

Food Delivery Problem Solving Project

Phase 1: Problem Understanding & Industry Analysis

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.

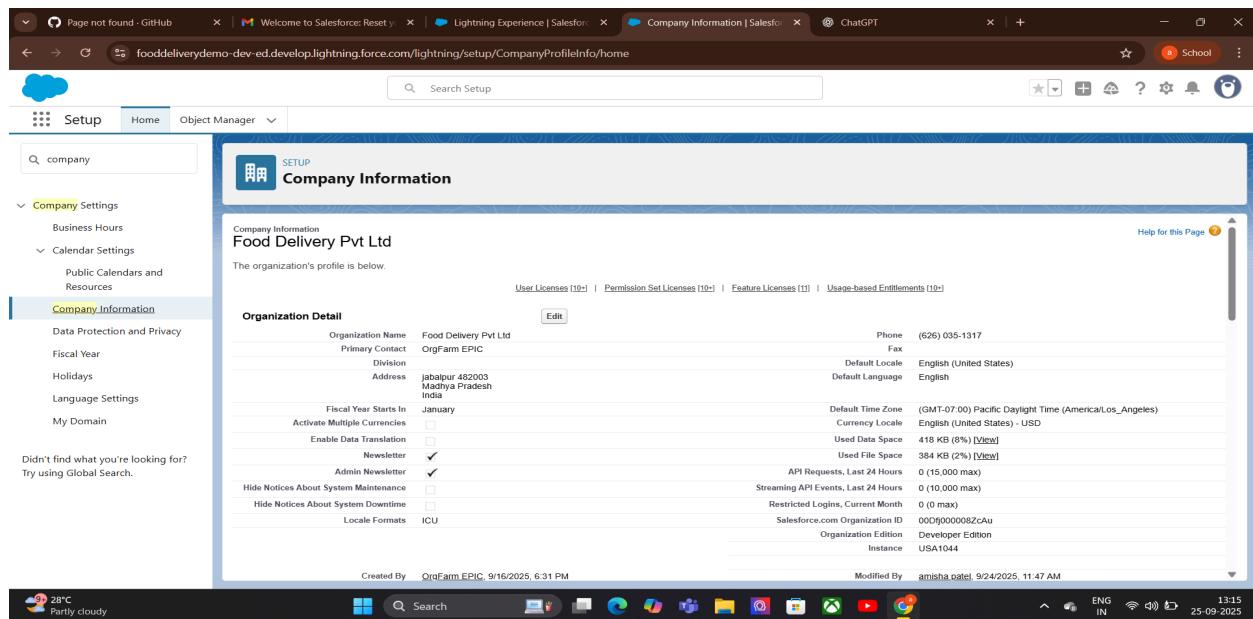
Phase 2: Org Setup & Configuration

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

Organization Business Hours

Select the days and hours that your support team is available. These hours, when associated with escalation rules, determine the times at which cases can escalate.

If you enter blank business hours for a day, that means your organization does not operate on that day.

[Holidays \[0\]](#)

Business Hours Detail		Edit	
Business Hours Name	Default	Time Zone	(GMT-07:00) Pacific Daylight Time (America/Los_Angeles)
Business Hours	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	24 Hours 24 Hours 24 Hours 24 Hours 24 Hours 24 Hours 24 Hours	Default Business Hours <input checked="" type="checkbox"/>
Active	<input checked="" type="checkbox"/>		
Created By	OrgFarm EPIC	9/16/2025, 6:31 PM	Last Modified By amisha.patel 9/20/2025, 1:24 PM
Edit			

4.Ficsal Year Settings (Food Delivery App):

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

The screenshot shows the Salesforce Setup interface. On the left, the navigation sidebar is open, showing sections like Company Settings, Business Hours, Calendar Settings, Public Calendars and Resources, Company Information, Data Protection and Privacy, and **Fiscal Year**. The **Fiscal Year** section is currently selected. The main content area displays the "Organization Fiscal Year Edit: Food Delivery Pvt Ltd" page. It asks to specify the fiscal year type for the organization. A warning message states: "Changing the fiscal year shifts fiscal periods and impacts opportunities and forecasts across your organization. If your forecast periods are set to quarterly, adjusting the fiscal year start month will erase existing forecast adjustments and quotas. Consider exporting a data backup before implementing this change." Below this, there's a "Change Fiscal Year Period" section with fields for Name (Food Delivery Pvt Ltd), Fiscal Year Start Month (January), and Fiscal Year Is Based On (The ending month). Buttons for Save and Cancel are at the bottom right of the modal.

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.

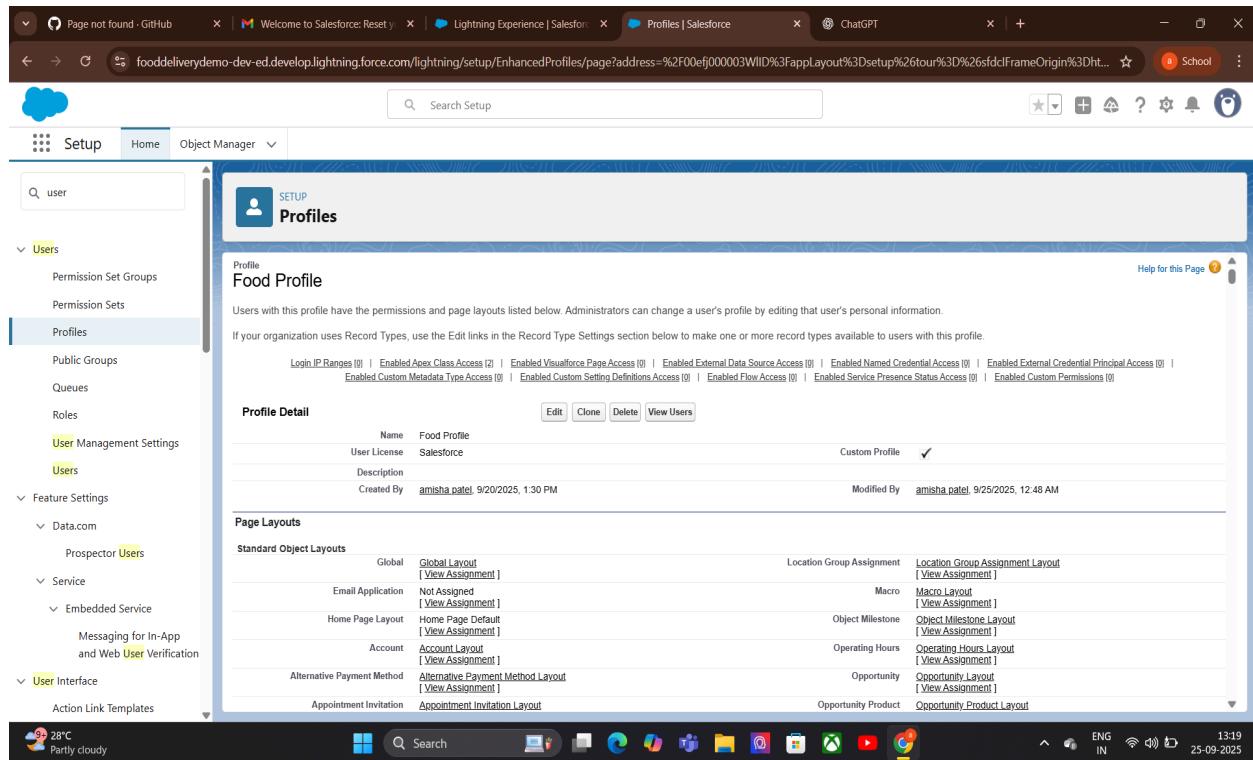
The screenshot shows the Salesforce Lightning Experience interface. The top navigation bar includes tabs for Page not Found - GitHub, Welcome to Salesforce: Reset, Lightning Experience | Salesforce, Users | Salesforce, and ChatGPT. The URL in the address bar is fooddeliverydemo-dev-ed.develop.lightning.force.com/lightning/setup/ManageUsers/page?address=%2F005fj000005hbZJ%3Fnoredirect%3D1%26isUserEntityOverride%3D1. The page title is "Users" under the "SETUP" tab. The main content area displays a user profile for "Delivery Person". The user details include Name: Delivery Person, Alias: dell, Email: delivery147@food.com [Verify] (blue link), Username: delivery@example.com, Nickname: User1758398943672160142, Title: , Company: , Department: , Division: , Address: , Time Zone: (GMT-07:00) Pacific Daylight Time (America/Los_Angeles), Locale: English (United States), Language: English, Delegated Approver: Manager, and Receive Approval Request Emails: Only if I am an approver. The "Delivery Role" section shows the user is assigned as a "Standard User". Other settings include User License Profile (Salesforce), Marketing User (Active checked), Offline User, Knowledge User, Flow User, Service Cloud User, Site.com Contributor User, Site.com Publisher User, WDC User, Mobile Push Registrations (Data.com User Type), Accessibility Mode (Classic Only), Debug Mode, and High Contrast Palette on Charts. The left sidebar shows navigation links for Setup, Home, Object Manager, and various user management sections like Permission Set Groups, Permission Sets, Profiles, Public Groups, Queues, Roles, User Management Settings (selected), and Users. The bottom of the screen features the Windows taskbar with icons for File Explorer, Search, Task View, and several pinned applications.

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.

Creating the Role Hierarchy

You can build on the existing role hierarchy shown on this page. To insert a new role, click **Add Role**.

Your Organization's Role Hierarchy

[Collapse All](#) [Expand All](#)



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.

The screenshot shows the 'Delivery Extra Access' permission set setup page. The top navigation bar includes links for 'SETUP' and 'PERMISSION SET 'DELIVERY EXTRA ACCESS''. The main title is 'Delivery Extra Access'. Below the title, there is a section titled 'Current Assignments' with a table showing one assignment for a 'Delivery Person' role. The table has columns for 'Full Name', 'Active', 'Role', 'Profile', 'User License', and 'Expires On'. The 'Active' column for the row shows a checked checkbox. The 'Role' column shows 'Delivery Role'. The 'Profile' column shows 'Standard User'. The 'User License' column shows 'Salesforce'. There are 'Edit', 'Delete', and 'Add Assignment' buttons above the table.

Full Name	Active	Role	Profile	User License	Expires On
Delivery Person	✓	Delivery Role	Standard User	Salesforce	

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.

Action	Criteria	Shared With	Access Level
Edit Del	Owner in Role_CEO	Role_Customer_Role	Read Only
Edit Del	Owner in Role_CEO	Role_Delivery_Role	Read Only

Object	Default Internal Access	Default External Access	Grant Access Using Hierarchies
Lead	Public Read/Write/Transfer	Private	✓
Account and Contact	Public Read/Write	Private	✓
Contact	Controlled by Parent	Controlled by Parent	✓
Order	Private	Private	✓
Asset	Controlled by Parent	Controlled by Parent	✓
Opportunity	Public Read/Write	Private	✓
Case	Public Read/Write/Transfer	Private	✓
Campaign	Public Full Access	Private	✓
Campaign Member	Controlled by Campaign	Controlled by Campaign	✓
User	Public Read Only	Private	✓

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'

The screenshot shows the 'Login Access Policies' setup page. At the top, there's a blue header bar with a shield icon and the word 'SETUP'. Below it, the title 'Login Access Policies' is displayed. On the right, there's a link 'Help for this Page' with a question mark icon. The main content area has a light gray background. It starts with a section titled 'Manage Support Options' with 'Save' and 'Cancel' buttons. Below this is a table with two rows. The first row has a 'Setting' column and an 'Enabled' column with a checked checkbox. The second row has a 'Support Organization' column containing 'Salesforce.com Support', a 'Packages' column with a yellow circle icon, an 'Available to Users' column with an empty circle icon, and an 'Available to Administrators Only' column with a blue information icon. The entire interface has a clean, modern design with a blue and white color scheme.

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.

The screenshot shows the 'My Domain Settings' setup page. At the top, there's a blue header bar with the title 'My Domain Settings' and a 'Help for this Page' link. The main content area has a light gray background. It starts with a section titled 'My Domain Details' with an 'Edit' button. Below this is a table with three rows. The first row shows 'Current My Domain URL' as 'fooddeliverydemo-dev-ed.develop.my.salesforce.com' with a note about partitioned enhanced domains. The second row shows 'My Domain Name' as 'fooddeliverydemo-dev-ed'. The third row shows 'Domain Suffix' as 'Standard (*.my.salesforce.com)'. There are also 'Edit' buttons for each row. The interface is consistent with the previous screenshot, featuring a blue and white color scheme and a clean layout.

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	Store additional restaurant info not covered by Account.

The screenshot shows the Salesforce Object Manager page. The URL in the browser is <https://fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/ObjectManager/home>. The page displays a table of objects with columns: Label, API Name, Type, Description, Last Modified, and Deployed. The objects listed are: Account, Activity, Address, Agent Work, Alternative Payment Method, API Anomaly Event Store, Appointment Category, Appointment Invitation, Appointment Invitee, Appointment Topic Time Slot, Approval Submission, Approval Submission Detail, and Approval Work Item. The 'Type' column indicates they are all Standard Objects. The 'Last Modified' and 'Deployed' columns show recent activity. The top navigation bar includes links for Setup, Home, and Object Manager, along with a search bar and various icons.

2. Fields:

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

Order Number → Auto Number (e.g., ORD-0001)

Order Date & Time → Date/Time

Order Status → Picklist

The screenshot shows the Salesforce Object Manager interface for the 'Order Item' object. On the left, there's a sidebar with various setup options like Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, etc. The main area is titled 'Fields & Relationships' and lists eight items, sorted by Field Label. The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Item Total	Item_Total__c	Formula (Currency)		
Last Modified By	LastModifiedById	Lookup(User)		
Menu Item	Menu_Item__c	Lookup(Menu Item)		
Order	Order__c	Master-Detail(Order)		
Order Item Name	Name	Auto Number		
Price	Price__c	Currency(18, 0)		
Quantity	Quantity__c	Number(16, 1)		

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

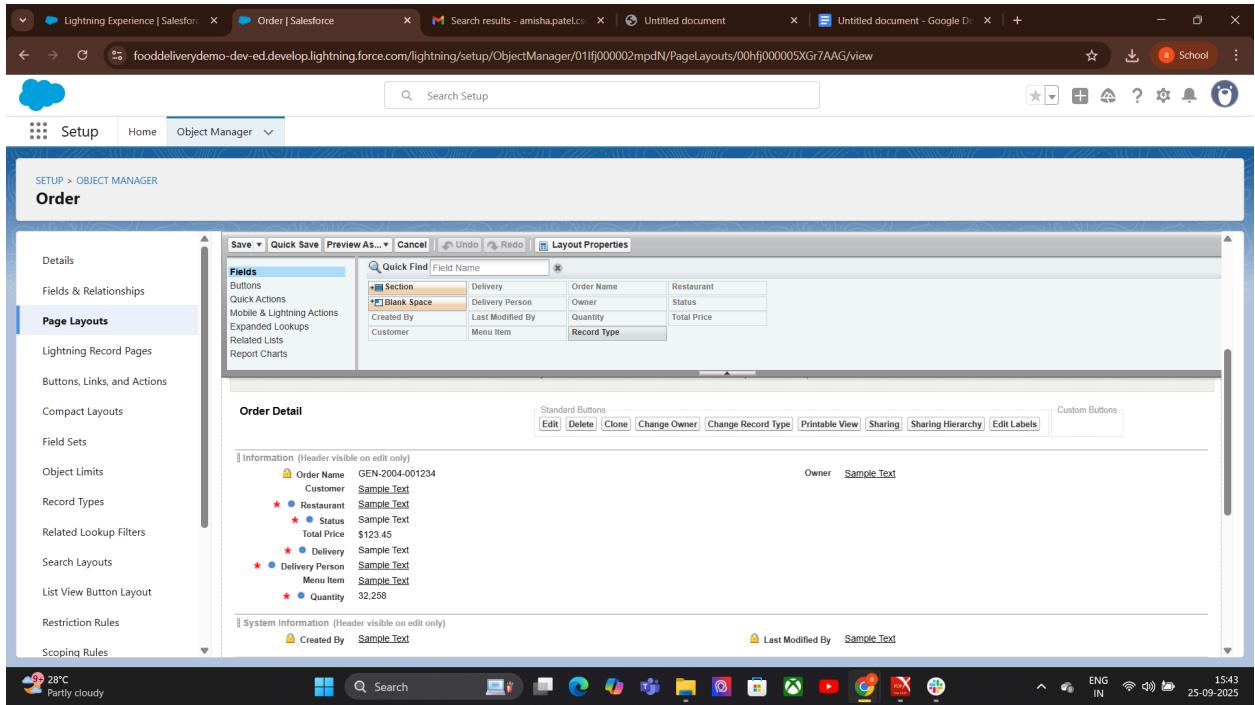
The screenshot shows the Salesforce Object Manager interface for the 'Order' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, and Record Types. The 'Record Types' option is selected and expanded, showing three items: Delivery Order, Dine-in Order, and Restaurant. Each record type has a checkmark in the 'ACTIVE' column and a modified timestamp and user in the 'MODIFIED BY' column.

RECORD TYPE LABEL	DESCRIPTION	ACTIVE	MODIFIED BY
Delivery Order		✓	amisha patel, 9/21/2025, 2:57 AM
Dine-in Order		✓	amisha patel, 9/21/2025, 2:56 AM
Restaurant		✓	amisha patel, 9/24/2025, 11:13 AM

4. Page Layouts:

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

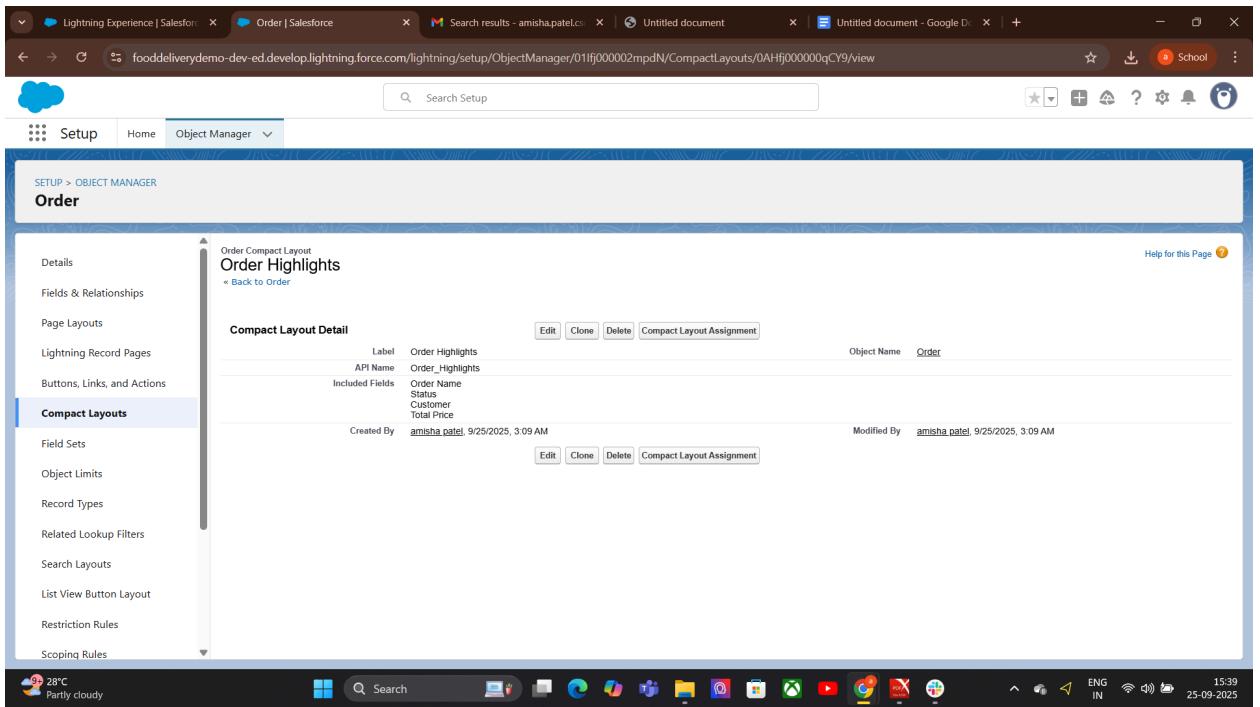
Object Manager → Object → Page Layouts → Edit Drag & drop fields, sections, related lists → Save.



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.

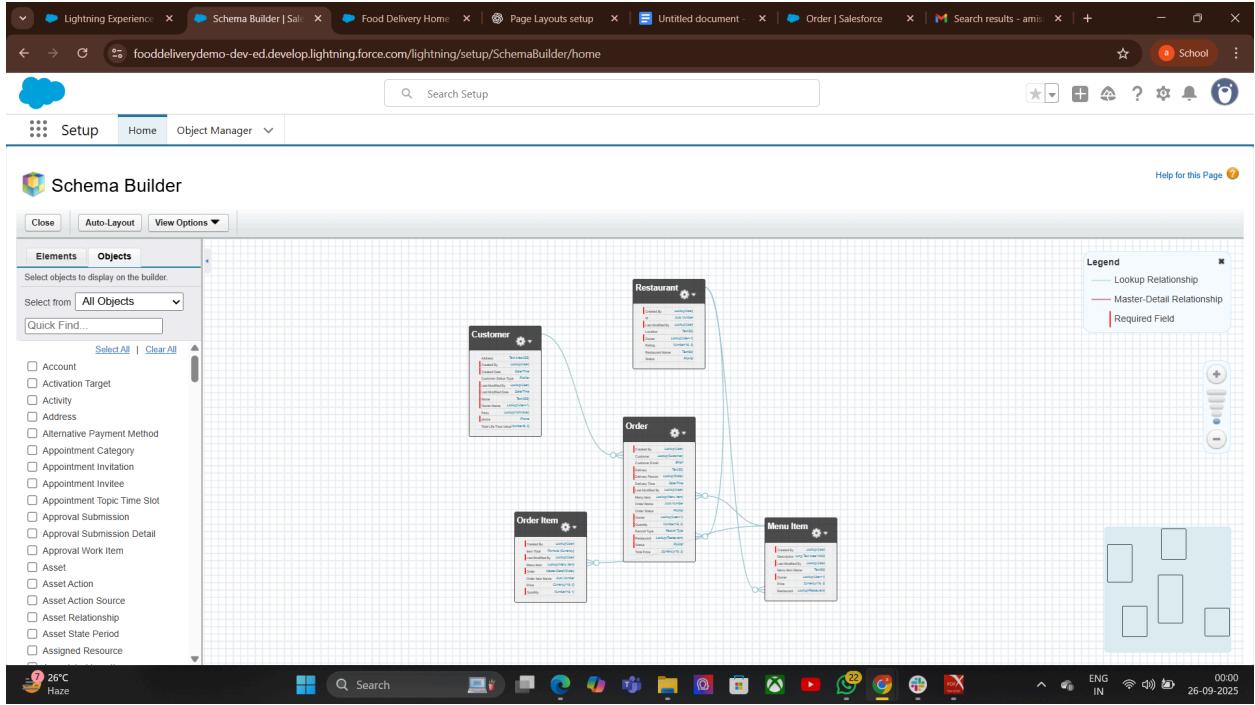


6. Schema Builder:

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.

Field	Type	Description	Relationship
Delivery Time	Date/Time		
Last Modified By	Lookup(User)		
Menu Item	Lookup(Menu Item)		
Order	Master-Detail(Order)		
Order Name	Name	Auto Number	
Order Status	Order_Status__c	Picklist	
Quantity	Quantity__c	Number(18, 0)	
Record Type	RecordTypeId	Record Type	

8. Junction Objects:

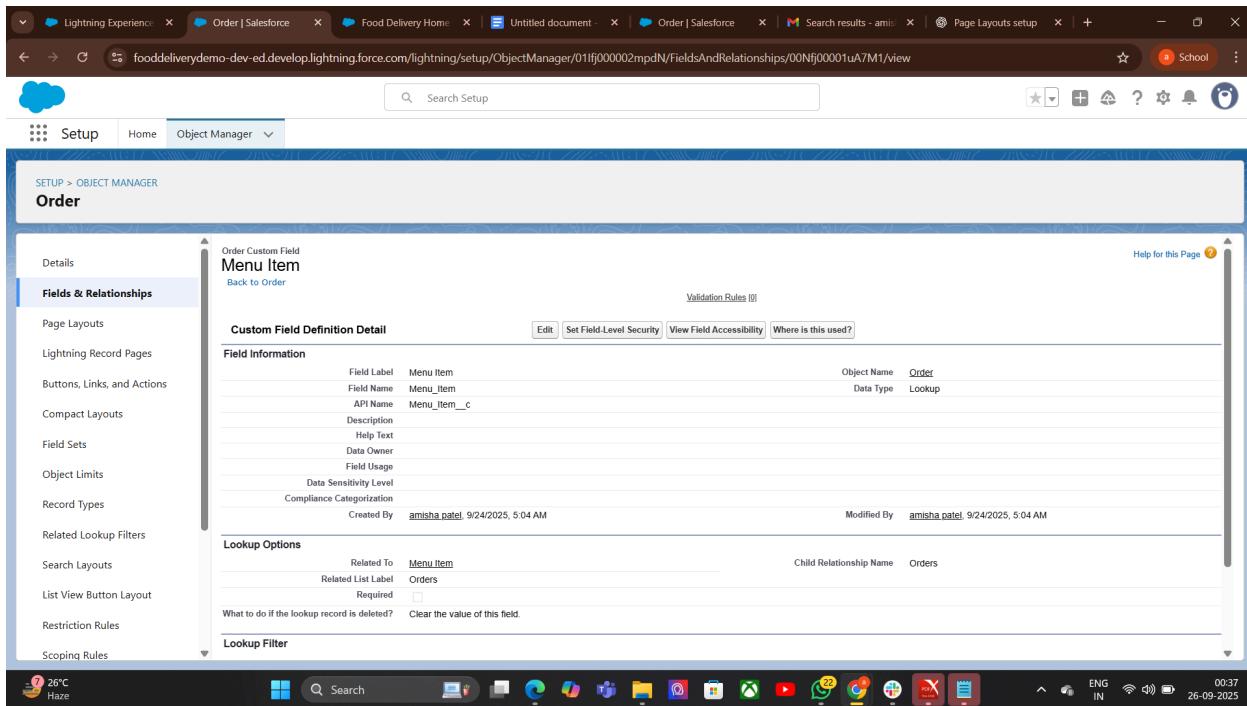
Setup → Object Manager → Create → Custom Object → Order_Item__c (Junction).

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.

The screenshot shows the Salesforce Lightning Experience interface. The left sidebar includes sections for Email, Apps (with External Client Apps selected), and Integrations (with External Data Sources selected). The main content area displays the 'External Data Sources' setup page under the 'SETUP' tab. A specific OData Menu Source named 'OData_Menu_Source' is being edited. The configuration includes a URL set to <https://api.restaurantinventory.com/odata/MenuItems>, a connection timeout of 120 seconds, and various parameters like High Data Volume, Server Driven Pagination, and Request Row Counts. The 'Authentication' section shows 'Current User' selected. The bottom of the screen shows a Windows taskbar with icons for various applications and the date/time.

Phase 4: Process Automation (Admin)

1. Validation Rules:

Food Delivery App ke Examples

Example 1 — Quantity must be greater than

Object: Order_Item__c

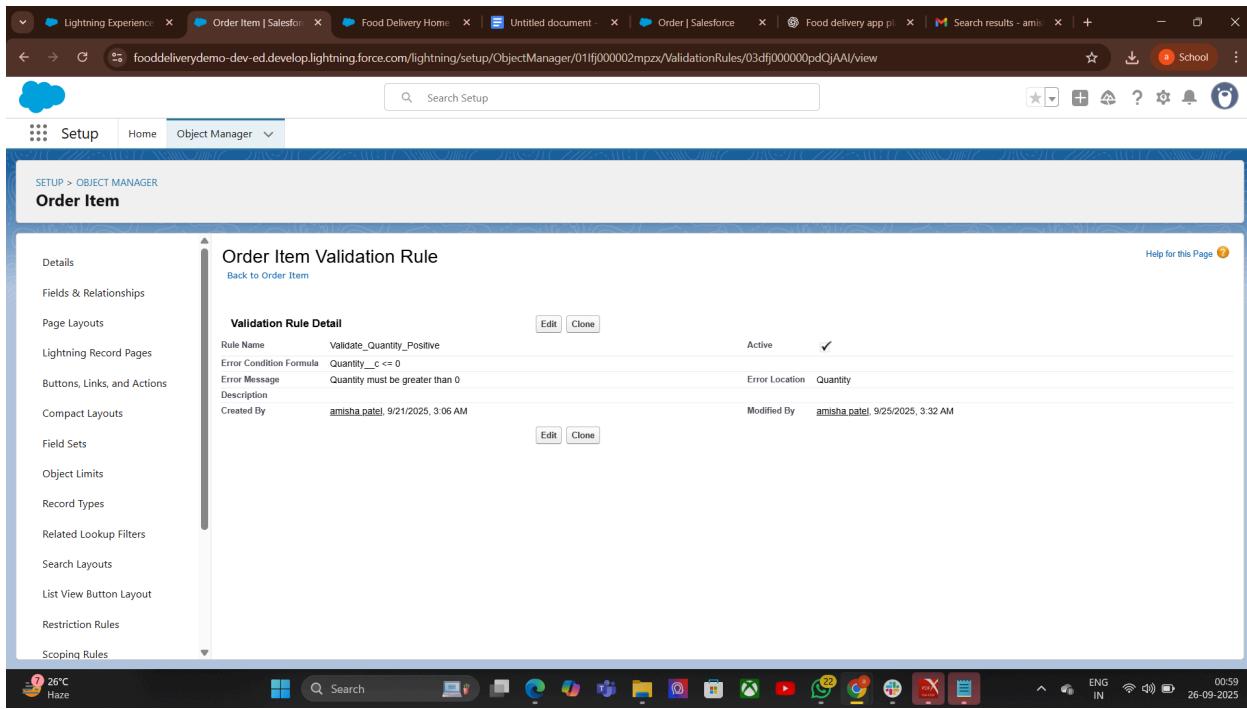
Field: Quantity__c

Formula:

`Quantity__c <= 0`

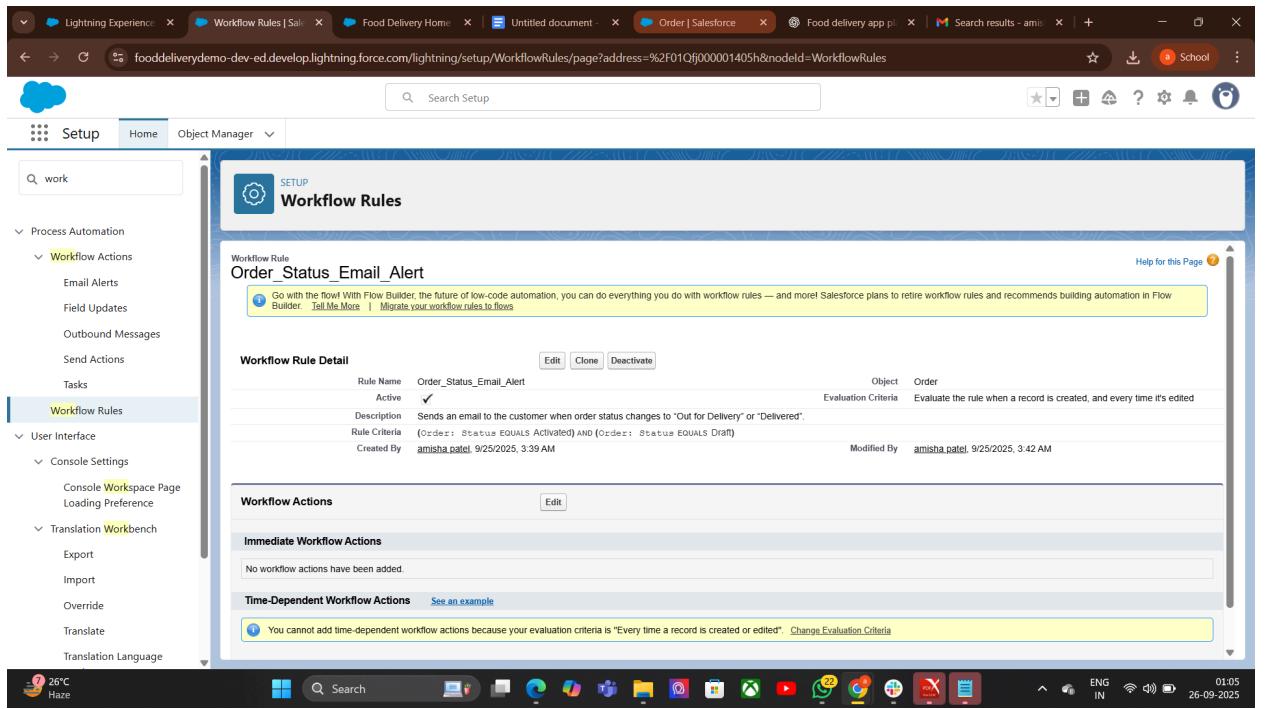
Error Message:

`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

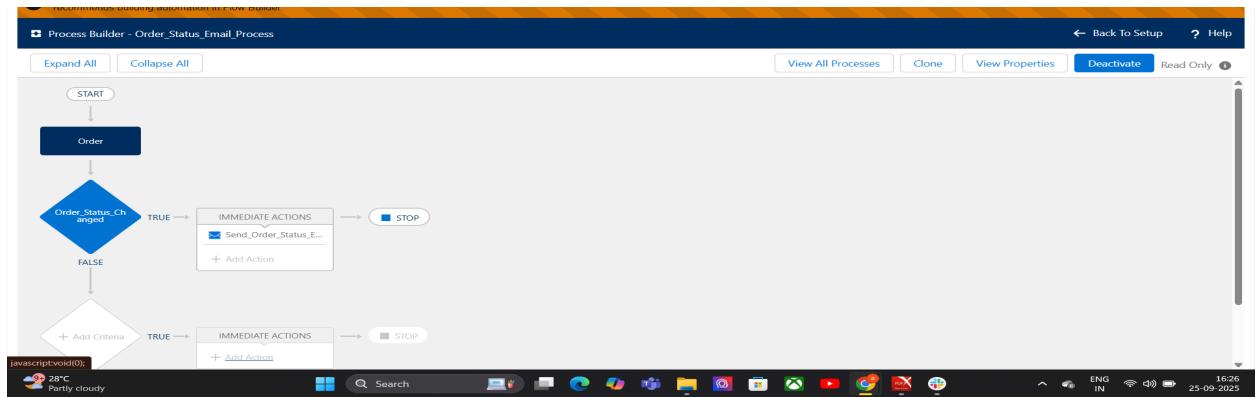
Give it a name: **Notify Delivery Person on Order Assignment**

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

The screenshot shows the Salesforce Setup interface for Approval Processes. The left sidebar navigation includes 'Entitlement Processes', 'Support Processes', 'Process Automation' (selected), 'Approval Processes' (selected), 'Automation Home (Beta)', 'Flows', 'Migrate to Flow', 'Next Best Action', 'Paused And Failed Flow', 'Interviews', 'Post Templates', 'Process Automation Settings', 'Process Builder', 'Workflow Actions' (selected), 'Email Alerts', 'Field Updates', 'Outbound Messages', 'Send Actions', 'Tasks', 'Workflow Rules', and 'Environments'. The main content area is titled 'Approval Processes' and shows a process named 'High_Value_Order_Approval'. The 'Process Definition Detail' section includes fields for Process Name (High_Value_Order_Approval), Unique Name (High_Value_Order_Approval), Description, Entry Criteria (Order: Total Price GREATER THAN 500), Record Editability (Administrator ONLY), Approval Assignment Email Template (Commerce Reorder Portal_Invitation), Initial Submitters (Order Owner), Created By (amisha.patel), and Modified By (amisha.patel). The 'Initial Submission Actions' section contains an action type 'Record Lock' with a description 'Lock the record from being edited'. The 'Approval Steps' section lists a single step: 'Manager Approval' (Step Number 1) with criteria 'Requires manager approval for orders where Total Price > £5,000 before Approval'. The bottom of the screen shows the Windows taskbar with various application icons and the date/time (25-09-2025).

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.

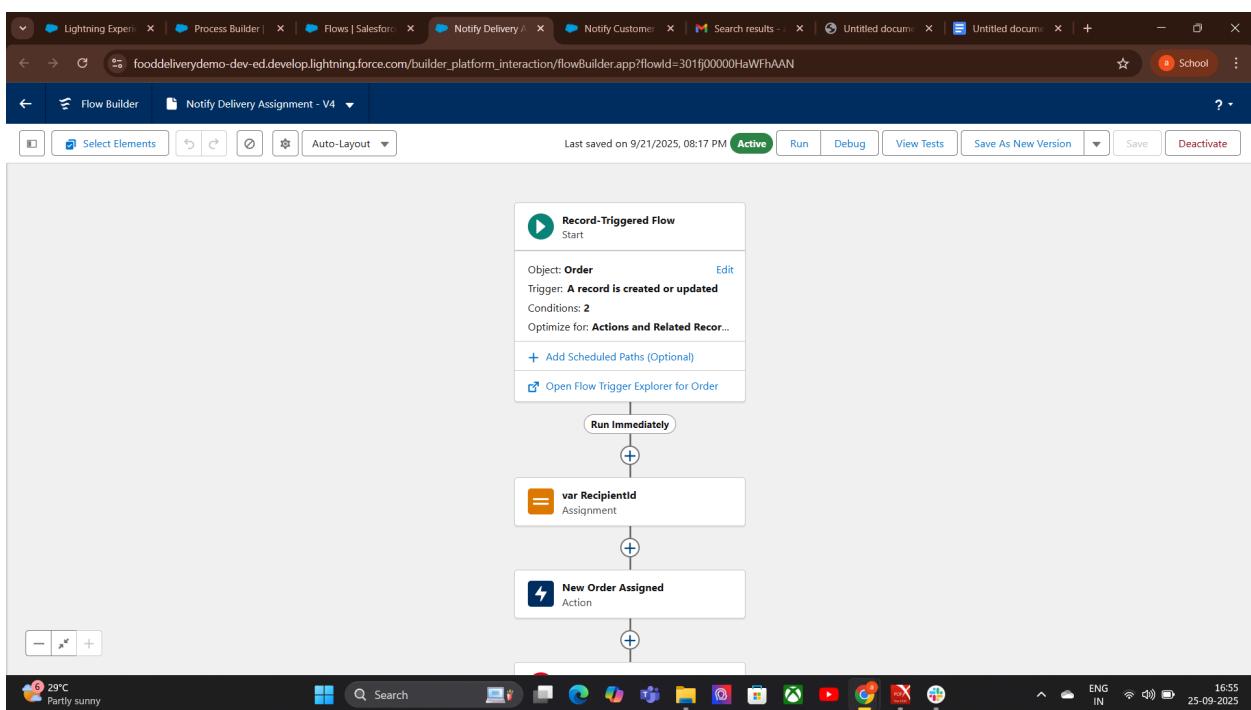
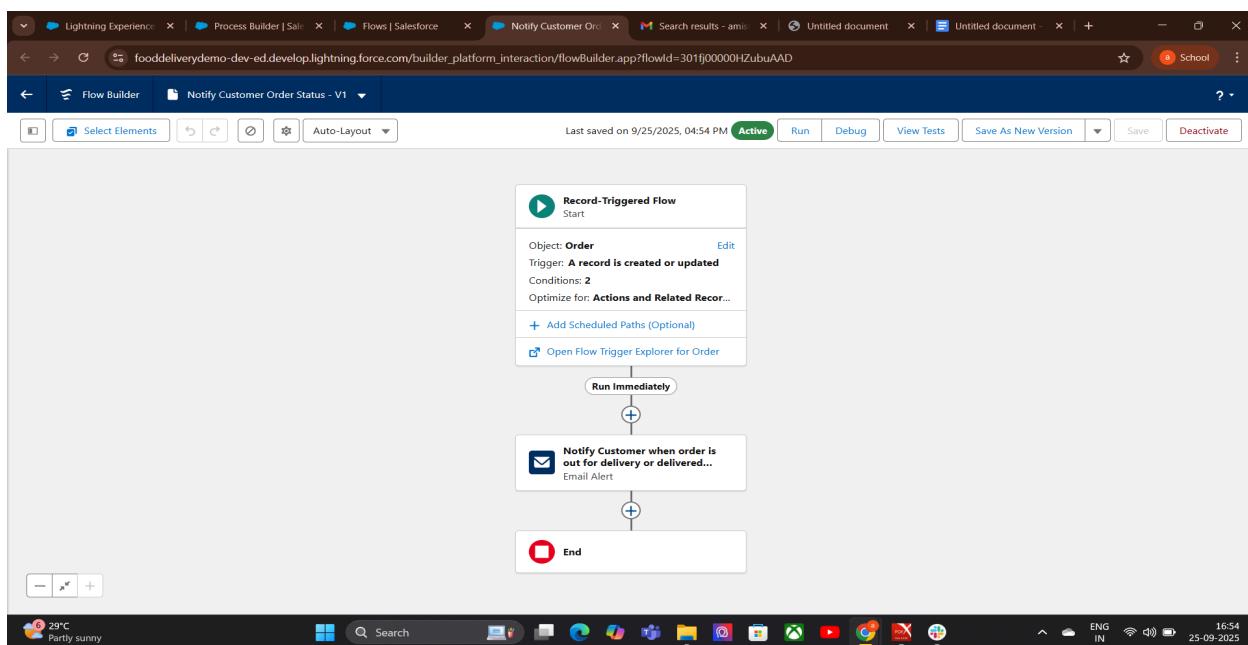
Auto-launched Flow-

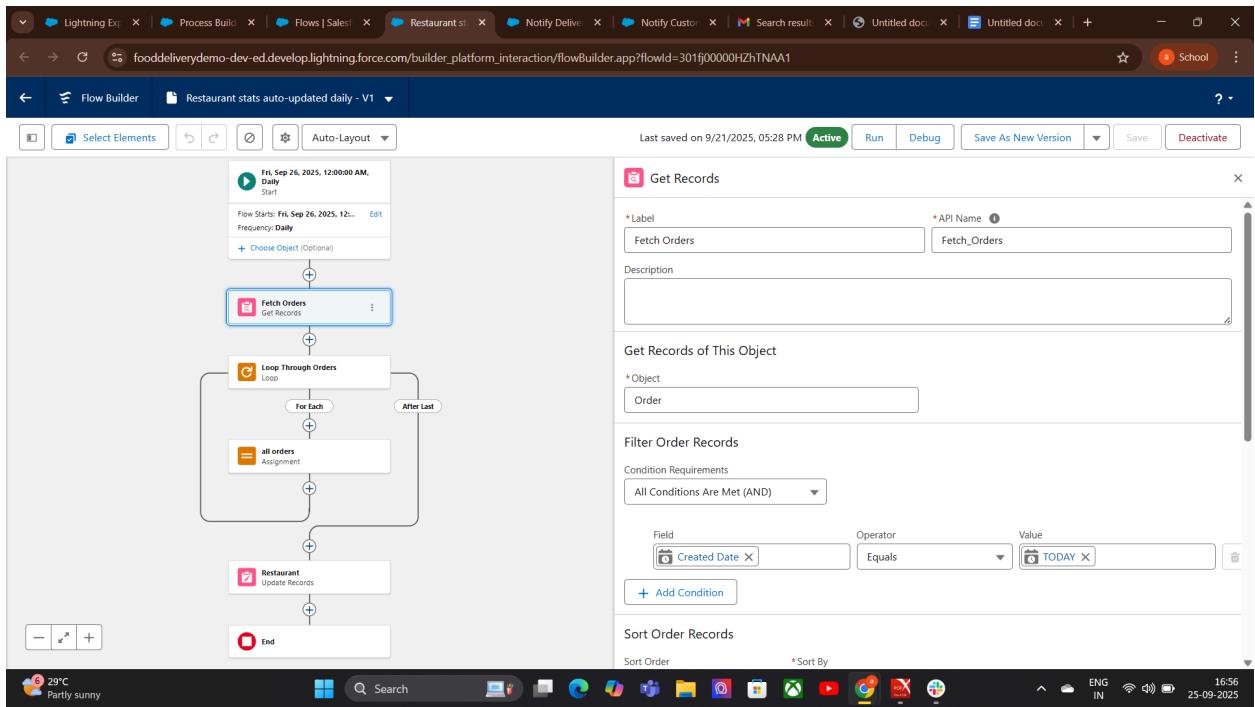
Purpose: Runs in the background without user interaction.

Use case: Triggered by processes, Apex, other flows, or APIs.

Key features:

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

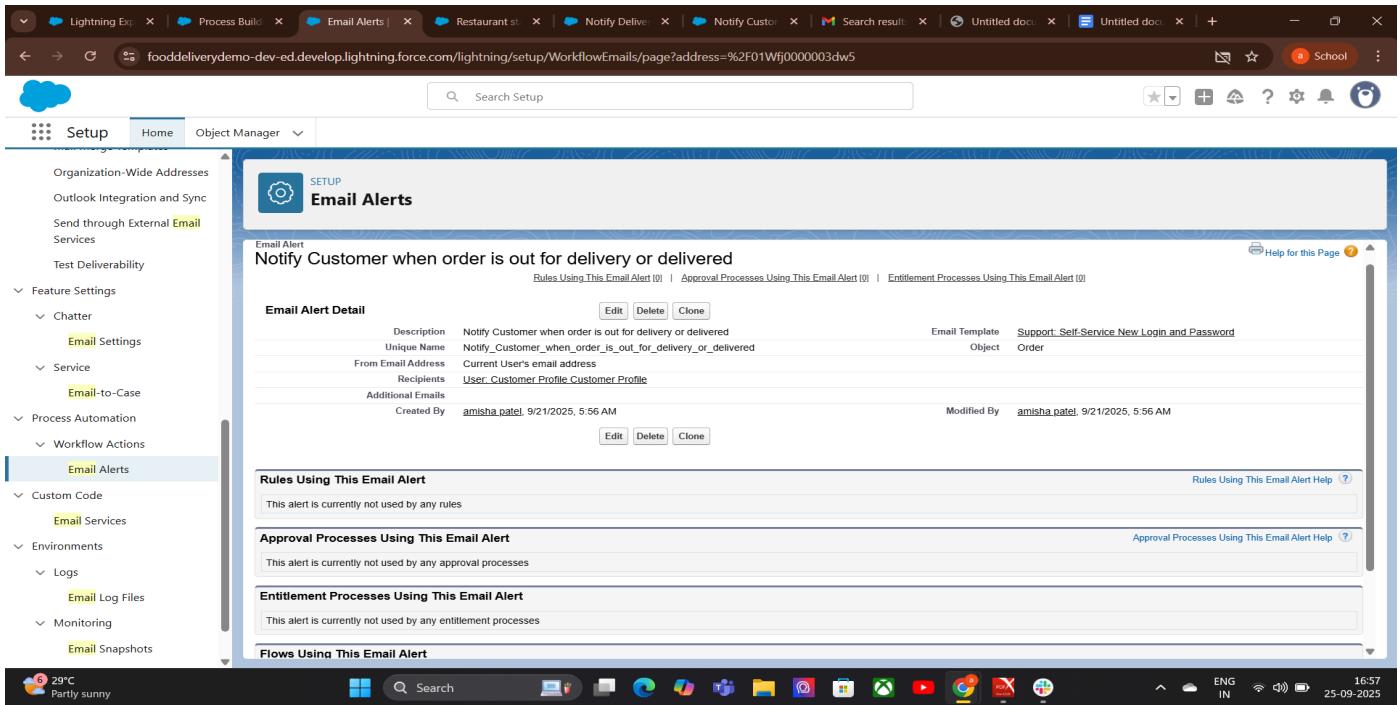




6. Email Alerts:

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.

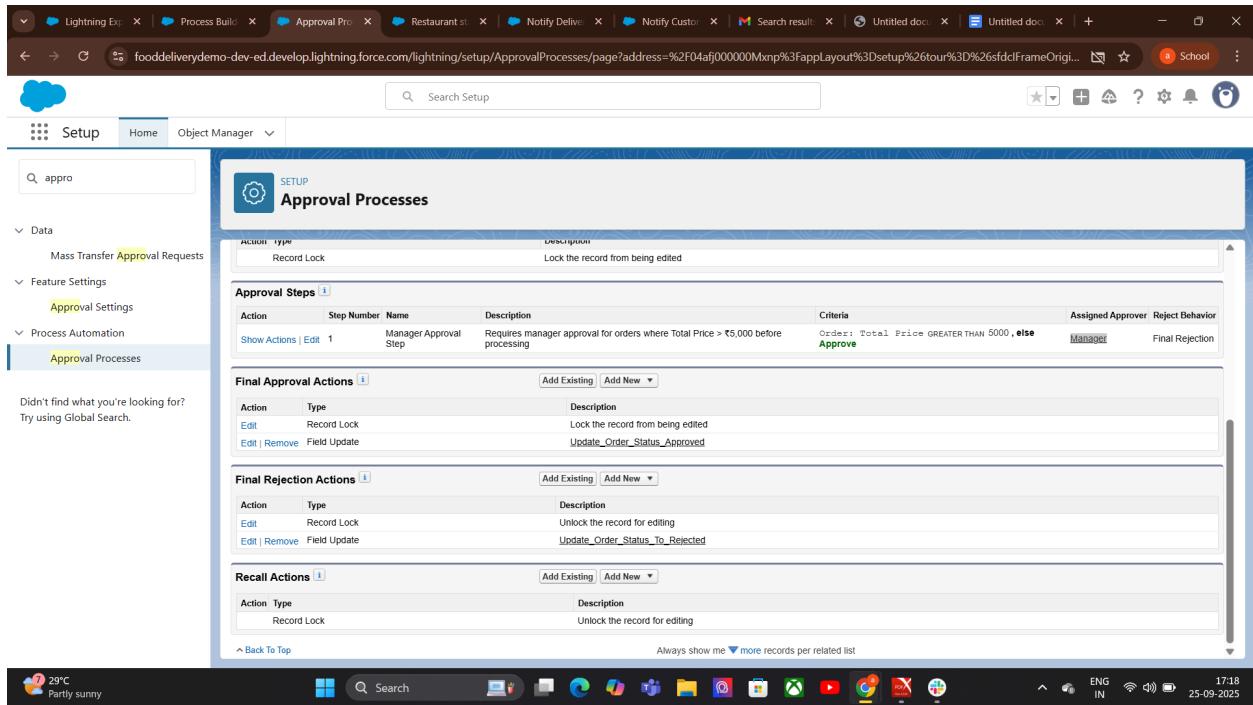


6.Field Updates:

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.



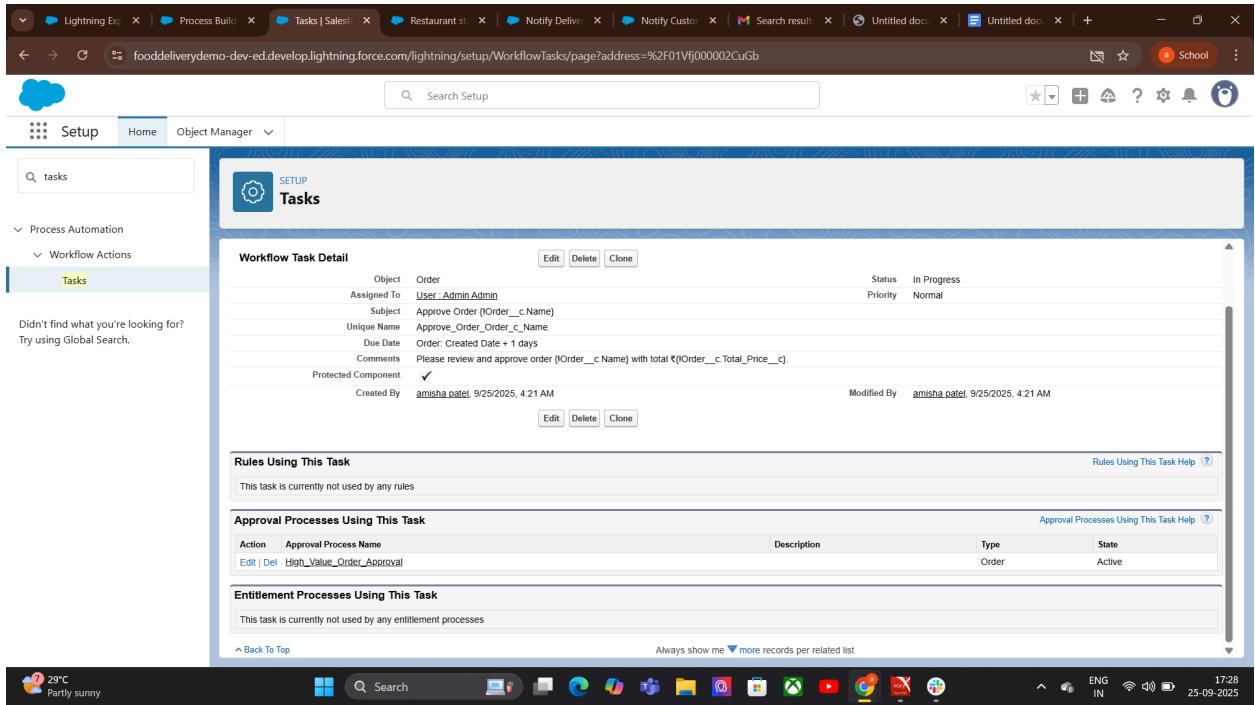
7. Tasks:

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.

The screenshot shows the Salesforce Lightning Experience interface. The top navigation bar has several tabs open, including "Lightning Experience", "Custom Notifications", "Process Builder | Sale", "Food Delivery Home", "Untitled document", "Order | Salesforce", and "Search results - amis". The URL in the address bar is `fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/CustomNotifications/home`. The main content area is titled "Custom Notifications" under the "SETUP" tab. A sidebar on the left lists various setup categories: Apps (Mobile Apps, Salesforce), Environments (Monitoring), API Usage Notifications, Notification Builder (Custom Notifications, Notification Delivery Settings), and Global Search. The "Custom Notifications" section is currently selected. The main content area displays information about custom notifications and a table of existing notification types.

Custom Notification Types

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		✓	✓

Phase 5: Apex Programming(Developer)

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

1. Open Salesforce Developer Console
 - o Click your avatar → Developer Console.
2. Create New Apex Class
 - o Go to File → New → Apex Class.
 - o Give it a name (e.g., OrderHandler).
3. Define the Class
 - o Start with public class ClassName { }.
4. Add Properties (Variables)
 - o 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.

The screenshot shows the Salesforce Developer Console interface. At the top, there's a navigation bar with File, Edit, Debug, Test, Workspace, Help, and a download icon. Below the navigation bar is a tabs section with 'Order.apxc' selected. The main area is a code editor with the following Apex code:

```
1 public class Order {
2     public String orderId;
3     public String customerName;
4     public Decimal totalAmount;
5
6     // Constructor
7     public Order(String id, String name, Decimal amount) {
8         this.orderId = id;
9         this.customerName = name;
10        this.totalAmount = amount;
11    }
12
13    // Method to display order details
14    public void displayOrderDetails() {
15        System.debug('Order ID: ' + orderId);
16        System.debug('Customer Name: ' + customerName);
17        System.debug('Total Amount: ' + totalAmount);
18    }
19 }
```

Below the code editor is a 'Logs' tab. The logs table has columns for User, Application, Operation, Time, Status, Read, and Size. One log entry is shown:

User	Application	Operation	Time	Status	Read	Size
amisha patel	Unknown	/services/data/v64.0/tooling/execute... /services/data/v64.0/tooling/execute... /services/data/v64.0/tooling/execute...	26/09/2025, 02:19:25	Success		5.29 KB

At the bottom of the interface is a Windows taskbar with various icons for system status and applications.

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:

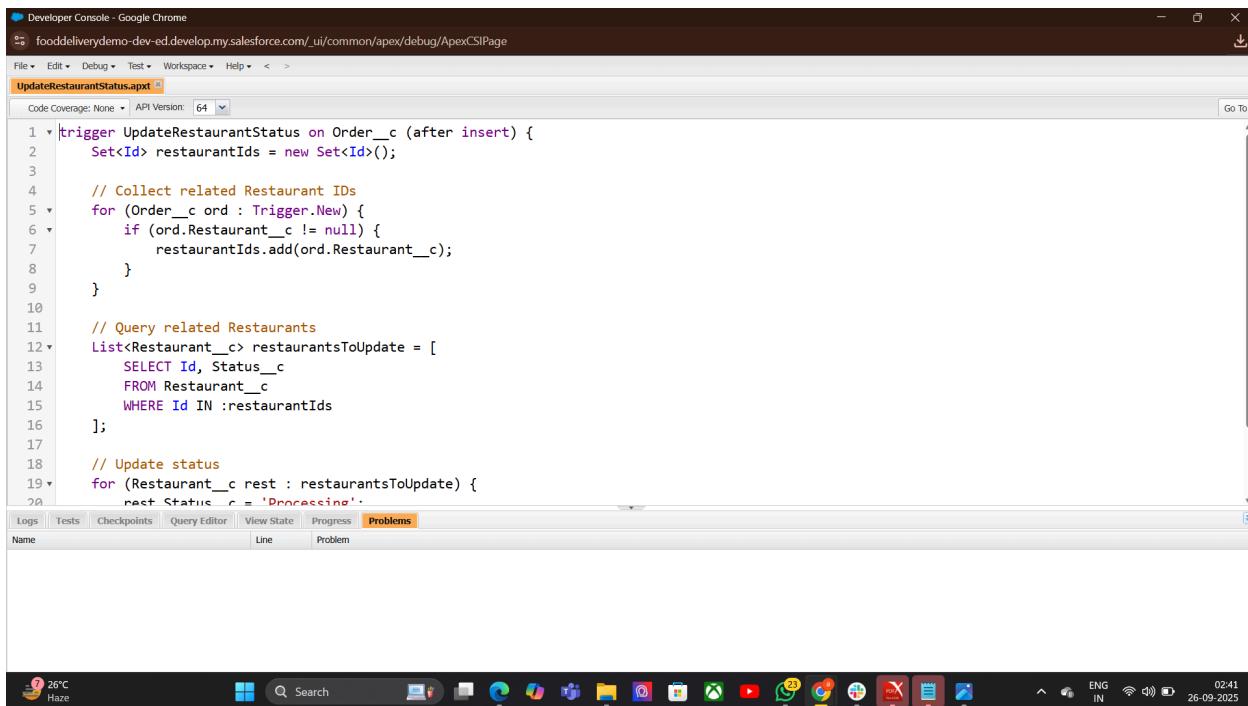
Before Triggers: Modify data before saving (e.g., validation).

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".



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `fooddeliverydemo-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tab title is `UpdateRestaurantStatus.apx`. The code editor contains the following Apex code:

```
1 trigger UpdateRestaurantStatus on Order__c (after insert) {
2     Set<Id> restaurantIds = new Set<Id>();
3
4     // Collect related Restaurant IDs
5     for (Order__c ord : Trigger.New) {
6         if (ord.Restaurant__c != null) {
7             restaurantIds.add(ord.Restaurant__c);
8         }
9     }
10
11    // Query related Restaurants
12    List<Restaurant__c> restaurantsToUpdate = [
13        SELECT Id, Status__c
14        FROM Restaurant__c
15        WHERE Id IN :restaurantIds
16    ];
17
18    // Update status
19    for (Restaurant__c rest : restaurantsToUpdate) {
20        rest.Status__c = 'Processing';
21    }
22}
```

The code defines an After Insert trigger named `UpdateRestaurantStatus` on the `Order__c` object. It collects the IDs of related `Restaurant__c` records, queries them, and then updates their `Status__c` field 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.

The screenshot shows the Salesforce Setup interface with the 'Apex Triggers' page open. The URL is fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/ApexTriggers/page?address=%2F01qfj000001enW5. The page displays the details for the 'OrderTrigger' Apex Trigger. The trigger code is as follows:

```
1trigger OrderTrigger on Order__c (after insert, after update) {
2    if (Trigger.isInsert) {
3        if (Trigger.isUpdate) {
4            OrderTriggerHandler.handleAfterInsert(Trigger.new);
5        }
6        if (Trigger.isUpdate) {
7            OrderTriggerHandler.handleAfterUpdate(Trigger.new, Trigger.oldMap);
8        }
9    }
10}
```

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.**

The screenshot shows the Salesforce Developer Console interface. At the top, there's a navigation bar with links like 'File', 'Edit', 'Debug', 'Test', 'Workspace', 'Help', and tabs for 'Menu_Item__c@7:02 PM', 'Order__c@7:05 PM', 'Order__c@7:06 PM', 'Order__c@7:06 PM', 'Order__c@7:07 PM'. Below the navigation is a query editor window with the following content:

```
SELECT Id, Name FROM Menu_Item__c LIMIT 5
```

Query Results - Total Rows: 5

Id	Name
a03f0000000IDX74AAH	dosa
a03f0000000GT0AAP	Veg Pizza
a03f0000000GT0gAAP	Chicken Sub
a03f0000000GT0hAAP	Pasta
a03f0000000GT0oAAP	Whopper

Below the results are buttons for 'Save Rows', 'Insert Row', 'Delete Row', and 'Refresh Grid'. To the right, there are links for 'Create New', 'Open Detail Page', and 'Edit Page'. A 'Logs' tab is selected, showing the query: 'SELECT Id, Name FROM Menu_Item__c LIMIT 5'. The 'History' section shows the same query was executed previously. The bottom status bar shows the date '25-09-2025' and time '19:07'.

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>();

The screenshot shows the Salesforce Developer Console interface. At the top, there's a navigation bar with links like File, Edit, Debug, Test, Workspace, Help, and tabs for Log executeAnonymous, Log executeAnonymous, and Log executeAnonymous. Below the navigation bar is a toolbar with buttons for This Frame, Executable, Open Log, Execute, and Execute Highlighted. The main area is divided into two sections: 'Execution Log' at the top and 'Enter Apex Code' below it.

Execution Log:

Timestamp	Event	Details
19:31:54:011	USER_DEBUG	[7]DEBUG Order Map: {}
19:31:54:011	USER_DEBUG	[13]DEBUG No orders found.

Enter Apex Code:

```

4 // Create map
5 Map<Id, Order__c> orderMap = new Map<Id, Order__c>();
6
7 System.debug('Order Map: ' + orderMap);
8
9 // Safe access check
10 if (!orderList.isEmpty()) {
11     System.debug('Order Name for first ID:');
12 } else {
13     System.debug('No orders found.');
14 }
15

```

At the bottom of the interface, there are tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently selected. The system tray at the bottom of the screen shows weather information (29°C, Partly sunny), a search bar, and various application icons.

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.

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is fooddeliverydemo-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The page displays the OrderProcessingHandler.apex code, which processes orders. Below the code editor is a test run log table:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	707fj000005Z06F	Thu Sep 25 2025 19:41:38 GM...		0	2

The status bar at the bottom shows weather (29°C, Partly sunny), system icons, and the date/time (19:56, 25-09-2025).

```
1 public class OrderProcessingHandler {
2
3     public static void processOrders() {
4
5         // Step 2.1 - SOQL: Get all orders with Status 'New'
6         List<Order__c> orders = [SELECT Id, Name, Total_Price__c, Status__c, Restaurant__c FROM Order__c WHERE Status__c = 'New'];
7
8         // Step 2.2 - Create Collections
9         Set<Id> restaurantIds = new Set<Id>();
10        Map<Id, Order__c> orderMap = new Map<Id, Order__c>();
11
12        // Step 2.3 - Loop through orders
13        for(Order__c o : orders) {
14
15            // Add restaurant IDs to set
16            restaurantIds.add(o.Restaurant__c);
17
18            // Add order to map
19            orderMap.put(o.Id, o);
20        }
21    }
22}
```

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.

The screenshot shows the Salesforce Developer Console interface. At the top, the URL is fooddeliverydemo-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The main area displays the Apex code for OrderProcessingBatch.apxc:

```

1 global class OrderProcessingBatch implements Database.Batchable<SObject> {
2     global Database.QueryLocator start(Database.BatchableContext BC) {
3         return Database.getQueryLocator(
4             'SELECT Id, Status__c, Total_Price__c FROM Order__c WHERE Status__c = \'Pending\''
5         );
6     }
7
8     global void execute(Database.BatchableContext BC, List<SObject> scope) {
9         List<Order__c> ordersToUpdate = new List<Order__c>();
10
11        for (SObject s : scope) {
12            Order__c ord = (Order__c)s;
13            ord.Status__c = 'In Progress';
14            ordersToUpdate.add(ord);
15        }
16
17        if (!ordersToUpdate.isEmpty()) {
18            update ordersToUpdate;
19        }
20    }

```

Below the code, there are tabs for Logs, Tests, Checkpoints, Query Editor, ViewState, Progress, and Problems. The Tests tab is selected, showing a table of test runs:

Status	Test Run	Enqueued Time	Duration	Failure
✓	707fj000005Zfsx	Thu Sep 25 2025 20:17:26 GM...		0
✓	707fj000005Zo6F	Thu Sep 25 2025 19:41:38 GM...		0

The system status bar at the bottom shows the weather (29°C, Partly sunny), a search bar, and various system icons.

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;
    }
}

```

```
}
```

The screenshot shows the Salesforce Developer Console interface. At the top, the URL is `foodeliverydemo-dev-ed.develop.my.salesforce.com/ui/common/apex/debug/ApexCSIPage`. Below the header, there's a code editor window titled `UpdateOrdersQueueable.apc` with the following Apex code:

```
1 public class UpdateOrdersQueueable implements Queueable {
2     private List<Id> orderIds;
3
4     public UpdateOrdersQueueable(List<Id> orderIds) {
5         this.orderIds = orderIds;
6     }
7
8     public void execute(QueueableContext context) {
9         List<Order__c> orders = [SELECT Id, Status__c FROM Order__c WHERE Id IN :orderIds];
10    for (Order__c o : orders) {
11        o.Status__c = 'Processed';
12    }
13    update orders;
14 }
15 }
```

Below the code editor is a test results table:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	707fj000005Zfsx	Thu Sep 25 2025 20:17:26 GM...		0	2
✓	707fj000005ZQ6F	Thu Sep 25 2025 19:41:38 GM...		0	2

At the bottom of the interface, there's a Windows taskbar showing various application icons and system status.

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.

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `foodeliverydemo-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage`. The tab is titled "Scheduled.apxc". The code editor displays the following Apex class:

```
1 global class OrderBatchProcess implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         // Code to process orders
4         System.debug('Running Scheduled Apex: OrderBatchProcess');
5
6         // Example: call batch apex
7         Database.executeBatch(new OrderBatchJob());
8     }
9 }
10
```

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.

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is fooddeliverydemo-dev-ed.develop.my.salesforce.com/ui/common/apex/debug/ApexCSIPage. The page displays the code for the OrderEmailSender class. The code includes a future method for sending order confirmations, querying for order details, and sending an email. A catch block handles exceptions. The developer has added comments and debug statements. The bottom of the screen shows the Windows taskbar with various icons and the system tray.

```
1 public class OrderEmailSender {
2
3     @future
4     public static void sendOrderConfirmation(String orderId) {
5         try {
6             // Fetch order details
7             Order__c ord = [SELECT Id, Name, Customer_Email__c
8                             FROM Order__c
9                             WHERE Id = :orderId LIMIT 1];
10
11            // Simulate sending email (replace with actual email logic)
12            System.debug('Sending order confirmation email to: ' + ord.Customer_Email__c);
13
14            // Example: Email logic here
15            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
16            email.setToAddresses(new String[] {ord.Customer_Email__c});
17            email.setSubject('Order Confirmation - ' + ord.Name);
18            email.setPlainTextBody('Your order has been received and is being processed.');
19            Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});
20
21        } catch(Exception e) {
22            System.debug('Error sending order email: ' + e.getMessage());
23        }
24    }
25}
```

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`.

The screenshot shows a web browser window with multiple tabs open. The active tab is 'OrderExceptionHandler.apxc' in the 'Developer Console'. The code editor contains the following Apex class:

```
18 update orders;
19
20 } catch(DmlException e) {
21     System.debug('DML Exception: ' + e.getMessage());
22 } catch(QueryException e) {
23     System.debug('Query Exception: ' + e.getMessage());
24 } catch(CustomOrderException e) {
25     System.debug('Custom Exception: ' + e.getMessage());
26 } catch(Exception e) {
27     System.debug('General Exception: ' + e.getMessage());
28 } finally {
29     System.debug('Order processing attempt finished.');
30 }
31
32 // Custom exception definition
33 public class CustomOrderException extends Exception {}
34
35 }
36
```

Below the code editor is a 'Logs' section with tabs for 'Logs', 'Tests' (which is selected), 'Checkpoints', 'Query Editor', 'View State', 'Progress', and 'Problems'. The 'Tests' tab shows two test runs:

Status	Test Run	Enqueued Time	Duration	Failures	Total
✓	707fj000005ZGFK	Thu Sep 25 2025 20:56:35 GM...		0	2
✓	707fj000005ZFsx	Thu Sep 25 2025 20:17:26 GM...		0	2

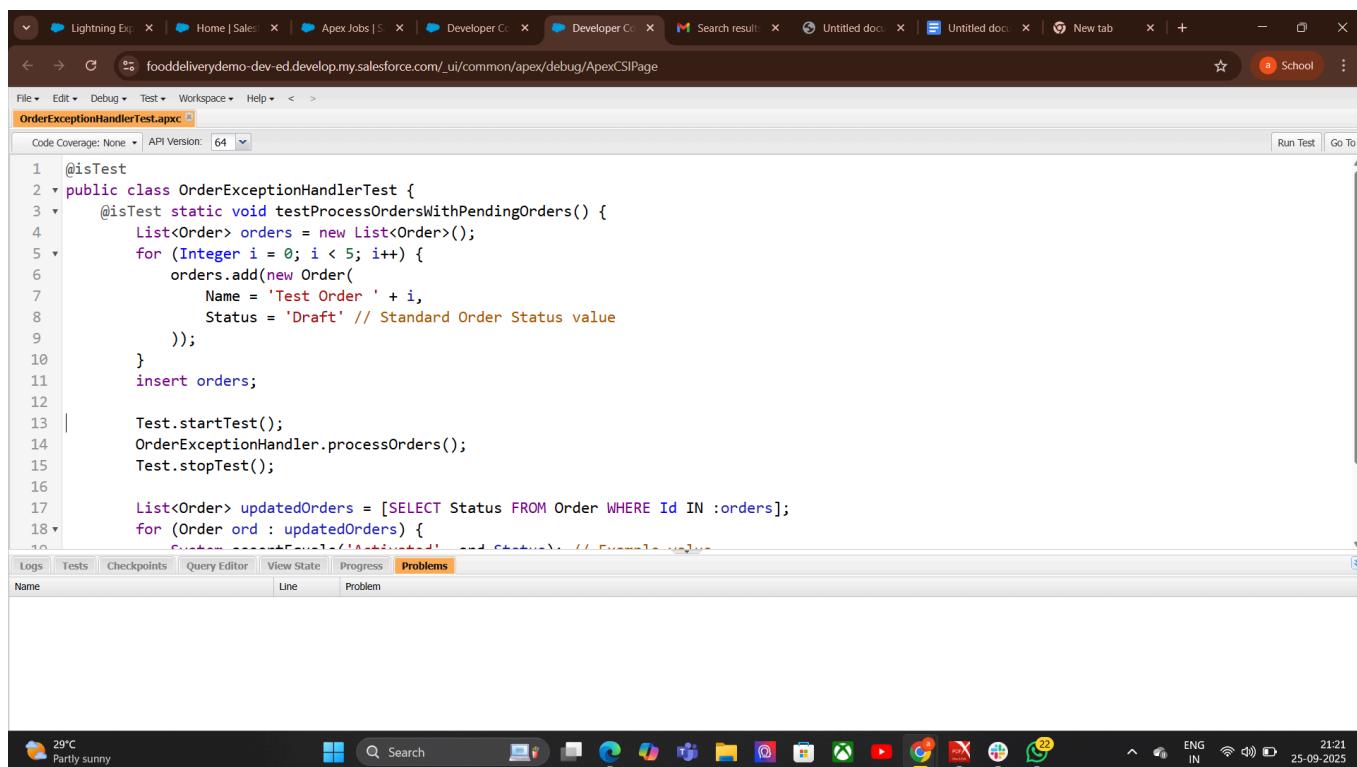
The system status bar at the bottom of the browser window displays weather information (29°C, Partly sunny), a search bar, and various system icons.

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
 - It calls `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`.



The screenshot shows the Salesforce IDE interface with the following details:

- Title Bar:** Lightning Ex..., Home | Sales..., Apex Jobs | S..., Developer Co..., Developer Co..., Search result..., Untitled doc..., Untitled doc..., New tab...
- Toolbar:** File, Edit, Debug, Test, Workspace, Help
- Code Editor:** The active file is `OrderExceptionHandlerTest.apxc`. The code implements a test class for the `OrderExceptionHandler` class. It creates a list of 5 test orders, inserts them, starts a test, calls the processOrders method, stops the test, and then queries the updated orders to check their status.
- API Version:** Set to 64
- Run Test:** A button in the top right corner of the code editor.
- Logs, Tests, Checkpoints, Query Editor, View State, Progress, Problems:** Tabs at the bottom of the code editor.
- Bottom Status Bar:** Shows weather (29°C, Partly sunny), system icons (Search, Task View, File Explorer, Paint, Photos, Taskbar, YouTube, Google Chrome, Microsoft Edge, Microsoft Store, Xbox), and system status (ENG IN, 25-09-2025, 21:21).

```

1  @isTest
2  public class OrderExceptionHandlerTest {
3    @isTest static void testProcessOrdersWithPendingOrders() {
4      List<Order> orders = new List<Order>();
5      for (Integer i = 0; i < 5; i++) {
6        orders.add(new Order(
7          Name = 'Test Order ' + i,
8          Status = 'Draft' // Standard Order Status value
9        ));
10    }
11    insert orders;
12
13    Test.startTest();
14    OrderExceptionHandler.processOrders();
15    Test.stopTest();
16
17    List<Order> updatedOrders = [SELECT Status FROM Order WHERE Id IN :orders];
18    for (Order ord : updatedOrders) {
19

```

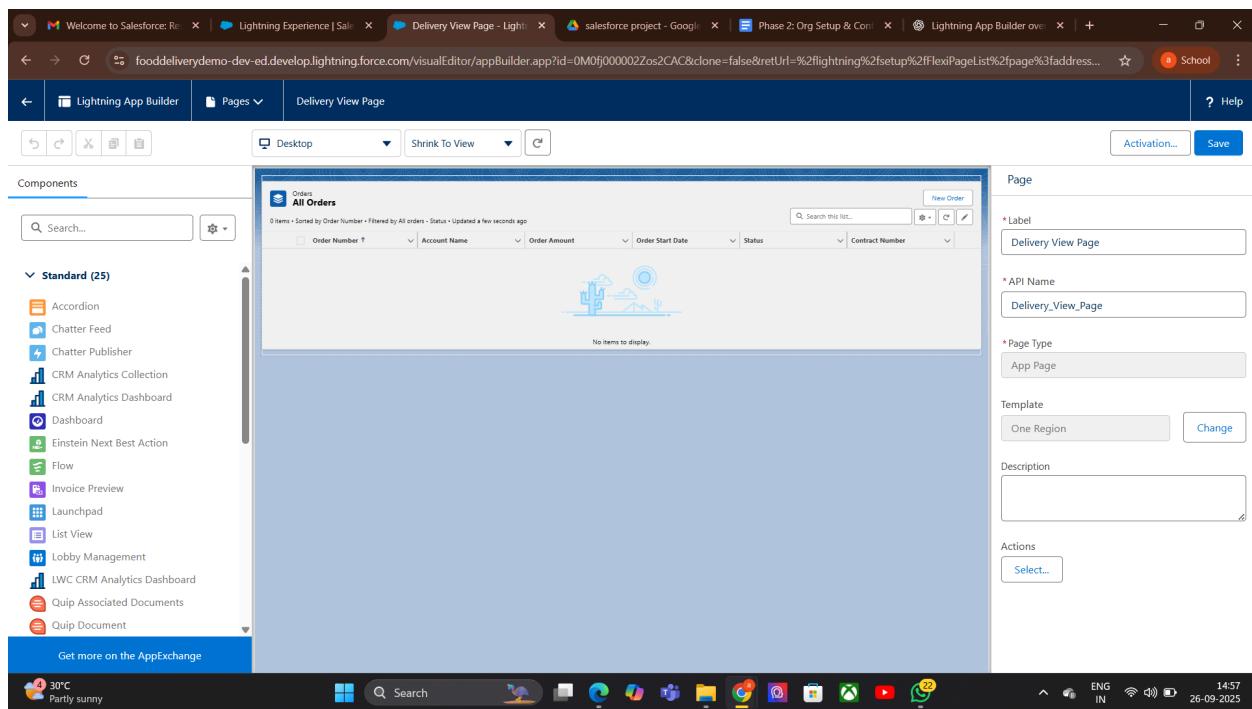
Phase 6: User Interface Development

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

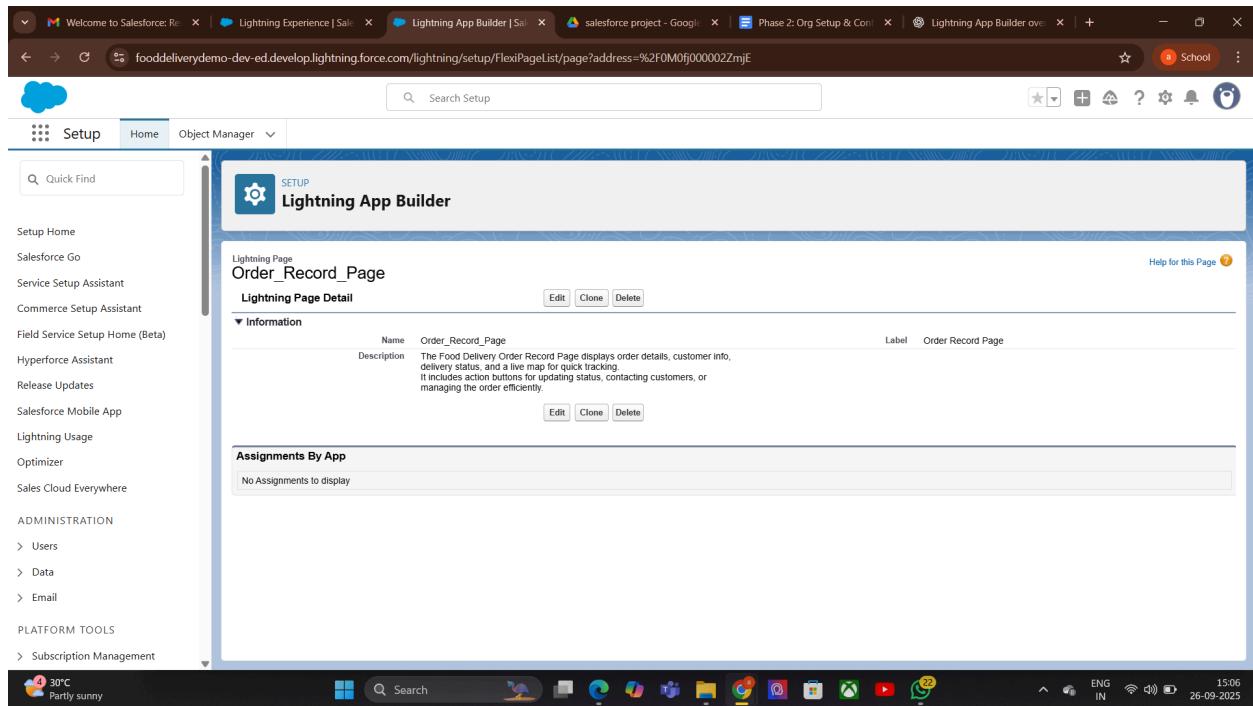
- Go to Setup → Search for *Lightning App Builder*.
- Click New.
- Select page type (App Page, Home Page, Record Page, etc.).
- Choose a template layout.
- Drag and drop components onto the canvas.
- Configure component properties.
- Save and activate the page (set for org, app, or profile-specific visibility).



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).

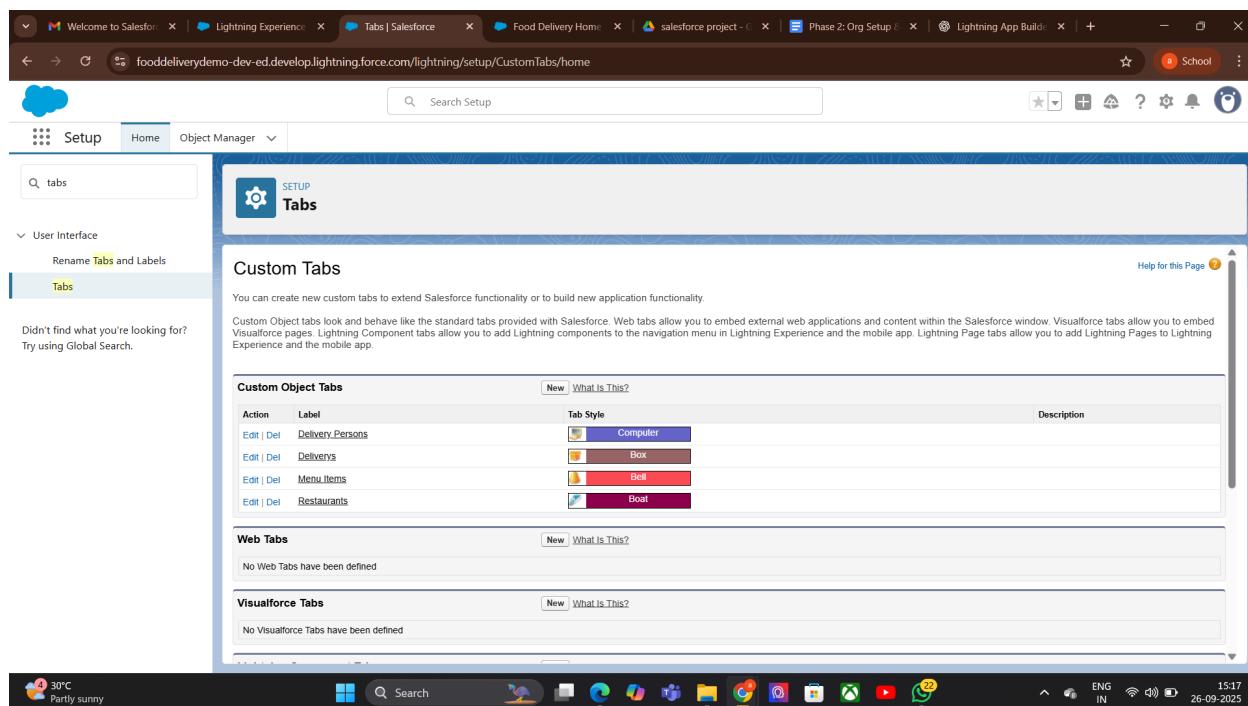


3. Tabs:

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

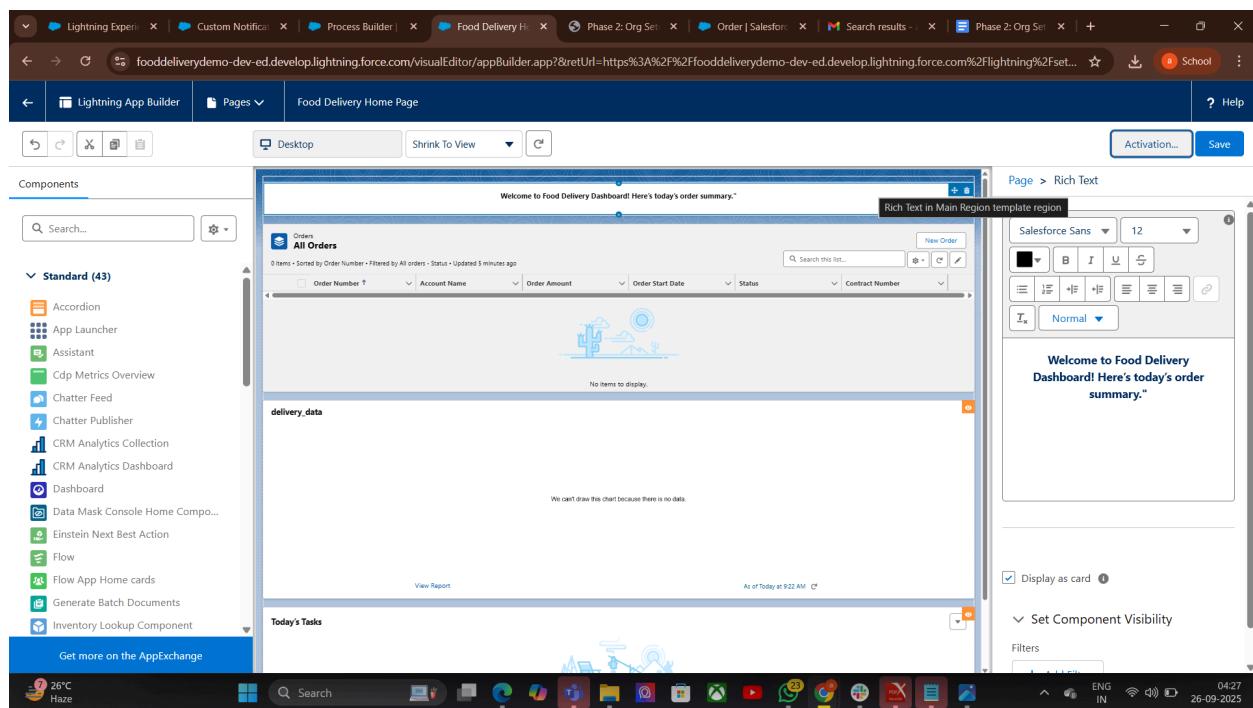
Tabs could be:

- **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:

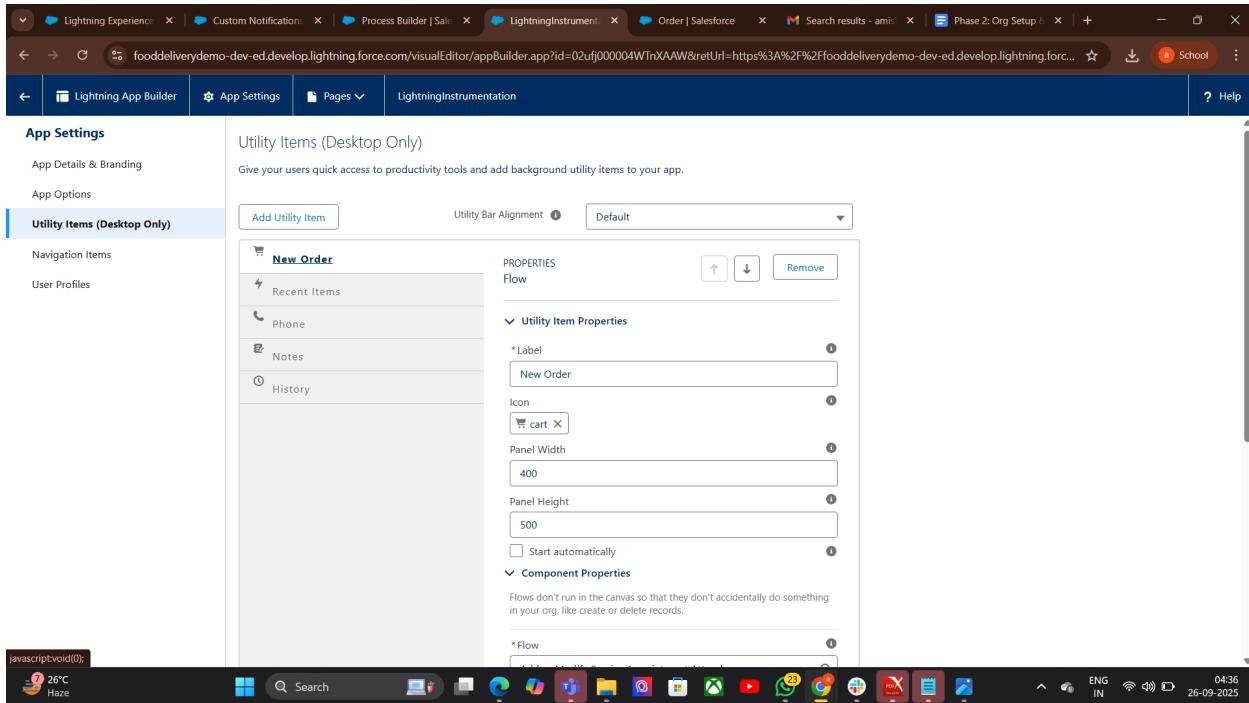
- Definition: Custom layouts for the Salesforce Home Page that organize components and data for users.
- Purpose: Provide quick access to important information, reports, and tools.
- Examples: Dashboards, recent records, tasks, notifications.
- Usage: Customize for roles, profiles, or apps to improve 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`).

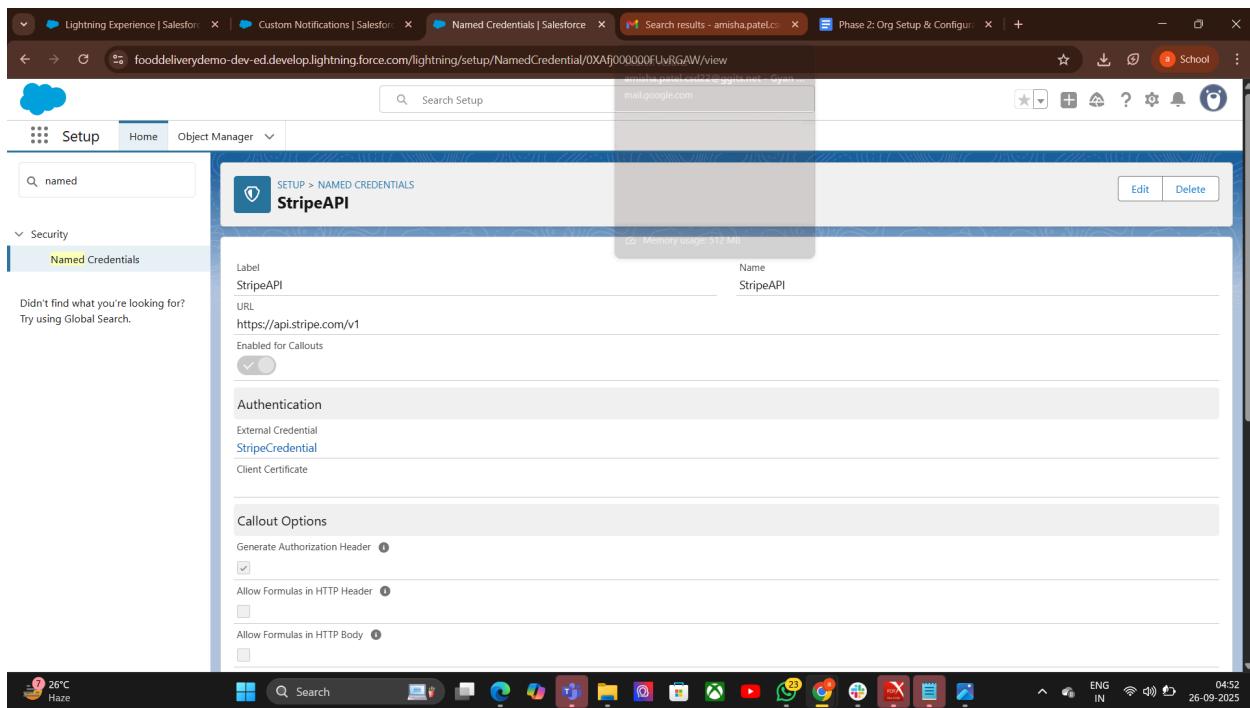
Authentication Types → Supports Password, OAuth 2.0, JWT, AWS Signature, etc.

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 →

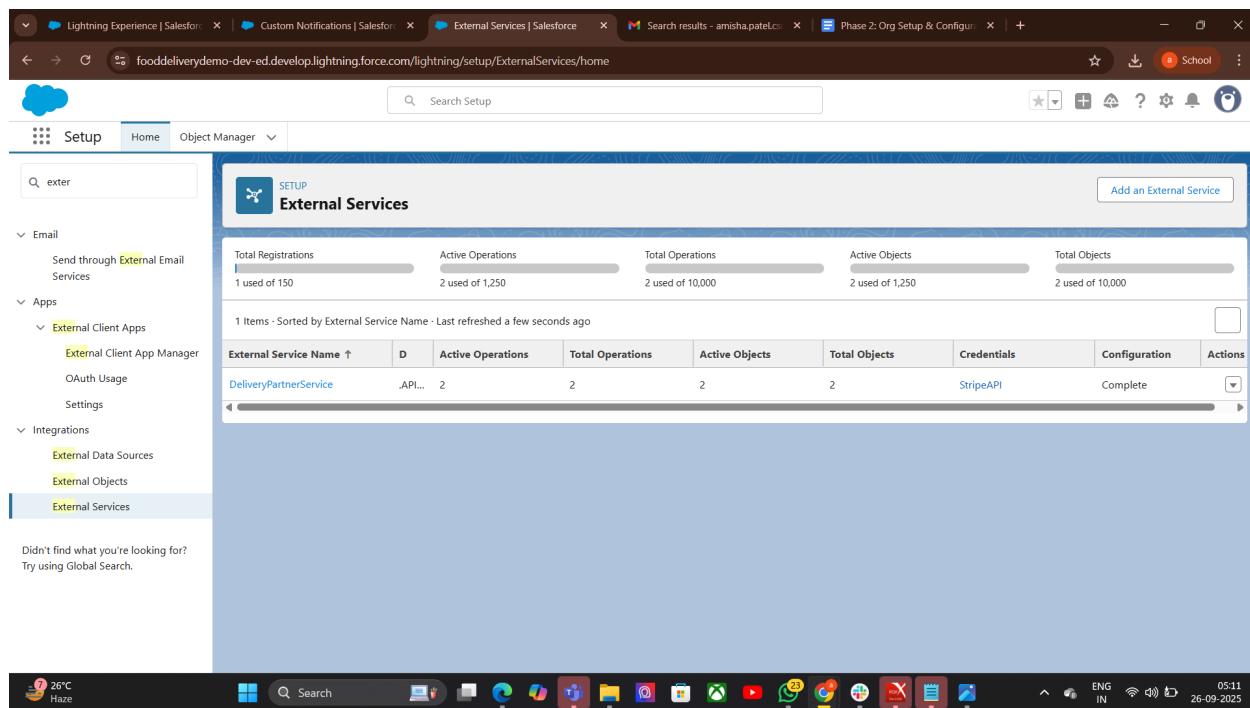
- Go to Setup → **External Services** → New.
- Upload/OpenAPI schema.
- Link it with the **Named Credential**.

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.

```

1 * public class DeliveryPartnerCallout {
2     @future(callout=true)
3     public static void sendOrder(String orderId) {
4         HttpRequest req = new HttpRequest();
5         req.setEndpoint('callout:DeliveryPartnerAPI/v1/orders');
6         req.setMethod('POST');
7         req.setHeader('Content-Type', 'application/json');
8         req.setBody('{ "orderId": "'+orderId+'" }');
9
10        Http http = new Http();
11        HttpResponse res = http.send(req);
12
13        System.debug(res.getBody());
14    }
15 }

```

User	Application	Operation	Time	Status	Read	Size
amisha patel	Unknown	ApexTestHandler	26/09/2025, 05:54:21	Success	Unread	523 bytes
amisha patel	Unknown	ApexTestHandler	26/09/2025, 05:54:20	Success		2.2 KB

4. Platform Events:

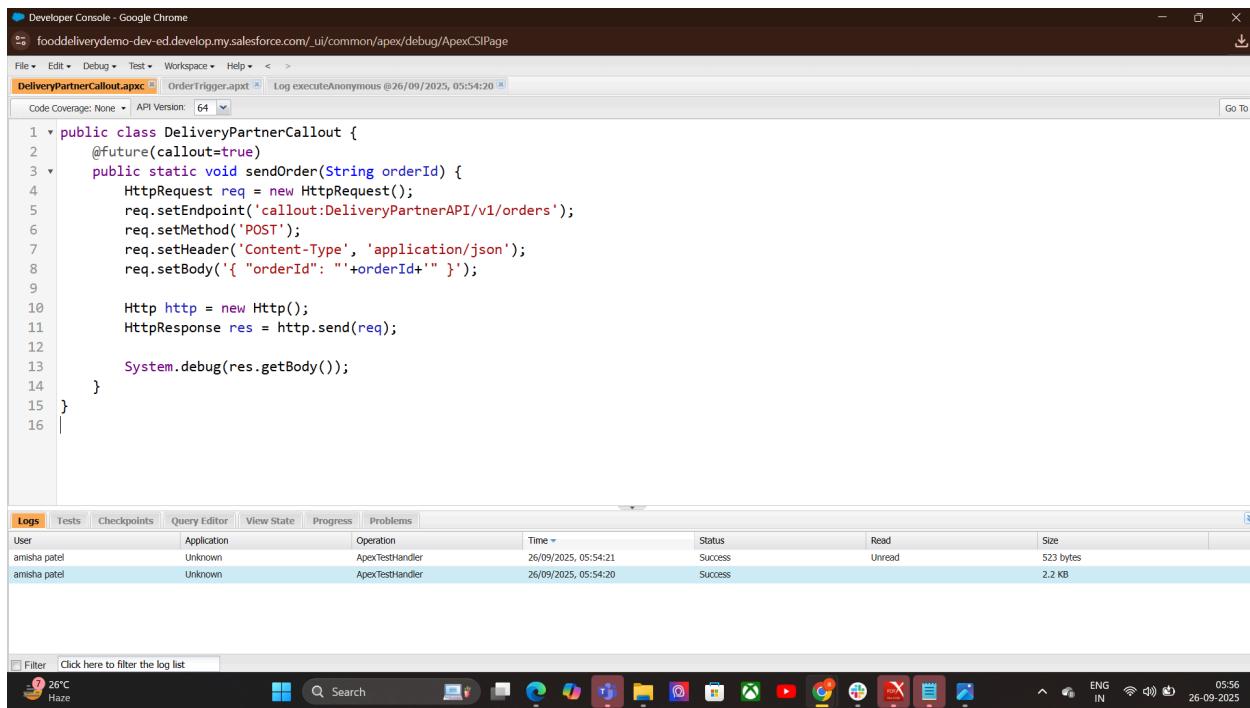
Purpose

- 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.



```
1 public class DeliveryPartnerCallout {
2     @future(callout=true)
3     public static void sendOrder(String orderId) {
4         HttpRequest req = new HttpRequest();
5         req.setEndpoint('callout:DeliveryPartnerAPI/v1/orders');
6         req.setMethod('POST');
7         req.setHeader('Content-Type', 'application/json');
8         req.setBody('{ "orderId": "' +orderId+'" }');
9
10        Http http = new Http();
11        HttpResponse res = http.send(req);
12
13        System.debug(res.getBody());
14    }
15
16 }
```

User	Application	Operation	Time	Status	Read	Size
amisha patel	Unknown	ApexTestHandler	26/09/2025, 05:54:21	Success	Unread	523 bytes
amisha patel	Unknown	ApexTestHandler	26/09/2025, 05:54:20	Success		2.2 KB

5. Change Data Capture:

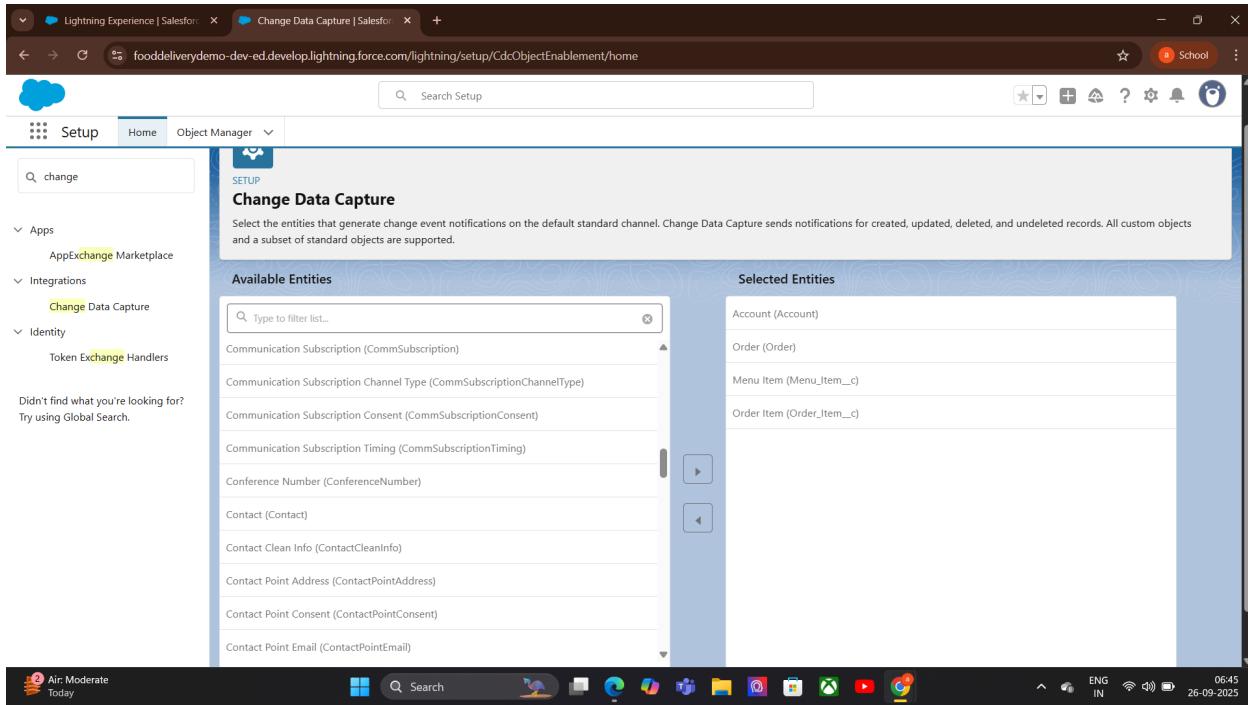
Purpose

- 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.



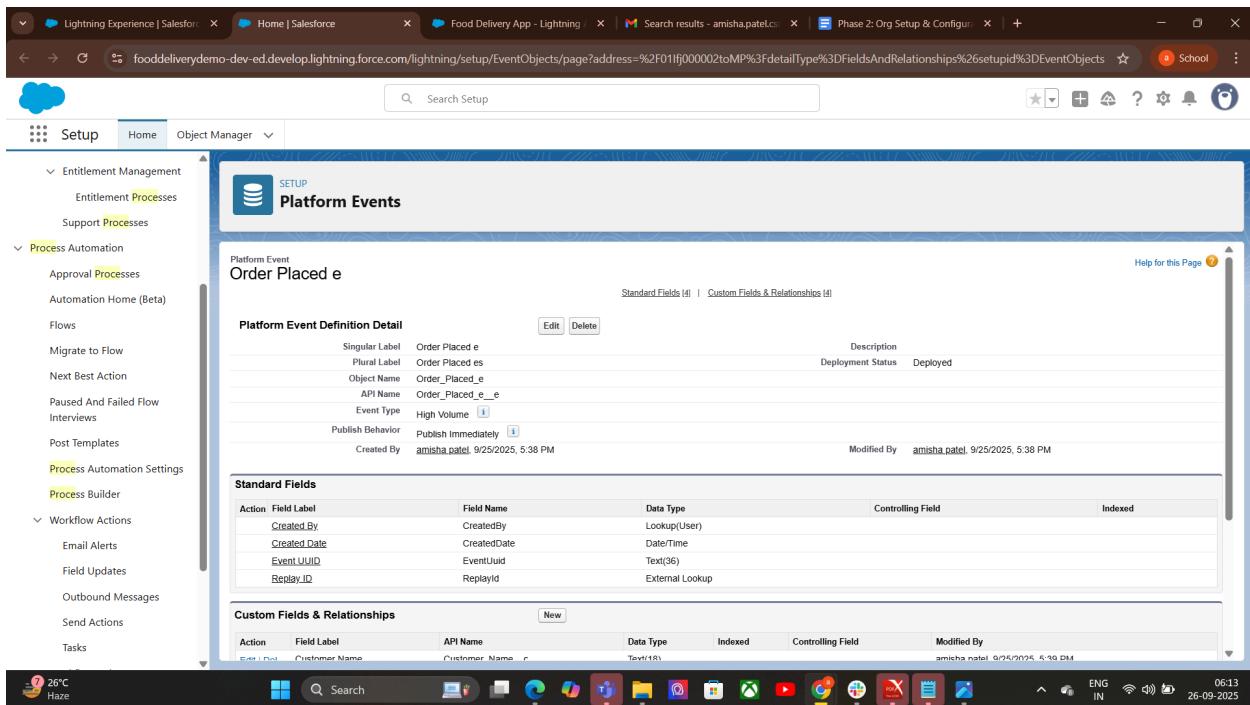
6. Platform Events:

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



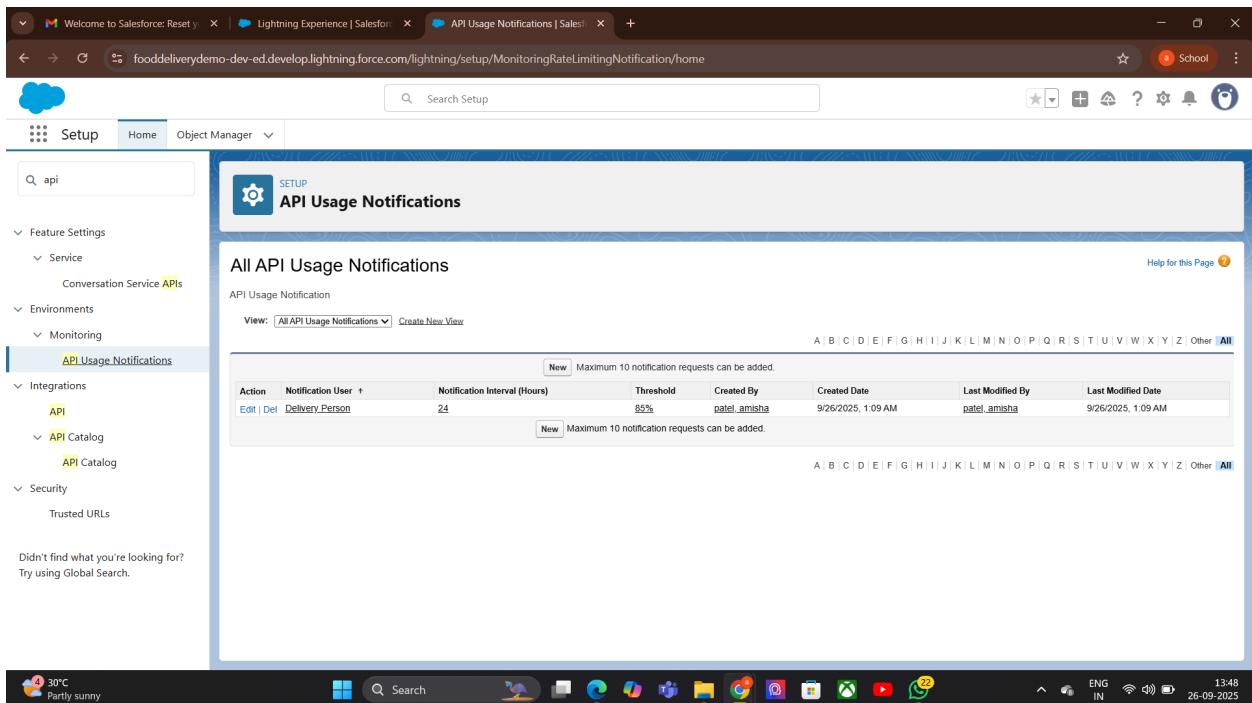
7.API Limits:

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:

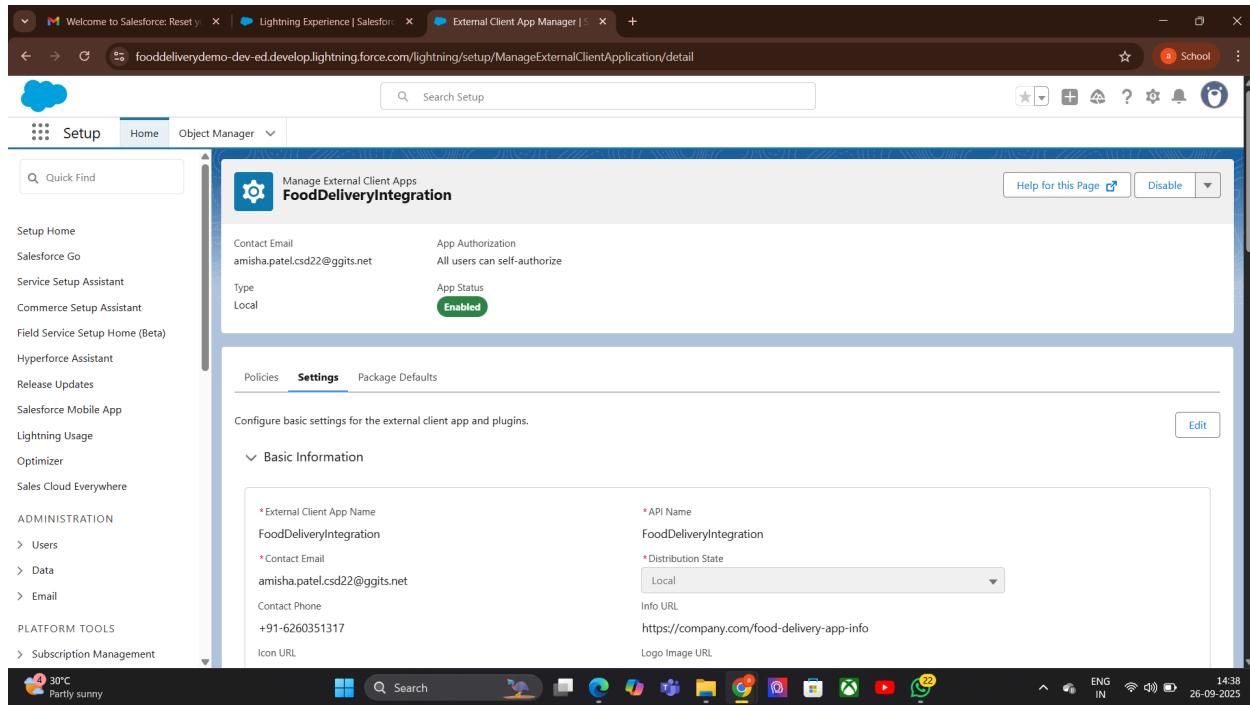
Purpose

- 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.

Salesforce Setup screenshot showing the Remote Site Settings page.

The URL in the browser is: `fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/SecurityRemoteProxy/page?address=%2F0rpfj000001583t`

The page title is: **Remote Site Settings**

The sidebar navigation includes:

- Custom Code
- Remote Access (selected)
- Security
- Remote Site Settings (selected)

The main content area displays the **Remote Site Details** for the entry "FoodDeliveryAPI".

Field	Value	Action
Remote Site Name	FoodDeliveryAPI	Edit Delete Clone
Remote Site URL	https://api.restaurantinventory.com	
Disable Protocol Security	<input type="checkbox"/>	
Description	Remote site for Food Delivery API	
Active	<input checked="" type="checkbox"/>	
Created By	amisha.patel	Created on 9/26/2025, 2:14 AM

Page footer:

- 30°C Partly sunny
- Search bar
- Windows taskbar icons (File Explorer, Edge, File, Task View, etc.)
- System tray: ENG IN, 14:44, 26-09-2025

Phase 8: Data Management & Deployment:

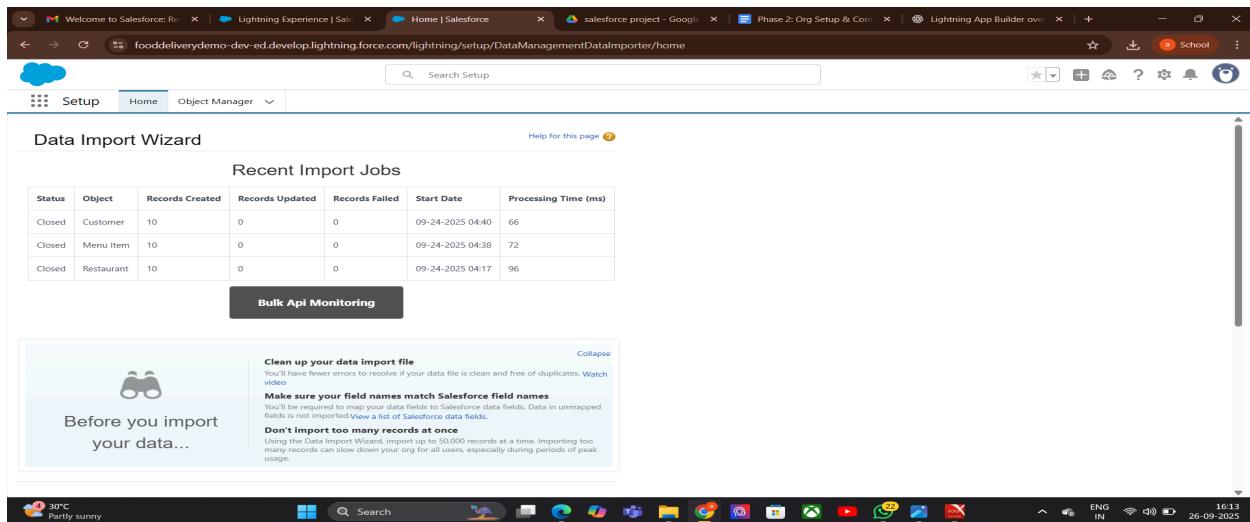
1. Data Import Wizard:

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.



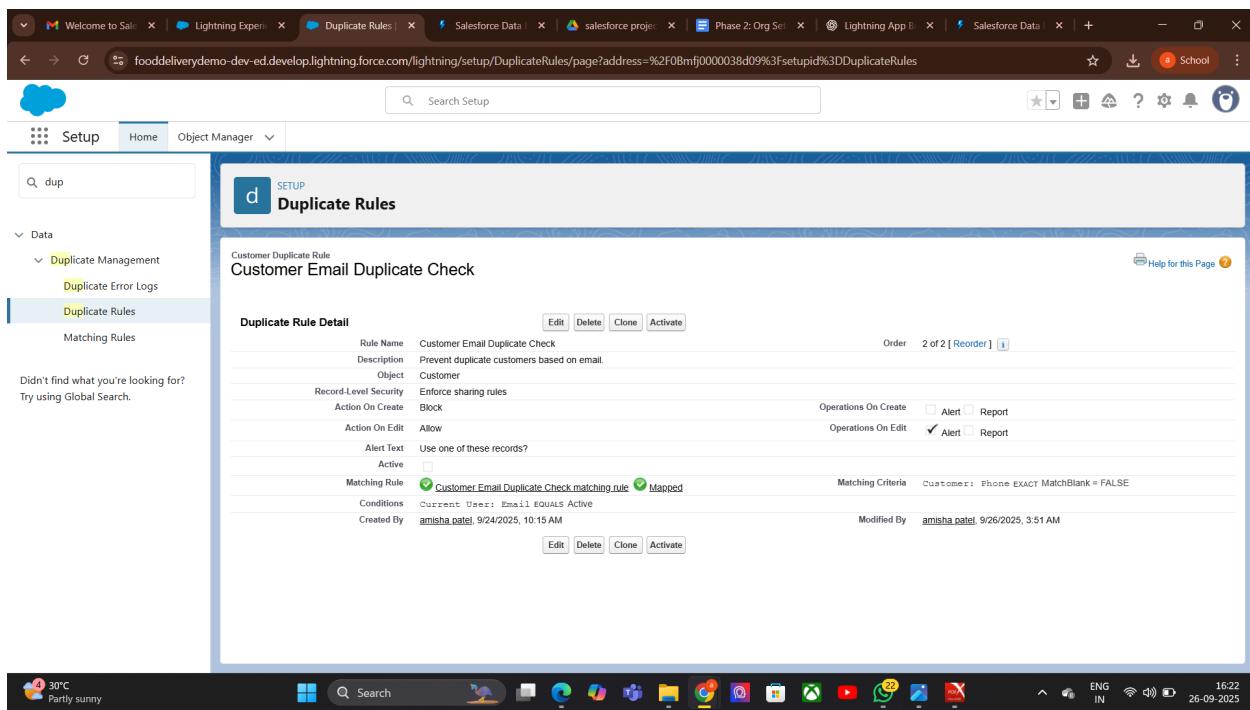
2. Duplicate Rules:

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.



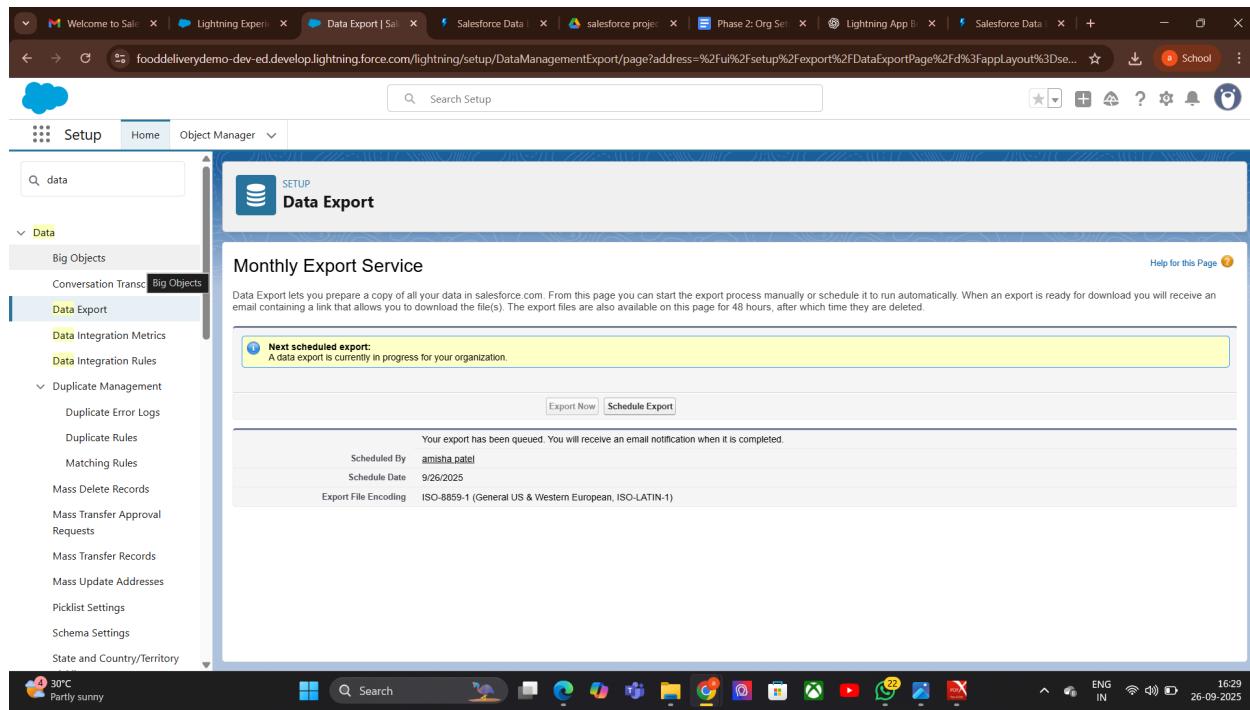
3.Data Export :

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.

The screenshot shows the Salesforce Package Manager interface. The URL in the browser is fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/Package/03Jf0000001pZyv. The page title is "SETUP Package FoodDeliveryApp". The package details show it was created by "amisha.patel" on 9/26/2025, 4:22 AM. It has no dependencies and is owned by "Food_Delivery_App". The components section lists several components: "All" (Menu Item, List View, Included By: Menu Item), "All" (Delivery Person, List View, Included By: Delivery_Person), "All" (Restaurant, List View, Included By: Restaurant), "All" (Delivery, List View, Included By: Delivery), "All Restaurants" (Restaurant, List View, Included By: Restaurant), "Delivery" (Tab, Included By: Food_Delivery_App), "Delivery" (Custom Object, Included By: Delivery), "Delivery Number Layout" (Delivery, Custom Field, Included By: Delivery), and "Delivery Number Layout" (Delivery, Pane Layout, Included By: Delivery). The left sidebar shows "Installed Packages" and "Package Manager" selected. The bottom status bar shows the weather as 30°C partly sunny, the date as 26-09-2025, and the time as 16:54.

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

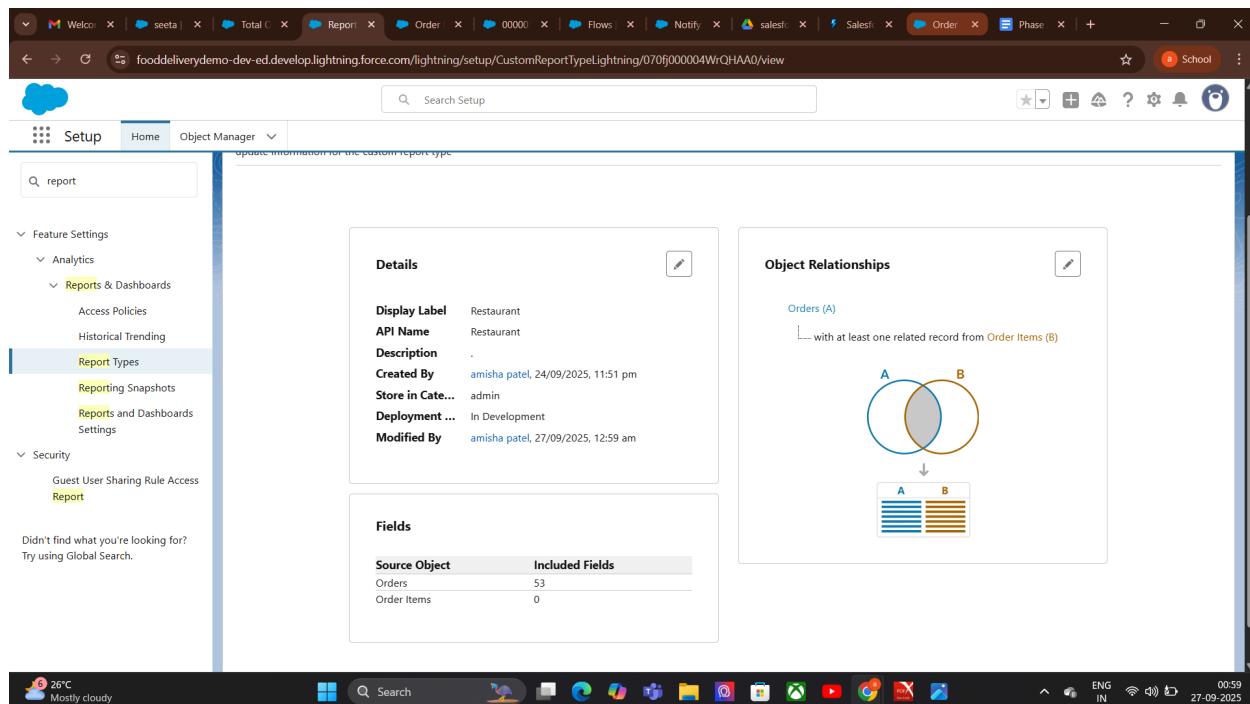
1. Report Types:

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.



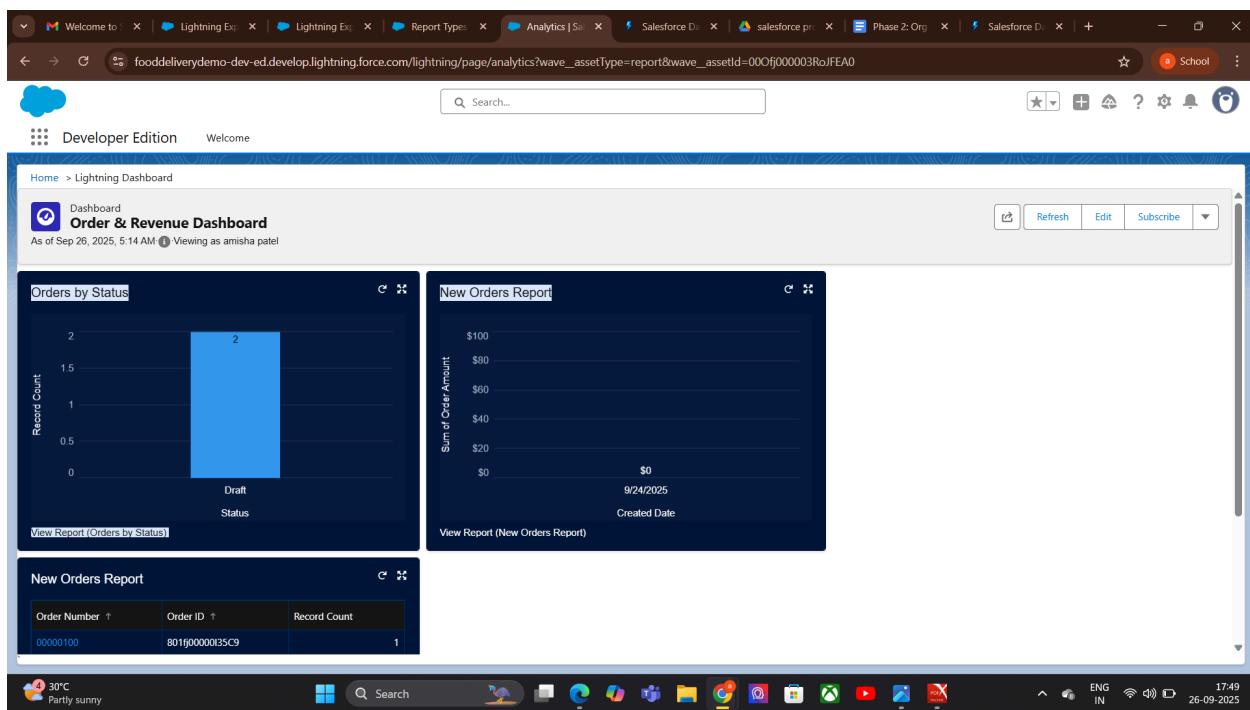
2. Dashboards:

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:

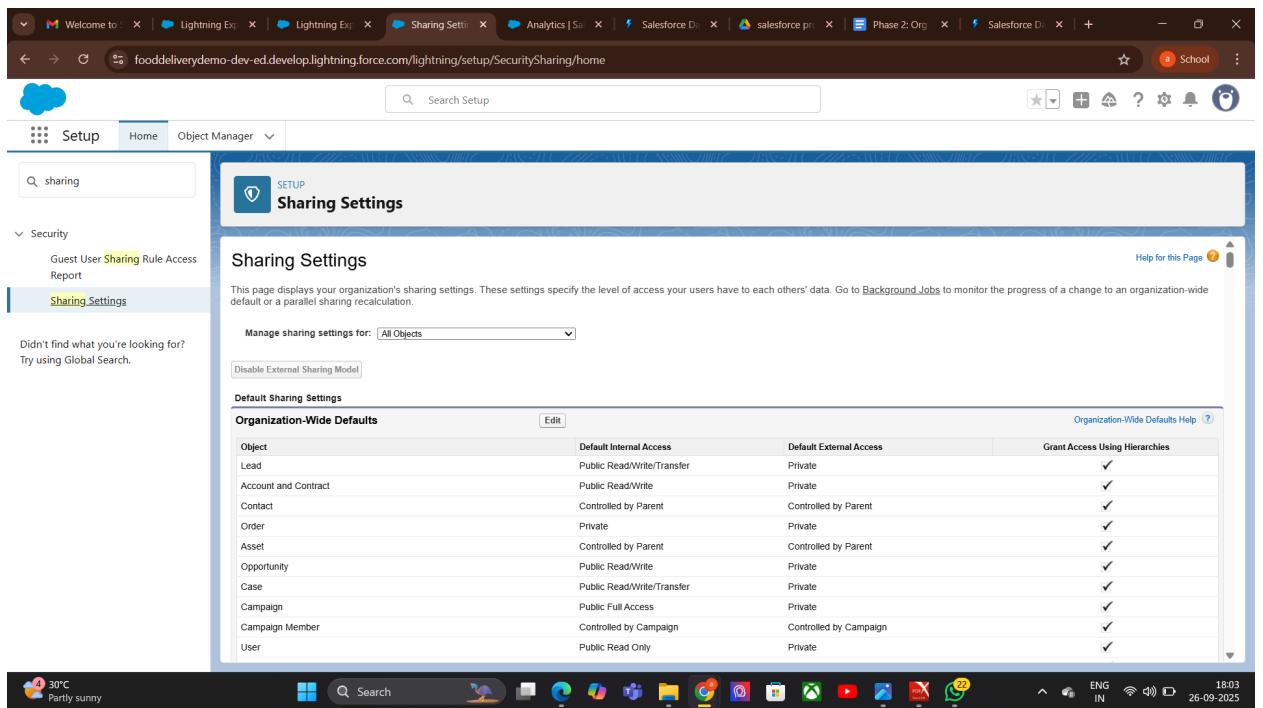
Purpose

- 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.

The screenshot shows the Salesforce Setup interface with the following details:

- Page URL:** fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01Jfj000002mpdN/fieldsAndRelationships/00Nfj00001vNrv1/view
- Section:** SETUP > OBJECT MANAGER
- Object:** Order
- Field Name:** Delivery Time
- Field Label:** Delivery Time
- Data Type:** Date/Time
- Description:** Delivery_Time_c
- Help Text:** (empty)
- Data Owner:** (empty)
- Field Usage:** (empty)
- Data Sensitivity Level:** (empty)
- Compliance Categorization:** (empty)
- Created By:** amisha.patel, 9/25/2025, 9:02 AM
- Modified By:** amisha.patel, 9/25/2025, 9:02 AM
- General Options:**
 - Required:
 - Default Value: (empty)
- Validation Rules:** No validation rules defined.

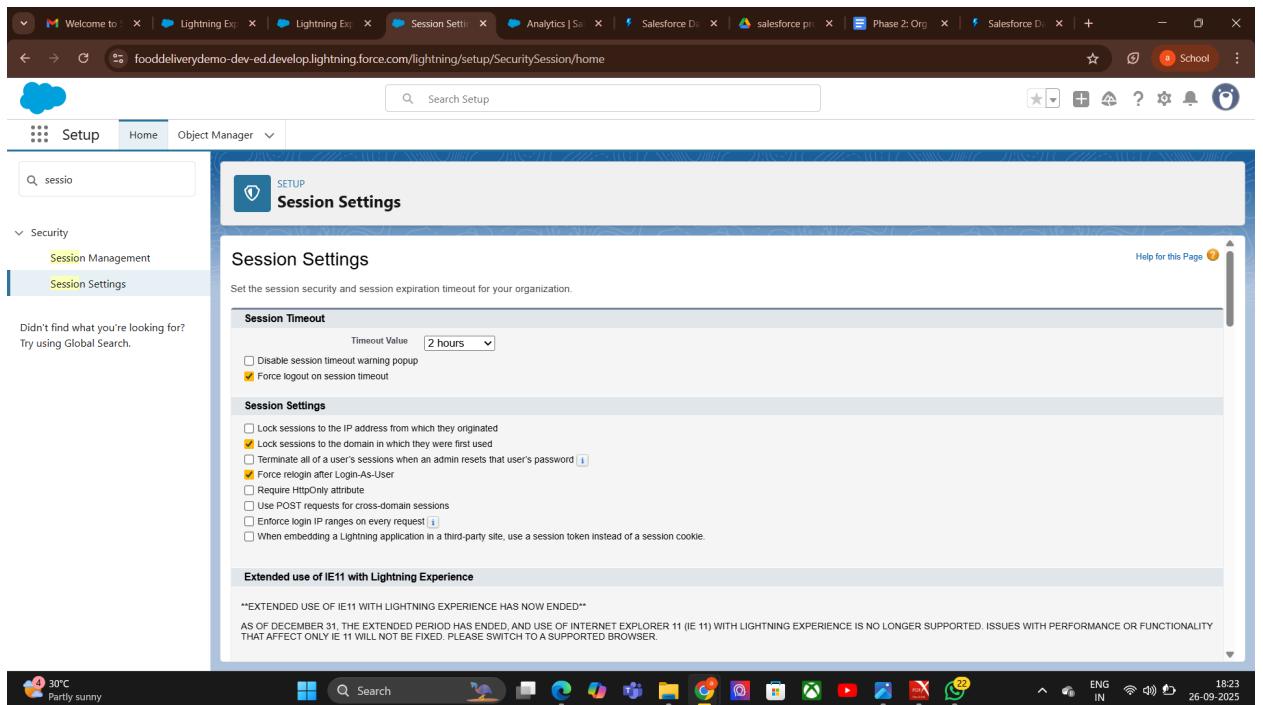
5.Session Settings:

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., IP ranges, 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.



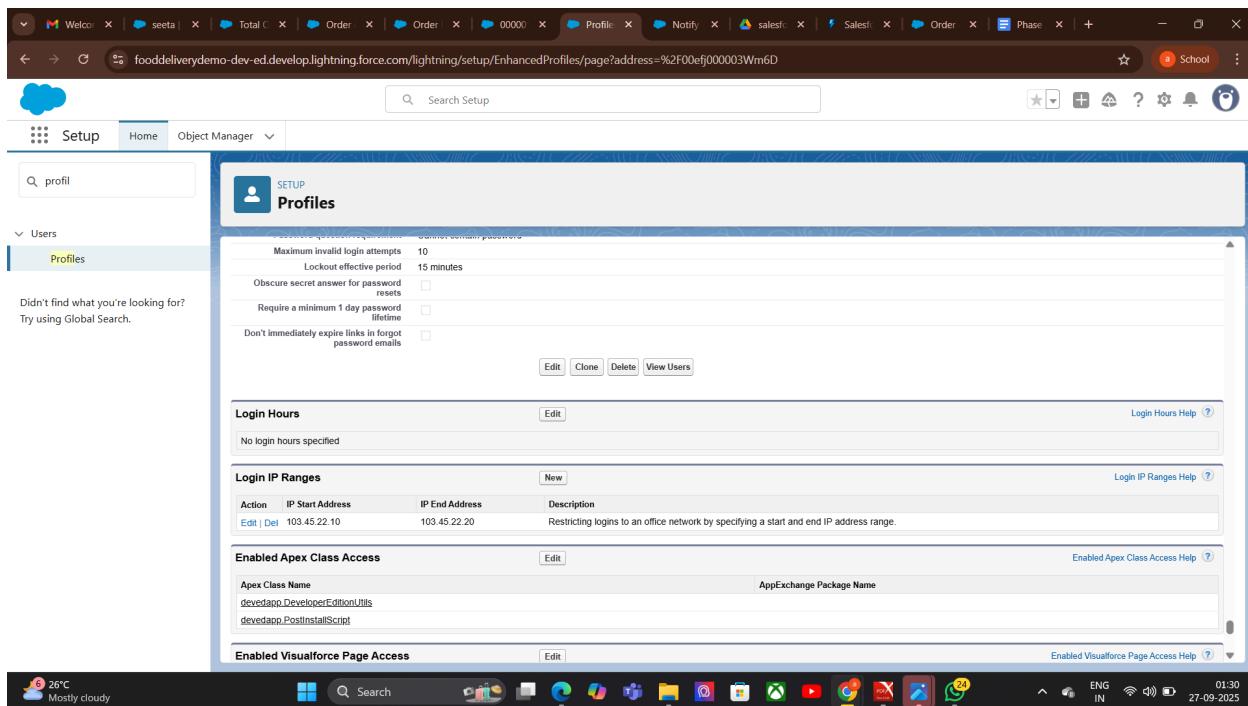
6.Login IP Ranges:

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.



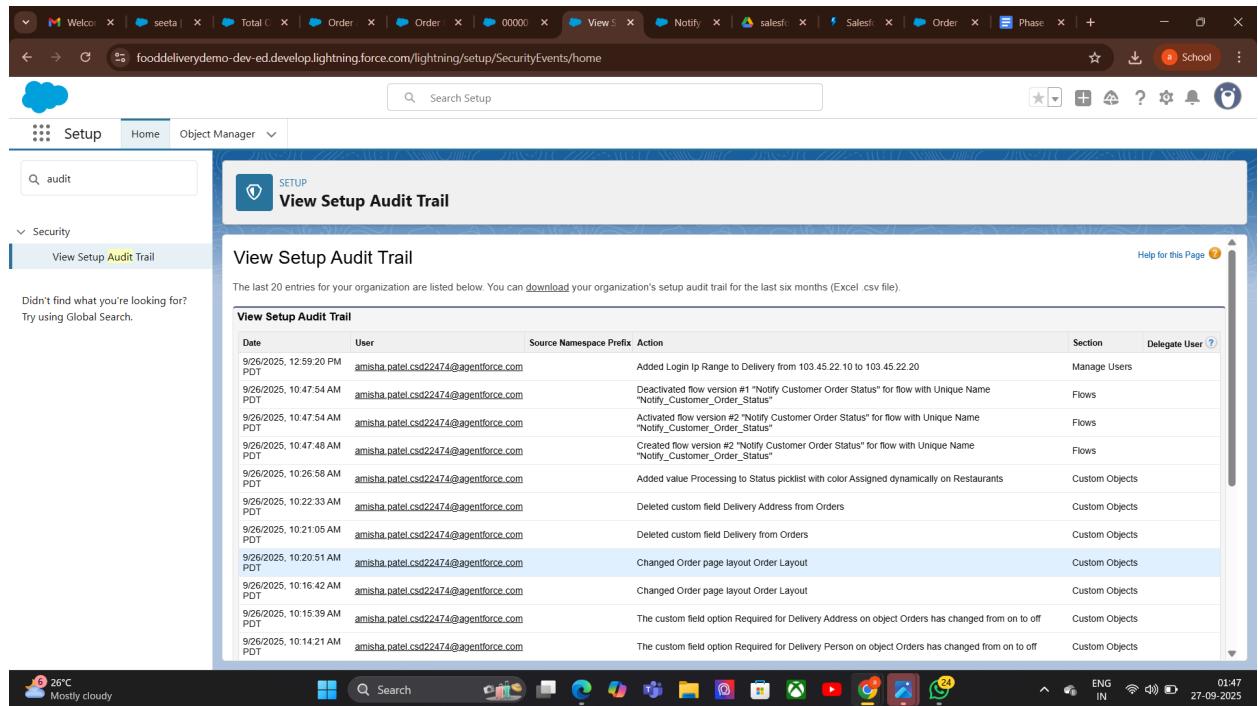
7.Audit Trail:

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.



The screenshot shows the Salesforce Setup Audit Trail page. The URL in the browser is `fooddeliverydemo-dev-ed.lightning.force.com/lightning/setup/SecurityEvents/home`. The page title is "View Setup Audit Trail". The left sidebar has a search bar with "audit" typed in and a "View Setup Audit Trail" button. Below the sidebar, it says "Didn't find what you're looking for? Try using Global Search." The main content area is titled "View Setup Audit Trail" and displays a table of audit logs. The table has columns: Date, User, Source Namespace Prefix, Action, Section, and Delegate User. The table lists 14 entries from September 26, 2025, at 12:59 PM PDT to 10:14 AM PDT. The actions include adding a login IP range, creating and activating flows, changing order page layouts, and modifying custom objects. The "Section" column indicates the changes were made to Manage Users, Flows, Custom Objects, and Order Layouts. The "Delegate User" column shows the user who made the changes, amisha.patel.cs22474@agentforce.com.

Date	User	Source Namespace Prefix	Action	Section	Delegate User
9/26/2025, 12:59:20 PM PDT	amisha.patel.cs22474@agentforce.com		Added Login Ip Range to Delivery from 103.45.22.10 to 103.45.22.20	Manage Users	
9/26/2025, 10:47:54 AM PDT	amisha.patel.cs22474@agentforce.com		Deactivated flow version #1 "Notify Customer Order Status" for flow with Unique Name "Notify_Customer_Order_Status"	Flows	
9/26/2025, 10:47:54 AM PDT	amisha.patel.cs22474@agentforce.com		Activated flow version #2 "Notify Customer Order Status" for flow with Unique Name "Notify_Customer_Order_Status"	Flows	
9/26/2025, 10:47:48 AM PDT	amisha.patel.cs22474@agentforce.com		Created flow version #2 "Notify Customer Order Status" for flow with Unique Name "Notify_Customer_Order_Status"	Flows	
9/26/2025, 10:26:58 AM PDT	amisha.patel.cs22474@agentforce.com		Added value Processing to Status picklist with color Assigned dynamically on Restaurants	Custom Objects	
9/26/2025, 10:22:33 AM PDT	amisha.patel.cs22474@agentforce.com		Deleted custom field Delivery Address from Orders	Custom Objects	
9/26/2025, 10:21:05 AM PDT	amisha.patel.cs22474@agentforce.com		Deleted custom field Delivery from Orders	Custom Objects	
9/26/2025, 10:20:51 AM PDT	amisha.patel.cs22474@agentforce.com		Changed Order page layout Order Layout	Custom Objects	
9/26/2025, 10:16:42 AM PDT	amisha.patel.cs22474@agentforce.com		Changed Order page layout Order Layout	Custom Objects	
9/26/2025, 10:15:39 AM PDT	amisha.patel.cs22474@agentforce.com		The custom field option Required for Delivery Address on object Orders has changed from on to off	Custom Objects	
9/26/2025, 10:14:21 AM PDT	amisha.patel.cs22474@agentforce.com		The custom field option Required for Delivery Person on object Orders has changed from on to off	Custom Objects	

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

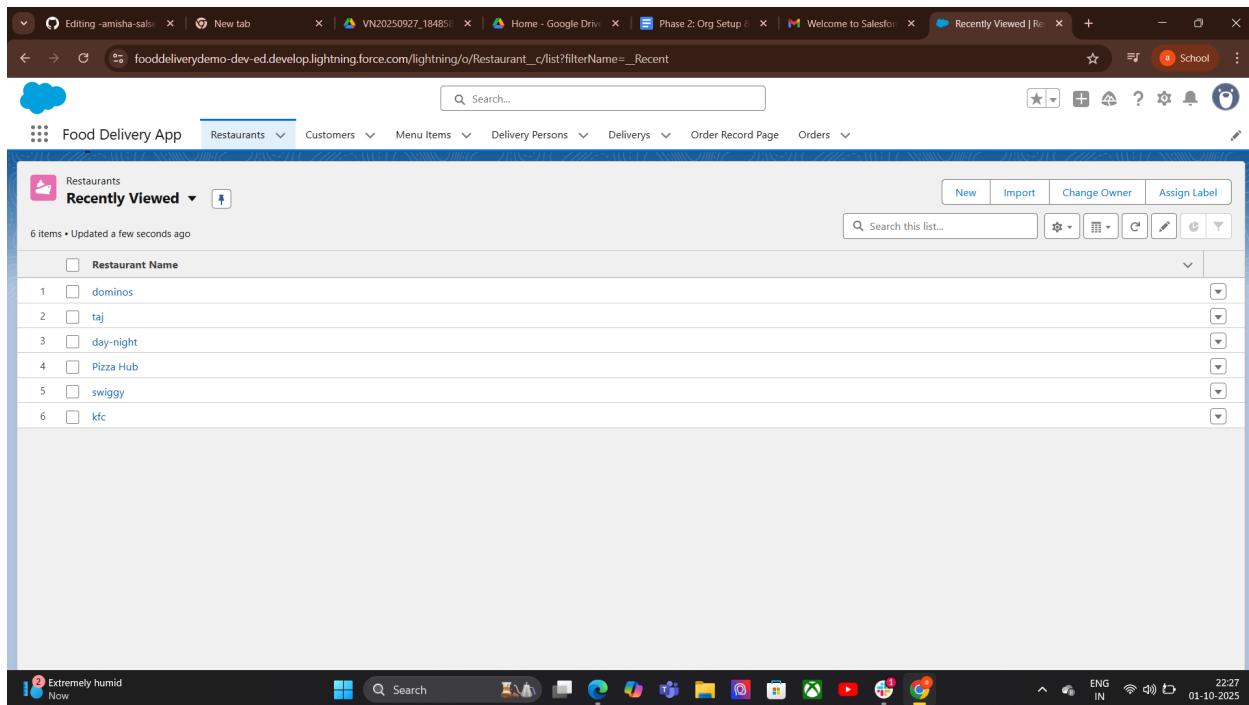
Faster deliveries, higher customer satisfaction, increased orders for restaurants, and sustainable business growth.

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.

