

CRM Application for Jewel Management -

(Developer)

College Name: Sri Ramakrishna College of Arts and Science for Women

College Code: br28

Team ID: NM2025TMID24149

Team Size: 4

Team Leader: AMIRTHA B

Email: srcw2326j105@srcw.ac.in

Team member: AKSHATHA B

Email: srcw2326j103@srcw.ac.in

Team member: AKSHITHA B

Email: srcw2326j104@srcw.ac.in

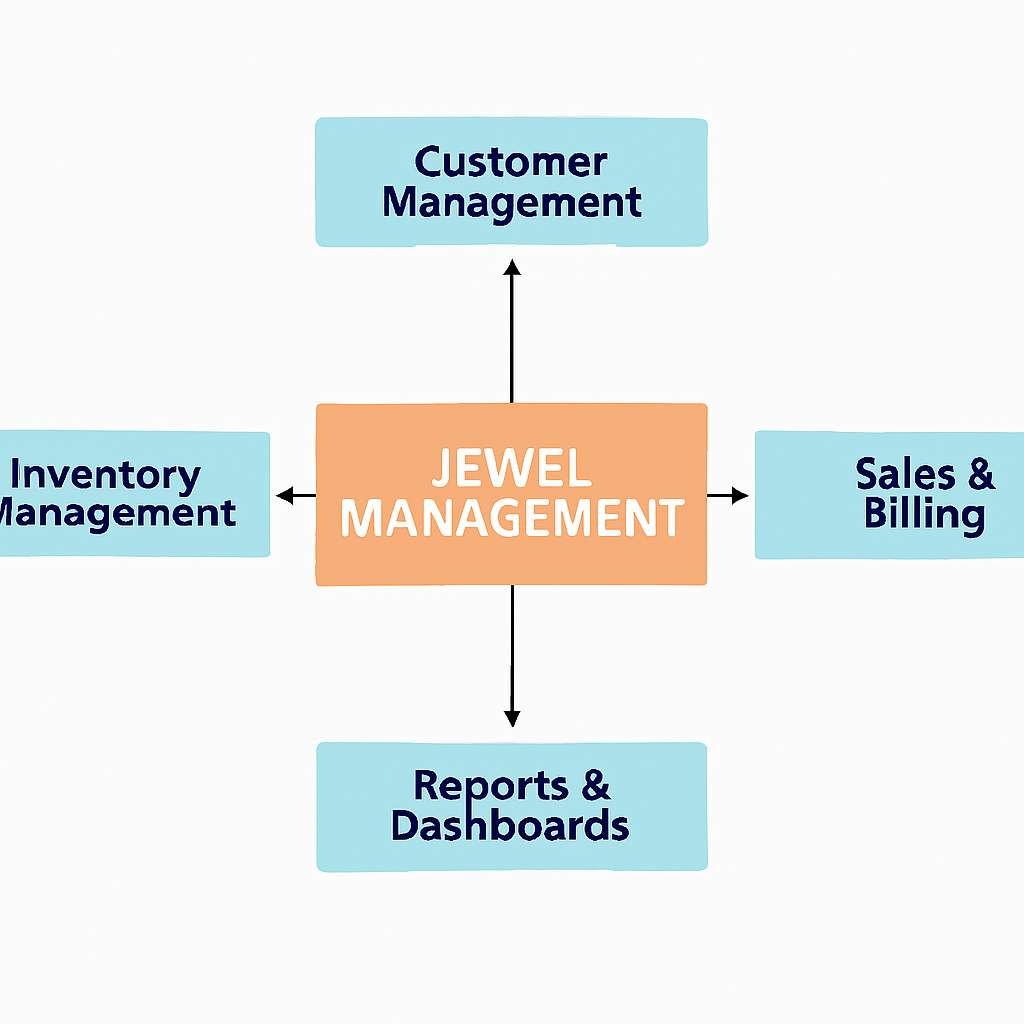
Team member: DHARSHINI V

Email: srcw2326j113@srcw.ac.in

# 1.INTRODUCTION

1.1 Project Overview

The Jewel Management CRM Application is a Salesforce-based solution developed to manage jewelry sales, inventory, customer interactions, and employee activities within a single unified system. This application enables jewelry



businesses to efficiently track stock levels, manage customer purchases, handle billing, and automate communications. Leveraging Salesforce features such as custom objects, validation rules, workflows, approval processes, and dashboards, the system improves operational efficiency,

ensures accurate records, and enhances customer satisfaction.

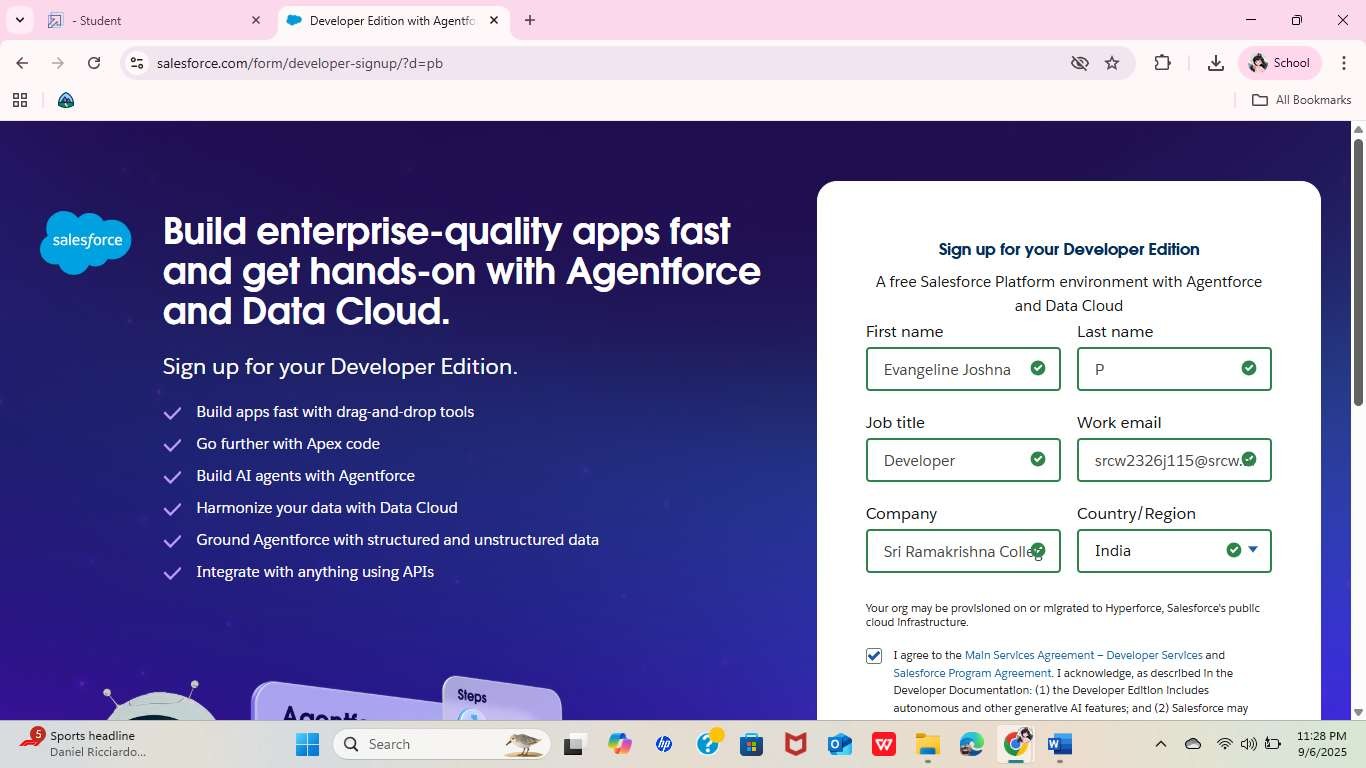
1.2 Purpose of the Project

The purpose of the Jewel Management CRM Application is to streamline jewelry business operations by managing inventory, sales, and customer relationships in a centralized Salesforce platform. It aims to reduce manual errors, improve customer satisfaction, and provide real-time insights for better decision-making.

1. DEVELOPMENT PHASE

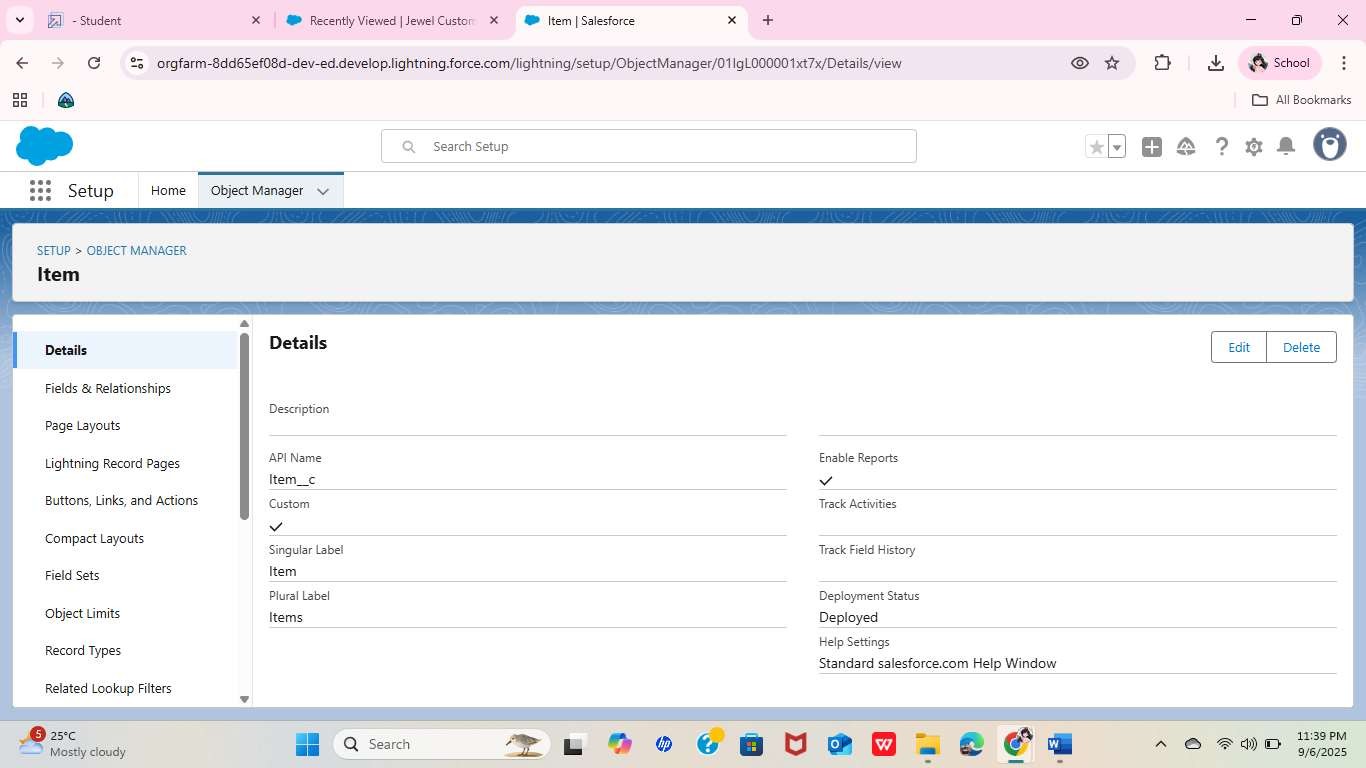
2.1 Creating Developer Account

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

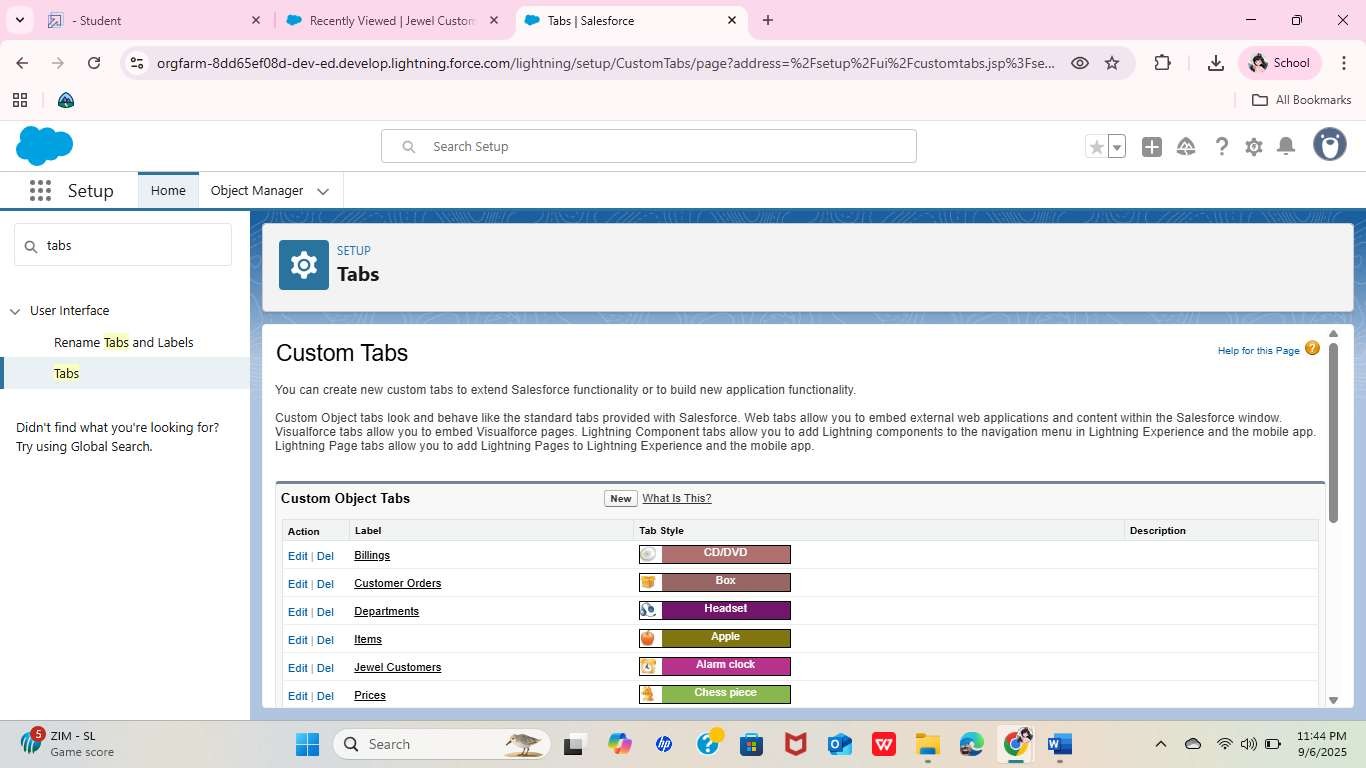


* + Created objects: Jewel Customer and Item

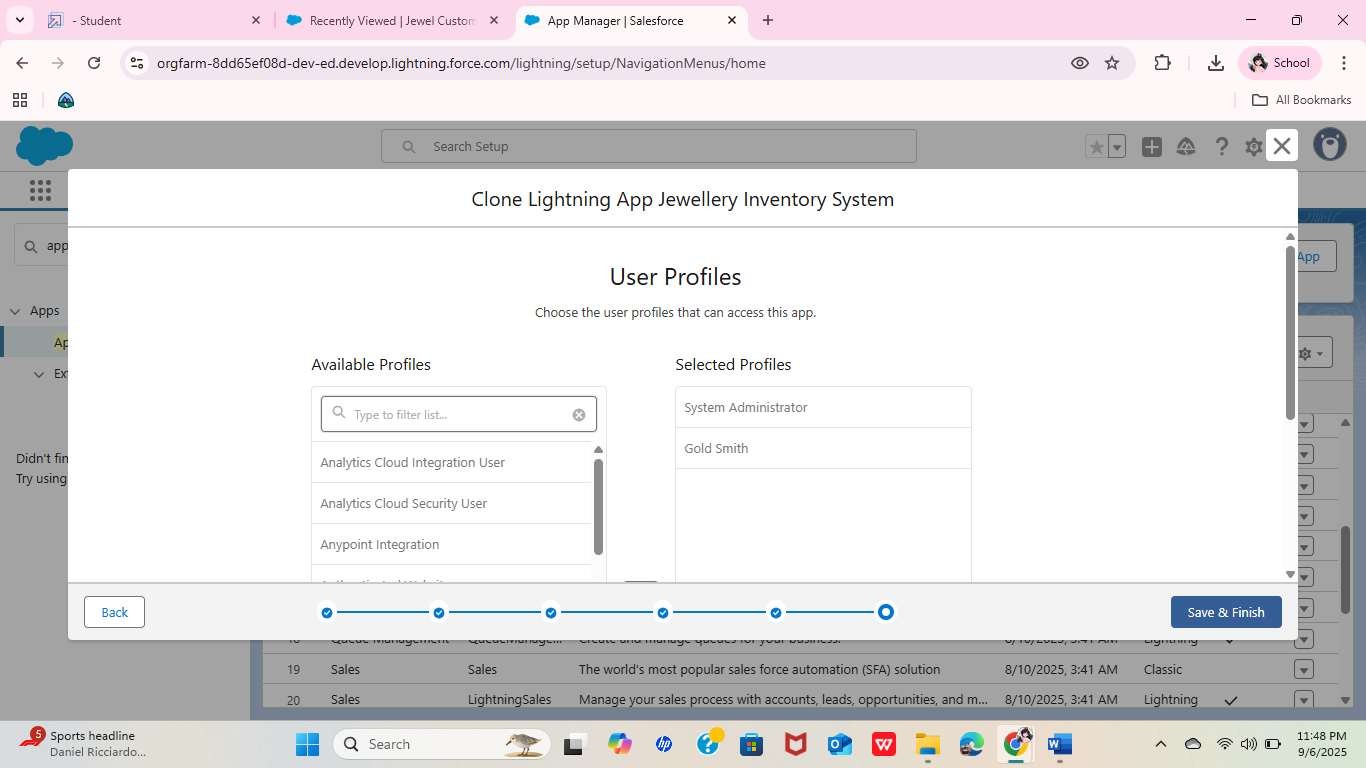
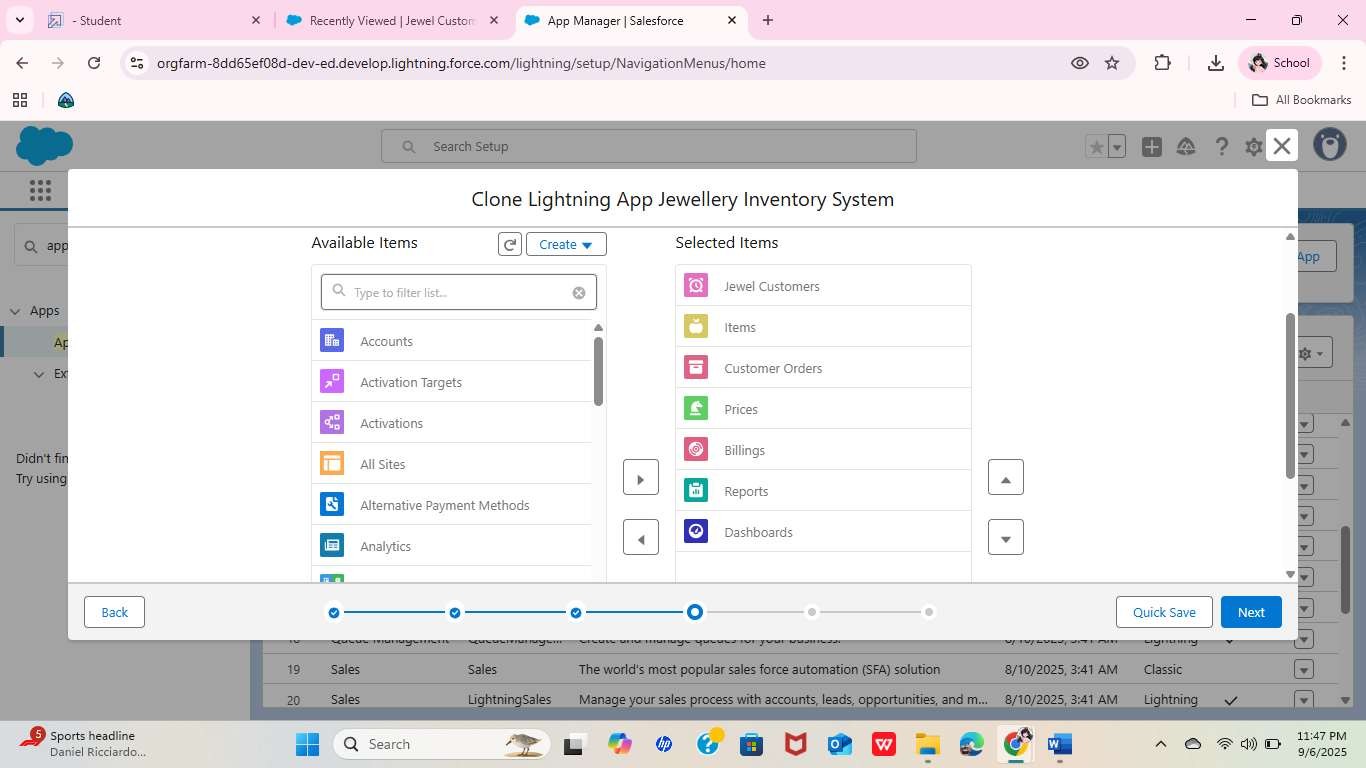
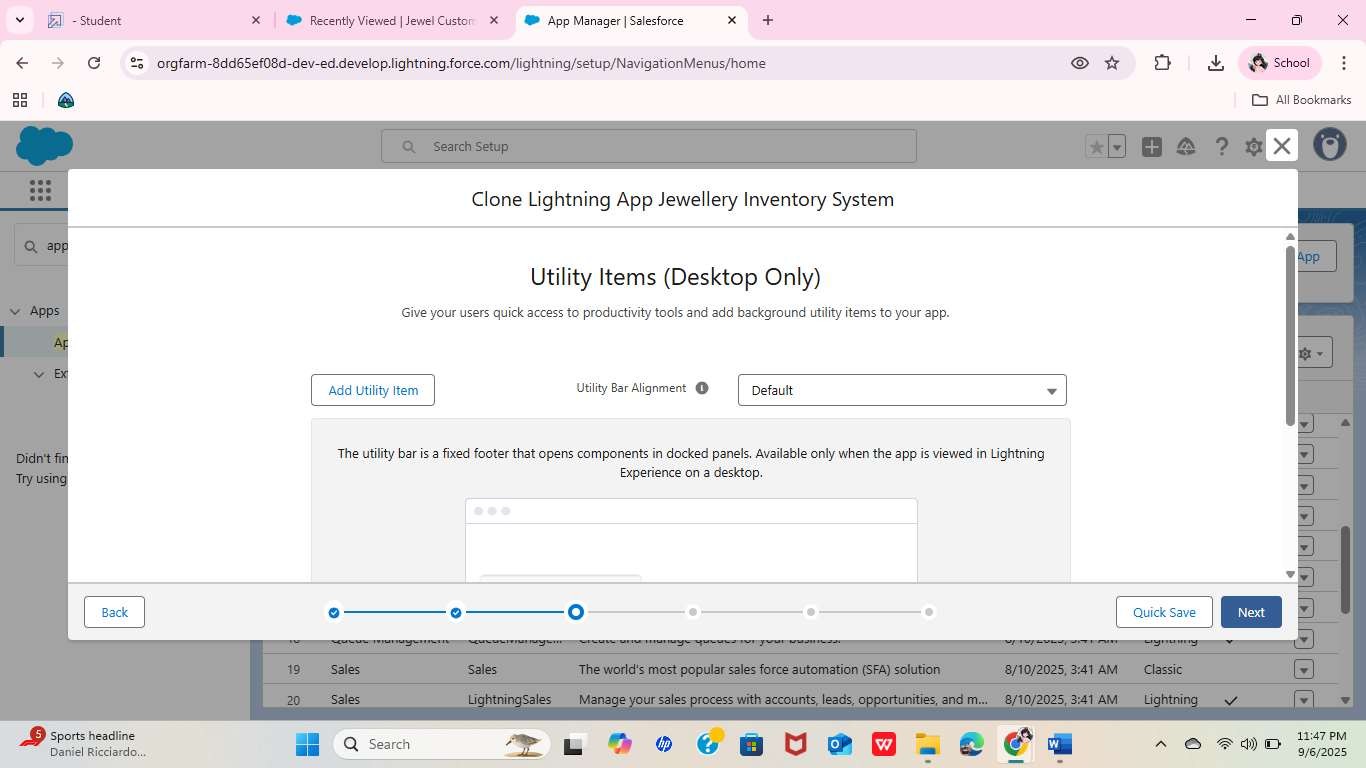
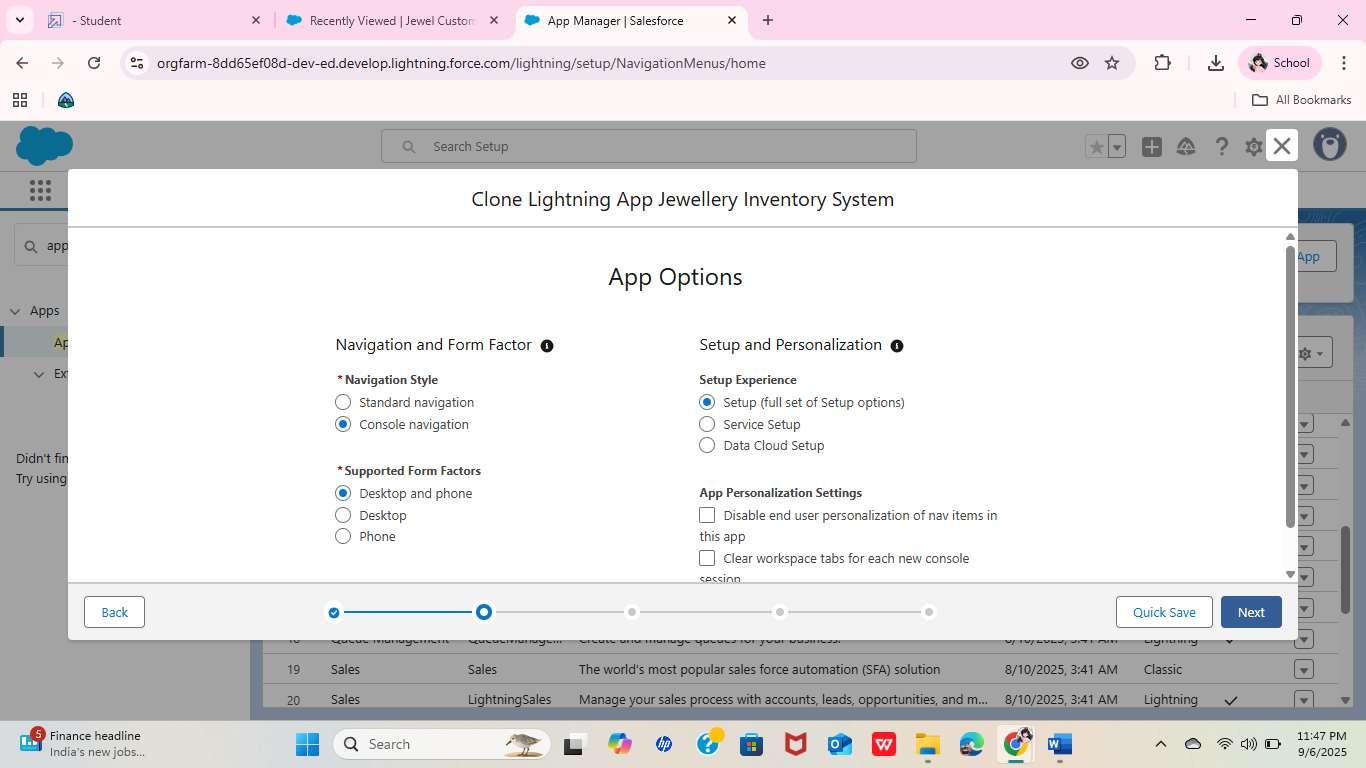




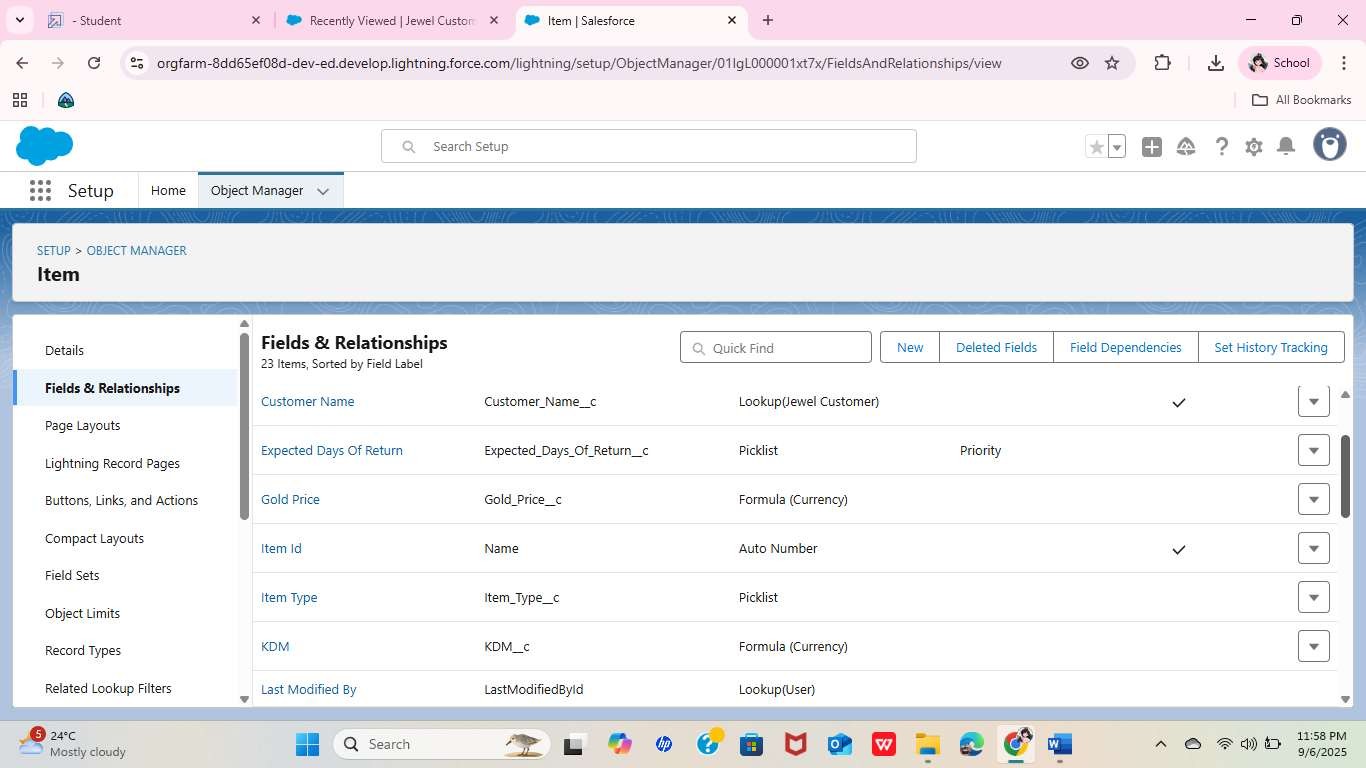
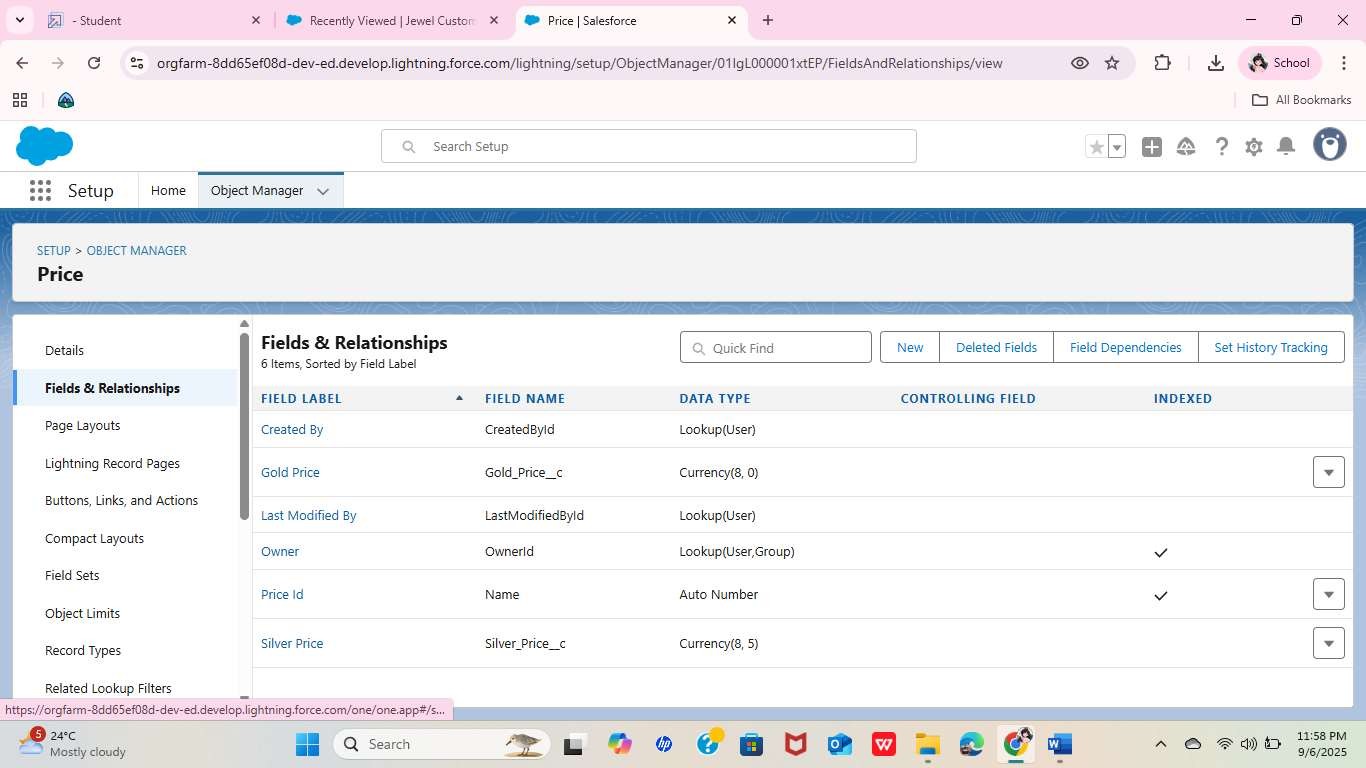
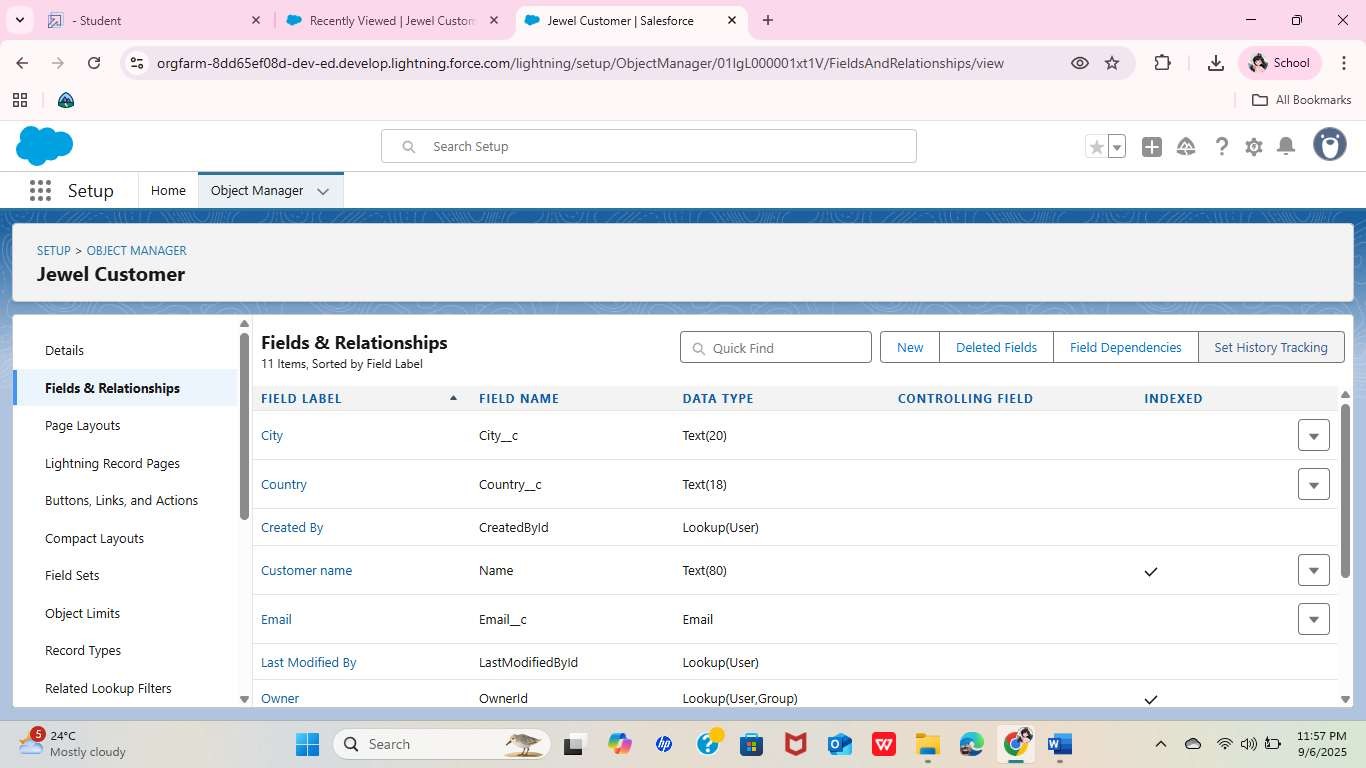
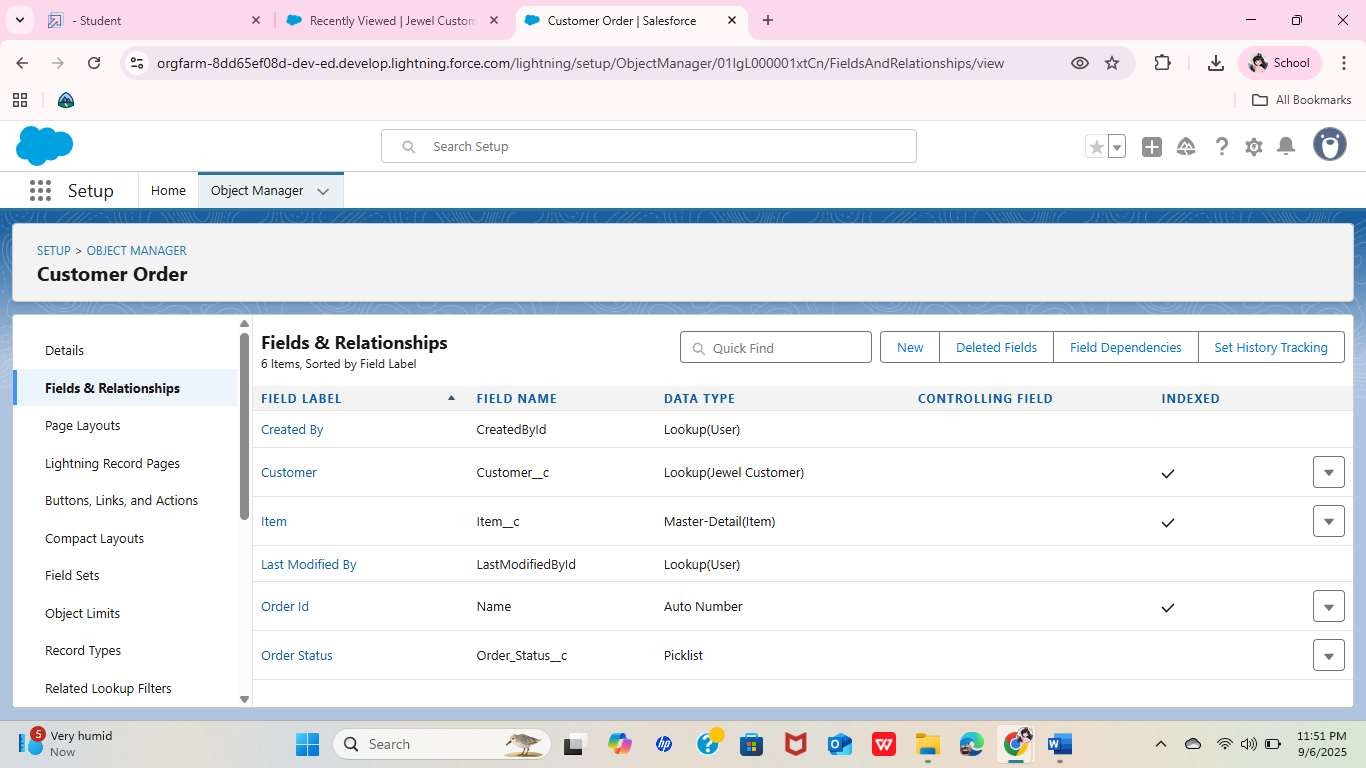
* + Creating a Custom Tab and Item Tab

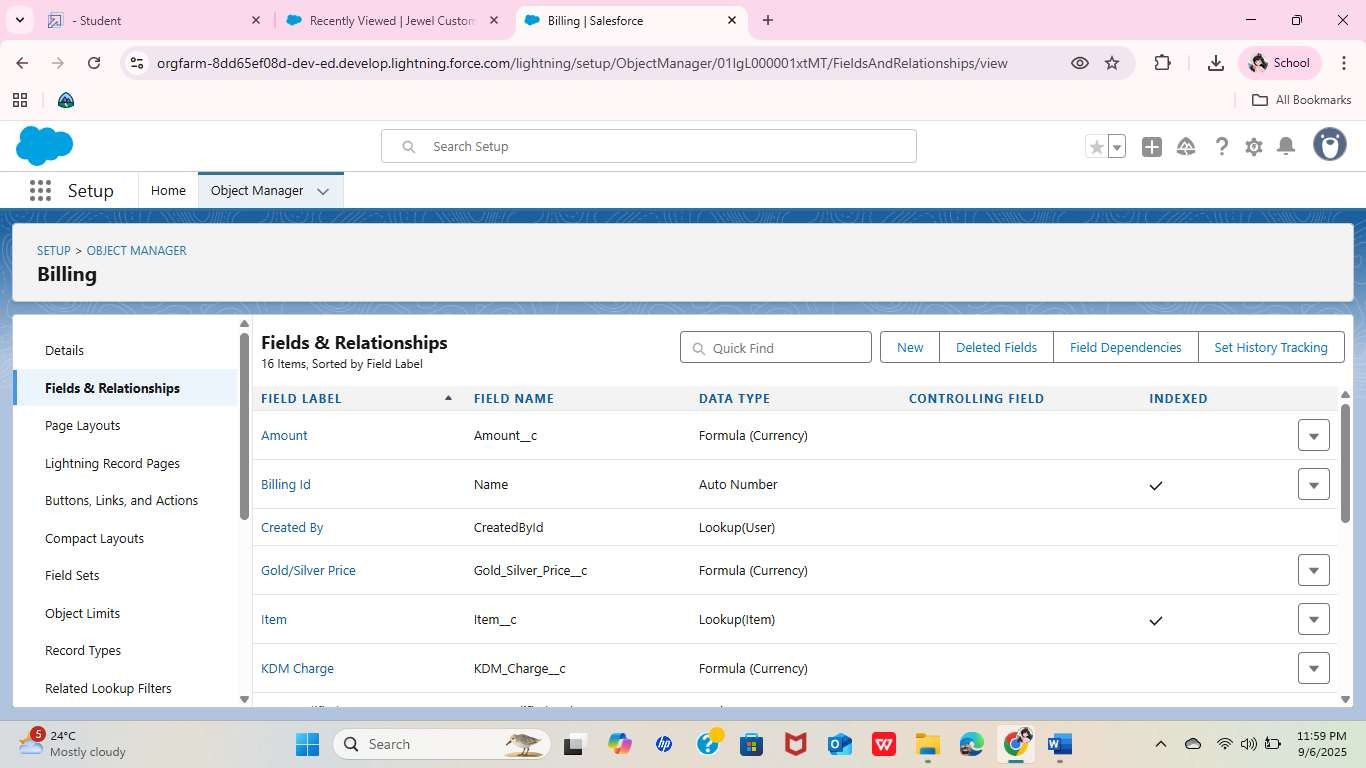


* + Create a Lightning App

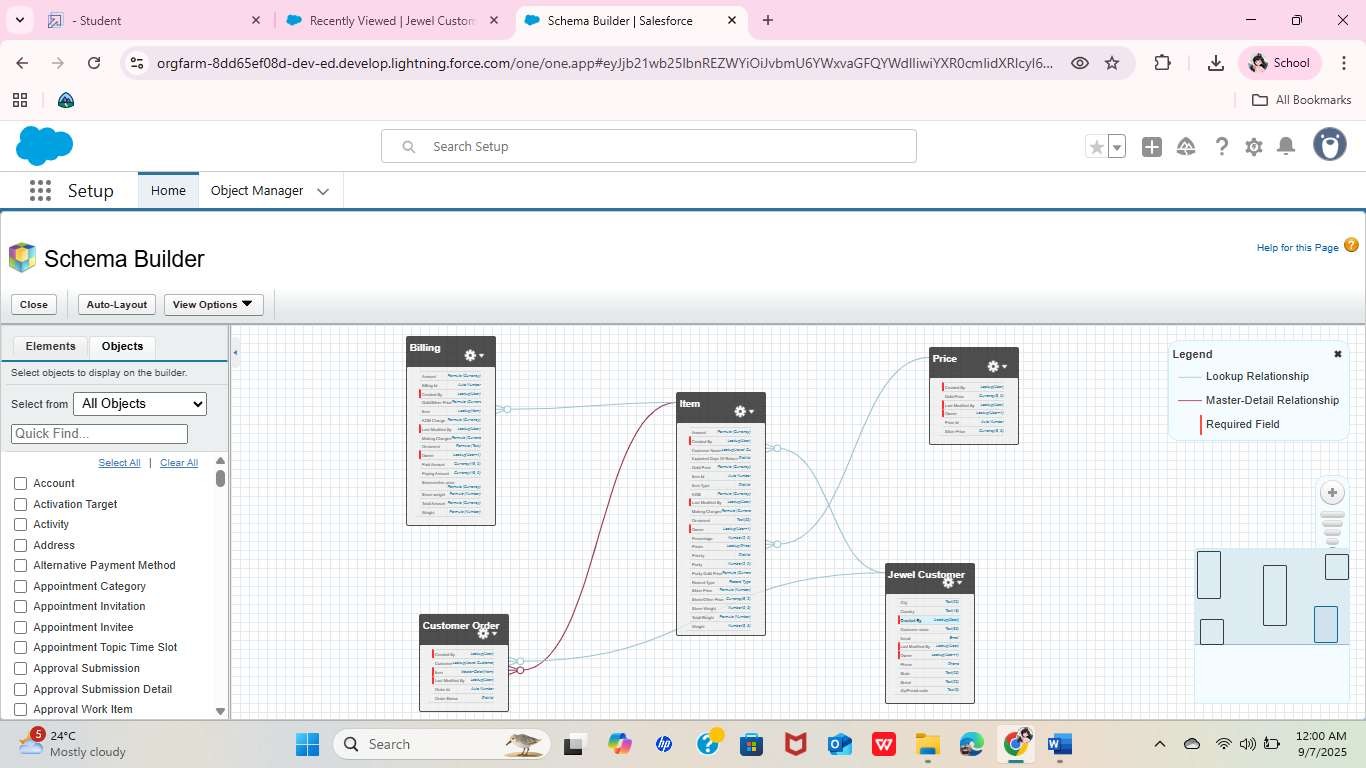


* + Creating Fields : Customer, Item, City, Phone, Email, Purity, Item Type, Gold Price, Gold Price (Cross Object), etc.

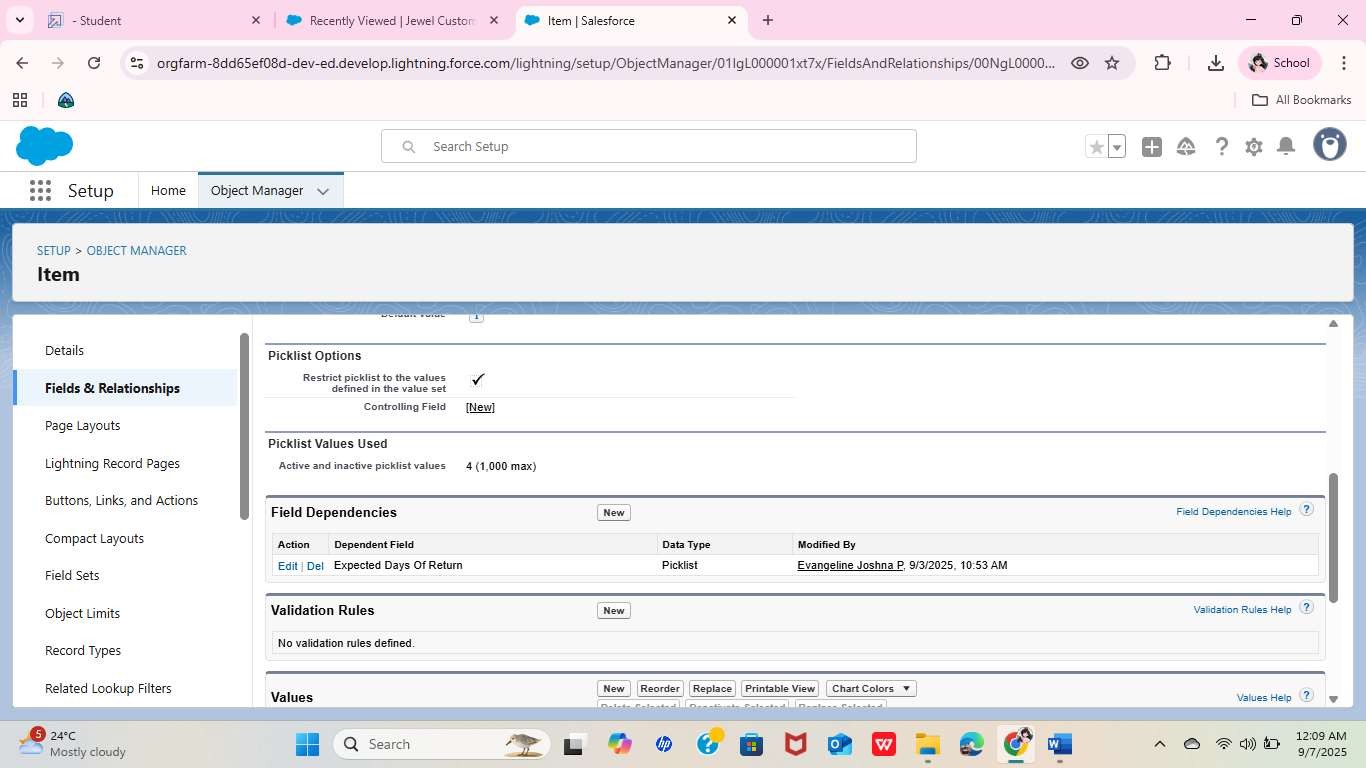




* + Schema Builder



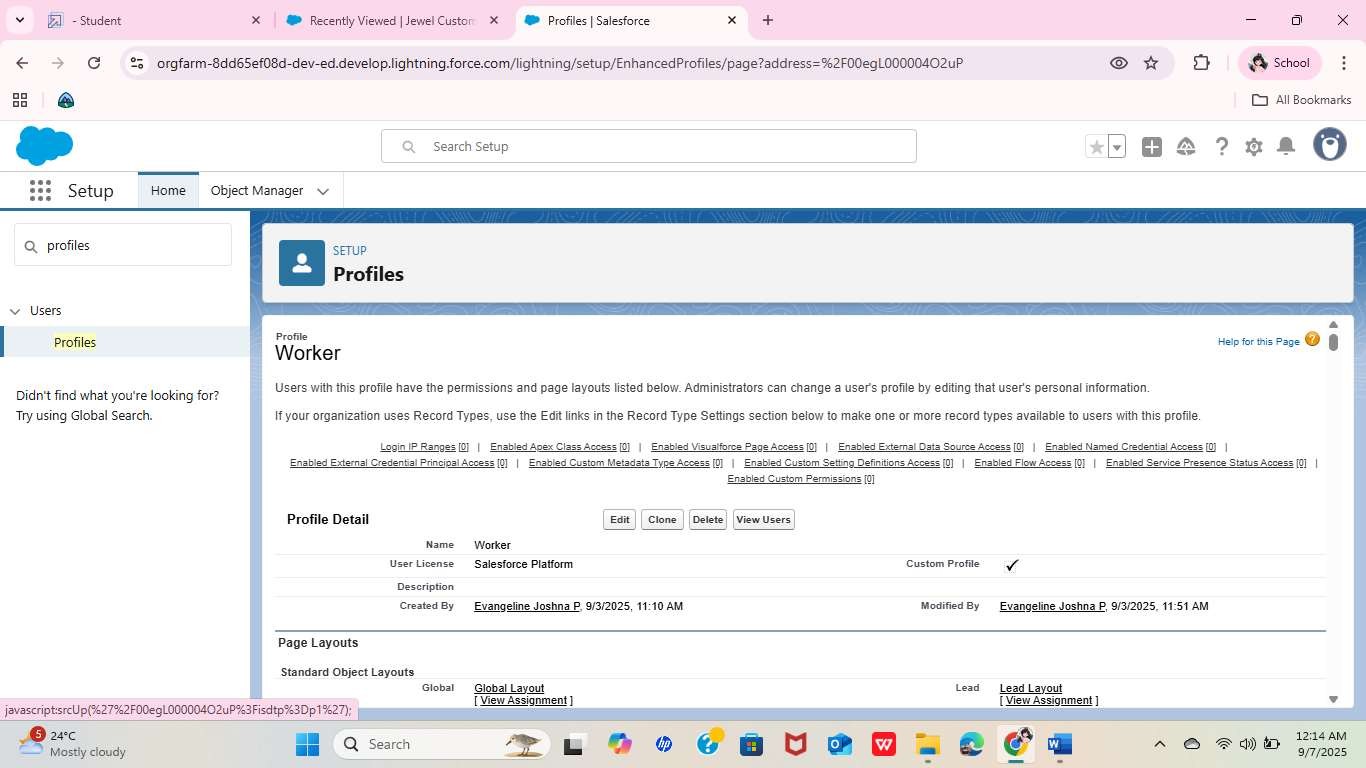
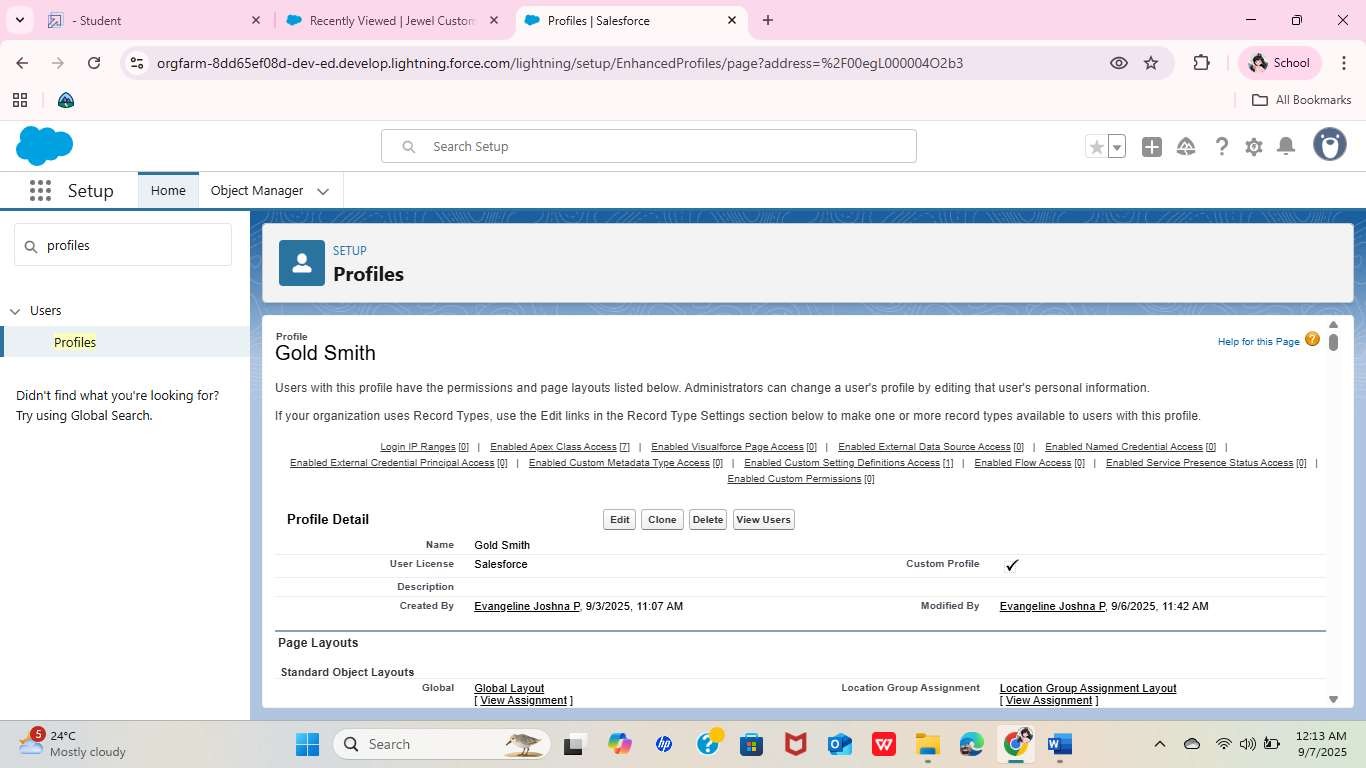
Creating the Field Dependencies



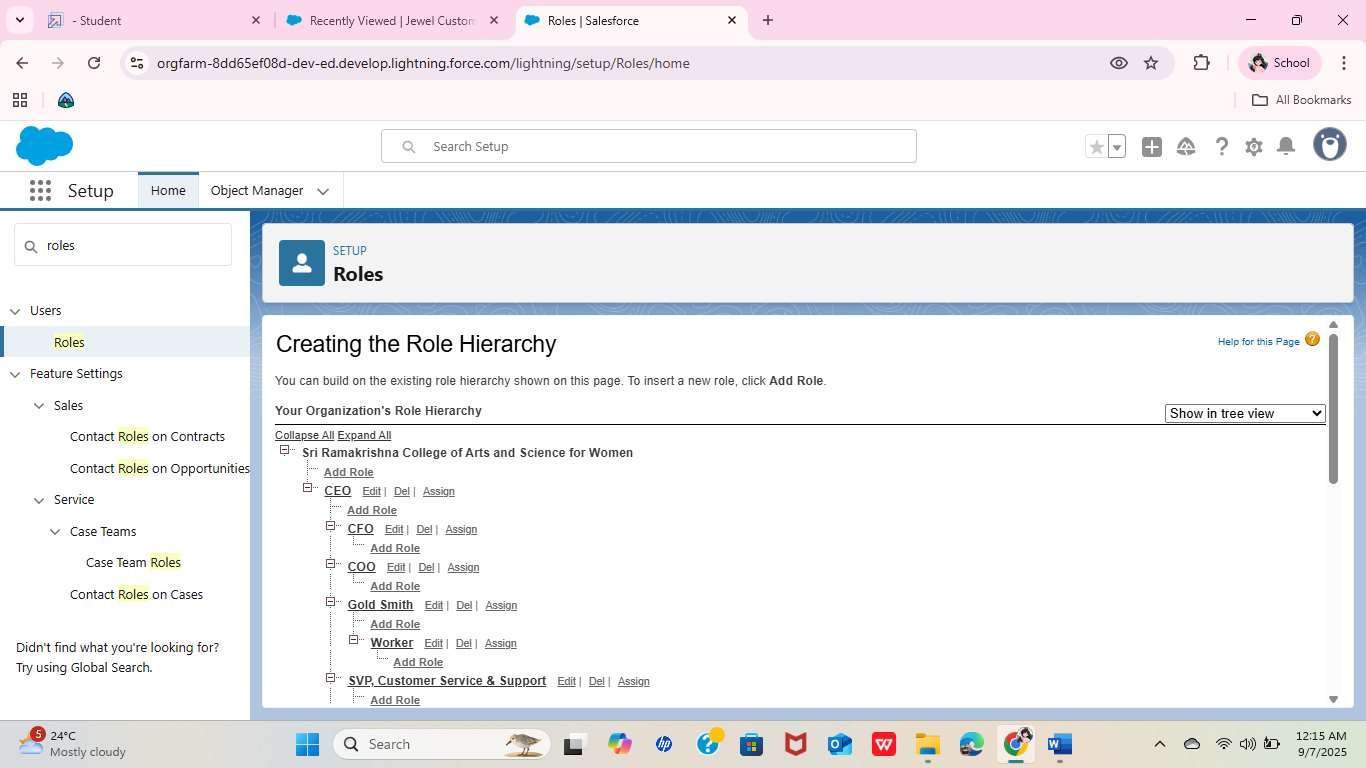
* + Creating the validation rule



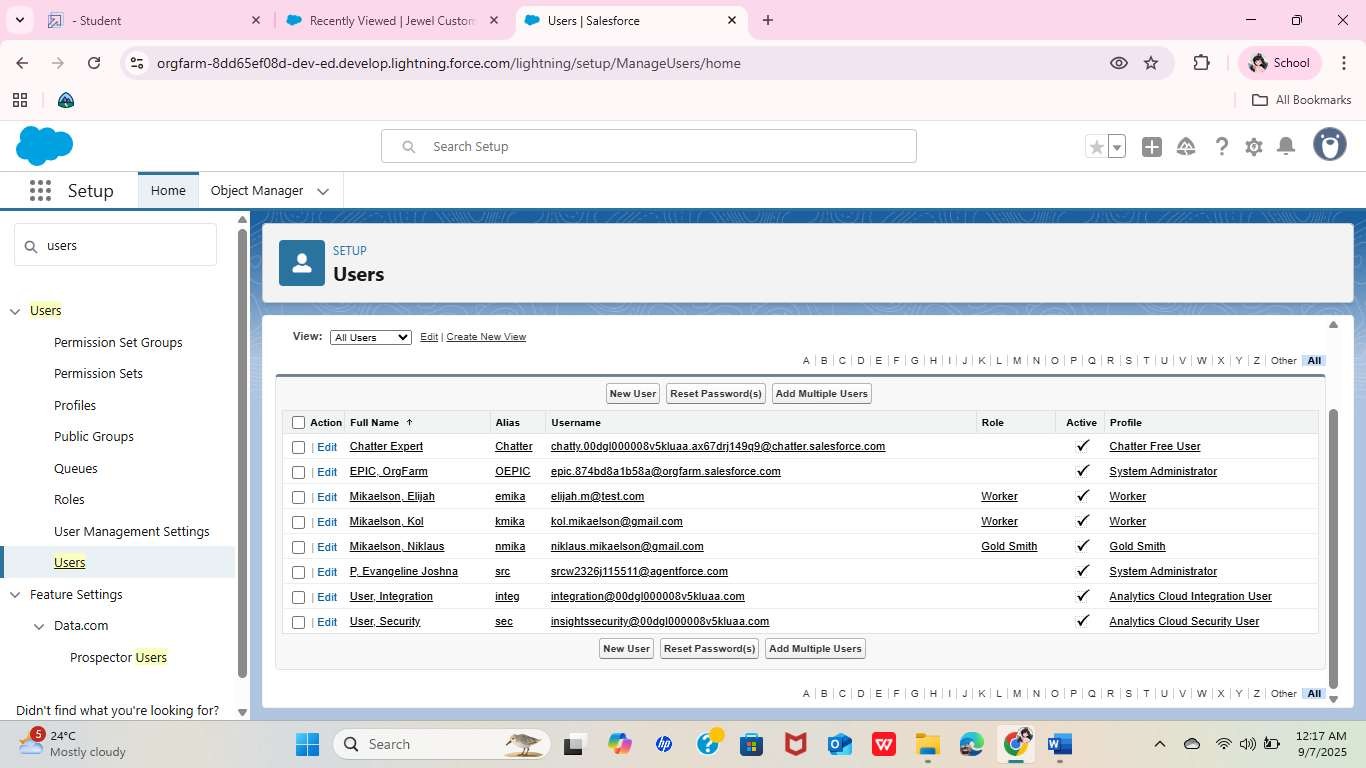
Creating Profiles : Gold Smith & Worker



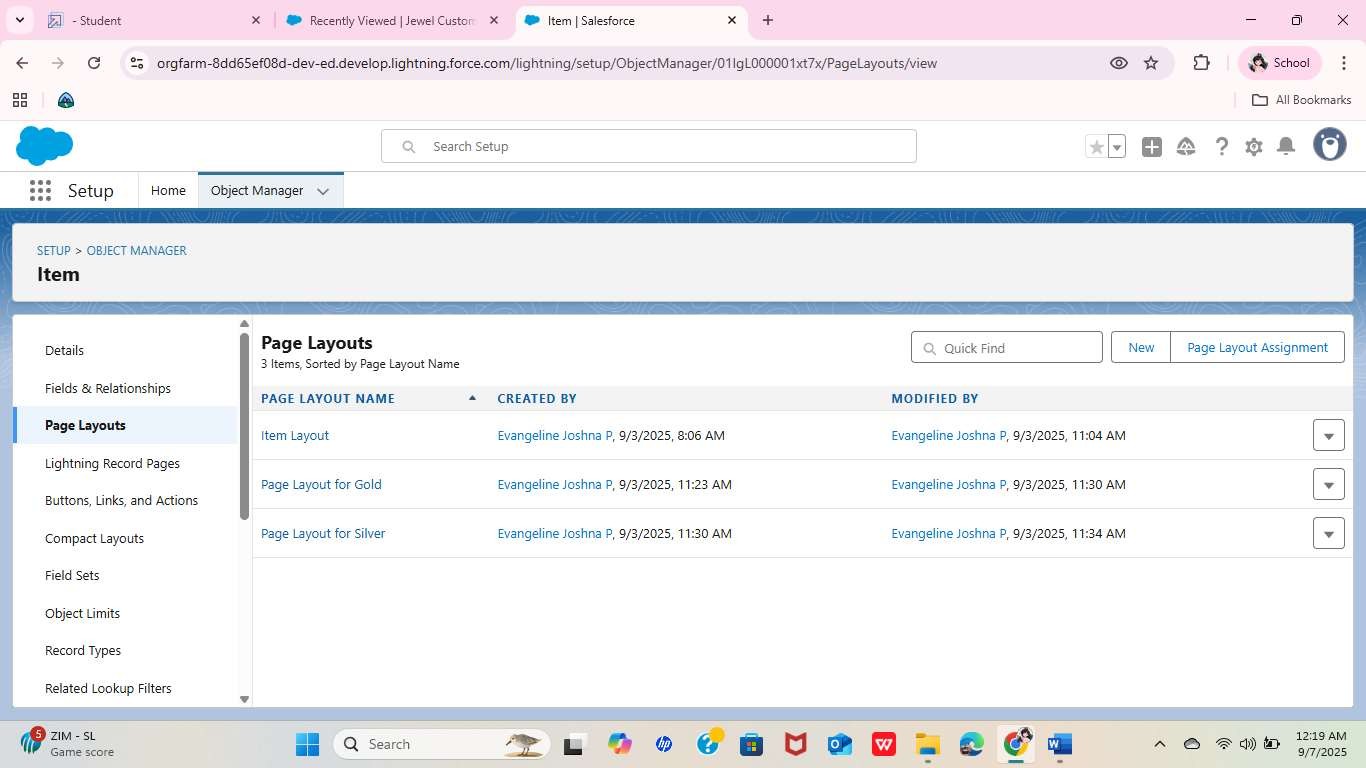
Creating Gold Smith Role and one more role as Worker which reports to Gold Smith.



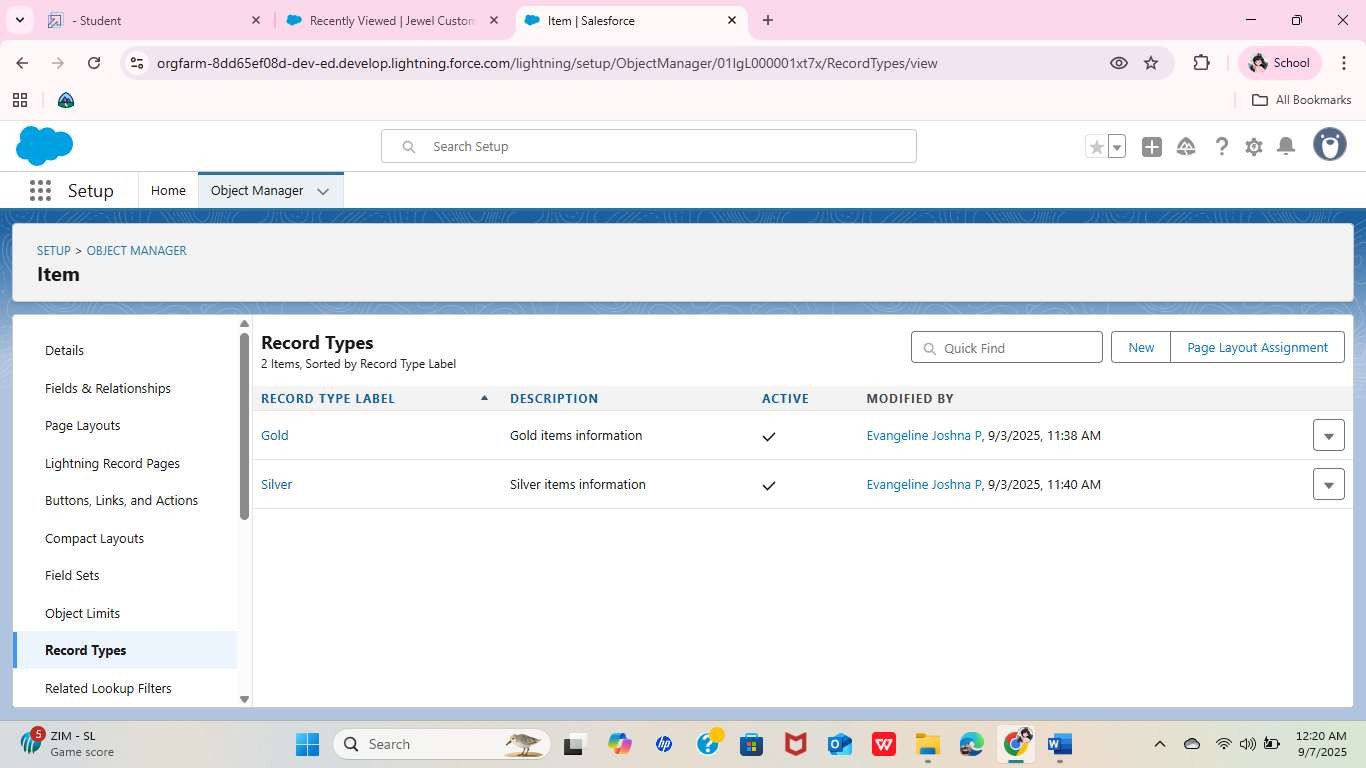
* + Create Users



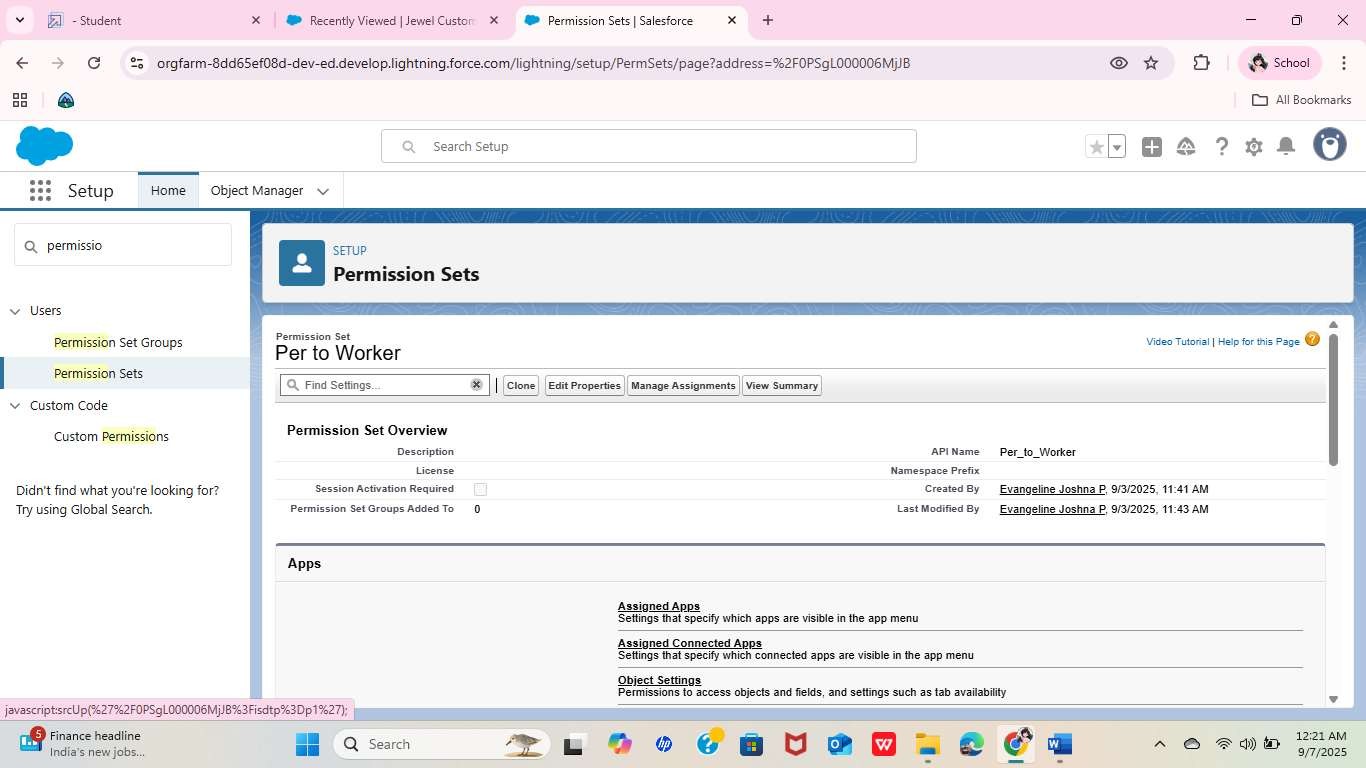
Create a Gold Page layout & Silver Page layout



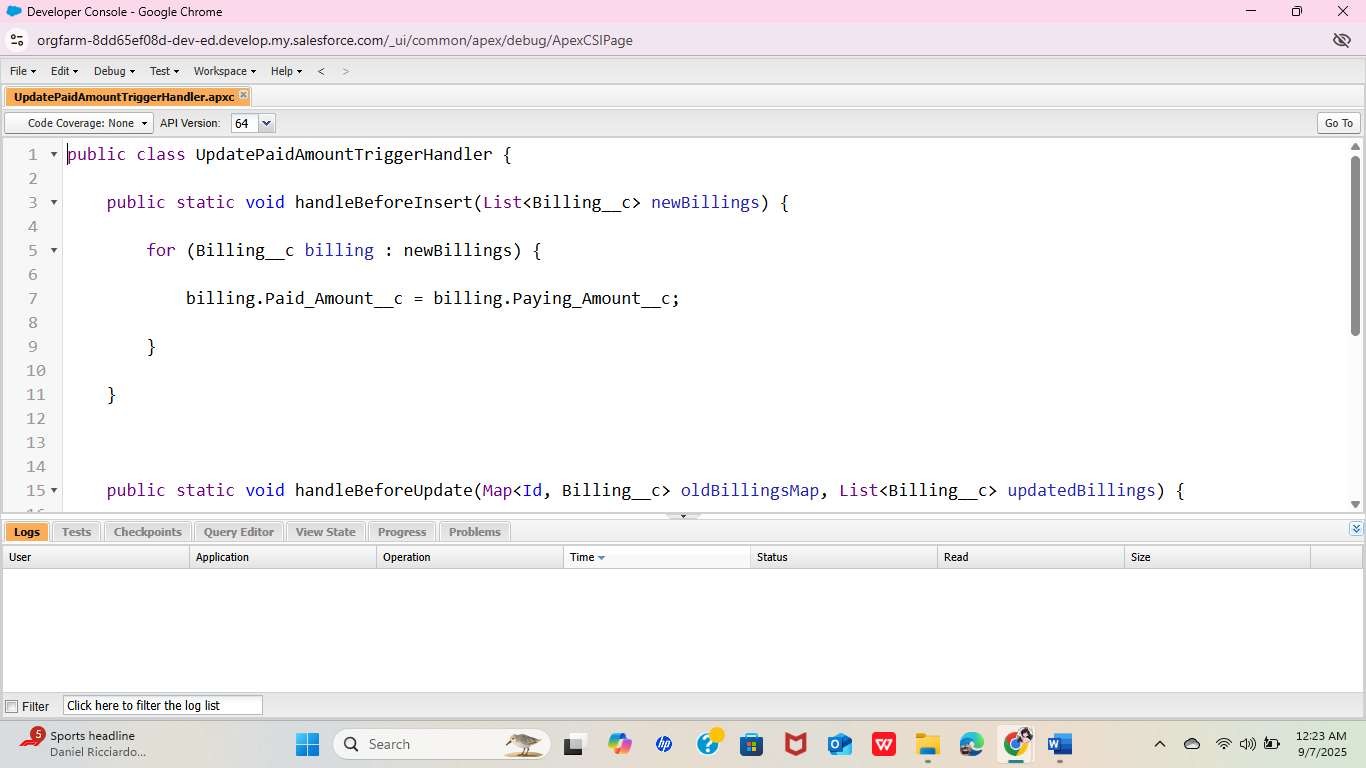
* + Create a Record Type

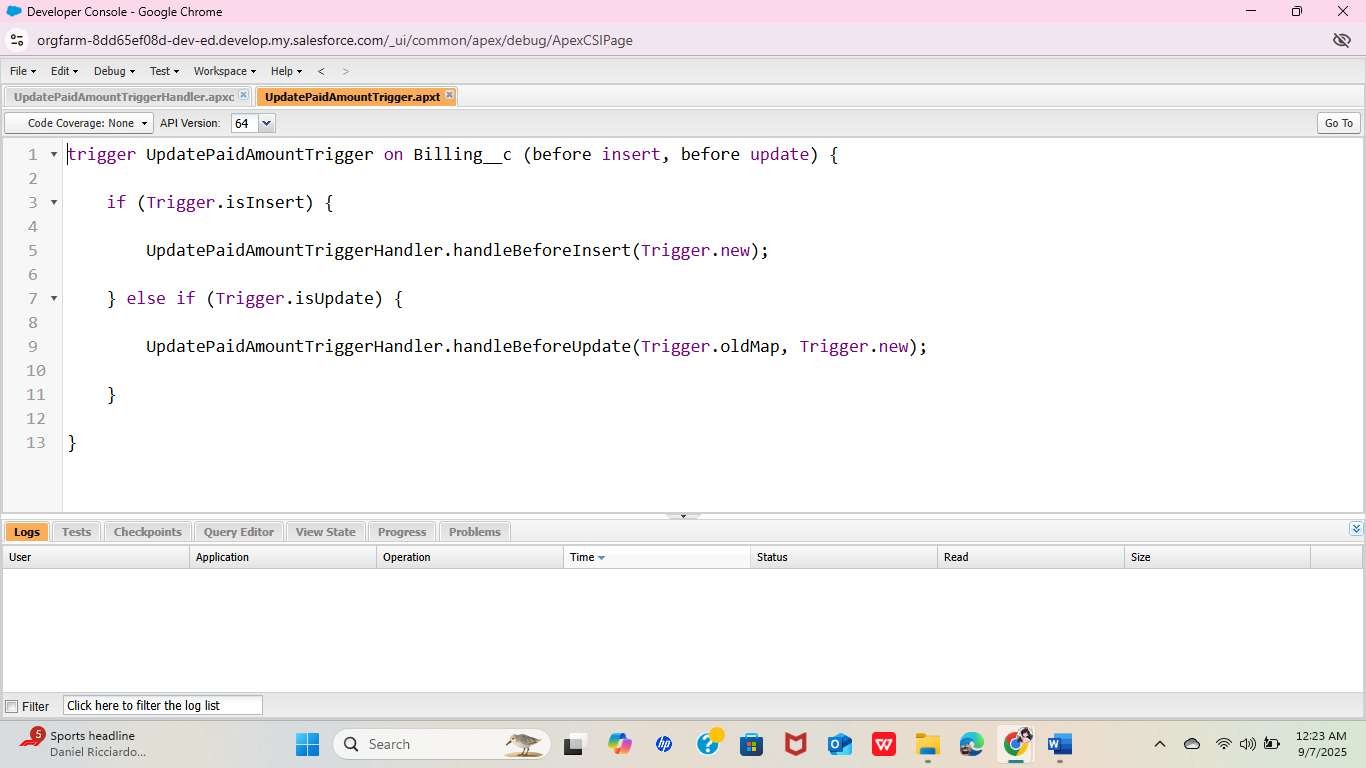


Creating permission set

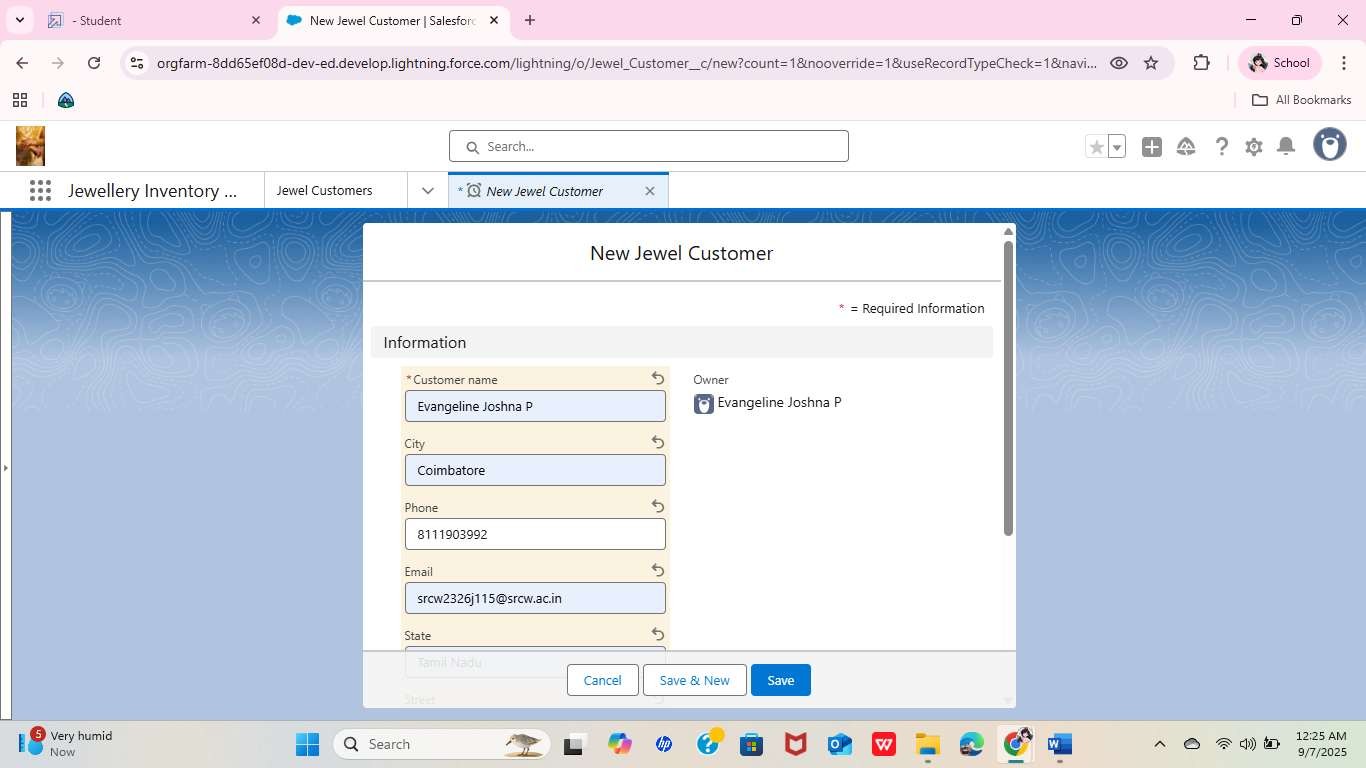


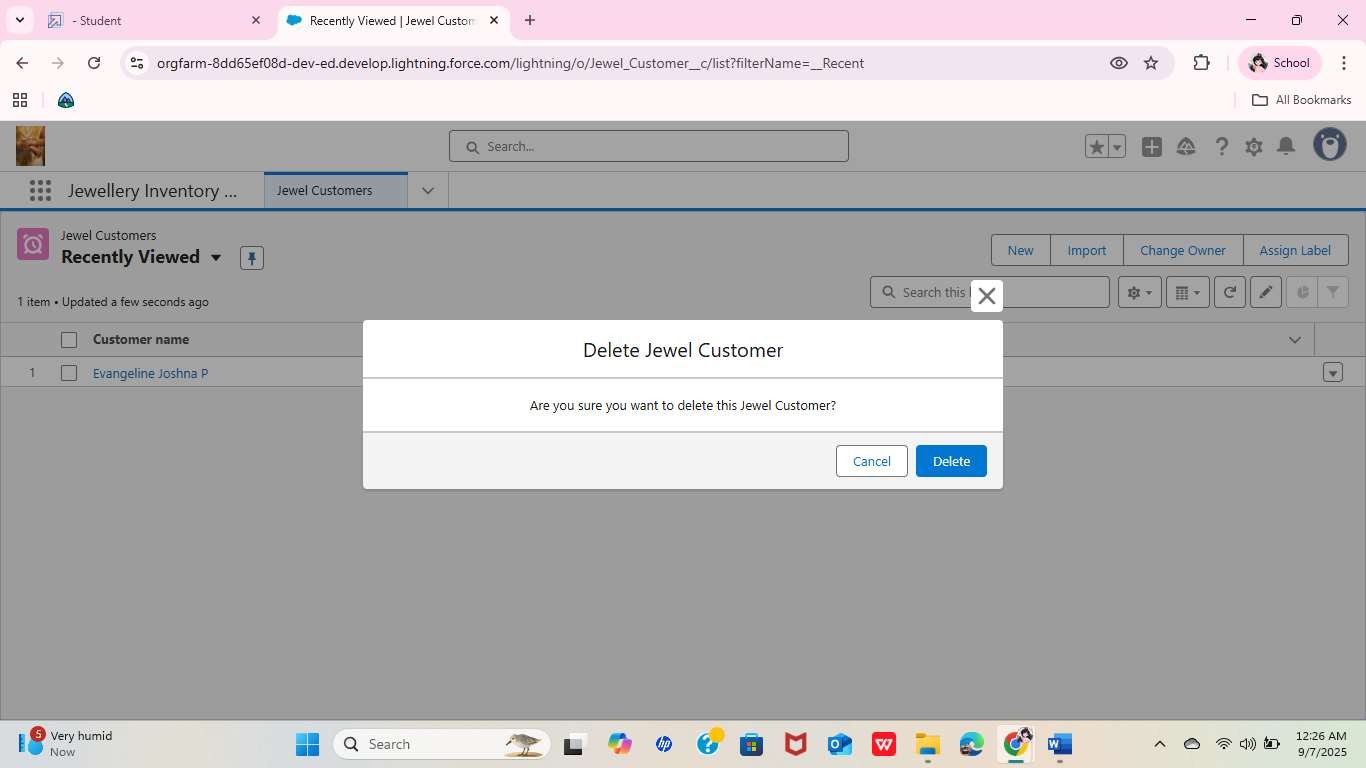
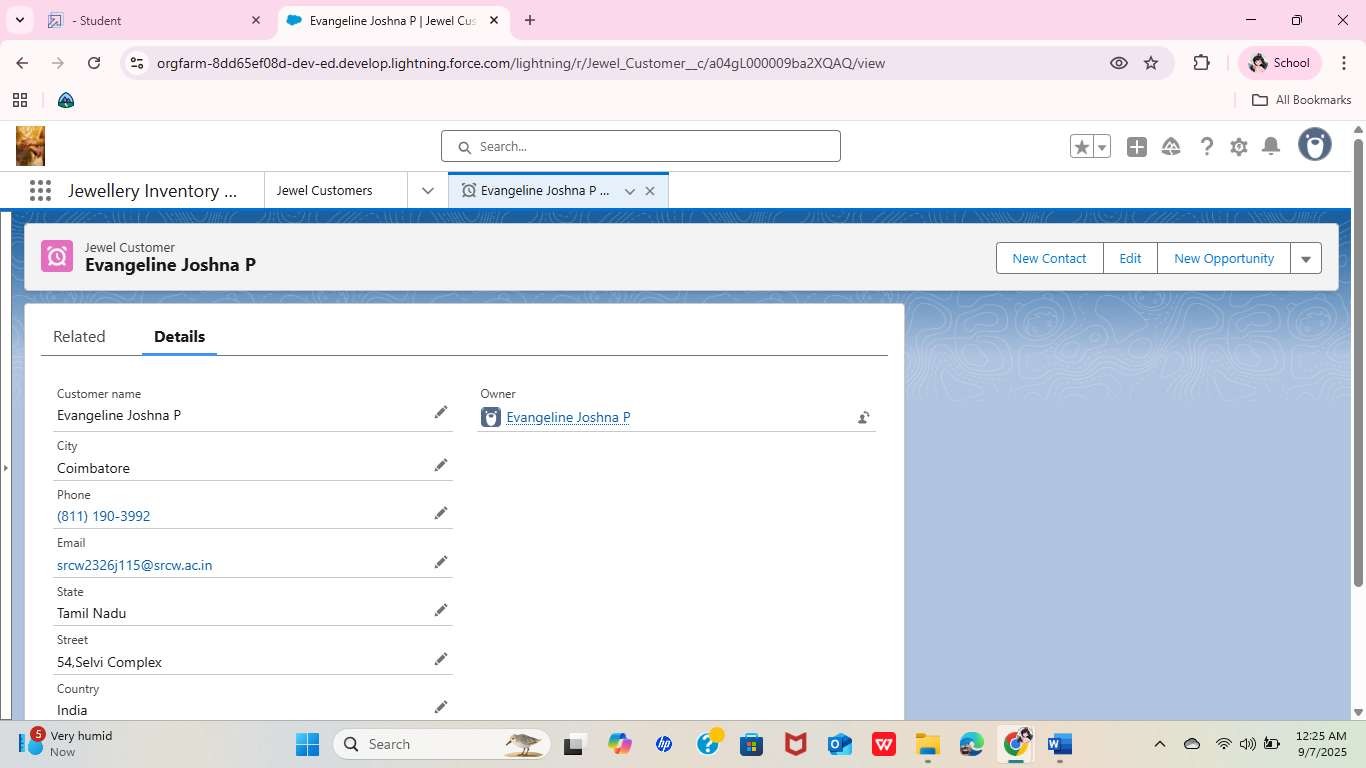
* + Create a Trigger



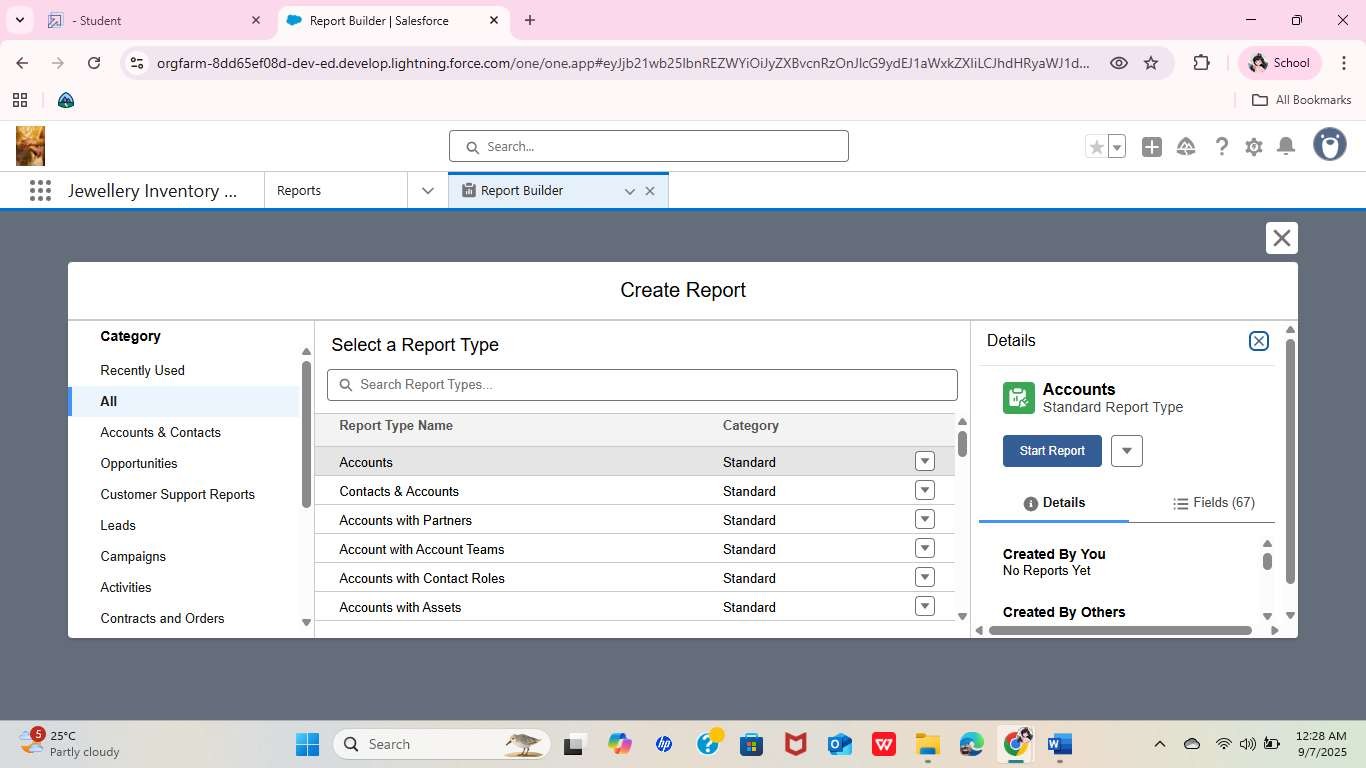
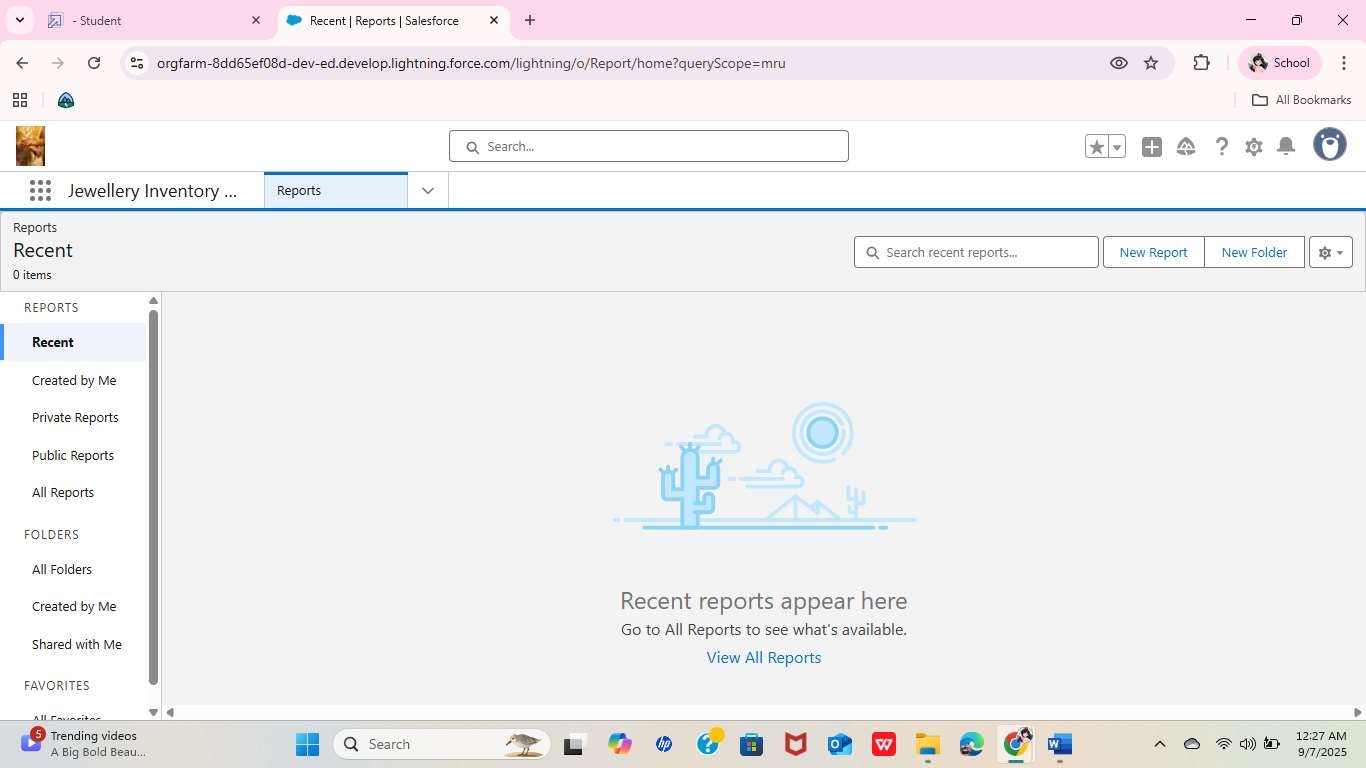


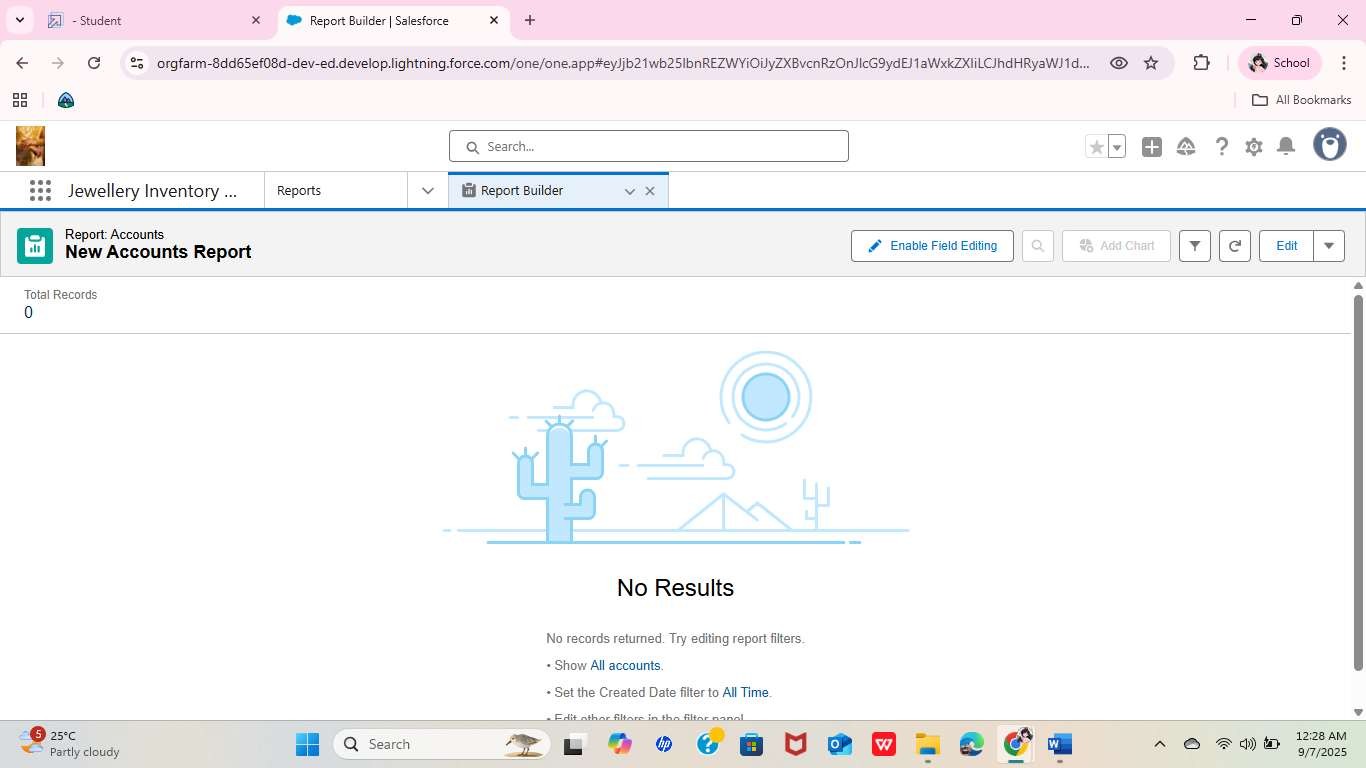
* + Create, View and Delete a Record



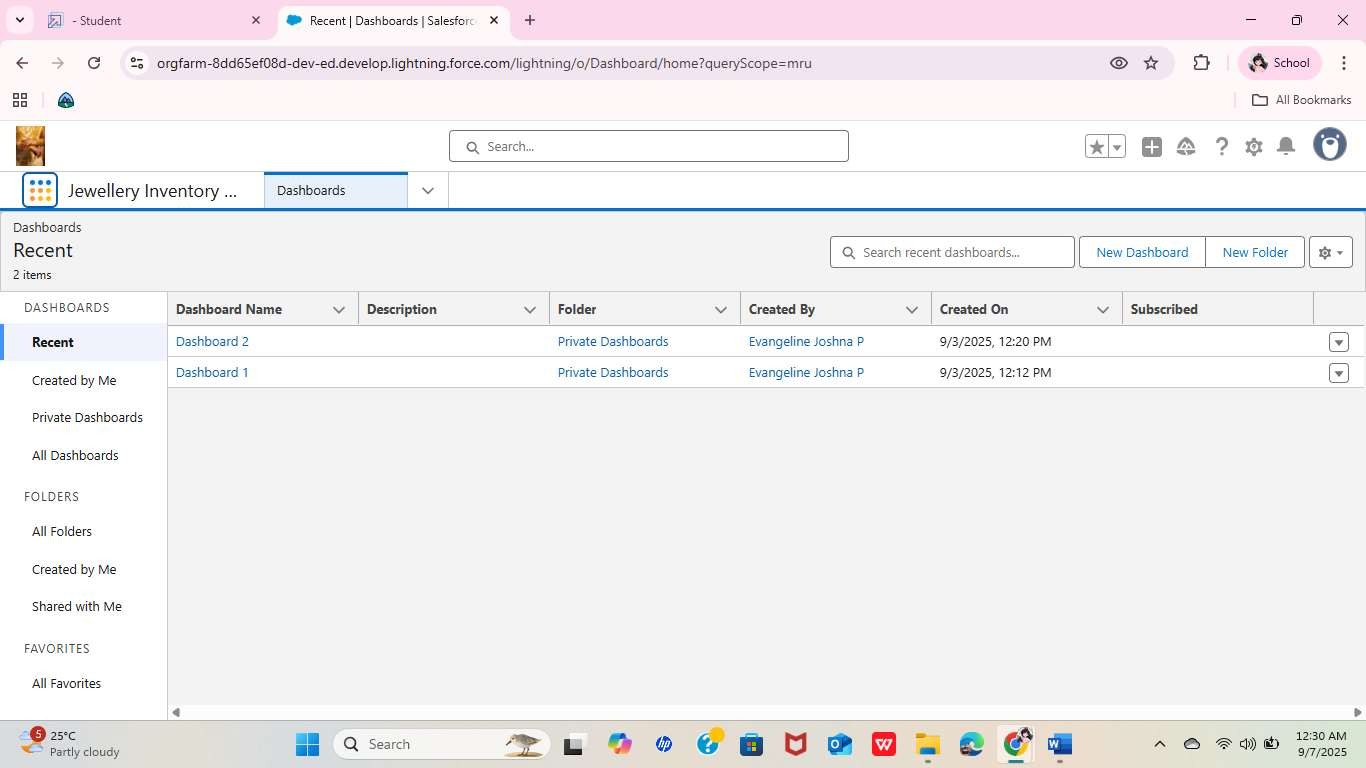


* + Reports

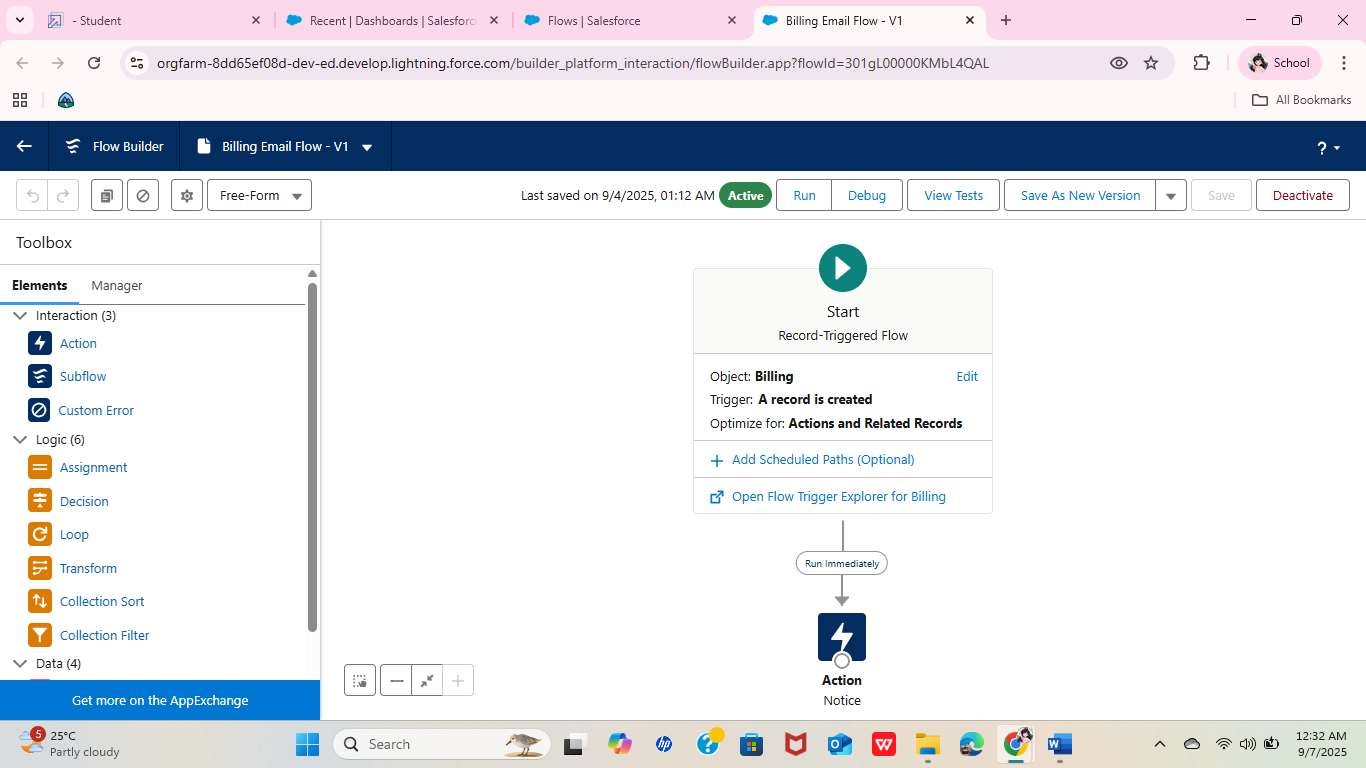




* + Create Dashboard



* + Create a Flow



# 3.ADVANTAGES AND DISADVANTAGES

Advantages

1. Centralized Data Management – All customer, sales, and inventory data is stored in one place.
2. Automation – Reduces manual work through workflows, approval processes, and email alerts.
3. Improved Customer Service – Tracks customer preferences, purchase history, and communication.
4. Real-Time Reports – Dashboards and analytics give insights into sales performance and stock levels.
5. Scalability – Can grow with the business by adding new features like AI suggestions or chatbots.
6. Error Reduction – Automation minimizes human errors in billing, stock tracking, and reporting.
7. Accessibility – Cloud-based Salesforce CRM allows access anytime, anywhere.

Disadvantages

1. High Initial Setup Cost – Salesforce licensing and customization can be expensive.
2. Complexity – Requires training for staff to use effectively.
3. Customization Dependency – Advanced customization needs a developer.
4. Internet Dependence – As a cloud system, it needs a stable internet connection.
5. Maintenance – Regular updates and monitoring are needed for smooth operation.
6. Data Security Concerns – Though Salesforce is secure, handling sensitive customer data always carries risks.

1. FUTURE ENHANCEMENTS

Briefly describe what improvements or new features could be added in the future to make the CRM system even better.

Example Content:

* + Integration of AI-based Recommendations to suggest jewelry based on customer preferences.
  + Implement a Chatbot for Customer Support to answer common queries instantly.
  + Develop a Mobile Application for sales staff to manage orders and inventory on the go.
  + Enable Multi-Currency Support for international customers.
  + Add Advanced Security Features like 2FA (Two-

Factor Authentication) for better data protection.

1. TESTING APPROACH

To ensure that the Jewel Management CRM Application works as expected, the following testing methods were applied:

1. Unit Testing

* + - Tested individual components such as

custom objects, validation rules, and triggers using sample data in the Salesforce Developer Console.

* + - Verified that triggers properly prevented duplicate entries and updated fields as intended.

1. Flow Testing

oCreated sample records to simulate gatepass requests and observed the correct execution of automated flows. oVerified that email alerts and field updates

triggered as expected.

1. Approval Process Testing

oSimulated customer purchase approval flows to ensure records followed the correct approval path. oVerified that approval or rejection

notifications were sent correctly.

1. Report & Dashboard Validation

oCreated test transactions to confirm that the dashboard and reports displayed accurate, up-to-date data.

1. User Access Testing

oEnsured that profiles and permission sets correctly restricted or allowed access based on user roles (Gold Smith, Worker).

1. LIMITATIONS

List any known limitations of your current system.

Example Content:

* + Does not support offline functionality.
  + Limited customization without developer intervention.
  + Dependent on stable internet connection.

1. CONCLUSION

The Jewel Management CRM Application built on Salesforce provides a powerful solution to manage jewelry business operations efficiently. By automating customer management, inventory tracking, sales billing, and reporting, the system eliminates the inefficiencies of traditional manual processes.

The project successfully demonstrates how Salesforce’s features such as custom objects, validation rules, approval processes, workflows, and dashboards can be utilized to create a scalable and user-friendly solution. The CRM improves decisionmaking by providing real-time insights, enhances customer satisfaction through timely communication, and reduces manual errors through automation.

Although there are some limitations like internet dependency and the need for further customization, the system lays a strong foundation for future growth. Planned enhancements such as AI-powered recommendations and mobile app support will further strengthen the system’s capabilities.

Overall, this project highlights the practical application of Salesforce in solving real-world business problems, and provides significant value to jewelry businesses aiming for efficiency, accuracy, and better customer service.

1. APPENDIX – Trigger Handler Class and

Trigger Implementation

8.1 Trigger Handler Class

A trigger handler is a design pattern that organizes trigger logic into separate classes. It helps keep the code organized, reusable, and easier to maintain. The handler class contains specific methods to handle logic for different trigger events, improving readability and reducing code duplication.

Purpose:

* Maintain modularity
* Improve structure and readability
* Enable easy reuse of logic in multiple places

Sample Code

UpdatePaidAmountTriggerHandler.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;

}

}

}

8.2 Trigger Implementation

The trigger listens for before insert and before update events on the Billing object and delegates the logic to the handler class methods.

Sample Code

UpdatePaidAmountTrigger.apxt

trigger UpdatePaidAmountTrigger on Billing\_\_c

(before insert, before update) { if (Trigger.isInsert) {

UpdatePaidAmountTriggerHandler.handleBeforeIns ert(Trigger.new);

} else if (Trigger.isUpdate) {

UpdatePaidAmountTriggerHandler.handleBeforeUp date(Trigger.oldMap, Trigger.new);

}

}