Food Delivery Problem Solving Project

Phase 1: Problem Understanding & IndustryAnalysis

Problem Statement

- Food delivery businesses often face challenges like delayed deliveries, order mismatches, lack
  of real-time tracking, and poor customer satisfaction.
- There is a need for a digital solution that can streamline operations, ensure transparency, and
  enhance customer experience.

Project Overview

- The project aims to develop a Food Delivery Management System to address operational inefficiencies.
- It will integrate real-time order tracking, efficient delivery assignment, stakeholder management,
  and smooth business workflows.

Objectives

- To create a robust food delivery platform with accurate order handling.
- To reduce delivery delays through optimized route planning.
- To improve customer satisfaction with real-time order tracking and updates.
- To enable restaurants and delivery partners to manage operations effectively.
- To explore AppExchange solutions for Salesforce-based integrations.

Requirement Gathering

- Understand customer needs (real-time tracking, multiple payment options, quick delivery).
- • Identify restaurant requirements (easy order management, integration with POS systems).
- • Define delivery partner needs (optimized routes, order notifications, earning records).

Stakeholder Analysis

- Customers: Expect timely and accurate deliveries.
- Restaurants: Need smooth order processing and revenue tracking.
- Delivery Partners: Require optimized delivery routes and fair compensation.
- Business Admins: Seek data-driven insights and system scalability.

Business Process Mapping

- Order Placement → Order Acceptance → Food Preparation → Delivery Assignment →
- Real-time Tracking → Delivery Completion.
- Identify bottlenecks such as late preparation, delivery delays, or incorrect orders.
- Map system touchpoints for automation and integration.

Industry-specific Use Case Analysis

Use Case	Description
POS System Integration	Seamless order flow from restaurants' POS systems to the delivery app.
Real-time GPS Tracking	Delivery agents tracked in real-time for transparency and reliability.
Customer Feedback	Post-delivery ratings and reviews for quality improvement.
Demand Forecasting	Prediction of peak demand hours for better resource allocation.

AppExchange Exploration

- Explore Salesforce solutions like Delivery Management apps, Route Optimization tools, and Customer Feedback systems.
- Evaluate third-party integrations for payments, customer engagement, and analytics.

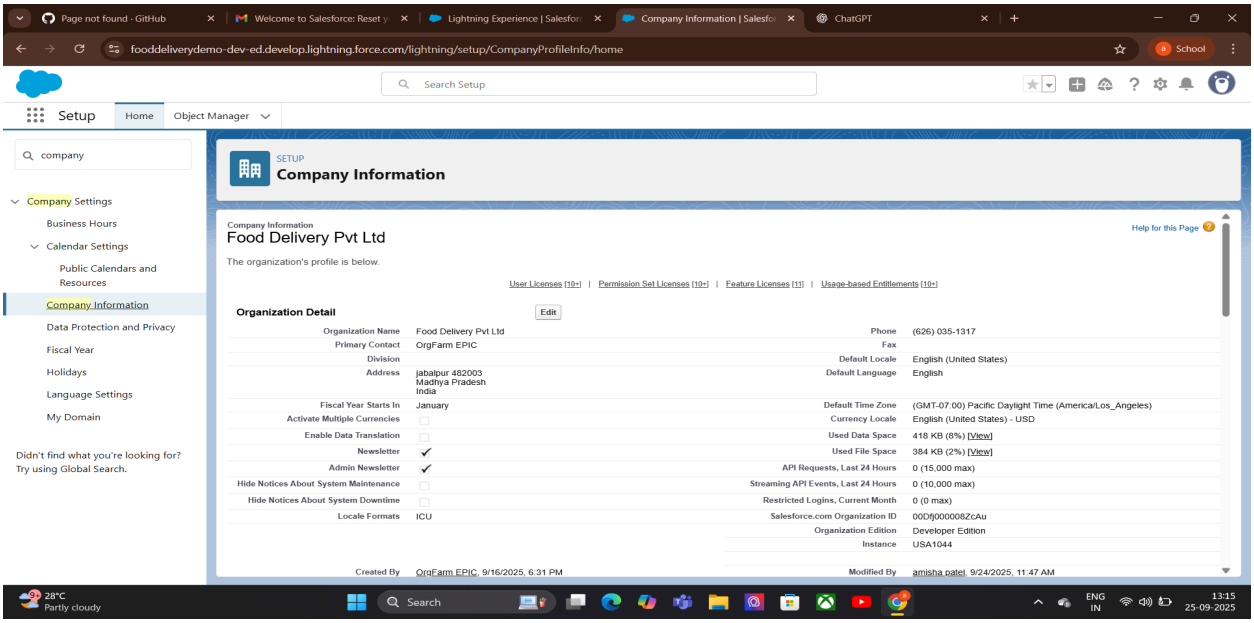
Phase 2: Org Setup & Configuration

1. Salesforce Editions

- Salesforce Editions define the features, customization level, and limits available in your Salesforce org. Choosing the right edition is crucial because it determines how you can build and scale your **Food Delivery App**.
- Edition -Features Relevant to Food Delivery App
- Developer -Free, supports full customization, ideal for prototype and testing
- Professional-Good for small-scale apps; supports custom objects, automation, reports

2. Company Profile Setup

- **Company Information:**
- Company Name: *FoodDelivery Pvt Ltd*
- Address: Relevant business location
- Primary Contact: Admin email & phone

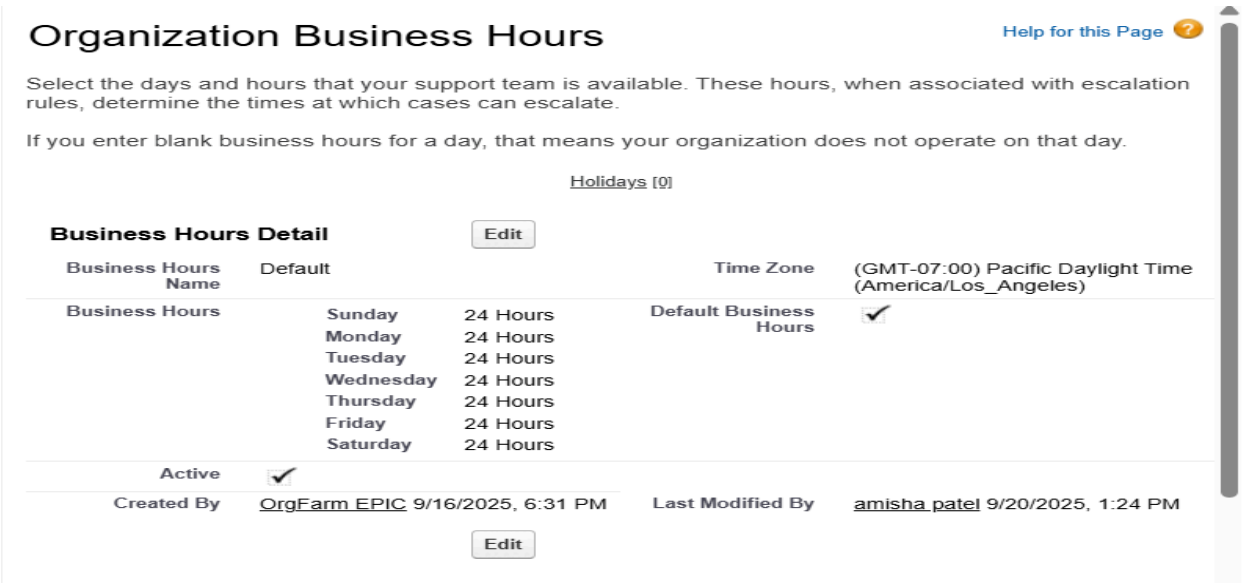


3.Business Hours:

Purpose: Define the hours during which your food delivery service operates to ensure accurate order processing and delivery tracking.

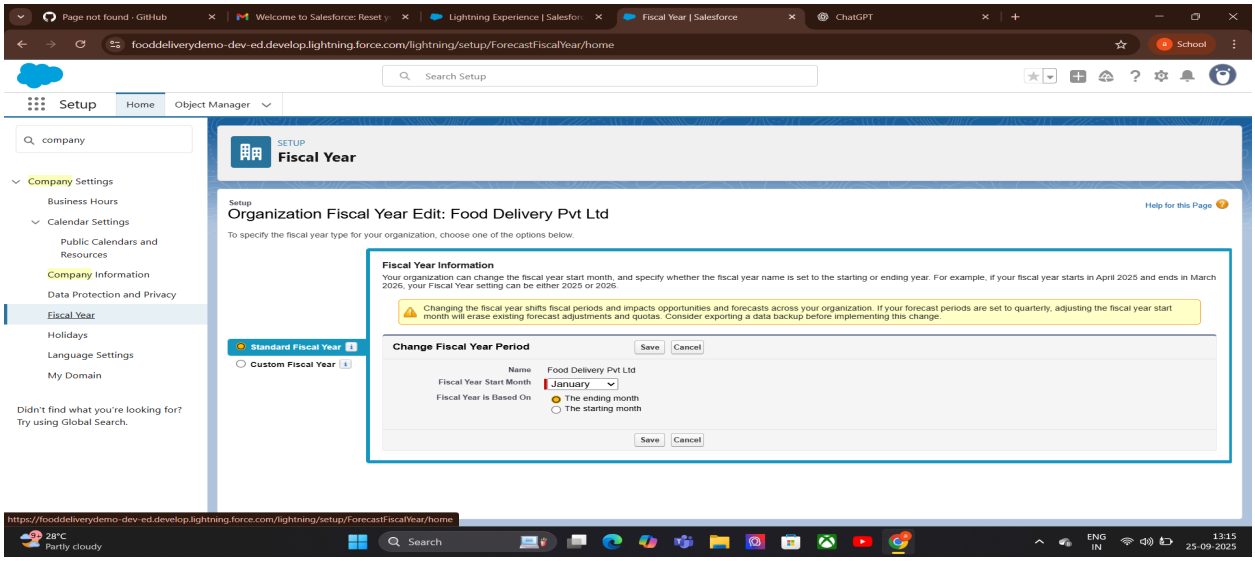
Holidays:

- Add public holidays or days restaurants are closed



4.Ficsal Year Settings (Food Delivery App):

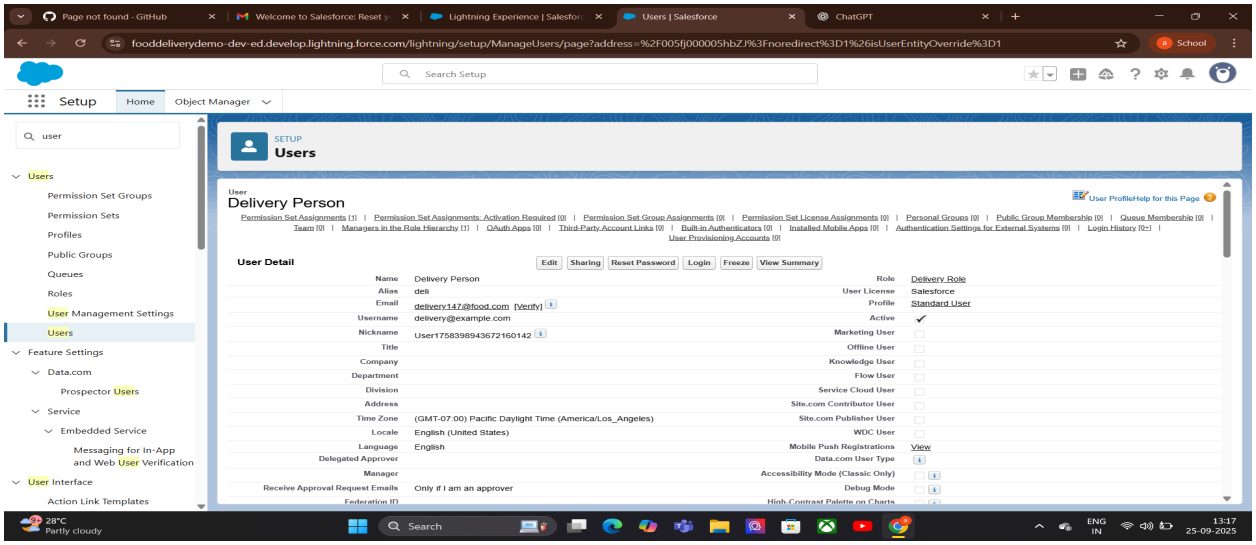
- Define standard (Jan–Dec) or custom fiscal year for revenue tracking



5.User setup & Licenses:

For a Food Delivery App, different users are involved:

Customers: View menu, place orders.
Restaurant Managers: Manage orders, update menu, track delivery status.
Delivery Agents: View assigned orders, update delivery status.
Admins: Full access to configure the system.

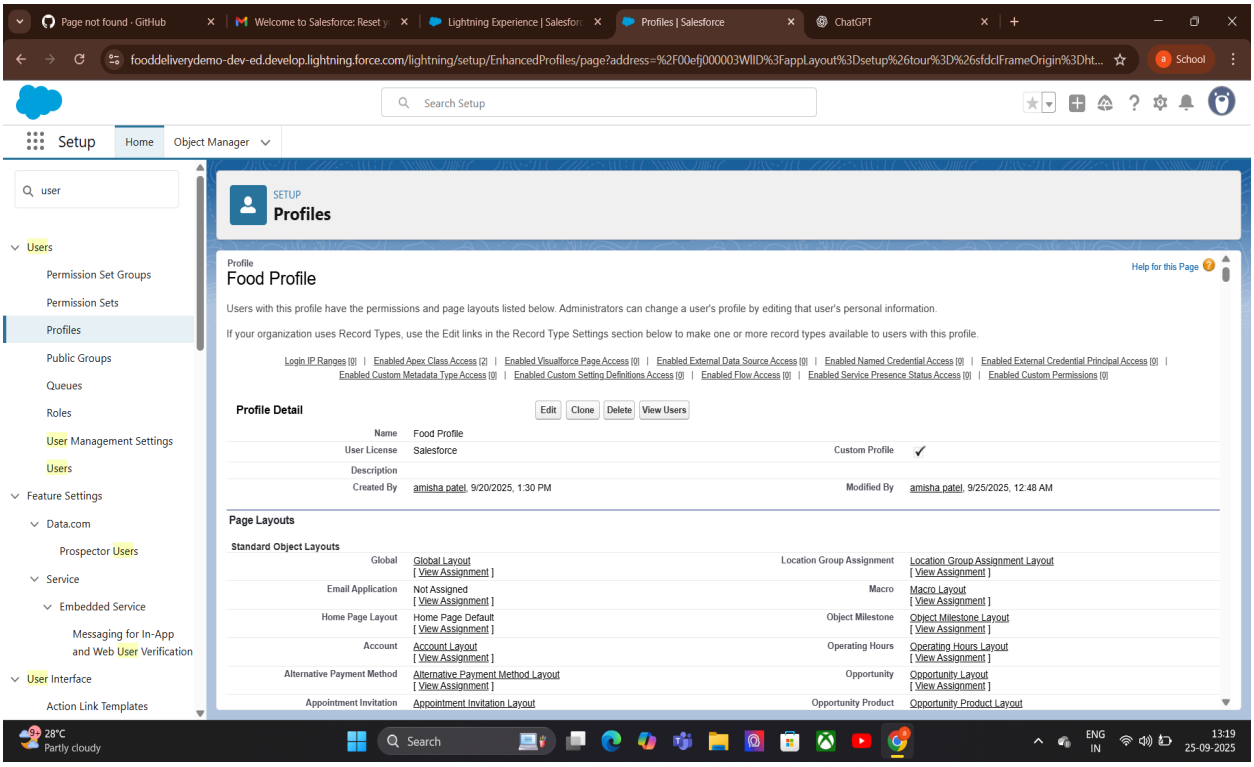


6.Profile:

Profiles (Food Delivery App)

In Salesforce, a **Profile** defines a user's permissions, access levels, and settings. It controls what a user can see and do in the system.For a Food Delivery App, profiles are important to separate access for different roles:

- **Customer Profile:** Access to order creation, order history, and menu viewing.
- **Restaurant Manager Profile:** Access to orders, menu management, reports, and dashboards.
- **Delivery Agent Profile:** Access to assigned orders, status updates, and delivery history.
- **Admin Profile:** Full access to all features, settings, and data.

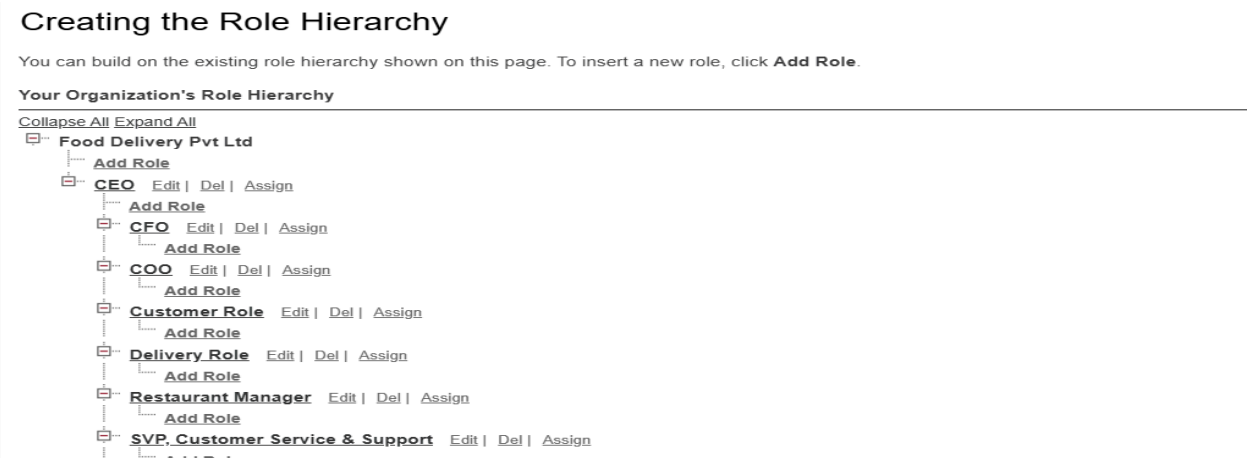


7.Roles:

Roles (Food Delivery App)

In Salesforce, a **Role** defines a user's position in the organization's hierarchy and controls record-level access. Roles work with **Sharing Rules** to determine who can see and act on specific records.

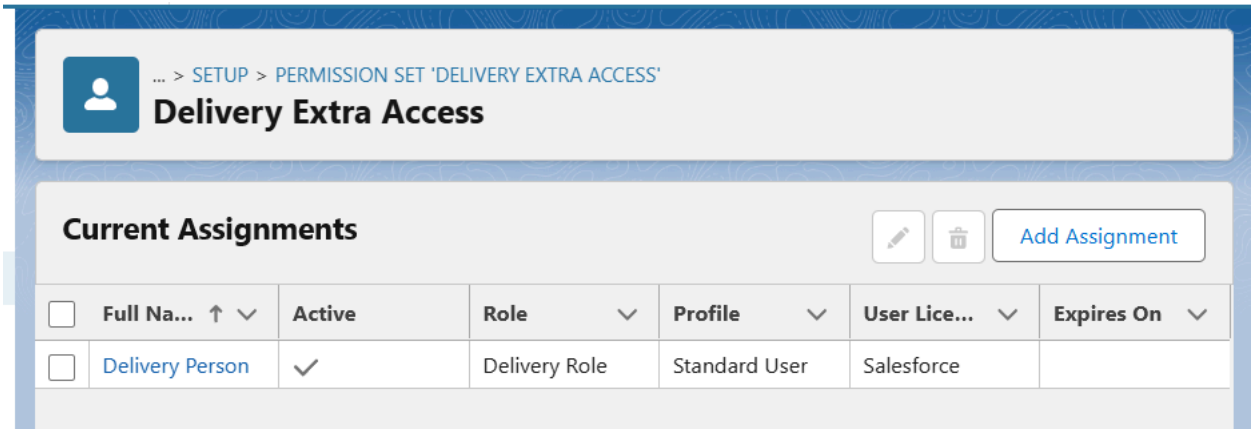
Admin Role, Restaurant Manager Role,Delivery Agent Role,Customer Role.



8.Permission set:

For a Food Delivery App, permission sets can be used to:

- Allow certain delivery agents to access advanced tracking features.
- Give restaurant managers extra permissions for special menu updates.



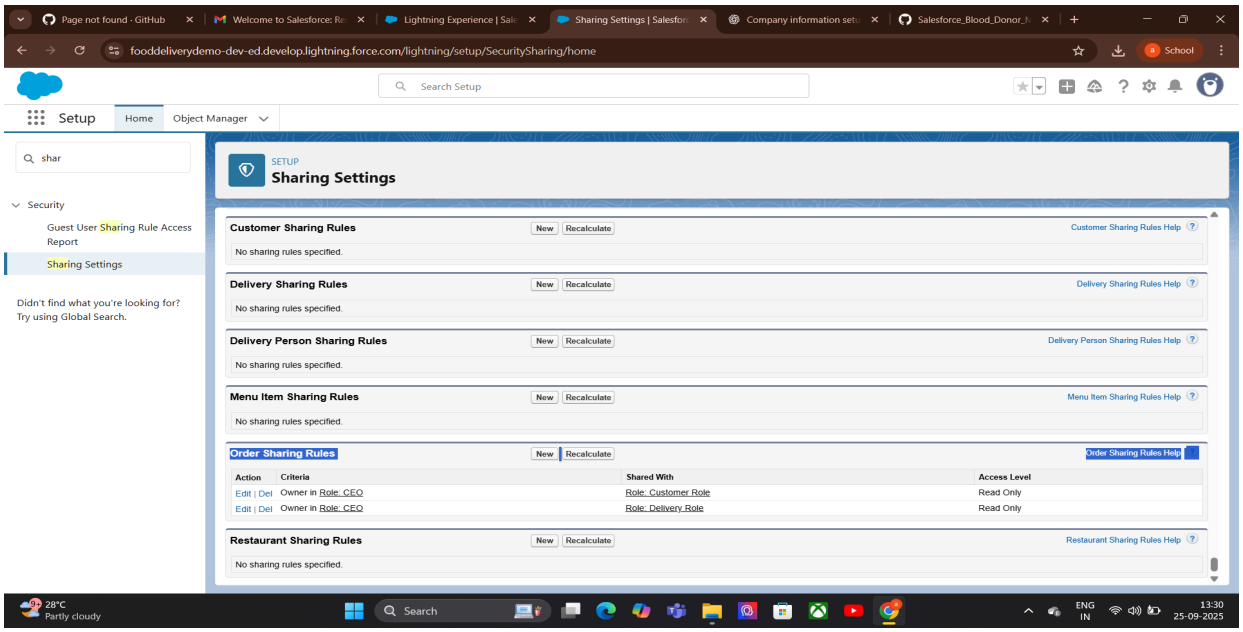
9.Owd & Sharing Rule:

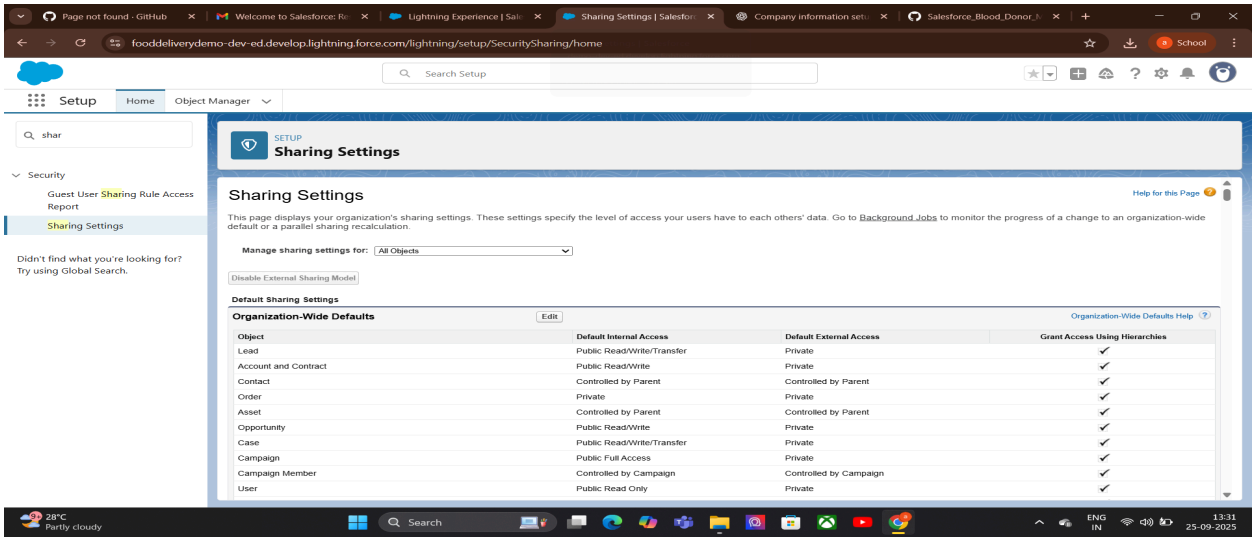
purpose: OWD determines the baseline level of access users have to records they do not own. It's the foundation of data security.

- Set **Menus** or **Restaurants** objects to **Public Read Only** so all users can view menus but not edit them.

Purpose: Sharing Rules expand access beyond OWD settings, allowing specific roles or groups to view or edit records.

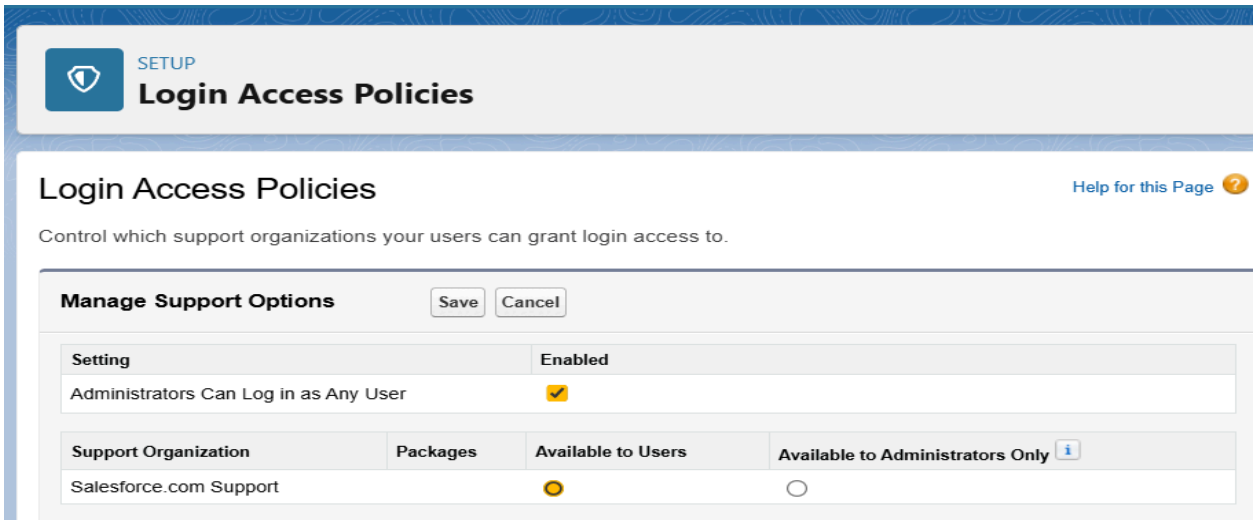
- Share orders with the relevant restaurant manager role so they can manage orders for their restaurant.
- Share order details with delivery agents assigned to those orders.





10.Login Access Policies:

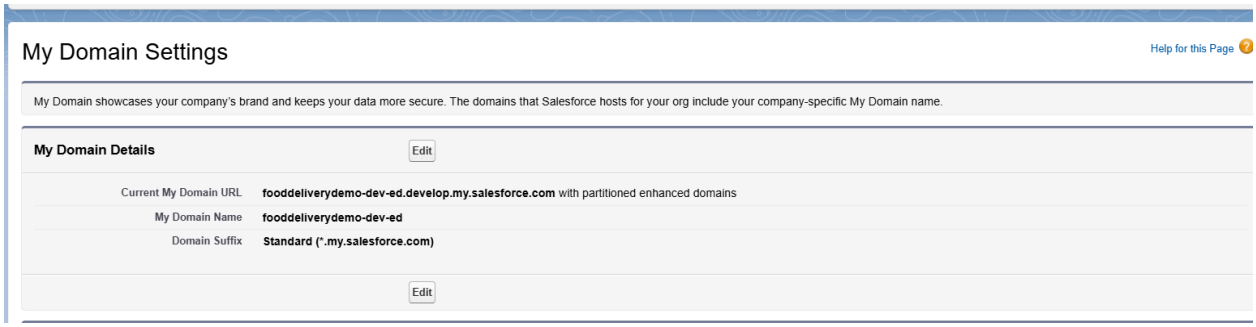
Purpose: Login Access Policies control how and when users can log in to Salesforce. They help ensure security and manage access for different roles'



11.Dev Org setup:

Purpose: A Developer Org in Salesforce is a free environment used for building, testing, and learning before deploying to production. It provides full access to most Salesforce features without affecting live data.

Use **Developer Edition** to build and test objects like Orders, Restaurants, and Delivery Agents.



12.Sandbox Usage:

.**Developer Edition of Salesforce does NOT include full sandbox environments**.
Developer Edition itself acts like a personal sandbox where you can build, test, and experiment
For team development or staging, sandboxes are available only in **Enterprise, Unlimited, Performance, and Developer Pro editions**, not in Developer Edition.

13.Deployment Basic:

Purpose: Deployment is the process of moving changes (objects, fields, workflows, code, etc.) from one Salesforce environment to another, such as from a Developer Org or Sandbox to Production.

For a Food Delivery App:

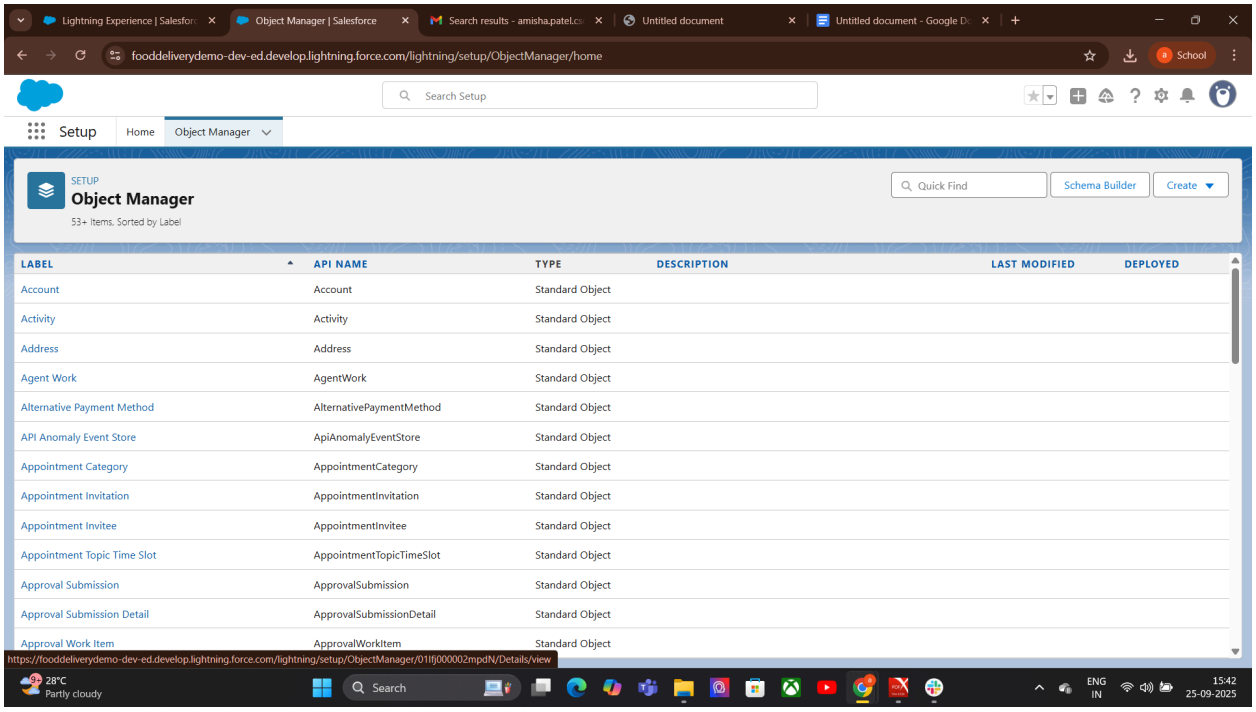
Change Sets: Simple way to deploy from Sandbox to Production (best for small changes).
Salesforce CLI & Git: Use for version control and advanced deployments.
Unmanaged Package: Package all app components for deployment or sharing.
Testing Before Deployment: Ensure workflows, sharing rules, and LWCs are tested in the Dev Org/Sandbox.

Phase 3: Data Modeling &Relationships

1.Standard & Custom Objects:

We create objects specific to food delivery:

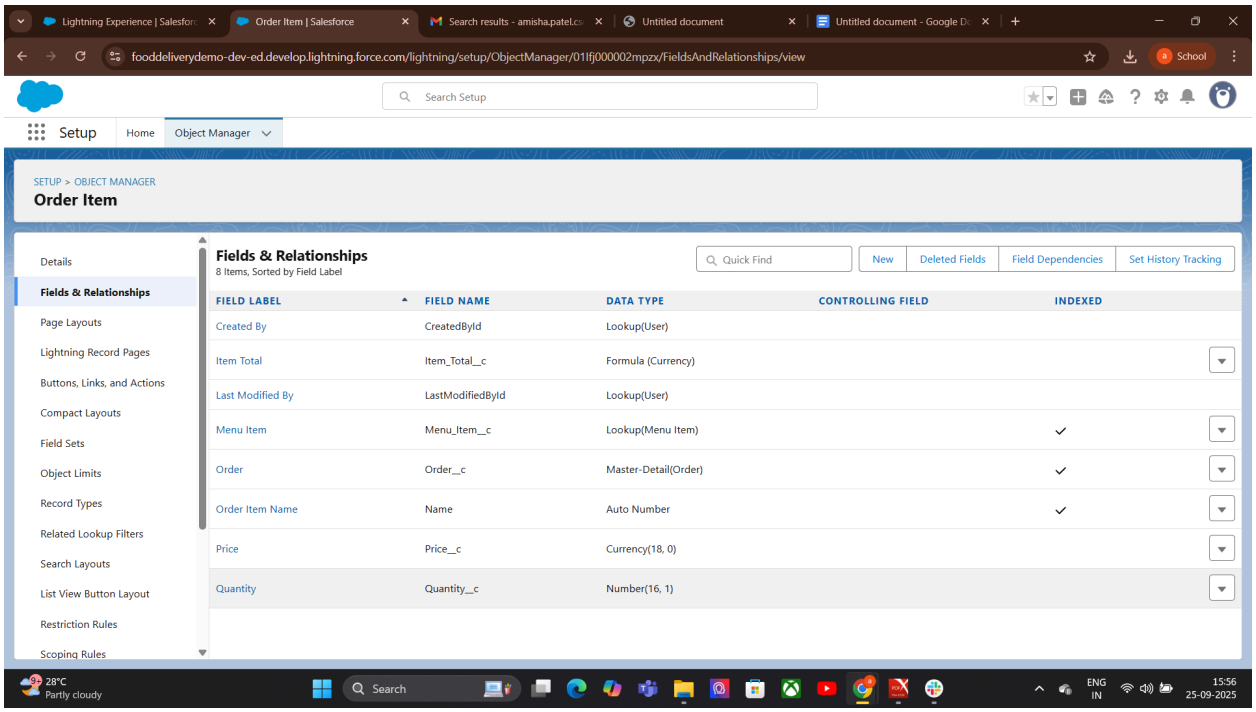
Custom Object	Purpose
Order_c	**Store order details.**
Menu_Item__c	**Store menu items for restaurants.**
Delivery__c	**Track delivery assignments and status.**
Restaurant__c	**tore additional restaurant info not covered by Account.**



2.Fields:

very Object (Standard or Custom) is made up of Field

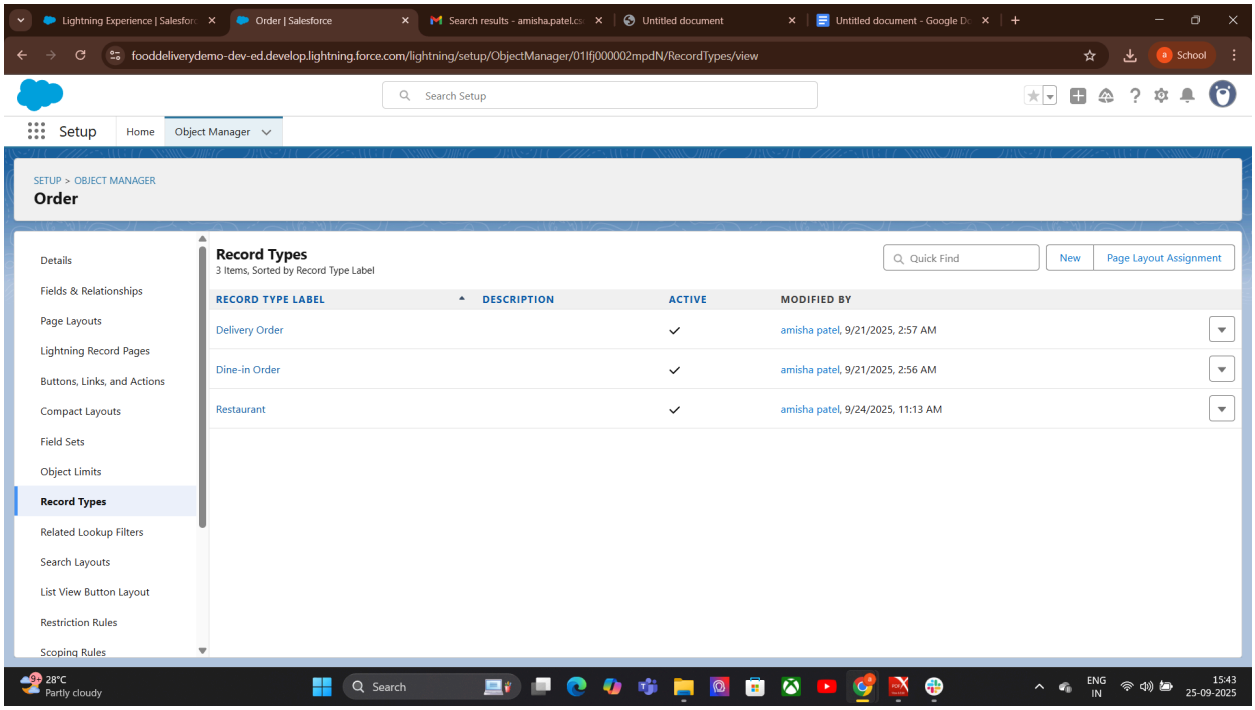
Order Number → Auto Number (e.g., ORD-0001)
Order Date & Time → Date/Time
Order Status → Picklis



3.Record Types:

Record Types let you create different business processes within the same object.

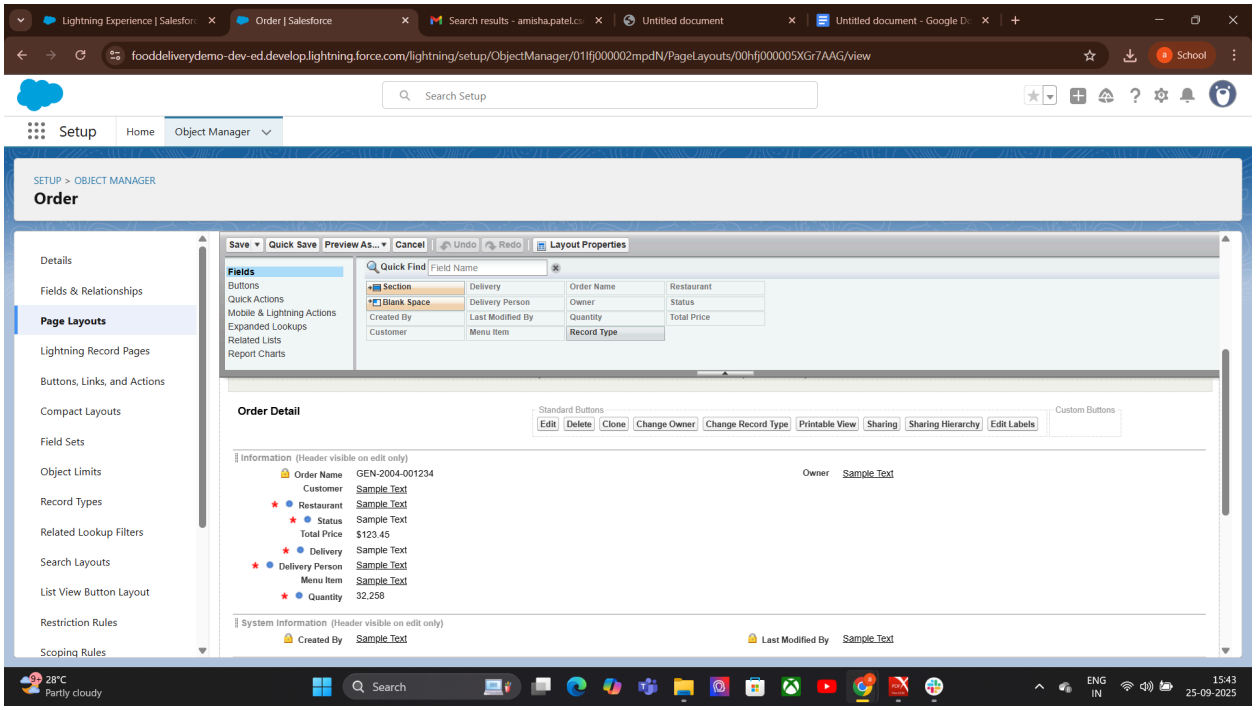
- Dine-In Order - Fields: Table Number, Waiter Assigned.
- Deli very Order - Fields: Delivery Address, Delivery Person.
- Restaurant__c - Menu_Item__c



4.Page Layouts:

Purpose: Control which fields, sections, related lists, and buttons appear on a record page.

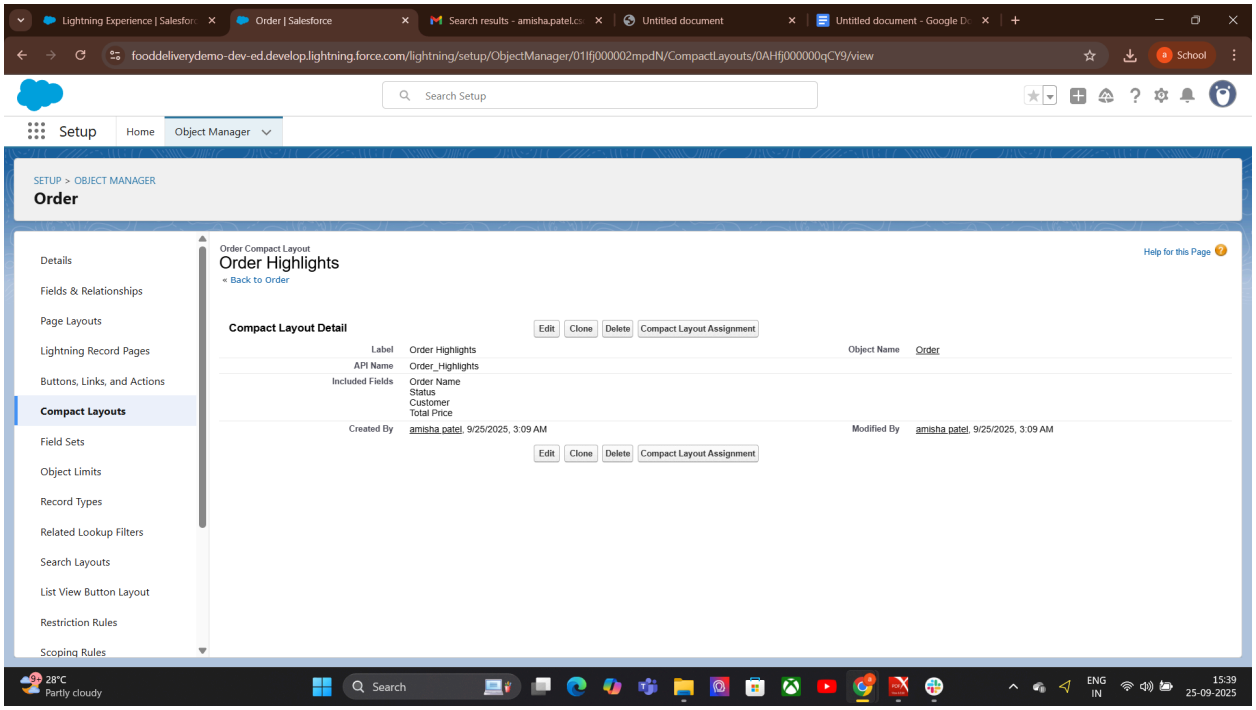
Object Manager → Object → Page Layouts → Edit Drag & drop fields, sections, related lists → Save.



5.Compact Layout:

How to Create/Edit a Compact Layout:

- Go to **Setup → Object Manager**.
- Select the object (e.g., `Order__c`).
- Click **Compact Layouts** in the left sidebar.
- Click **New** → Enter a **Label** (e.g., “Delivery Highlights”).
- **Select up to 10 fields** to display (best practice: 4–6 fields for clarity).
- **Save.**
- **Compact Layout Assignment** → choose your layout → **Save**.

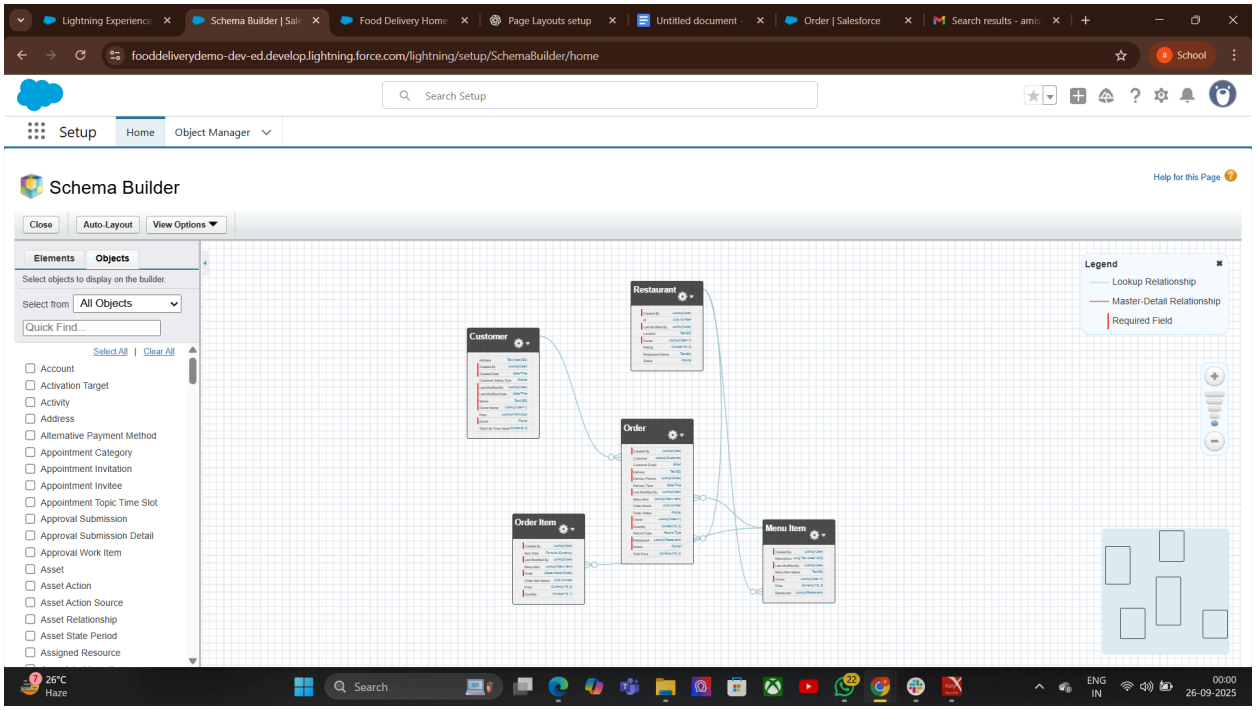


6. Schema Builder:

Schema Builder Purpose: Visual tool to see objects and their relationships.
How to use:

Setup → Schema Builder

Drag objects → view fields, relationships, junction objects



7. Lookup vs Master-Detail vs Hierarchical Relationships:

Lookup Relationship — step wise

Use when you want a loose/optional link between two objects.

1. Decide: pick **Parent** (e.g., `Customer__c`) and **Child** (e.g., `Order__c`).
2. Setup → **Object Manager** → open **Child object** (Order_c).
3. **Fields & Relationships** → **New**.
4. **Fields & Relationships** → **New**.
5. Choose **Lookup Relationship** → **Next**.
6. **Related To:** select the **Parent object** (`Customer__c`) → **Next**.
7. Set **Field Label** (e.g., `Customer`) and **Field Name** → **Next**.

Master-Detail Relationship — step wise

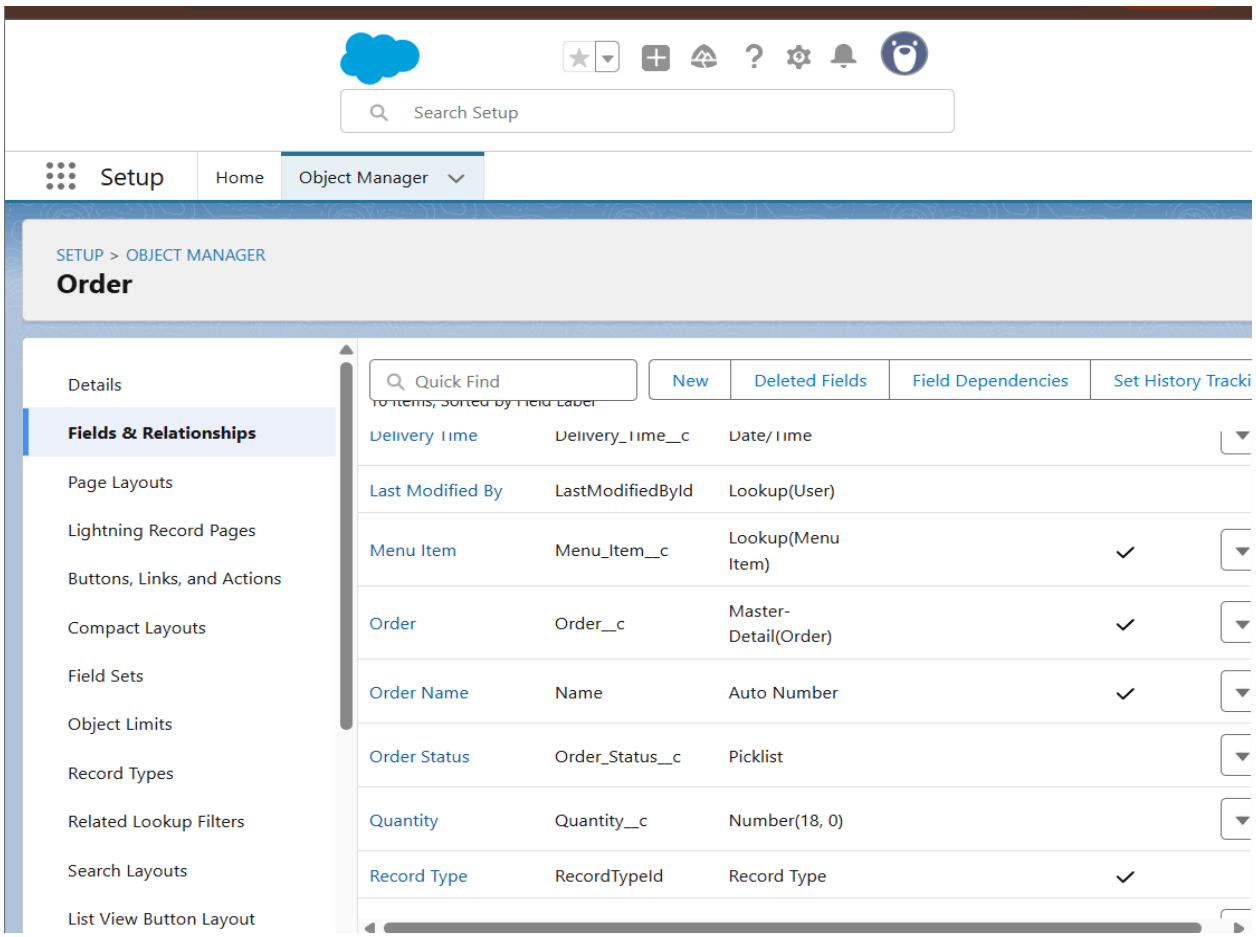
`Order_Item__c` (junction) → Master-Detail → `Order__c`

`Order_Item__c` → Master-Detail → `Menu_Item__c`

Hierarchical Relationships:

`User (Delivery Person)` → Manager is another `User (Team Lead)`.

Lets you track which Delivery Person reports to which Manager.

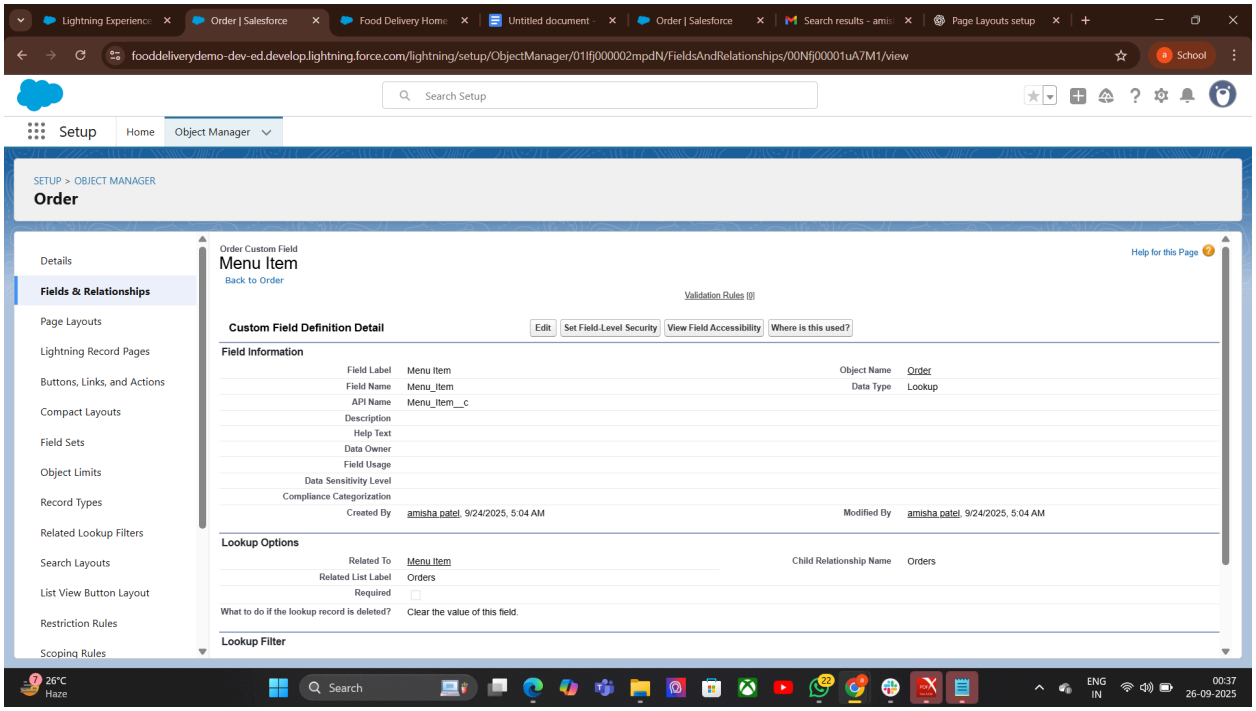


8. Junction Objects:

Setup → Object Manager → Create → Custom Object → Order_Item__c (Junction).
In Order_Item__c → Fields & Relationships → New → Master-Detail Relationship → pick Order__c → Save.
Repeat: New → Master-Detail Relationship → pick Menu_Item__c → Save

.Add extra fields on Order_Item__c: Quantity, Price, Special Instructions.

Test: Create an Order and add multiple Order_Item records linking different Menu Items.



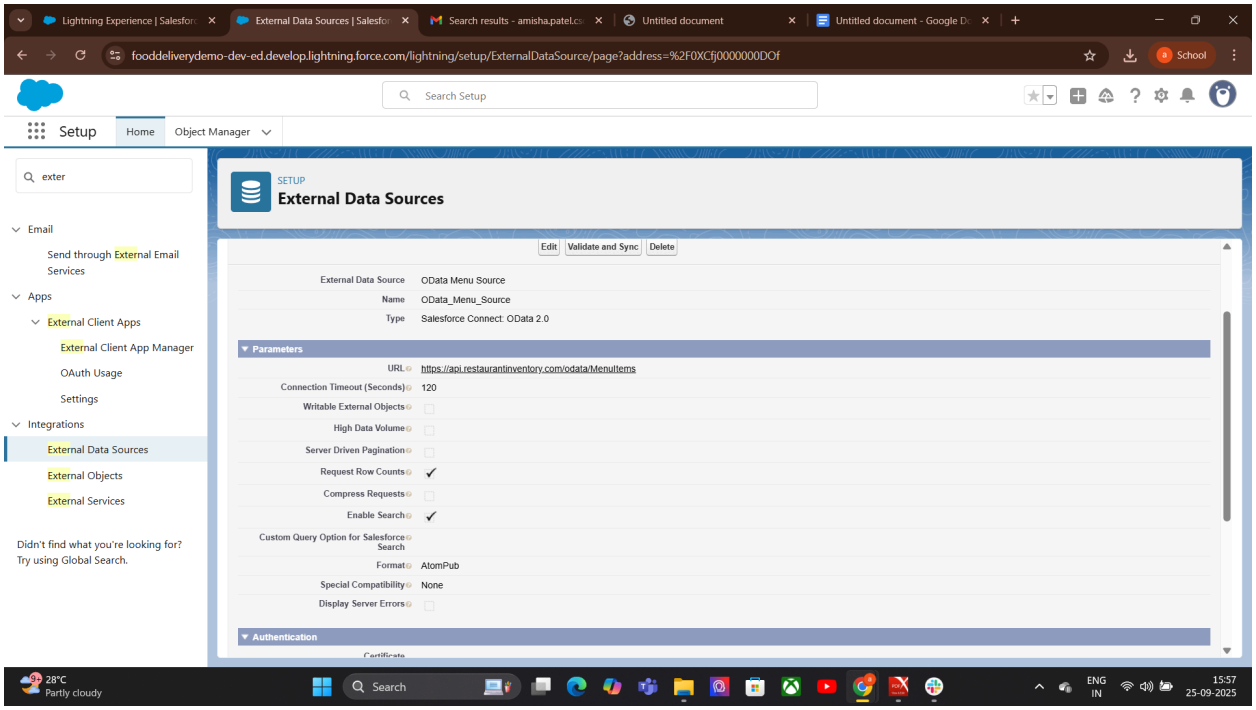
9. External Objects:

Step 1 — Create the External Data Source

- Go to **Setup → Quick Find → External Data Sources**.
- Click **New External Data Source**.
- Fill in:
- **Name:** `OData_Menu_Source`
- **Type:** `OData 4.0` (or 2.0 depending on your external API)
- Click **Save**.

Use External Object in App

- Go to **App Launcher → OData_Menu_Source** tab.
- You'll see live menu data from external system
- You can use it in reports, list views, and reference it in your Food Delivery app.



Phase 4: Process Automation (Admin)

1. Validation Rules:

Food Delivery App ke Examples

Example 1 — Quantity must be greater than

Object: Order_Item__c

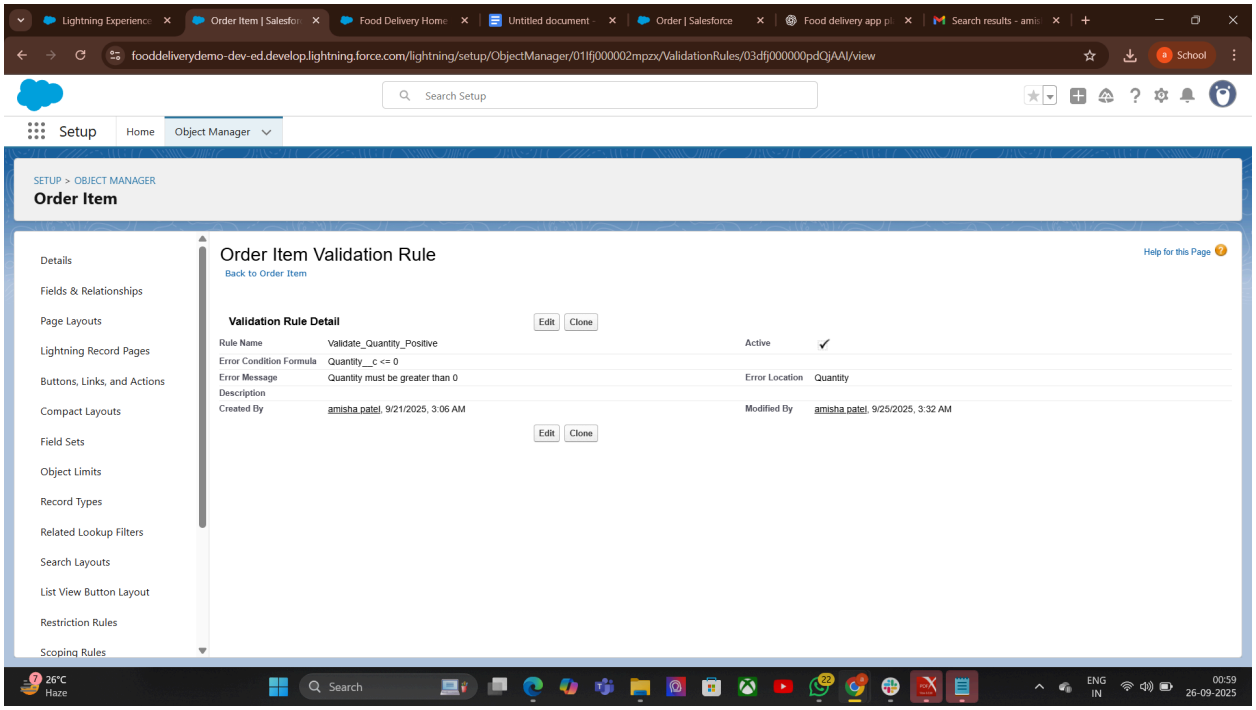
Field: Quantity__c

Formula:

Quantity__c <= 0

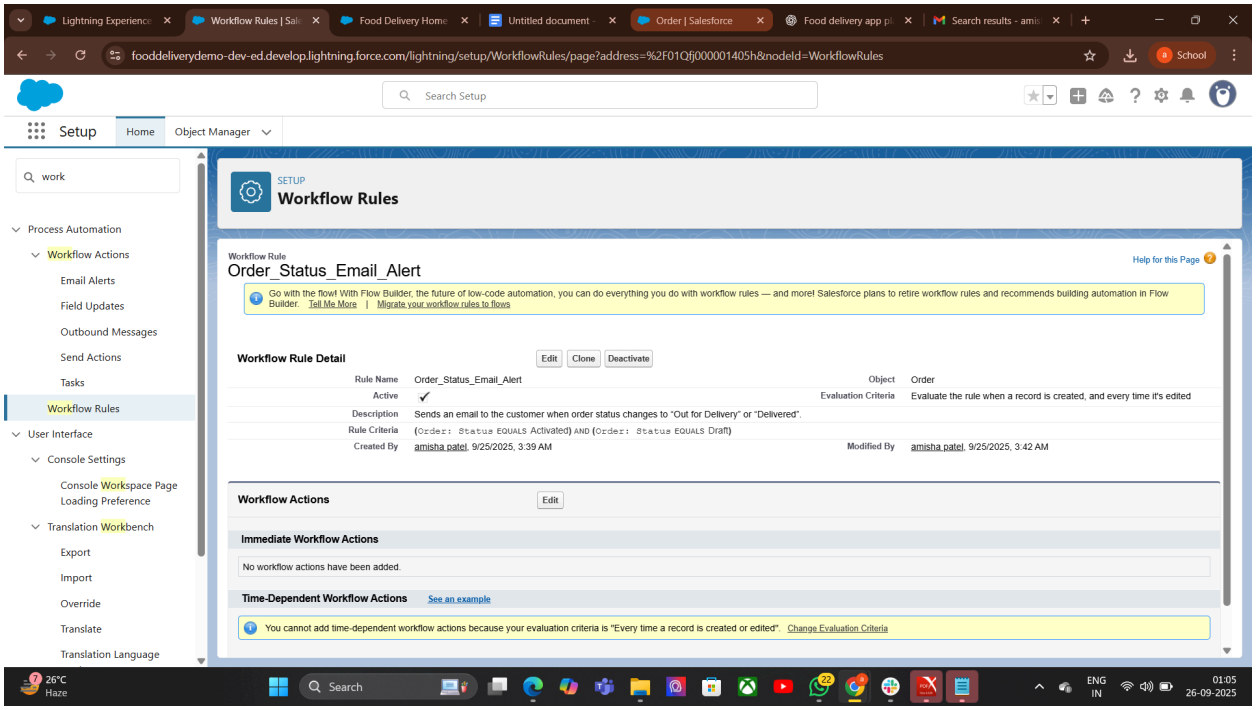
Error Message:

`Quantity must be greater than 0.`



2.Workflow Rules:

- How to Create Workflow Rule
- Go to Setup
  Search → Workflow Rule
- Click New Rule
- Select Object (e.g., Order__c)
  Define Rule Criteria & Evaluation Criteria
  Add Immediate or Time-Based Actions
  Activate Rule

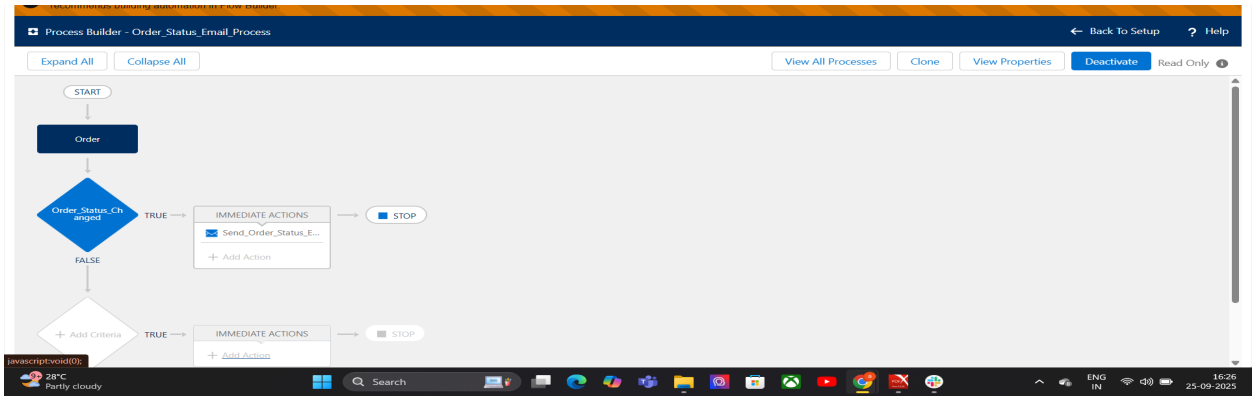


3.Process Builder:

How it Works (Process Builder approach)

Go to Setup → Process Builder → New
Give it a name: Notify Delivery Person on Order Assignment
Choose the Order__c object

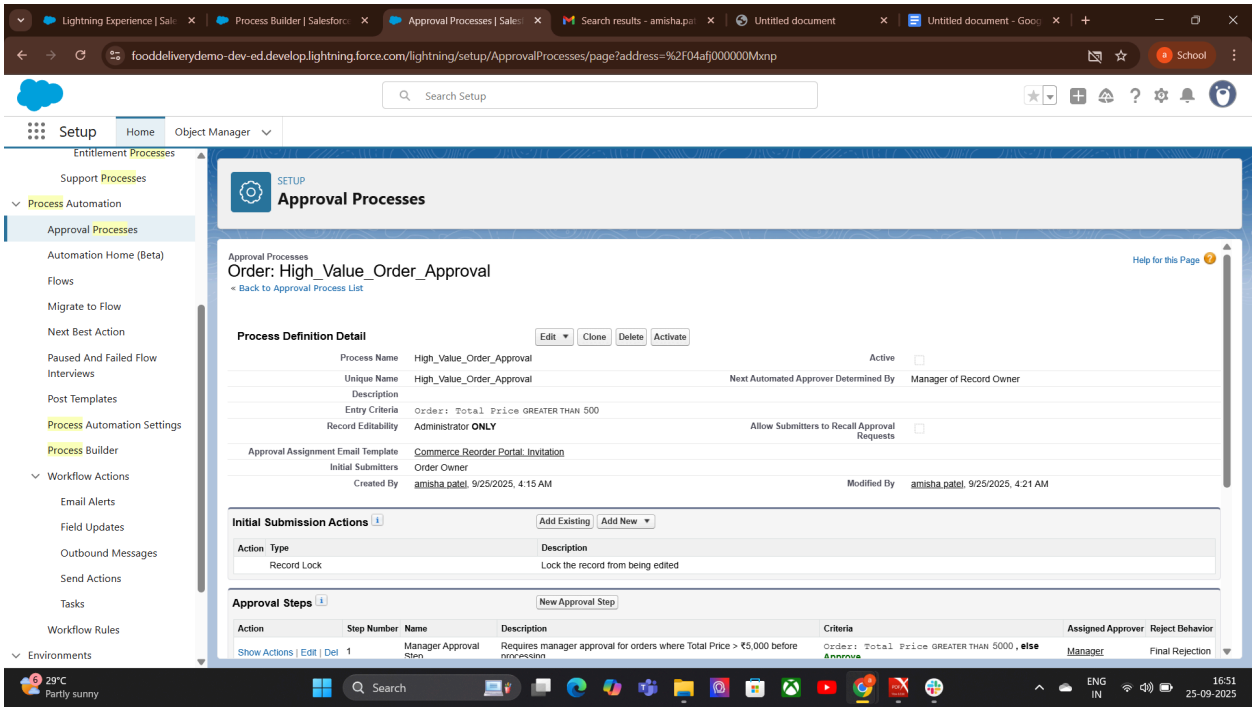
Trigger: When a record is created or edited
Criteria:
Immediate Actions:



4. Approval Process:

components of an Approval Process

Entry Criteria – conditions that determine which records enter the approval process
Approval Steps – who approves and in what order
Actions – what happens when a record is approved, rejected, or recalled
Final Approval/Rejection Actions – automated updates, notifications, field changes



5.Flow Builder (Screen, Record-Triggered, Scheduled, Auto-launched):

Screen Flow: - Screen Flow
Purposenteractive flows that require user input.

- Use case: Guided data entry, wizards, onboarding processes.
- Key features:
- Uses screens to collect data.
- Can display UI elements like forms, choices, checkboxes.

Record Flow: - Purpose: Automatically runs when a record is created, updated, or deleted.

- Use case: Automations tied to record changes.
- Key features:
- Runs automatically in response to record changes.
- Can run before save (fast updates) or after save (actions, notifications).
- Example: Auto-update account status when an opportunity is closed.

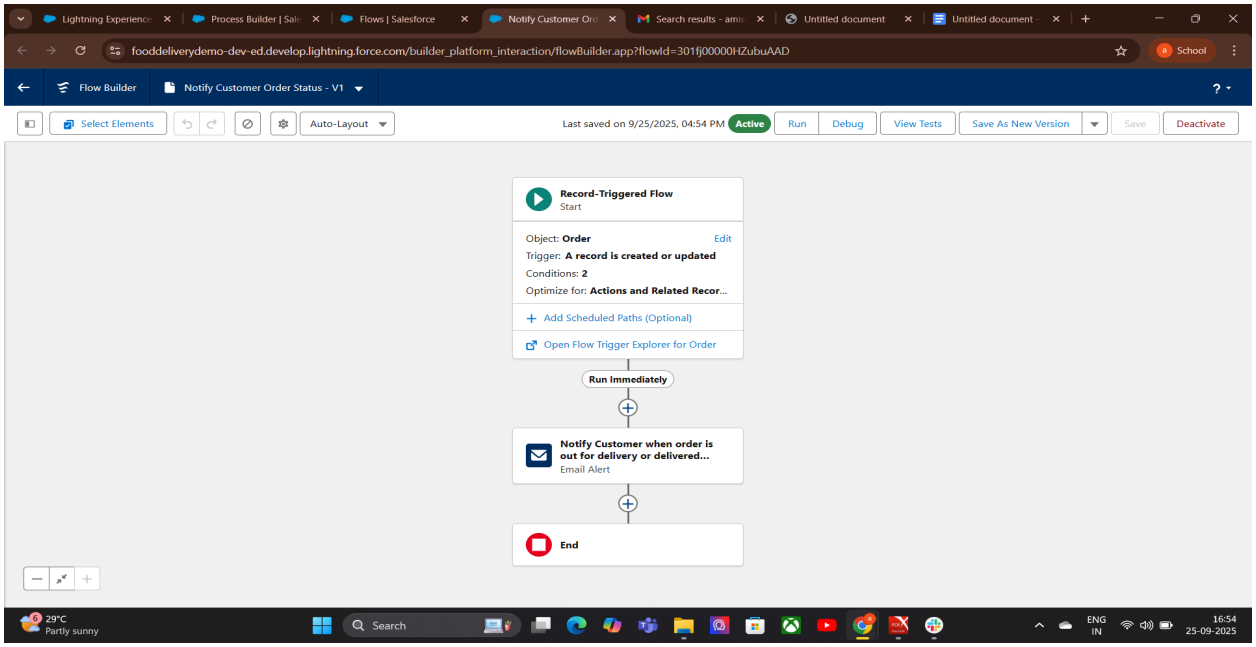
Scheduled-Purpose: Runs automatically at a specific time or recurring schedule.
Use case: Time-based automation without human action.

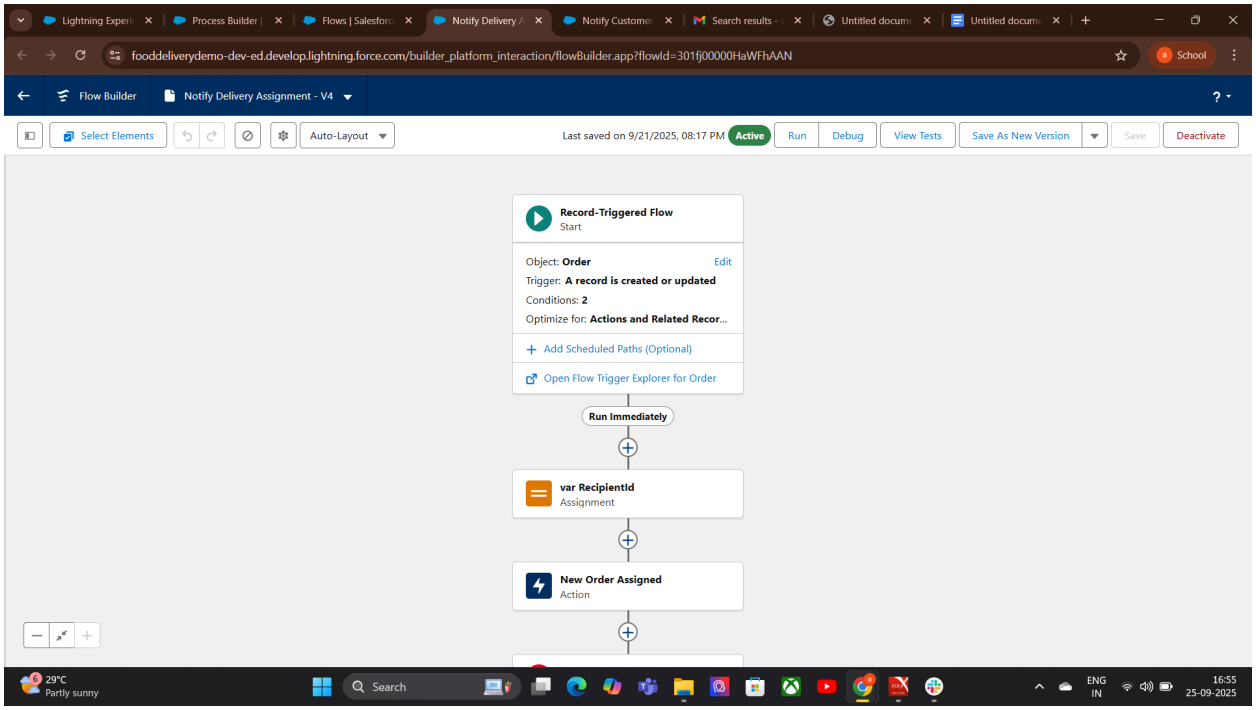
- Key feature
- Runs on a schedule (daily, weekly, hourly)
- Can query records and process them in batch.

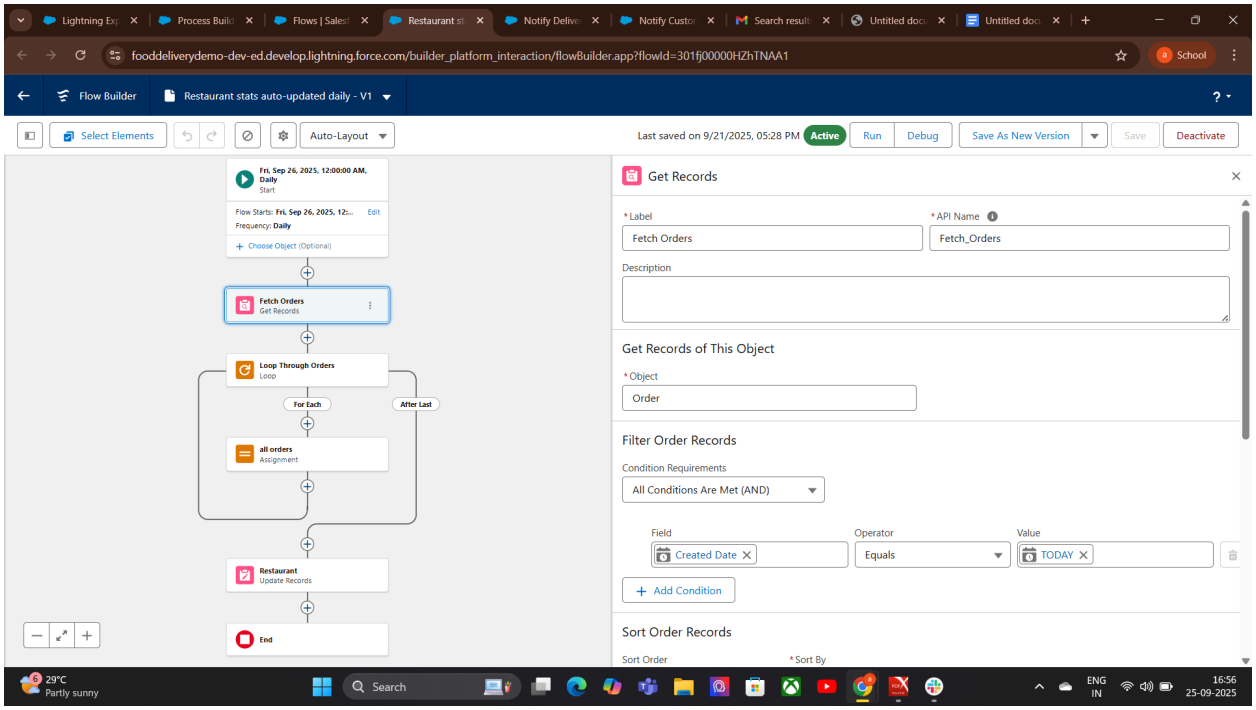
Auto-launched Flow-

Purpose: Runs in the background without user interaction.
Use case: Triggered by processes, Apex, other flows, or APIs.
Key features:

- No screens or user interaction.
- Often used for backend automation


Lightning Experience
Process Builder | Sale
Flows | Salesforce
Notify Customer Orc
Search results - amis
Untitled document
Untitled document -
fooddeliverydemo-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301fj00000HZubuAAD
School
Flow Builder
Notify Customer Order Status - V1
Select Elements
Auto-Layout
Last saved on 9/25/2025, 04:54 PM
Active
Run
Debug
View Tests
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Object: Order
Edit
Trigger: A record is created or updated
Conditions: 2
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Order
Run Immediately
Notify Customer when order is out for delivery or delivered...
Email Alert
End
29°C
Partly sunny
Search
ENG
IN
16:54
25-09-2025

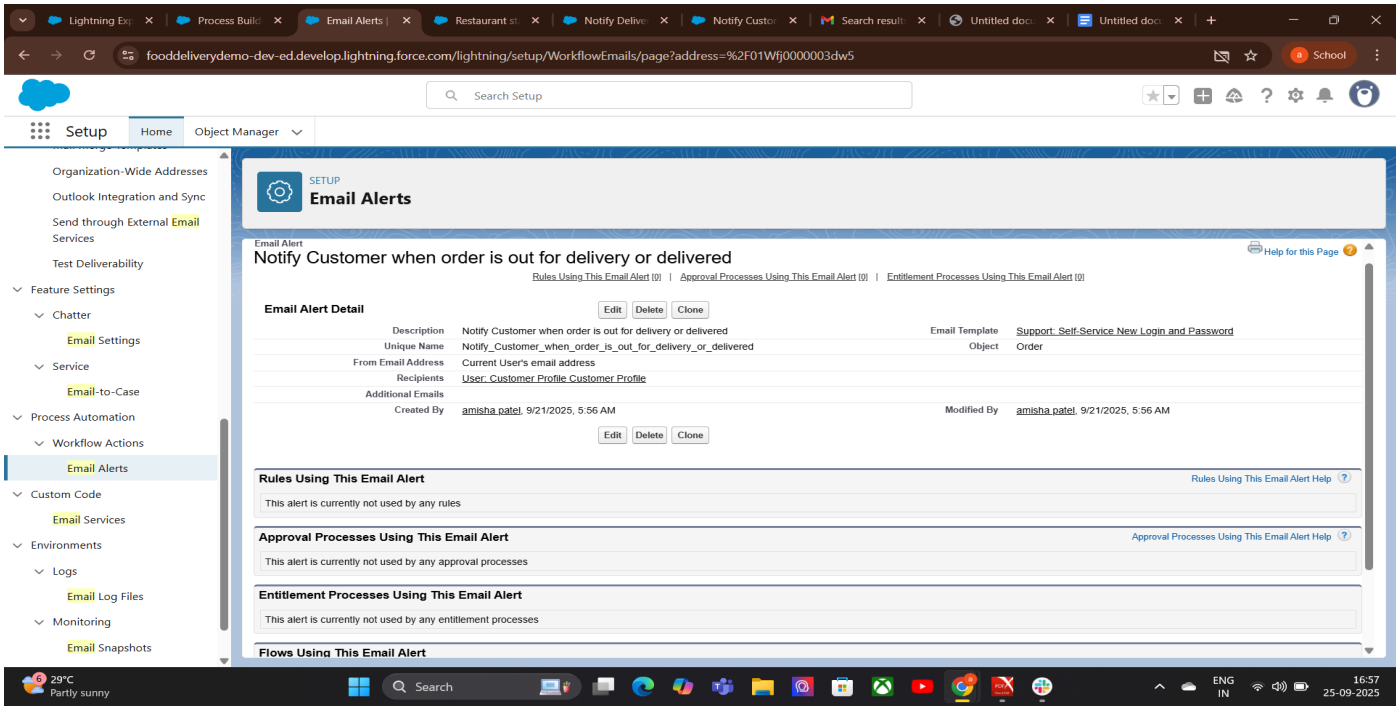

Lightning Experie
Process Builder |
Flows | Salesforc
Notify Delivery A
Notify Customer
Search results -
Untitled docume
Untitled docume
fooddeliverydemo-dev-ed.develop.lightning.force.com/builder_platform_interaction/flowBuilder.app?flowId=301fj00000HaWFhAAN
School
Flow Builder
Notify Delivery Assignment - V4
Select Elements
Auto-Layout
Last saved on 9/21/2025, 08:17 PM
Active
Run
Debug
View Tests
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Object: Order
Edit
Trigger: A record is created or updated
Conditions: 2
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Order
Run Immediately
var RecipientId
Assignment
New Order Assigned
Action
29°C
Partly sunny
Search
ENG
IN
16:55
25-09-2025



6. Email Alerts:

How to Create an Email Alert

- Go to Setup → Email Alerts.
- Click New Email Alert.
- Choose:
- Email Template (design the message here first).
- Object (the record type this email will relate to).
- Recipients (users, roles, fields, groups).
- Save the email alert.

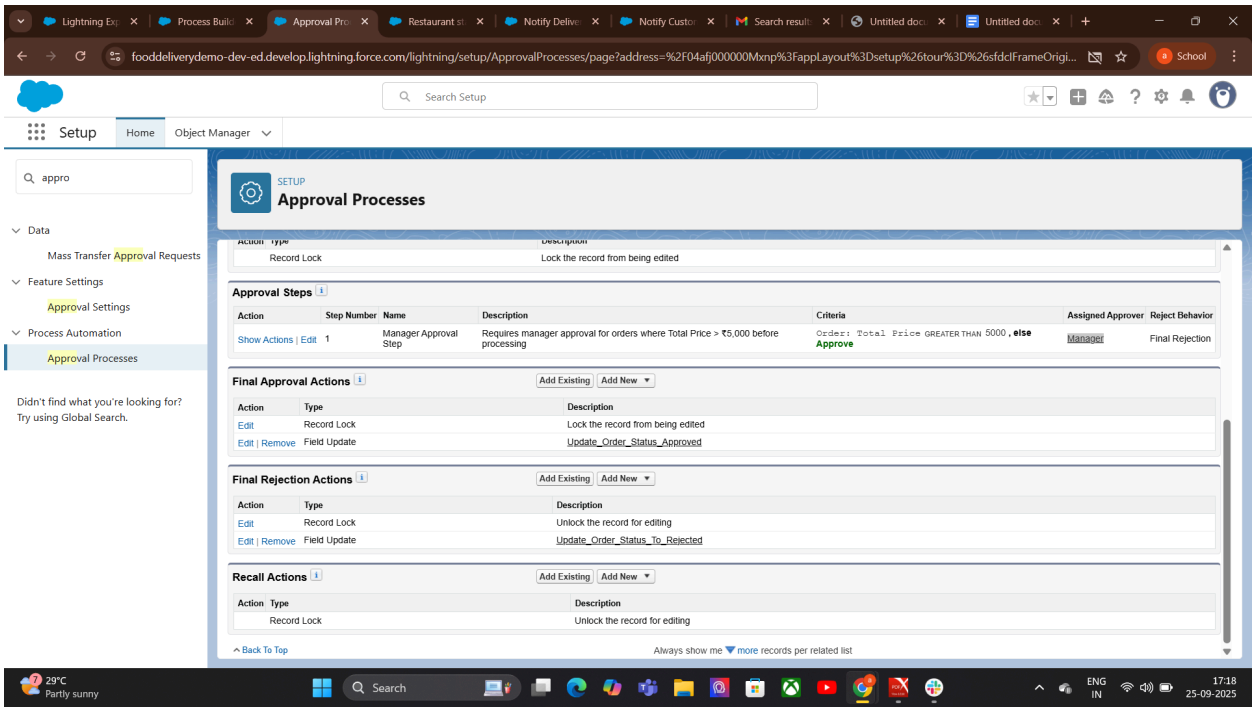


6.Field Updates:

Go to Setup → Workflow Rules

Select or create a **Workflow Rule** for an object.

- Add a **Field Update action**:
- Choose the field to update.
- Choose the new value:
- **A specific value** (e.g., “Closed” for Status)
- **A formula result**
- **A value from a related record**
- Save and activate.

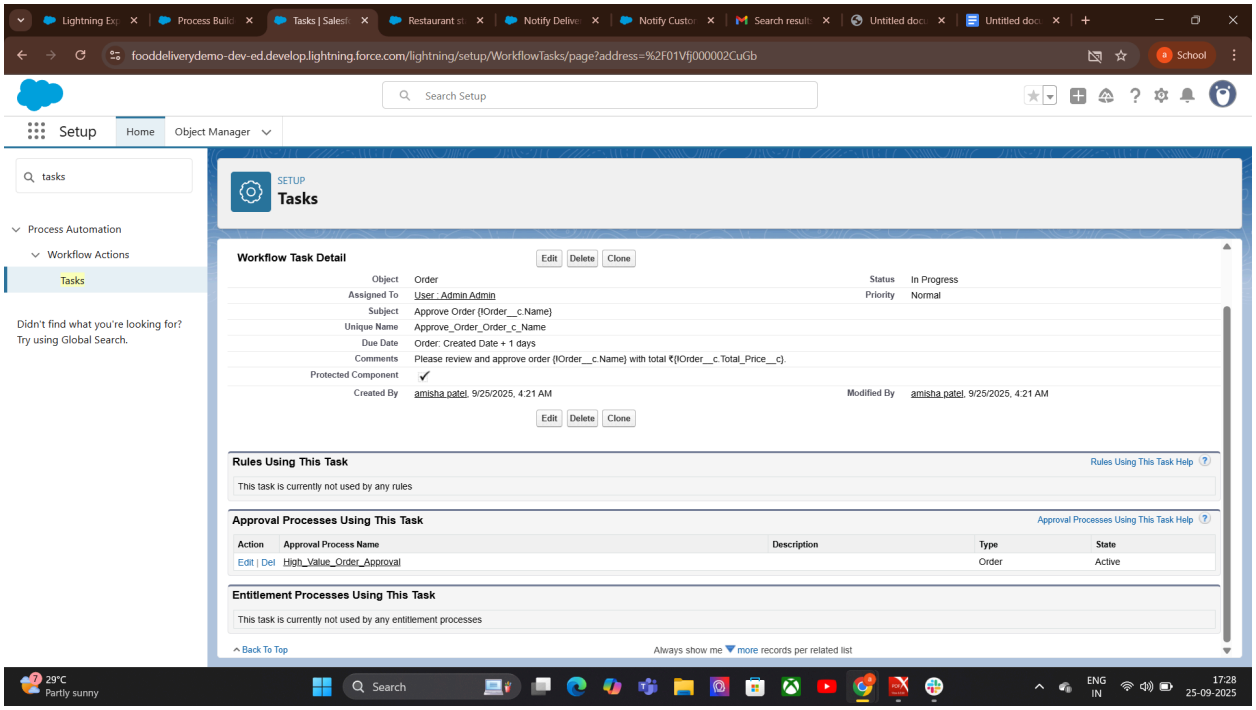


7. Tasks:

Automatically (Workflow/Process/Flow):

Create automation rule (workflow rule, process builder, or flow).
Add a Create Task action.
Define:

- Assigned user
- Task details (subject, due date, etc.)

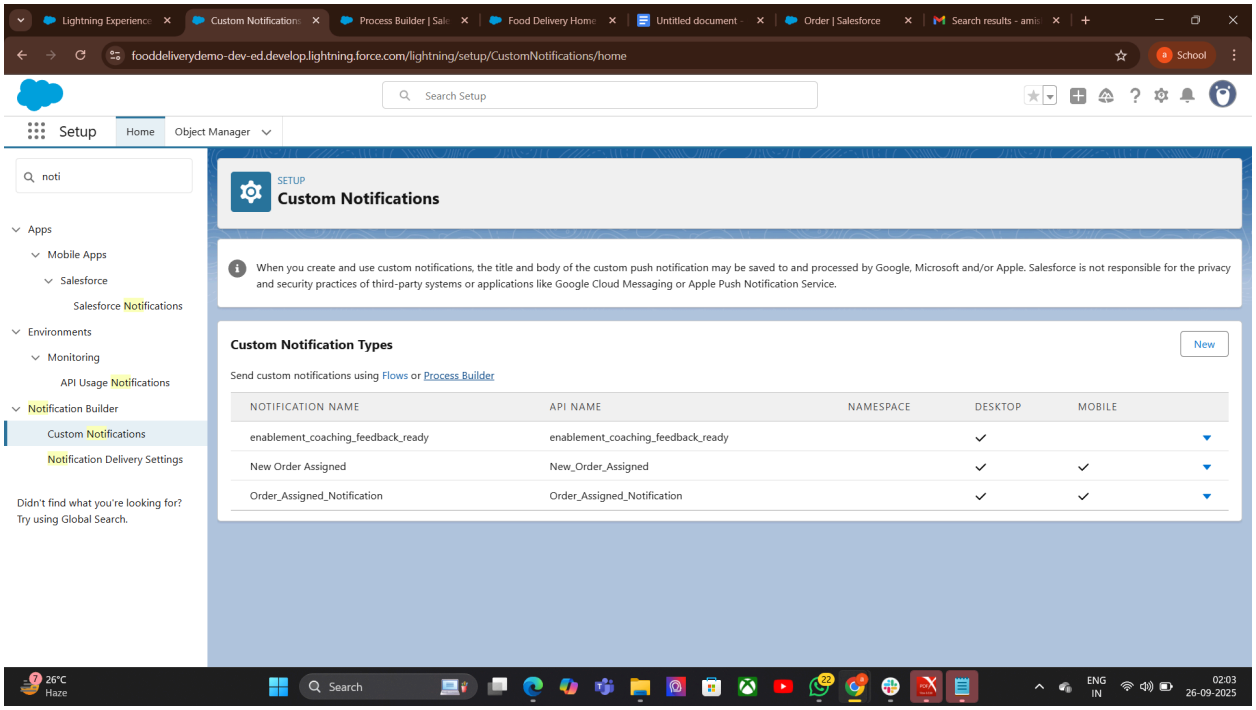


8.Custom Notifications:

How to Create a Custom Notification

- Create a Custom Notification Type
- Go to Setup → Custom Notifications.
- Click New.
- Enter name, API name, description.
- Choose supported channels (Desktop, Mobile, In-App)

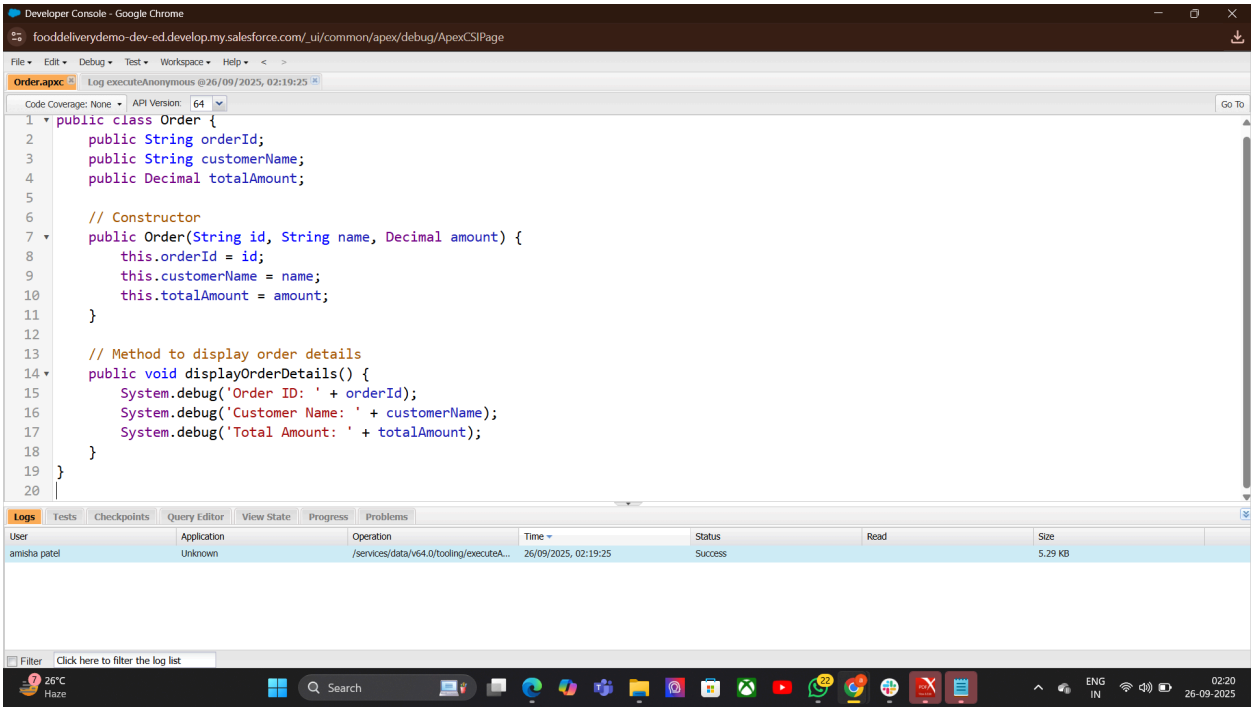
  Save.


Lightning Experience
Custom Notifications
Process Builder | Sale
Food Delivery Home
Untitled document -
Order | Salesforce
Search results - amis
fooddeliverydemo-dev-ed.develop.lightning.force.com/lightning/setup/CustomNotifications/home
School
Search Setup
Setup
Home
Object Manager
noti
Apps
Mobile Apps
Salesforce
Salesforce Notifications
Environments
Monitoring
API Usage Notifications
Notification Builder
Custom Notifications
Notification Delivery Settings
Didn't find what you're looking for?
Try using Global Search.
SETUP
Custom Notifications
When you create and use custom notifications, the title and body of the custom push notification may be saved to and processed by Google, Microsoft and/or Apple. Salesforce is not responsible for the privacy and security practices of third-party systems or applications like Google Cloud Messaging or Apple Push Notification Service.
Custom Notification Types
New
Send custom notifications using Flows or Process Builder
NOTIFICATION NAME
API NAME
NAMESPACE
DESKTOP
MOBILE
enablement_coaching_feedback_ready
enablement_coaching_feedback_ready
New Order Assigned
New_Order_Assigned
Order_Assigned_Notification
Order_Assigned_Notification
26°C
Haze
Search
ENG
IN
02:03
26-09-2025

Phase 5:Apex Programming(Developer)

1 Classes & Objects:Steps to Create an Apex Class File

1. Open Salesforce Developer Console
    - Click your avatar → Developer Console.
2. Create New Apex Class
    - Go to File → New → Apex Class.
    - Give it a name (e.g., OrderHandler).
3. Define the Class
    - Start with public class ClassName { }.
4. Add Properties (Variables)
    - Example: orderId, customerName, totalAmount.
5. Add Constructor
6. Add Methods
7. Save the Class
8. Test the Class
9. **Check Logs** -Go to **Debug → Logs** to see the output.



2.Apex Triggers (before/after insert/update/delete)

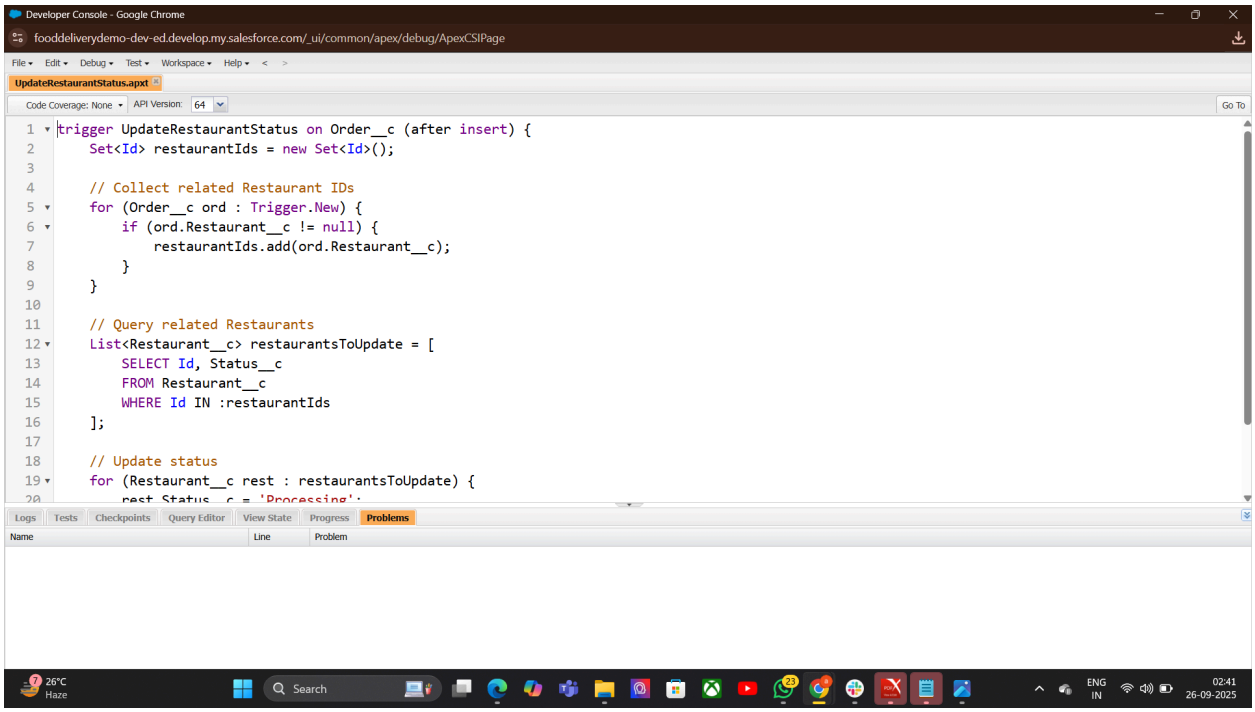
Definition: Pieces of Apex code that execute before or after records are inserted, updated, deleted, or undeleted.

Types:

Before Triggers: Modify data before saving (e.g., validation).
After Triggers: Access system fields or related records after saving.
Example: Automatically update a related record when an Order is inserted.

Explanation:

- Runs **after insert** of Order records.
- Collects related restaurant IDs.
- Updates the status of each related restaurant to `"Processing"`.



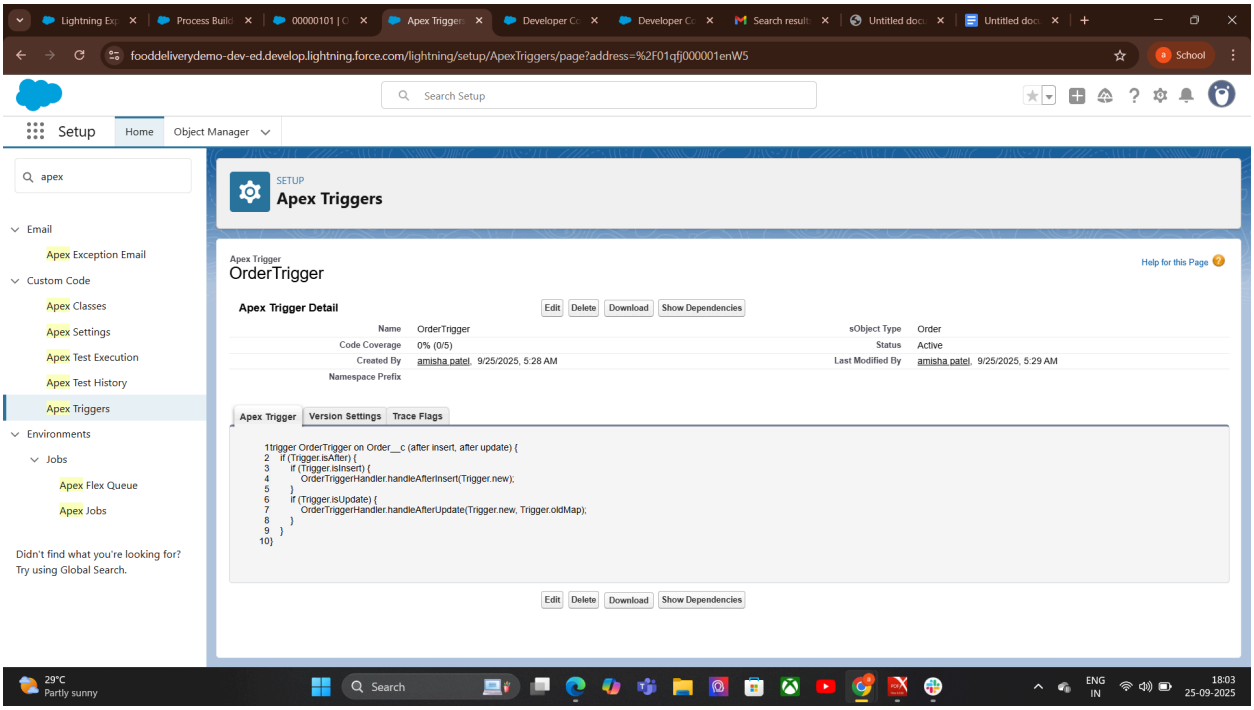
3. Trigger Design Pattern:

Purpose: Organize trigger logic cleanly and efficiently.

Trigger Syntax-

trigger TriggerName on ObjectName (events) {
// Trigger logic
}

TriggerName → custom name of trigger
ObjectName → object
events → insert, update, delete, undelete, after insert, before insert, etc.

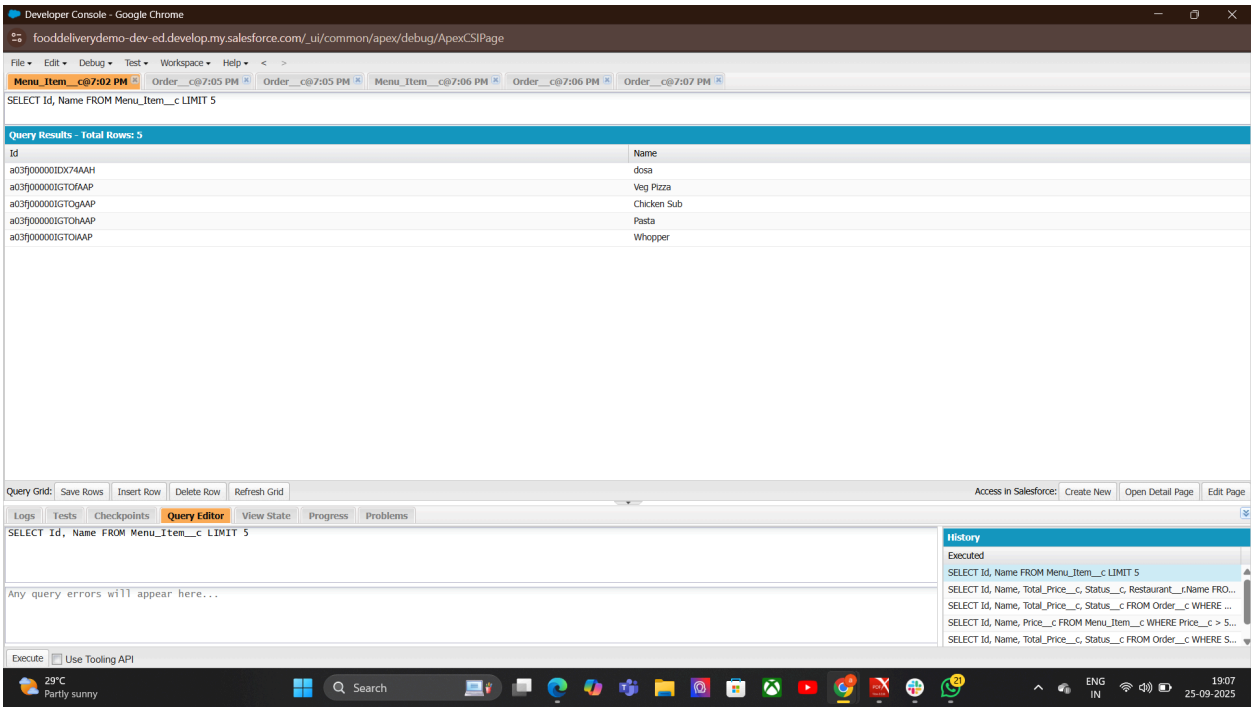


4.SOQL & SOSL:

SOQL (Salesforce Object Query Language)

Definition: Salesforce's query language that retrieves data from a single object.

- Purpose: Structured data retrieval with filtering and ordering.
- **Usage:** When you need specific records based on conditions.
- **Syntax:-SELECT fieldList FROM objectName WHERE condition.**



5.Collections:List,Set,Map:

List

Definition: Ordered collection of elements.

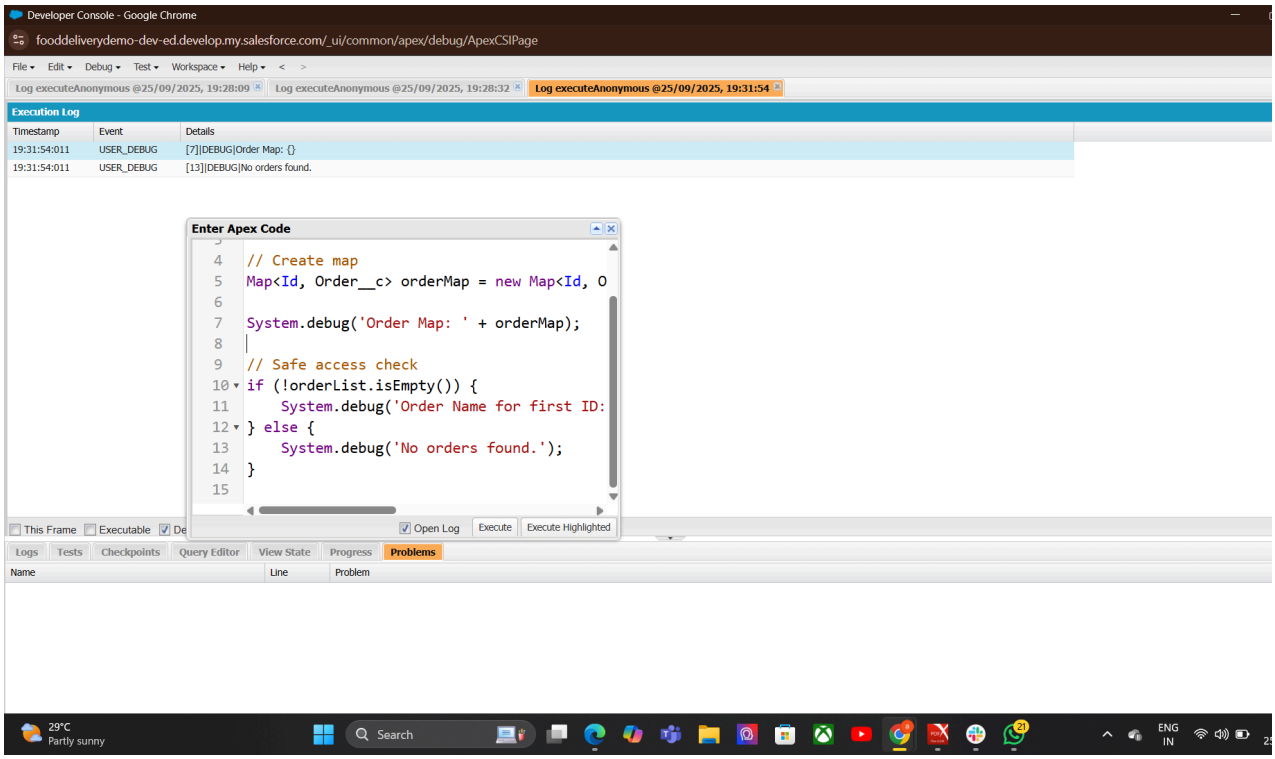
Purpose: Store records in a sequence.
Syntax:List<DataType> listName = new List<DataType>();

Set

Definition: Unordered collection of unique elements.
Purpose: Store distinct values without duplicates.
Syntax:Set<DataType> setName =new Set<DataType>();

Map

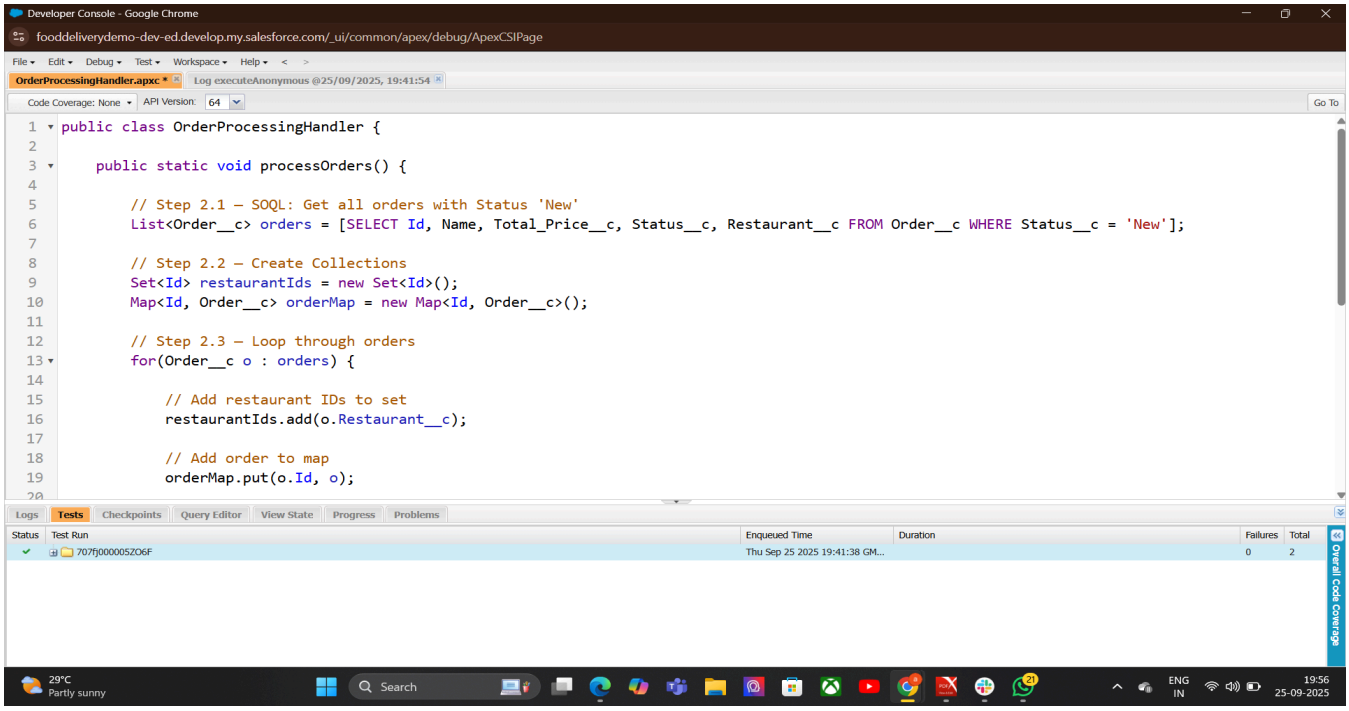
Definition: Collection of key-value pairs.
Purpose: Store and access data by a unique key.
Syntax:Map<KeyType, ValueType> mapName = new Map<KeyType,valueType();



5.Control Statements:

Control Statements direct the flow of program execution. They include:

- **Conditional statements** (`if`, `if...else`, `else if`, `switch`) → decide which code runs based on conditions.
- **Loop statements** (`for`, `while`, `do...while`, `for-each`) → repeat code until a condition is met.
- **Jump statements** (`break`, `continue`) → change loop or switch flow.

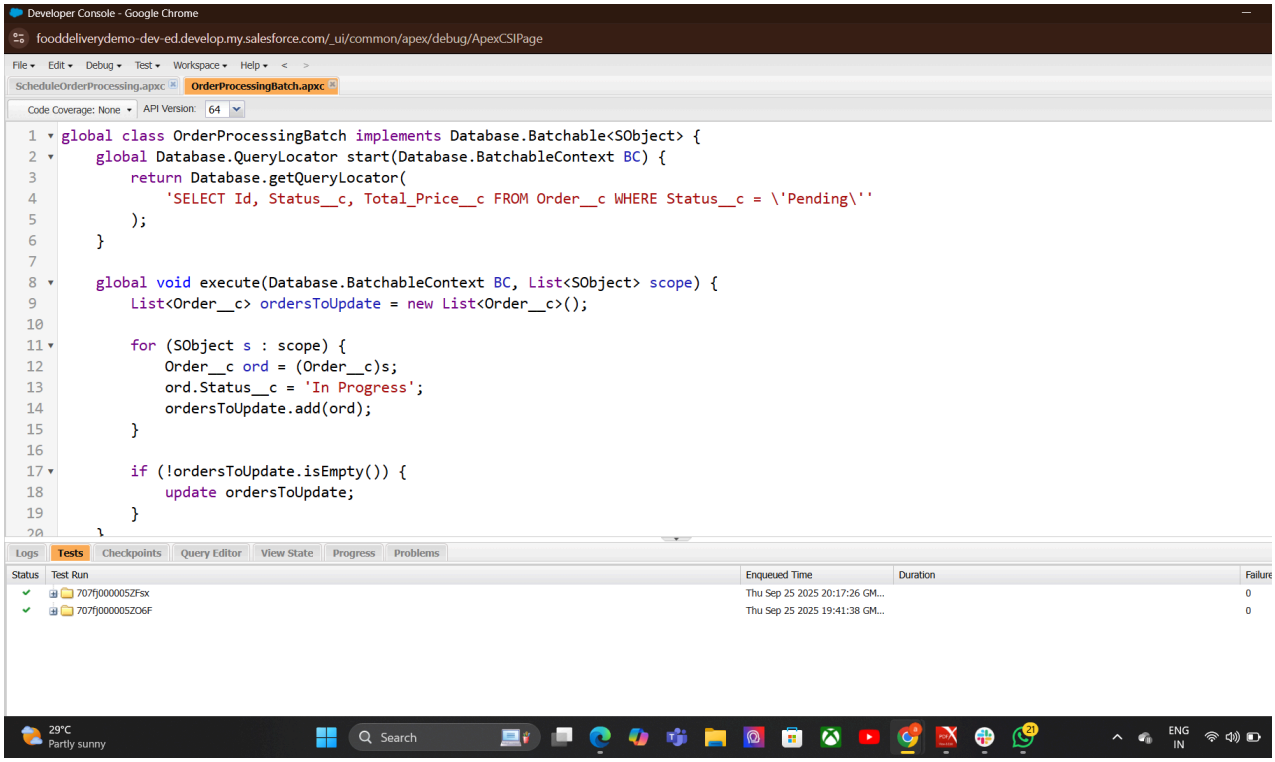


7. Batch Apex:

Basic Structure

A Batch Apex class has three main methods:

start() → Prepares the query or collection to process.
execute() → Processes each batch of records.
finish() → Final operations after all batches are processed.



8.Queueable Apex:

Queueable Apex is used in Salesforce to run asynchronous jobs that allow complex processing and chaining of jobs with flexible execution. It's like a more advanced version of Future Methods.

Purpose → Run long-running or resource-intensive tasks in the background.

Execution → Asynchronous, queued for execution when system resources allow.

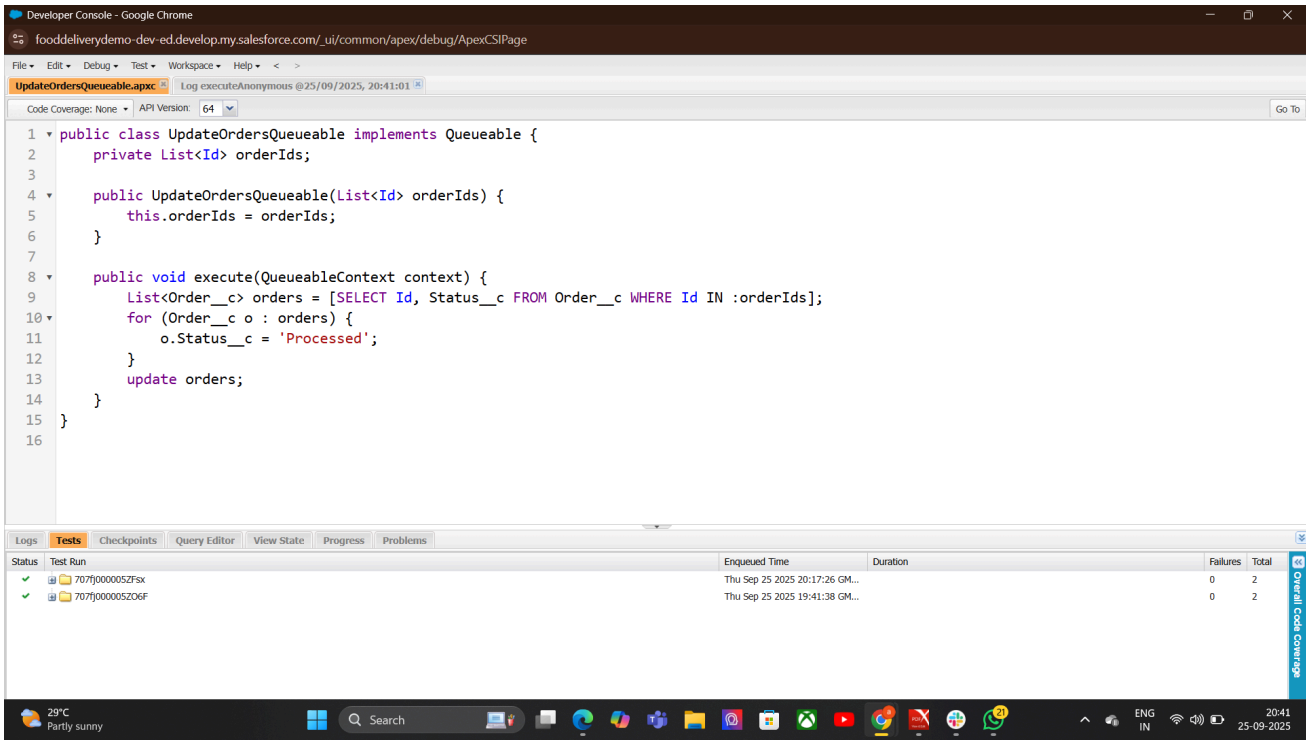
CODE–

```
public class UpdateOrdersQueueable implements Queueable {
  private List<Id> orderIds;

  public UpdateOrdersQueueable(List<Id> orderIds) {
    this.orderIds = orderIds;
  }

  public void execute(QueueableContext context) {
    List<Order__c> orders = [SELECT Id, Status__c FROM Order__c WHERE Id IN :orderIds];
    for (Order__c o : orders) {
      o.Status__c = 'Processed';
    }
    update orders;
```

```
  }
}
```

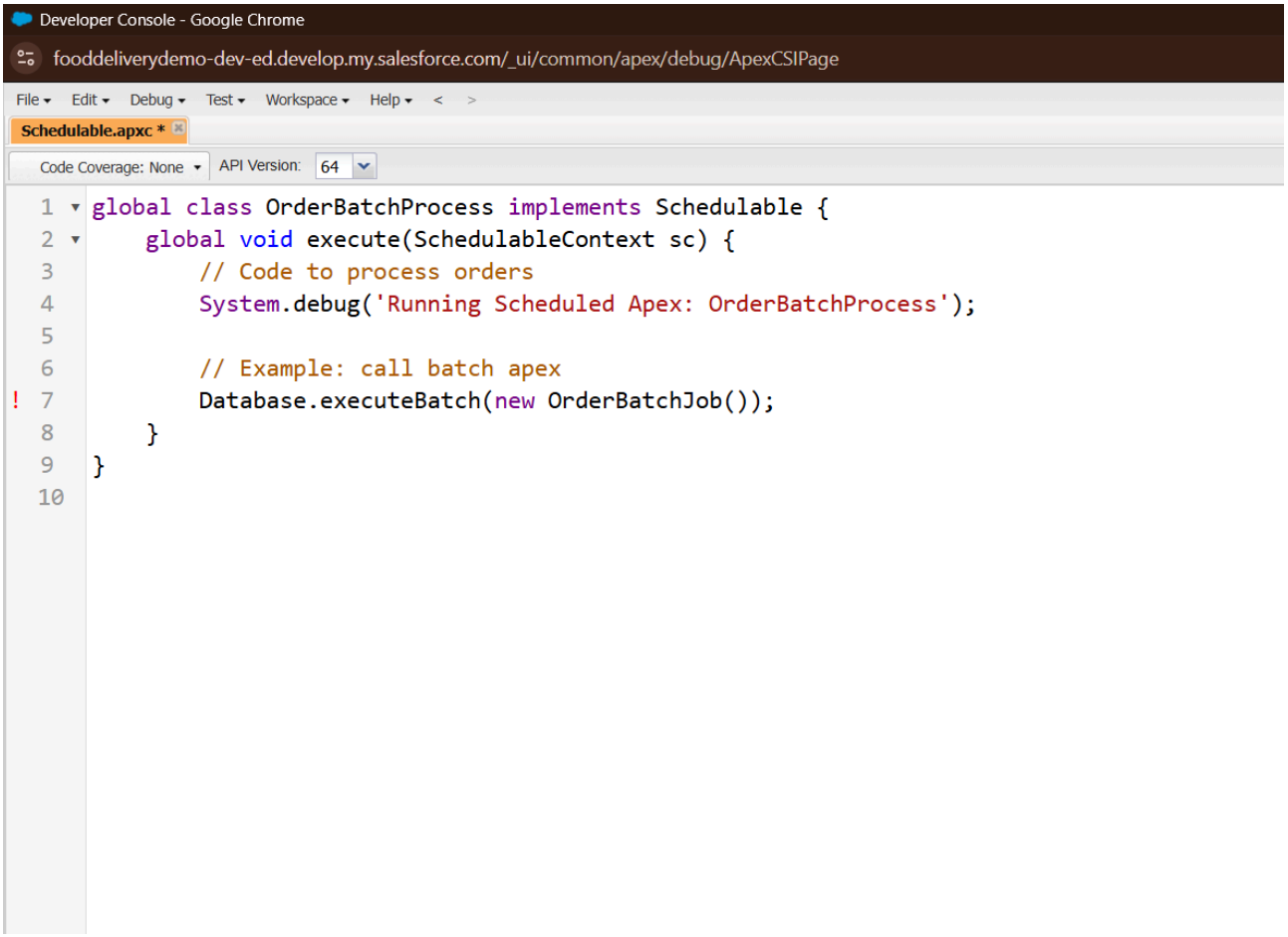


9.Scheduled Apex:

Scheduled Apex allows you to run Apex classes at specific times or intervals automatically. It's useful for jobs that need to run periodically without manual intervention.

Key Points

- Purpose → Automate processes at scheduled times.
- Execution → Runs asynchronously in background at specified schedule.
- Interfaces Used → Implements `Schedulable` interface.
- Flexibility → Can schedule daily, weekly, monthly, or custom times.

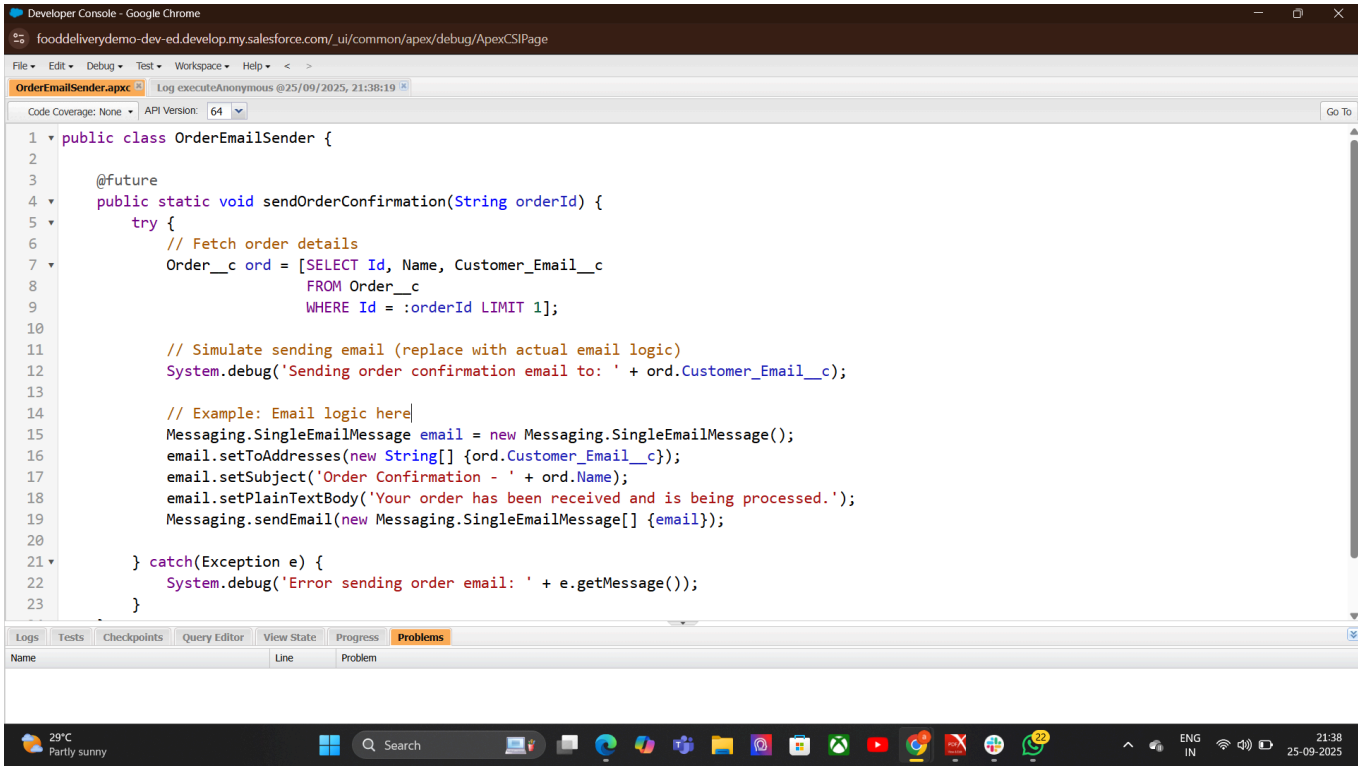


10. Future Methods:

Future methods are asynchronous Apex methods that run in the background at a later time. They are useful for tasks that don't need to run immediately, such as external callouts or bulk processing.

Class Name

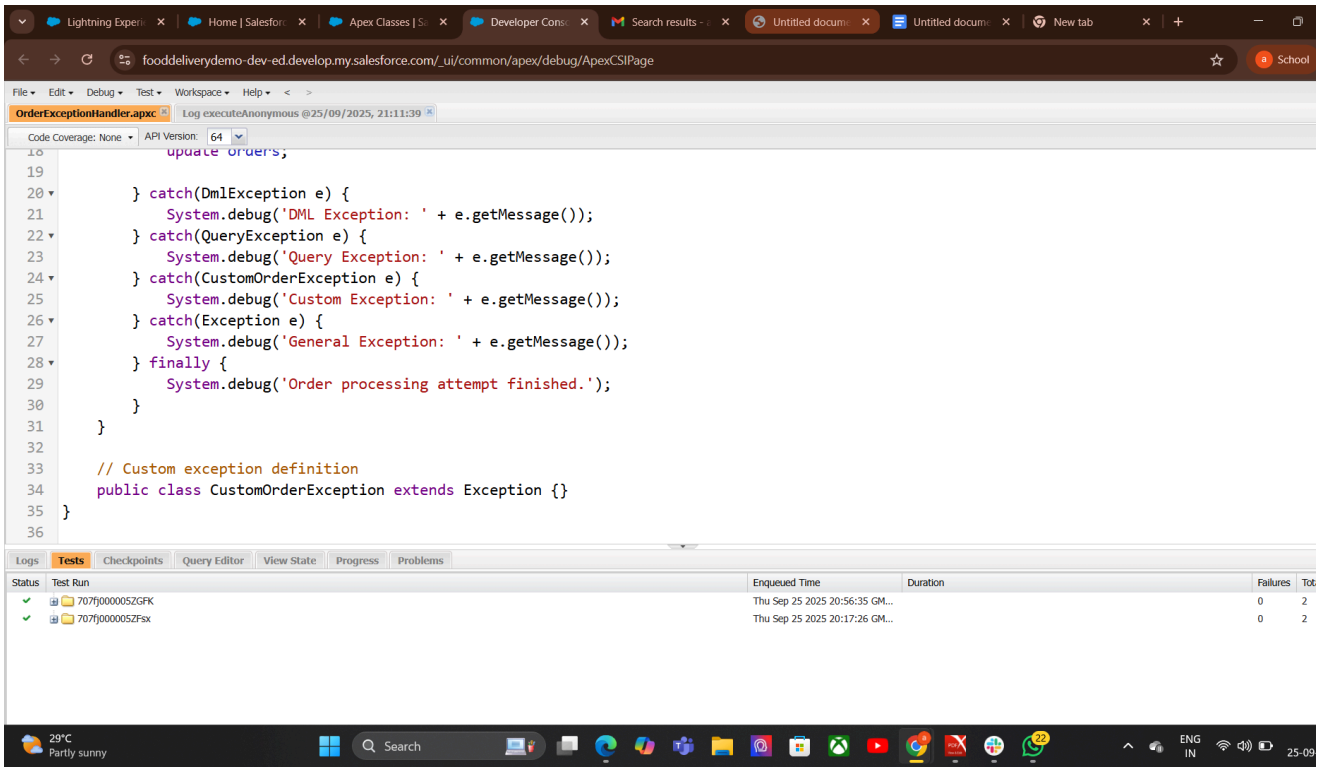
`OrderEmailSender` → A helper class to send order confirmation emails asynchronously.



11. Exception Handling:

Key Points

- Purpose → Handle runtime errors and maintain program stability.
- Types of Exceptions →
    - System Exceptions → Built-in errors (e.g., `NullPointerException`, `DmlException`).
    - Custom Exceptions → User-defined errors for special scenarios.
      Keywords Used → `try`, `catch`, `finally`, `throw`.



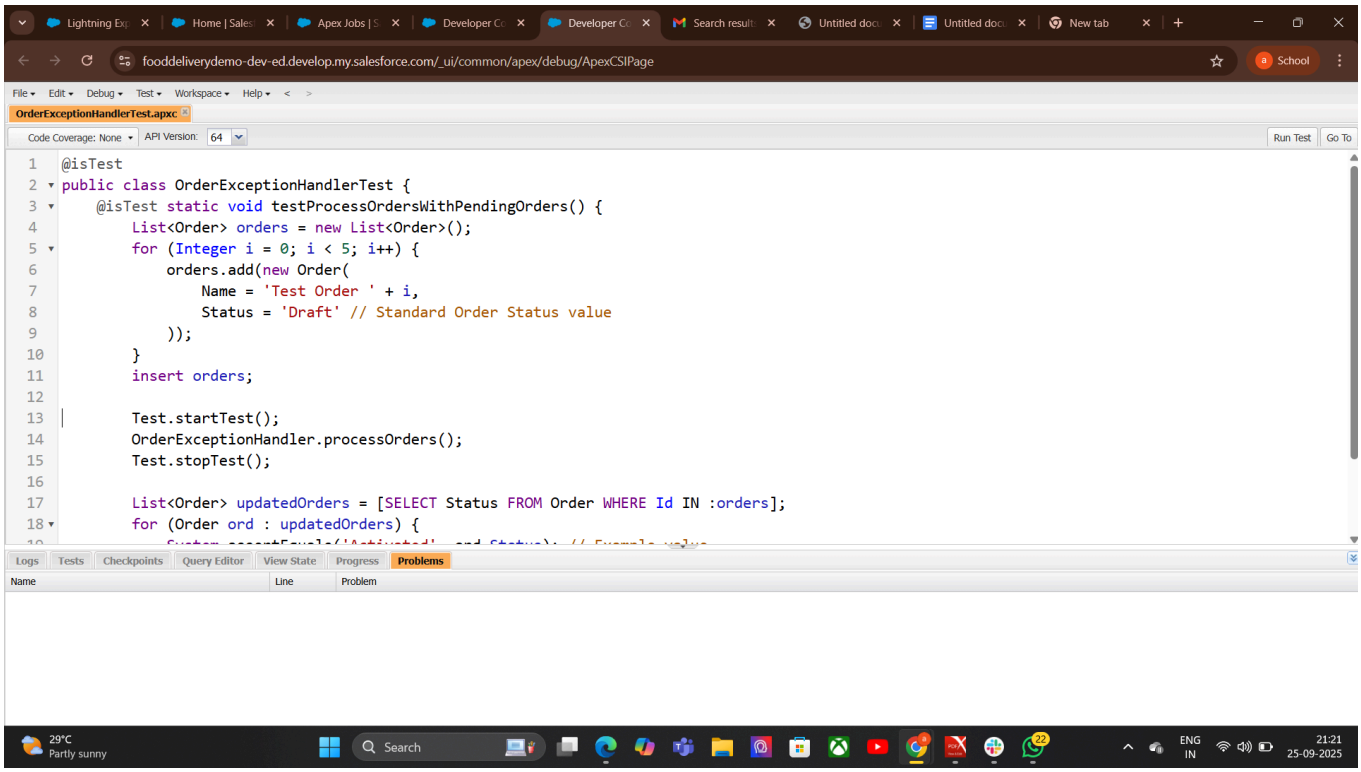
12.Test Classes:

- Creates Test OrdersIt creates a list of 5 orders with the status "Draft".

    - These are inserted into the database.

- Runs the Method Under Test
    - I tcalls OrderExceptionHandler.processOrders() inside Test.startTest() and Test.stopTest().
    - This ensures testing is done in a fresh context and respects governor limits.

- Verifies the Results
  It queries the orders back from the database.
  It checks that each order's status is now "Activated" using System.assertEquals.



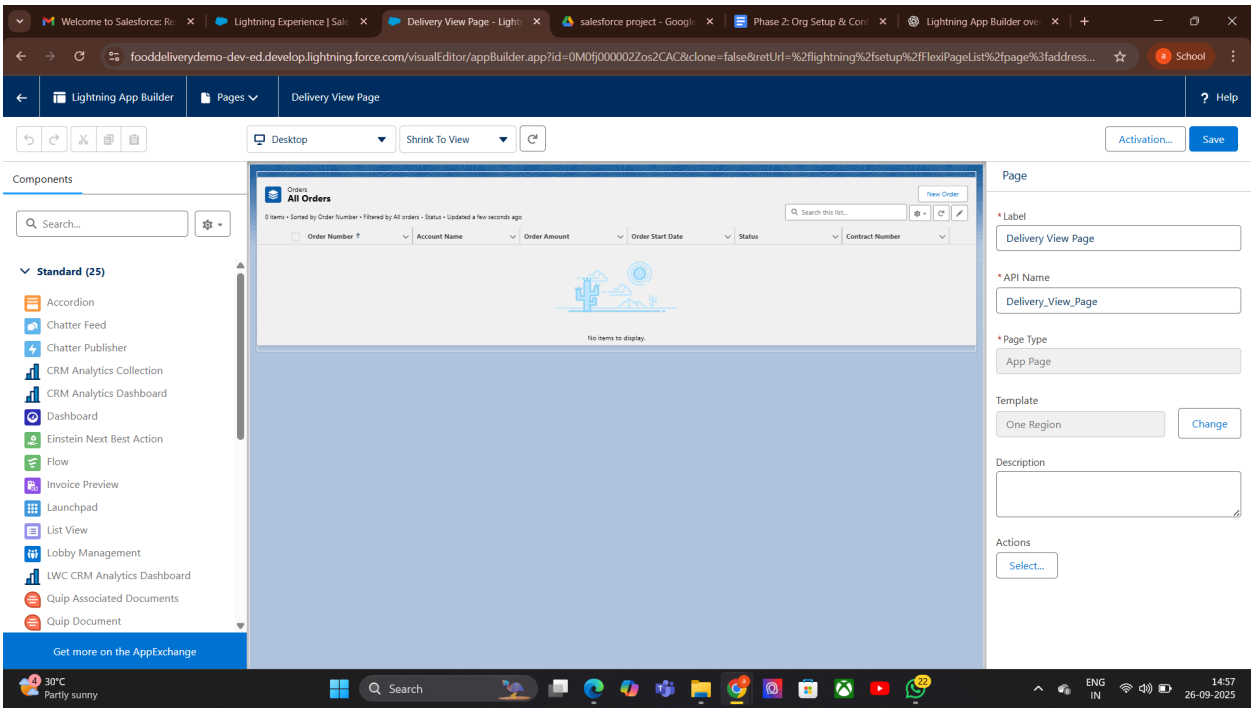
Phase 6: User Interface Development

1.Lightning App Builder:

Lightning App Builder is a point-and-click tool in Salesforce that lets you create custom pages for the Lightning Experience and mobile app without writing code.

Steps to Use Lightning App Builder

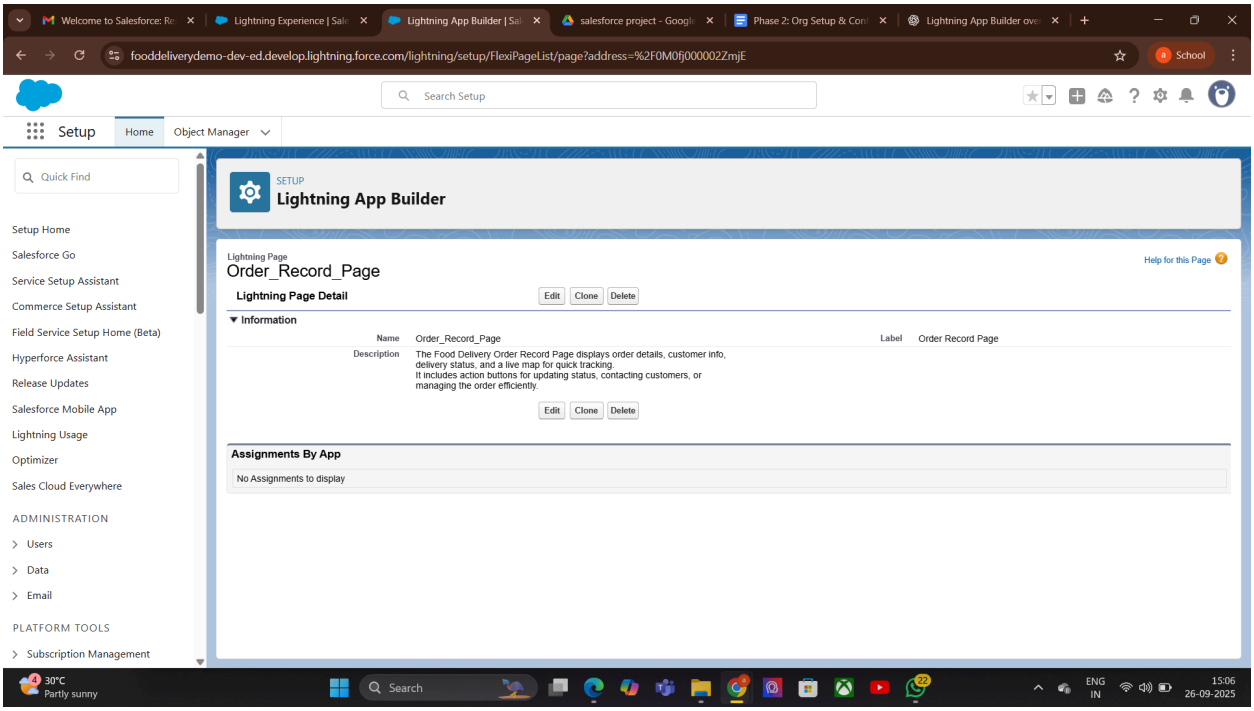
- Go to Setup → Search for *Lightning App Builder*.
- Click New.
- Select page type (App Page, Home Page, Record Page, etc.).
- Choose a template layout.
- Drag and drop components onto the canvas.
- Configure component properties.
- Save and activate the page (set for org, app, or profile-specific visibility).



2. Record Pages:

How to Create a Record Page:

- Go to Setup → Lightning App Builder.
- Click New → Record Page.
- Select object (e.g., Order__c).
- Name it (e.g., “= record Order Page”).
- Choose a layout template.
- Drag & drop standard/custom components.
- Save and Activate → Choose activation rules (Org Default, App Default, or Profile-specific).

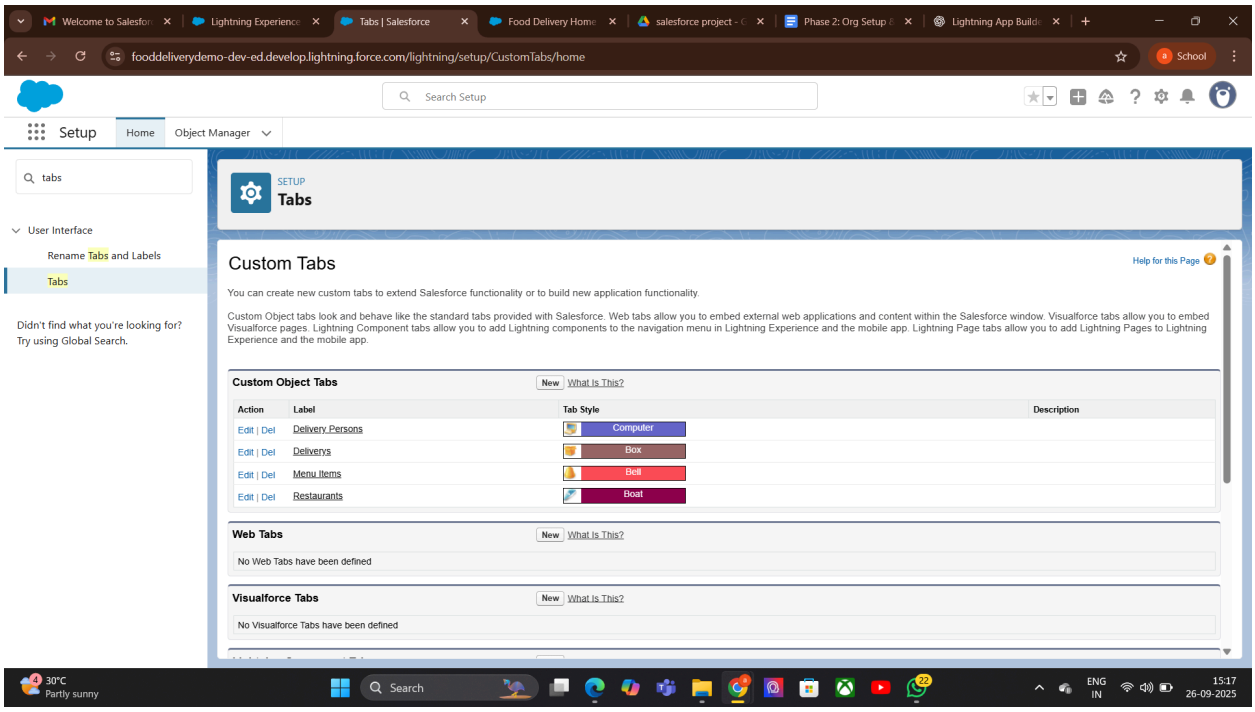


3. Tabs:

Tabs in Lightning App Builder are sections within an app or page that organize content into separate, easy-to-access areas.

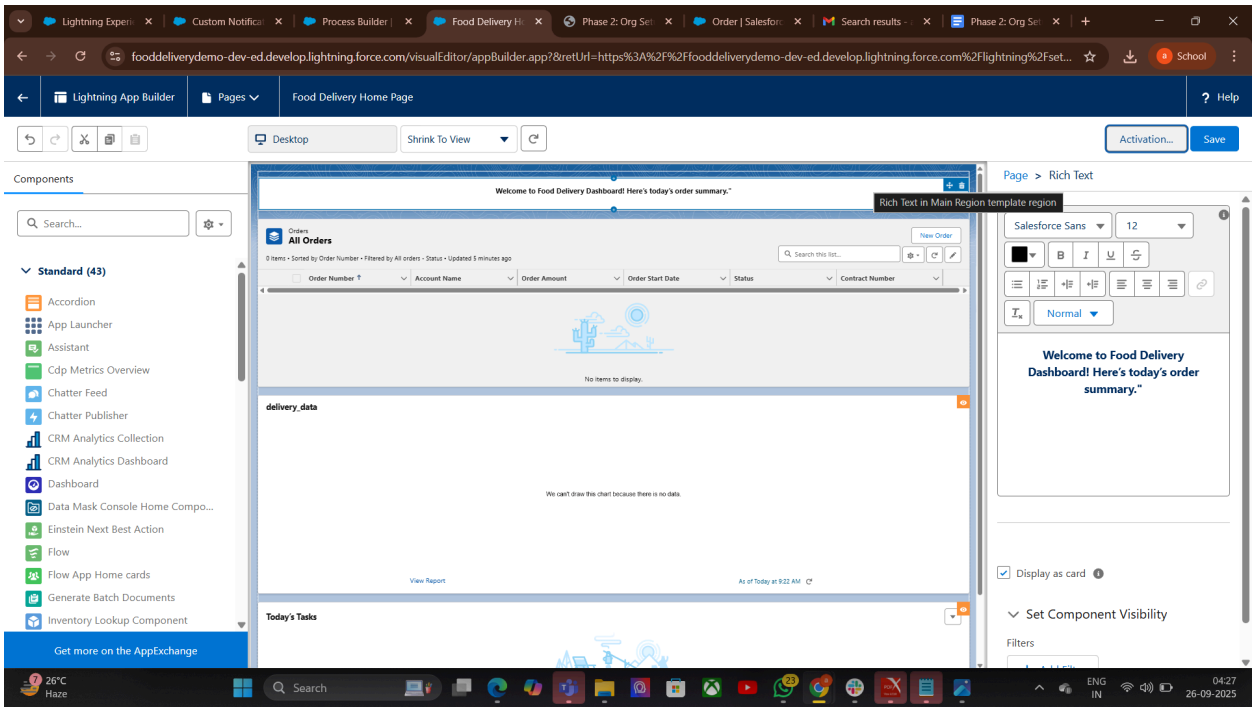
Tabs could be:

- **Delivery Persons** – People who deliver orders.
- **Deliveries** – Records of orders and their status.
- **Menu Items** – Food items offered by restaurants.
- **Restaurants** – Places that prepare and sell food.



4. Home Page Layouts:

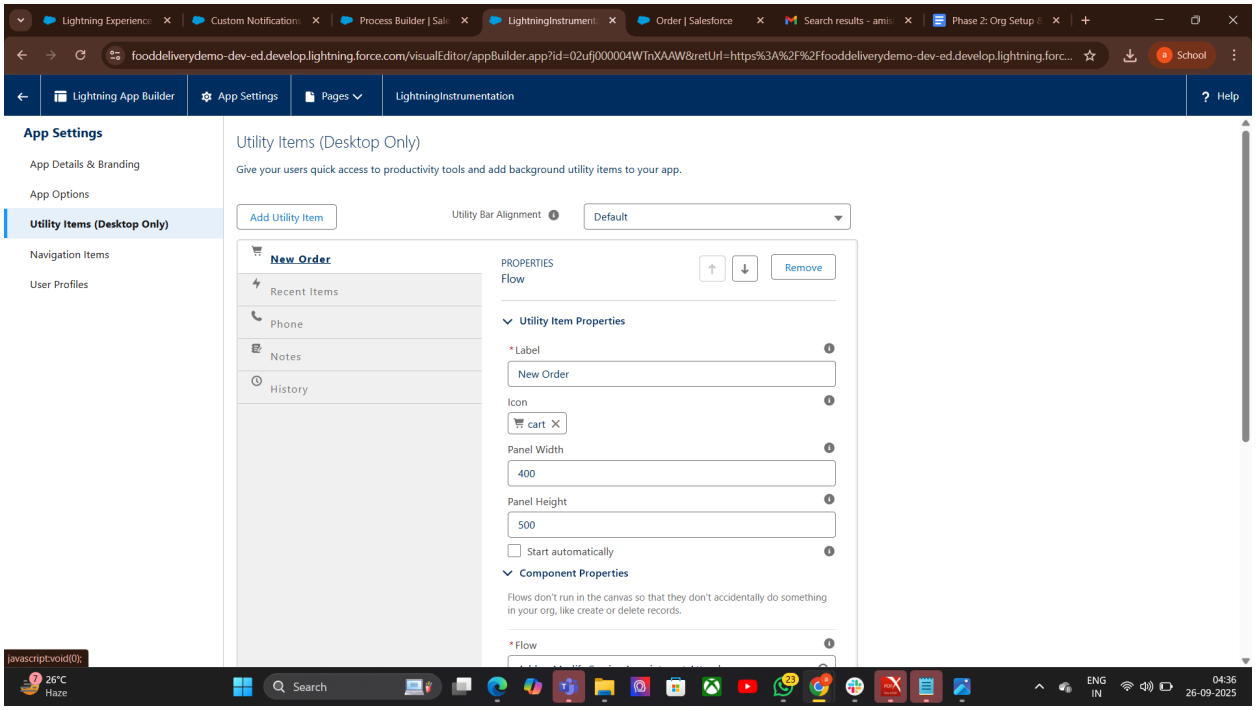
- Definition: Custom layouts for the Salesforce Home Page that organize components and data for users.
- Purpose: Provide quick access to important information, reports, and tools.
- Examples: Dashboards, recent records, tasks, notifications.
- Usage: Customize for roles, profiles, or apps to improve productivity.
- **Go to Setup** → Search **"Home Page Layouts"**.
- **Click "New"** to create a new layout.
- **Enter Layout Name** and description.
- **Select Components** to add (e.g., dashboards, reports, tasks, news, recent records).
- **Arrange Components** in desired order.
- **Save Layout**.



5.Utility Bar:

The Utility Bar is a customizable toolbar in Lightning Experience that appears at the bottom of the app. It gives users quick access to frequently used tools without leaving their current page.

- Go to Setup → App Manager.
- Select the Lightning App → Click Edit.
- Go to Utility Bar.
- Click Add Utility and choose the component/tool.
- Configure label, icon, and behavior.
- Save and activate.



7. LWC (Lightning Web Components):(not implemented)

LWC is Salesforce's modern JavaScript-based framework that uses web standards (HTML, CSS, JavaScript).

It provides a lightweight, high-performance way to build reusable UI components.

8.Apex with LWC:

- Definition: Apex is the server-side language in Salesforce that processes business logic and database operations, while LWC (Lightning Web Component) is the client-side UI framework. Together, they create a dynamic and responsive Food Delivery app.
- Purpose in Food Delivery:
    - LWC displays menus, orders, and delivery status to users.
    - Apex fetches menu data, creates orders, updates delivery status, and manages backend logic.
- How it works:

    - LWC calls Apex methods using `@wire` or imperative calls.
    - Apex processes the request, interacts with Salesforce database, and returns data.
    - LWC updates the UI based on Apex response.
- Example Use:
    - Show restaurant menu.
    - Place new order.
    - Track delivery status.

9.Events in LWC:

Definition: Events in Lightning Web Components (LWC) are custom or standard signals used to communicate between components or to trigger actions.

Purpose: To allow components to communicate in a structured way without tight coupling.

Types of Events in LWC

1. Standard DOM Events
    - Built-in events like `click`, `change`, `input`.
    - Example: `<button onclick={handleClick}>Click Me</button>`.
2. Custom Events
    - Used to send data between components.
    - Created using `CustomEvent` in JavaScript.
    - Can carry data in the `detail` property.

10. Imperative Apex Calls:

Definition: Imperative Apex calls in LWC are manual calls to Apex methods (not automatic like @wire) made when certain actions occur, such as button clicks.

Purpose: Gives more control over when Apex methods run and allows passing dynamic parameters.

```
public with sharing class FoodDeliveryController {

 @AuraEnabled

   public static List<Menu_Item__c> getMenuItems(Id restaurantId) {

      return [

         SELECT Id, Name, Price__c

         FROM Menu_Item__c

         WHERE Restaurant__c = :restaurantId

      ];

   }

}
```

Key Points:

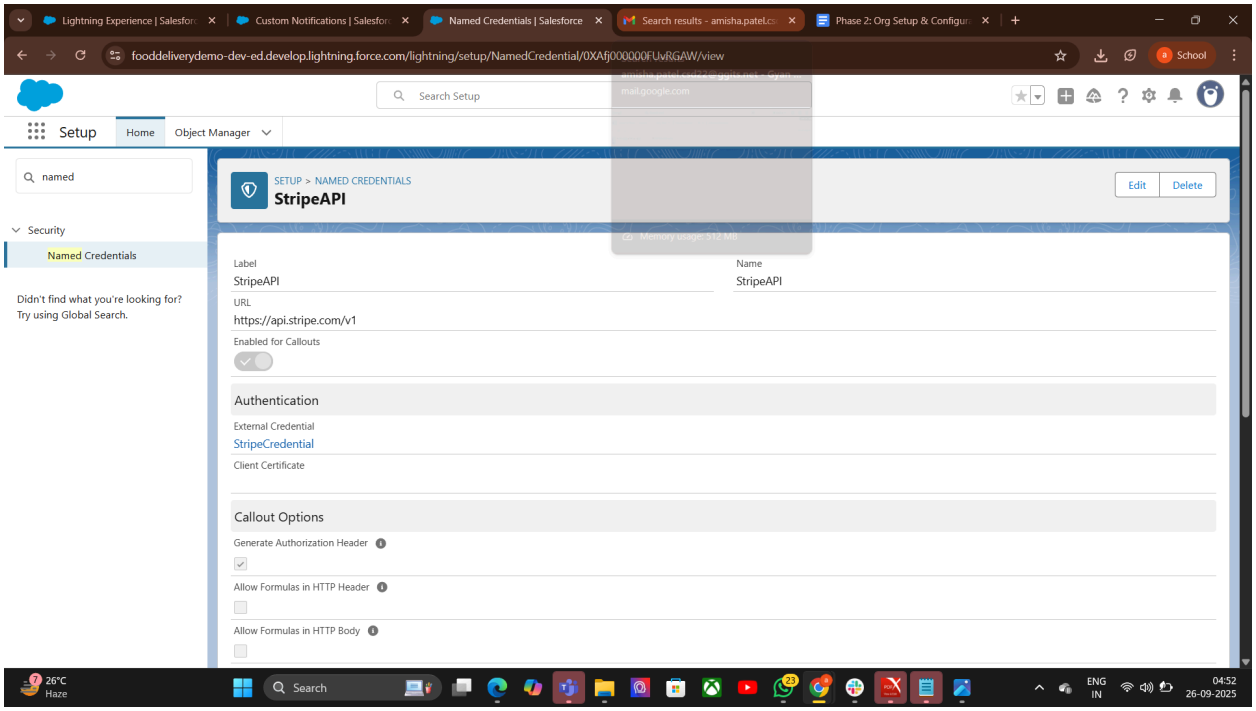
- Must import Apex method in LWC JavaScript.
- Apex method must be `@AuraEnabled`.
- Called inside a JavaScript function, usually in response to user actions.
- Returns a Promise, so `.then()` and `.catch()` are used for handling results and errors.

7.Integration & External Access:

1.Named Credentials

Purpose: Securely store API URLs and login details, simplify integrations.
Security → Credentials are stored securely by Salesforce, not visible in plain text.

Easy Callouts → Developers can call APIs using a simple reference (callout:Name).
Authentication Types → Supports Password, OAuth 2.0, JWT, AWS Signature, etc.
Identity Types →
Named Principal → One login for all users.
Per User → Each user has their own login.
Usage in Apex/Flow → Can be used directly in Apex callouts, Flows, and External Services.



2.External Services

Purpose

Connect Salesforce with **external REST APIs** easily.
Allow **admins (non-developers)** to use external APIs in **Flow or Process Builder**.

Pepare API Schema → Get an **OpenAPI (Swagger) definition** of the external REST API.

Set Up Named Credential → Store API base URL + authentication securely.

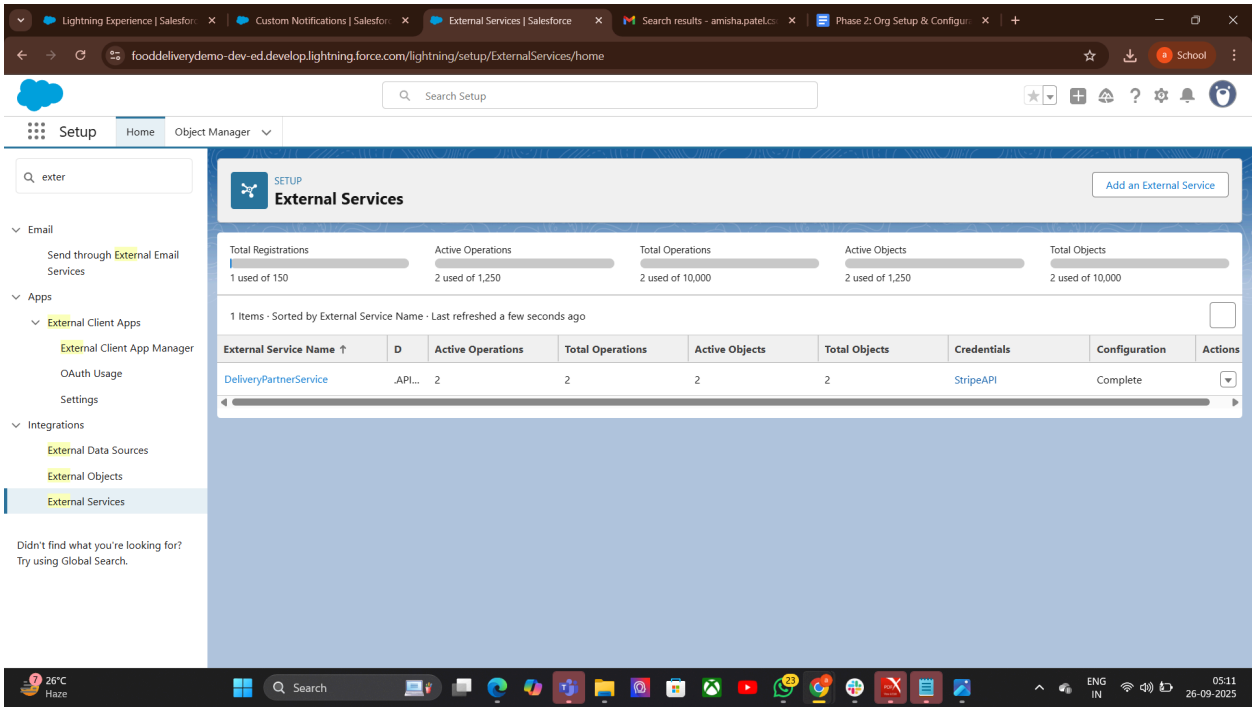
Register External Service →

- Go to Setup → **External Services** → New.
- Upload/OpenAPI schema.
- Link it with the **Named Credential**.

Generate Actions → Salesforce auto-creates invocable actions from the schema

Use in Flow/Apex →

- In Flow, drag **Action** → search API action → use it like a standard Salesforce action.
- In Apex, invoke via Named Credential callout if needed.
  Test Integration → Run Flow/Process to confirm external API is working



3.• Callouts

Purpose

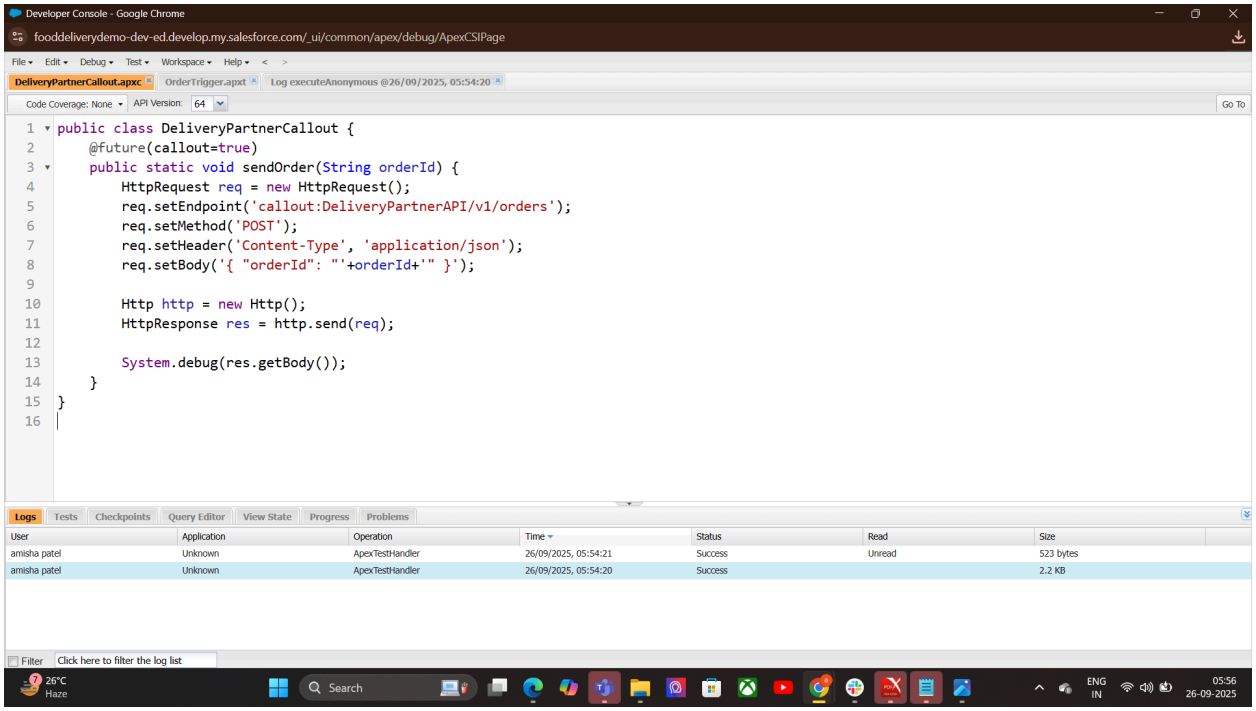
- Allow Salesforce to send requests to external systems (APIs/web services).
- Fetch or send real-time data (e.g., payment status, shipping updates).

- Enable integration between Salesforce and third-party applications.

  Callouts – Process

  Whitelist Endpoint → Add API URL in Remote Site Settings or use Named Credentials.
  Create HTTP Request → Define endpoint, method (GET/POST), headers, body.

  Send Callout → Use Http class in Apex (or Flow with External Services).
  Get Response → Parse JSON/XML result.
  Update Salesforce Data → Save response in records/fields.



4.Platform Events:

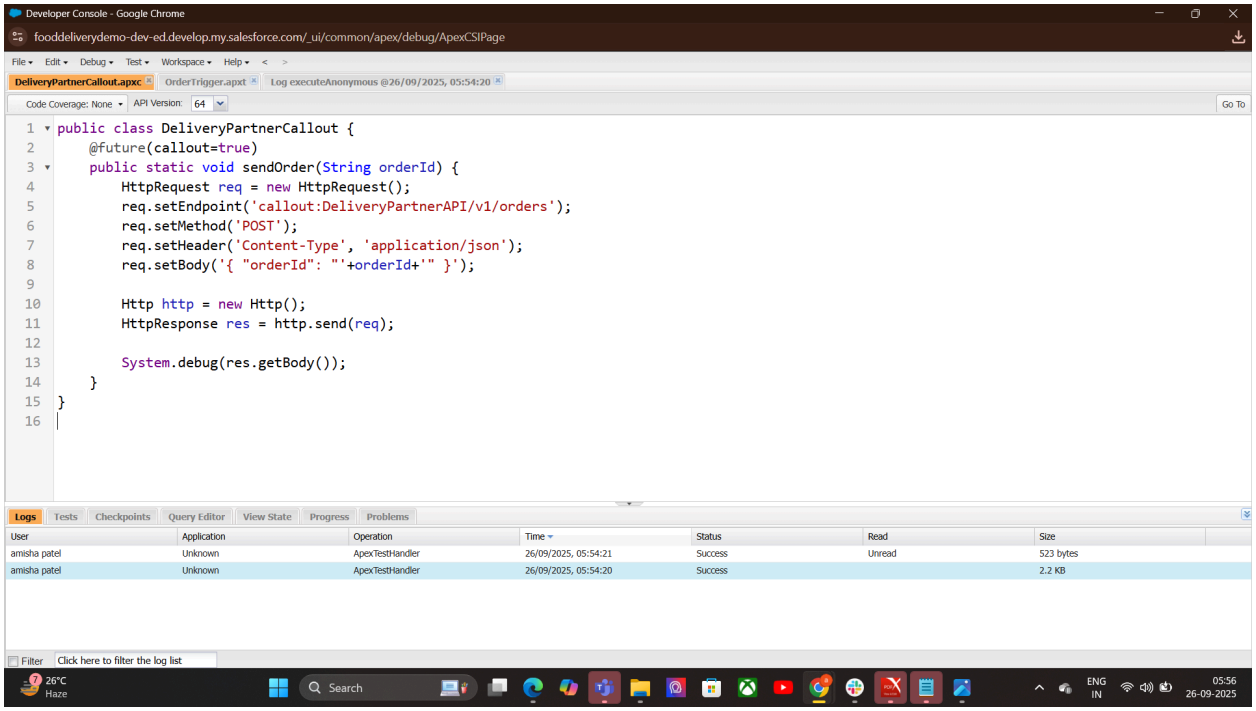
Purpose

- Enable real-time communication between Salesforce and external systems.
- Work on publish–subscribe model (like messaging queues)

Process-Create Platform Event → In Setup, define fields (like OrderId, Status).

Publish Event → From Apex, Flow, or API when something happens.
Subscribe to Event →

- Inside Salesforce (Triggers, Flows).
- Outside Salesforce (via CometD API, Middleware).
- React in Real-Time → Update records, notify users, or call external apps.



5. Change Data Capture:

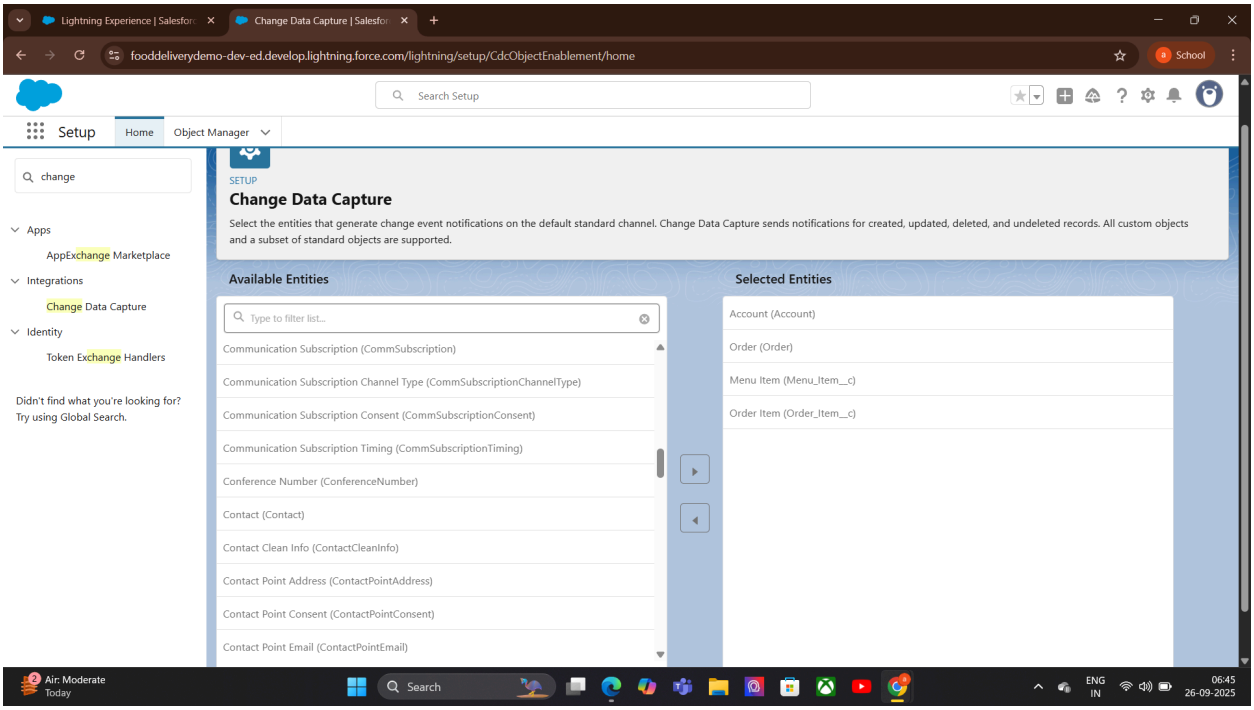
Purpose

- Automatically broadcast changes in Salesforce records (Insert, Update, Delete, Undelete).
- Keep external systems in sync without polling.

Process

1. Enable CDC → Go to Setup → Change Data Capture → Select Objects.
2. Publish Events Automatically → Salesforce generates events when records change
3. Subscribe to Events → External system subscribes via CometD API or Salesforce Streaming API.

4. Process Changes → Update external system or take actions in Salesforce.



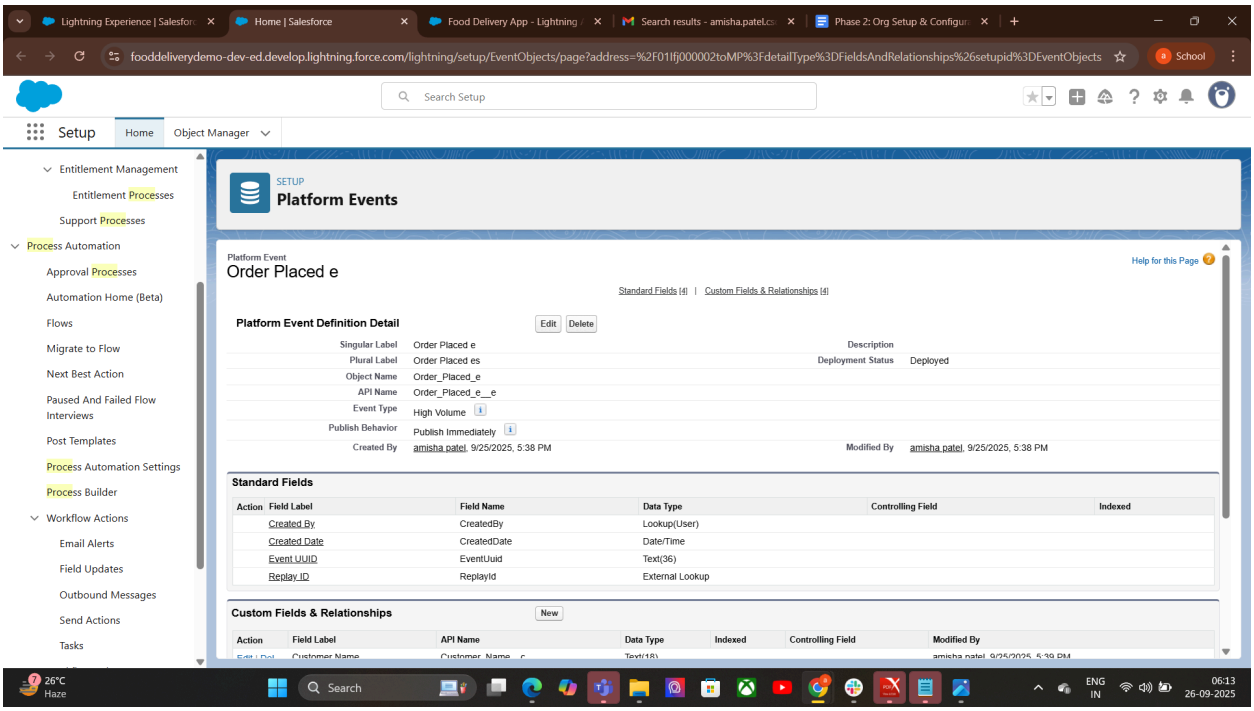
6.Platform Events:

Purpose

- Enable real-time communication between Salesforce and other systems.
- Automate processes using event-driven architecture.
- Keep systems loosely coupled and scalable.

Process

1. Create Platform Event → Define event fields in Setup.
2. Publish Event → From Apex, Flow, Process Builder, or API.
3. Subscribe to Event →
    - Apex Trigger (internal).
    - Flow.
    - External system via CometD API.
4. Process Event Data → Update records, trigger workflows, or call APIs



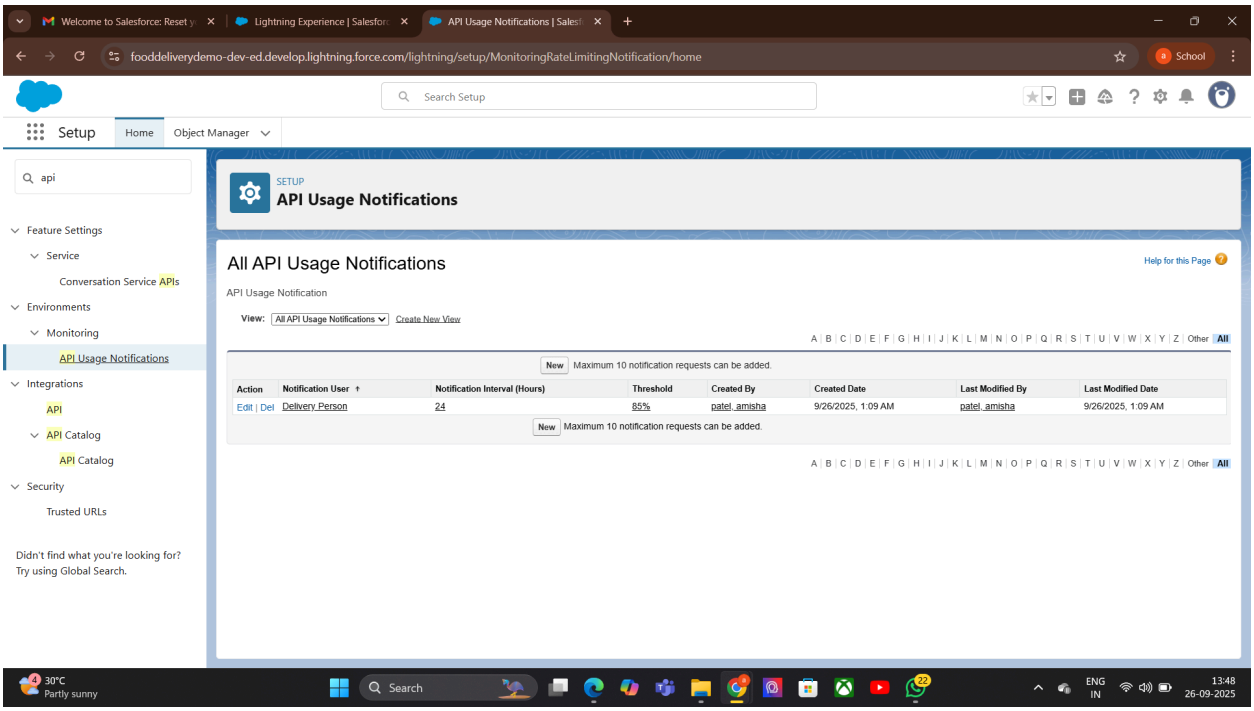
7.API Limits:

Purpose

- Protect Salesforce performance and stability.
- Prevent excessive API usage that could overload the system.
- Ensure fair usage across users and integrations.

Steps (Process)

1. Understand Your Limit → Check API usage in Setup → System Overview or API Usage.
2. Track API Calls → Every REST, SOAP, Bulk API, and streaming event counts.
3. Optimize Calls → Combine requests, use Bulk API, or cache results to reduce calls.
4. Monitor Usage → Use reports or API Usage dashboards to avoid hitting limits.
5. Plan for Scaling → Upgrade edition or request extra API calls if needed.



8.OAuth & Authentication:

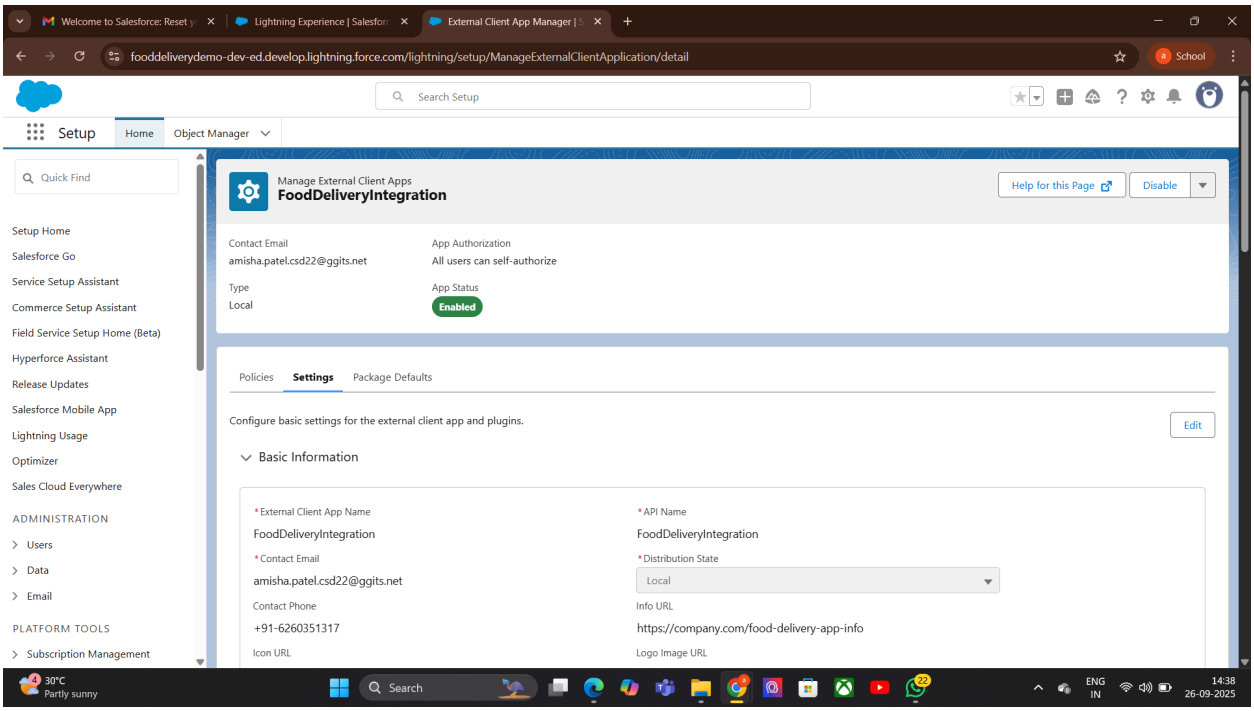
Purpose

- Securely connect external applications with Salesforce without sharing passwords.
- Control access using tokens instead of credentials.
- Support multiple authentication flows for different scenarios.

Steps (Process)

1. Choose Authentication Flow → e.g., Web Server Flow, JWT, Username-Password Flow.
2. Create Connected App → In Salesforce Setup → specify app details, callback URL, scopes.
3. Get Client Credentials → Client ID and Client Secret from Connected App.
4. Request Access Token → External app authenticates and receives token from Salesforce.
5. Use Token in API Calls → Send token in HTTP headers for authentication.

6. Refresh Token When Needed → Tokens expire, refresh without re-login



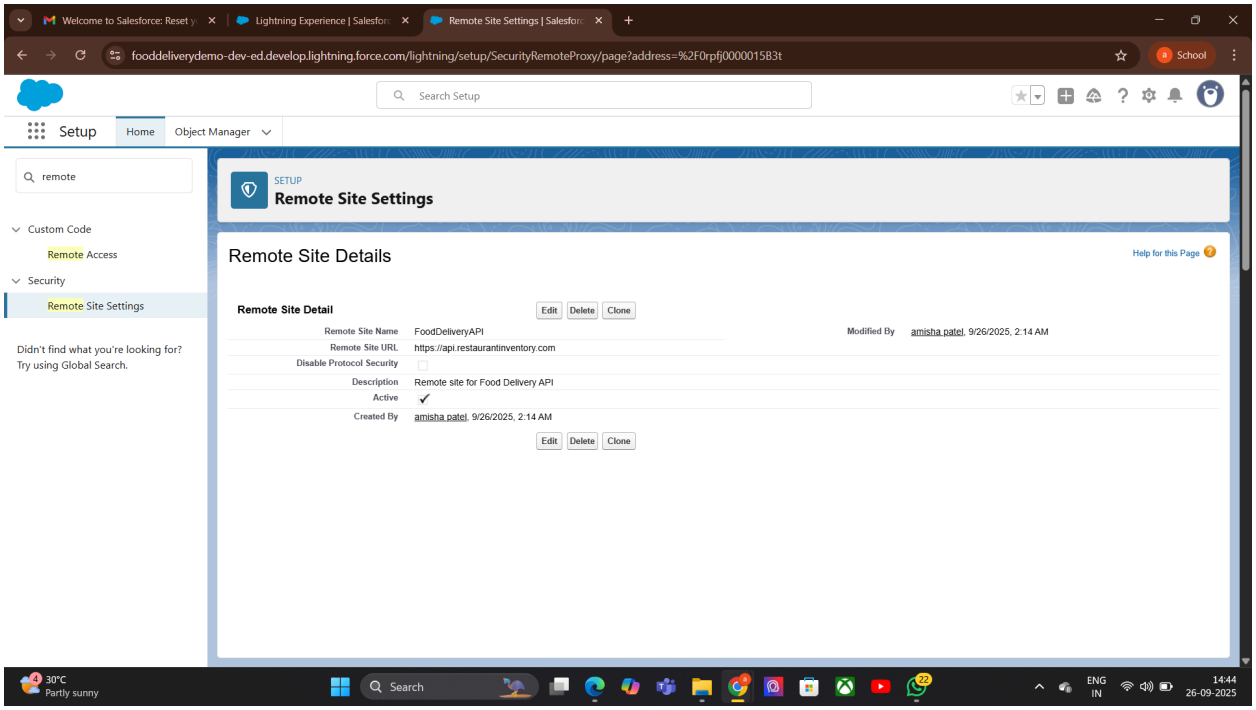
9.Remote Site Settings:

Purpose

- Whitelist external URLs so Salesforce can make HTTP callouts to them.
- Protect Salesforce from unauthorized external requests.
- Ensure security by restricting callouts to known endpoints

Steps (Process)

1. Go to Setup → Search Remote Site Settings.
2. Click “New Remote Site”.
3. Enter Details →
    - Remote Site Name → descriptive name (e.g., FoodAPI).
    - Remote Site URL → API base URL (e.g., https://api.fooddelivery.com).
4. Save → Salesforce now allows callouts to this URL.
5. Use in Callouts → Apex/Flow callouts can now connect to the whitelisted endpoint.


Welcome to Salesforce: Reset y
Lightning Experience | Salesforc
Remote Site Settings | Salesforc
fooddeliverydemo-dev-ed.develop.lightning.force.com/lightning/setup/SecurityRemoteProxy/page?address=%2F0rpfj0000015B3t
School
Search Setup
Setup
Home
Object Manager
remote
Custom Code
Remote Access
Security
Remote Site Settings
Didn't find what you're looking for?
Try using Global Search.
SETUP
Remote Site Settings
Remote Site Details
Help for this Page
Remote Site Detail
Edit
Delete
Clone
Remote Site Name
FoodDeliveryAPI
Remote Site URL
https://api.restaurantinventory.com
Disable Protocol Security
Description
Remote site for Food Delivery API
Active
Created By
amisha patel, 9/26/2025, 2:14 AM
Modified By
amisha patel, 9/26/2025, 2:14 AM
Edit
Delete
Clone
30°C
Partly sunny
Search
ENG
IN
14:44
26-09-2025

Phase 8: Data Management & Deployment:

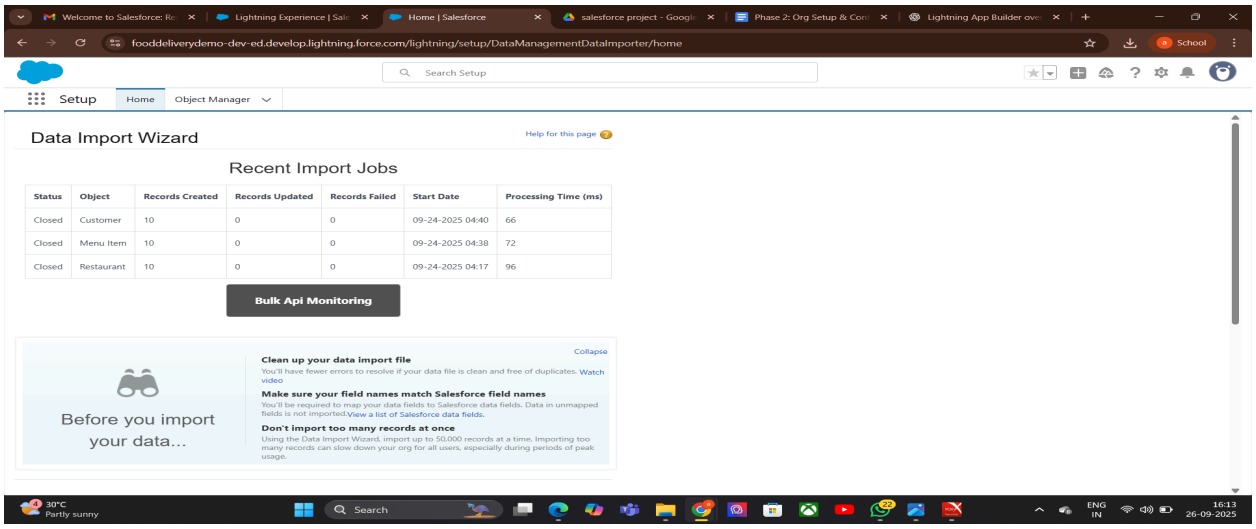
1. Data Import Wizard:

Purpose

- Simplify data import into Salesforce without coding.
- Import data for standard and custom objects

. Steps (Process)

1. Go to Setup → Search Data Import Wizard.
2. Select Object → Choose standard/custom object (e.g., Contacts, Orders).
3. Choose Data File → Upload CSV file with data.
4. Map Fields → Match CSV columns to Salesforce fields.
5. Start Import → Salesforce processes the file and imports data.
6. Review Import Results → Check success/errors.



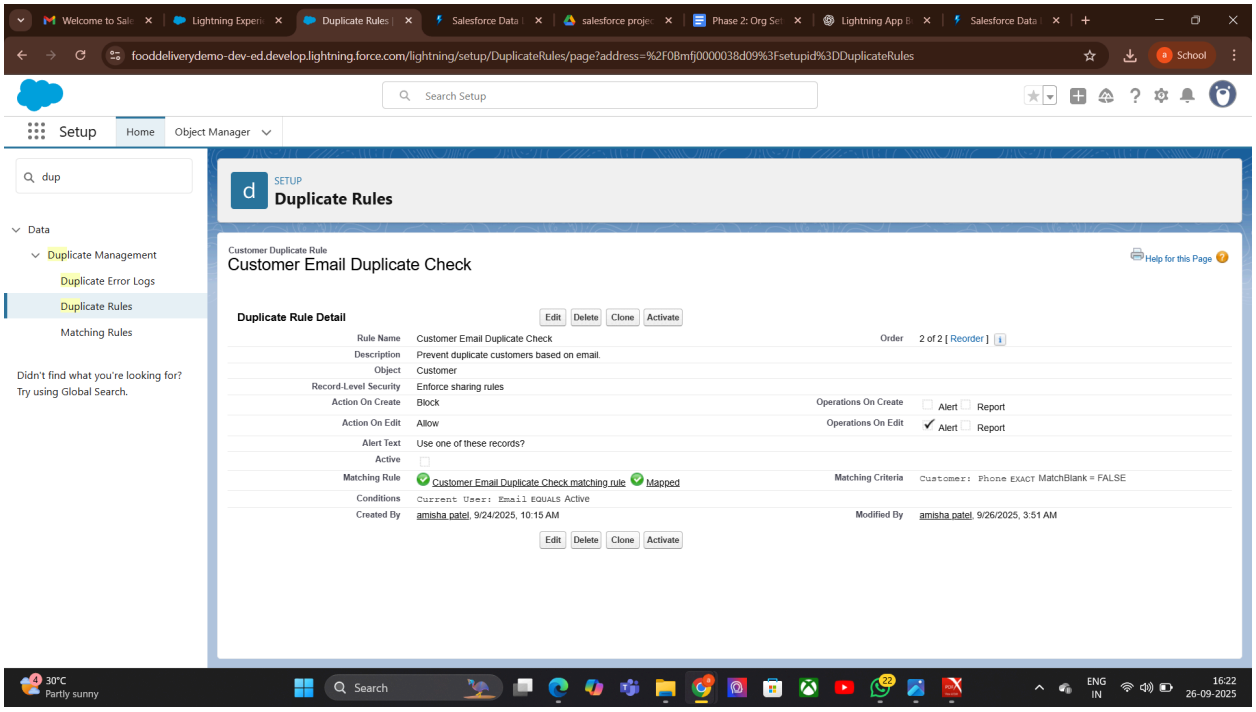
2. Duplicate Rules:

Purpose

- Prevent creation of duplicate records in Salesforce.
- Improve data quality and maintain clean records.
- Alert users or block duplicates during record creation or import.

Steps (Process)

1. Go to Setup → Search Duplicate Rules.
2. Create New Rule → Select object (e.g., Contacts, Leads, Accounts).
3. Select Matching Rule → Choose an existing rule or create a new one to define matching criteria.
4. Set Rule Action →
    - Alert → Notify user but allow save.
    - Block → Prevent duplicate record creation.
5. Activate Rule → Turn it ON to enforce duplicates checking.
6. Test Rule → Try creating duplicate record and verify behavior.



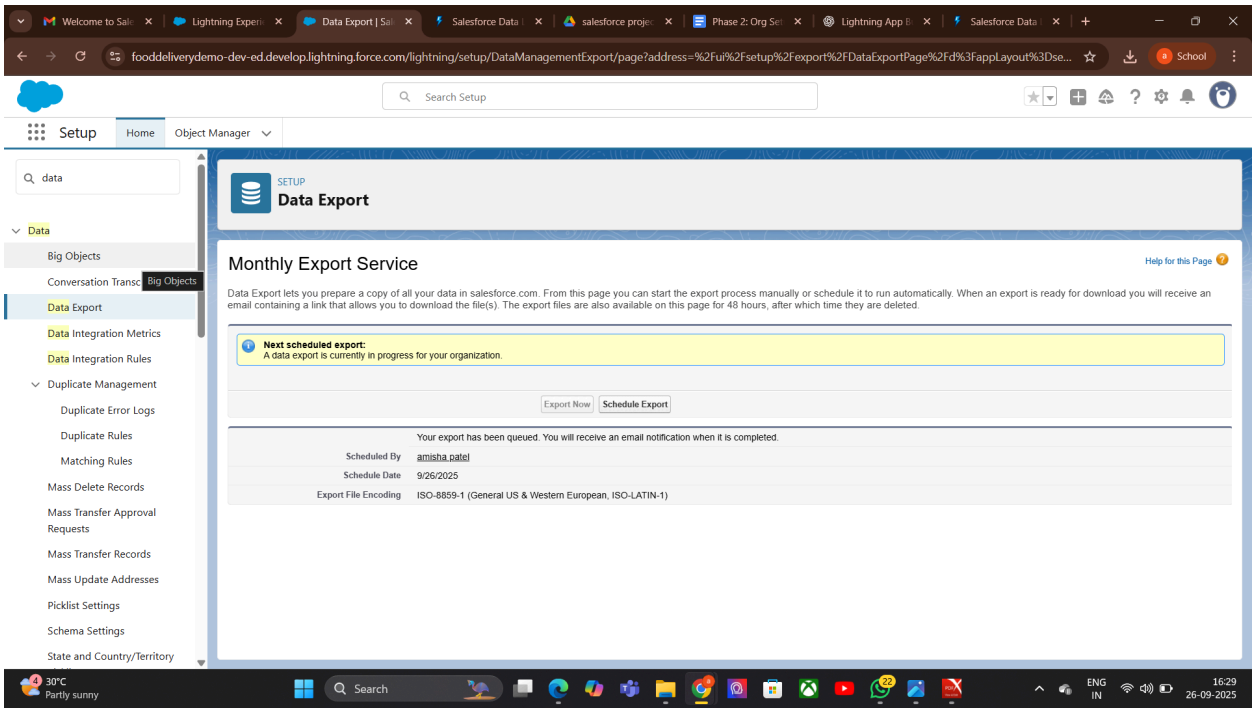
3.Data Export :

Purpose

- Extract Salesforce data for backup, migration, or reporting.
- Export data for both standard and custom objects.
- Schedule regular exports to maintain backups.

Steps (Process)

1. Go to Setup → Search Data Export.
2. Choose Export Type →
    - Manual Export → Run export instantly.
    - Scheduled Export → Set a schedule (weekly/monthly).
3. Select Objects → Choose specific objects or all data.
4. Select Export Options → Include attachments, documents, chatter files if needed.
5. Start Export → Salesforce prepares ZIP files with CSV data.
6. Download Files → Use provided link once export is complete.



4. Unmanaged vs Managed Packages:

unmanaged-

Purpose-Used for sharing code/configuration for learning or one-time installation.

No upgrade support — changes must be manually applied.

Unmanaged Package

- Go to Setup → Search Packages.

- Click New → Name package.
- Add components (objects, fields, Apex classes, etc.).
- Upload → Generate install link.
- Install in target org.

Managed-

Used for distributing apps in AppExchange or for controlled upgrades.

Supports upgrades and version control

process-

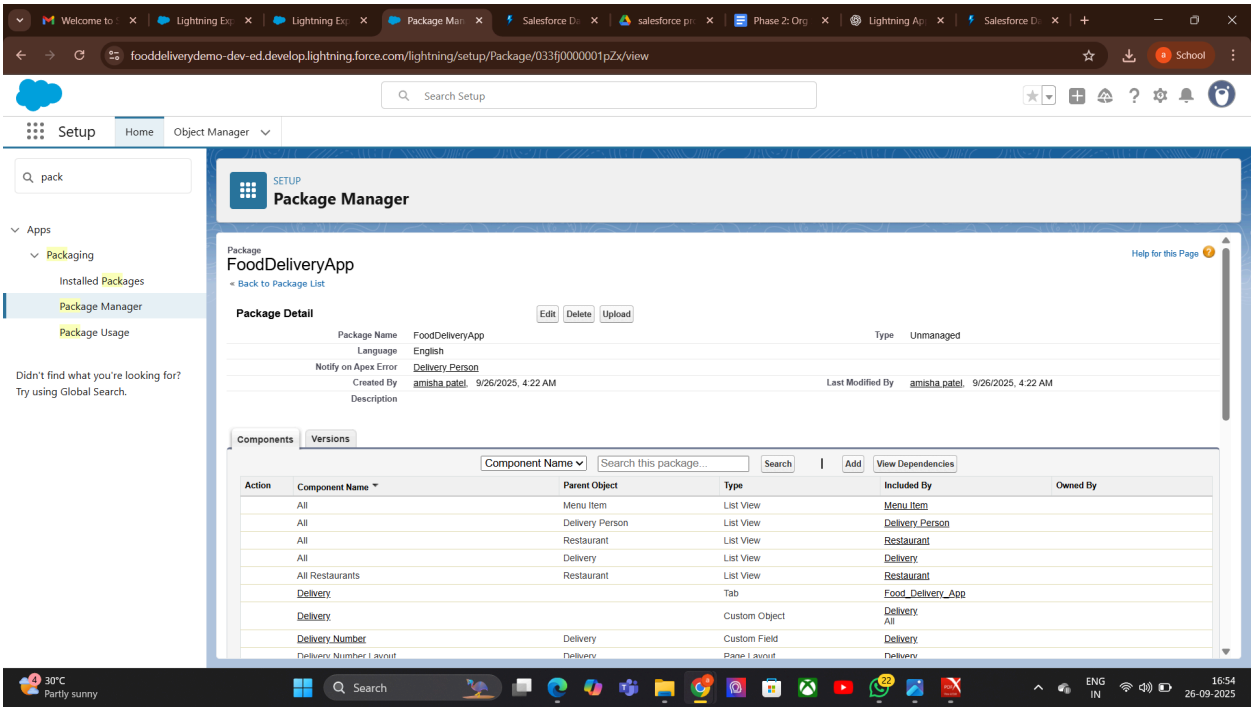
Go to Setup → Create a **Developer Edition org**.

Create package and add components.

Register namespace (unique).

Upload package as managed.

Install via AppExchange link.



5.ANT Migration Tool:

Purpose

- Automate deployment of metadata between Salesforce orgs.
- Useful for continuous integration (CI) and version control.
- Works well for large projects with many components.

Steps (Process)

1. **Install Java & ANT** → Ensure Java and Apache ANT are installed on your system.
2. **Download Salesforce ANT Migration Tool** → From Salesforce Setup → Tools → Force.com Migration Tool.
3. **Set Up project folder** → Create a folder for project and build files.
4. **Configure build.xml & package.xml** → Define metadata to retrieve/deploy.
5. **Retrieve Metadata** → Use ANT command:
6. ant retrieve
7. **Deploy Metadata** → Use ANT command:
8. ant deploy
9. **Check Deployment Results** → Review logs

6.Data Loader:

Purpose

- Bulk import, export, update, delete Salesforce records.
- Useful for large data operations (thousands or millions of records).
- Supports CSV file format and automation via command line.

Steps (Process)

- Download & Install Data Loader → From Salesforce Setup → Data Loader.
- Login → Use Salesforce credentials (or OAuth).
- Select Operation → Insert, Update, Upsert, Delete, Export, Export All.

- Choose Object → Select the Salesforce object to work on (e.g., Contact, Account).
- Select CSV File → Map CSV columns to Salesforce fields.
- Run Operation → Process the data.
- Check Results → Review success and error files generated by Data Loader.

7.VS Code & SFDX:

Purpose

- VS Code → Powerful code editor for Salesforce development.
- SFDX (Salesforce DX) → CLI tool for modern Salesforce development and deployment.
- Enable source-driven development, version control, and team collaboration

Small Steps (Process)

1. **Install VS Code** → Download from Visual Studio Code website.
2. **Install Salesforce Extensions Pack** → From VS Code Extensions Marketplace.
3. **Install Salesforce CLI (SFDX)** → From Salesforce Developer site.

Authorize Org →

```
sfdx force:auth:web:login -a MyOrgAlias
```

Create Project →

```
sfdx force:project:create -n MyProject
```

Retrieve Metadata →

```
sfdx force:source:retrieve -m ApexClass:MyClass
```

Deploy Metadata →

```
sfdx force:source:deploy -p force-app
```

Run Tests →

```
sfdx force:apex:test:run
```

Phase 9: Reporting, Dashboards & Security Review

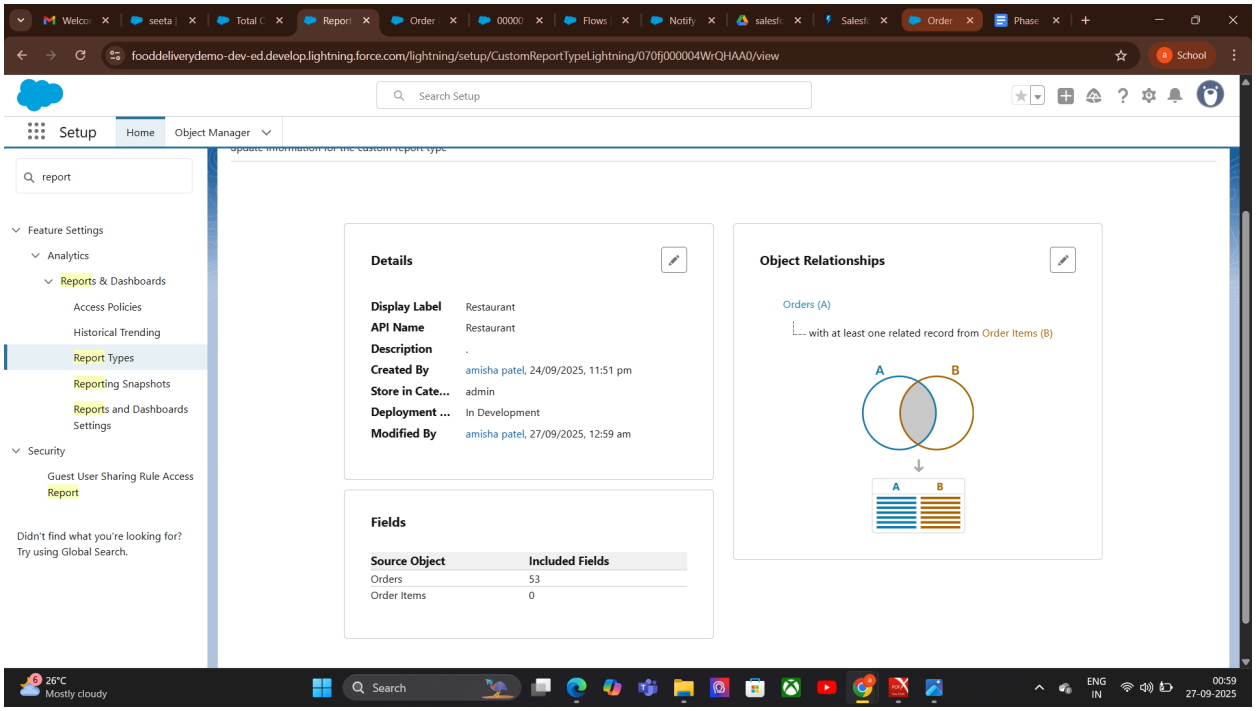
1. Report Types:

Purpose

- Define which objects and fields are available for reports in Salesforce.
- Control how records are related in reports.
- Allow creation of custom reports for specific business needs.

Small Steps (Process)

- **Go to Setup** → Search **Report Types**.
- **Click "New Custom Report Type"**.
- **Select Primary Object** → Choose the main object (e.g., Accounts, Orders).
- **Define Related Objects** → Add related objects if needed.
- **Set Deployment Status** → In Development (editable) or Deployed (available to all users).
- **Save** → Use this report type when creating reports.



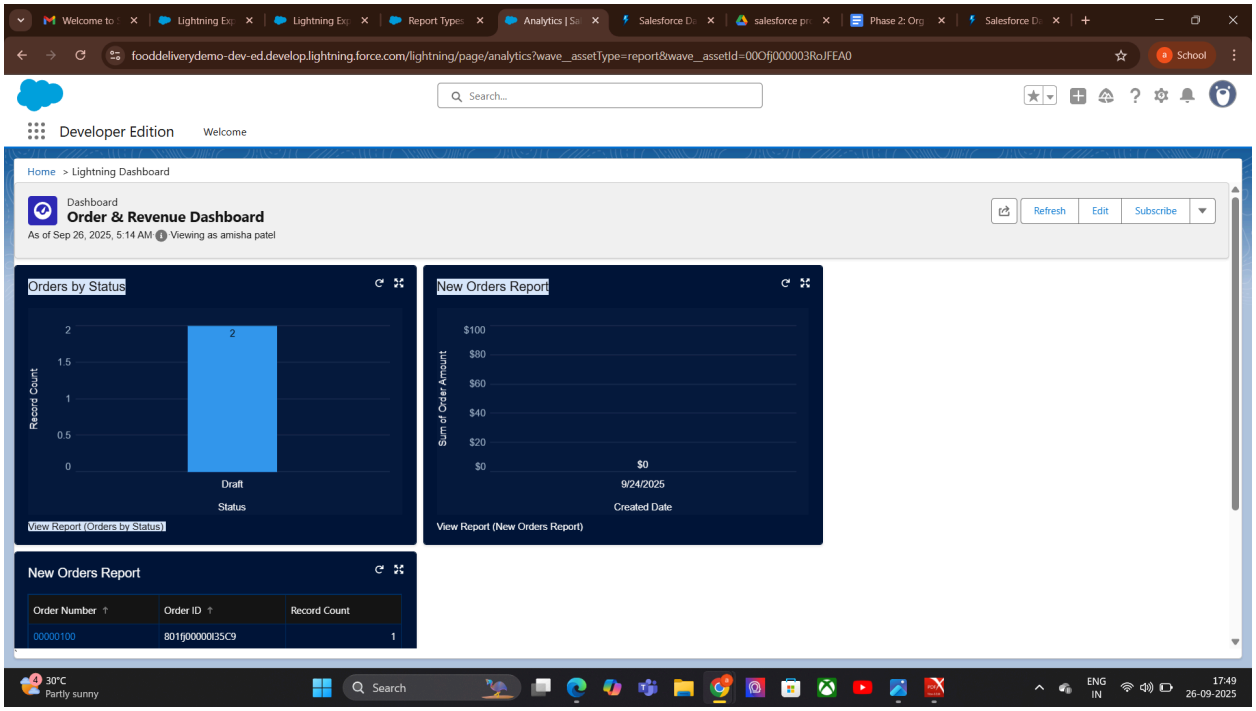
2. Dashboards:

Purpose

- Visually display Salesforce data in charts, graphs, tables, and gauges.
- Provide quick insights and analytics for decision-making.
- Summarize reports for managers and teams.

Steps (Process)

1. Go to Dashboards Tab → Click New Dashboard.
2. Name Dashboard → Give a meaningful name and select a folder.
3. Add Components → Charts, graphs, tables, gauges.
4. Select Source Reports → Choose reports that will feed data to dashboard components.
5. Customize Components → Adjust chart types, filters, labels.
6. Save & Refresh → Ensure data is up-to-date.



3.Sharing Settings:

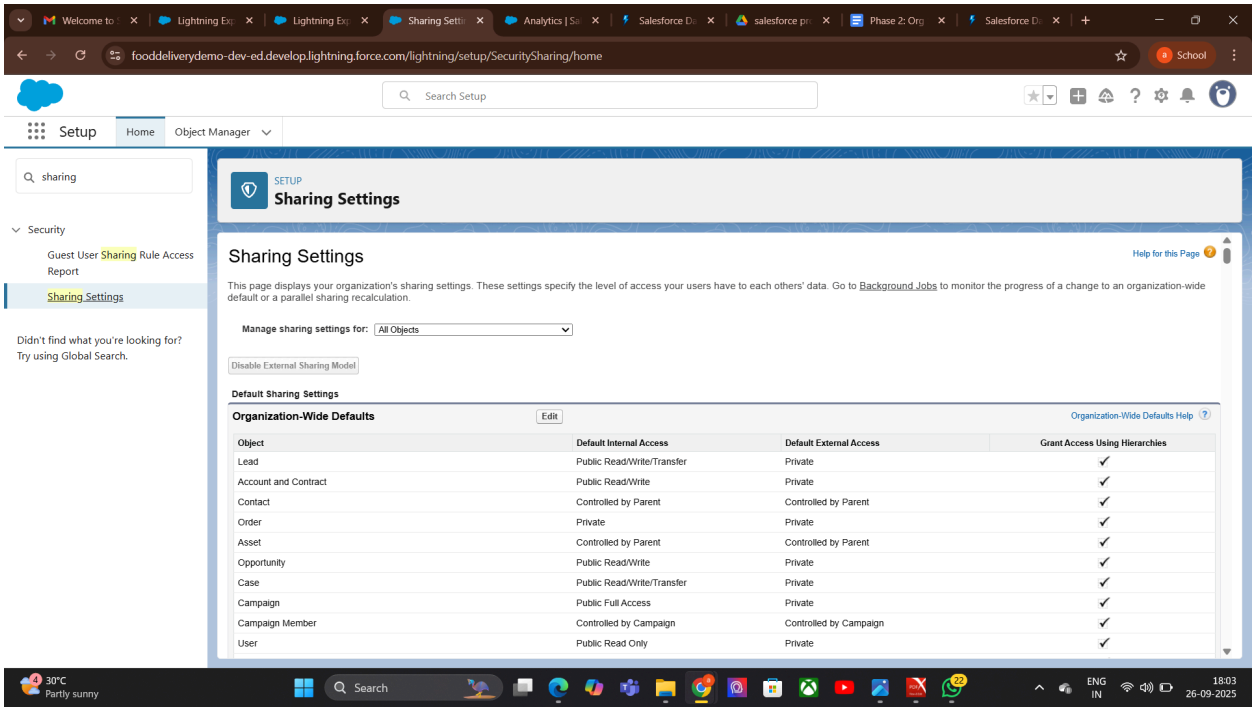
Purpose

- Control record-level access in Salesforce.
- Define who can view, edit, or delete records.

- Maintain data security while enabling collaboration.

Steps (Process)

1. **Go to Setup** → Search **Sharing Settings**.
2. **Set Organization-Wide Defaults (OWD)** → Define baseline access for each object (Public, Private, Controlled).
3. **Create Sharing Rules** → Grant access to specific groups/roles.
4. **Use Role Hierarchies** → Inherit access based on role levels.
5. **Set Manual Sharing** → Allow record owners to share records individually.
6. **Test Access** → Log in as a user to verify permissions.



4.Field Level Security:

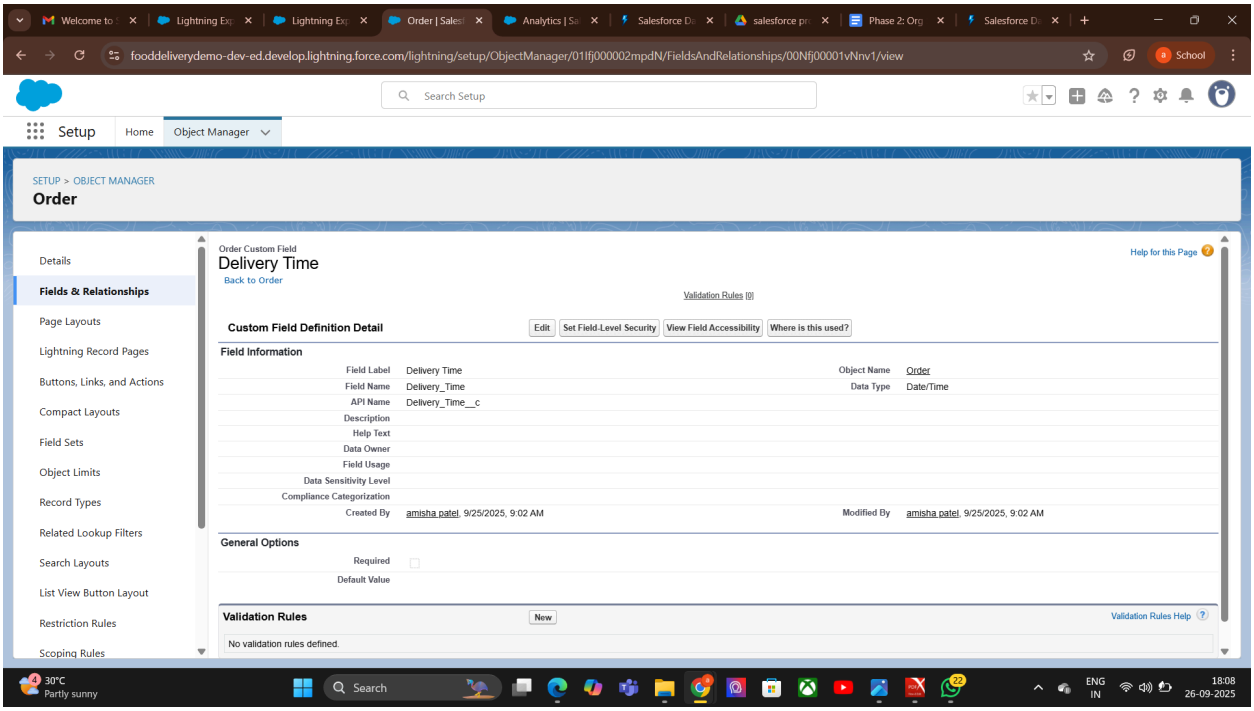
Purpose

- Control visibility and editability of individual fields in Salesforce objects.
- Protect sensitive data while allowing access to necessary information.
- Apply security at the profile or permission set level.

- Steps (Process)

1. Go to Setup → Search Field Accessibility or Profiles.

2. Select Object → Choose the object containing the field.
3. Select Field → Click on the field to edit security.
4. Set Access Levels →
    - Visible → User can see the field.
    - Read-Only → User can view but not edit.
    - Hidden → Field is not visible.
5. Save Changes → Apply settings for profiles or permission sets.
6. Test Access → Log in as user to confirm field visibility.



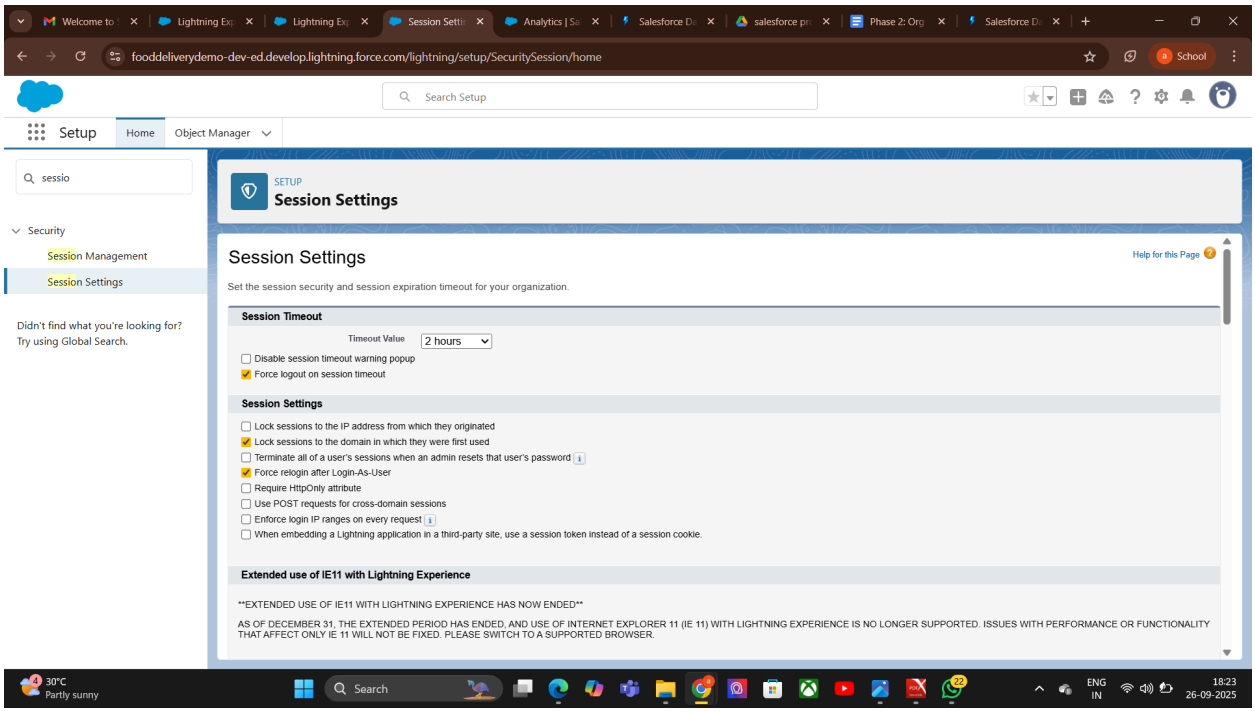
5.Session Settings:

Purpose

- Control user session behavior in Salesforce for security and performance.
- Define session timeout, security policies, and login access.
- Protect org data by managing how and when sessions expire.

Steps (Process)

1. Go to Setup → Search Session Settings.
2. Set Session Timeout → Choose duration (e.g., 30 min, 2 hours).
3. Configure Security Policies → Enable/disable secure session settings (e.g., IPranges, login hours).
4. Enable/Disable Persistent Sessions → Decide whether users stay logged in across browser restarts.
5. Save Changes → Apply settings.
6. Test → Log in as a user to verify session behavior.



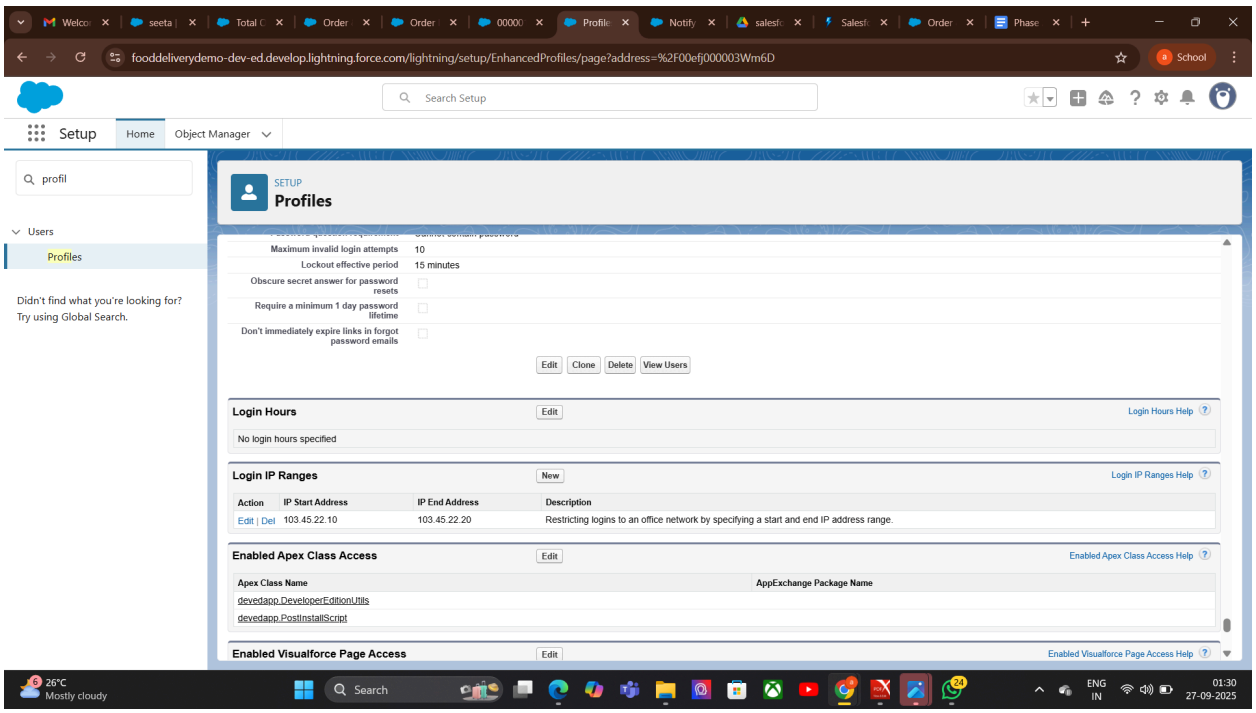
6.Login IP Ranges:

Purpose

- Restrict Salesforce login access to specific IP address ranges.
- Enhance org security by controlling where users can log in from.
- Useful for enforcing network-based access control.

Steps (Process)

1. Go to Setup → Search Profiles (IP ranges are set per profile).
2. Select Profile → Click the profile you want to set IP ranges for.
3. Find Login IP Ranges Section → Click New.
4. Enter IP Range → Start IP and End IP addresses (e.g., `103.45.22.10` to `103.45.22.20`).
5. Save Changes → Restriction applies immediately.



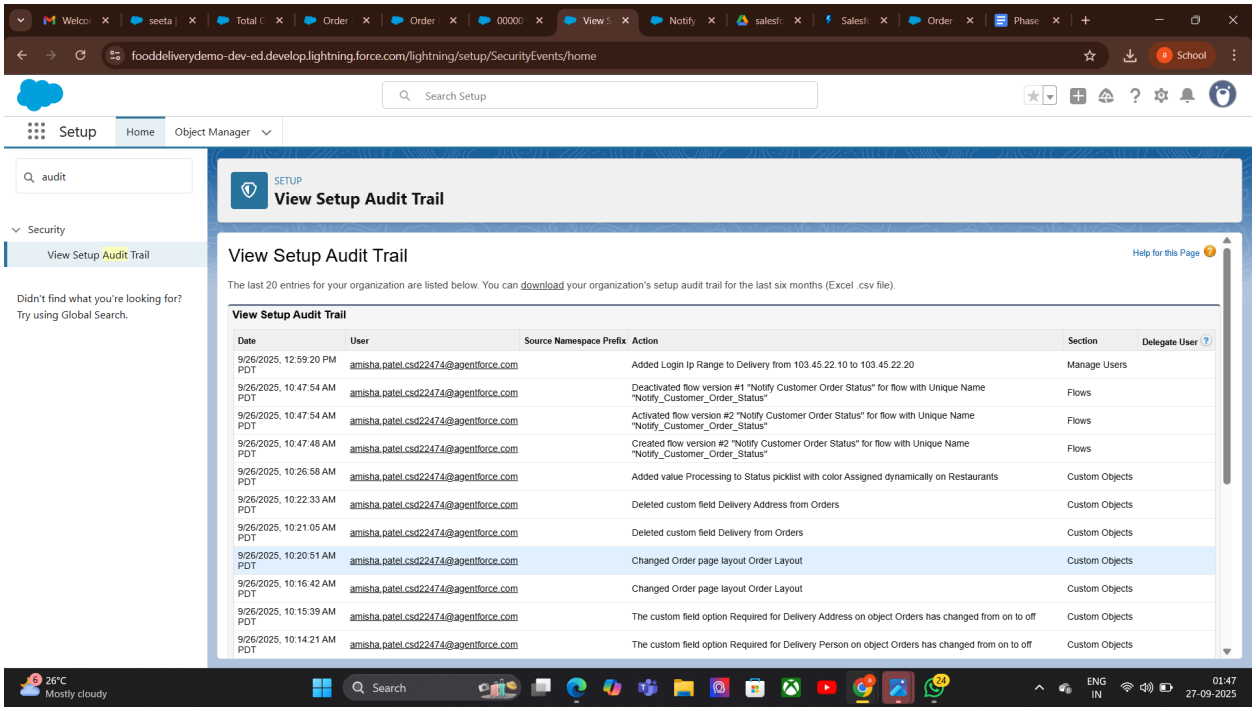
7.Audit Trail:

Purpose

- Track administrative changes for accountability.
- Support compliance and security audits.
- Help troubleshoot unexpected behavior by reviewing configuration changes.

Steps (Process)

1. Go to Setup → Search View Setup Audit Trail.
2. Review Recent Changes → See changes for the last 180 days.
3. Download Audit Log → Export for analysis or compliance documentation.
4. Filter by Date/User → Focus on specific changes.



Phase 10: Final Presentation & Demo Day

Goal : Successfully Complete the **FoodDelivery App Management**

1. Pitch Presentation:

Introduction-

- Welcome & introduce the team.
- State the goal: *"To revolutionize food ordering with faster, smarter, and more convenient delivery."*
- Briefly state the problem:
    - Long delivery times
    - Limited restaurant choices
    - Poor tracking experience

Problem

In today's fast-paced world, customers face slow deliveries, limited restaurant choices, and poor order tracking.

Solution

smart Food Delivery app offering real-time tracking, wide restaurant selection, easy ordering, and secure payment.

Benefits

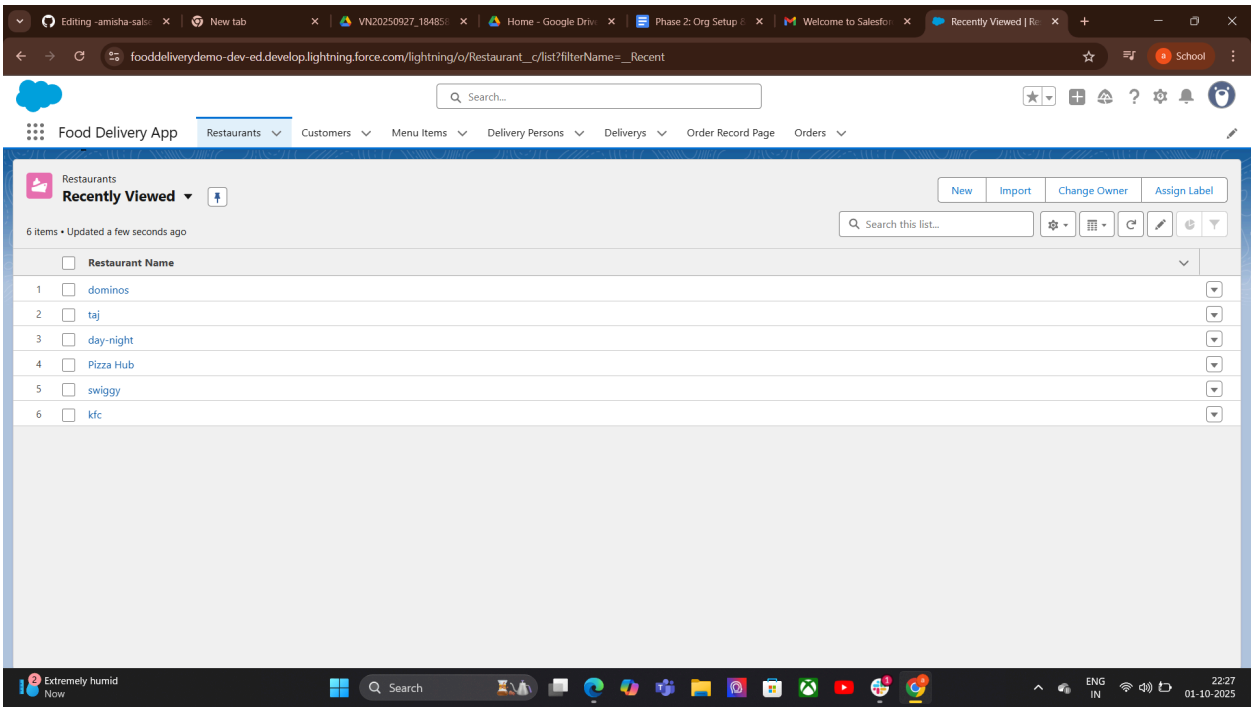
Faster deliveries, higher customer satisfaction, increased orders for restaurants, and sustainable business growth.

2. Demo Walkthrough:

Walk through the app's core functionality in a clear, logical order so the audience easily understands its features and benefits.

Customer & Order Management

- Customer Onboarding: Demonstrate creating a new customer account with details such as name, address, contact number, and preferences.



- Placing an Order: Show how the customer browses restaurants, selects menu items, customizes them, and places an order.

- Order Summary: Display order confirmation with details such as order number, estimated delivery time, and payment summary.

2. Real-Time Tracking & Notifications

- Order Status Updates: Demonstrate how the app updates customers with real-time order statuses:
    - Order Received
    - Preparing
    - Out for Delivery
    - Delivered
- Live Map Tracking: Show the delivery tracking map that updates location in real time.

3. Business Insights

- Delivery Dashboard: Display a dashboard showing key metrics:
- Customer Insights: Show analytics like repeat customers, average order value, and ratings.

3.Feedback & Ratings

- Demonstrate how customers can rate and review orders.
- Show how this feedback is stored and displayed for restaurant performance improvement.

4. Handoff Documentation:

. User Guide

- Placing an Order: Search restaurants, choose items, customize orders, checkout, and payment.
- Order Tracking: Real-time updates and live map tracking.
- Order History & Feedback: View past orders and give ratings.
- Support: FAQs and contact details.

.Technical Guide

A brief guide for future administrators to maintain and manage the app.

Contents:

- **System Overview:** Key modules — Customer Management, Order Management, Inventory, Delivery Tracking, Analytics.
- **Custom Objects:** Customers, Orders, Restaurants, Delivery.
- **Automation Rules:** Inventory updates and low-stock alerts.
- **Security Settings:** User roles and access permissions.
- **Dashboard:** Sales trends and key metrics.

5. LinkedIn/Portfolio Project Showcase:

1. Project Title- "*Food Delivery App – Real-Time Tracking & Automation*"

Problem Statement: Customers face slow deliveries, limited restaurant choices, and poor tracking experiences in current food delivery systems.