickly through the Lightning App.  
  **Project Implementation:**
* Added **Employee** and **Recognition** tabs in the custom Lightning App.
* Tabs assigned to appropriate profiles (HR, Manager, Employee).

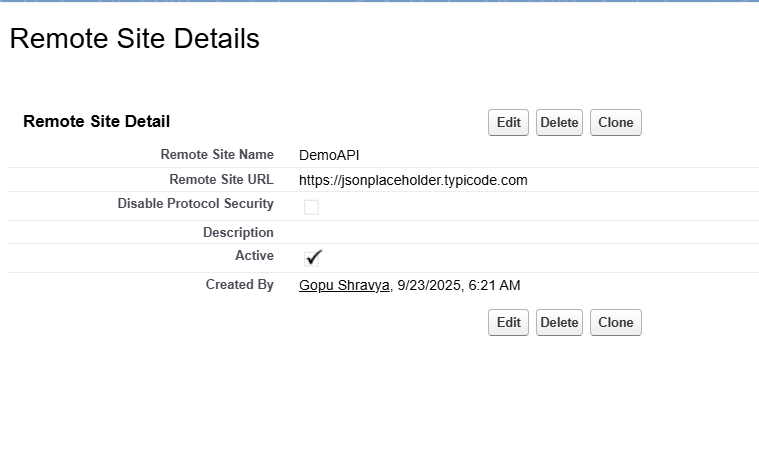


* Home Page Layouts / Utility Bar
* **Project Implementation:**
* Optional for this project. If implemented:
  + Customized home page to show **recent recognitions** or summary of **total reward points**.
  + Utility bar could include **quick links** to create new Recognition or view top performers.

# Phase 7: Integration & External Access

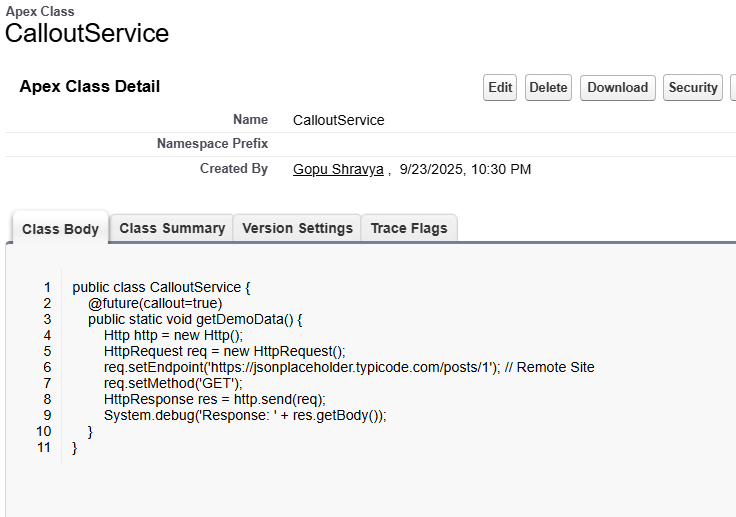
* Remote Site Settings

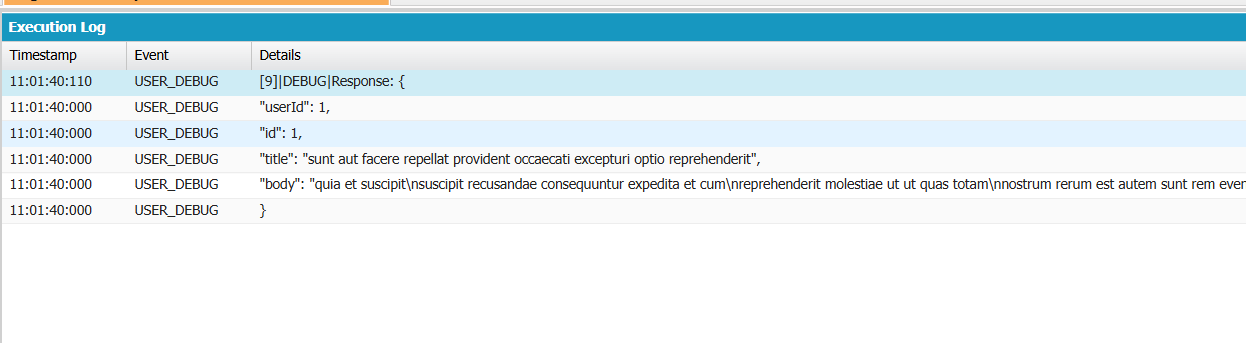
1. A **Remote Site Setting** was created (DemoAPI) to establish trust with the external service https://jsonplaceholder.typicode.com.
2. This allowed Salesforce to securely make HTTP callouts to the API without throwing authorization errors.



* REST API Callout with Apex

1. A custom **Apex class (CalloutService)** was implemented using @future(callout=true).
2. The callout retrieves data from the external service.
3. The response is logged and validated through **Debug Logs**, confirming successful integration.
4. This demonstrates how the Employee Recognition system can connect with external platforms to exchange data when required.

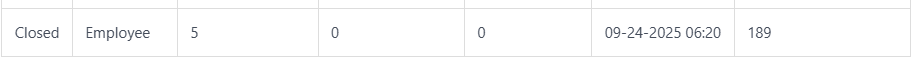




# Phase 8: Data Management & Deployment

* Data Import Wizard

1. Imported **Employee records** using CSV.
2. Ensures Employee data is available for Recognition linking.

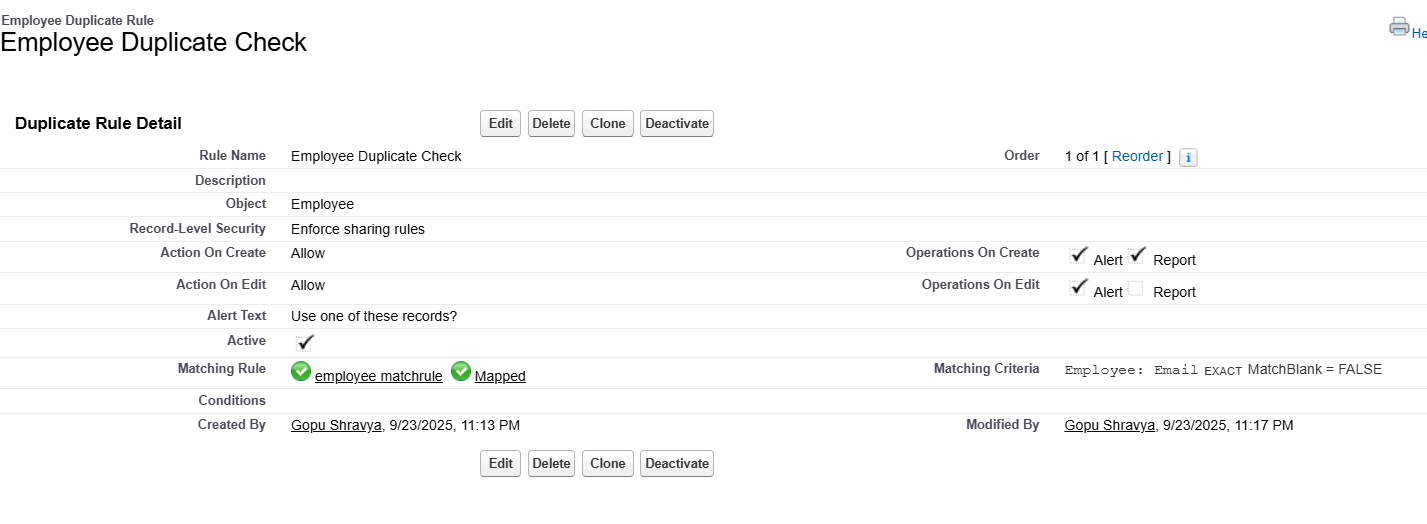


* Data Loader

1. Can be used for **bulk import/export**.
2. Employee import done using Data Loader.

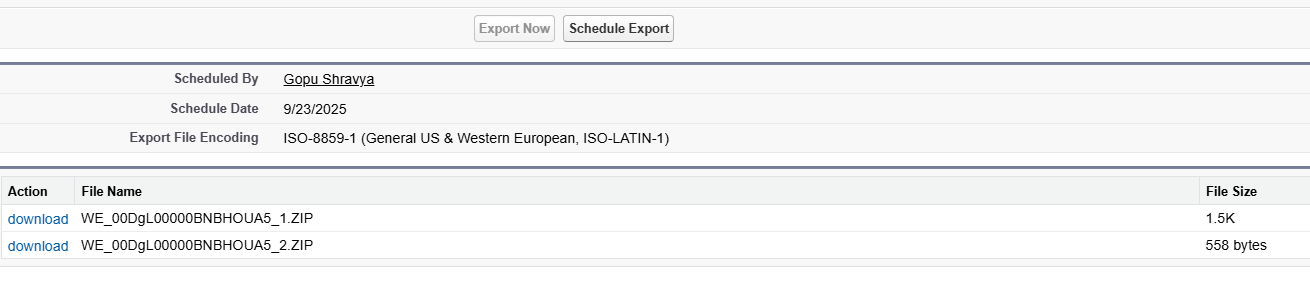
* Duplicate Rules

1. Prevents creating duplicate Employee records.
2. Rules applied on **Employee Email**



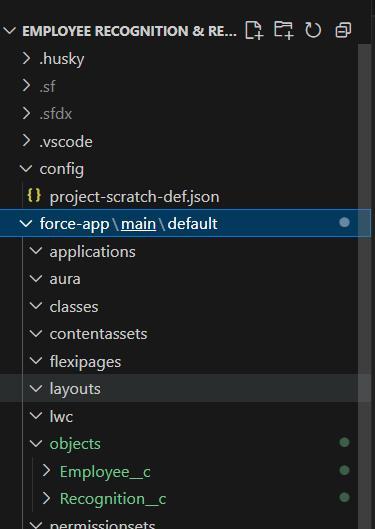
* Data Export & Backup

1. Exported Employee and Recognition records as backup.
2. Can schedule periodic exports for safety



* VS Code & SFDX

1. Used to retrieve and deploy metadata.
2. Helpful for version control and tracking changes

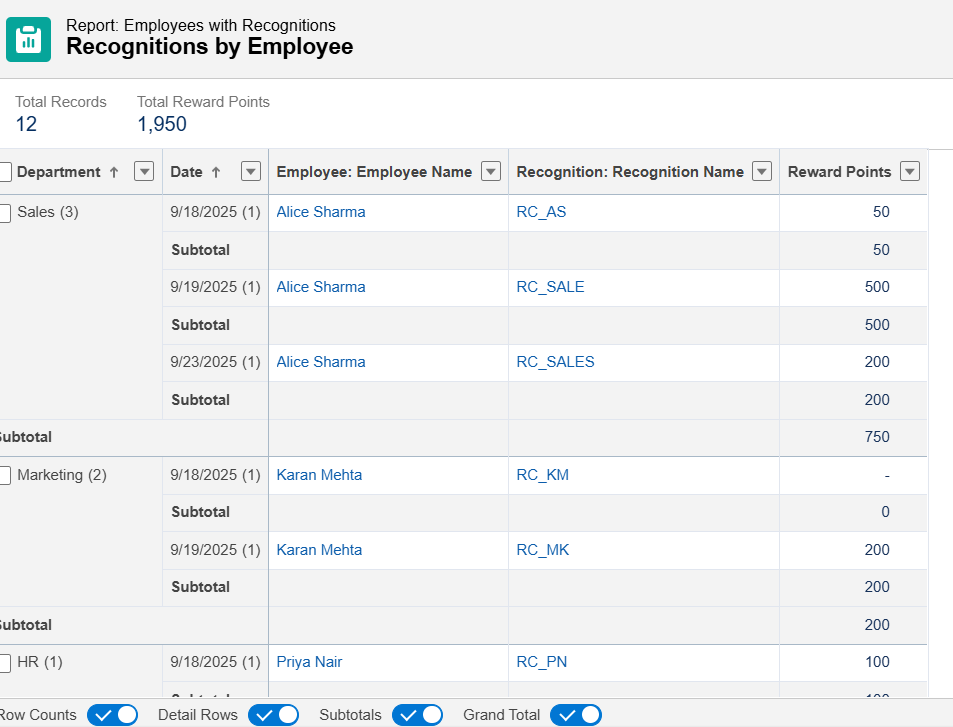


# Phase 9: Reporting, Dashboards & Security Review

* Reports (Summary, Matrix)

1. **Summary Report:** Top 5 Employees by reward points (grouped by points).
2. **Matrix Report:** Recognitions by Department / Month.





* Report Types

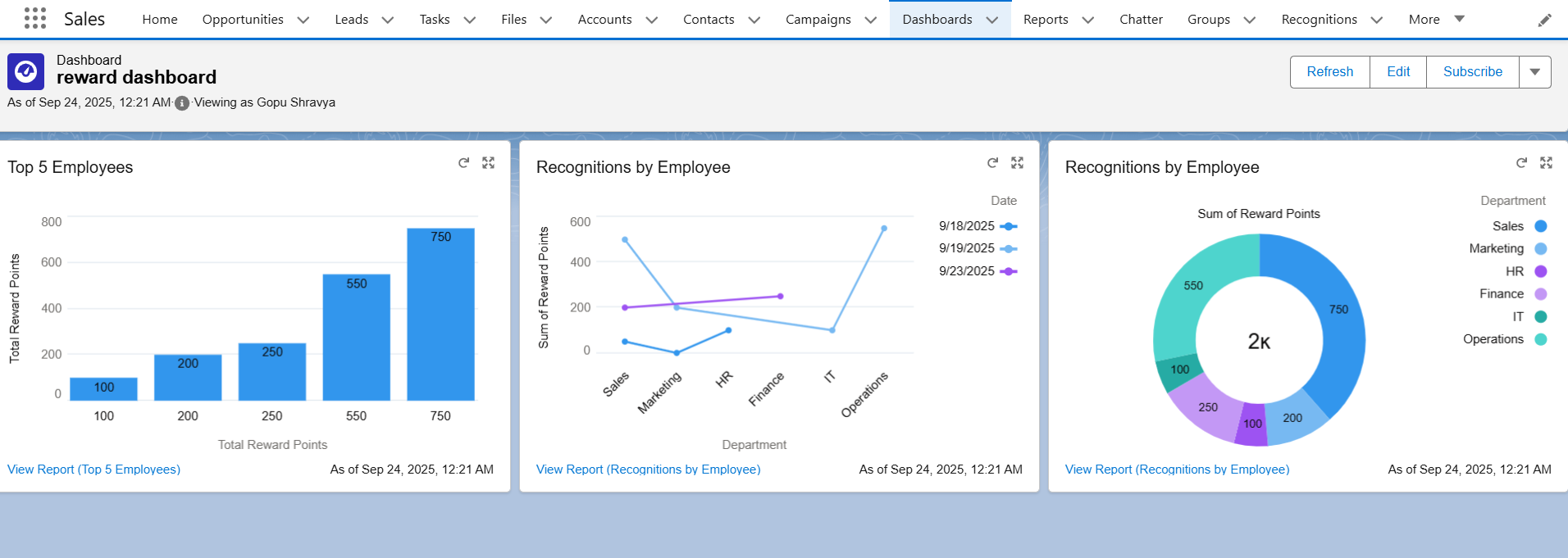
1. **Purpose:** Define object relationships for reporting.
2. **Project notes:** Custom report type **Employee with Recognitions** created.

* Dashboards

1. **Purpose:** Visual representation of reports for easy analysis.
2. **Project notes:**

Components:

* 1. Top Performers chart (bar chart from Top 5 Employees report).
  2. Monthly Trends chart (line/column chart from Recognitions by Department/Month report).



* Sharing Settings

1. **Purpose:** Control record access.
2. **Project notes:**

* Org-wide defaults: Employee and Recognition records private.
* Sharing rules added for managers to view team data.