

## Online Food Delivery CRM Project

### Problem Statement:

The food delivery industry handles thousands of daily orders from websites, apps, and partner channels. Without a centralized CRM, restaurants struggle to manage orders, assign deliveries, track status, and handle customer complaints. Delivery agents rely on manual updates, managers lack visibility, and executives don't get real-time insights.

A Salesforce-based Online Food Delivery CRM is needed to:

- Automate order capture and assignment
- Enable real-time delivery tracking
- Centralize customer feedback and complaints
- Help managers monitor operations
- Provide executives with performance dashboards

### Phase 1: Problem Understanding & Industry Analysis

#### Requirement Gathering

- Talk to stakeholders (restaurant managers, delivery agents, executives, customers).
- **Example requirements:**
  - Track all restaurants and their menu items.
  - Allow managers to assign orders to delivery agents.
  - Prevent delivery conflicts (no double assignment).
  - Capture customer complaints/feedback.
  - Generate sales and delivery performance reports.

#### Stakeholder Analysis

- **Restaurant Managers** → Manage restaurants, menu items, orders, and customer complaints.
- **Delivery Agents** → Update delivery status, ensure timely order completion.
- **Executives / Management** → View performance dashboards, monitor KPIs.
- **System Administrator (CEO)** → Configure Salesforce org, oversee security and data integrity.

## **Business Process Mapping**

- **Order Placement** → Customer places order → System captures order → Assigned to Restaurant Manager.
- **Delivery Assignment** → Restaurant Manager assigns order to Delivery Agent.
- **Delivery Update** → Delivery Agent updates status (Assigned → Out for Delivery → Delivered).
- **Feedback Handling** → Complaints/feedback logged and monitored by Manager.
- **Reporting** → Executives analyze performance metrics via dashboards.

## **Industry-Specific Use Case Analysis**

- Real-time order tracking similar to Swiggy/Zomato.
- Menu & restaurant management within CRM.
- Delivery agent assignment and status updates.
- Customer complaints handling for service improvement.
- Performance dashboards for management review.

## **AppExchange Exploration**

- Explore AppExchange packages:
  - Food Delivery Management Apps for reference.
  - Maps & Route Optimization apps for delivery tracking.
  - SMS/Email Notification apps for customer communication.
- 

## **Phase 2: Org Setup & Configuration**

### **Salesforce Editions**

- Using Salesforce Developer Edition Org (free, permanent).
- Supports custom objects, flows, Apex, and dashboards.

### **Company Profile Setup**

- Setup → Company Information → Edit.
- **Company Name:** Online Food Delivery CRM

- **Default Locale:** English (India)
- **Time Zone:** Asia/Kolkata (IST)
- **Currency:** INR
- Saved changes for org-wide consistency.

**Company Information**  
Online Food Delivery CRM

The organization's profile is below.

User Licenses [10+] | Permission Set Licenses [10+] | Feature Licenses [11] | Usage-based Entitlements [10+]

**Organization Detail**

Organization Name	Online Food Delivery CRM	Phone	
Primary Contact	Pallepati Ritupriya	Fax	
Division	India	Default Locale	English (India)
Address	India	Default Language	English
Fiscal Year Starts In	January	Default Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Activate Multiple Currencies	<input type="checkbox"/>	Currency Locale	Hindi (India) - INR
Enable Data Translation	<input type="checkbox"/>	Used Data Space	342 KB (7%) <a href="#">[View]</a>
Newsletter	<input checked="" type="checkbox"/>	Used File Space	17 KB (0%) <a href="#">[View]</a>
Admin Newsletter	<input checked="" type="checkbox"/>	API Requests, Last 24 Hours	0 (15,000 max)
Hide Notices About System Maintenance	<input type="checkbox"/>	Streaming API Events, Last 24 Hours	0 (10,000 max)
Hide Notices About System Downtime	<input type="checkbox"/>	Restricted Logins, Current Month	0 (0 max)
Locale Formats	ICU	Salesforce.com Organization ID	00Dg1.00000BQVBV
		Organization Edition	Developer Edition
		Instance	CAN98

Created By: OrgFarm\_EPIC, 9/11/2025, 2:11 PM      Modified By: Pallepati Ritupriya, 9/18/2025, 12:03 AM

## Business Hours & Holidays

- Setup → Business Hours → New Business Hours.
- **Working Hours:** 9:00 AM – 6:00 PM (Mon–Sat).
- Setup → Holidays → Add national holidays (e.g., 15th August – Independence Day).
- Ensures workflows/approvals don't trigger during holidays.

**Business Hours**

**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 \[1\]](#)

**Business Hours Detail**

Business Hours Name	Default Business Hours	Time Zone																								
Business Hours	<table border="1"> <thead> <tr> <th>Day</th> <th>Start Time</th> <th>End Time</th> </tr> </thead> <tbody> <tr> <td>Sunday</td> <td>No Hours</td> <td></td> </tr> <tr> <td>Monday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> <tr> <td>Tuesday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> <tr> <td>Wednesday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> <tr> <td>Thursday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> <tr> <td>Friday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> <tr> <td>Saturday</td> <td>9:00 AM</td> <td>6:00 PM</td> </tr> </tbody> </table>	Day	Start Time	End Time	Sunday	No Hours		Monday	9:00 AM	6:00 PM	Tuesday	9:00 AM	6:00 PM	Wednesday	9:00 AM	6:00 PM	Thursday	9:00 AM	6:00 PM	Friday	9:00 AM	6:00 PM	Saturday	9:00 AM	6:00 PM	(GMT+05:30) India Standard Time (Asia/Kolkata)
Day	Start Time	End Time																								
Sunday	No Hours																									
Monday	9:00 AM	6:00 PM																								
Tuesday	9:00 AM	6:00 PM																								
Wednesday	9:00 AM	6:00 PM																								
Thursday	9:00 AM	6:00 PM																								
Friday	9:00 AM	6:00 PM																								
Saturday	9:00 AM	6:00 PM																								
Active	<input checked="" type="checkbox"/>	Default Business Hours	<input checked="" type="checkbox"/>																							

Created By: Pallepati Ritupriya 9/17/2025, 3:44 AM      Last Modified By: Pallepati Ritupriya 9/18/2025, 12:08 AM

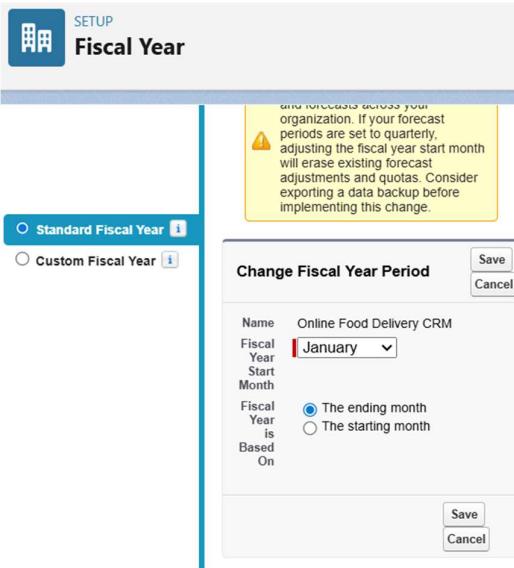
[Edit](#)

**Holidays**

Holiday Name	Description	Date and Time
Independence Day		8/15/2026 All Day <a href="#">[Edit]</a>

## Fiscal Year Settings

- Setup → Fiscal Year → Standard (Jan–Dec).
- Useful for revenue and sales reporting.



## User Setup & Licenses

Created project-specific users:

- Me (Pallepati Ritupriya) = System Administrator (CEO) → Full access
- Anita Sharma (Restaurant Manager) → Salesforce License
- Ravi Kumar (Delivery Agent) → Salesforce License
- Priya Verma (Executive/Management) → Salesforce Platform License

The screenshot shows the 'Users' list page under 'Setup'. At the top, it says 'Created By Me'. Below that is a note: 'On this page you can create, view, and manage users.' and 'To get more licenses, use the Your Account app [Let's Go](#)'. There are buttons for 'View' (set to 'Created By Me'), 'Edit', and 'Create New View'. A navigation bar at the bottom includes links for A through Z and 'Other'. The main table has columns: Action, Full Name, Alias, Username, Role, Active, and Profile. The data is as follows:

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Login</a>	Anitha Sharma	sani	anitha.sharma@foodcrm.com	Restaurant Manager	✓	Restaurant_Manager_Profile
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Login</a>	Priya Verma	vpriy	priya.verma.foodcrm@test.com	Executive	✓	Standard Platform User
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Login</a>	Ravi Kumar	kravi	ravi.kumar.foodcrm@test.com	Delivery Agent	✓	Delivery_Agent_Profile
<input type="checkbox"/> <a href="#">Edit</a>	Ritupriya Pallepati	rit	ritupriya.p176698@salesforce.com	CEO	✓	System Administrator

## Profiles

- Cloned **Standard User** into custom profiles:
- Restaurant\_Manager\_Profile
- Delivery\_Agent\_Profile
- Executive\_Profile



## Profiles

Created by me ▾			Edit   Delete   Create New View
New Profile		A   B   C   D   E   F   G   H   I   J   K	
<input type="checkbox"/>	Action	Profile Name	↑
<input type="checkbox"/>	<a href="#">Edit   Del   ...</a>	<a href="#">Delivery_Agent_Profile</a>	Salesforce
<input type="checkbox"/>	<a href="#">Edit   Del   ...</a>	<a href="#">Executive_Profile</a>	Salesforce
<input type="checkbox"/>	<a href="#">Edit   Del   ...</a>	<a href="#">Restaurant_Manager_Profile</a>	Salesforce

## Roles

- **CEO (System Admin)** → Top-level, full access to all records.
- **Executive / Management** → Reports to CEO, sees all managers & agents records.
- **Restaurant Manager** → Reports to Executive, manages restaurants, menus, orders, and complaints.
- **Delivery Agent** → Reports to Manager, sees and updates only their assigned orders/deliveries.



## 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](#)



## Permission Sets

Permission sets provide extra access without changing user profiles.

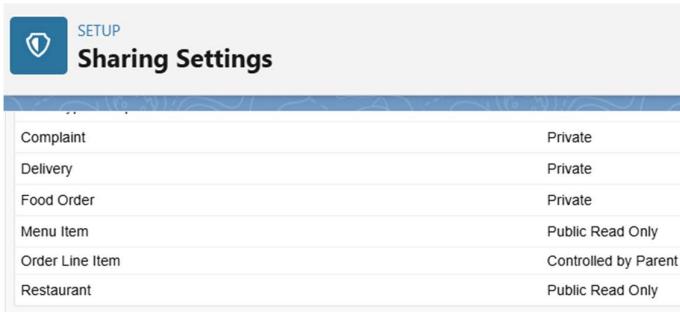
- **Reports & Dashboards Access** → Grants executives (Priya Verma) ability to run reports and view dashboards.
- **Complaint Manager Access** → Allows delivery agents (Ravi Kumar) to read/edit customer complaints when needed.
- **Delivery Assignment Access** → Gives restaurant managers (Anita Sharma) permission to reassign or update deliveries.

The screenshot shows the Salesforce 'Permission Sets' page under the 'SETUP' tab. The page title is 'Permission Sets'. A sub-header says 'Permission Sets'. Below it, a note states 'On this page you can create, view, and manage permission sets.' A dropdown menu shows 'Created by me' with options to 'Edit' or 'Delete' or 'Create New View'. There is a 'New' button and a refresh icon. A navigation bar at the top has links from A to P. The main table lists three permission sets:

Action	Permission Set Name	Description
<input type="checkbox"/> Del   Clone	Complaint Manager Access	Allows agents to read/edit customer complaints when needed.
<input type="checkbox"/> Del   Clone	Delivery Assignment Access	Gives permission to reassign or update deliveries.
<input type="checkbox"/> Del   Clone	Reports & Dashboards Access	Ability to run reports and view dashboards.

## Organization-Wide Defaults (OWD)

- Setup → Sharing Settings → Organization-Wide Defaults
- Configure default access for each object:
  - **Restaurant\_c:** Public Read Only
    - All users can view restaurants, but only managers/admins can edit.
  - **Menu\_Item\_c:** Public Read Only
    - All users can see menu items.
  - **Food\_Orders\_c:** Private
    - Only the owner (customer) and assigned staff can see the order.
  - **Order\_Line\_Item\_c:** Controlled by Parent
    - Inherits access from Food\_Orders\_c.
  - **Delivery\_c:** Private
    - Only assigned delivery agent and manager/admin can see.
  - **Complaint\_c:** Private
    - Only owner (customer) and support/admin can see.
- This ensures baseline record visibility and data security for all users.



Sharing Settings	
Complaint	Private
Delivery	Private
Food Order	Private
Menu Item	Public Read Only
Order Line Item	Controlled by Parent
Restaurant	Public Read Only

## Sharing Rules

- Setup → Sharing Settings → Sharing Rules
- Sharing Rules grant additional access beyond OWD for specific business scenarios, ensuring that users in certain roles can view or edit records as needed.

### Rules:

#### 1. Complaint Sharing Rule

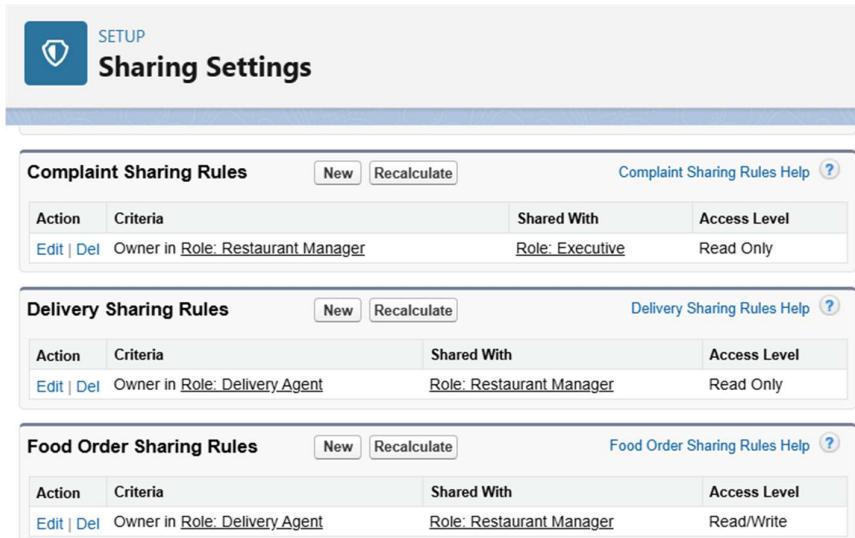
- Access Level:** Read Only
- Allows executives to view complaints raised by restaurant managers.

#### 2. Delivery Sharing Rule

- Access Level:** Read Only
- Enables restaurant managers to view deliveries assigned to their agents.

#### 3. Food Order Sharing Rule

- Access Level:** Read/Write
- Lets restaurant managers view and update orders handled by delivery agents.



Complaint Sharing Rules		New	Recalculate	Complaint Sharing Rules Help
Action	Criteria	Shared With	Access Level	
Edit   Del	Owner in Role: Restaurant Manager	Role: Executive	Read Only	
<b>Delivery Sharing Rules</b>				
Action	Criteria	Shared With	Access Level	Delivery Sharing Rules Help
Edit   Del	Owner in Role: Delivery Agent	Role: Restaurant Manager	Read Only	
<b>Food Order Sharing Rules</b>				
Action	Criteria	Shared With	Access Level	Food Order Sharing Rules Help
Edit   Del	Owner in Role: Delivery Agent	Role: Restaurant Manager	Read/Write	

## Login Access Policies

- Setup → Login Access Policies.
- Enabled Administrators Can Log in as Any User.
- Allows Admin to test features as Manager, Agent, or Executive.



### Login Access Policies

Control which support organizations your users can grant login access to.

Manage Support Options		Save	Cancel
Setting		Enabled	
Administrators Can Log in as Any User		<input checked="" type="checkbox"/>	

## Deployment Basics

- Development in VS Code + Salesforce CLI.
- Metadata synced via package.xml.
- Source code stored in GitHub repo.
- **Deployment flow:**  
VS Code (local changes) → Push to Salesforce Org → Commit to GitHub.

## Phase 3: Data Modeling & Relationships

### Standard & Custom Objects

#### Standard Objects:

- **Contact:** Represents customers placing orders.
- **User:** Represents internal users like Restaurant Managers, Delivery Agents, and Executives.

## Custom Objects:

- Setup → Object Manager → Create → Custom Object
- Used to manage business-specific data that is not captured by standard objects like Account, Contact, or User.

### 1. Restaurant\_\_c

#### Purpose:

Stores information about restaurants in the Food Delivery CRM, including contact details, location, and cuisine type.

The screenshot shows the 'Object Manager' interface for creating a new custom object named 'Restaurant'. The left sidebar lists various configuration tabs: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, and Related Lookup Filters. The 'Details' tab is selected. On the right, the 'Details' section displays the following configuration:

Setting	Value
Description	
API Name	Restaurant__c
Custom	✓
Singular Label	Restaurant
Plural Label	Restaurants
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

At the bottom right of the 'Details' section are 'Edit' and 'Delete' buttons.

#### Fields:

- Restaurant Name (Name, Text 80): The name of the restaurant.
- Address\_\_c (Long Text Area 32768): Full address of the restaurant.
- Location\_\_c (Text 255): Short location or area name.
- Contact\_Phone\_\_c (Phone): Restaurant contact number.
- Cuisine\_\_c (Picklist): Type of cuisine offered (e.g., Indian, Italian).
- OwnerId (Lookup to User/Group): The user or group who owns the record.
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

## Relationships:

- Menu\_Item\_\_c → Restaurant\_\_c (Lookup): Links menu items to the restaurant.
- Food\_Orders\_\_c → Restaurant\_\_c (Lookup): Links orders to the restaurant.

The screenshot shows the 'Object Manager' interface for the 'Restaurant' object. The left sidebar lists various configuration options: Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, and Scoping Rules. The main content area is titled 'Fields & Relationships' and displays a table of fields:

FIELD LABEL	FIELD NAME	DATA TYPE
Address	Address__c	Long Text Area(32768)
Contact Phone	Contact_Phone__c	Phone
Created By	CreatedById	Lookup(User)
Cuisine	Cuisine__c	Picklist
Last Modified By	LastModifiedById	Lookup(User)
Location	Location__c	Text(255)
Owner	OwnerId	Lookup(User,Group)
Restaurant Name	Name	Text(80)

## Page Layouts:

- Contact Info Section: Address, Contact Phone
- Restaurant Info Section: Restaurant Name, Location, Cuisine

The screenshot shows the 'Object Manager' interface for the 'Restaurant' object. The left sidebar lists various configuration options: Details, Fields & Relationships, Page Layouts (selected), Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, and Scoping Rules. The main content area shows the 'Page Layouts' configuration for the 'Restaurant' object. It includes sections for 'Fields' (Buttons, Quick Actions, Mobile & Lightning Actions, Expanded Lookups, Related Lists, Report Charts) and 'Standard Buttons' (Edit, Delete, Clone, Change). Below these are two sections: 'Contact Info' (Address: Sample Text, Contact Phone: 1-415-555-1212) and 'Restaurant Info' (Restaurant Name: Sample Text, Location: Sample Text, Cuisine: Sample Text).

## 2. Menu\_Item\_\_c

### Purpose:

Stores details of food items offered by restaurants, including price, category, availability, and description.

The screenshot shows the Salesforce Object Manager interface for the 'Menu Item' object. The top navigation bar says 'SETUP > OBJECT MANAGER'. The main title is 'Menu Item'. On the left, there's a sidebar with a 'Details' tab selected, followed by a list of options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, and Related Lookup Filters. The main content area has a 'Details' tab selected. It shows the following fields and their values:

Field	Value
Description	
API Name	Menu_Item__c
Custom	✓
Singular Label	Menu Item
Plural Label	Menu Item
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

At the bottom right of the main content area are 'Edit' and 'Delete' buttons.

### Fields:

- Menu Item Name (Name, Text 80): Name of the menu item.
- Price\_\_c (Currency 16,2): Price of the menu item.
- Category\_\_c (Picklist): Type/category of the food item (e.g., Appetizer, Main Course).
- Description\_\_c (Long Text Area 32768): Detailed description of the item.
- Is\_Available\_\_c (Checkbox): Indicates if the item is currently available.
- Restaurant\_\_c (Lookup to Restaurant\_\_c): Associated restaurant for the menu item.
- OwnerId (Lookup to User/Group): Owner of the record.
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

### Relationships:

- Menu\_Item\_\_c → Restaurant\_\_c (Lookup): Links menu items to the restaurant.
- Order\_Line\_Item\_\_c → Menu\_Item\_\_c (Lookup): Links line items to menu items in orders.

SETUP > OBJECT MANAGER

### Menu Item

Fields & Relationships			
9 Items, Sorted by Field Label			
	FIELD LABEL	FIELD NAME	DATA TYPE
	Category	Category__c	Picklist
	Created By	CreatedBy	Lookup(User)
	Description	Description__c	Long Text Area(32768)
	Is Available	Is_Available__c	Checkbox
	Last Modified By	LastModifiedBy	Lookup(User)
	Menu Item Name	Name	Text(80)
	Owner	OwnerId	Lookup(User,Group)
	Price	Price__c	Currency(16, 2)
	Restaurant	Restaurant__c	Lookup(Restaurant)

### Page Layouts:

- Item Info Section: Menu Item Name, Price, Category, Description, Is Available
- Restaurant Info Section: Restaurant

SETUP > OBJECT MANAGER

### Menu Item

Page Layouts	
	Fields
	Buttons
	Quick Actions
	Mobile & Lightning Actions
	Expanded Lookups
	Related Lists
	Report Charts

**Menu Item Detail**

Item Info		
★ ●	Menu Item Name	Sample Text
	Price	₹123.45
	Category	Sample Text
	Description	Sample Text
	Is Available	<input checked="" type="checkbox"/>

Restaurant Info	
Restaurant	Sample Text

### 3. Food\_Orders\_\_c

#### Purpose:

Represents customer orders in the Food Delivery CRM, including order details, customer information, restaurant, and payment status.

The screenshot shows the Salesforce Object Manager interface. At the top, it says "SETUP > OBJECT MANAGER" and the object name is "Food Order". On the left, there's a sidebar with a "Details" tab selected, followed by a list of options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, and Related Lookup Filters. The main area is titled "Details" and contains the following configuration fields:

API Name	Food_Orders__c	Enable Reports
Custom	✓	Track Activities
Singular Label	Food Order	Track Field History
Plural Label	Food Orders	Deployment Status
		Deployed
		Help Settings
		Standard salesforce.com Help Window

At the bottom right of the main area, there are "Edit" and "Delete" buttons.

#### Fields:

- Order Number (Name, Auto Number): Unique identifier for the order.
- Customer\_\_c (Lookup to Contact): Customer who placed the order.
- Delivery\_Address\_\_c (Text Area 255): Address where the order should be delivered.
- Restaurant\_\_c (Lookup to Restaurant\_\_c): Restaurant fulfilling the order.
- Order\_Date\_\_c (Date/Time): Date and time when the order was placed.
- Status\_\_c (Picklist): Current status of the order (e.g., Pending, Completed).
- Total\_Amount\_\_c (Currency 16,2): Total amount for the order.
- Grand\_Total\_\_c (Roll-Up Summary, SUM of Order Line Items): Total of all line items.
- Payment\_Mode\_\_c (Picklist): Mode of payment (e.g., Cash, Card, Online).
- Is\_Paid\_\_c (Checkbox): Indicates whether the order has been paid.
- OwnerId (Lookup to User/Group): Owner of the record.
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

## **Relationships:**

- Food\_Orders\_\_c → Customer\_\_c (Lookup to Contact): Links order to customer.
- Food\_Orders\_\_c → Restaurant\_\_c (Lookup): Links order to restaurant.
- Order\_Line\_Item\_\_c → Food\_Orders\_\_c (Master-Detail): Links line items to the parent order.
- Delivery\_\_c → Food\_Orders\_\_c (Lookup): Links delivery record to the order.
- Complaint\_\_c → Food\_Orders\_\_c (Lookup): Links complaints to the order.

SETUP > OBJECT MANAGER

### Food Order

Details

**Fields & Relationships**

13 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Customer	Customer__c	Lookup(Contact)
Delivery Address	Delivery_Address__c	Text Area(255)
Grand Total	Grand_Total__c	Roll-Up Summary (SUM Order Line Item)
Is Paid	Is_Paid__c	Checkbox
Last Modified By	LastModifiedById	Lookup(User)
Order Date	Order_Date__c	Date/Time
Order Number	Name	Auto Number
Owner	OwnerId	Lookup(User,Group)
Payment Mode	Payment_Mode__c	Picklist
Restaurant	Restaurant__c	Lookup(Restaurant)
Status	Status__c	Picklist
Total Amount	Total_Amount__c	Currency(16, 2)

## **Page Layouts:**

- Order Info Section: Order Number, Order Date, Status, Total Amount, Payment Mode, Is Paid, Grand Total
- Customer Info Section: Customer, Delivery Address
- Restaurant Info Section: Restaurant

SETUP > OBJECT MANAGER

### Food Order

**Page Layouts**

Food Order Detail

Order Info		Customer Info		Restaurant Info	
Order Number	GEN-2004-001234	Customer	Sample Text	Delivery Address	Sample Text
Status	Sample Text	Payment Mode	Sample Text	Order Date	9/20/2025, 3:43 AM
Payment Mode	Sample Text	Grand Total	₹123.45	Total Amount	₹123.45
				Is Paid	<input checked="" type="checkbox"/>

## 4. Order\_Line\_Item\_\_c

### Purpose:

Represents individual items within a food order, including the menu item, quantity, pricing, and calculated total.

SETUP > OBJECT MANAGER

### Order Line Item

**Details**

Description	Enable Reports
API Name Order_Line_Item__c	<input checked="" type="checkbox"/>
Custom	<input checked="" type="checkbox"/>
Singular Label Order Line Item	Track Activities
Plural Label Order Line Items	Track Field History
	Deployment Status
	Deployed
	Help Settings
	Standard salesforce.com Help Window

### Fields:

- Order Line Item Name (Name, Auto Number): Unique identifier for the line item.
- Food\_Order\_\_c (Master-Detail to Food\_Orders\_\_c): Parent order to which this line item belongs.

- Menu\_Item\_\_c (Lookup to Menu\_Item\_\_c): Menu item included in this line item.
- Quantity\_\_c (Number 3,0): Number of units ordered.
- Unit\_Price\_\_c (Currency 16,2): Price per unit of the menu item.
- Total\_\_c (Formula, Currency): Calculated total for this line item (Quantity × Unit Price).
- OwnerId (Lookup to User/Group): Owner of the record.
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

### **Relationships:**

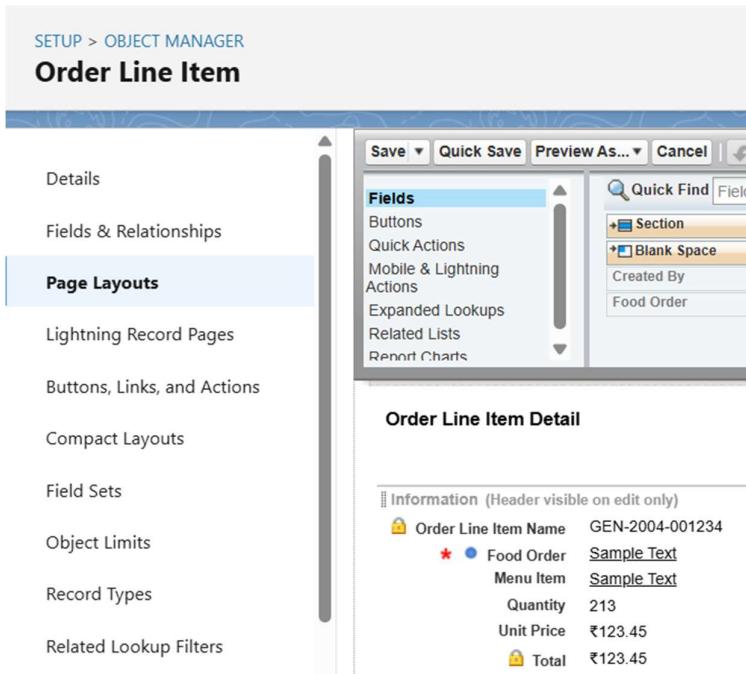
- Order\_Line\_Item\_\_c → Food\_Orders\_\_c (Master-Detail): Links line item to parent food order.
- Order\_Line\_Item\_\_c → Menu\_Item\_\_c (Lookup): Links line item to the menu item being ordered.

SETUP > OBJECT MANAGER  
**Order Line Item**

Fields & Relationships		
8 Items, Sorted by Field Label		
FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Food Order	Food_Order__c	Master-Detail(Food Order)
Last Modified By	LastModifiedById	Lookup(User)
Menu Item	Menu_Item__c	Lookup(Menu Item)
Order Line Item Name	Name	Auto Number
Quantity	Quantity__c	Number(3, 0)
Total	Total__c	Formula (Currency)
Unit Price	Unit_Price__c	Currency(16, 2)

### **Page Layouts:**

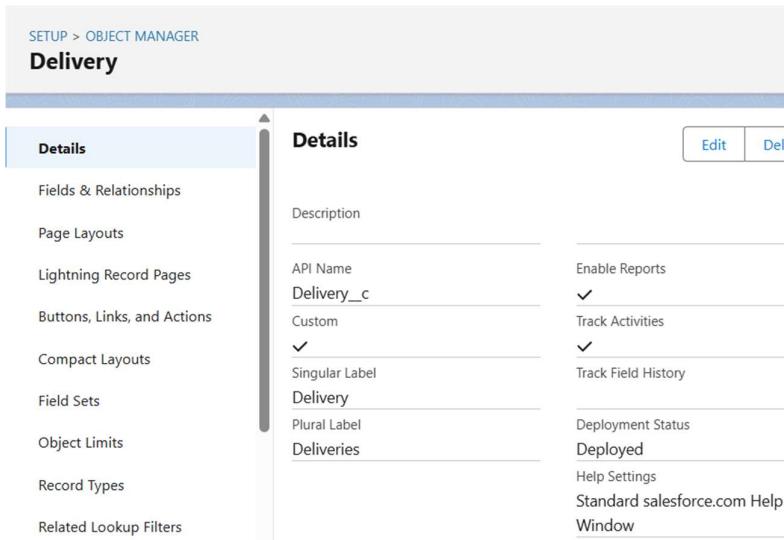
- Information Section: Order Line Item Name, Food Order, Menu Item, Quantity, Unit Price, Total



## 5. Delivery\_c

### Purpose:

Tracks the delivery details of food orders, including assigned delivery agent, estimated time of arrival (ETA), and delivery status.



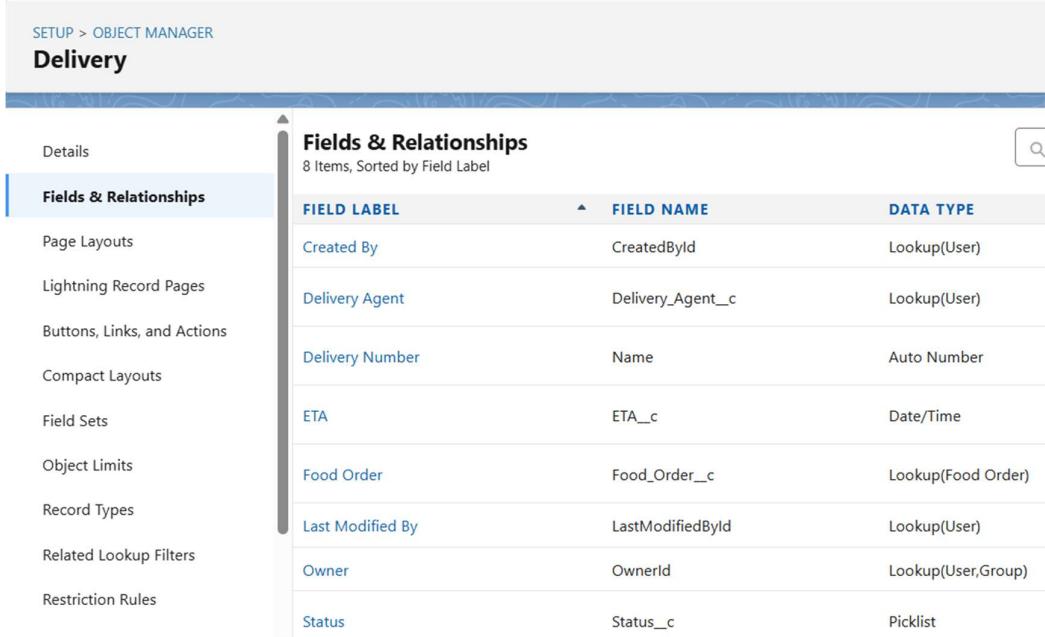
### Fields:

- Delivery Number (Name, Auto Number): Unique identifier for each delivery.
- Delivery\_Agent\_\_c (Lookup to User): Delivery agent assigned to this delivery.

- Food\_Order\_\_c (Lookup to Food\_Orders\_\_c): Links the delivery to its associated food order.
- ETA\_\_c (Date/Time): Estimated time of arrival for the delivery.
- Status\_\_c (Picklist): Current status of the delivery (e.g., Assigned, Out for Delivery, Delivered).
- OwnerId (Lookup to User/Group): The owner of the record.
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

### Relationships:

- Delivery\_\_c → Food\_Orders\_\_c (Lookup): Associates each delivery with a specific food order.
- Delivery\_\_c → User (Lookup): Links the delivery record to the assigned delivery agent.



The screenshot shows the Salesforce Object Manager interface for the 'Delivery' object. The left sidebar has a 'Fields & Relationships' section selected. The main area displays a table titled 'Fields & Relationships' with 8 items, sorted by Field Label. The table columns are FIELD LABEL, FIELD NAME, and DATA TYPE.

FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Delivery Agent	Delivery_Agent__c	Lookup(User)
Delivery Number	Name	Auto Number
ETA	ETA__c	Date/Time
Food Order	Food_Order__c	Lookup(Food Order)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Status	Status__c	Picklist

### Page Layouts:

- Delivery Info Section: Delivery Number, Status, ETA
- Assignment Info Section: Food Order, Delivery Agent

SETUP > OBJECT MANAGER

## Delivery

The screenshot shows the 'Object Manager' interface for the 'Delivery' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, etc. The 'Page Layouts' option is selected. The main area displays the 'Delivery Detail' page layout. It includes a 'Fields' section with a 'Quick Find' search bar, a 'Delivery Info' section with fields for Delivery Number (GEN-2004-001234), ETA (9/20/2025, 3:59 AM), Status (Status), and Sample Text, and an 'Assignment Info' section with fields for Food Order (Food Order), Delivery Agent (Delivery Agent), and Sample Text.

## 6. Complaint\_c

### Purpose:

Captures customer complaints related to food orders, including complaint type, status, and customer feedback.

SETUP > OBJECT MANAGER

## Complaint

The screenshot shows the 'Object Manager' interface for the 'Complaint' object. The left sidebar lists various setup options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, etc. The 'Details' option is selected. The main area displays the 'Details' page for the Complaint object. It includes fields for Description, API Name (Complaint\_c), Custom (✓), Singular Label (Complaint), Plural Label (Complaints), Enable Reports (✓), Track Activities (✓), Track Field History, Deployment Status (Deployed), Help Settings, and Standard salesforce.com Help Window.

### **Fields:**

- Complaint Number (Name, Auto Number): Unique identifier for each complaint.
- Complaint\_Type\_\_c (Picklist): Type of complaint (e.g., Late Delivery, Wrong Item, Quality Issue).
- Status\_\_c (Picklist): Current status of the complaint (e.g., Open, In Progress, Resolved, Closed).
- Comments\_\_c (Long Text Area 32768): Detailed description or comments provided by the customer.
- Food\_Order\_\_c (Lookup to Food\_Orders\_\_c): Links the complaint to the related food order.
- Customer\_\_c (Lookup to Contact): Customer who raised the complaint.
- OwnerId (Lookup to User/Group): Owner of the record (e.g., Support Agent).
- CreatedById (Lookup to User): System field indicating who created the record.
- LastModifiedById (Lookup to User): System field indicating who last modified the record.

### **Relationships:**

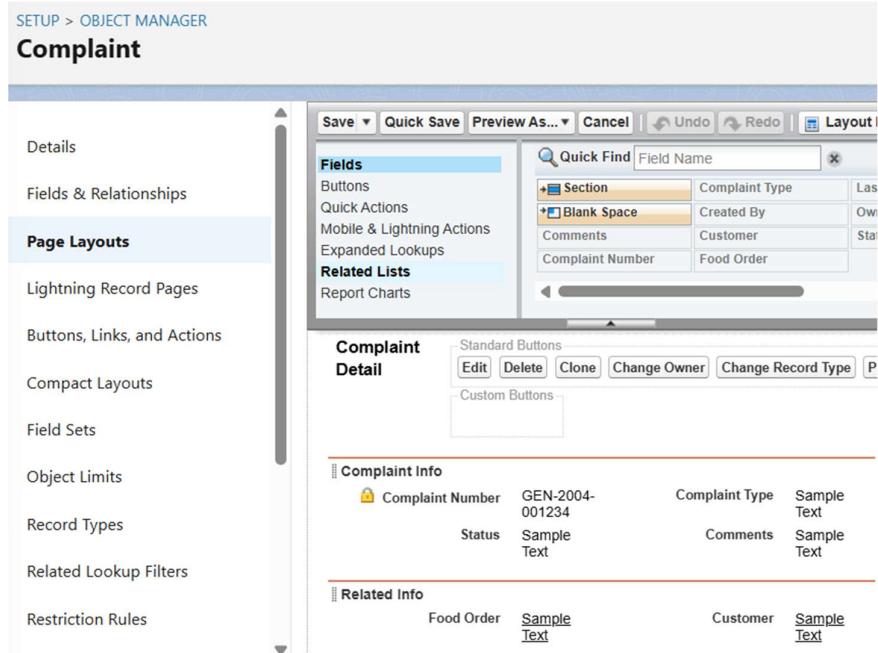
- Complaint\_\_c → Food\_Orders\_\_c (Lookup): Associates the complaint with a specific order.
- Complaint\_\_c → Contact (Lookup): Links the complaint to the customer who submitted it.

The screenshot shows the Salesforce Object Manager interface for the 'Complaint' object. The left sidebar lists various setup options like Details, Page Layouts, Lightning Record Pages, etc. The main area is titled 'Fields & Relationships' with a sub-section header 'Fields & Relationships' and a note '9 Items, Sorted by Field Label'. A search bar is at the top right. The table lists nine fields with their corresponding field labels, names, and data types:

FIELD LABEL	FIELD NAME	DATA TYPE
Comments	Comments__c	Long Text Area(32768)
Complaint Number	Name	Auto Number
Complaint Type	Complaint_Type__c	Picklist
Created By	CreatedById	Lookup(User)
Customer	Customer__c	Lookup(Contact)
Food Order	Food_Order__c	Lookup(Food Order)
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Status	Status__c	Picklist

## Page Layouts:

- Complaint Info Section: Complaint Number, Complaint Type, Status, Comments
- Related Info Section: Food Order, Customer



## Lookup vs Master-Detail vs Hierarchical Relationships

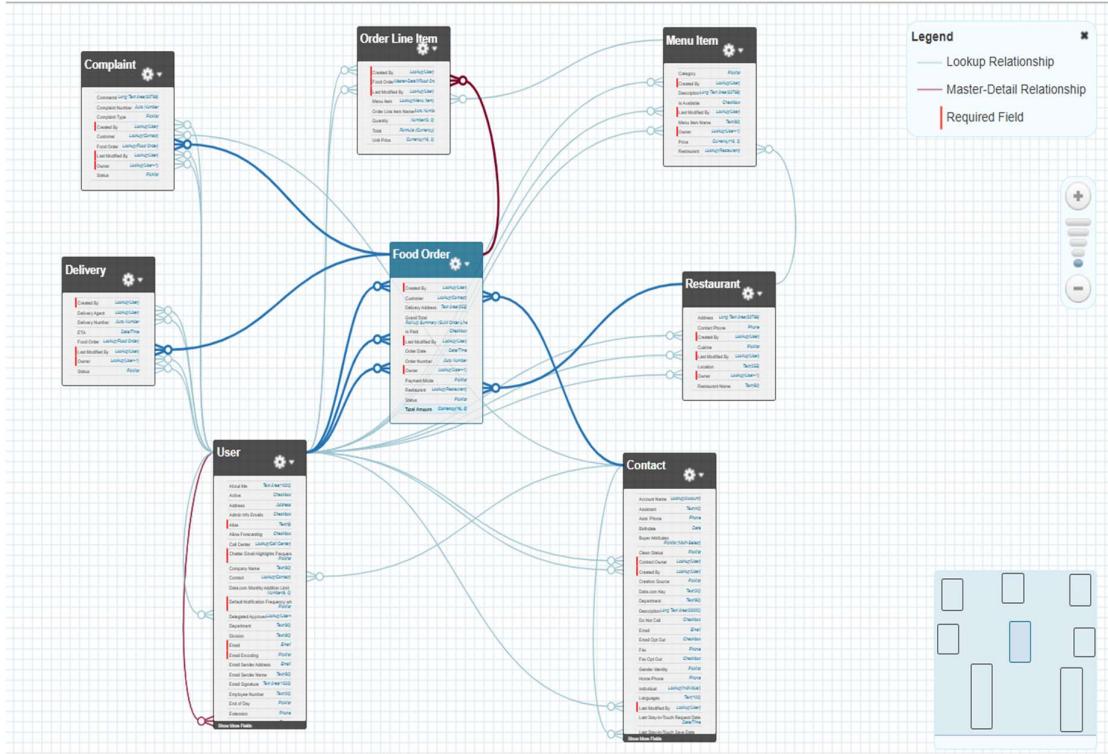
- **Restaurant\_\_c ↔ Menu\_Item\_\_c → Lookup**  
A menu item is linked to a restaurant, but deleting a restaurant doesn't automatically delete its menu items.
- **Food\_Orders\_\_c ↔ Restaurant\_\_c → Lookup**  
An order is associated with a restaurant, but restaurants don't own the orders.
- **Food\_Orders\_\_c ↔ Contact (Customer) → Lookup**  
Orders are linked to customers (Contacts), but customers don't "own" the order records.
- **Food\_Orders\_\_c ↔ Order\_Line\_Item\_\_c → Master-Detail**  
Line items cannot exist without a parent order. Deleting a Food Order deletes all related Order Line Items.
- **Delivery\_\_c ↔ Food\_Orders\_\_c → Lookup**  
Each delivery record is linked to an order, but the delivery doesn't own the order.
- **Complaint\_\_c ↔ Food\_Orders\_\_c → Lookup**  
Complaints are tied to specific orders, but they can exist independently.

- **Complaint\_c ↔ Contact (Customer) → Lookup**

Complaints are linked to the customer who raised them, but customers don't own the complaints.

## Schema Builder

- Setup → Object Manager → Schema Builder
- A visual tool in Salesforce that lets you view and manage objects, fields, and relationships in a single interface.



## Phase 4: Process Automation (Admin)

### Validation Rules

- Setup → Object Manager → Select Object → Validation Rules
- Validation Rules enforce data integrity and business logic, preventing incomplete or invalid data entry.

## Rules:

### 1. Delivery – ETA\_Required

- Error Condition Formula: AND(ISPICKVAL(Status\_\_c, "Assigned"), ISBLANK(ETA\_\_c))
- Error Message: ETA must be set when a delivery is assigned.

The screenshot shows the 'Delivery Validation Rule' setup page in the Object Manager. The left sidebar lists options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, and Object Limits. The main area displays the 'Validation Rule Detail' for the rule named 'ETA\_Required'. The rule is active and has the formula: AND(ISPICKVAL(Status\_\_c, "Assigned"), ISBLANK(ETA\_\_c)). The error message is 'ETA must be set when a delivery is assigned.' The error location is set to 'Top of Page'. The rule was created by Pallepati Ritupriya on 9/19/2025, 4:07 AM, and modified by the same user on the same date and time. There are 'Edit' and 'Clone' buttons at the bottom.

### 2. Food Order – Positive\_Amount

- Error Condition Formula: Total\_Amount\_\_c <= 0
- Error Message: Order amount must be greater than zero.

The screenshot shows the 'Food Order Validation Rule' setup page in the Object Manager. The left sidebar lists options like Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, and Object Limits. The main area displays the 'Validation Rule Detail' for the rule named 'Positive\_Amount'. The rule is active and has the formula: Total\_Amount\_\_c <= 0. The error message is 'Order amount must be greater than zero.' The error location is set to 'Top of Page'. The rule was created by Pallepati Ritupriya on 9/19/2025, 4:06 AM, and modified by the same user on the same date and time. There are 'Edit' and 'Clone' buttons at the bottom.

### 3. Food Order – Require\_Delivery\_Agent

- Error Condition Formula: AND(ISPICKVAL(Status\_\_c, "Out for Delivery"),

ISBLANK(\$ObjectType.Delivery\_\_c.Fields.Delivery\_Agent\_\_c))

- Error Message: Please assign a Delivery Agent before marking order Out for Delivery.

The screenshot shows the 'Food Order Validation Rule' setup page under 'Food Order'. The left sidebar lists various configuration options: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, and Record Types. The main content area is titled 'Food Order Validation Rule' with a 'Help for this Page' link. It displays the 'Validation Rule Detail' section, which includes fields for 'Rule Name' (Require\_Delivery\_Agent), 'Error Condition Formula' (AND(ISPICKVAL(Status\_\_c, "Out for Delivery"), ISBLANK(\$ObjectType.Delivery\_\_c.Fields.Delivery\_Agent\_\_c))), 'Error Message' (Please assign a Delivery Agent before marking order Out for Delivery), and 'Active' status (checked). Below this, there are sections for 'Description', 'Created By' (Pallepati Ritupriya, 9/19/2025, 4:05 AM), and 'Modified By' (Pallepati Ritupriya, 9/19/2025, 4:05 AM). At the bottom are 'Edit' and 'Clone' buttons.

## Flow Builder

- Setup → Flow Builder → New Flow → Record-Triggered → Auto-launched Flow
- Record-Triggered Flows run automatically when a record is created or updated, enabling automation without user intervention.

### 1. Auto\_Create\_Delivery\_On\_Accepted

- **Trigger:** Record—Run After Save
- **Description:** When a Food Order is marked as Accepted, this flow automatically creates a Delivery record linked to the order.

The screenshot shows the 'Auto\_Create\_Delivery\_On\_Accepted' flow setup page under 'Flows'. The top navigation bar includes 'SETUP', 'Flows', and a back link 'Back to List: Flows'. The main content area is titled 'Flow Detail' for the flow labeled 'Auto\_Create\_Delivery\_On\_Accepted'. It shows the flow's description: 'When a Food Order is marked as Accepted, this Flow automatically creates a Delivery record linked to the Order.' The flow is of type 'Autolaunched Flow' with API name 'Auto\_Create\_Delivery\_On\_Accepted'. It was created by Pallepati Ritupriya on 9/19/2025 at 10:32 AM and activated by the same user on 9/19/2025 at 10:31 AM. The trigger is 'Record—Run After Save'. The 'Flow Versions' section shows one version of the flow, labeled 'Auto\_Create\_Delivery\_On\_Accepted 1', created by Pallepati Ritupriya on 9/19/2025 at 10:31 AM using the Flow Builder, with an active status and default mode.

## 2. Update\_Order\_on\_Delivery\_Completed

- **Trigger:** Record—Run After Save
- **Description:** When a Delivery is completed, this flow automatically updates the related Food Order status to Delivered.

The screenshot shows the 'Flows' section under 'SETUP'. A specific flow named 'Update Order on Delivery Completed' is selected. The 'Flow Detail' table includes fields like Flow Label (Update Order on Delivery Completed), Description (When a Delivery is completed, this Flow automatically updates the related Order's status to Delivered.), Flow API Name (Update\_Order\_on\_Delivery\_Completed), Namespace Prefix, Type (Autolaunched Flow), URL (/flow/Update\_Order\_on\_Delivery\_Completed), Environment (Default), Active Version (1), Trigger (Record—Run After Save), Modified By (Pallepati Ritupriya), and Created By (Pallepati Ritupriya). The 'Flow Versions' table shows one version (Version 1) with the same details as the main flow.

Action	Flow Label	Version	Description	Built with	Created Date	Type	Status	Progress Status	Run in Mode	API Version for Running the Flow
Open   Run   Deactivate	Update Order on Delivery Completed	1	When a Delivery is completed, this Flow automatically updates the related Order's status to Delivered.	Flow Builder	9/19/2025, 10:43 AM	Autolaunched Flow	Active	Activated	Default Mode	64.0

## 3. Order\_Confirmation Flow

- **Trigger:** Record—Run After Save
- **Description:** When a Food Order is placed, this flow automatically sends an order confirmation email to the customer.

The screenshot shows the 'Flows' section under 'SETUP'. A specific flow named 'Order\_Confirmation' is selected. The 'Flow Detail' table includes fields like Flow Label (Order\_Confirmation), Description, Flow API Name (Order\_Confirmation), Namespace Prefix, Type (Autolaunched Flow), URL (/flow/Order\_Confirmation), Environment (Default), Active Version (1), Trigger (Record—Run After Save), Modified By (Pallepati Ritupriya), and Created By (Pallepati Ritupriya). The 'Flow Versions' table shows one version (Version 1) with the same details as the main flow.

Action	Flow Label	Version	Description	Built with	Created Date	Type	Status	Progress Status	Run in Mode	API Version for Running the Flow
Open   Run   Deactivate	Order_Confirmation	1		Flow Builder	9/23/2025, 2:36 AM	Autolaunched Flow	Active	Activated	Default Mode	64.0

#### 4. Auto\_Email\_Complaint\_Resolved

- **Trigger:** Record—Run After Save
- **Description:** When a Complaint status is updated to “Resolved”, this flow automatically sends an email alert to notify the customer.

The screenshot shows the 'Flows' section under 'SETUP'. A specific flow named 'Auto\_Email\_Complaint\_Resolved' is selected. The 'Flow Detail' section displays the following information:

Flow Label	Auto_Email_Complaint_Resolved	Action Buttons
Description		[Edit] [Open] [Run] [Delete]
Environments	Default	Flow API Name: Auto_Email_Complaint_Resolved
Active Version	1	Namespace Prefix:
Trigger	Record—Run After Save	Type: Autolaunched Flow
Modified By	Pallepalli Ritupriya, 9/23/2025, 2:46 AM	URL: /flow/Auto_Email_Complaint_Resolved
		Activated/Deactivated By: Pallepalli Ritupriya, 9/23/2025, 2:46 AM
		Created By: Pallepalli Ritupriya, 9/23/2025, 2:45 AM

The 'Flow Versions' section shows one version of the flow:

Action	Flow Label	Version	Description	Built with	Created Date	Type	Status	Progress Status	Run in Mode	API Version for Running this Flow
Open   Run   Deactivate	Auto_Email_Complaint_Resolved	1		Flow Builder	9/23/2025, 2:45 AM	Autolaunched Flow	Active	Activated	Default Mode	64.0

#### Purpose / Outcome:

- Automates business processes to reduce manual work.
- Ensures data consistency and real-time updates across related objects.
- Supports accurate tracking of orders and deliveries.
- Improves customer engagement and operational efficiency.

#### Email Alerts

- Setup → Workflow & Approvals → Email Alerts → New Email Alert.
- Email alerts automatically send emails based on specific conditions or triggers in Salesforce, ensuring timely communication with customers or internal users.

#### 1. Order Confirmation Email Alert

- **Purpose:** Sends confirmation email to customer when a new order is placed.
- **Object:** Food\_Orders\_\_c.
- Setup → Email → Classic Email Templates → New Template.
- **Template Name:** Order Confirmation – Food Orders
- Template used to confirm orders to customers.

**Email Template**

**Subject** | Order Confirmation — {!Food\_Orders\_\_c.Name}

**Plain Text Preview**

```
Hi {!Food_Orders__c.Customer__r.Name}.

Thanks — your order {!Food_Orders__c.Name} has been received.

Order Total: ₹{!Food_Orders__c.Total_Amount__c}
Restaurant: {!Food_Orders__c.Restaurant__r.Name}
Order Date: {!Food_Orders__c.Order_Date__c}

We'll notify you when your order is out for delivery.

Regards,
Online Food Delivery CRM
```

**SETUP Email Alerts**

**Email Alert** Order\_Confirmation [Help for this Page](#)

[Rules Using This Email Alert \(0\)](#) | [Approval Processes Using This Email Alert \(0\)](#) | [Entitlement Processes Using This Email Alert \(0\)](#)

Email Alert Detail		<a href="#">Edit</a>	<a href="#">Delete</a>	<a href="#">Clone</a>
Description	Order_Confirmation	Email Template	Order Confirmation - Food Order	
Unique Name	Order_Confirmation	Object	Food Order	
From Email Address	Current User's email address			
Recipients	Food Order Owner Related Contact: Customer			
Additional Emails				
Created By	Pallepati Ritupriya, 9/23/2025, 1:24 AM	Modified By	Pallepati Ritupriya, 9/23/2025, 3:43 AM	

[Edit](#) [Delete](#) [Clone](#)

## 2. Complaint Resolved Email Alert

- Purpose:** Sends notification email to customer when a complaint is resolved.
- Object:** Complaint\_\_c
- Template Name:** Complaint Resolved - Notification
- Template used to notify customers when their complaint is resolved.

**Email Template**

**Subject** | Complaint Resolved — {!Complaint\_\_c.Name}

**Plain Text Preview**

```
Hi {!Complaint__c.Customer__r.Name}.

Your complaint ({!Complaint__c.Name}) for Order {!Complaint__c.Food_Order__r.Name} has been resolved.

Comments: {!Complaint__c.Comments__c}

If you have further issues, please reply.

Regards,
Support Team — Online Food Delivery CRM
```

**SETUP Email Alerts**

**Email Alert** Complaint Resolved [Help for this Page](#)

[Rules Using This Email Alert \(0\)](#) | [Approval Processes Using This Email Alert \(0\)](#) | [Entitlement Processes Using This Email Alert \(0\)](#)

Email Alert Detail		<a href="#">Edit</a>	<a href="#">Delete</a>	<a href="#">Clone</a>
Description	Complaint Resolved	Email Template	Complaint Resolved - Notification	
Unique Name	Complaint_Resolved	Object	Complaint	
From Email Address	Current User's email address			
Recipients	Related Contact: Customer			
Additional Emails				
Created By	Pallepati Ritupriya, 9/23/2025, 2:41 AM	Modified By	Pallepati Ritupriya, 9/23/2025, 2:41 AM	

[Edit](#) [Delete](#) [Clone](#)

## Phase 5: Apex Programming (Developer)

### Apex Classes & Objects

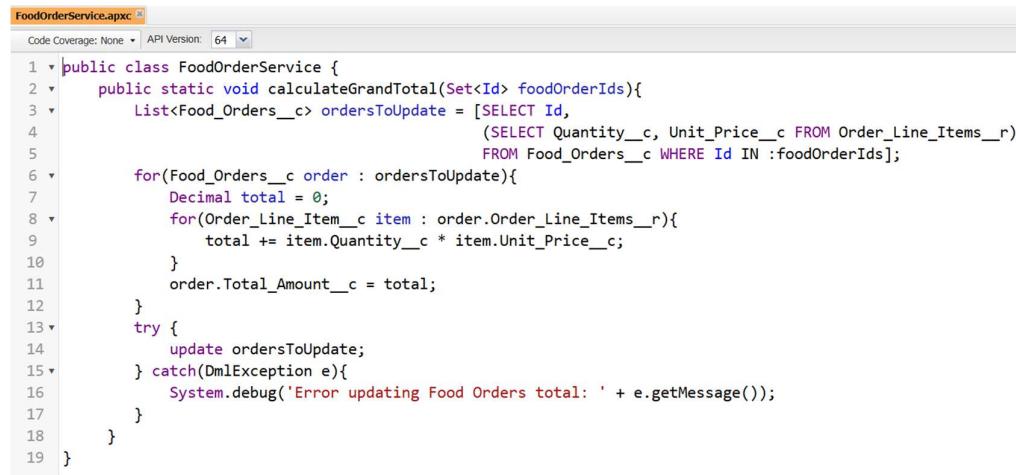
- Developer Console → File → New → Apex Class.

- **Purpose:** These classes encapsulate all business logic for Food Orders, Order Line Items, Menu Items, and Complaints. They ensure that triggers remain clean and maintainable while allowing reusability and separation of concerns.

## Apex Classes:

### 1. FoodOrderService

- **Object:** Food\_Orders\_\_c
- **Purpose:** Handles business logic for Food Orders, including calculating totals, updating statuses, and other operations triggered by Apex Triggers.



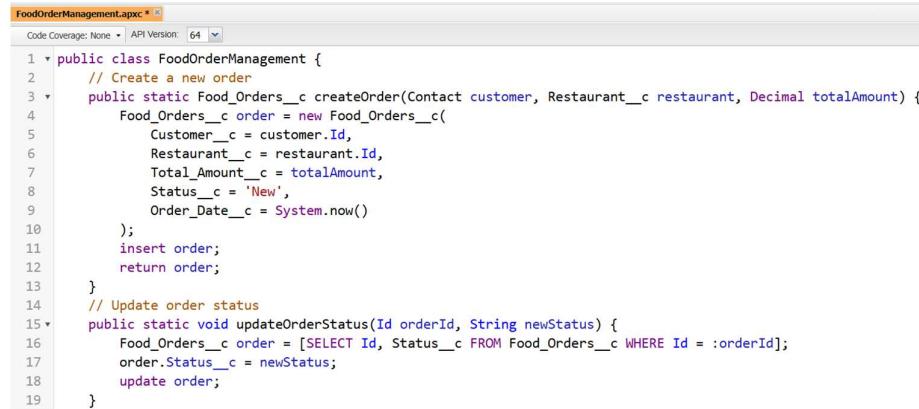
```

FoodOrderService.apxc
Code Coverage: None API Version: 64
1 public class FoodOrderService {
2     public static void calculateGrandTotal(Set<Id> foodOrderIds){
3         List<Food_Orders__c> ordersToUpdate = [SELECT Id,
4             (SELECT Quantity__c, Unit_Price__c FROM Order_Line_Items__r)
5             FROM Food_Orders__c WHERE Id IN :foodOrderIds];
6         for(Food_Orders__c order : ordersToUpdate){
7             Decimal total = 0;
8             for(Order_Line_Item__c item : order.Order_Line_Items__r){
9                 total += item.Quantity__c * item.Unit_Price__c;
10            }
11            order.Total_Amount__c = total;
12        }
13        try {
14            update ordersToUpdate;
15        } catch(DmlException e){
16            System.debug('Error updating Food Orders total: ' + e.getMessage());
17        }
18    }
19 }

```

### 2. FoodOrderManagement

- **Object:** Food\_Orders\_\_c
- **Purpose:** Acts as a centralized handler for Food Order triggers to keep triggers clean and maintainable.



```

FoodOrderManagement.apxc
Code Coverage: None API Version: 64
1 public class FoodOrderManagement {
2     // Create a new order
3     public static Food_Orders__c createOrder(Contact customer, Restaurant__c restaurant, Decimal totalAmount) {
4         Food_Orders__c order = new Food_Orders__c(
5             Customer__c = customer.Id,
6             Restaurant__c = restaurant.Id,
7             Total_Amount__c = totalAmount,
8             Status__c = 'New',
9             Order_Date__c = System.now()
10        );
11        insert order;
12        return order;
13    }
14    // Update order status
15    public static void updateOrderStatus(Id orderId, String newStatus) {
16        Food_Orders__c order = [SELECT Id, Status__c FROM Food_Orders__c WHERE Id = :orderId];
17        order.Status__c = newStatus;
18        update order;
19    }
}

```

```

20 // Calculate Grand Total from Order Line Items
21 public static Decimal calculateGrandTotal(Id orderId) {
22     Decimal total = 0;
23     List<Order_Line_Item__c> items = [SELECT Total__c FROM Order_Line_Item__c WHERE Food_Order__c = :orderId];
24     for(Order_Line_Item__c item : items){
25         total += item.Total__c;
26     }
27     return total;
28 }
29 }
```

### 3. ComplaintHandler

- Object: Complaint\_\_c
- Purpose: Manages business logic for complaints, including default values, validation, and related operations.



The screenshot shows the code editor interface for ComplaintHandler.apxc. The code is a Apex class named ComplaintHandler with a static method updateOrderOnComplaint. The code logic involves comparing new and old complaint records to identify food orders that have been resolved. It then updates the status of these food orders to 'Delivered'. A try-catch block is used to handle any DML exceptions during the update process.

```

1 public class ComplaintHandler {
2     public static void updateOrderOnComplaint(List<Complaint__c> newComplaints,
3                                                 Map<Id, Complaint__c> oldMap) {
4         Set<Id> orderIds = new Set<Id>();
5         for(Complaint__c comp : newComplaints){
6             Complaint__c oldComp = oldMap.get(comp.Id);
7             if(comp.Status__c == 'Resolved' && oldComp.Status__c != 'Resolved'){
8                 if(comp.Food_Order__c != null)
9                     orderIds.add(comp.Food_Order__c);
10            }
11        }
12        if(orderIds.isEmpty()) return;
13
14        List<Food_Orders__c> ordersToUpdate = [SELECT Id, Status__c FROM Food_Orders__c WHERE Id IN :orderIds];
15        for(Food_Orders__c order : ordersToUpdate){
16            order.Status__c = 'Delivered';
17        }
18        try {
19            update ordersToUpdate;
20        } catch(DmlException e){
21            System.debug('Error updating Food Orders from Complaint: ' + e.getMessage());
22        }
23    }
24 }
```

### 4. OrderLineItemHandler

- Object: Order\_Line\_Item\_\_c
- Purpose: Calculates totals for order line items and updates parent Food Orders. Ensures consistent data integrity.

```
OrderLineItemHandler.apxc
Code Coverage: None API Version: 64
1 public class OrderLineItemHandler {
2
3     public static void updateFoodOrderTotal(Map<Id, Order_Line_Item__c> newMap,
4                                             Map<Id, Order_Line_Item__c> oldMap,
5                                             Boolean isDelete) {
6         Set<Id> foodOrderIds = new Set<Id>();
7
8         if(isDelete){
9             for(Order_Line_Item__c oldItem : oldMap.values()){
10                 if(oldItem.Food_Order__c != null)
11                     foodOrderIds.add(oldItem.Food_Order__c);
12             }
13         } else {
14             for(Order_Line_Item__c item : newMap.values()){
15                 if(item.Food_Order__c != null)
16                     foodOrderIds.add(item.Food_Order__c);
17             }
18         }
19
20         FoodOrderService.calculateGrandTotal(foodOrderIds);
21     }
22 }
```

## Apex Triggers

- Developer Console → File → New → Apex Trigger.
- **Purpose:** Apex triggers automate business processes when records are created, updated, or deleted. In your Food Delivery CRM, they handle actions like updating Order totals and creating Delivery records, while keeping logic centralized and bulk-safe via handler classes.

### ComplaintTrigger

- **Object:** Complaint\_\_c
- **Purpose:** Automates actions when complaints are created or updated, such as assigning owners or sending notifications via the handler class.

```
ComplaintTrigger.apxt *
Code Coverage: None API Version: 64
1 trigger ComplaintTrigger on Complaint__c (after update) {
2     ComplaintHandler.updateOrderOnComplaint(Trigger.new, Trigger.oldMap);
3 }
```

### FoodOrderTrigger

- **Object:** Food\_Orders\_\_c
- **Purpose:** Handles actions after an order is updated, like calculating the grand total or triggering other business processes via FoodOrderService.

```

FoodOrderTrigger.apxt *
Code Coverage: None | API Version: 64
1 trigger FoodOrderTrigger on Food_Orders__c (after update) {
2     Set<Id> foodOrderIds = new Set<Id>();
3
4     for(Food_Orders__c order : Trigger.new){
5         Food_Orders__c oldOrder = Trigger.oldMap.get(order.Id);
6
7         // If any relevant field changed (e.g., Status), we can add it here
8         if(order.Status__c != oldOrder.Status__c){
9             foodOrderIds.add(order.Id);
10        }
11    }
12
13    if(!foodOrderIds.isEmpty()){
14        // Recalculate totals or other logic in your service class
15        FoodOrderService.calculateGrandTotal(foodOrderIds);
16    }
17 }

```

## OrderLineItemTrigger

- Object:** Order\_Line\_Item\_\_c
- Purpose:** Updates the parent Food Order's total whenever order line items are inserted, updated, or deleted, keeping order amounts accurate.

```

OrderLineItemTrigger.apxt *
Code Coverage: None | API Version: 64
1 trigger OrderLineItemTrigger on Order_Line_Item__c (after insert, after update, after delete) {
2     if(Trigger.isInsert || Trigger.isUpdate){
3         OrderLineItemHandler.updateFoodOrderTotal(Trigger.newMap, Trigger.oldMap, false);
4     } else if(Trigger.isDelete){
5         OrderLineItemHandler.updateFoodOrderTotal(null, Trigger.oldMap, true);
6     }
7 }

```

## Trigger Design Pattern

In this project, we used the Trigger + Handler design pattern to keep business logic separated from triggers.

- Triggers only handle context (before/after insert, update, delete).
- Handler classes contain the actual business logic.

This makes the code more readable, reusable, and easier to maintain.

Where used:

- ComplaintTrigger → ComplaintHandler (manage complaint creation and updates)
- FoodOrderTrigger → FoodOrderService (automate total calculation, validations on orders)

- OrderLineItemTrigger → OrderLineItemHandler (update order totals when line items change)

## SOQL & SOSL

### SOQL (Salesforce Object Query Language)

- Similar to SQL, but used only for Salesforce data.
- Allows you to query records from standard and custom objects.
- You can filter, order, and select specific fields.

### SOSL (Salesforce Object Search Language)

- Used for full-text search across multiple objects at once.
- Useful when you don't know which object/field has the data.
- Returns results grouped by object.

## Collections

Collections are data structures that hold multiple records in memory. In Salesforce Apex, we mainly use List, Set, and Map.

### 1. List

- Ordered collection (like an array in Java).
- Can contain duplicate records.
- Indexed (you can access by position).

Used in SOQL queries, inserting multiple records at once, and storing results from Apex methods.

### 2. Set

- Unordered collection of unique elements.
- Automatically removes duplicates.
- Useful when working with Ids.

Used in Service classes and Triggers to avoid duplicate processing.

### 3. Map

- Key-value pair collection.

- Keys must be unique; values can repeat.
- Very powerful when working with relationships.

Useful in Trigger Handlers when you need to update parent Food Order totals based on child Order Line Items.

## Control Statements in Apex

Control statements in Apex determine the flow of execution in the program. They allow conditional execution and iteration over data.

- **If–Else:** Used to make decisions based on conditions.
- **For Loop:** Executes repeatedly for a fixed number of iterations or over a collection.

### In our project:

We used if–else for checking order and complaint statuses, and for loops for iterating through records in triggers and handlers.

## Exception Handling in Apex

Exception handling in Apex ensures that runtime errors don't break execution and are managed gracefully.

- **Try-Catch-Finally:** Used to catch and handle exceptions.
- **System.debug & Custom Messages:** Help identify and log issues.
- **Throw:** Used to raise custom exceptions.

### In our project:

We applied exception handling in service classes and triggers to make sure errors (like invalid record updates) are handled properly without failing the transaction.

## Test Classes

- Developer Console → File → New → Apex Class → Add @isTest Annotation`
- **Purpose:** Test classes validate that your Apex triggers and classes work as expected. Salesforce requires test coverage to deploy to production, and they also ensure your business logic is correct for different scenarios.

## FoodOrderServiceTest

- **Purpose:** Tests the FoodOrderService class and FoodOrderTrigger.
- **Key Actions Tested:**

- Creating sample Restaurant, Food Order, Menu Item, and Order Line Item records.
- Verifying that order totals are correctly calculated when line items are inserted or updated.
- Updating food order status to ensure trigger logic runs properly.



```

FoodOrderServiceTest.apxc
Code Coverage: None | API Version: 64
1 @isTest
2 public class FoodOrderServiceTest {
3
4     @testSetup
5
6     static void setupData(){
7         // Create a Restaurant
8         Restaurant__c res = new Restaurant__c(
9             Name = 'Test Restaurant',
10            Location__c = 'Test Area',
11            Cuisine__c = 'Italian'
12        );
13        insert res;
14
15        // Create a Food Order
16        Food_Orders__c order = new Food_Orders__c(
17            Status__c = 'Pending',
18            Restaurant__c = res.Id,
19            Total_Amount__c = 0
20        );
21        insert order;
22
23        // Create Menu Item
24        Menu_Item__c menu = new Menu_Item__c(
25            Name = 'Pizza',
26            Price__c = 100,
27            Restaurant__c = res.Id
28        );
29        insert menu;
30
31        // Create Order Line Item
32        Order_Line_Item__c line = new Order_Line_Item__c(
33            Food_Order__c = order.Id,
34            Menu_Item__c = menu.Id,
35            Quantity__c = 2,
36            Unit_Price__c = 100
37        );
38        insert line;
39    }
40
41    static testMethod void testCalculateGrandTotal(){
42        Food_Orders__c order = [SELECT Id, Total_Amount__c FROM Food_Orders__c LIMIT 1];
43
44        // Call the service method
45        Set<Id> orderIds = new Set<Id>{order.Id};
46        FoodOrderService.calculateGrandTotal(orderIds);
47
48        // Verify the total
49        order = [SELECT Id, Total_Amount__c FROM Food_Orders__c WHERE Id = :order.Id];
50        System.assertEquals(200, order.Total_Amount__c);
51    }
52
53    static testMethod void testFoodOrderTrigger(){
54        Food_Orders__c order = [SELECT Id, Status__c, Total_Amount__c FROM Food_Orders__c LIMIT 1];
55
56        // Update status to simulate trigger (no Delivery)
57        order.Status__c = 'Processing';
58        update order;
59
60        // Verify totals remain correct
61        Food_Orders__c updatedOrder = [SELECT Id, Total_Amount__c FROM Food_Orders__c WHERE Id = :order.Id];
62        System.assertEquals(200, updatedOrder.Total_Amount__c);
63    }
}

```

## ComplaintHandlerTest

- **Purpose:** Tests ComplaintHandler and ComplaintTrigger.
- **Key Actions Tested:**
  - Creating sample Restaurant, Food Order, and Complaint records.
  - Updating complaint status to verify related Food Order status or notifications are handled correctly.

```

ComplaintHandlerTest.apxc *
Code Coverage: None API Version: 64
1 @isTest
2 public class ComplaintHandlerTest {
3     @testSetup
4     static void setupData(){
5         // Restaurant
6         Restaurant__c res = new Restaurant__c(Name='Test Res', Location__c='Area', Cuisine__c='Italian');
7         insert res;
8         // Food Order
9         Food_Orders__c order = new Food_Orders__c(Status__c='Pending', Restaurant__c=res.Id, Total_Amount__c=0);
10        insert order;
11        // Complaint
12        Complaint__c comp = new Complaint__c(Food_Order__c=order.Id, Status__c='Open');
13        insert comp;
14    }
15
16    static testMethod void testUpdateOrderOnComplaint(){
17        Complaint__c comp = [SELECT Id, Status__c, Food_Order__c FROM Complaint__c LIMIT 1];
18        // Resolve complaint
19        comp.Status__c = 'Resolved';
20        update comp;
21        // Verify Food Order status updated (if your handler does that)
22        Food_Orders__c order = [SELECT Id, Status__c FROM Food_Orders__c WHERE Id = :comp.Food_Order__c];
23        System.assertEquals('Delivered', order.Status__c);
24    }
25 }

```

## OrderLineItemHandlerTest

- Purpose:** Tests OrderLineItemHandler and OrderLineItemTrigger.
- Key Actions Tested:**
  - Creating sample Restaurant, Food Order, Menu Item, and Order Line Item records.
  - Updating order line item quantity to ensure Food Order totals are updated automatically.

```

OrderLineItemHandler.apxc *
Code Coverage: None API Version: 64
1 public class OrderLineItemHandler {
2     public static void updateFoodOrderTotal(Map<Id, Order_Line_Item__c> newMap,
3                                             Map<Id, Order_Line_Item__c> oldMap,
4                                             Boolean isDelete) {
5         Set<Id> foodOrderIds = new Set<Id>();
6
7         if(isDelete){
8             for(Order_Line_Item__c oldItem : oldMap.values()){
9                 if(oldItem.Food_Order__c != null)
10                     foodOrderIds.add(oldItem.Food_Order__c);
11             }
12         } else {
13             for(Order_Line_Item__c item : newMap.values()){
14                 if(item.Food_Order__c != null)
15                     foodOrderIds.add(item.Food_Order__c);
16             }
17         }
18
19         FoodOrderService.calculateGrandTotal(foodOrderIds);
20     }
21 }

```

## Phase 6: User Interface Development

### Lightning App Builder

- Setup → App Manager → New Lightning App
- Used Lightning App Builder to create the Online Food Delivery CRM app.
- Added tabs for key objects: Food Orders, Order Line Items, Menu Items, Complaints, Restaurants.
- Configured navigation and page assignments for relevant profiles (Restaurant Manager, Delivery Agent, Executive).

The screenshot shows the 'App Details & Branding' section of the Lightning App Builder. On the left, a sidebar lists 'App Settings' and 'App Details & Branding'. The main area contains fields for 'App Name' (Online Food Delivery CRM), 'Developer Name' (Online\_Food\_Delivery\_CRM), 'Image' (a placeholder box with an 'Upload' button), 'Primary Color Hex Value' (#2B22C7), and a 'Description' box (Manage orders, deliveries, menu items, and complaints). A 'Org Theme Options' checkbox is also present. Below this is an 'App Launcher Preview' section showing a blue square icon with 'OF' and the app details.

The screenshot shows the 'App Options' section of the Lightning App Builder. On the left, a sidebar lists 'App Settings' and 'App Options'. The main area contains sections for 'Navigation and Form Factor' (with 'Standard navigation' selected), 'Setup and Personalization' (with 'Setup Experience' set to 'Setup (full set of Setup options)'), and 'App Personalization Settings' (with checkboxes for end user personalization, temporary tabs, and Omni-Channel sidebar). Under 'App Options', there are also sections for 'Utility Items (Desktop Only)', 'Navigation Items', and 'User Profiles'.

Lightning App Builder | App Settings | Pages | Online Food Delivery CRM

### App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items**
- User Profiles

#### Navigation Items

Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename them. Navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

**Available Items**

Type to filter list...

- Accounts
- Activation Targets
- Activations
- All Sites
- Alternative Payment Methods
- Analytics

**Selected Items**

- Food Orders
- Menu Item
- Restaurants
- Order Line Items
- Complaints
- Deliveries

>Create ▾

Lightning App Builder | App Settings | Pages | Online Food Delivery CRM

### App Settings

- App Details & Branding
- App Options
- Utility Items (Desktop Only)
- Navigation Items
- User Profiles**

#### User Profiles

Choose the user profiles that can access this app.

**Available Profiles**

Type to filter list...

- Analytics Cloud Integration User
- Analytics Cloud Security User
- Anypoint Integration

**Selected Profiles**

- System Administrator
- Restaurant\_Manager\_Profile
- Delivery\_Agent\_Profile
- Executive\_Profile

Online Food Deliver... | Food Orders | Menu Item | Restaurants | Order Line Items | Complaints | Deliveries

Food Orders | Recently Viewed |

0 items • Updated 11 minutes ago

New Import Change Owner Assign Label

Grid List C E G Y

Nothing to see here  
There's nothing in your list yet. Try adding a new record.

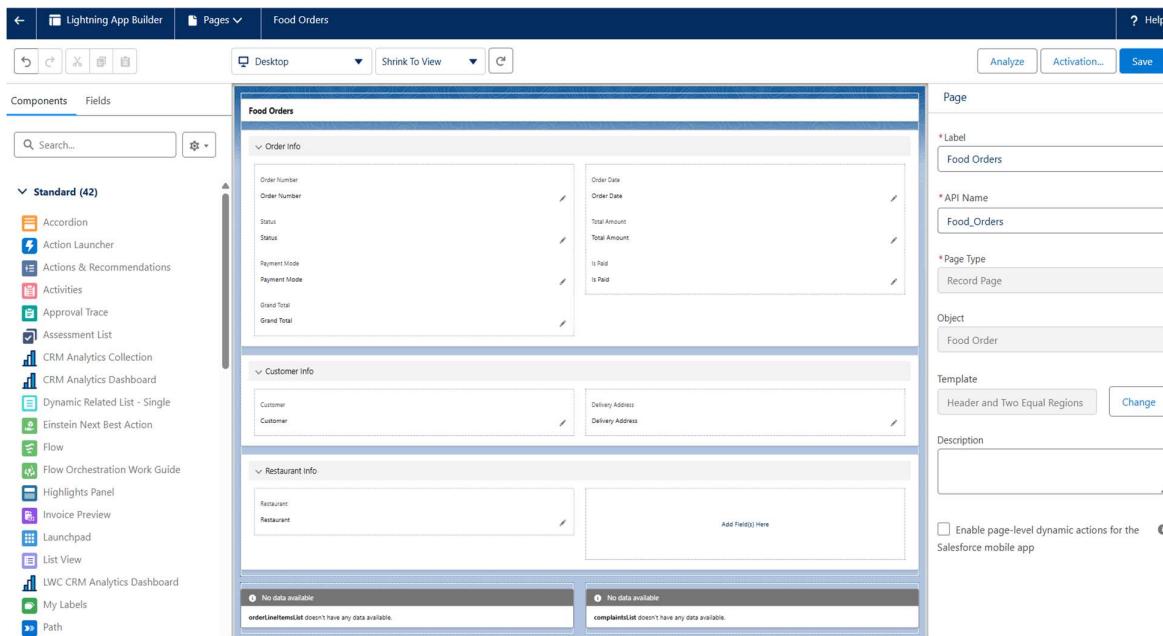
To Do List Report Chart

## Record Pages

- Record Pages in Salesforce provide a customized view of individual records for each object.

### Food Orders Record Page

- Setup → Object Manager → Food\_Orders\_\_c → Lightning Record Pages → New
- Added Record Detail component to show Order Info and Customer Info.
- Added three LWCs to display:
  1. **Food Order Summary** – key order details.
  2. **Order Line Items List** – items in the order.
  3. **Complaints List** – complaints related to the order.



### Restaurant Record Page

- Setup → Object Manager → Restaurant\_\_c → Lightning Record Pages → New
- Added Record Detail component to show Restaurant details (Name, Location, Cuisine).
- Added one LWC to display: Menu Items List – menu items available for that restaurant.

The screenshot shows the Lightning App Builder interface. At the top, there's a navigation bar with 'Lightning App Builder', 'Pages', and a search bar. Below it is a toolbar with icons for back, forward, refresh, and other actions. On the left, a sidebar titled 'Components' lists 'Standard (42)' components such as Accordion, Action Launcher, and Approval Trace. A search bar and a component preview area are also present. The main workspace displays a page for a 'Restaurant' object. The page has three sections: 'Contact Info' (Address, Contact Phone), 'Restaurant Info' (Restaurant Name, Location, Cuisine), and 'System Information' (Created By, Last Modified By). A message at the bottom states 'No data available' and 'menuItemsList doesn't have any data available.' On the right, a 'Page' configuration panel shows fields for 'Label' (Restaurant), 'API Name' (Restaurant), 'Page Type' (Record Page), 'Object' (Restaurant), 'Template' (Grouped Header and One ...), and a 'Description' field. There are also buttons for 'Analyze', 'Activation...', and 'Save'.

## Utility Bar

- Setup → App Manager → Food Delivery CRM App → Edit → Utility Bar
- Added To-Do List → for quick task management.
- Added Report Chart → to show key metrics like total orders or complaints.

**Purpose:** Provides users with quick access to frequently used tools, improving productivity and navigation.

The screenshot shows the 'App Settings' screen in the Lightning App Builder. The left sidebar includes 'App Details & Branding', 'App Options', and 'Utility Items (Desktop Only)', which is currently selected. Under 'Utility Items (Desktop Only)', there are sections for 'Navigation Items' and 'User Profiles'. The main workspace displays 'Utility Items (Desktop Only)' with a sub-section for 'To Do List' and 'Report Chart'. The 'To Do List' item is highlighted. On the right, there's a 'Utility Item Properties' panel with fields for 'Label' (To Do List), 'Icon' (task), 'Panel Width' (427), 'Panel Height' (680), and a checked 'Start automatically' checkbox. There are also up and down arrows to reorder items and a 'Remove' button.

To Do List    Report Chart

## Lightning Web Components (LWC)

**Purpose:** LWCs provide a modern, responsive, and reusable UI for displaying and interacting with Salesforce data in the Food Delivery CRM app.

### 1. Food Order Summary

- Displays a list of food orders with key details like name and total amount.
- Added to the Food Orders Record Page.

```
↳ foodOrderSummary.html ●
force-app > main > default > lwc > foodOrderSummary > ↳ foodOrderSummary.html > ...
1   <template>
2     <lightning-card title="Food Orders">
3       <template if:true={orders}>
4         <ul>
5           <template for:each={orders} for:item="order">
6             <li key={order.Id}>
7               {order.Name} - ${order.Total_Amount__c}
8             </li>
9           </template>
10        </ul>
11      </template>
12      <template if:false={orders}>
13        <p>No orders available.</p>
14      </template>
15    </lightning-card>
16  </template>
17
```

```
JS foodOrderSummary.js X
force-app > main > default > lwc > foodOrderSummary > JS foodOrderSummary.js > ...
1 import { LightningElement, wire } from 'lwc';
2 import getFoodOrders from '@salesforce/apex/FoodOrderController.getFoodOrders';
3
4 export default class FoodOrderSummary extends LightningElement {
5   orders; // stores the food orders
6   error; // stores errors if any
7
8   @wire(getFoodOrders)
9   wiredOrders({ error, data }) {
10     if (data) {
11       this.orders = data;
12       this.error = undefined;
13     } else if (error) {
14       this.error = error;
15       this.orders = undefined;
16     }
17   }
18 }
```

```
↳ foodOrderSummary.js-meta.xml ×
force-app > main > default > lwc > foodOrderSummary > ↳ foodOrderSummary.js-meta.xml > ...
1   <?xml version="1.0" encoding="UTF-8"?>
2   <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3     <apiVersion>64.0</apiVersion>
4     <isExposed>true</isExposed>
5     <targets>
6       <target>lightning__RecordPage</target>
7       <target>lightning__AppPage</target>
8     </targets>
9   </LightningComponentBundle>
```

## 2. Order Line Items List

- Shows items (menu items, quantity, unit price) linked to a food order.
- Added to the Food Orders Record Page.

```
↳ orderLineItemsList.html ×
force-app > main > default > lwc > orderLineItemsList > ↳ orderLineItemsList.html > ...
1   <template>
2     <lightning-card title="Order Line Items">
3       <template if:true={lineItems.length}>
4         <ul>
5           <template for:each={lineItems} for:item="item">
6             <li key={item.Id}>
7               {item.Menu_Item__r.Name} - Qty: {item.Quantity__c} - ${item.Total__c}
8             </li>
9           </template>
10        </ul>
11      </template>
12      <template if:false={lineItems.length}>
13        <p>No line items found.</p>
14      </template>
15    </lightning-card>
16  </template>
```

```
JS orderLineItemsList.js ×
force-app > main > default > lwc > orderLineItemsList > JS orderLineItemsList.js > ...
1  import { LightningElement, api, wire } from 'lwc';
2  import getOrderLineItems from '@salesforce/apex/OrderLineItemController.getOrderLineItems';
3
4  export default class OrderLineItemsList extends LightningElement {
5    @api recordId;
6    lineItems;
7
8    @wire(getOrderLineItems, { orderId: '$recordId' })
9    wiredItems({ error, data }) {
10      if(data){
11        this.lineItems = data;
12      } else if(error){
13        this.error = error;
14      }
15    }
16 }
```

```
orderLineItemsList.js-meta.xml ×
force-app > main > default > lwc > orderLineItemsList > orderLineItemsList.js-meta.xml > ...
1   <?xml version="1.0" encoding="UTF-8"?>
2   <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3     <apiVersion>64.0</apiVersion>
4     <isExposed>true</isExposed>
5     <targets>
6       <target>lightning__RecordPage</target>
7       <target>lightning__AppPage</target>
8     </targets>
9   </LightningComponentBundle>
```

### 3. Complaints List

- Displays complaints related to a food order.
- Added to the Food Orders Record Page.

```
complaintsList.html ×
force-app > main > default > lwc > complaintsList > complaintsList.html > ...
1   <template>
2     <lightning-card title="Complaints">
3       <template if:true={complaints.length}>
4         <ul>
5           <template for:each={complaints} for:item="c">
6             <li key={c.Id}>
7               {c.Complaint_Number__c} - {c.Status__c} ({c.Complaint_Type__c})
8             </li>
9           </template>
10        </ul>
11      </template>
12      <template if:false={complaints.length}>
13        <p>No complaints found.</p>
14      </template>
15    </lightning-card>
16  </template>
```

```
complaintsList.js ×
force-app > main > default > lwc > complaintsList > complaintsList.js > ...
1 import { LightningElement, api, wire } from 'lwc';
2 import getComplaints from '@salesforce/apex/ComplaintController.getComplaints';
3
4 export default class ComplaintsList extends LightningElement {
5   @api recordId; // Food Order Id
6   complaints;
7
8   @wire(getComplaints, { orderId: '$recordId' })
9   wiredComplaints({ error, data }) {
10     if(data){
11       this.complaints = data;
12     } else if(error){
13       this.error = error;
14     }
15   }
16 }
```

```
complaintsList.js-meta.xml X
force-app > main > default > lwc > complaintsList > complaintsList.js-meta.xml > ...
1  <?xml version="1.0" encoding="UTF-8"?>
2  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3    <apiVersion>64.0</apiVersion>
4    <isExposed>true</isExposed>
5    <targets>
6      <target>lightning__RecordPage</target>
7      <target>lightning__AppPage</target>
8    </targets>
9  </LightningComponentBundle>
```

#### 4. Menu Items List

- Shows all menu items available for a restaurant.
- Added to the Restaurant Record Page.

```
menultemsList.html X
force-app > main > default > lwc > menultemsList > menultemsList.html > ...
1  <template>
2    <lightning-card title="Menu Items">
3      <template if:true={menuItems.length}>
4        <ul>
5          <template for:each={menuItems} for:item="menu">
6            <li key={menu.Id}>
7              {menu.Name} - ${menu.Price__c} ({menu.Category__c})
8            </li>
9          </template>
10         </ul>
11       </template>
12       <template if:false={menuItems.length}>
13         <p>No menu items found.</p>
14       </template>
15     </lightning-card>
16   </template>
```

```
JS menultemsList.js X
force-app > main > default > lwc > menultemsList > menultemsList.js > ...
1  import { LightningElement, api, wire } from 'lwc';
2  import getMenuItems from '@salesforce/apex/MenuItemController.getMenuItems';
3
4  export default class MenultemsList extends LightningElement {
5    @api recordId; // Restaurant Id
6    menuItems;
7
8    @wire(getMenuItems, { restaurantId: '$recordId' })
9    wiredItems({ error, data }) {
10      if(data){
11        this.menuItems = data;
12      } else if(error){
13        this.error = error;
14      }
15    }
16 }
```

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>64.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
    </targets>
</LightningComponentBundle>

```

## Apex with LWC

### Purpose:

Apex controllers provide backend logic for LWCs. They handle SOQL queries, apply business logic, and return data to components. This allows LWCs to display dynamic Salesforce data.

#### 1. FoodOrderController

- **Method:** getFoodOrders()
- Returns the latest Food Orders with Id, Name, and Total Amount.
- Used by Food Order Summary LWC.

```

public with sharing class FoodOrderController {
    @AuraEnabled(cacheable=true)
    public static List<Food_Orders__c> getFoodOrders() {
        return [SELECT Id, Name, Total_Amount__c FROM Food_Orders__c LIMIT 10];
    }
}

```

#### 2. OrderLineItemController

- **Method:** getOrderLineItems(Id orderId)
- Returns items linked to a specific Food Order.
- Used by Order Line Items LWC.

```

public with sharing class OrderLineItemController {
    @AuraEnabled(cacheable=true)
    public static List<Order_Line_Item__c> getOrderLineItems(Id orderId) {
        return [SELECT Id, Quantity__c, Total__c, Menu_Item__r.Name
                FROM Order_Line_Item__c
                WHERE Food_Order__c = :orderId];
    }
}

```

### 3. ComplaintController

- **Method:** getComplaints(Id orderId)
- Returns complaints raised for a given Food Order.
- Used by Complaints List LWC.

```
ComplaintController.apxc
Code Coverage: None API Version: 64
1 public with sharing class ComplaintController {
2     @AuraEnabled(cacheable=true)
3     public static List<Complaint__c> getComplaints(Id orderId) {
4         return [SELECT Id, Name, Status__c, Complaint_Type__c
5                 FROM Complaint__c
6                 WHERE Food_Order__c = :orderId];
7     }
8 }
```

### 4. MenuItemController

- **Method:** getMenuItems(Id restaurantId)
- Returns all menu items for a restaurant.
- Used by Menu Items List LWC.

```
MenuItemController.apxc
Code Coverage: None API Version: 64
1 public with sharing class MenuItemController {
2     @AuraEnabled(cacheable=true)
3     public static List<Menu_Item__c> getMenuItems(Id restaurantId) {
4         return [SELECT Id, Name, Price__c, Category__c
5                 FROM Menu_Item__c
6                 WHERE Restaurant__c = :restaurantId];
7     }
8 }
```

## Wire Adapters

- **Purpose:** Provide a reactive way to fetch Salesforce data directly into LWCs without manually calling Apex.
- Used @wire with getFoodOrders in Food Order Summary LWC.
- Orders list auto-refreshes whenever data changes in Salesforce.

## Imperative Apex Calls

- **Purpose:** Used for on-demand server calls triggered by user actions (button clicks, form submissions).

- Example: When a user clicks “Place Order”, an imperative call to an Apex method can insert a Food\_Orders\_\_c record and return confirmation.
- Allows handling logic like validation and conditional record creation.

## Navigation Service

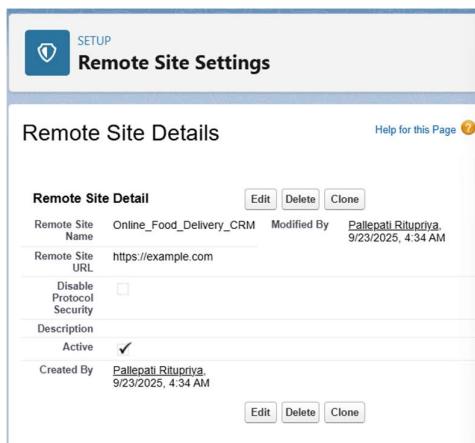
- **Purpose:** Lets LWCs programmatically navigate users to record pages, list views, or other Lightning pages.
- After placing a Food Order, the user can be redirected to that Food Order Record Page.
- Ensures smooth user flow from creation → viewing order details.

## Phase 7: Integration & External Access

Although external integrations are not fully implemented in this project, Salesforce provides powerful features to connect with outside systems. For the Food Delivery CRM, we have prepared the environment for future integrations.

### Remote Site Settings

- Setup → Security → Remote Site Settings → New.
- Added a sample external URL (e.g., <https://example.com>) to allow Salesforce to make callouts.
- **Purpose:** Ensures Salesforce can securely communicate with external APIs like payment gateways or delivery tracking systems.



## Future Scope (Not Implemented)

- Named Credentials – to securely store authentication details for external services.
- REST Callouts – to fetch delivery partner status or payment confirmation.
- External Services / Salesforce Connect – to link with external databases or microservices.

## Purpose / Outcome

- Prepares the CRM for real-world integration with payment systems, Google Maps, or delivery tracking.
  - Provides a pathway for future scalability without changing the core application.
- 

## Phase 8: Data Management & Deployment

### Data Import Wizard

#### Purpose:

- To quickly import small datasets (like Restaurants, Menu Items, Food Orders) into Salesforce without using external tools.
- Helps in setting up demo data for testing your flows, LWCs, triggers, and reports.

#### Steps:

1. Setup → Data → Data Import Wizard → Launch Wizard.
2. Uploaded CSV file(Restaurant.csv) containing the records.
3. Map the CSV columns to Salesforce fields.

### Object: Restaurant\_\_c

Getting closer

Choose data Edit mapping Start import

Import your Data into Salesforce  
You can import up to 50,000 records at a time.

Help for this page ⓘ

What kind of data are you importing? ⓘ

Standard objects Custom objects

Complaints Deliveries Food Orders

What do you want to do? ⓘ

Add new records

Match by: -None-

Which User field in your file designates record owners? ⓘ

-None-

Trigger workflow rules and processes? ⓘ

Trigger workflow rules and processes for new and updated records

Where is your data located? ⓘ

Drag CSV file here to upload

CSV

File Choose File Restaurant.csv

Character Code ⓘ

ISO-8859-1 General US & Western European, ISO-LATIN-1

Values Separated By

Cancel Previous Next

Almost done

Choose data Edit mapping Start import

### Edit Field Mapping: Restaurants

Your file has been auto-mapped to existing Salesforce fields, but you can edit the mappings if you wish. Unmapped fields will not be imported.

[Help for this page](#) ?

Edit	Mapped Salesforce Object	CSV Header	Example	Example	Example
			Spice '	Pasta	Sushi World
Change	Restaurant ...	Name	Hyderabad	Banga	Mumbai
Change	Location	Location	Indian	Italian	Japanese
Change	Cuisine	Cuisine			

Cancel Previous **Next**

Great job

Choose data Edit mapping Start import

### Review & Start Import

[Help for this page](#) ?

Review your import information and click Start Import.

Your selections:

Your import **will include:**

Your import **will not include:**

- Restaurants ✓
- Add new records ✓
- Restaurant.csv ✓

Mapped fields

3

Unmapped fields

0

Cancel Previous **Start Import**

**Congratulations,** your import has started!

Click OK to view your import status on the Bulk Data Load Job page.

**OK**

## Object: Menu\_Item\_\_c

Getting closer...

Choose data Edit mapping Start import

Import your Data into Salesforce  
You can import up to 50,000 records at a time.

Help for this page ?

What kind of data are you importing? [?](#)

Standard objects	Custom objects
Complaints	>
Deliveries	>
Food Orders	>
Menu Item	>

What do you want to do? [?](#)

Add new records	✓
Match by: <a href="#">?</a>	-None-
Which User field in your file designates record owners? <a href="#">?</a>	-None-
Which Restaurant field in your file do you want to match against to set the Restaurant lookup field? <a href="#">?</a>	-None-

Where is your data located? [?](#)

Drag CSV file here to upload

CSV  
File [Choose File](#) | Menu Items.csv  
Character Code [?](#)  
ISO-8859-1 (General US & Western European, ISO-LATIN-1)  
Values Separated By

Cancel Previous Next

Almost done

Choose data Edit mapping Start import

Help for this page ?

fields will not be imported:

Edit	Mapped Salesforce Object	CSV Header	Example	Example	Example
Change	Menu Item ...	Name	Veg Bi	Margh	Sushi Roll
Change	Price	Price	250	350	400
Change	Restaurant	Restaurant	Spice	Pasta	Sushi World

Cancel Previous Next

Great job

Choose data Edit mapping Start import

### Review & Start Import

Help for this page ?

Review your import information and click Start Import.

Your selections:

Your import **will include:**

Your import **will not include:**

Menu Item ✓  
Add new records ✓  
Menu Items.csv ✓

Mapped fields	Unmapped fields
3	0

Cancel Previous Start Import

## Duplicate Rules

- Setup → Duplicate Rules → New Rule → Object
- Duplicate Rules in Salesforce ensure data quality by preventing users from creating or importing duplicate records.

### Restaurant Duplicate Rule

- Prevents the same restaurant (with same name and location) from being entered twice.
- **Matching Criteria:**
  - Restaurant Name → Exact Match
  - Location → Exact Match

The screenshot shows the 'Duplicate Rule Detail' page for a 'Restaurant Duplicate Rule'. The page has a header with 'd SETUP' and 'Duplicate Rules'. Below the header, it says 'Restaurant Duplicate Rule' and 'Restaurant Duplicate Rule'. There are buttons for 'Edit', 'Delete', 'Clone', and 'Activate'. The rule details include:

- Rule Name:** Restaurant Duplicate Rule
- Description:** Prevents duplicate restaurant entries.
- Object:** Restaurant
- Record-Level Security:** Enforce sharing rules
- Action On Create:** Block (Operations On Create:  Alert  Report)
- Action On Edit:** Block (Operations On Edit:  Alert  Report)
- Alert Text:** Use one of these records?
- Active:**
- Matching Rule:** **⚠️ Restaurant Duplicate Rule** (highlighted in green)  
matching rule **Mapped** (highlighted in green)
- Matching Criteria:** (Restaurant: Name EXACT MatchBlank = FALSE) AND (Restaurant: Location EXACT MatchBlank = FALSE)
- Conditions:** None
- Created By:** Pallepati Ritupriya
- Modified By:** Pallepati Ritupriya, 9/23/2025

## VS Code & SFDX

- VS Code with Salesforce DX (SFDX) allows developers to retrieve, edit, and deploy Salesforce metadata in a local development environment. It is particularly useful for version control, collaborative development, and deploying changes to different orgs.

## Package.xml

- Acts as a manifest listing Salesforce metadata to retrieve or deploy, enabling VS Code & SFDX to move components between orgs efficiently.

```
≡ package.xml ●
manifest > ≡ package.xml
1   <?xml version="1.0" encoding="UTF-8"?>
2   <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3       <types>
4           <members>********
```

## Terminal

```
PS C:\Users\ritup\OneDrive\Documents\OnlineFoodDelivery\Online Food Delivery CRM> sf project retrieve start --manifest/package.xml
>>
»   Warning: @salesforce/cli update available from 2.104.6 to 2.106.6.

Retrieving v59.0 metadata from ritupriya.p176698@agentforce.com using the v64.0 SOAP API

- [4/4] Done 0ms

Status: Succeeded

----- Retrieving Metadata -----


Retrieving v59.0 metadata from ritupriya.p176698@agentforce.com using the v64.0 SOAP API

✓ Preparing retrieve request 10ms
✓ Sending request to org 50ms
✓ Waiting for the org to respond 6.57s
✓ Done 0ms

Status: Succeeded
Elapsed Time: 7.04s
```

## **Phase 9: Reporting, Dashboards & Security Review**

Reports

- Monitor business activity, track orders, revenue, and complaints.

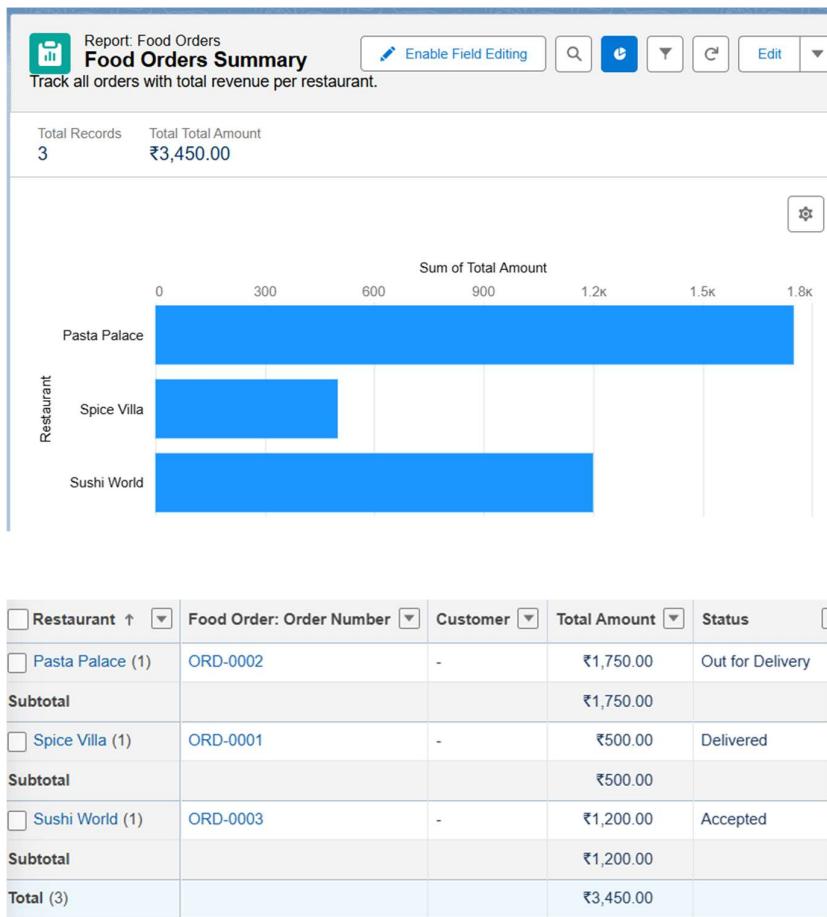
## Steps:

- **Reports** → New Report → Select object/report type.
  - Apply filters and choose columns.
  - Group rows, summarize numeric fields.
  - Add charts
  - Save to public folder.

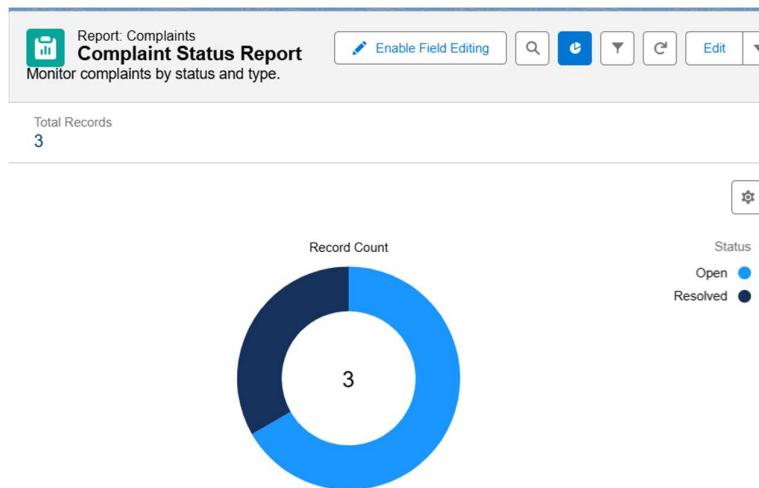
Reports						
Created by Me						
4 items						
REPORTS	Report Name	Description	Folder	Created By	Created On	Subscribed
Recent	Complaint Status Report	Monitor complaints by status and type.	My Folder	Pallepati Ritupriya	9/24/2025, 3:48 AM	
Created by Me	Food Orders Summary	Track all orders with total revenue per restaurant.	My Folder	Pallepati Ritupriya	9/24/2025, 1:19 AM	
Private Reports	Revenue by Cuisine	Show earnings per cuisine type (Indian, Italian, etc.).	My Folder	Pallepati Ritupriya	9/24/2025, 4:30 AM	
Public Reports	Top Menu Items	Show which menu items are most ordered.	My Folder	Pallepati Ritupriya	9/24/2025, 4:35 AM	
All Reports						

## 1. Food Orders Summary – Track all orders with total revenue per restaurant.

- Shows total revenue per restaurant; grouped by Restaurant Name, with bar chart.

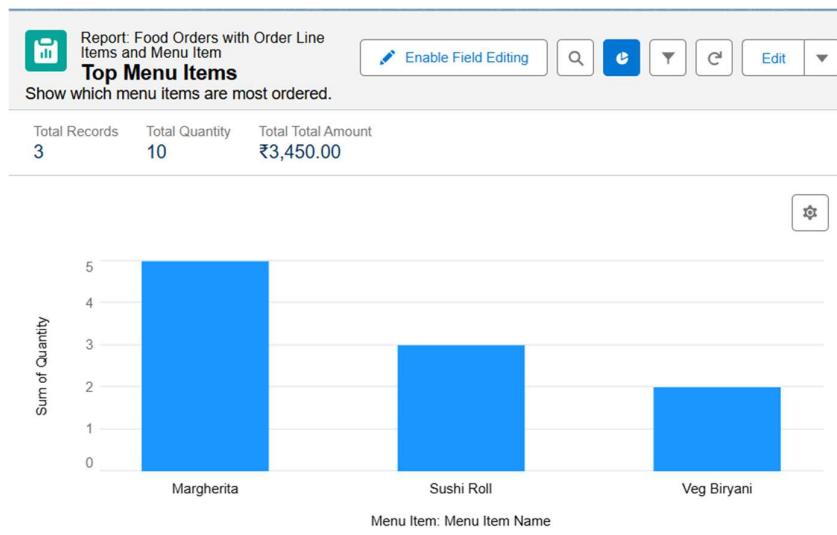


## 2. Complaint Status – Tracks complaints by status; grouped by Status, with pie chart.



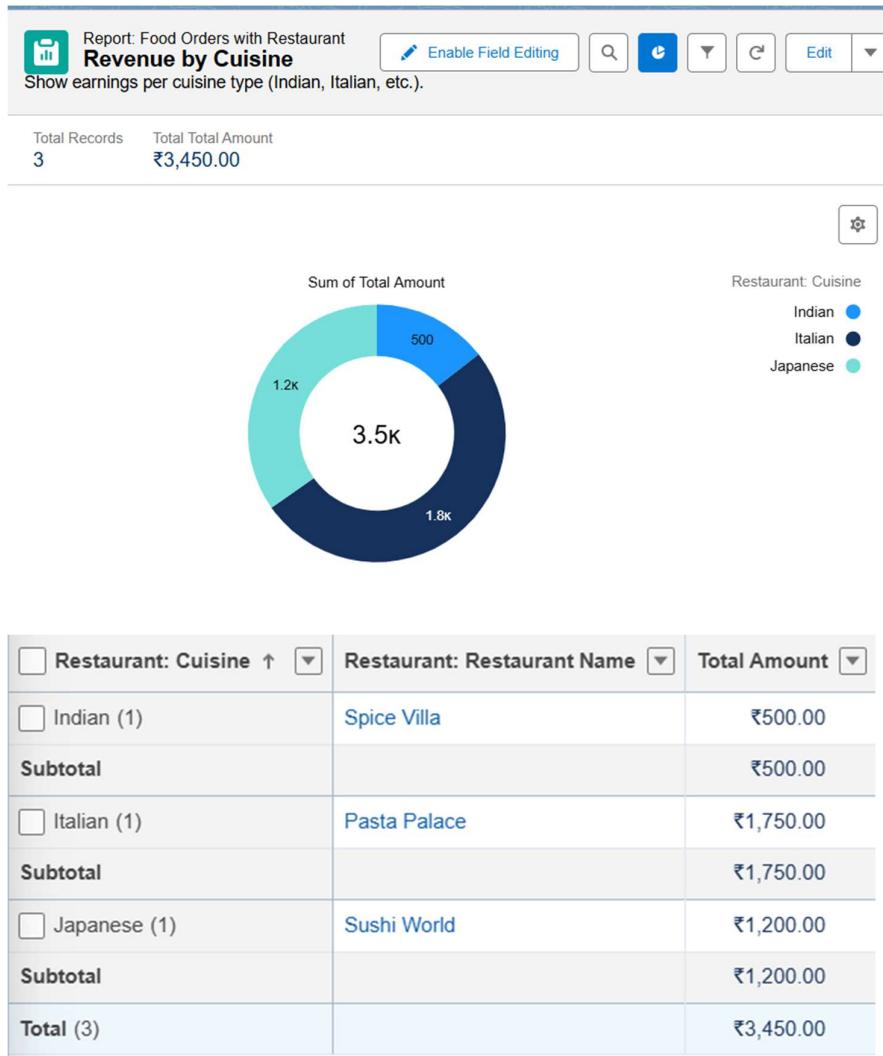
<input type="checkbox"/> Status ↑ ↓	Complaint: Complaint Number	Food Order	Complaint Type	Complaint: Created Date
<input type="checkbox"/> Open (2)	CMP-0001 CMP-0003	ORD-0001 ORD-0003	Late Delivery Quality Issue	9/24/2025 9/24/2025
<b>Subtotal</b>				
<input type="checkbox"/> Resolved (1)	CMP-0002	ORD-0002	Wrong Item	9/24/2025
<b>Subtotal</b>				
<b>Total (3)</b>				

3. **Top Menu Items** – Shows most ordered items; grouped by Menu Item, with horizontal bar chart.



<input type="checkbox"/> Menu Item: Menu Item Name ↑ ↓	Restaurant	Quantity	Total Amount
<input type="checkbox"/> Margherita (1)	Pasta Palace	5	₹1,750.00
<b>Subtotal</b>		5	₹1,750.00
<input type="checkbox"/> Sushi Roll (1)	Sushi World	3	₹1,200.00
<b>Subtotal</b>		3	₹1,200.00
<input type="checkbox"/> Veg Biryani (1)	Spice Villa	2	₹500.00
<b>Subtotal</b>		2	₹500.00
<b>Total (3)</b>		10	₹3,450.00

4. **Revenue by Cuisine** – Shows earnings per cuisine; grouped by Cuisine, with pie chart



## Dashboards

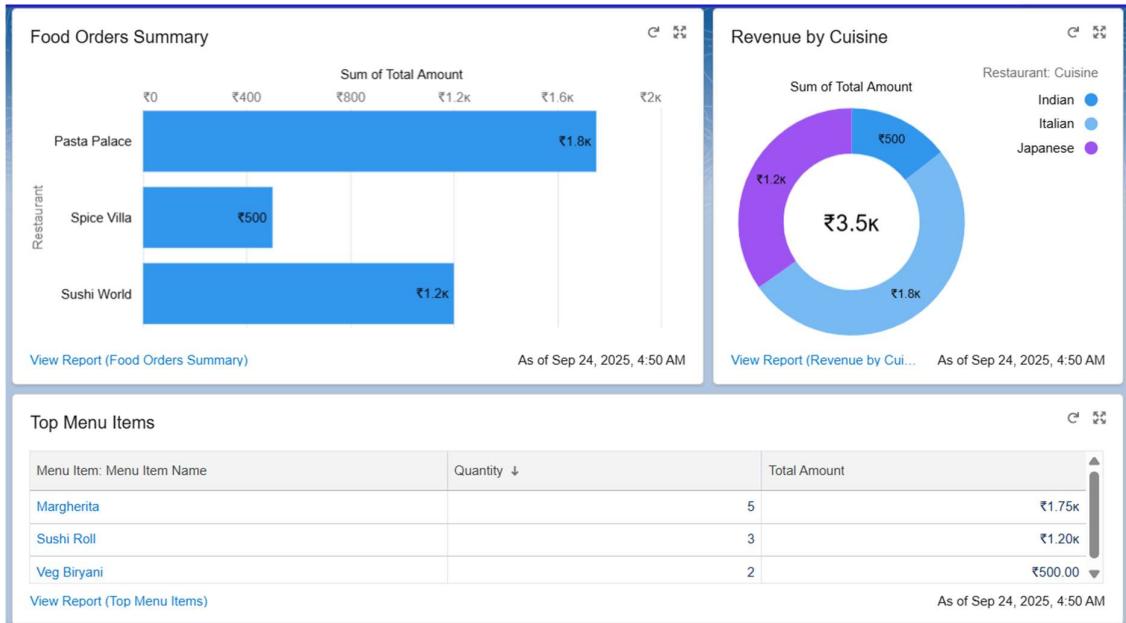
- Visualize key metrics for quick insights.

### Steps:

- **Dashboards → New Dashboard.**
- Add components (charts, tables) using saved reports.
- Configure filters and display options.
- Save & run in a public folder.

## Restaurant Business Overview

Provides a quick view of overall restaurant performance, including total orders, revenue, top menu items, and complaints status.



## Field-Level Security

- Setup → Object Manager → Food\_Orders\_\_c → Fields & Relationships → Select Customer\_\_c
- Click Set Field-Level Security → Uncheck Visible for Delivery Agent Profile → Save changes

Set Field-Level Security

Customer

Field Label	Customer	Save	Cancel
Data Type	Lookup(Contact)		

Delivery\_Agent\_Profile

## Login IP Ranges

- Setup → Users → Profiles → Delivery Agent Profile
- Scroll to Login IP Ranges section → Click New
- Enter:
  - Start IP Address** → e.g., 203.0.113.1 (your office network start)
  - End IP Address** → e.g., 203.0.113.255 (your office network end)
- Save changes

## Phase 10: Final Presentation & Demo Day

- In this final phase, I showcased my Salesforce project **Online Food Delivery CRM** in a live demo.
- The system simplifies food ordering and restaurant management by providing:
  - **Custom Lightning Application** – Centralized app for managing restaurants, orders, and complaints
  - **Food Order & Menu Management** – Easy creation and tracking of orders and menu items
  - **Role-Based Access** – Secure access for Restaurant Manager, Delivery Agent, and Executive roles
  - **Complaint Handling** – Tracks customer complaints and updates order status automatically
  - **Data Management** – Import, export, and maintain restaurant, menu, and order data efficiently using Salesforce tools like Data Import Wizard.
  - **Automated Notifications** – Email alerts for order confirmation and complaint resolution
  - **Reports for Analysis** – Detailed reports on orders, complaints, and top-selling menu items
  - **Dashboards** – Real-time insights through visual dashboards for restaurant performance and revenue.

The project demonstrates how Salesforce can streamline restaurant workflows, improve operational efficiency, and provide actionable business insights.

**Project PPT:** [https://docs.google.com/presentation/d/1xwLux-4\\_Jmjat2mKe13E3LiLrOidkAWq/edit?usp=sharing&ouid=113298212239433324406&rtpof=true&sd=true](https://docs.google.com/presentation/d/1xwLux-4_Jmjat2mKe13E3LiLrOidkAWq/edit?usp=sharing&ouid=113298212239433324406&rtpof=true&sd=true)

**Project Video:**

[https://drive.google.com/file/d/1cO3LnHkoLl6ZyOulSv6hFVrFy82F\\_2jI/view?usp=drive\\_link](https://drive.google.com/file/d/1cO3LnHkoLl6ZyOulSv6hFVrFy82F_2jI/view?usp=drive_link)

