### **CRM Application for Jewel Management**

**College name** : KPR college of Arts Science and Research

**College code** : bruaz

**Team ID** : NM2025TMID21450

**Team Leader Name**: Anaksha V

Email :23bit004@kprcas.ac.in

**Team members:**

Team member1: Pandilakshmi V

Email : 23bit039@kprcas.ac.in

Team member2: Madhumitha S

Email : 23bit030@kprcas.ac.in

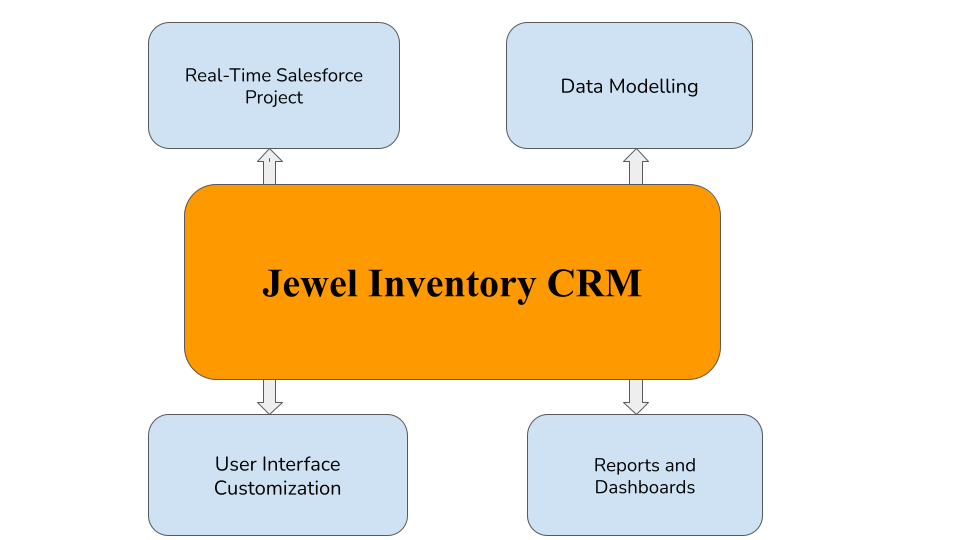
Team member3: Nalinapriya V

Email : 23bit033@kprcas.ac.in

**1.INTRODUCTION**

**1.1 Project Overview**

The Jewel Inventory System is a comprehensive software Solution designed to streamline and manage the inventory and sales processes of a jewellery store or a jewellery manufacturer. The system aims to provide an efficient and user-friendly solution to track and control the inventory of various jewellery items, maintain accurate records, and facilitate seamless sales transactions.



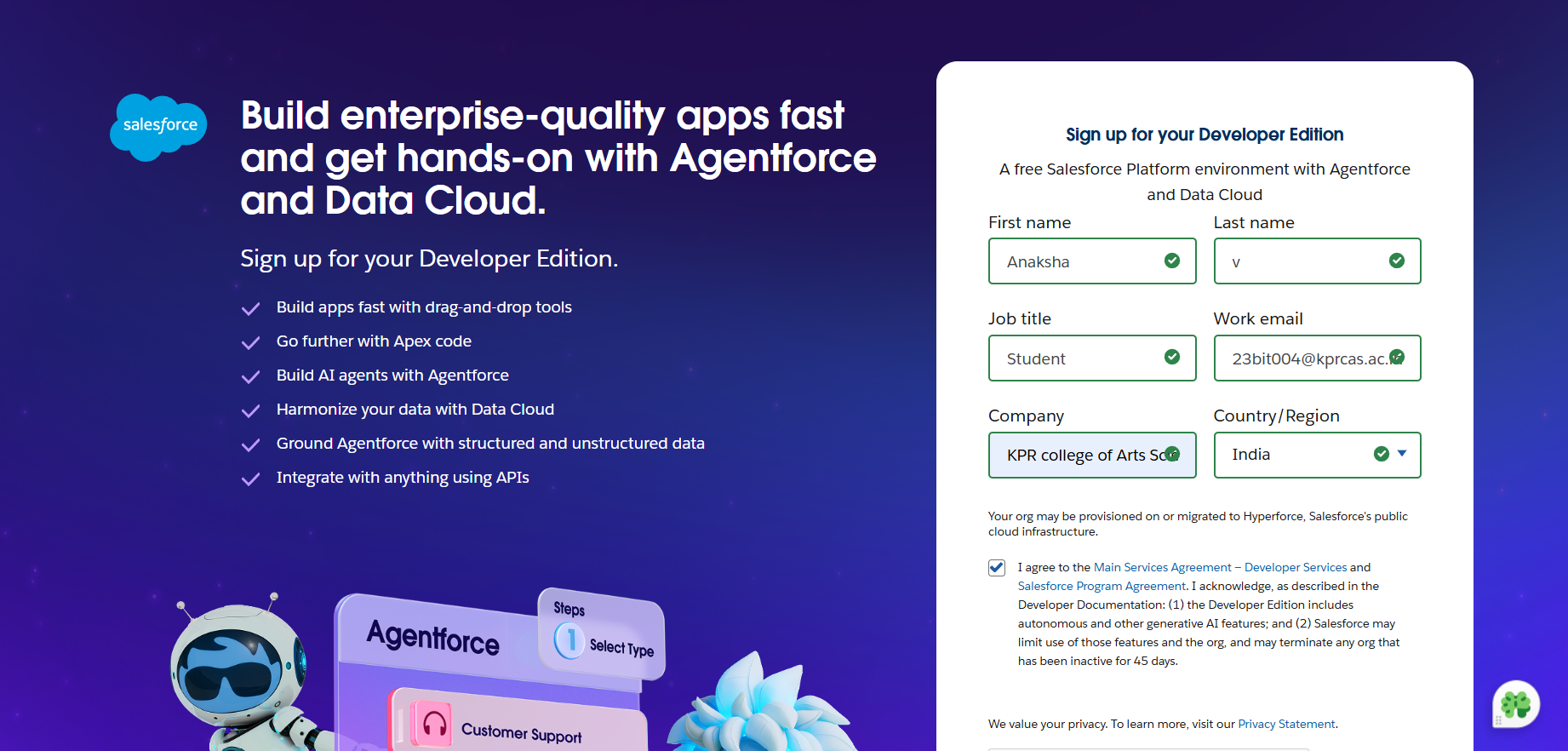
**1.2 Purpose**

The purpose of the Jewel Inventory System is to simplify and streamline the management of jewellery inventory and sales. It helps businesses track stock accurately, reduce manual errors, and ensure smooth transactions. By providing an organized and user-friendly platform, the system enhances efficiency, improves customer service, and supports better decision-making.

**DEVELOPMENT PHASE**

●**Creating Developer Account:**

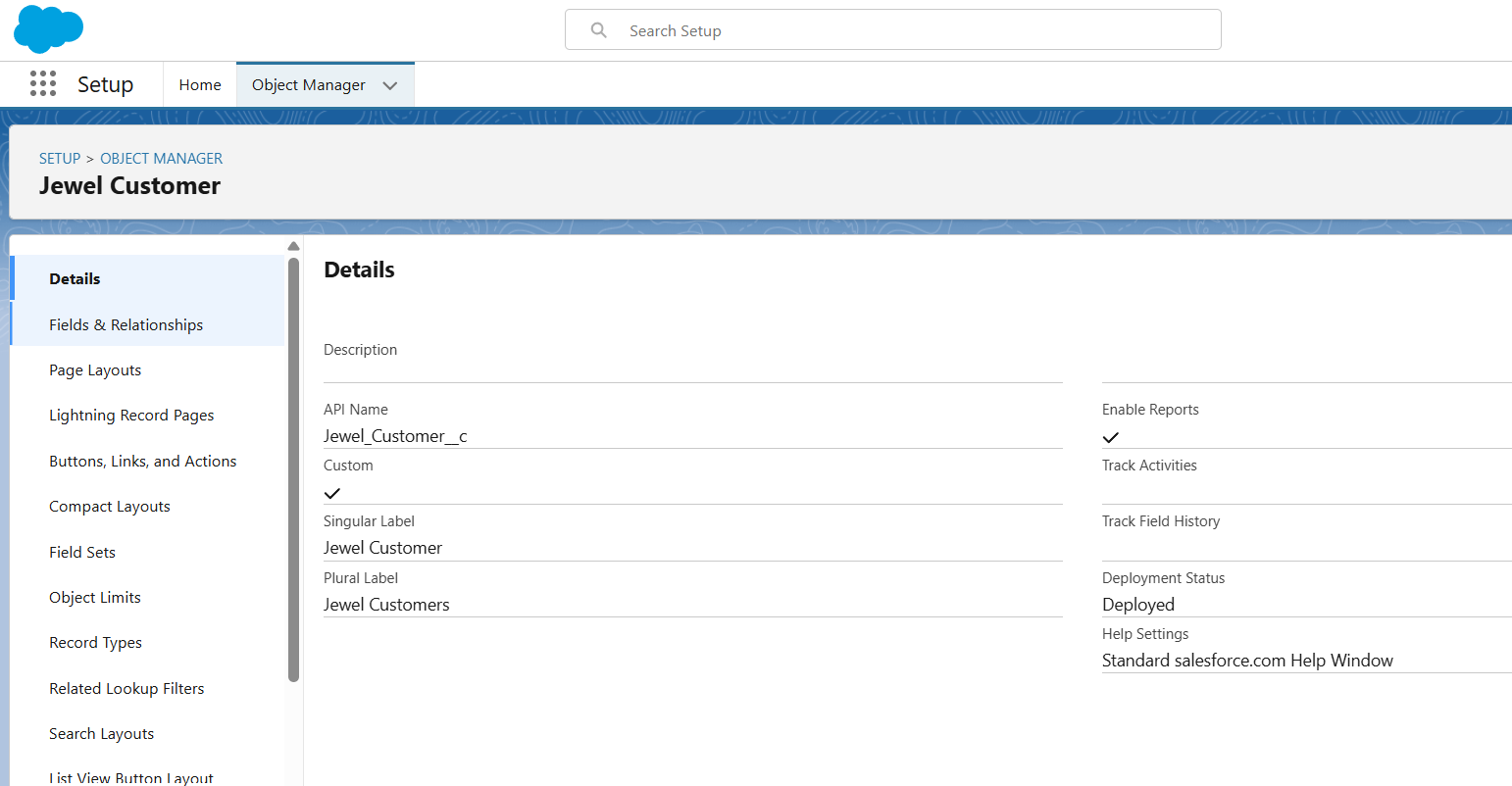
By using this URL **-** [**https://www.salesforce.com/form/developer-signup/?d=pb**](https://www.salesforce.com/form/developer-signup/?d=pb)

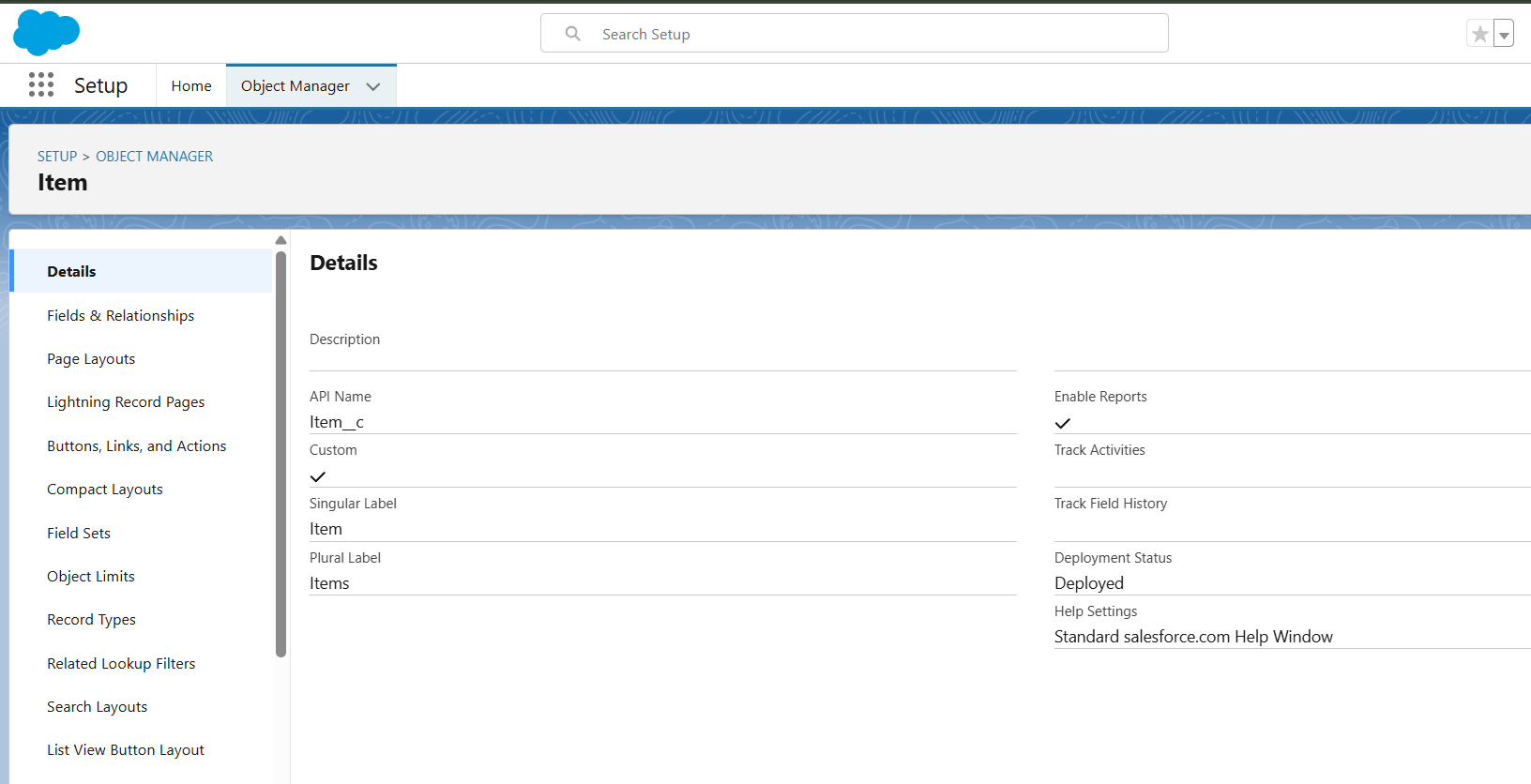
****

● **Creating Custom Objects:**

Objects serve as the foundation for organizing and leveraging critical information within Salesforce.

**Created objects:** Jewel customer object, Item object.

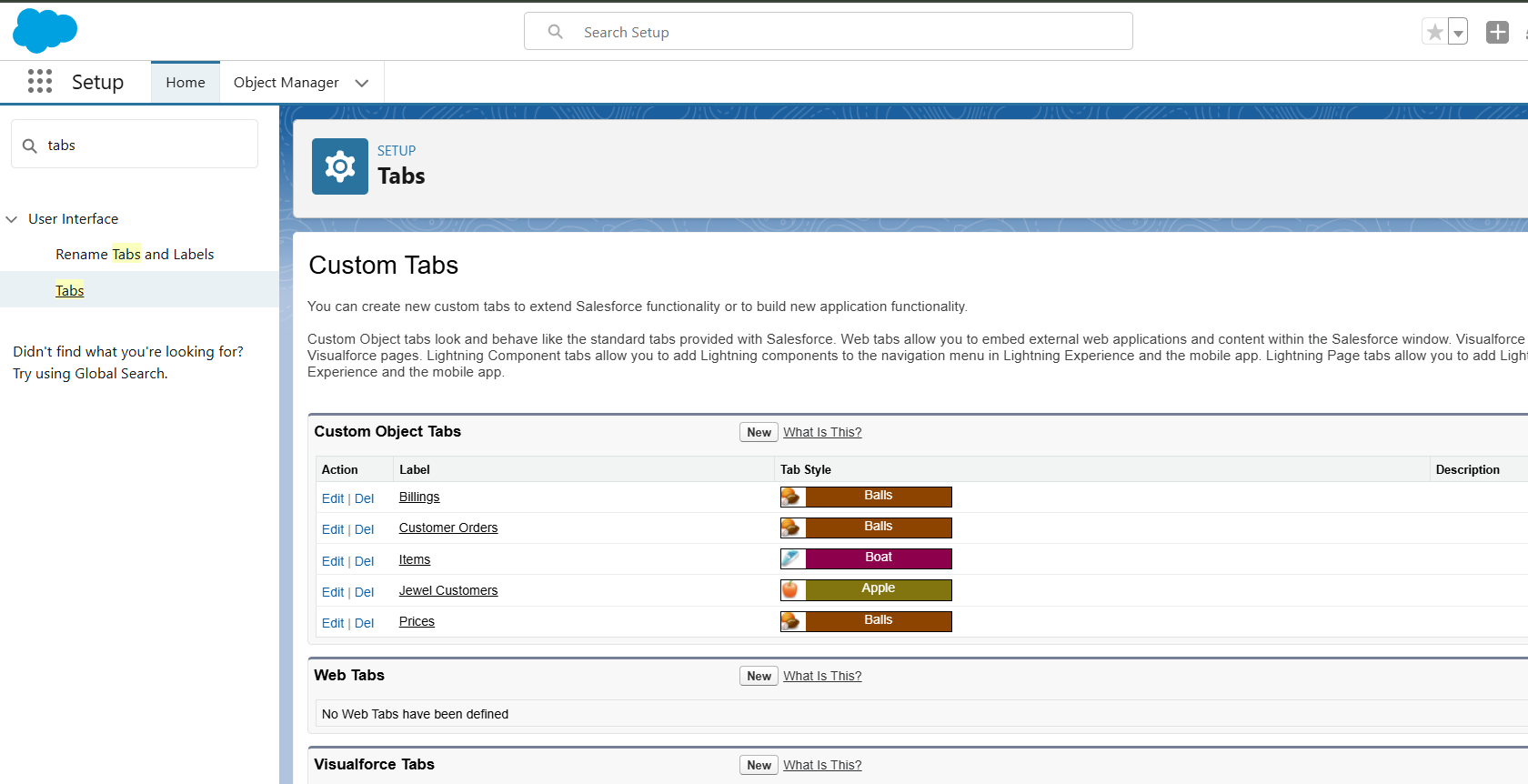




● **Creating Tabs:**

A tab is like a user interface that is used to build records for objects and to view the records in the objects.

1. Creating a Custom Tab
2. Creating a Tab(Item)



● **Creating the Lightning App :**

The Jewel Inventory System provides a unified platform to manage jewellery inventory, sales, and records efficiently.

**1. Setup Path:** App Manager → New Lightning App.

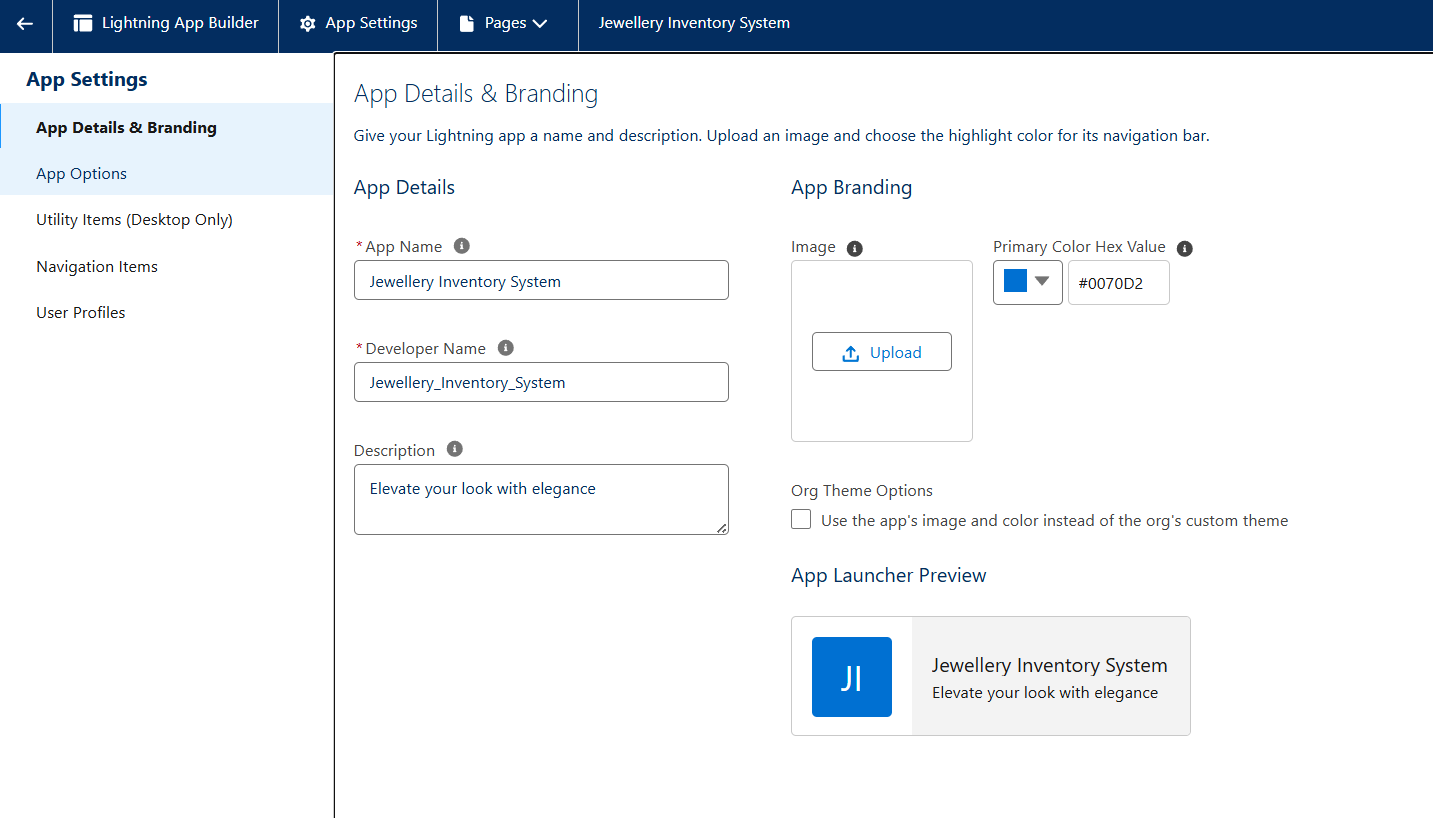
**2. App Details:**

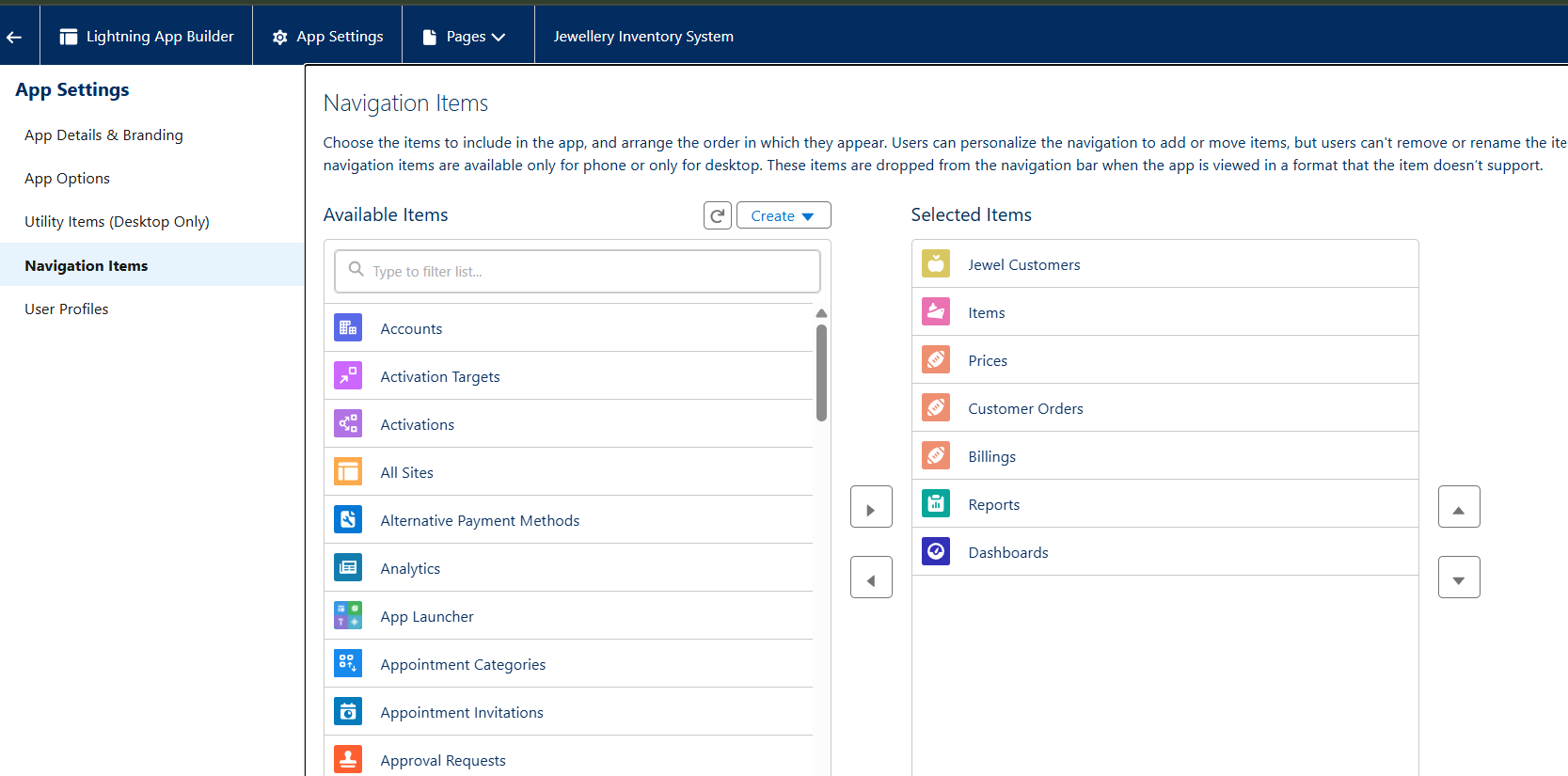
Name:

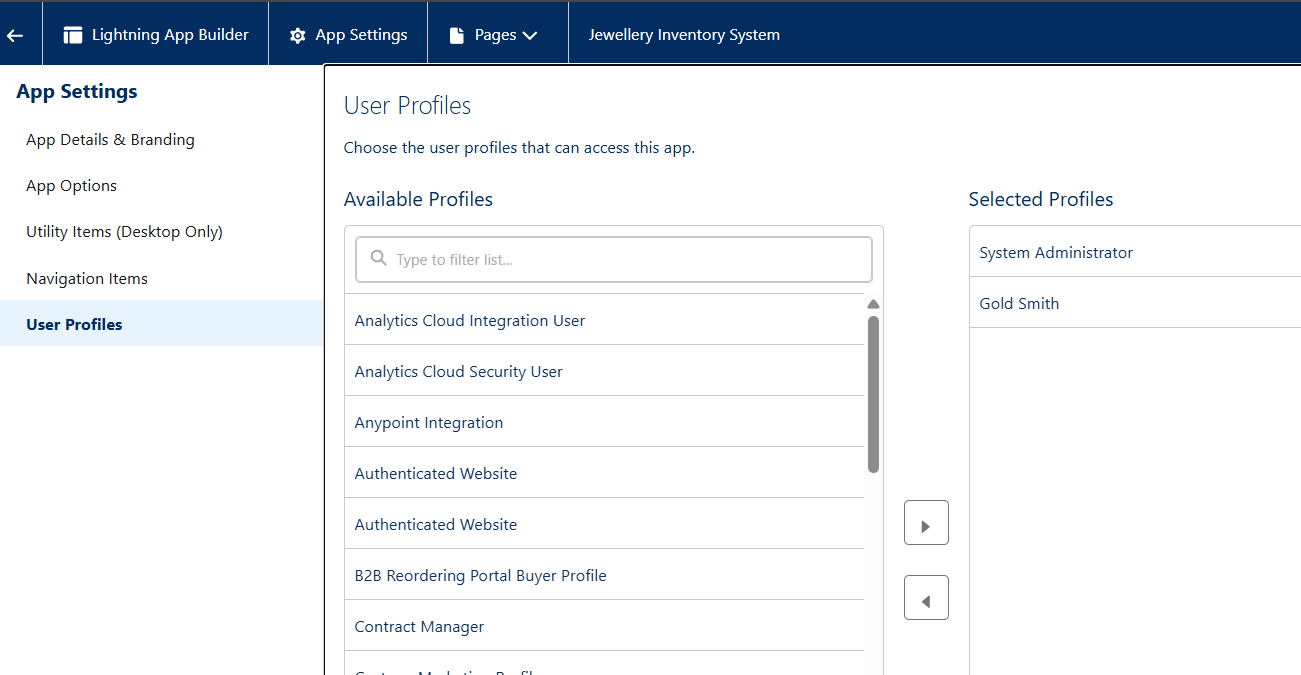
Developer Name: Auto-generated.

**3. Navigation Items Added:** Jewel Customer, Items, Prices, Customer Order, Billings, Reports and Dashboards.

**4. Profiles:** Assigned to System Administrator, Gold Smith



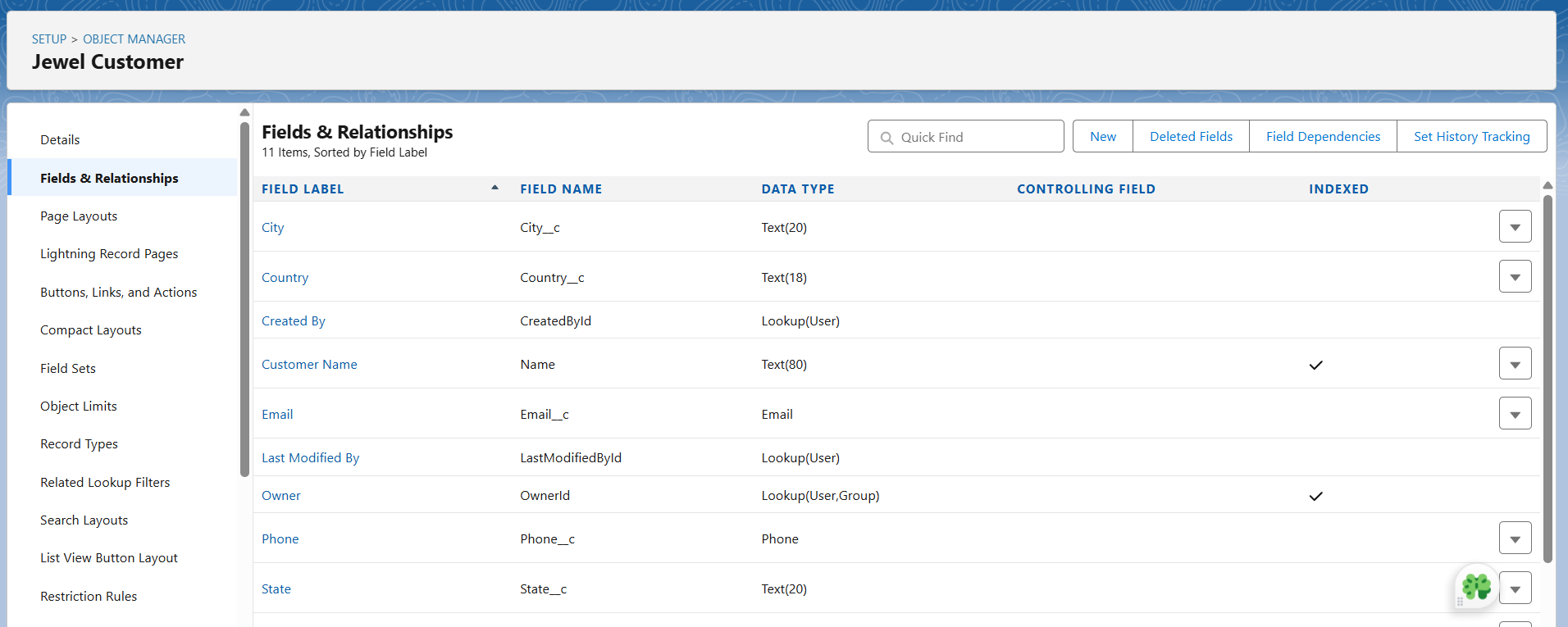
****

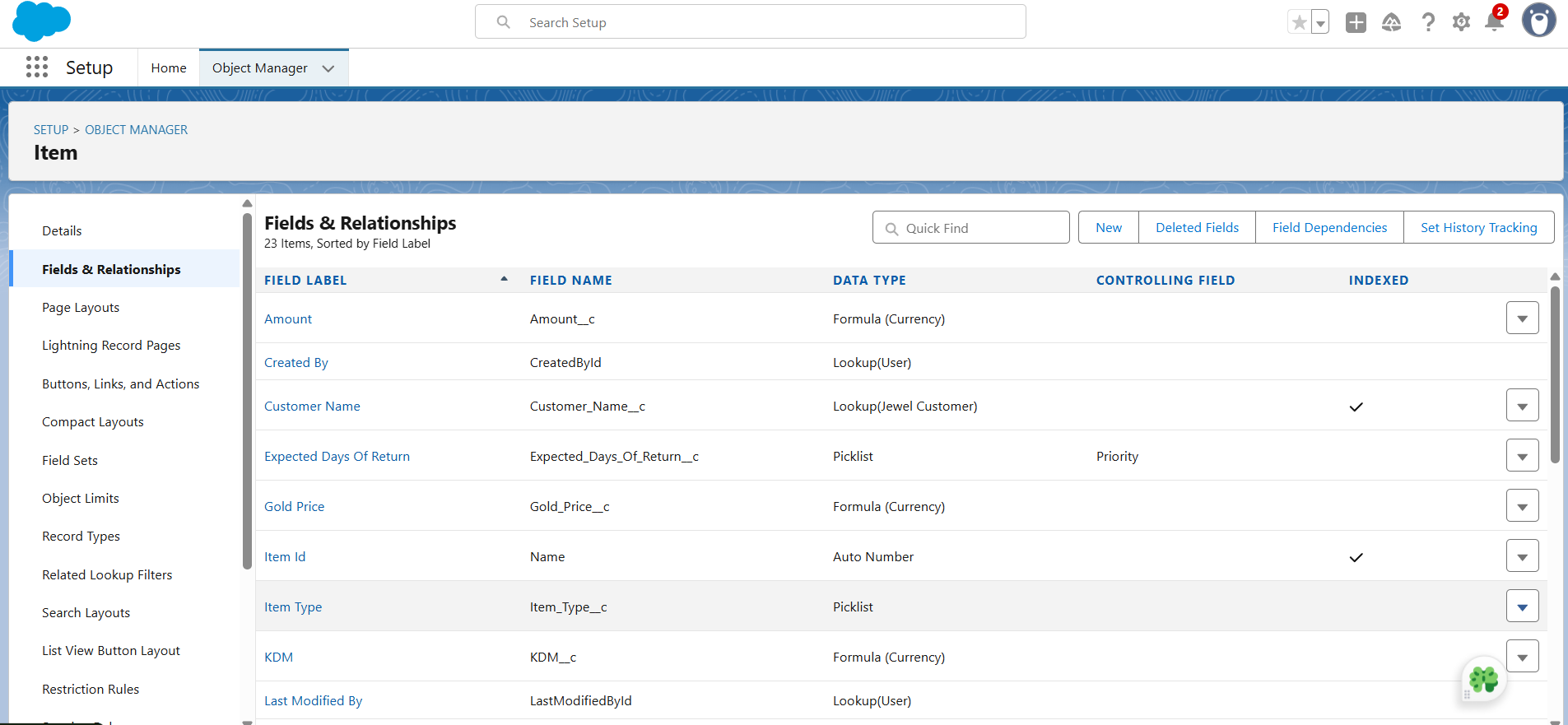
****

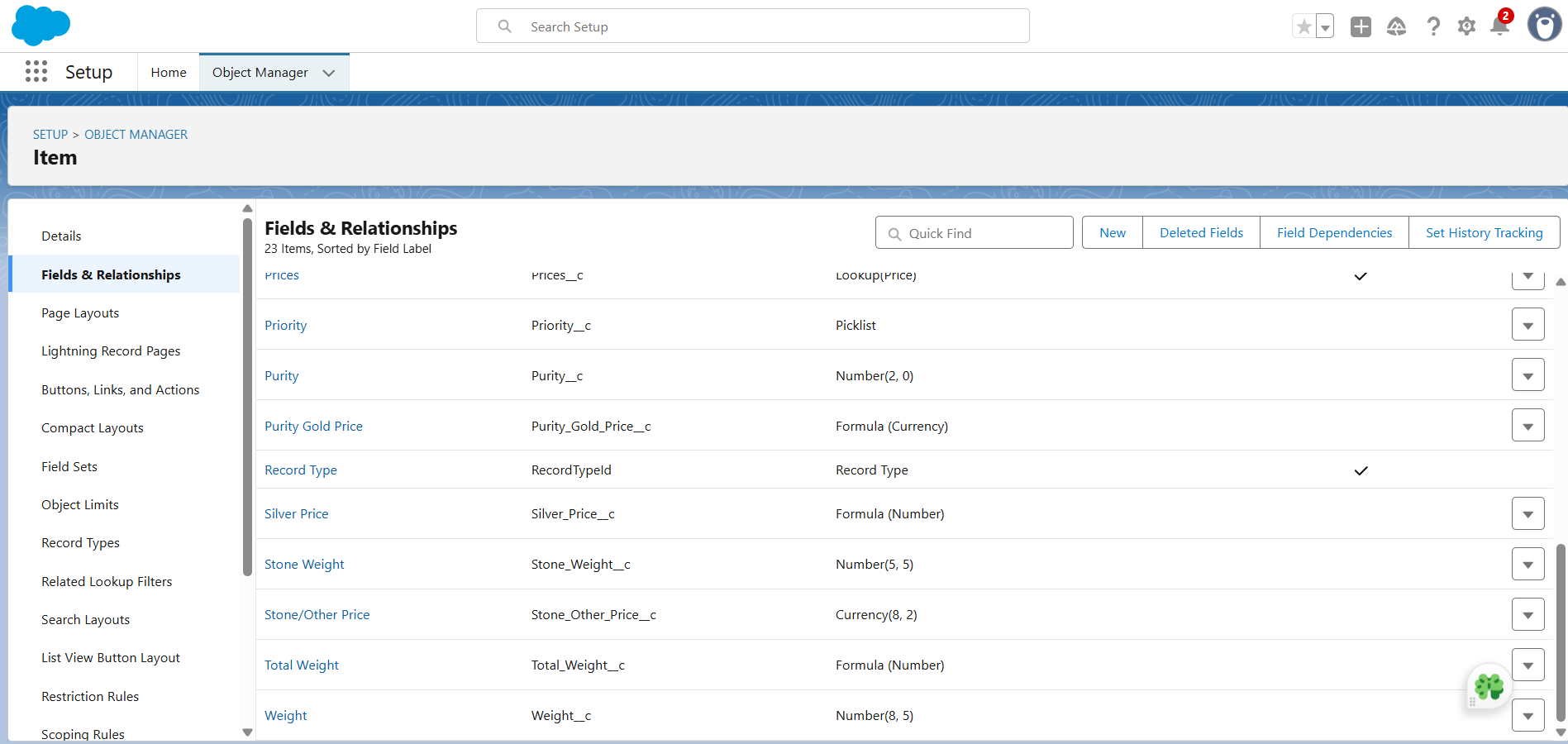
● **Creating Fields:**

Created fields:

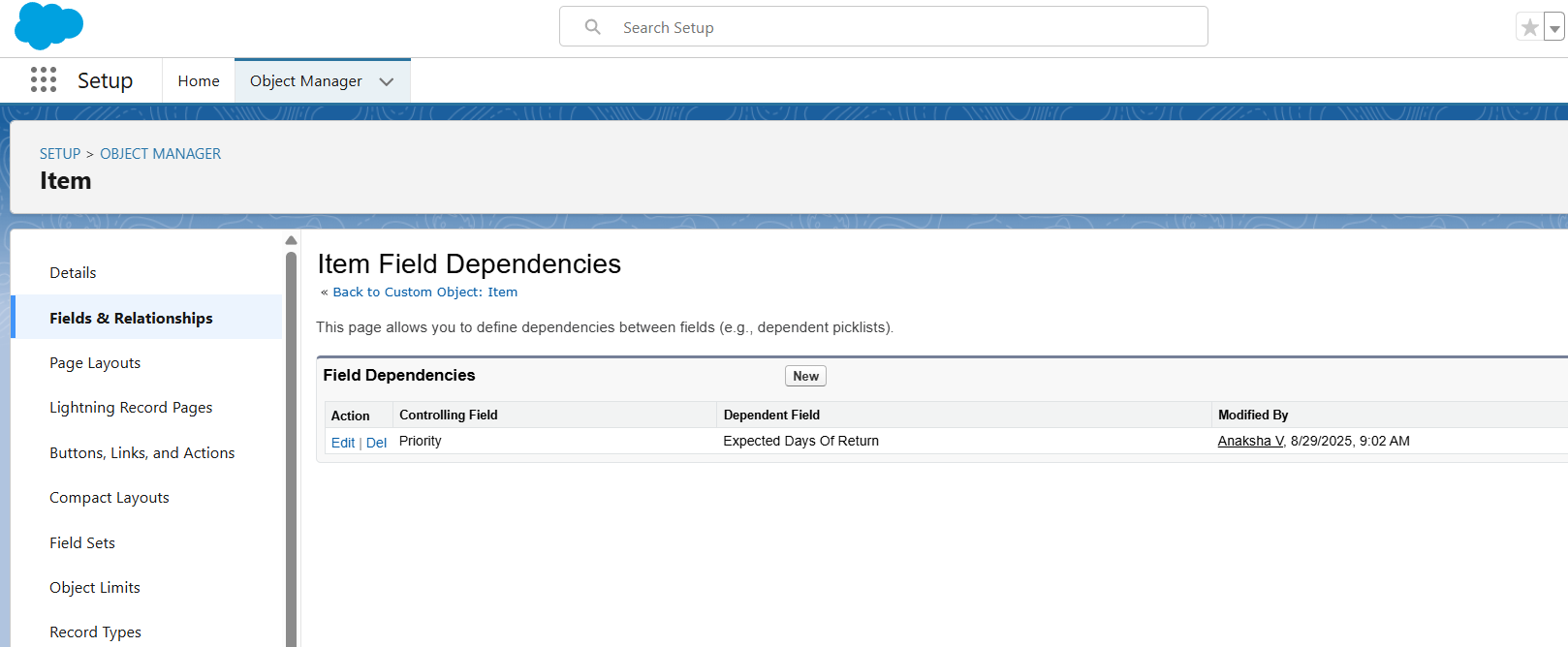
1. Customer Name, City, Country, Email, Phone, State, Street, Zip/Postal code(Jewel Customer)
2. Amount, Custom and Name, Expected Days of Return, Gold Price, Item ID, Item Type, KDM, Making Charges, Ornament, Percentage, Prices, Priority, Purity, Purity Gold Price, Silver Price, Stone Weight, Stone/ Other Price, Total Weight, Weight(Item)



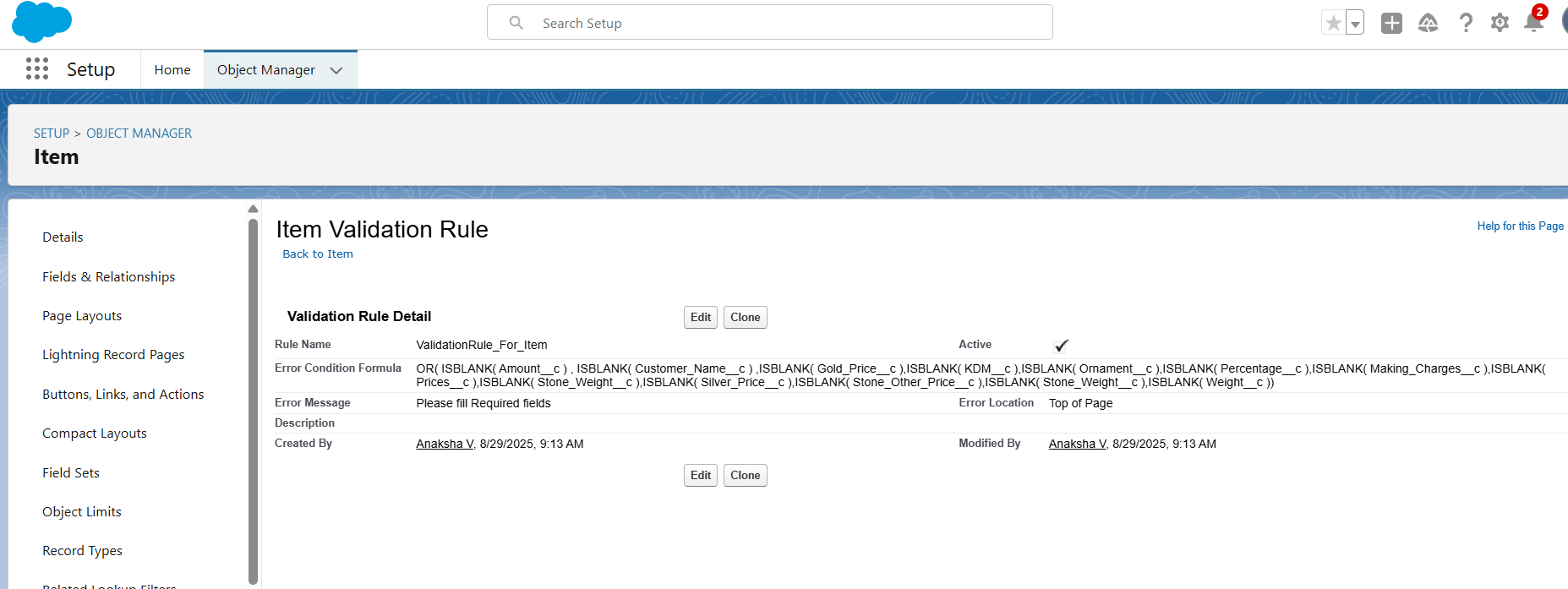




**Creating Field Dependencies:**

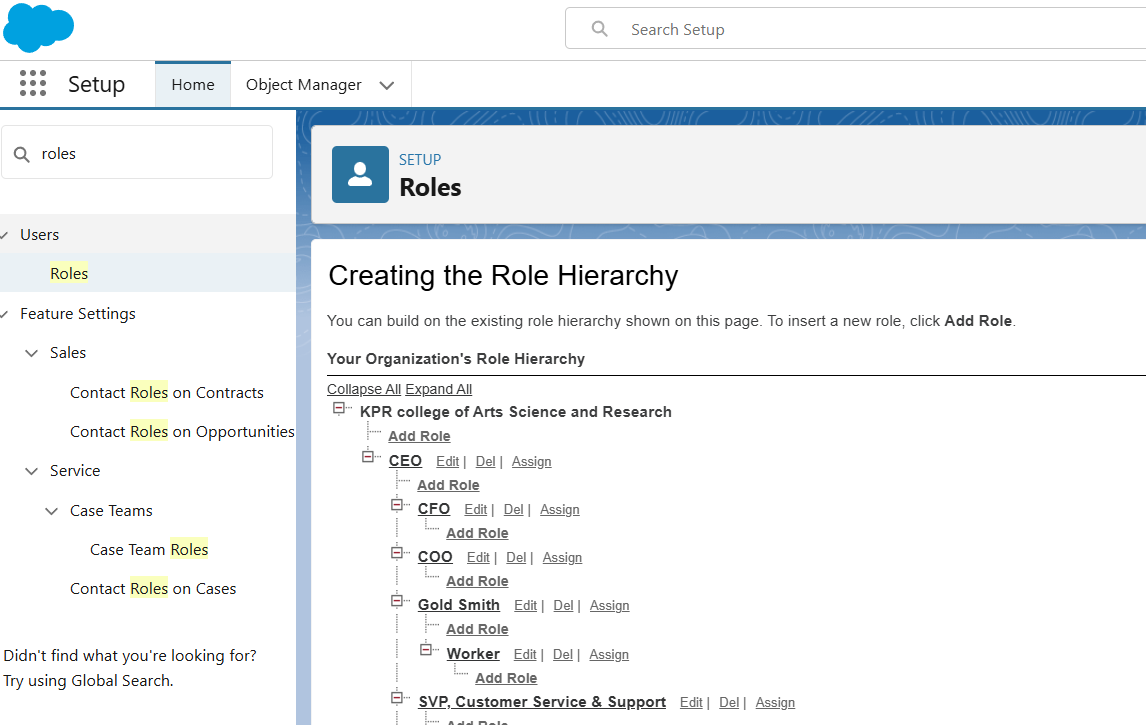


**Creating validation rule:**



● **Creating Roles:**

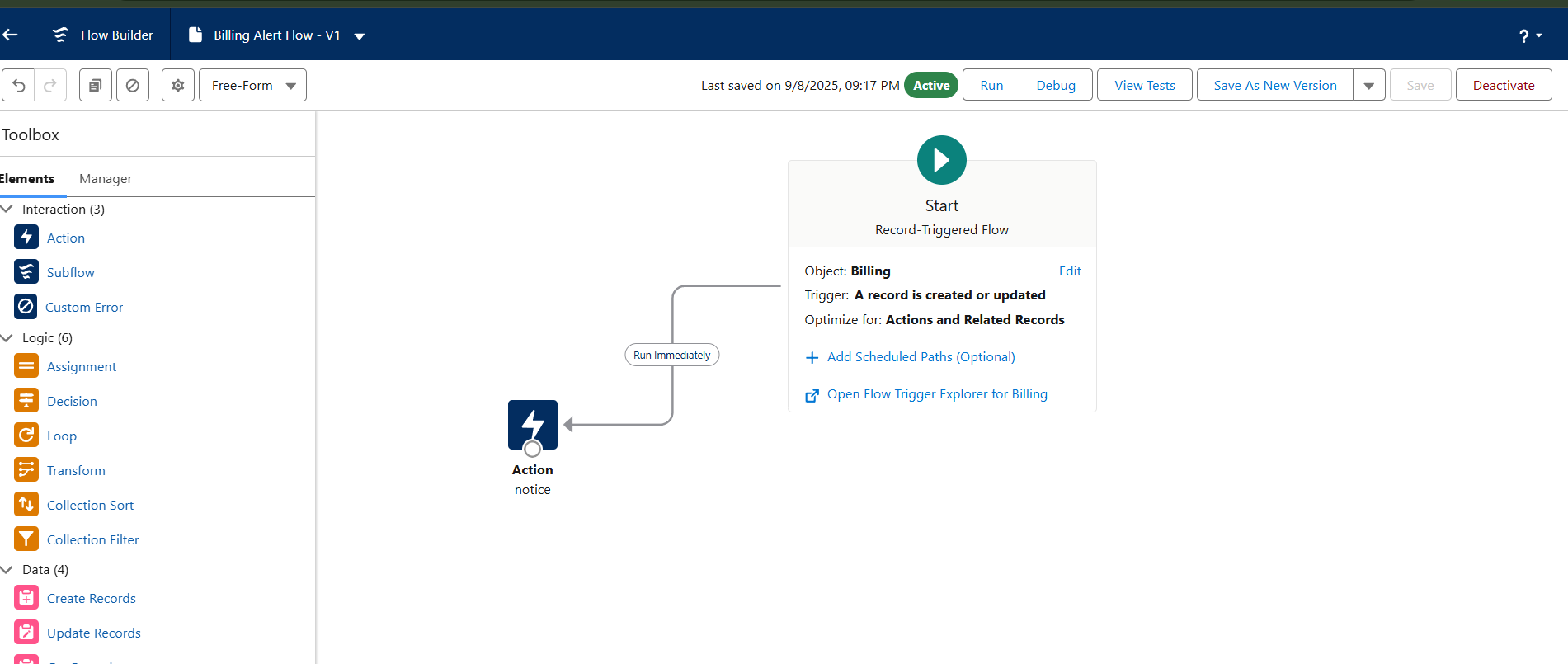
In the Jewel Inventory System, a role defines what data and features a user can access, ensuring proper security and control.



● **Flows:**

Flows automate record creation and updates in the Jewel Inventory System. A Record-Triggered Flow was created to send customer notifications:

* Fields captured: Item Type, Ornament, Weight, Amount, Customer Name, Customer Email.
* Flow action: Automatically sends an email with purchase details whenever a billing record is created or updated.



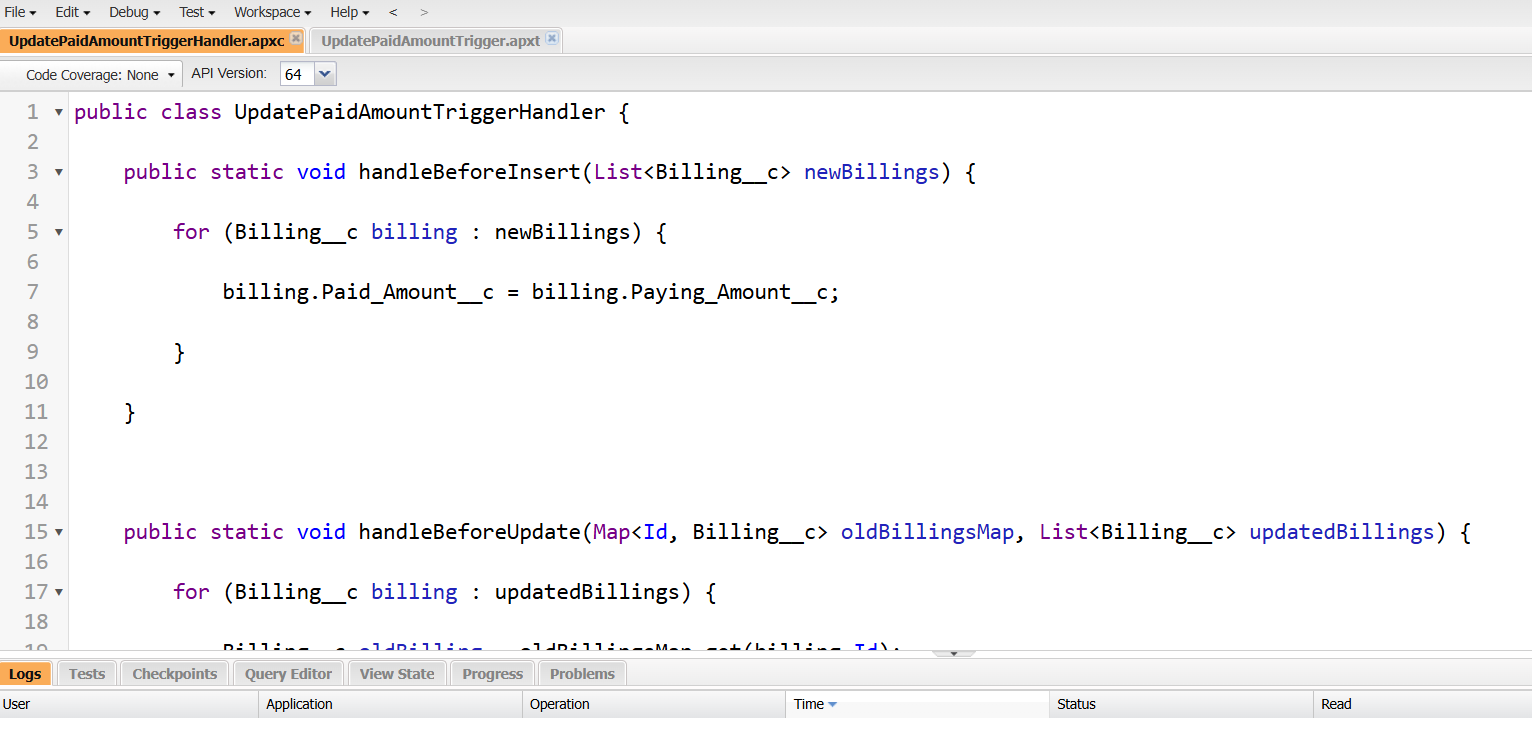
● **Creating Triggers:**

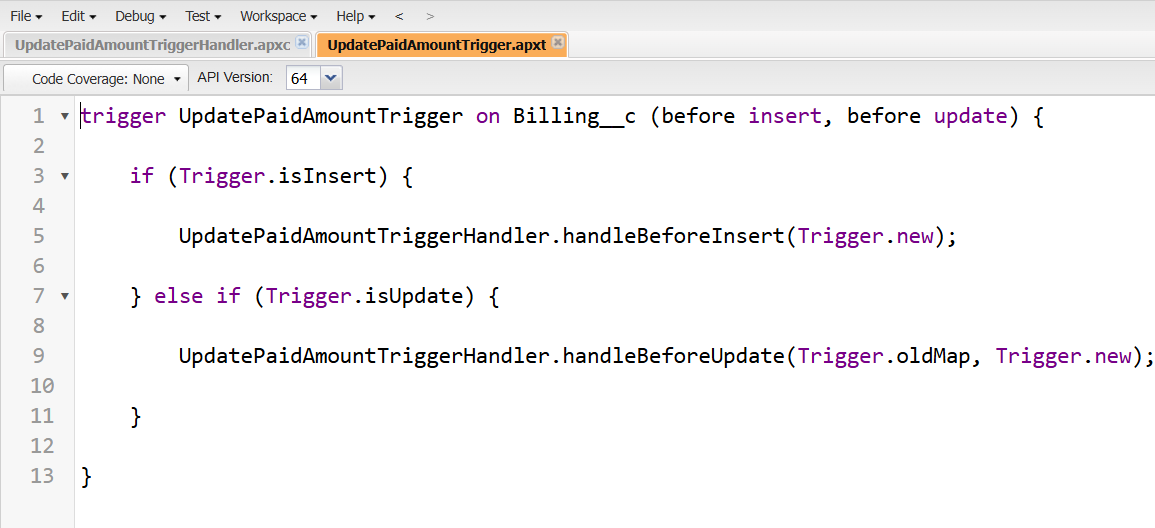
A trigger handler was created:

Handler Name: UpdatePaidAmountTriggerHandler

Object: Billing

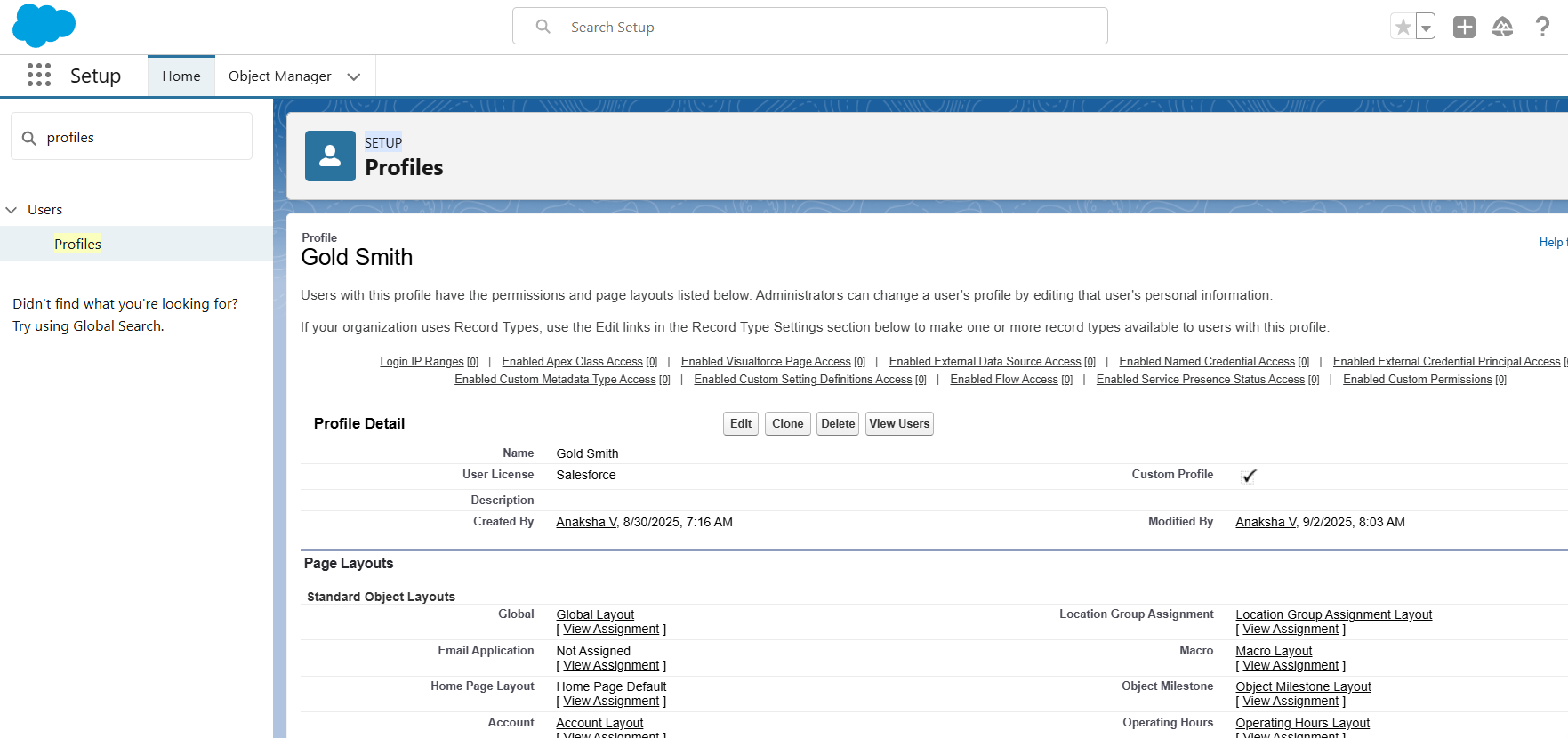
Purpose: Update the Paid Amount field based on the Paying Amount during insert and update operations.



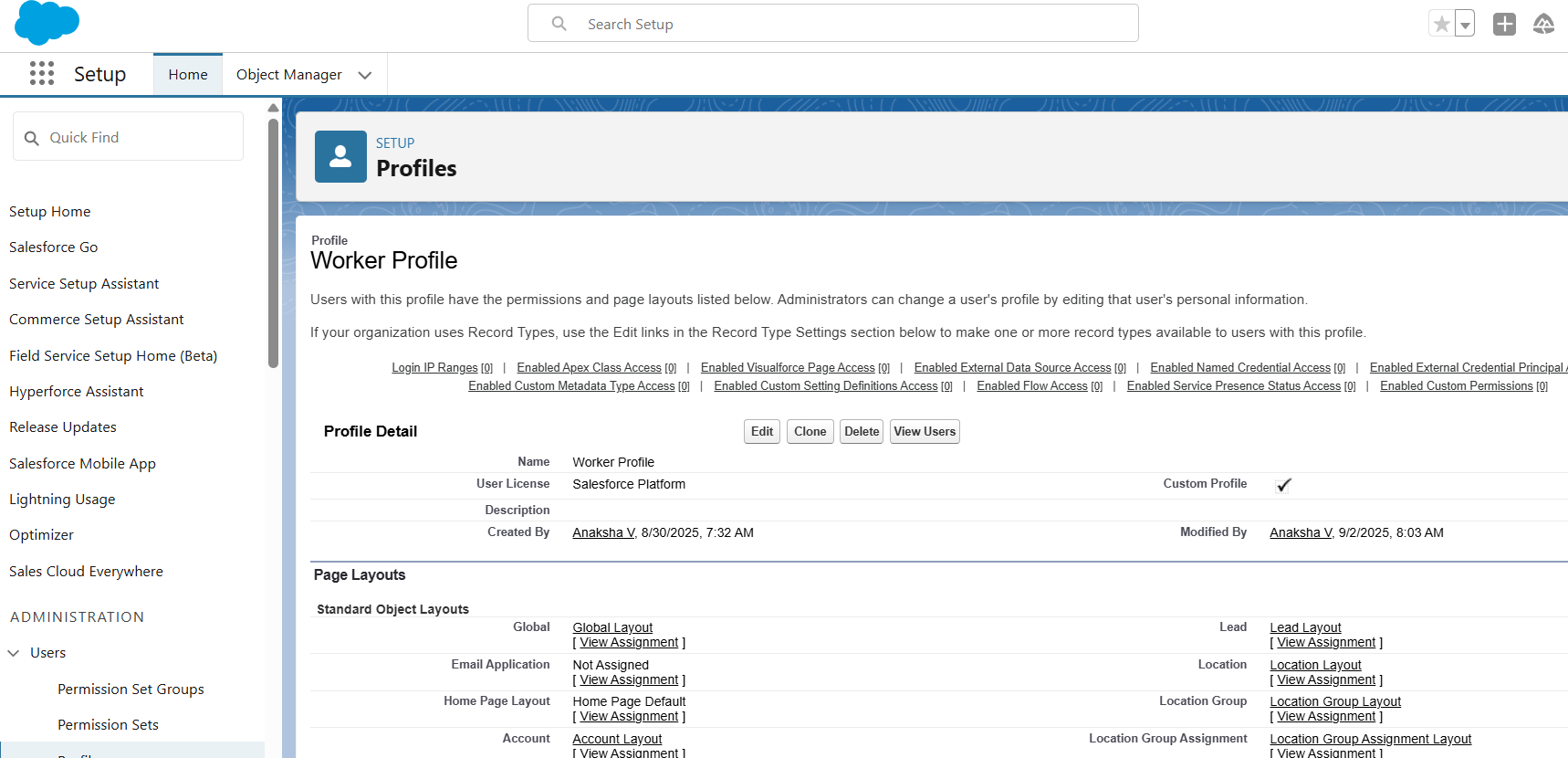


● **Profiles and Users:**

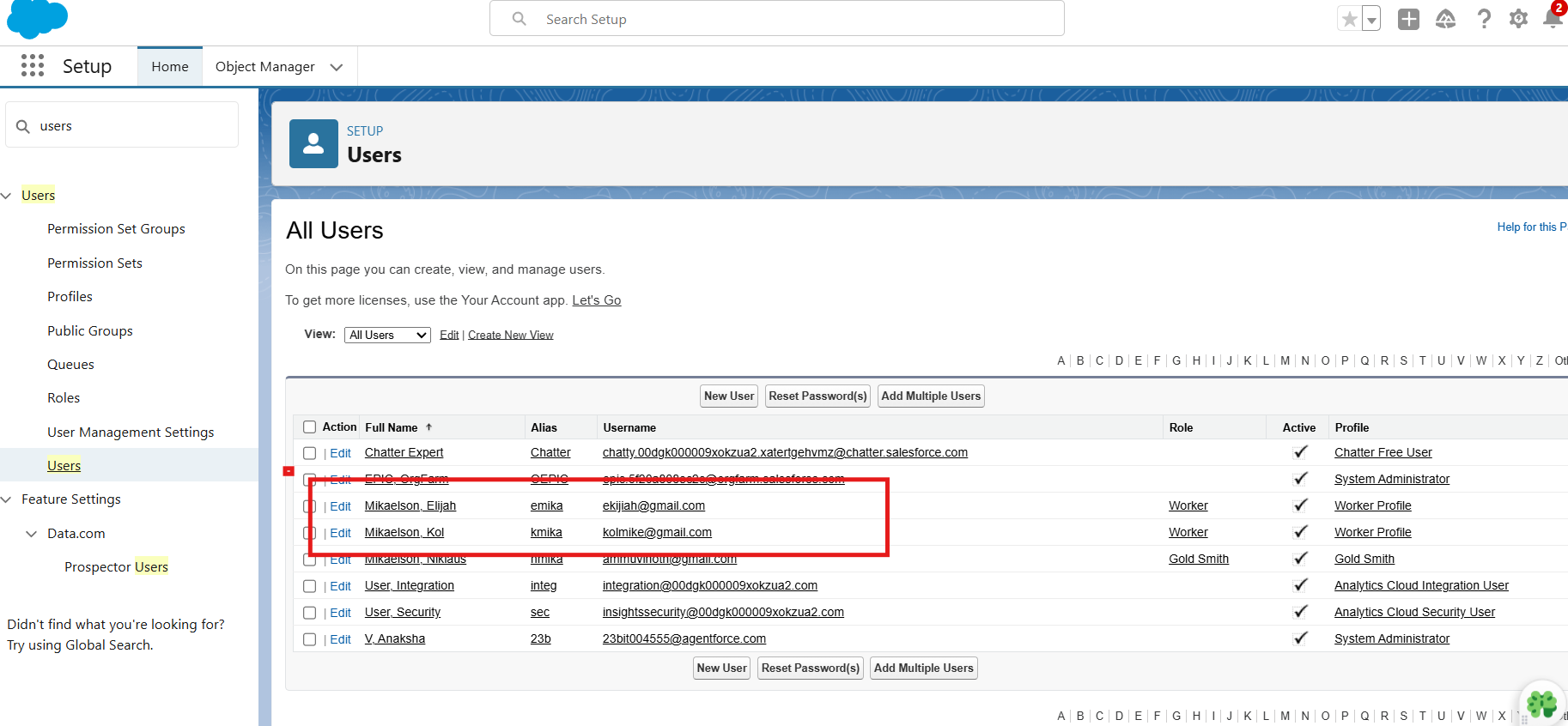
Gold Smith Profile:



Worker Profile:



**Users:** Created two users

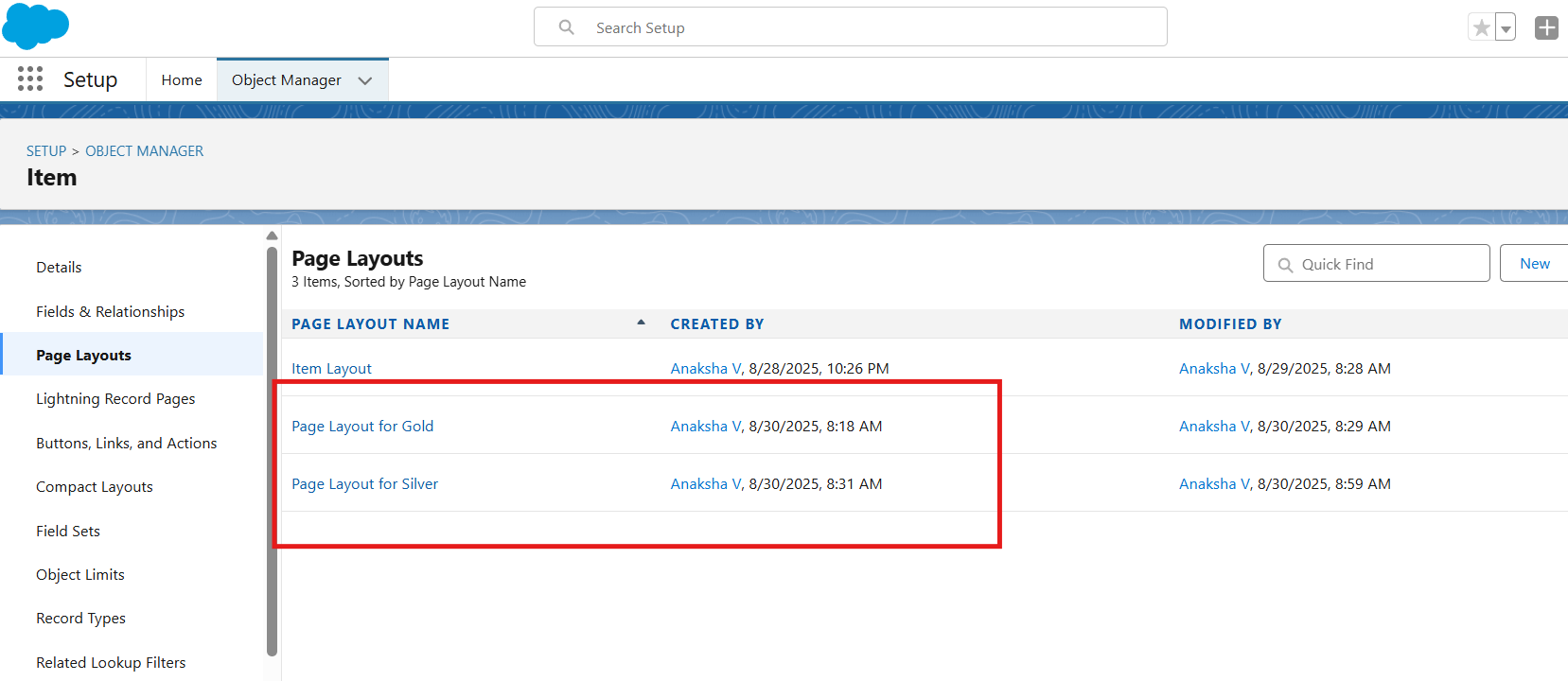


● **Creating Page Layouts:**

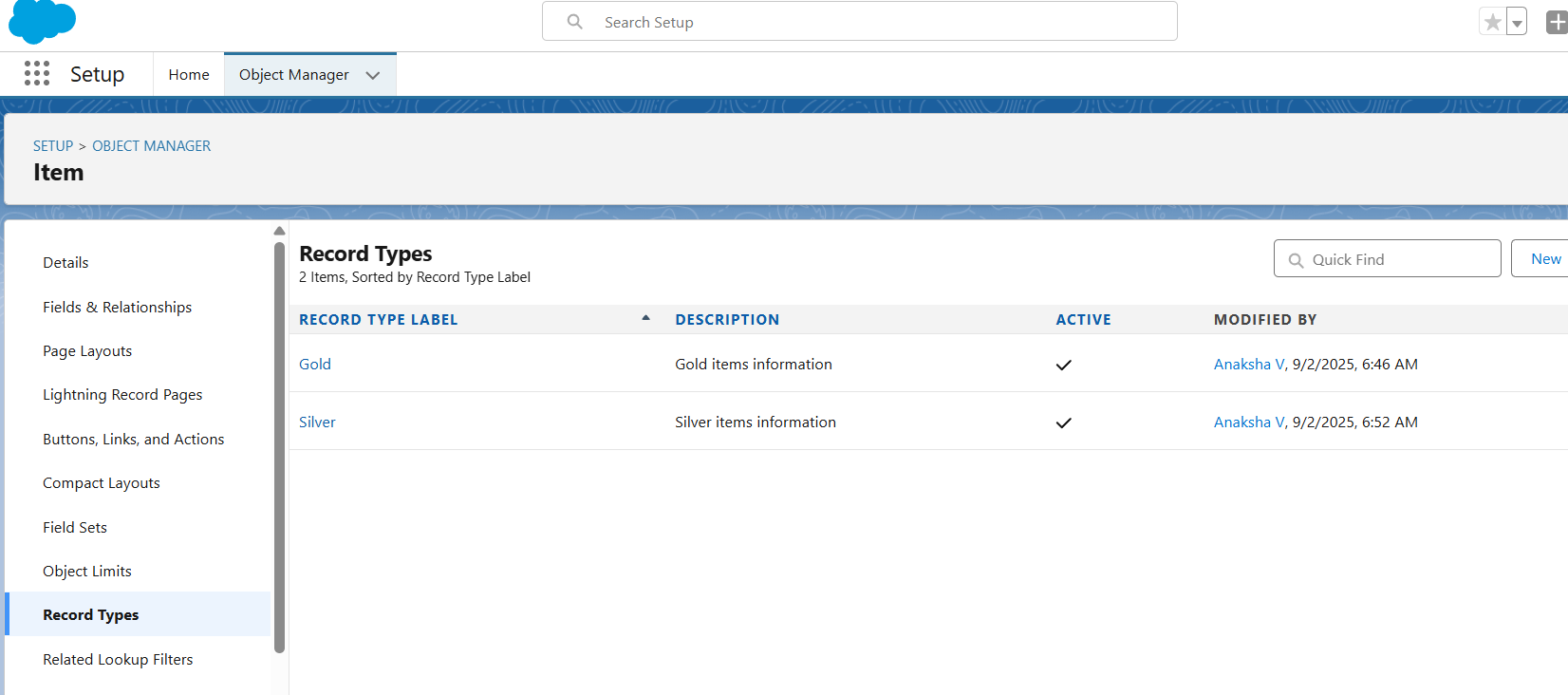
Page layouts can be used to control the appearance of fields, related lists, and custom links on standard and custom objects' detail and edit pages.

1. Gold Page Layout

2. Silver Page Layout



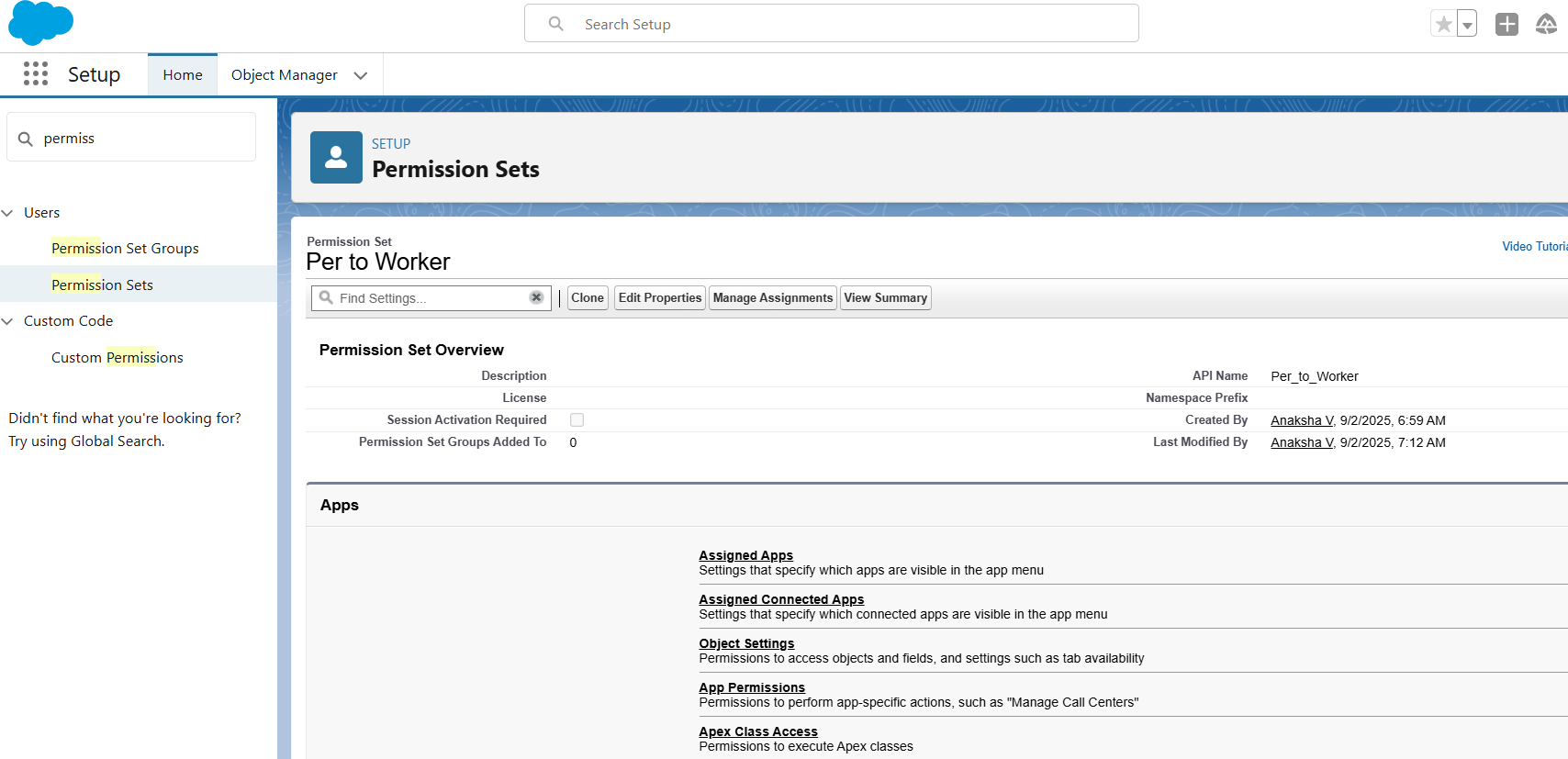
● **Creating Record Types:**

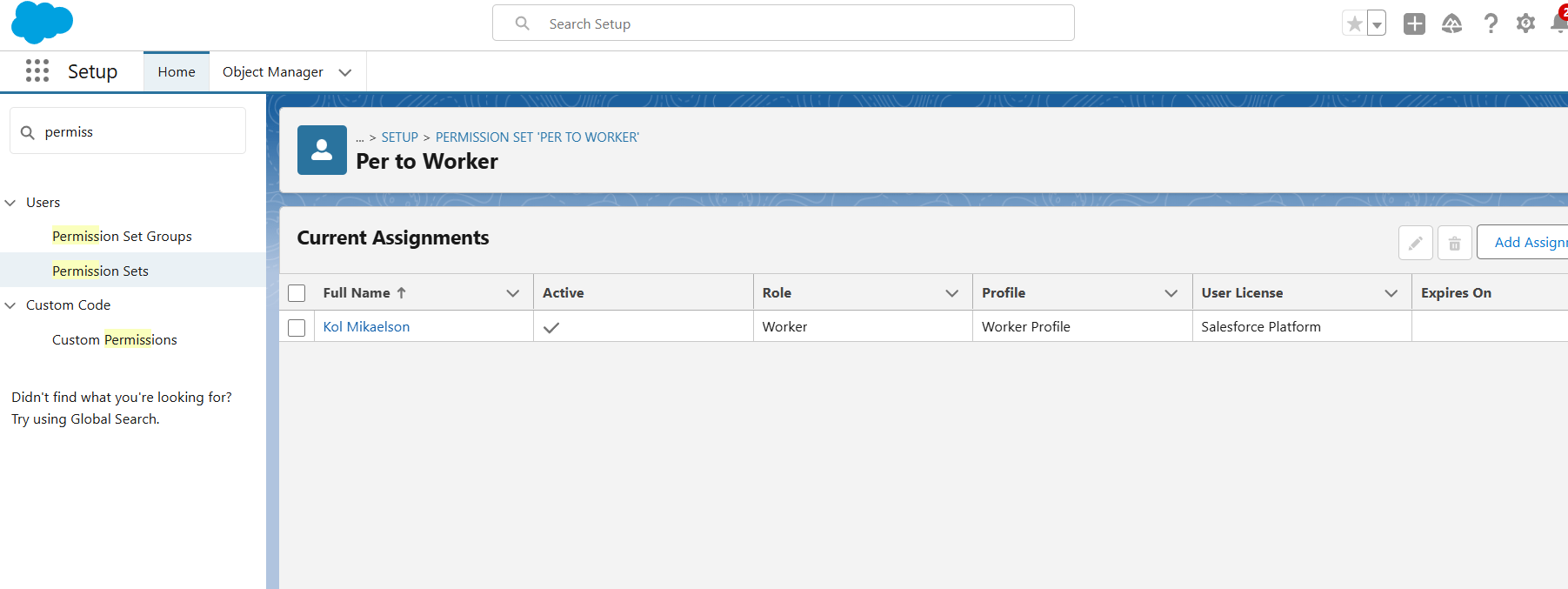
****

● **Creating Permission Sets:**

A permission set was created:

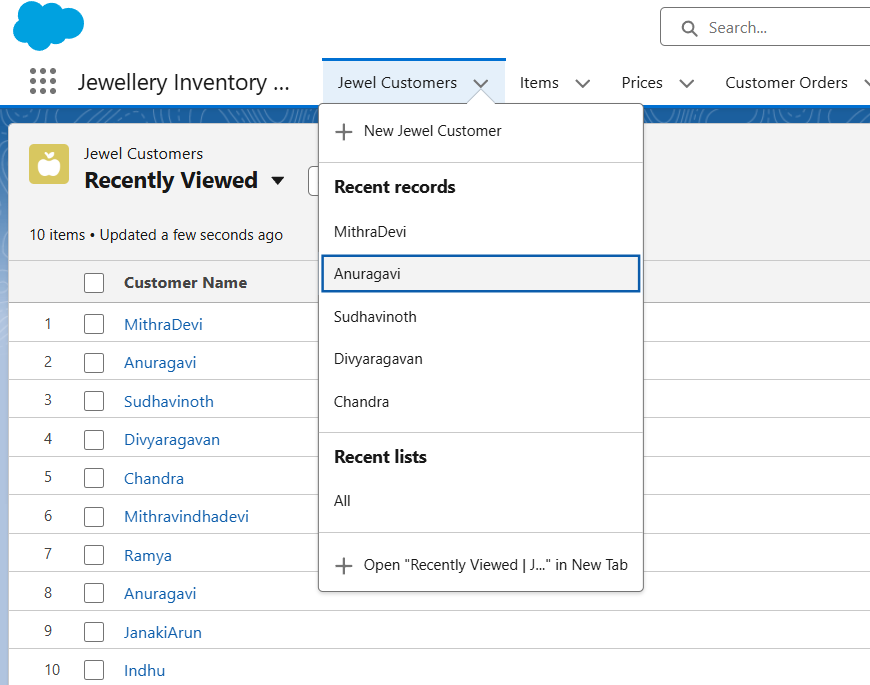
**Name**: Per to Worker  
 **Object**: Worker  
 **Purpose**: Extend the access of the Worker role without modifying the assigned profile. This allows the user to perform additional tasks while keeping the base profile unchanged.



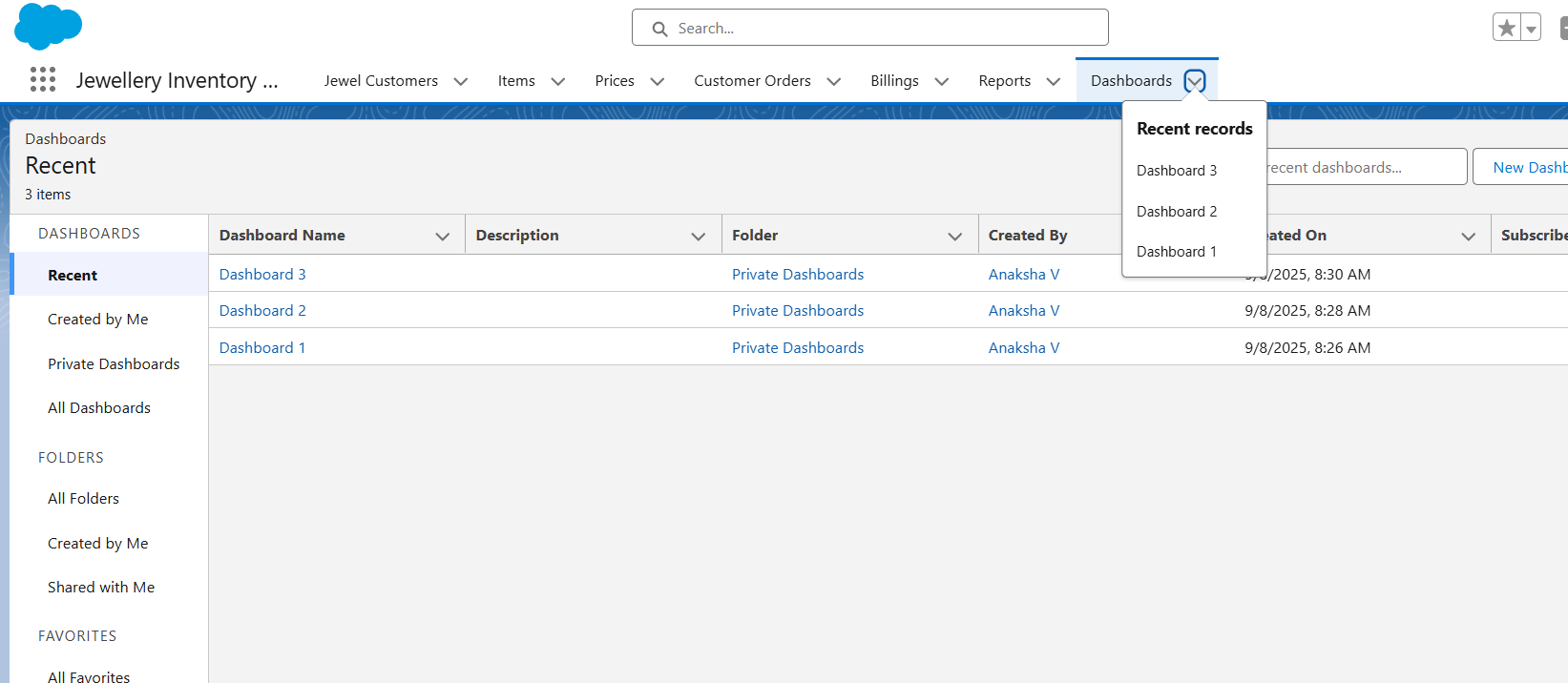


● **User Adoption:**

In the Jewel Inventory System, users are created and managed by the administrator to control access, assign permissions, and maintain customer records.

****

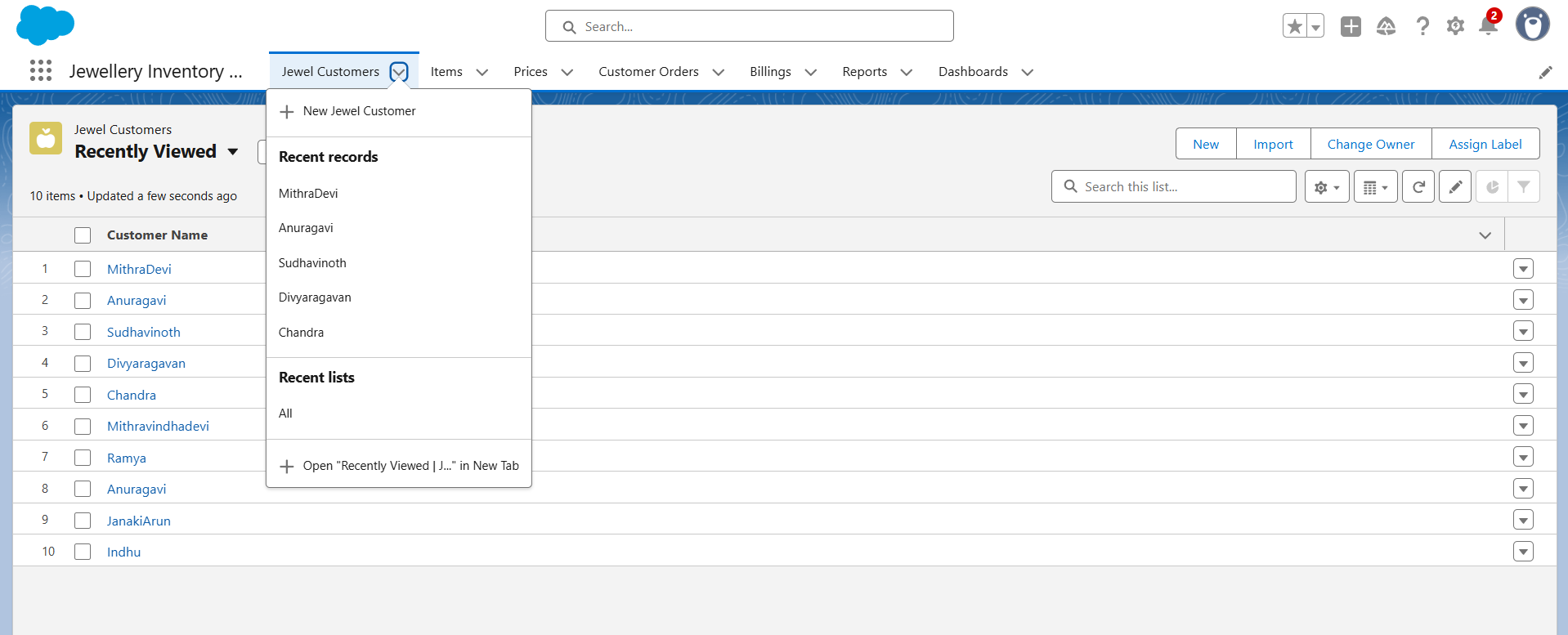
● **Reports and Dashboards:**

****

**CONCLUSION**

This project demonstrates how Salesforce can be configured to:

* Manage jewellery items, customers, orders, and billing records.
* Automate processes using Flows for seamless data entry.
* Use Apex Triggers and Trigger Handlers for field updates.
* Control access through Roles, Profiles, and Permission Sets.
* Generate reports and dashboards for sales and inventory insights.



The Jewel Inventory System serves as a model application showcasing Salesforce’s capability to streamline jewellery business operations, improve customer management, and ensure efficient inventory tracking.

**APPENDIX**

● **Source Code:** Provided in Apex Classes and Triggers

**billing.apxt:**

trigger UpdatePaidAmountTrigger on Billing\_\_c (before insert, before update) {

if (Trigger.isInsert) {

UpdatePaidAmountTriggerHandler.handleBeforeInsert(Trigger.new);

} else if (Trigger.isUpdate) {

UpdatePaidAmountTriggerHandler.handleBeforeUpdate(Trigger.oldMap, Trigger.new);

}

}

**billingHandler.apxc:**

public class UpdatePaidAmountTriggerHandler {

public static void handleBeforeInsert(List<Billing\_\_c> newBillings) {

for (Billing\_\_c billing : newBillings) {

billing.Paid\_Amount\_\_c = billing.Paying\_Amount\_\_c;

}

}

public static void handleBeforeUpdate(Map<Id, Billing\_\_c> oldBillingsMap, List<Billing\_\_c> updatedBillings) {

for (Billing\_\_c billing : updatedBillings) {

Billing\_\_c oldBilling = oldBillingsMap.get(billing.Id);

Decimal oldPaidAmount = oldBilling.Paid\_Amount\_\_c;

billing.Paid\_Amount\_\_c = oldPaidAmount + billing.Paying\_Amount\_\_c;

}

}

}