**CRM APPLICATION FOR JEWEL MANAGEMENT**

**College Name: VET INSTITUTE OF ARTS AND SCIENCE**

**College Code: asrubb**

**TEAM ID:**

**TEAM MEMBERS:**

**Team Leader Name: Harshini T**

**Email: harshinit23aid@vetias.ac.in**

**Team Member 1: Avisya V**

**Email: avisyav23aid@vetias.ac.in**

**Team Member 2: Dharmasamvarthini S**

**Email: dharmasamvarthinis23aid@vetias.ac.in**

**Team Member 3 : Priyadharshini S**

**Email: priyadharshinis23aid@vetias.ac.in**

**Team Member 4: Vikashini N**

**Email: vikashinin23aid@vetias.ac.in**

**1.INTRODUCTION**

**1.1 Project Overview**

The **CRM (Customer Relationship Management) for**  **Jewelry Management System (JMS)** is a Salesforce-based application designed to streamline the processes associated with a jewelry business. It handles inventory management, customer and vendor relationships, sales and order tracking, and repair services. The system includes automation features such as flows, approval processes, and email alerts to ensure efficient operations.



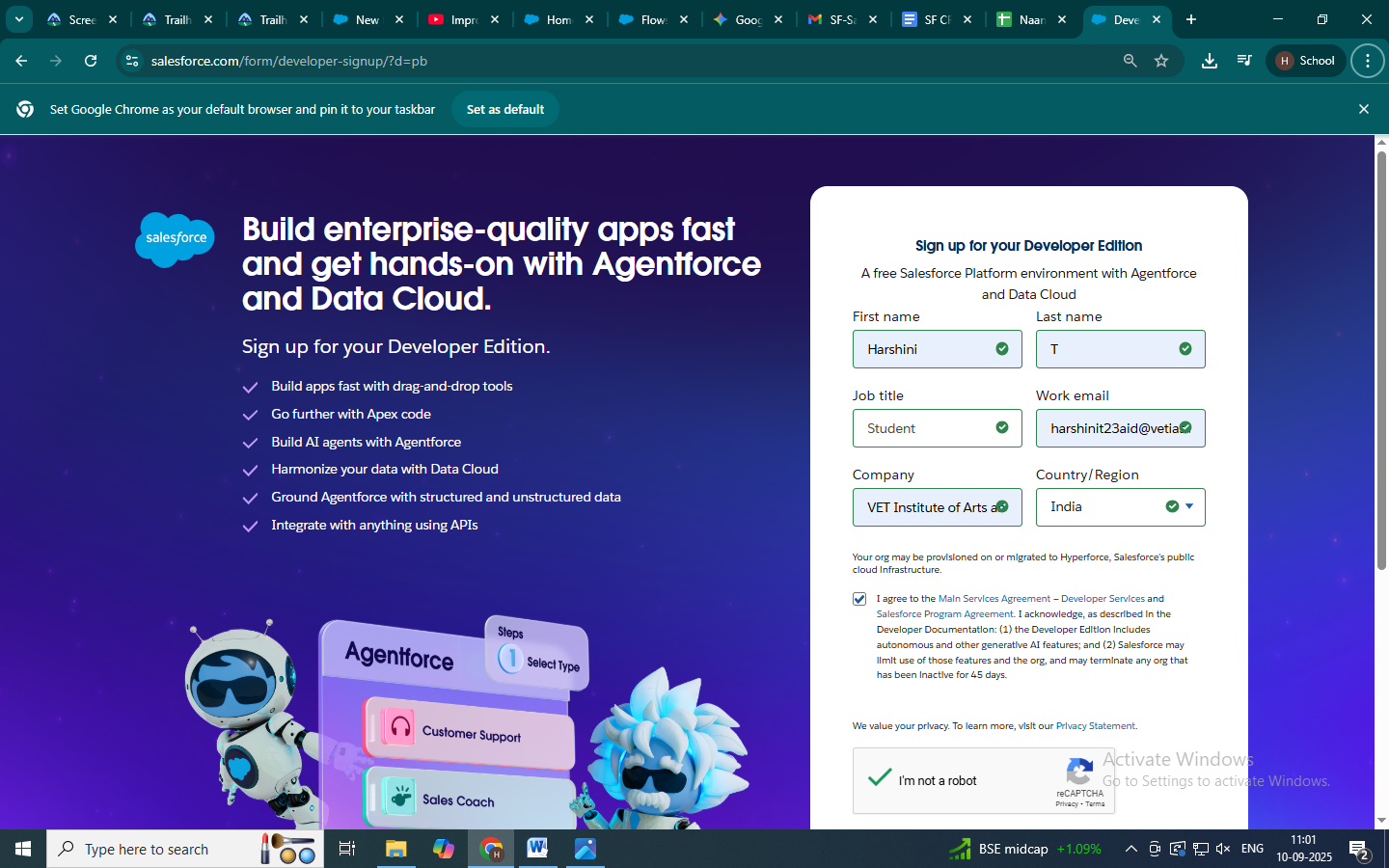
**1.2 Purpose**

The main objective of the **Jewelry Management System (JMS)** is to enable organizations to efficiently manage their jewelry inventory, customer interactions, sales processes, and after-sales services. It reduces manual intervention, improves accuracy in stock management and customer data, and ensures better compliance with business standards and seamless communication with clients and vendors.

**DEVELOPMENT PHASE**

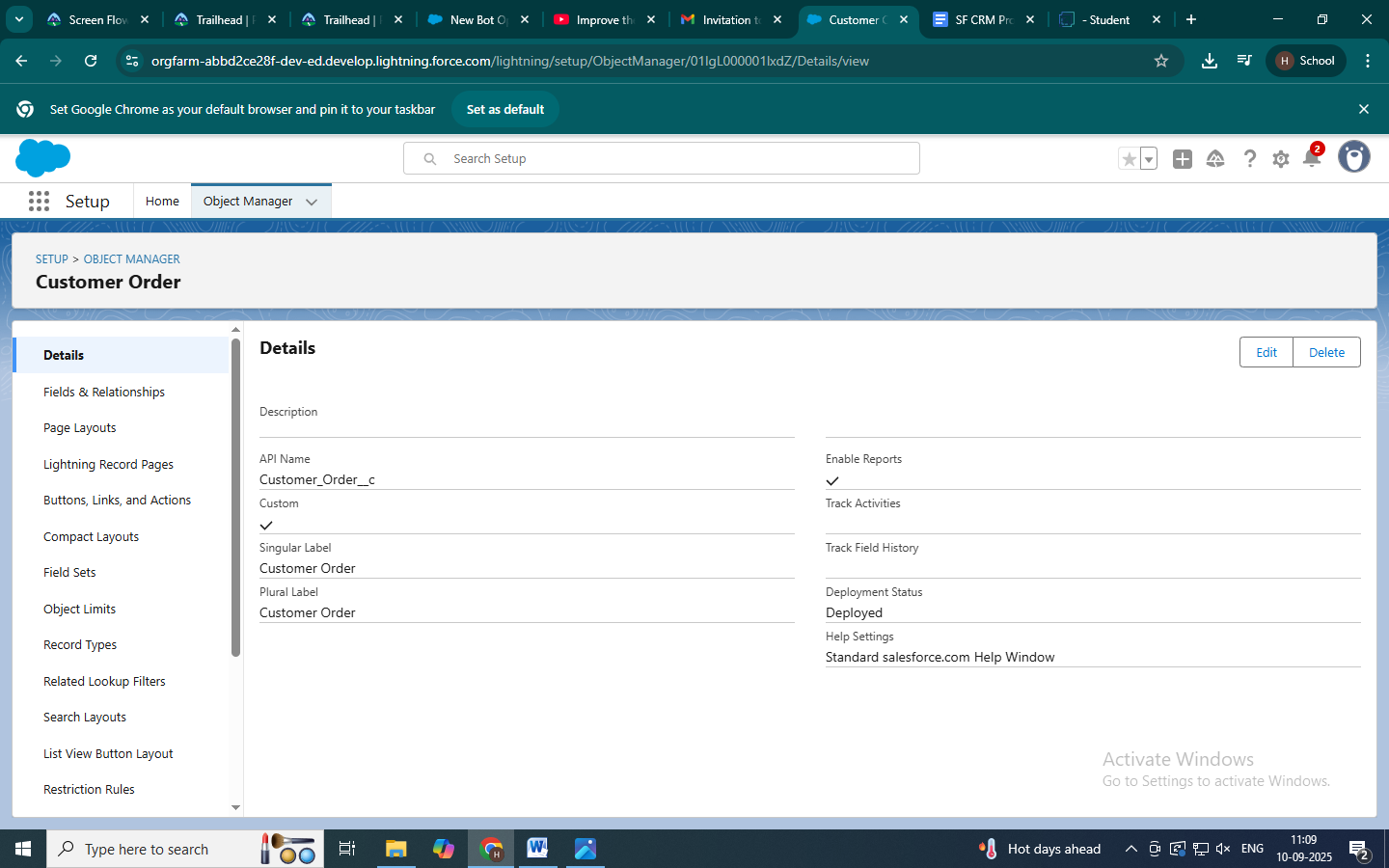
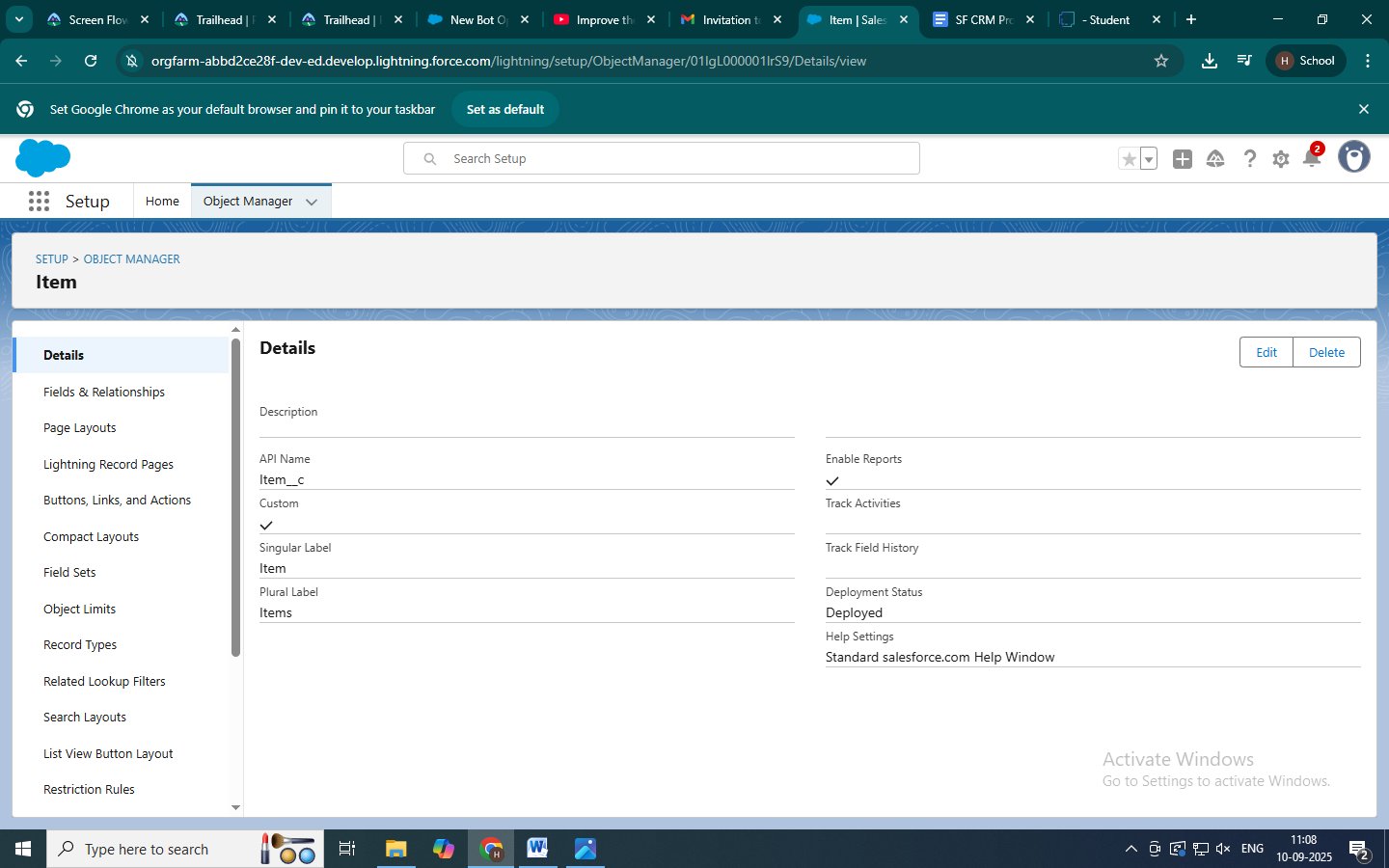
**Creating Developer Account:**

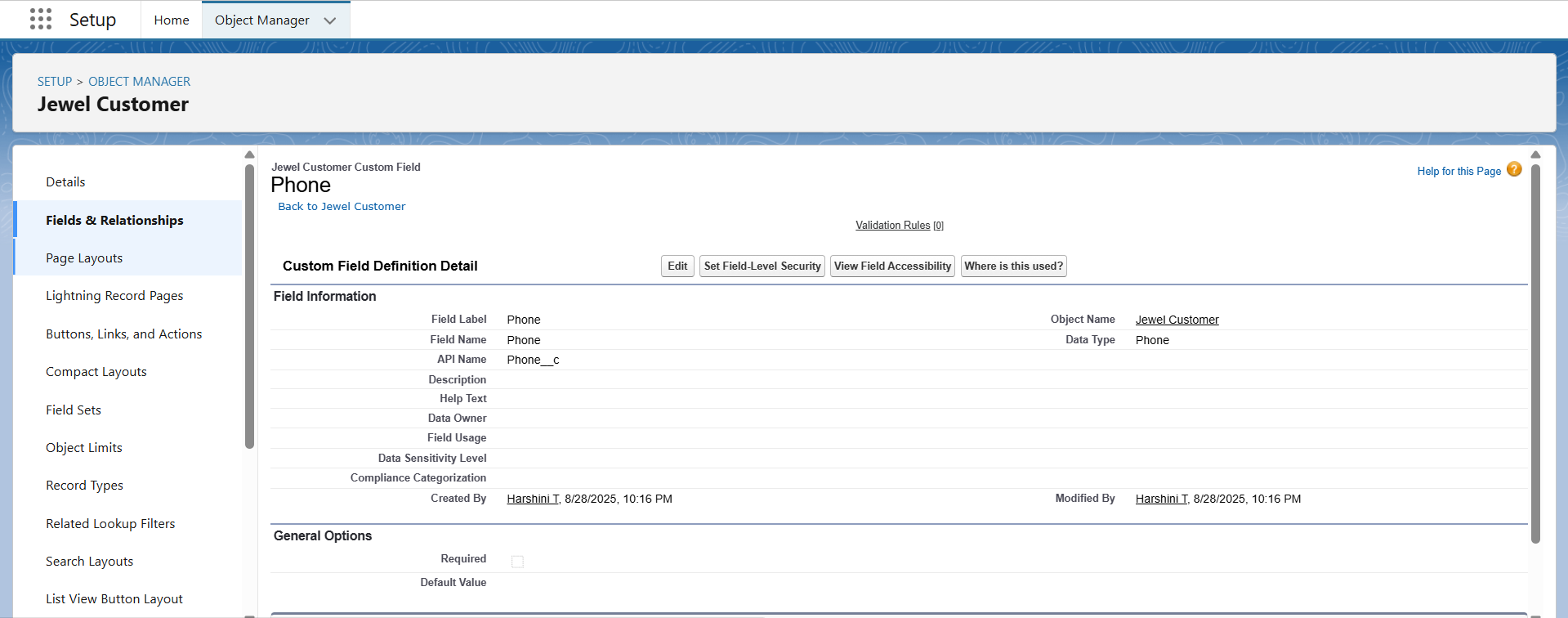
By using this URL **- https://www.salesforce.com/form/developer-signup/?d=pb**

****

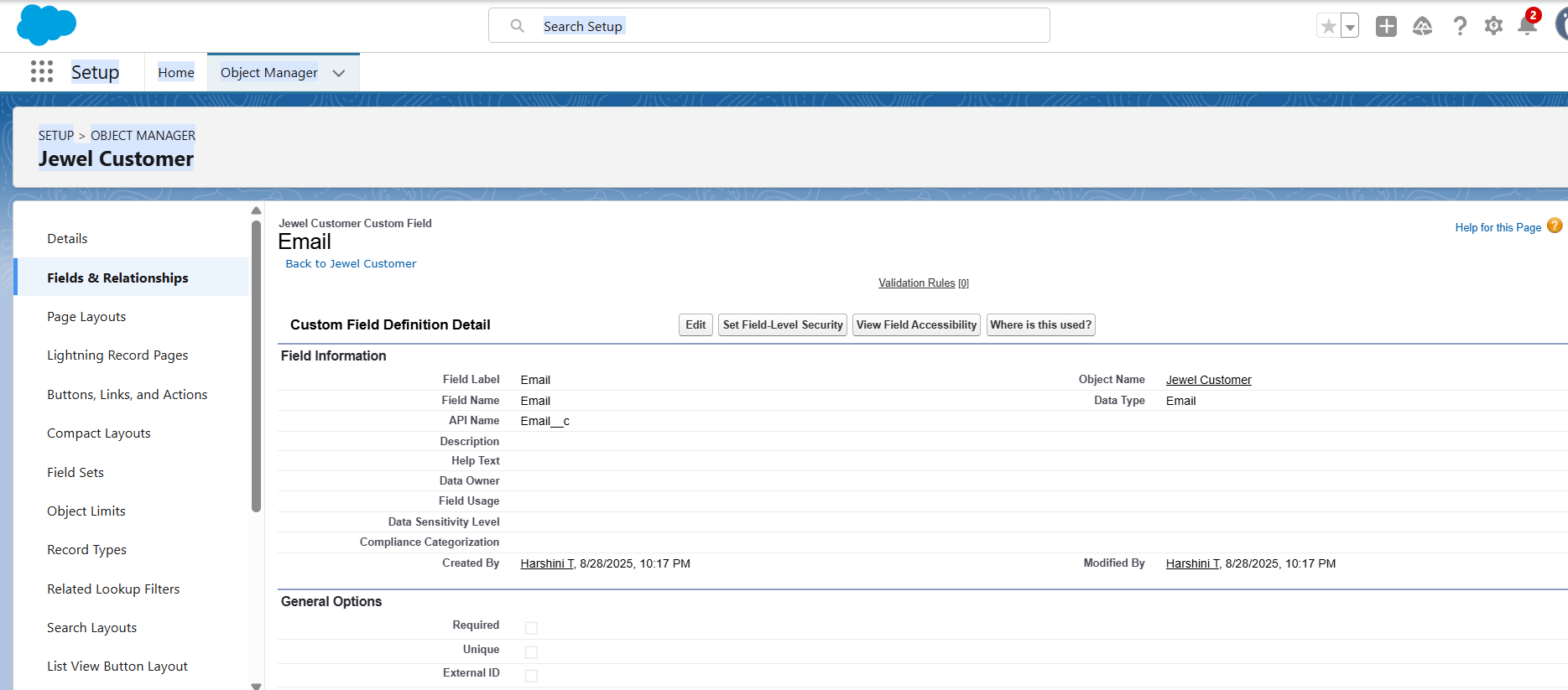
● Created objects: Jewel Customer, Item, Customer Order, Price, Billing

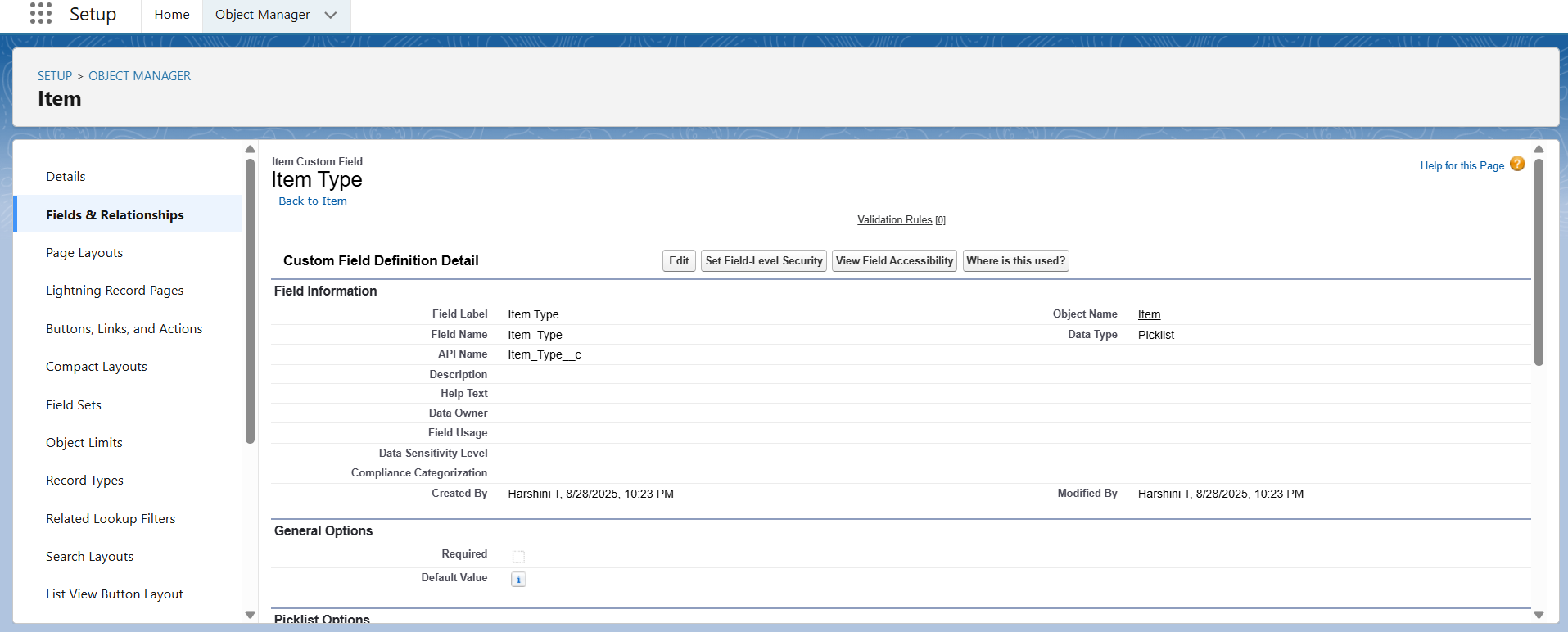
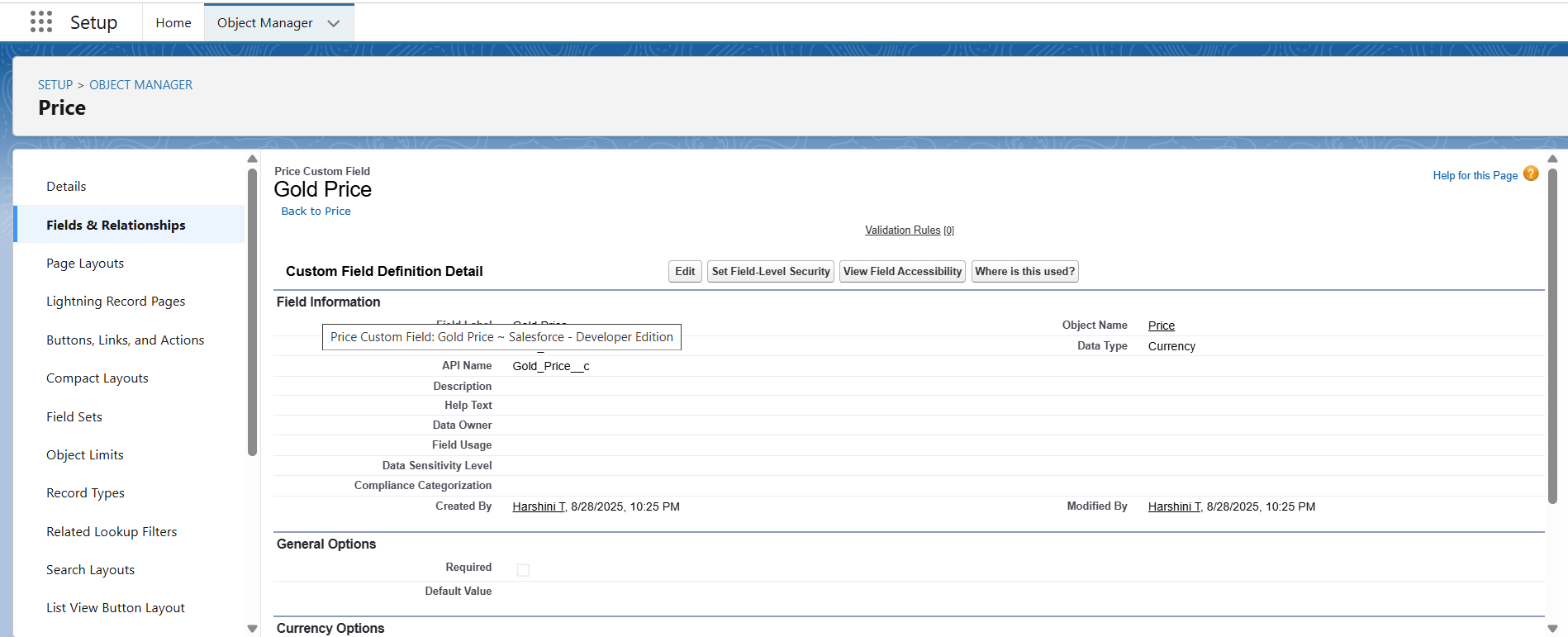
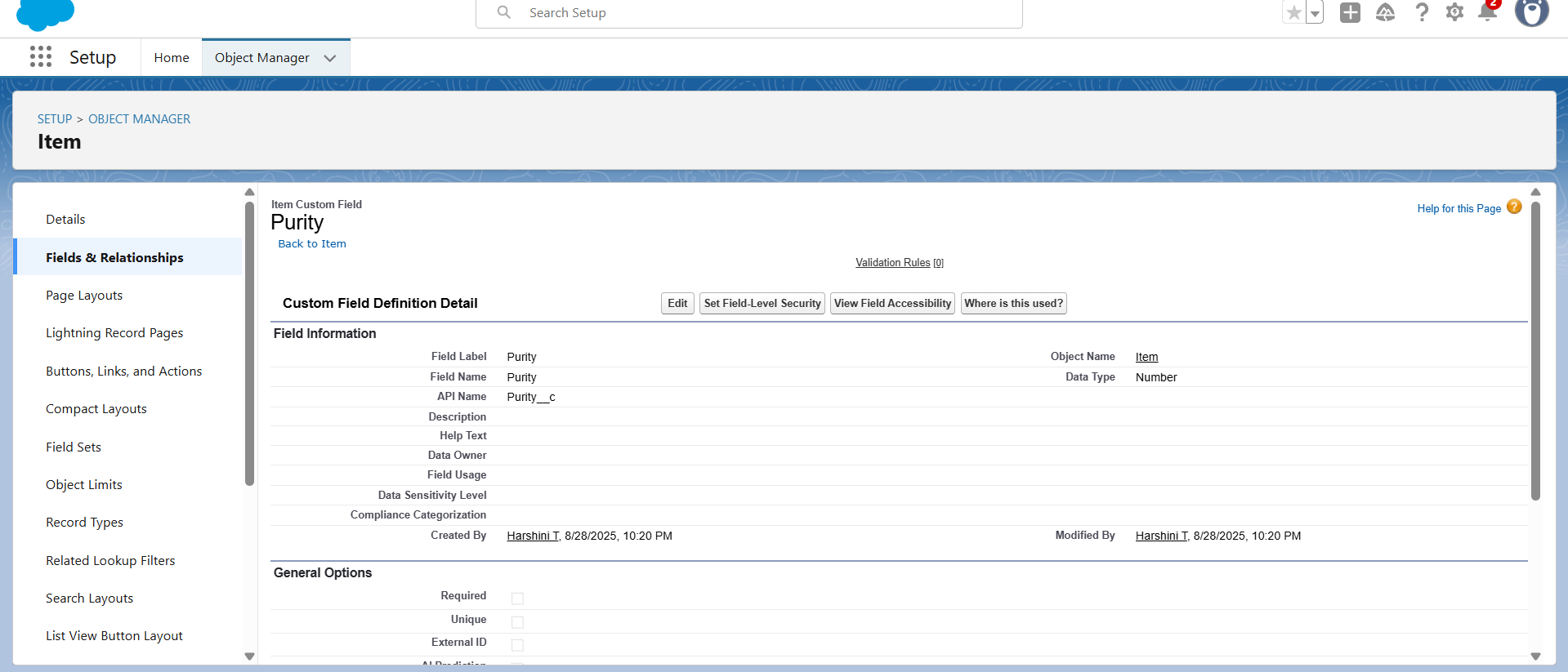




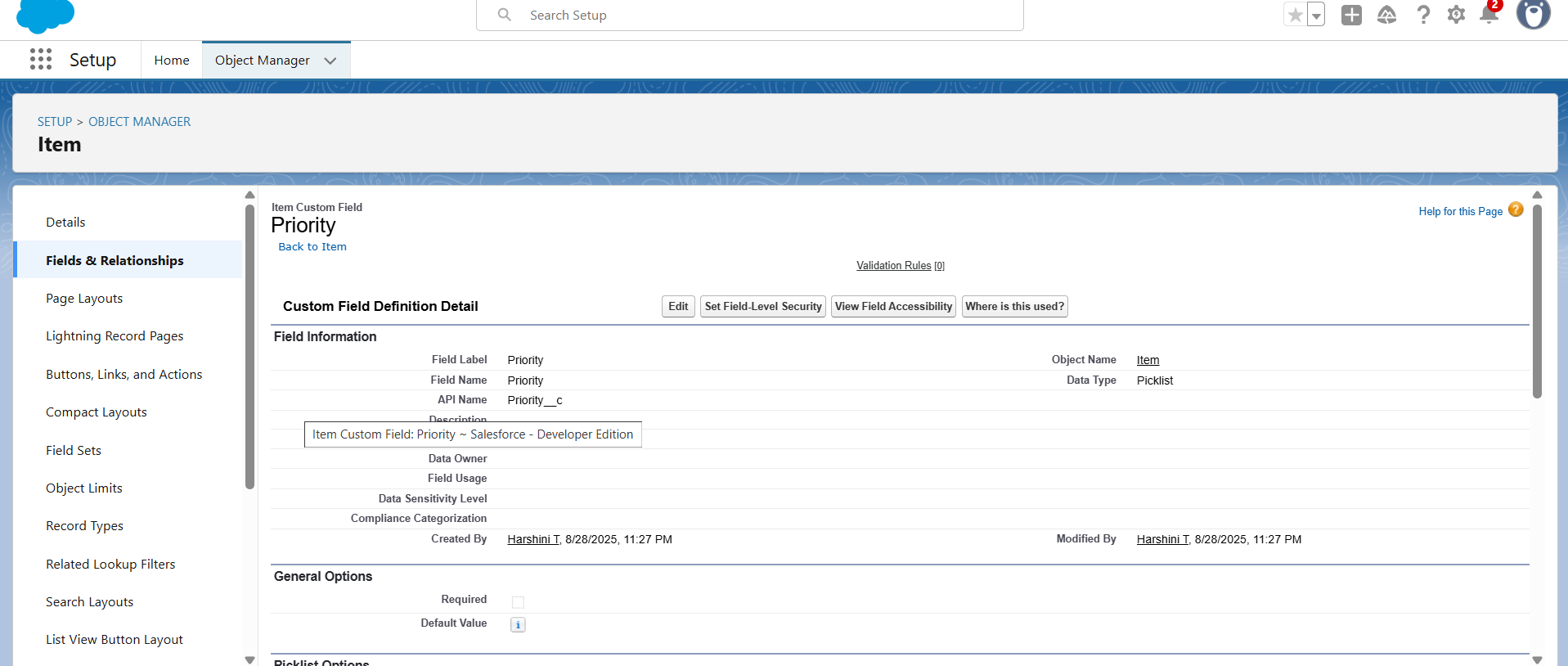


● Configured fields and relationships

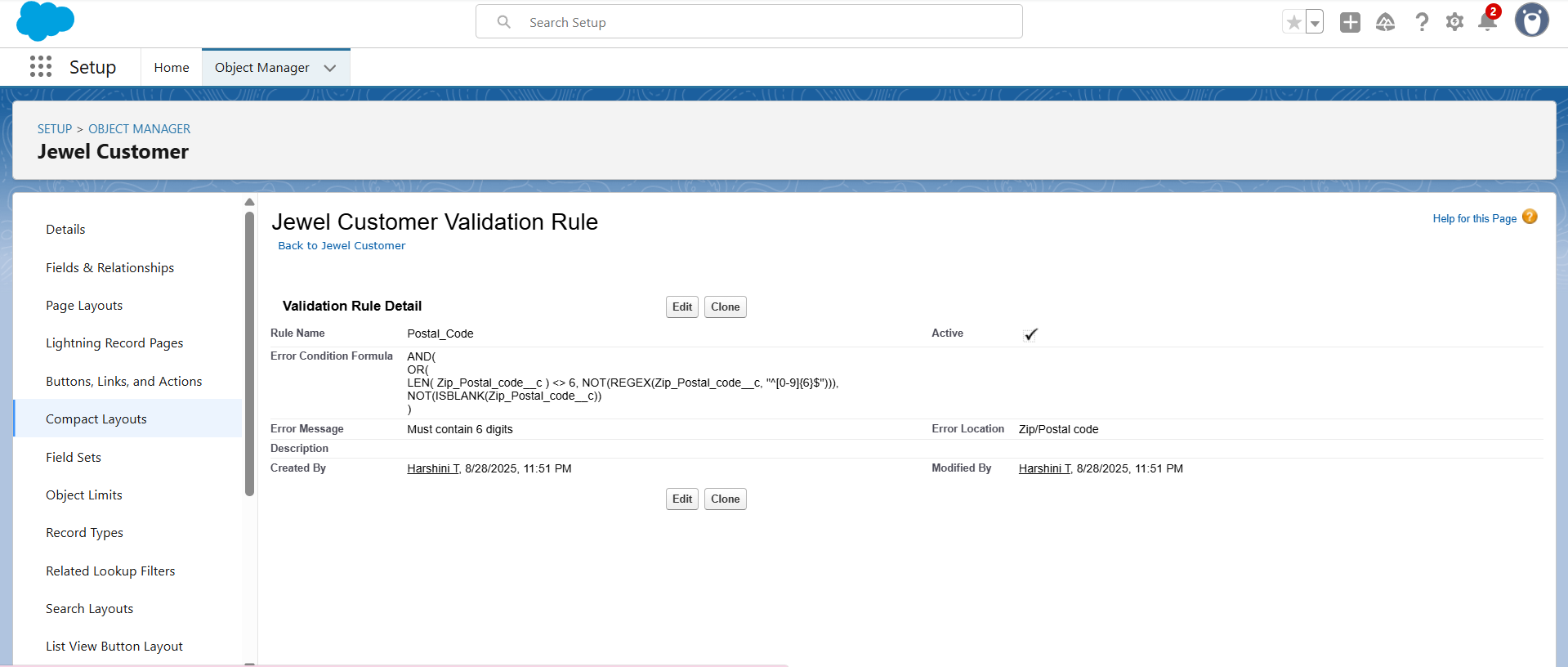


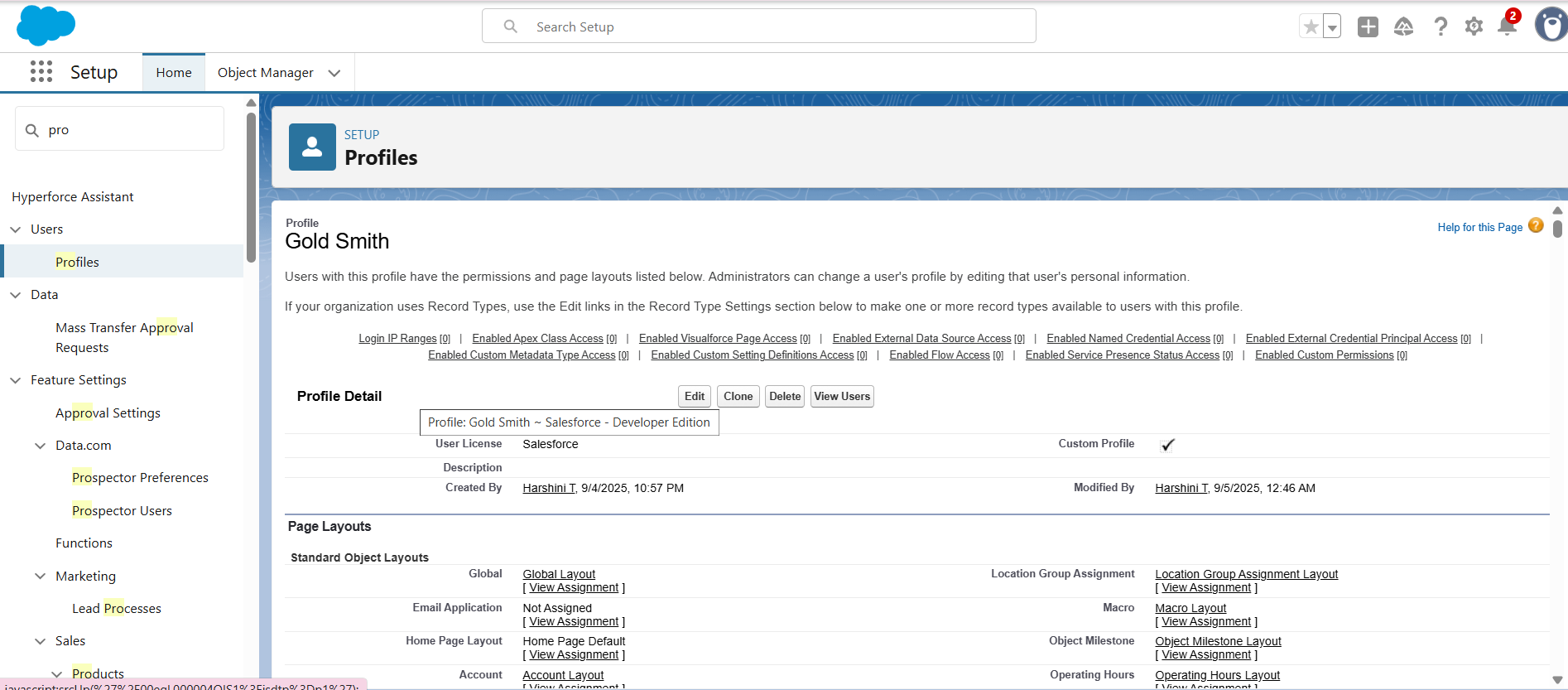


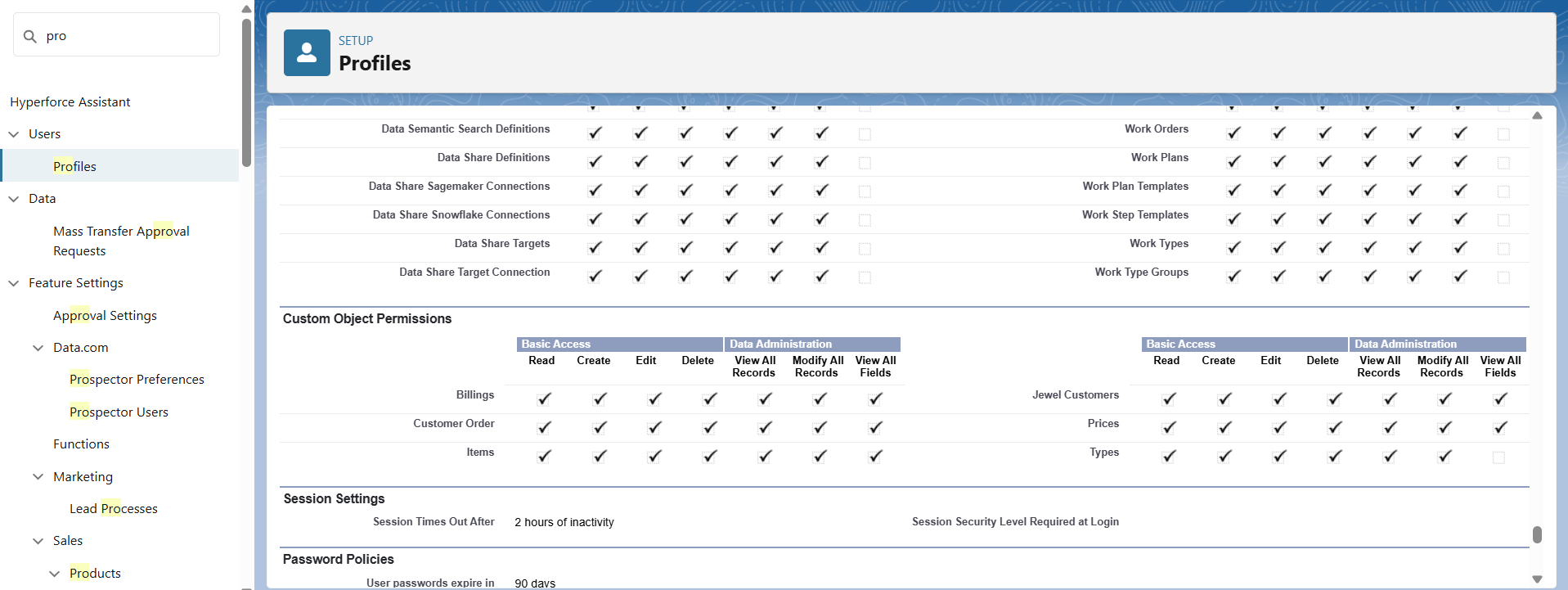
●Develop Lightning App with relevant tabs



●Created a validation rule for a Jewel Customer and Item object to ensure a valid sales date.

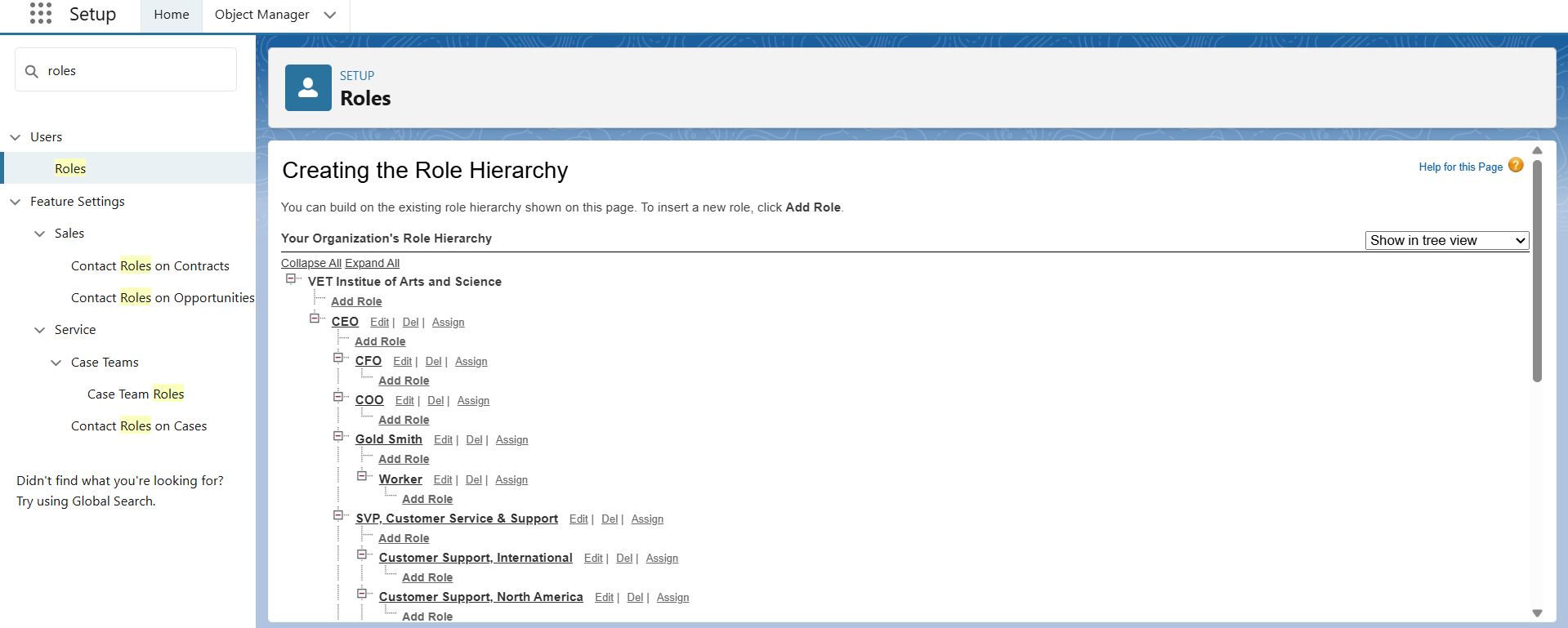




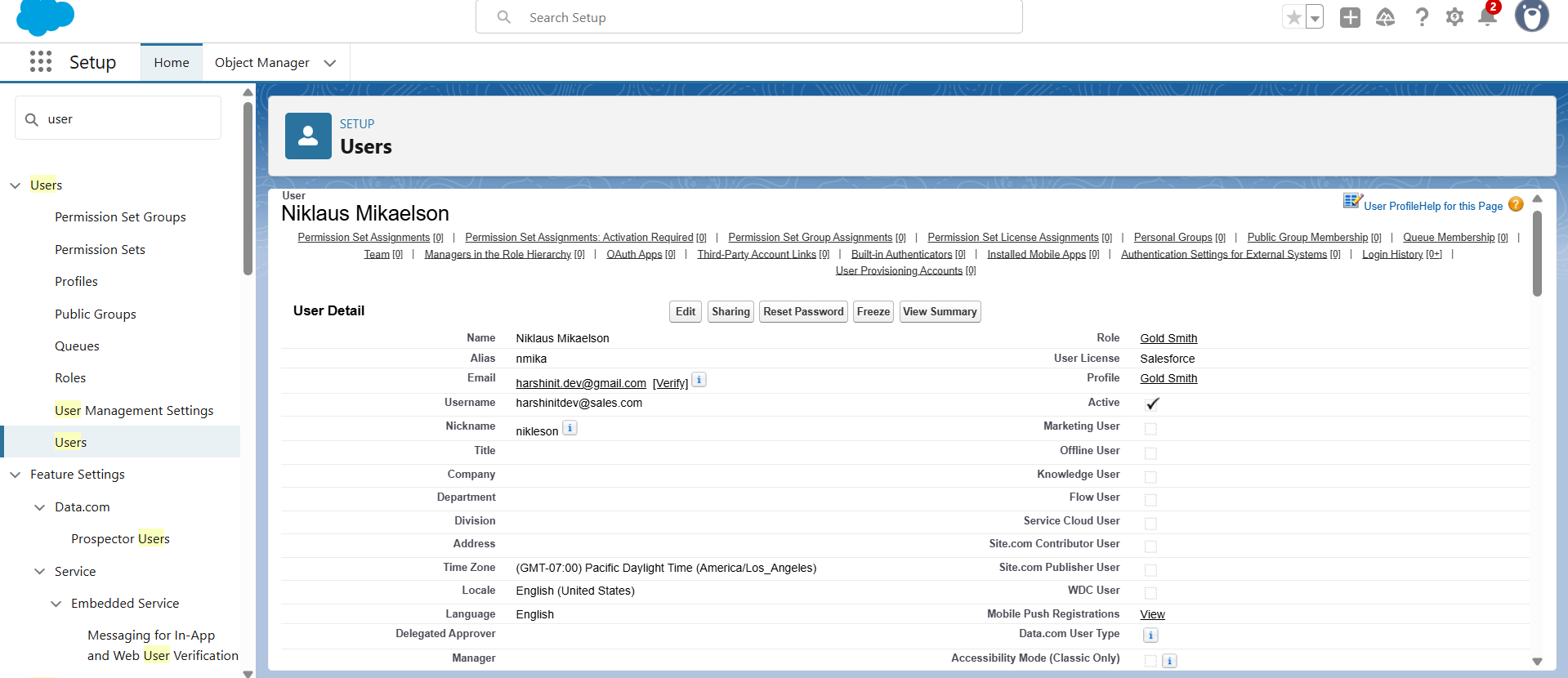


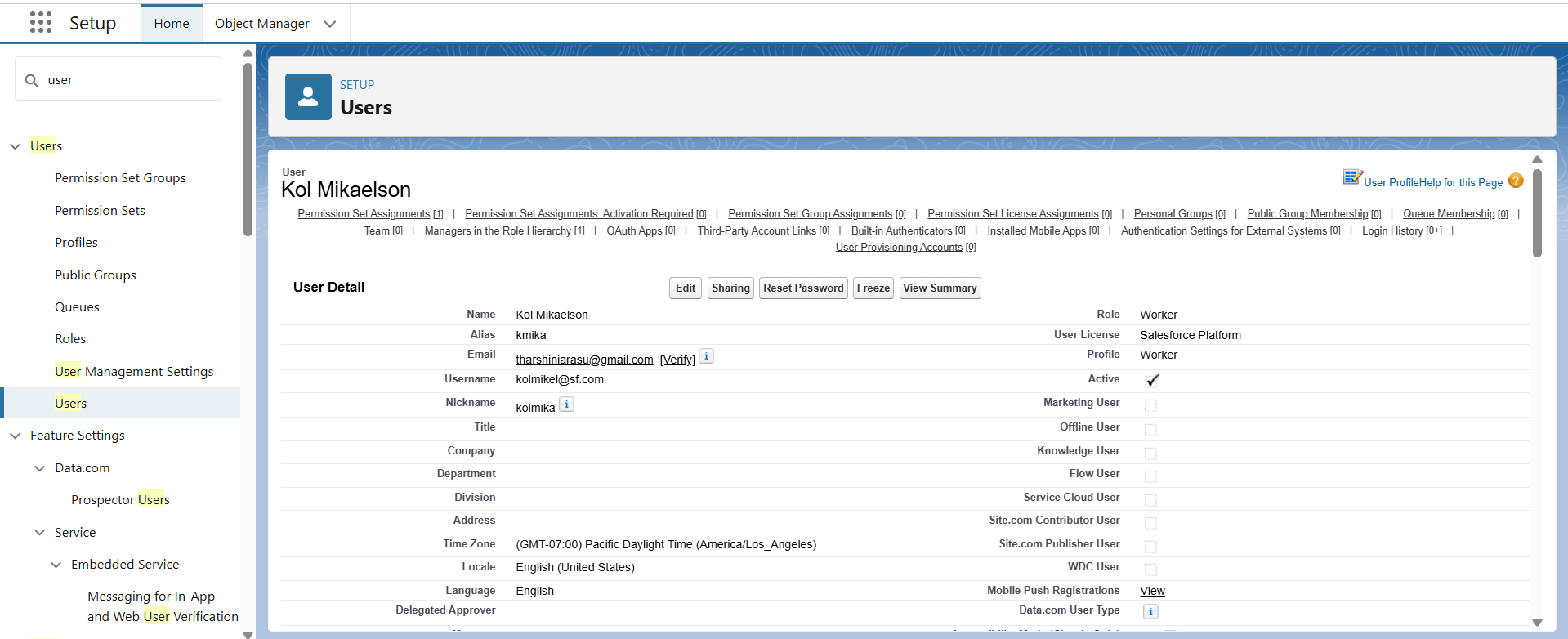
● Created Profiles: Gold Smith and Worker

● Created Roles : Gold Smith (reports to CEO) and Worker (reports to gold smith)



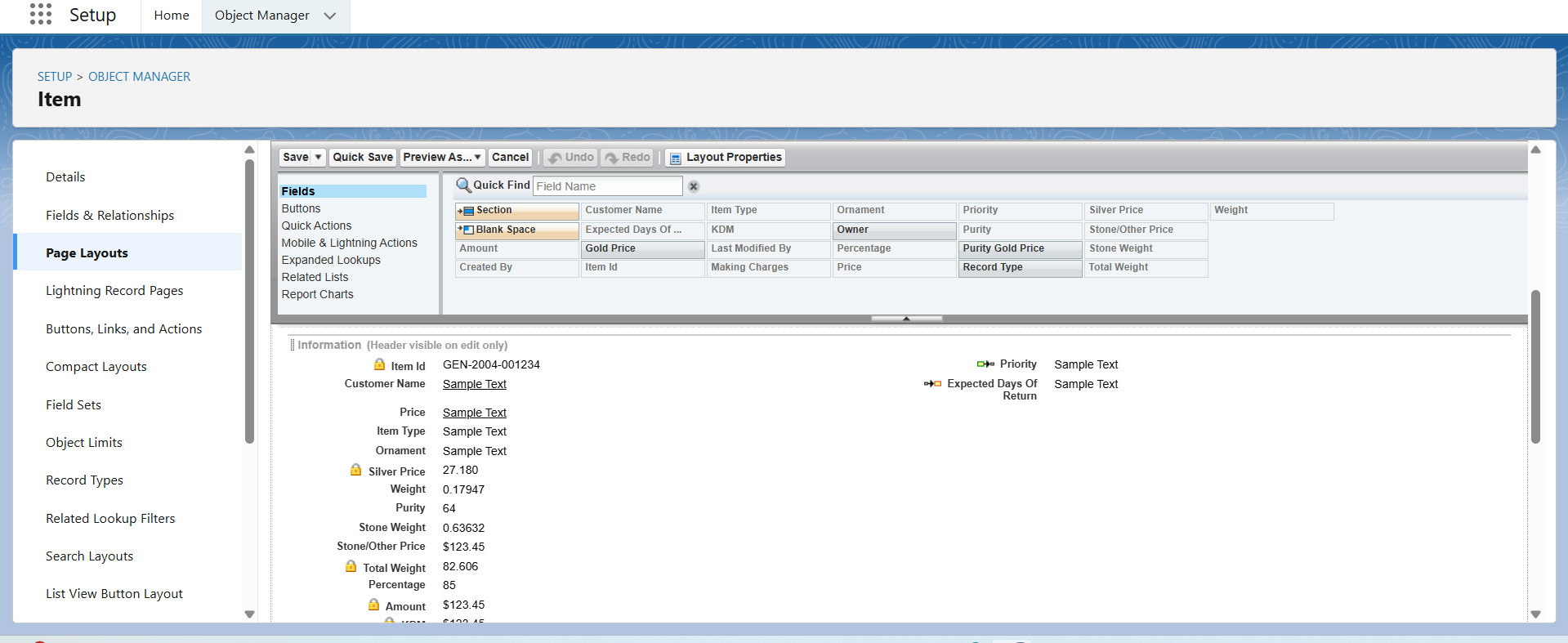
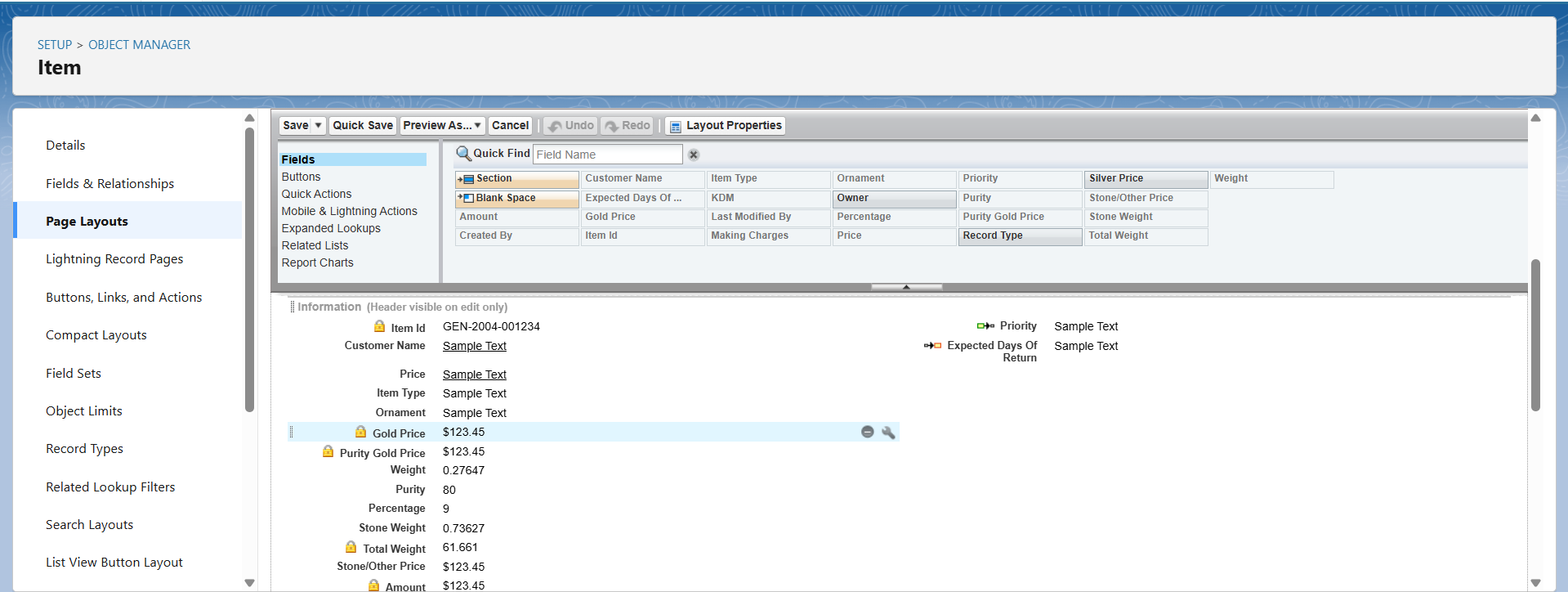
● Created Users:Niklaus Mikaelson,Kol Mikaelson

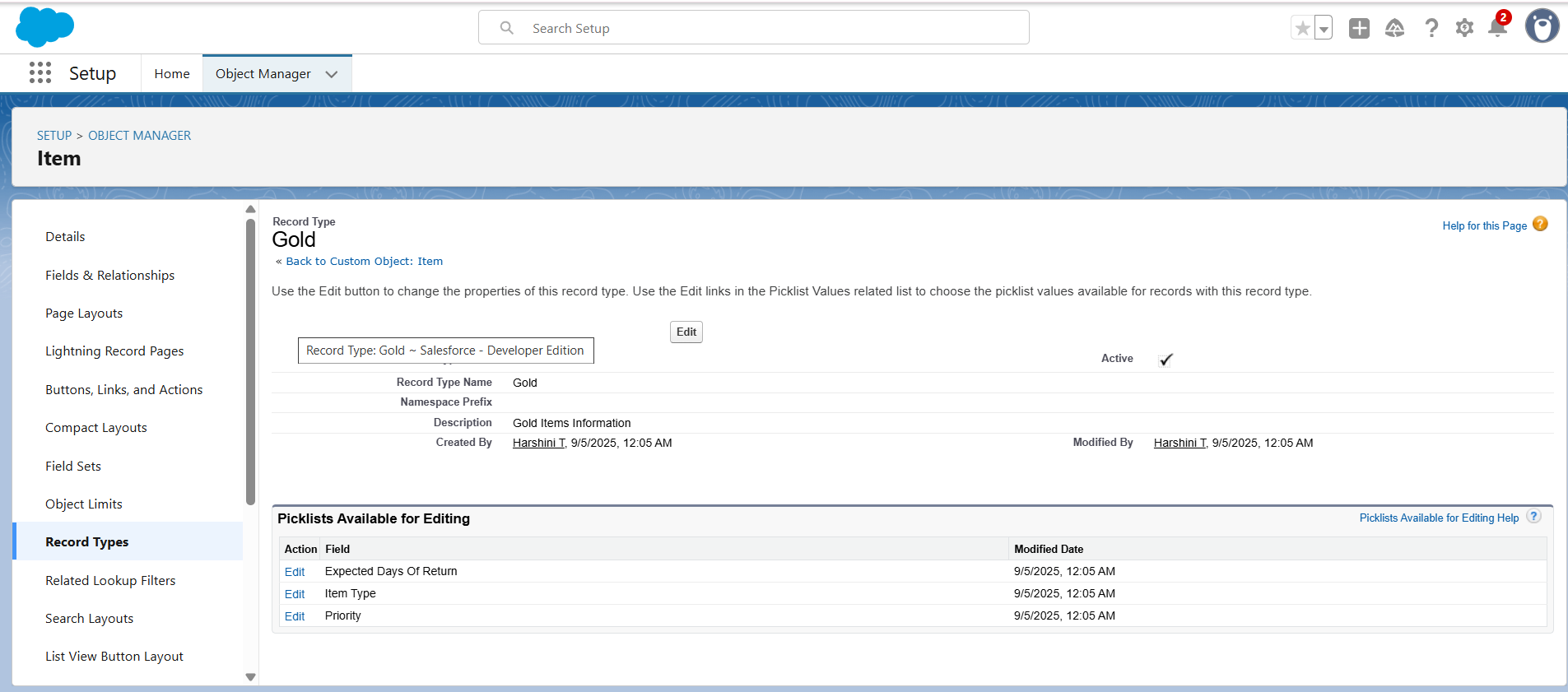


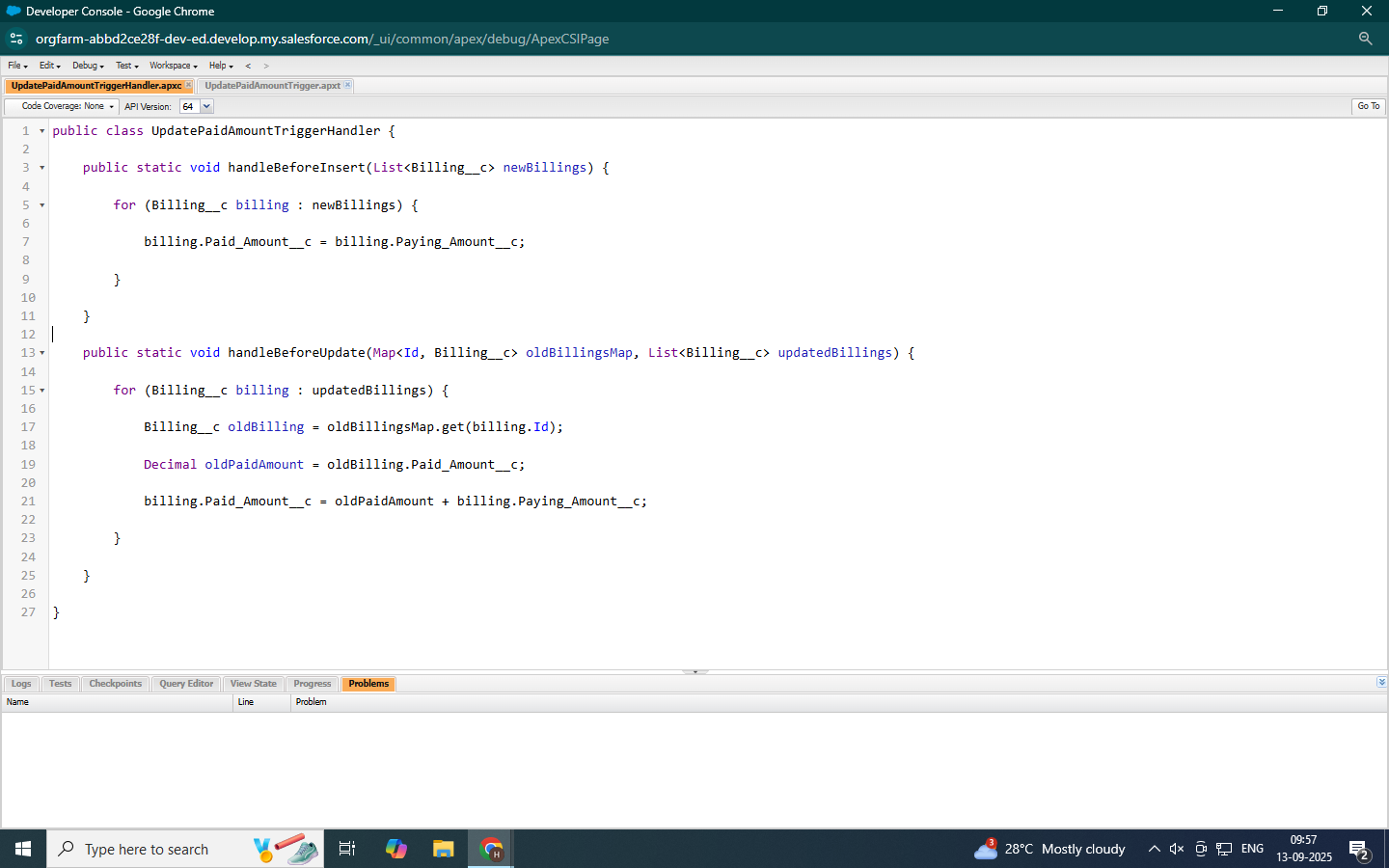
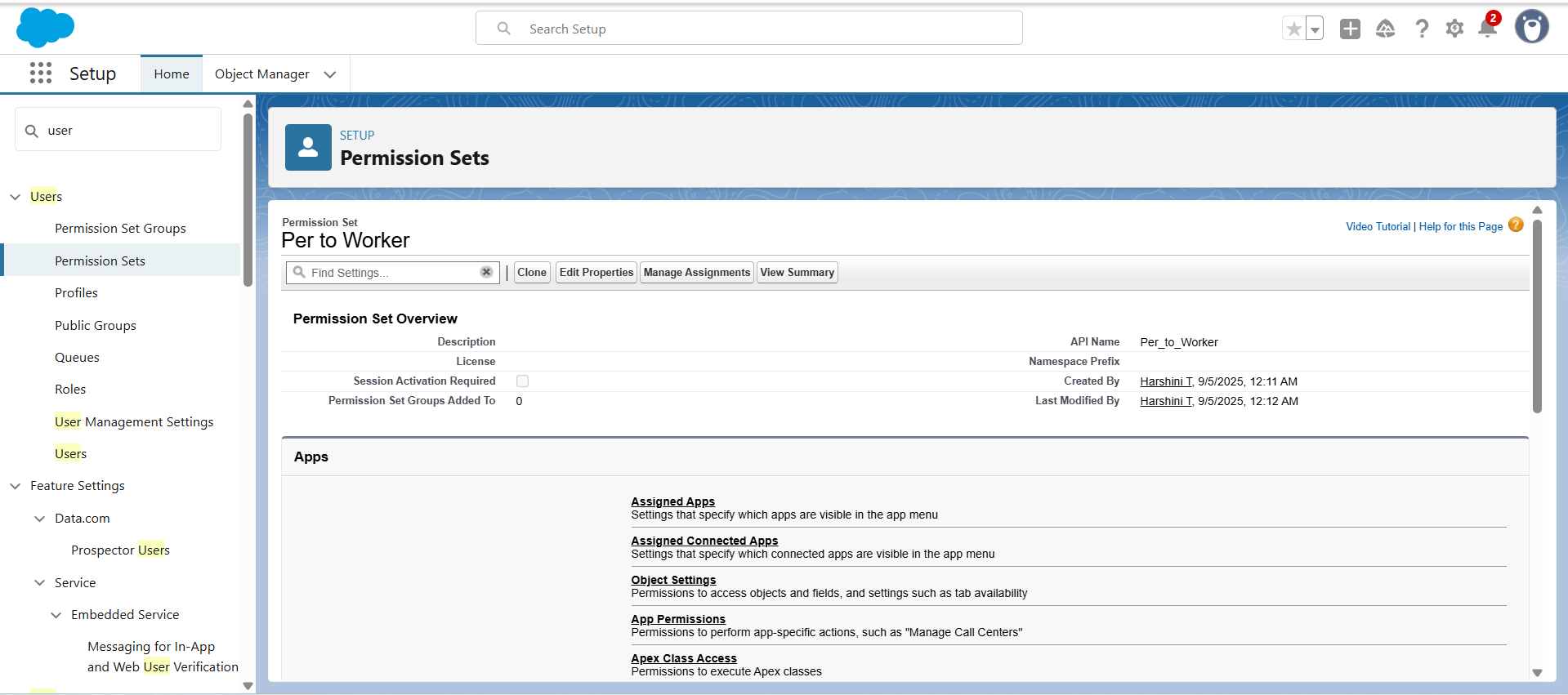


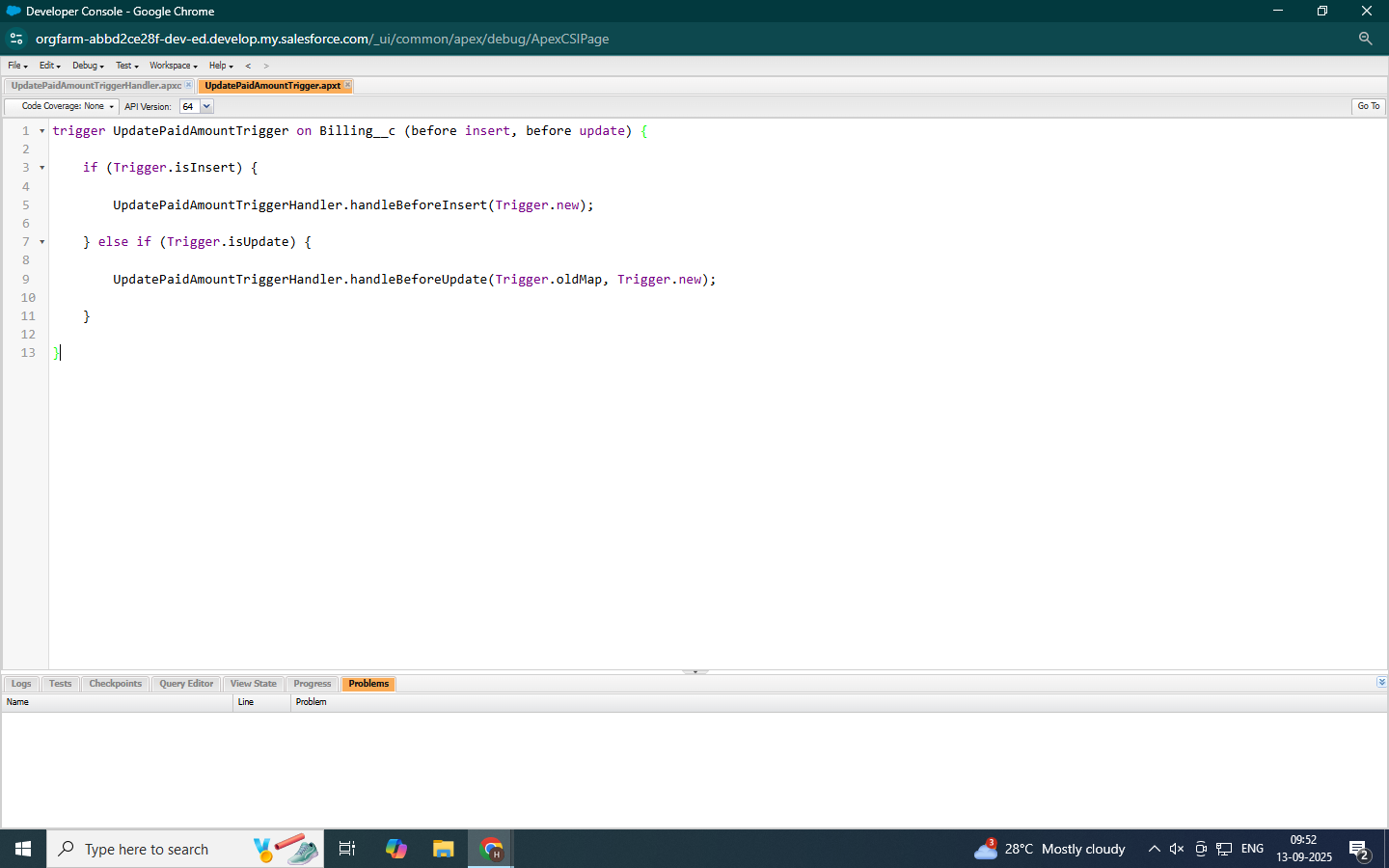
●Page Layout are used to customise the design and organise detail and edit pages of records in Salesforce

Created Layouts for Gold and Silver

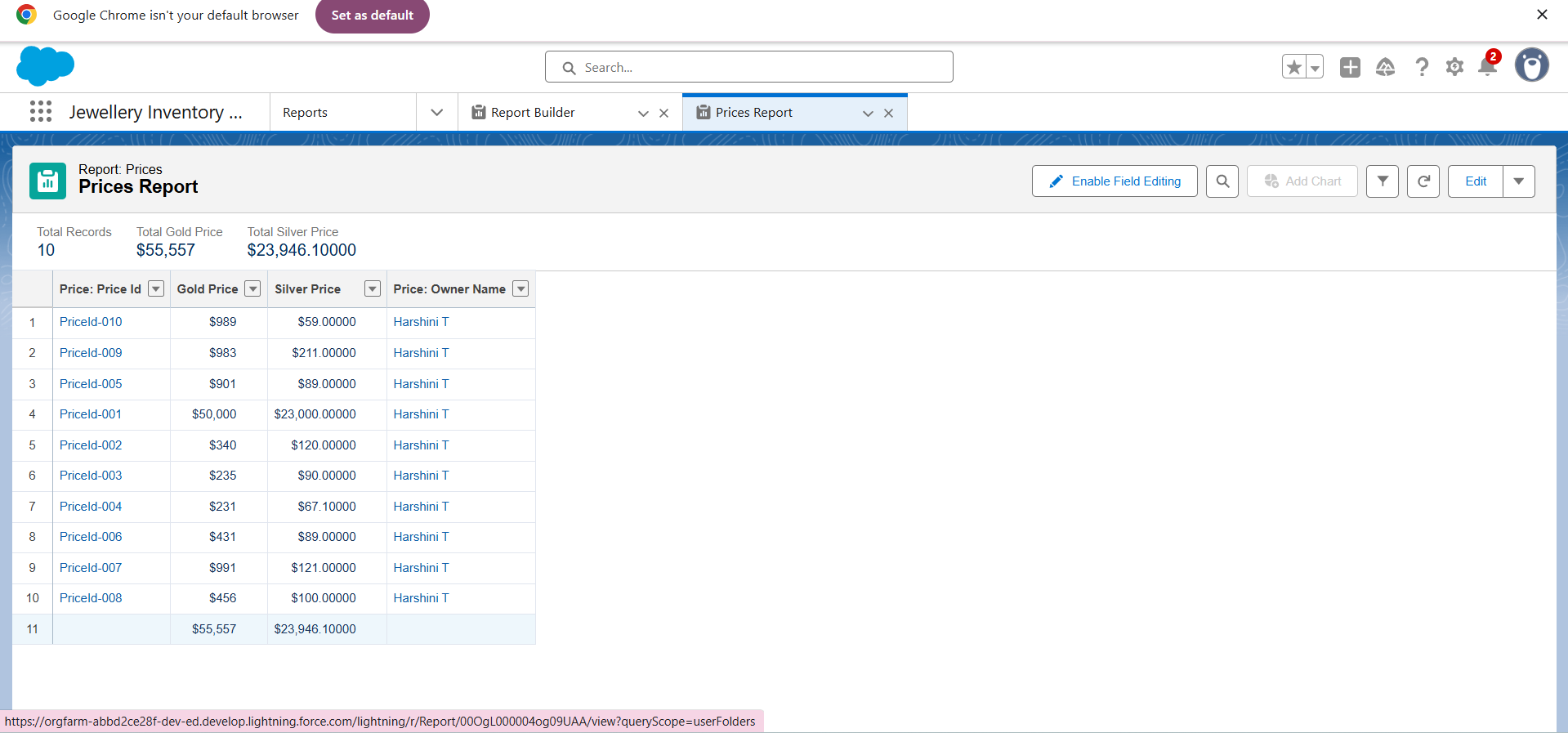
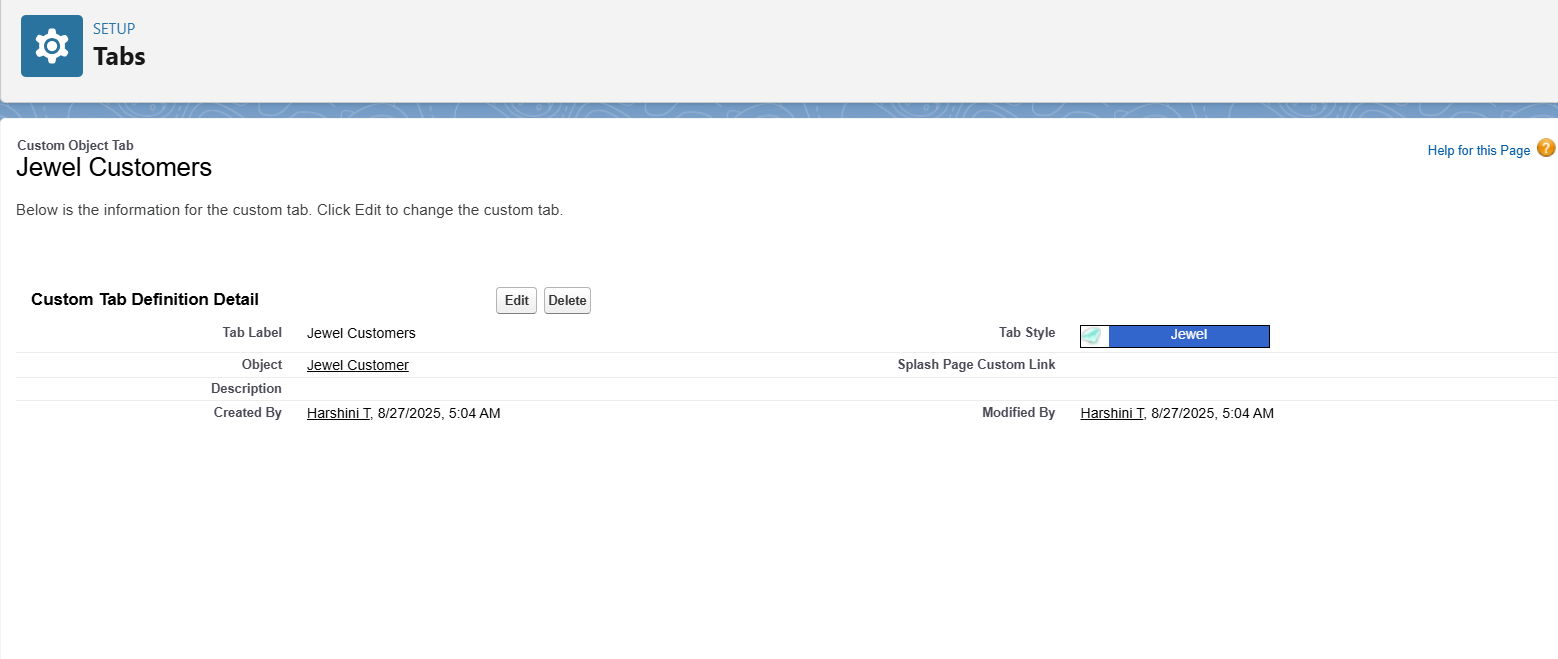


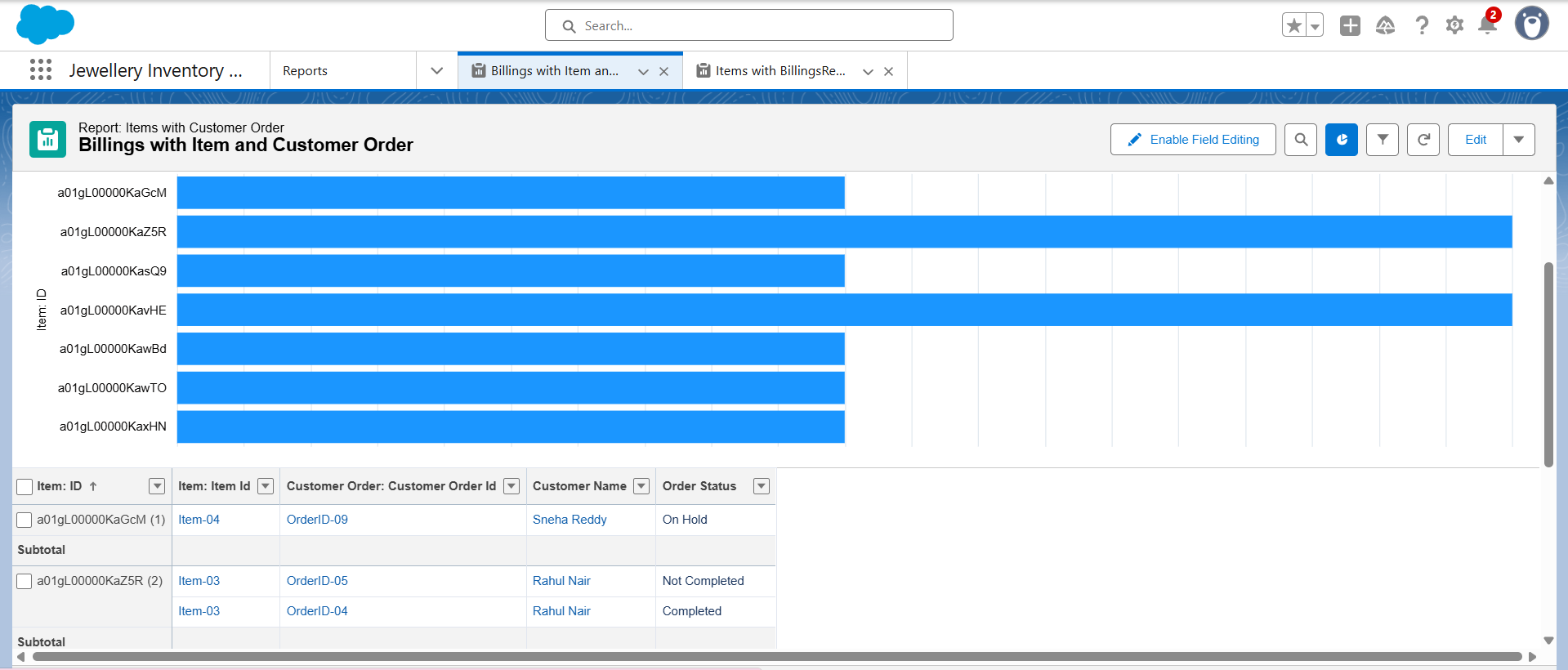
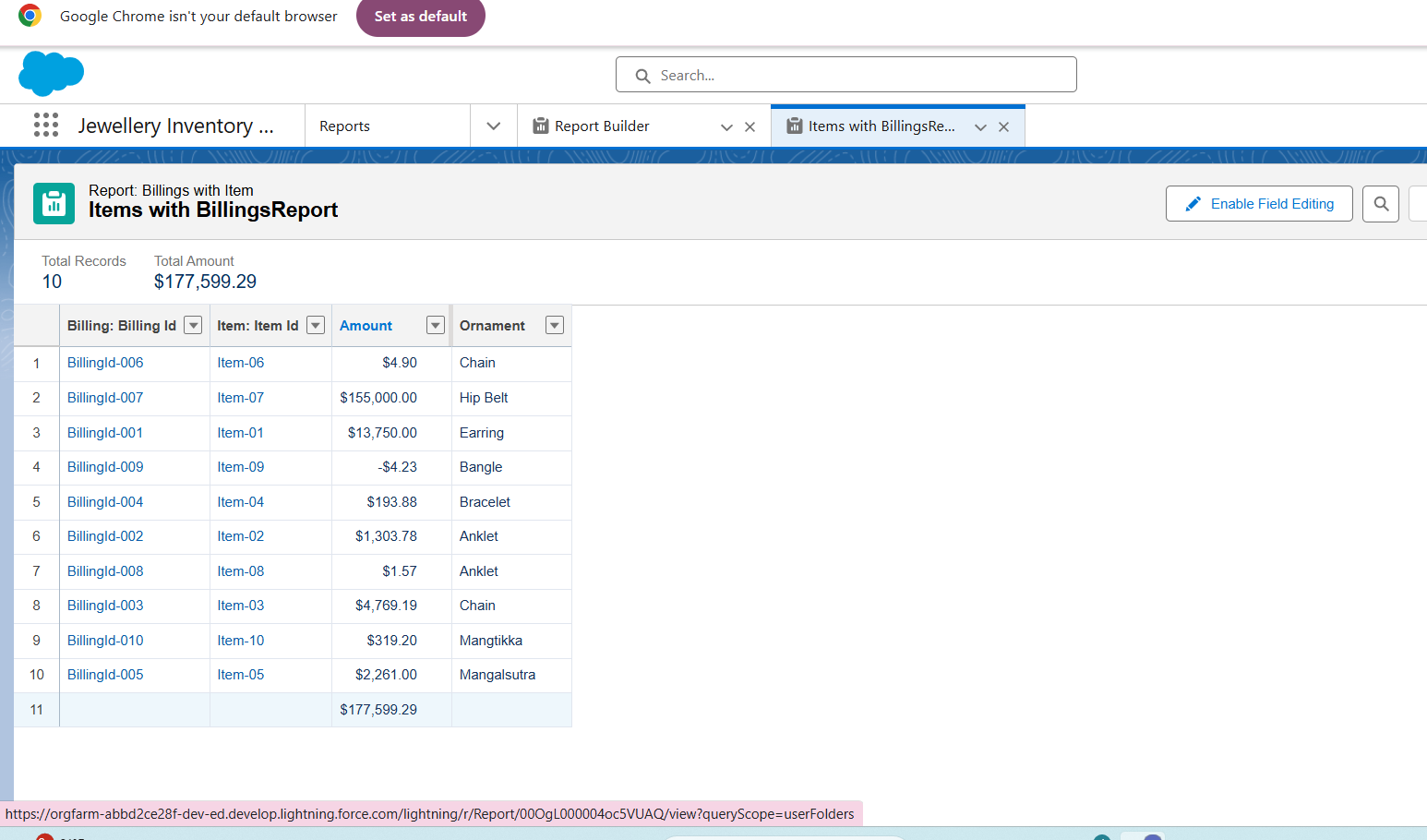


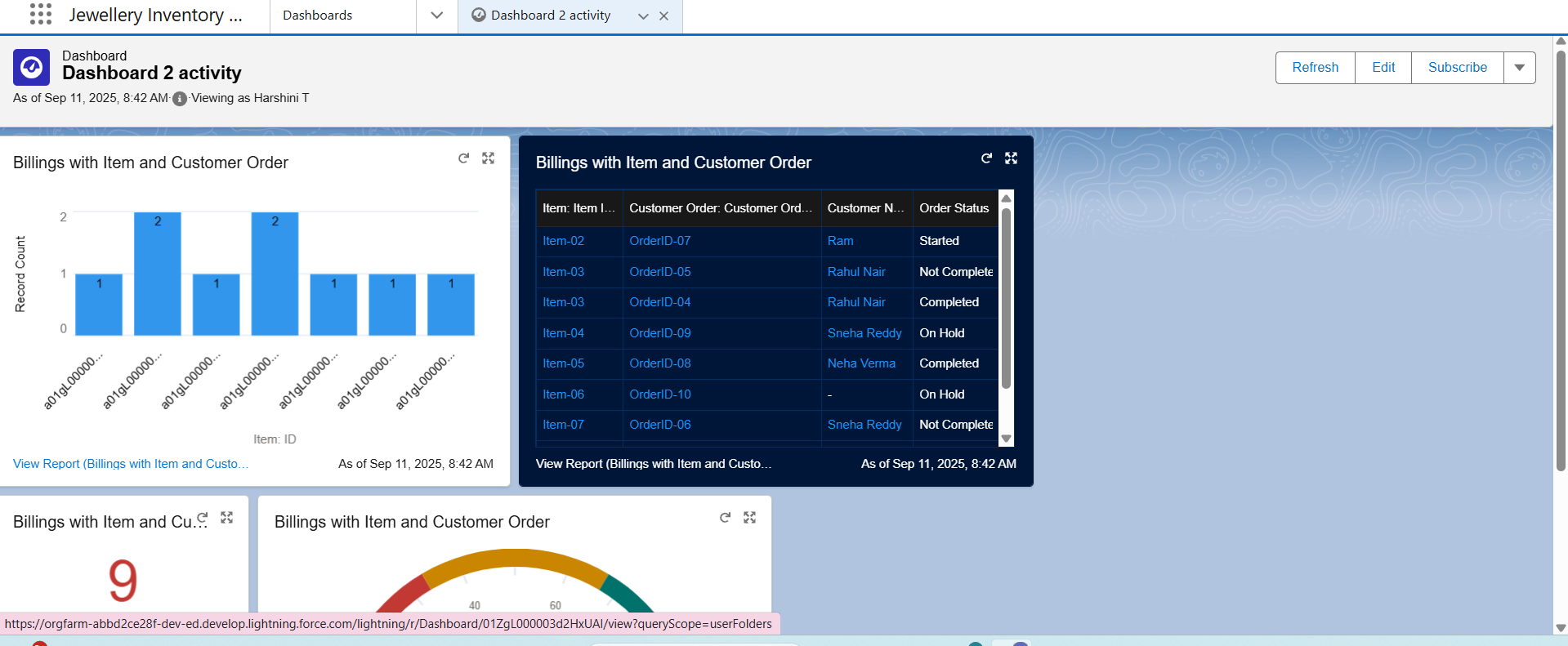




● Built and tested email templates for new order confirmations, sales receipts, repair status updates, and customer follow-up.

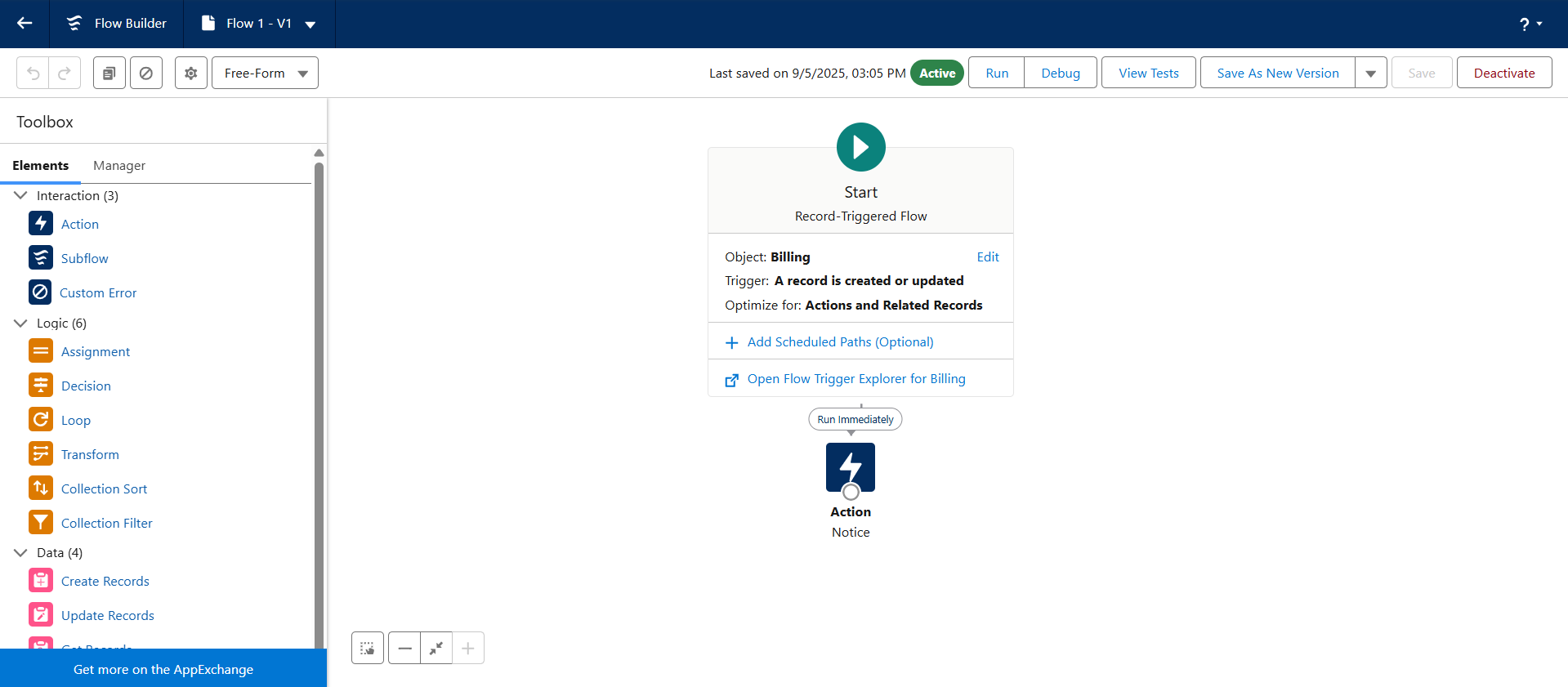






● Approval Process creation

For Tenant Leaving:

For Check for Vacant:



● Apex Trigger

Create an Apex Trigger



Create an Apex Handler class



● FLOWS





● Schedule class:

Create an Apex Class



Schedule Apex class







**FUNCTIONAL AND PERFORMANCE TESTING**

**Performance Testing**

● Trigger validation by entering duplicate tenant-property records



● Validation Rule checking



● Test flows on payment update

● Approval process validated through email alerts and status updates



**RESULTS**

**Output Screenshots**

● Tabs for Property, Tenant, Lease, Payment

● Email alerts

● Request for approve the leave



● Leave approved

● Leave rejected

● Flow runs

● Trigger error messages



● Approval process notifications



**ADVANTAGES & DISADVANTAGES** .

**CONCLUSION**

The **Jewelry Management System (JMS)** successfully streamlines the operations of a jewelry business through a structured, automated Salesforce application. It improves efficiency, communication, and data accuracy for both staff and customers.

**APPENDIX**

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

**UpdatePaidAmountTrigger.apxt:**

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

if (Trigger.isInsert) {

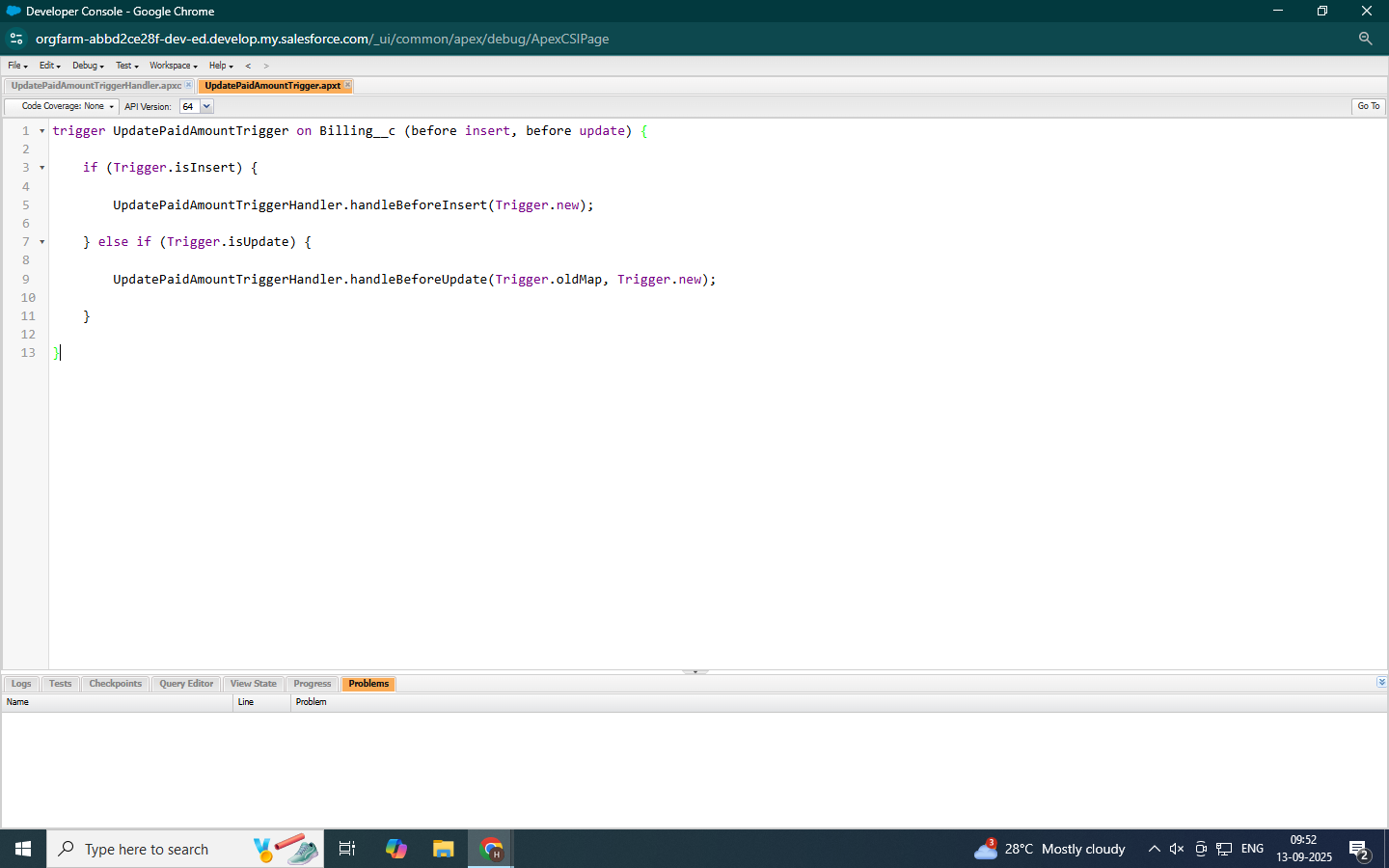
UpdatePaidAmountTriggerHandler.handleBeforeInsert(Trigger.new);

} else if (Trigger.isUpdate) {

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

}

}



**testHandler.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;

}

}

}

