FoodieConnect Salesforce CRM

Name : M.Lakshmi Narayana Varma

Email: varmamudhuluri@gmail.com

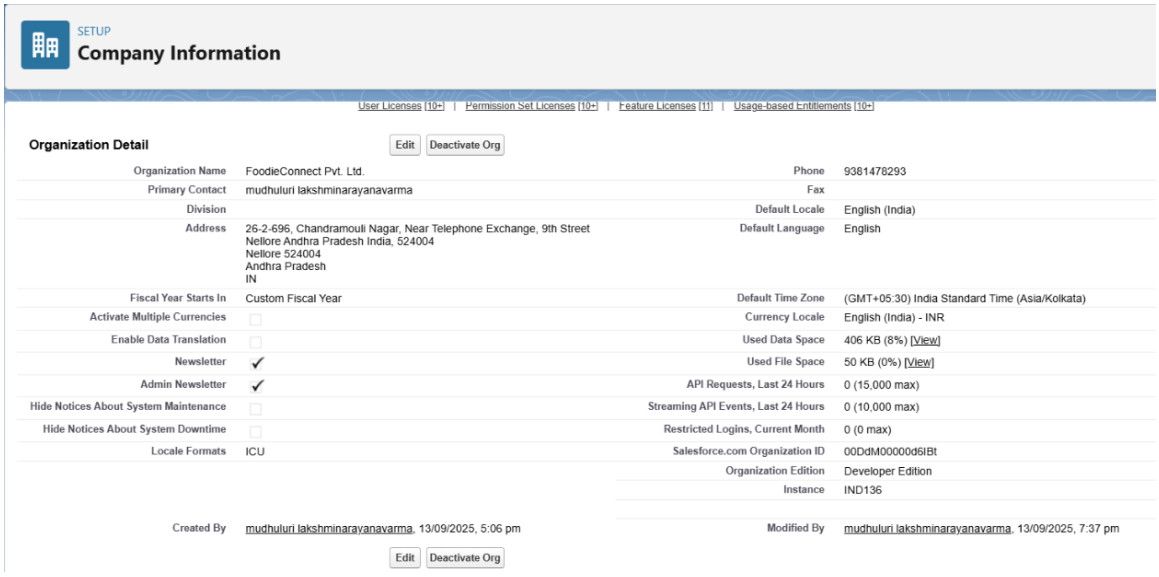
1. Introduction

FoodieConnect Pvt. Ltd. is building a Salesforce-powered CRM tailored for the food delivery industry. This phase covers organizational setup, user management, access control, deployment basics, data management, integration setup, and reporting. The goal is to ensure smooth operations, collaboration, and real-time insights for the business.

2. Company Profile Setup

The company information has been configured to reflect FoodieConnect's business identity and operating details.

Organization Name: FoodieConnect Pvt. Ltd.
Primary Contact: Mudhuluri Lakshminarayanavarma
Phone: 9381478293
Address: 26-2-696, Chandramouli Nagar, Near Telephone Exchange, 9th Street, Nellore, Andhra Pradesh, India – 524004
Default Locale: English (India)
Default Language: English
Time Zone: India Standard Time (GMT+05:30, Asia/Kolkata)
Currency Locale: INR (English - India)
Multiple Currencies: Not enabled



SETUP
Company Information

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

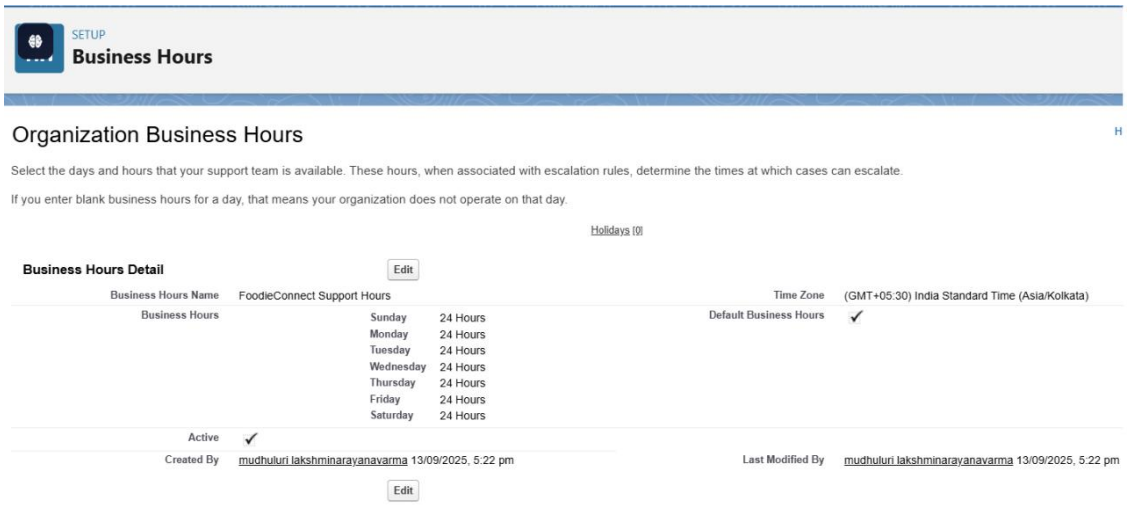
Organization Detail Edit | Deactivate Org

Organization Name	FoodieConnect Pvt. Ltd.	Phone	9381478293
Primary Contact	mudhuluri lakshminarayanavarma	Fax	
Division		Default Locale	English (India)
Address	26-2-696, Chandramouli Nagar, Near Telephone Exchange, 9th Street Nellore Andhra Pradesh India, 524004 Nellore 524004 Andhra Pradesh IN	Default Language	English
Fiscal Year Starts In	Custom Fiscal Year	Default Time Zone	(GMT+05:30) India Standard Time (Asia/Kolkata)
Activate Multiple Currencies	☐	Currency Locale	English (India) - INR
Enable Data Translation	☐	Used Data Space	406 KB (8%) [View]
Newsletter	✓	Used File Space	50 KB (0%) [View]
Admin Newsletter	✓	API Requests, Last 24 Hours	0 (15,000 max)
Hide Notices About System Maintenance	☐	Streaming API Events, Last 24 Hours	0 (10,000 max)
Hide Notices About System Downtime	☐	Restricted Logins, Current Month	0 (0 max)
Locale Formats	ICU	Salesforce.com Organization ID	00DdM00000d6IBt
		Organization Edition	Developer Edition
		Instance	IND136
Created By	mudhuluri lakshminarayanavarma, 13/09/2025, 5:06 pm	Modified By	mudhuluri lakshminarayanavarma, 13/09/2025, 7:37 pm

Edit | Deactivate Org

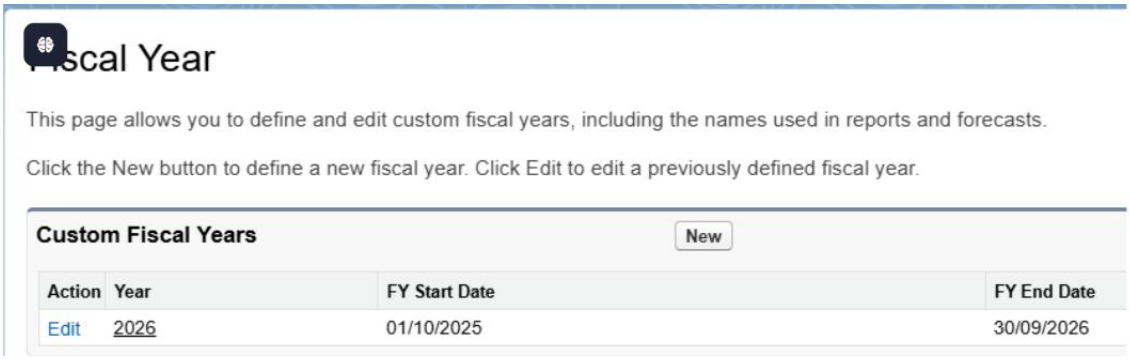
3. Business Hours & Holidays

Business Hours Name: FoodieConnect Support Hours
Schedule: Sunday to Saturday – 24 Hours
Default Business Hours: Enabled
Active: Yes
Holidays Configured: None



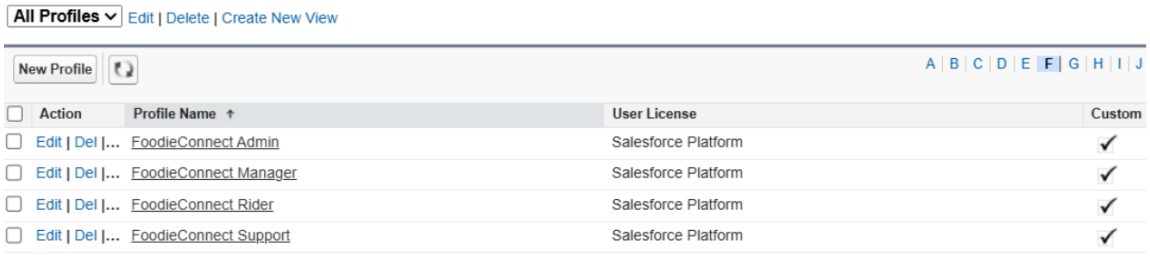
4. Fiscal Year Settings

Fiscal Year 2026:
Start Date: 01-Oct-2025
End Date: 30-Sep-2026



5. Profiles

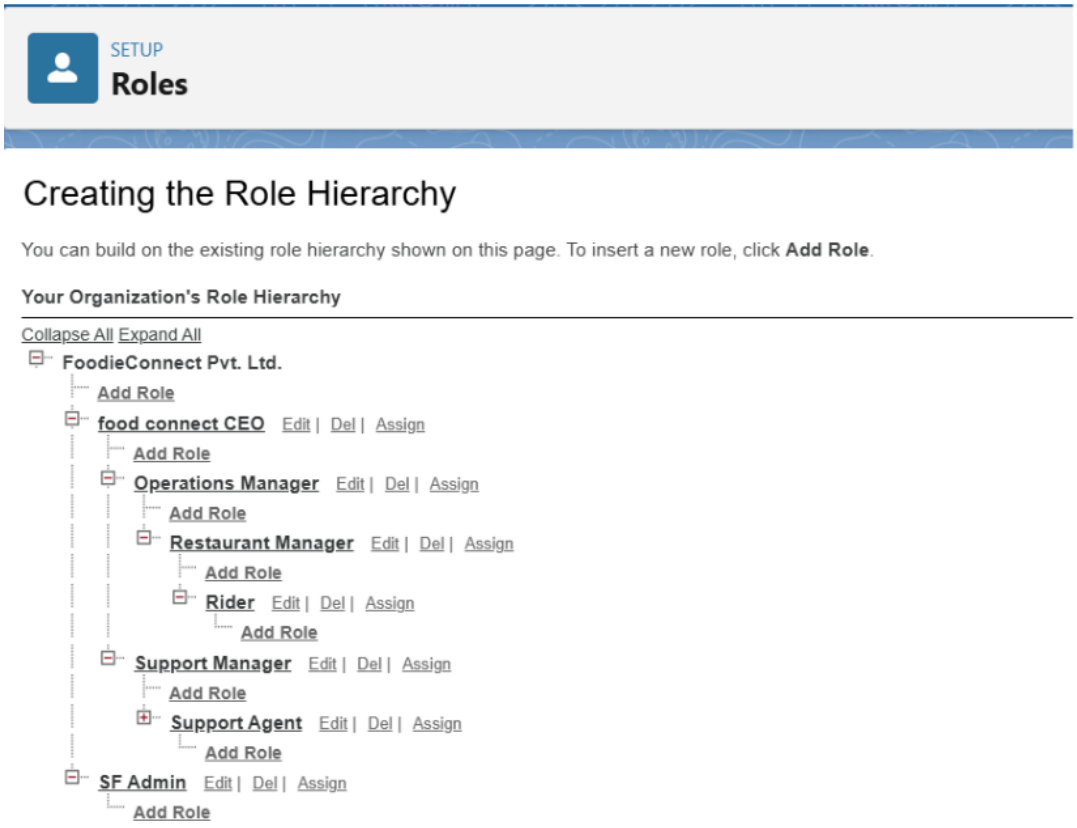
Profiles created under the Salesforce Platform license:
- FoodieConnect Admin – for system administrators with high-level access.
- FoodieConnect Manager – for operational and business managers.
- FoodieConnect Rider – for delivery staff with limited access to orders and tasks.
- FoodieConnect Support – for customer support representatives.



6. Roles

Role Hierarchy:

FoodieConnect Pvt. Ltd.
- CEO
  - Operations Manager
    - Restaurant Manager
    - Rider
  - Support Manager
    - Support Agent
  - SF Admin

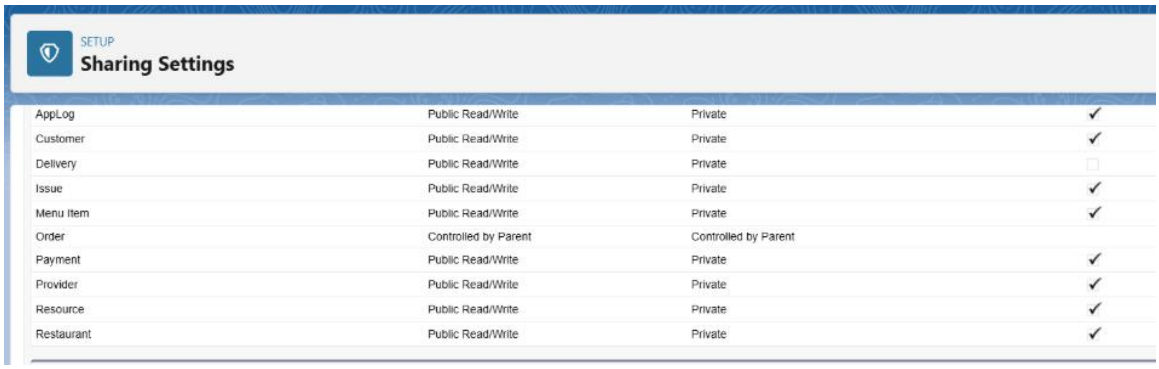


7. Organization-Wide Defaults (OWD)

Customer: Public Read/Write
Delivery: Public Read/Write
Issue: Public Read/Write
Menu Item: Public Read/Write
Order: Controlled by Parent
Payment: Public Read/Write
Provider: Public Read/Write
Resource: Public Read/Write
Restaurant: Public Read/Write

8.Sharing Rules

- Orders Sharing Rule: Managers and Admins have full access to Orders owned by Riders.
- Payments Sharing Rule: Managers have full access to all Payments.
- Customer Sharing Rule: Support Agents have read/write access to Customers.
- Restaurant Sharing Rule: Operations Managers and Admins have full access to Restaurants.

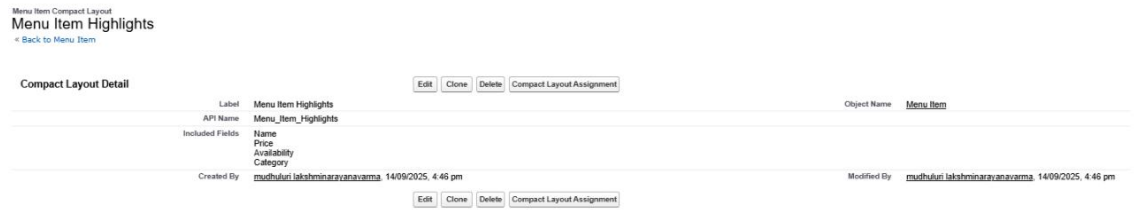


9.Standard & Custom Objects

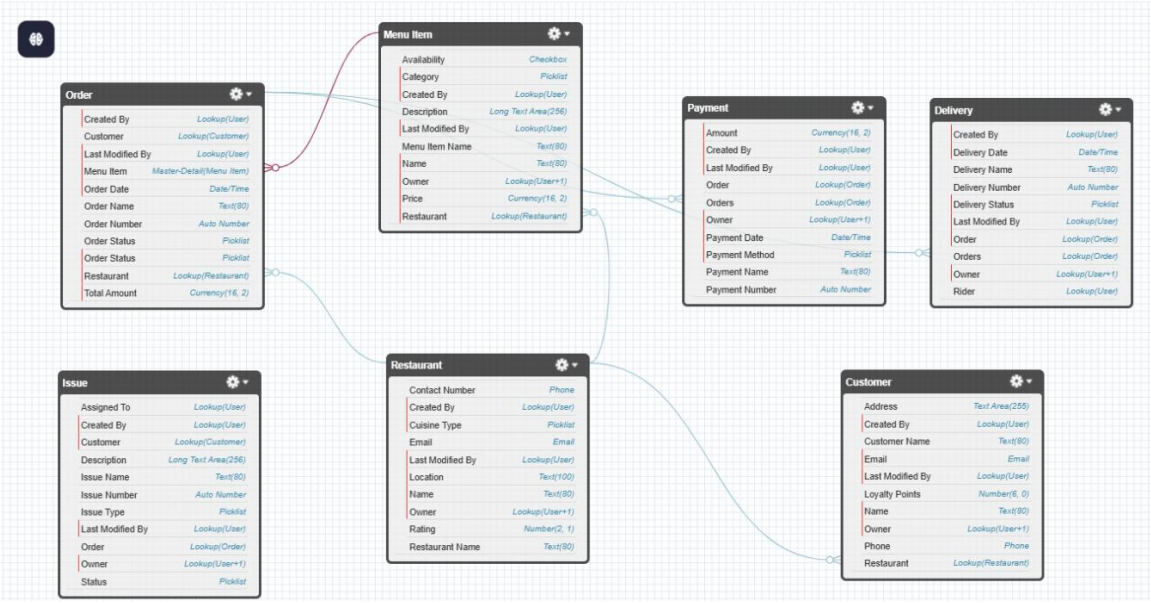
- **Restaurant:** Stores partner restaurant details.
- **Menu Item:** Lists food items offered by restaurants.
- **Customer:** Tracks customer details and loyalty points.
- **Order:** Manages all orders placed.
- **Delivery:** Monitors delivery assignments and status.
- **Payment:** Records payment information.
- **Issue (Support Case):** Logs customer complaints or queries.

10. Compact Layouts

Restaurant Highlights



11. Schema Builder & Relationships



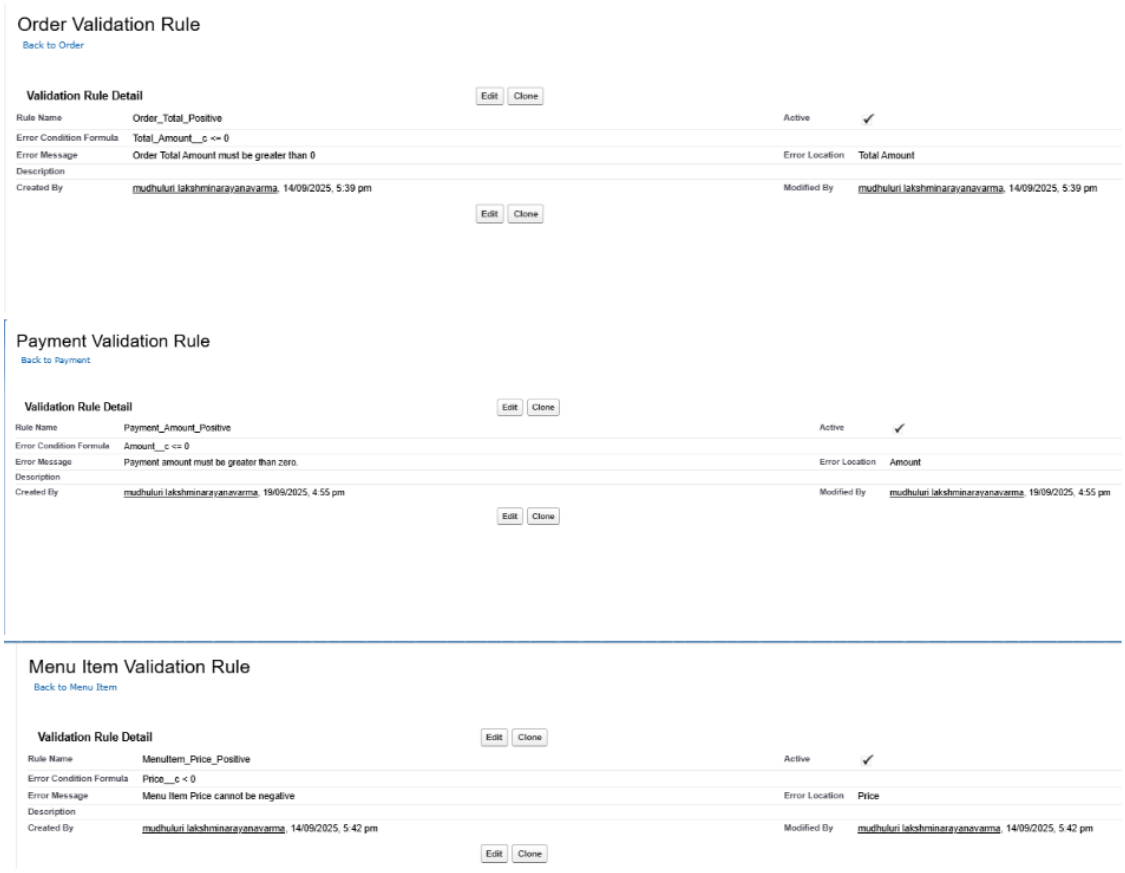
- **Lookup Relationships:** Connect related records, e.g., Menu Items → Restaurant.
- **Master-Detail Relationships:** Maintain tighter control, e.g., Orders → Menu Items.

12. Validation Rules

Purpose: Ensure data integrity by preventing invalid records from being saved.

Implemented Rules:

- Order_Total_Positive
    - Object: Order
    - Formula: Total_Amount__c <= 0
    - Error Message: "Order Total Amount must be greater than 0."
    - Status: Active ✅

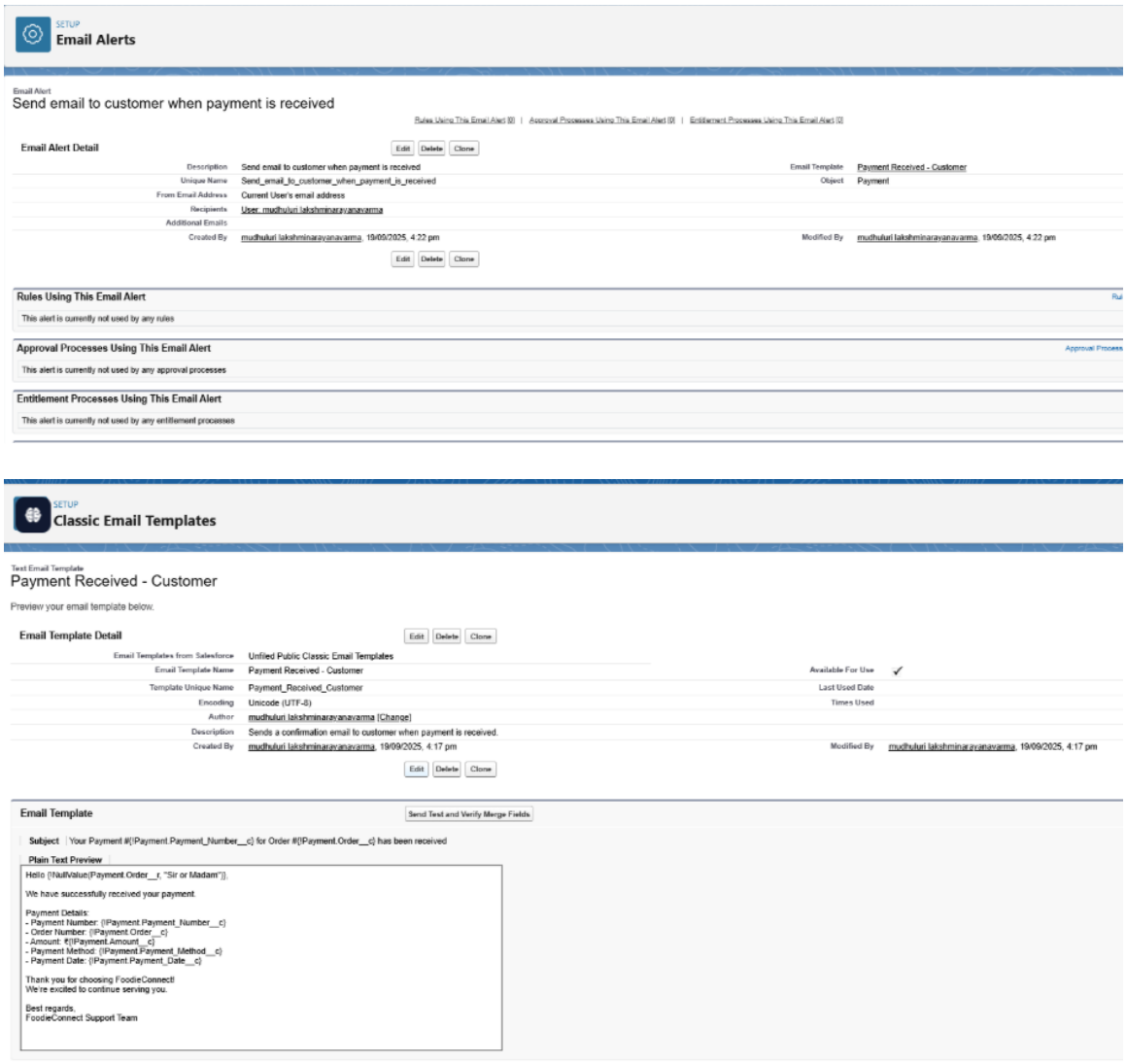


13. Email Alerts

Purpose: Automatically notify customers when key actions occur.

Implemented Alerts:

- Payment Received Email Alert
    - Object: Payment
    - Email Template: Payment Received - Customer
    - Recipients: Customer email (configured for testing to owner email

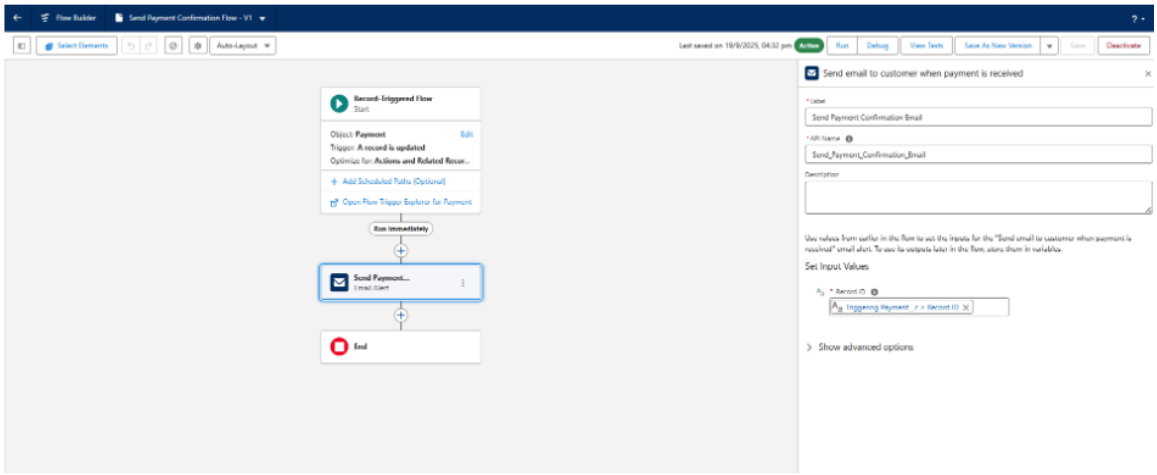


14.Flow Builder (Record-Triggered Flows)

Purpose: Automate actions based on record changes.

Implemented Flows:

- **Payment Confirmation Flow**
    - Triggered when a **Payment** record is created.
    - Automatically sends the **Payment Received Email Alert**.



15. Classes & Objects

RestaurantService

Handles restaurant-related operations such as fetching active restaurants and their menu items based on location.

Key Methods:

- getActiveRestaurantsByLocation(String location) → Fetches active restaurants and their menu items for a specific location
- updateRestaurantRating(Set<Id> restaurantIds) → Updates restaurant ratings based on published reviews

```
public with sharing class RestaurantService {

    // Method to get restaurants by location (active field removed)
    public static List<Restaurant__c> getRestaurantsByLocation(String location) {
        return [
            SELECT Id, Name, Name__c, Contact_Number__c, Email__c, Rating__c, Cuisine_Type__c, Location__c,
                   (SELECT Id, Name, Price__c, Description__c
                    FROM Menu_Items__r
                    ORDER BY Name)
            FROM Restaurant__c
            WHERE Location__c = :location
            ORDER BY Rating__c DESC NULLS LAST
            LIMIT 50
        ];
    }


}
```

OrderHandle

Manages order lifecycle, including auto-generating order numbers, validating order status, creating delivery records, and updating customer loyalty points.
Key Methods:

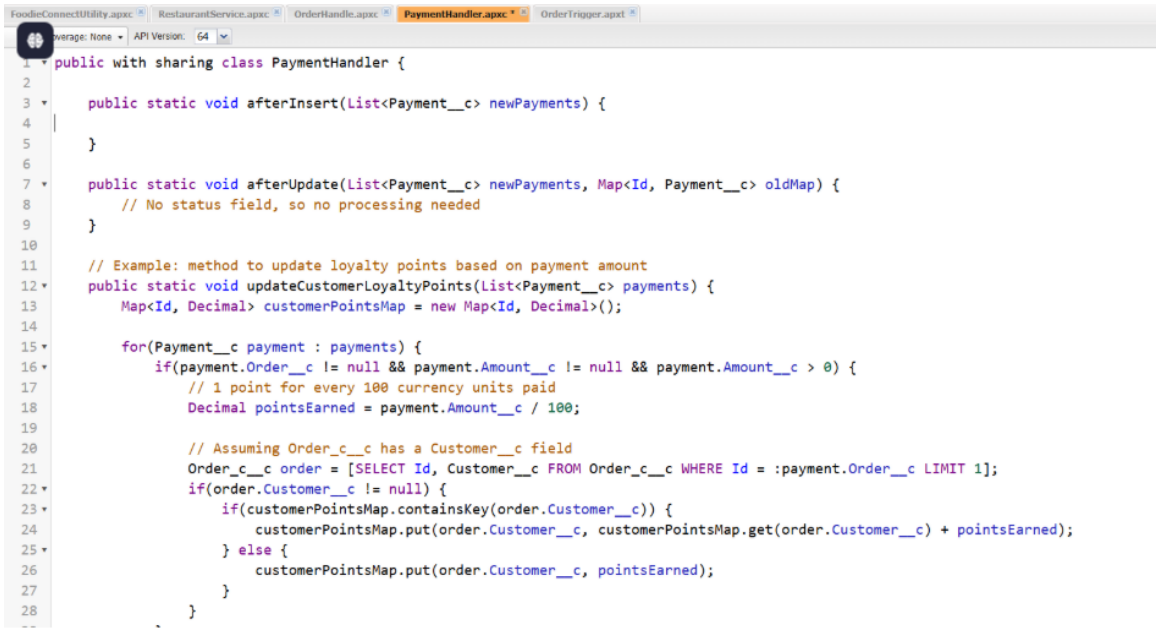
- beforeInsert(List<Order__c> newOrders) → Sets default status and generates order numbers
- afterInsert(List<Order__c> newOrders) → Creates delivery records and updates loyalty points
- beforeUpdate(List<Order__c> newOrders, Map<Id, Order__c> oldMap) → Validates order status changes
- afterUpdate(List<Order__c> newOrders, Map<Id, Order__c> oldMap) → Updates related deliveries


```apex
public with sharing class OrderHandle {

    public static void beforeInsert(List<Order_c__c> newOrders) {
        // Set default status if not provided
        for(Order_c__c order : newOrders) {
            if(order.Order_Status__c == null) {
                order.Order_Status__c = 'New';
            }
        }
    }

    public static void afterInsert(List<Order_c__c> newOrders) {
        // Update customer loyalty points
        updateCustomerLoyaltyPoints(newOrders);
    }

    public static void beforeUpdate(List<Order_c__c> newOrders, Map<Id, Order_c__c> oldMap) {
        validateOrderStatusChange(newOrders, oldMap);
    }

    public static void afterUpdate(List<Order_c__c> newOrders, Map<Id, Order_c__c> oldMap) {
        // No related deliveries to update in current schema
    }

    private static void validateOrderStatusChange(List<Order_c__c> newOrders, Map<Id, Order_c__c> oldMap) {
        for(Order_c__c newOrder : newOrders) {
            Order_c__c oldOrder = oldMap.get(newOrder.Id);

```

PaymentHandler

Manages payments and integrates payment data with orders. Updates order status after payment creation.
Key Methods:

- afterInsert(List<Payment__c> newPayments) → Updates related orders after payment creation
- afterUpdate(List<Payment__c> newPayments, Map<Id, Payment__c> oldMap) → Handles updates in payment information



```apex
public with sharing class PaymentHandler {

    public static void afterInsert(List<Payment__c> newPayments) {

    }

    public static void afterUpdate(List<Payment__c> newPayments, Map<Id, Payment__c> oldMap) {
        // No status field, so no processing needed
    }

    // Example: method to update loyalty points based on payment amount
    public static void updateCustomerLoyaltyPoints(List<Payment__c> payments) {
        Map<Id, Decimal> customerPointsMap = new Map<Id, Decimal>();

        for(Payment__c payment : payments) {
            if(payment.Order__c != null && payment.Amount__c != null && payment.Amount__c > 0) {
                // 1 point for every 100 currency units paid
                Decimal pointsEarned = payment.Amount__c / 100;

                // Assuming Order_c__c has a Customer__c field
                Order_c__c order = [SELECT Id, Customer__c FROM Order_c__c WHERE Id = :payment.Order__c LIMIT 1];
                if(order.Customer__c != null) {
                    if(customerPointsMap.containsKey(order.Customer__c)) {
                        customerPointsMap.put(order.Customer__c, customerPointsMap.get(order.Customer__c) + pointsEarned);
                    } else {
                        customerPointsMap.put(order.Customer__c, pointsEarned);
                    }
                }
```

Apex Triggers

OrderTrigger

Implements the trigger handler pattern for the Order__c object.
Trigger Events:

- Before Insert → Auto-generates order numbers and sets default status
- After Insert → Creates delivery records and updates customer loyalty points
- Before Update → Validates order status changes to prevent invalid transitions
- After Update → Updates delivery statuses based on order status changes

FoodieConnectUtility.apxc | RestaurantService.apxc | OrderHandle.apxc | PaymentHandler.apxc * | OrderTrigger.apxt

Coverage: None | API Version: 64

```
trigger OrderTrigger on Order_c__c (before insert, before update, after insert, after update) {

    if(Trigger.isBefore) {
        if(Trigger.isInsert) {
            OrderHandle.beforeInsert(Trigger.new);
        }
        if(Trigger.isUpdate) {
            OrderHandle.beforeUpdate(Trigger.new, Trigger.oldMap);
        }
    }

    if(Trigger.isAfter) {
        if(Trigger.isInsert) {
            OrderHandle.afterInsert(Trigger.new);
        }
        if(Trigger.isUpdate) {
            OrderHandle.afterUpdate(Trigger.new, Trigger.oldMap);
        }
    }
}
```

SOQL Queries

Used SOQL to fetch orders, customers, and restaurant details.
Example Queries:
SELECT Id, Customer__r.Name, Total_Amount__c FROM Order__c WHERE Order_Status__c = 'Confirmed'

SELECT Id, Order_Number__c, Total_Amount__c, Customer__c FROM Order__c LIMIT 10

```
SELECT Id, Order_Number__c, Total_Amount__c, Customer__c FROM Order_c__c LIMIT 10
```

Query Results - Total Rows: 1

Id	Order_Number__c	Total_Amount__c	Customer__c
a06dM00000F7JQHQA3	ORD-0001	250	0o6dM0000000IXQAI

History

Executed

SELECT Id, Customer__r.Name, Total_Amount__c FROM Order_c__c WHERE Order...

SELECT Id, Order_Number__c, Total_Amount__c, Customer__c FROM Order_c__c ...

OrderHandlerTest

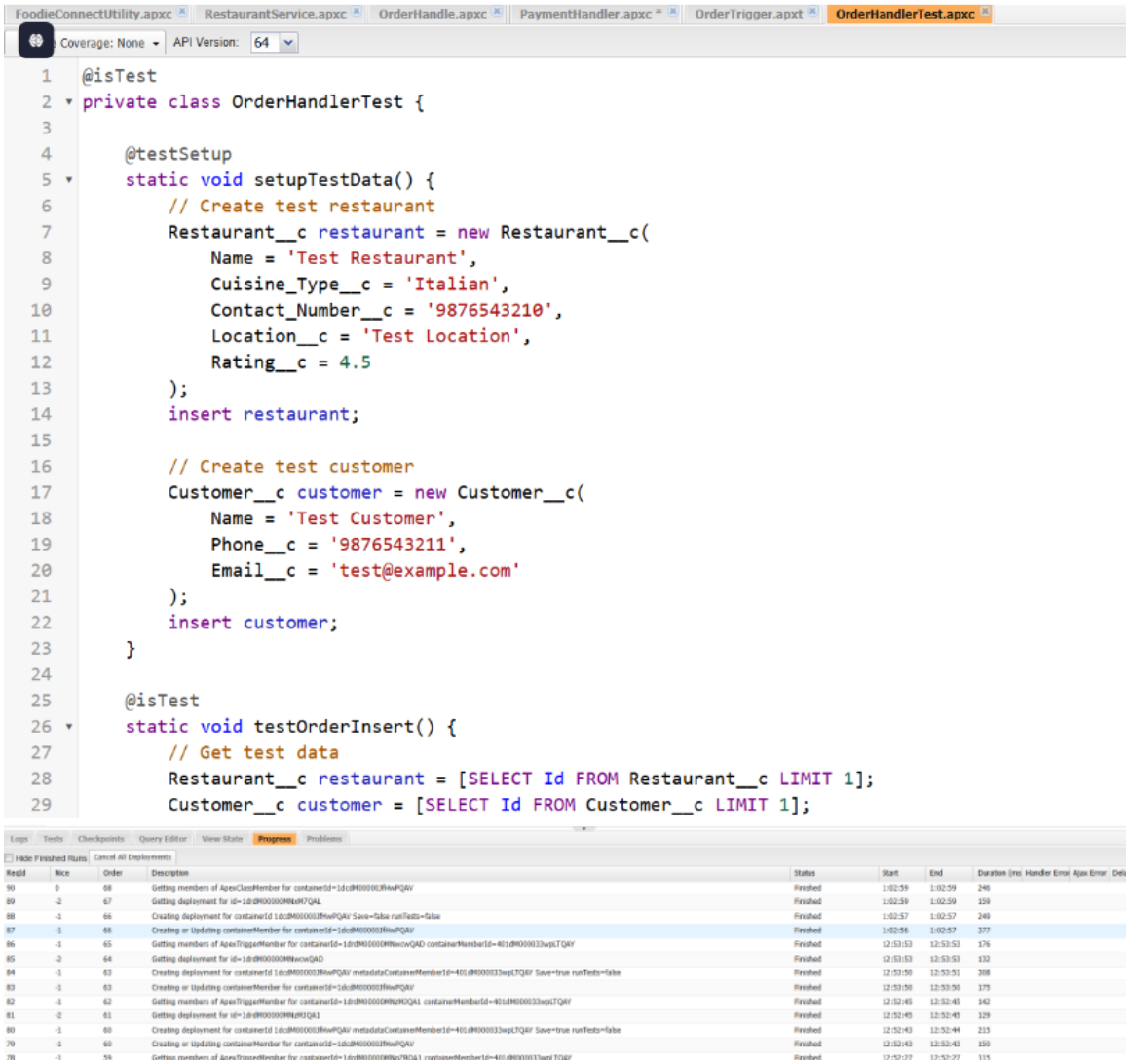
Covers unit testing for order insertion, delivery creation, and status validation.
Test Cases:

- Verify auto-generated order numbers are unique and sequential

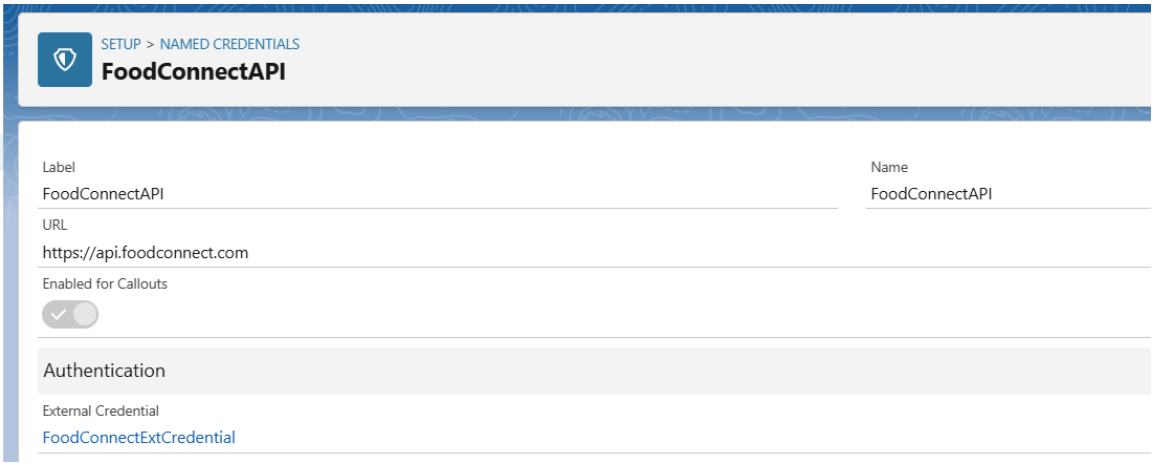
- Confirm delivery records are created for orders with 'Confirmed' status
- Prevent status regression (e.g., changing 'Delivered' orders back to 'Confirmed')
- Validate loyalty points are correctly calculated and assigned
- Test bulk order processing for performance

Test Coverage Features:

- @isTest annotation for test classes and methods
- Test.startTest() and Test.stopTest() for governor limit reset
- System.assert() methods for validation
- Test data factory methods for reusable test data creation

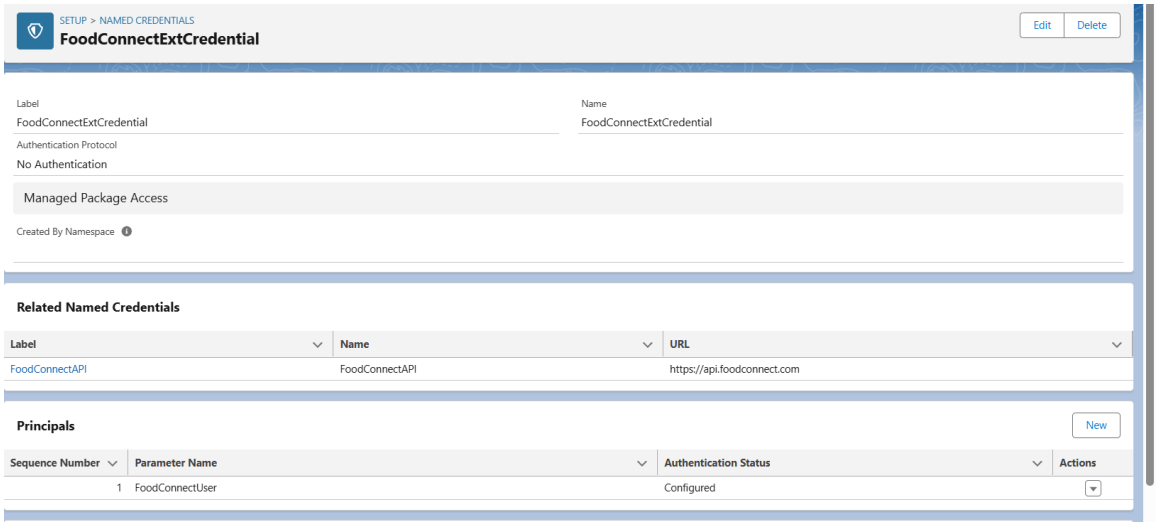


16.Named Credentials



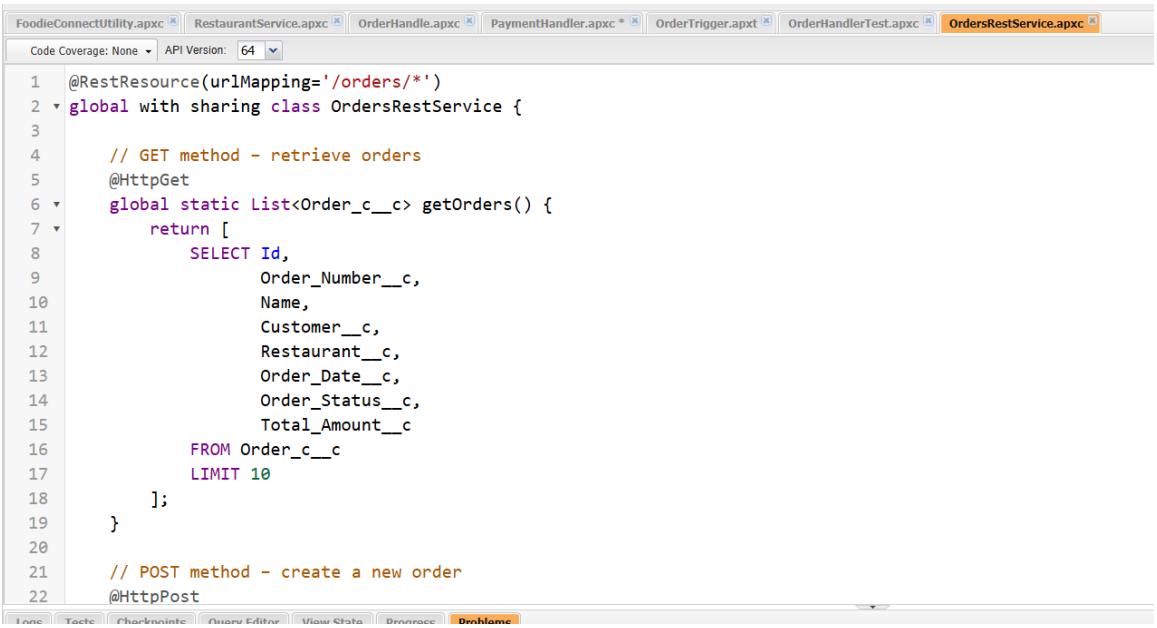
- **Named Credential Created:** FoodConnectAPI
- **URL:** https://api.foodconnect.com
- **Authentication:** Custom Header (Authorization: Bearer DUMMY_API_KEY)
- **Purpose:** Simplifies callouts to external APIs, securely stores credentials, and handles authentication automatically.
- **Status:** Completed
-

17. External Credentials



- **Status:** Configured
- **Purpose:** Enhanced security for external system authentication
- **Notes:** Used in conjunction with Named Credentials for advanced authentication scenarios

18. Web Services (REST/SOAP)



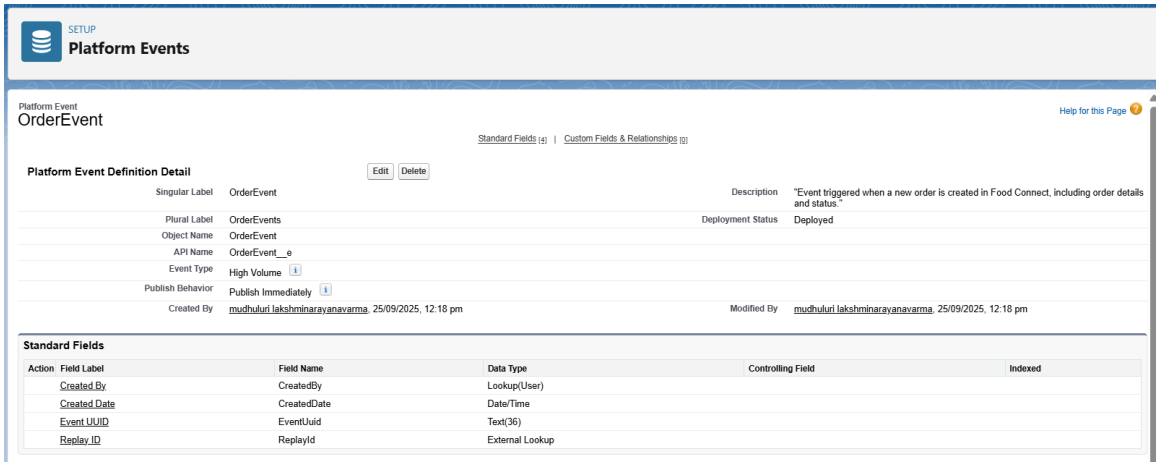
- **SOAP Web Service:** OrdersSoapService
  - getOrders() → Retrieves orders
  - createOrder(customerId, restaurantId, totalAmount) → Creates a new order
- **Status:** Implemented and functional in Apex
- **Purpose:** Enables external or internal systems to interact with Salesforce order data via SOAP.

19. Callouts

- **Status:** Configured using Named Credential
- **Purpose:** Apex callouts can access external API endpoints (https://api.foodconnect.com) without needing separate Remote Site Settings.
- **Example Usage:** HttpRequest req = new HttpRequest(); req.setEndpoint('callout:FoodConnectAPI/orders'); req.setMethod('GET');

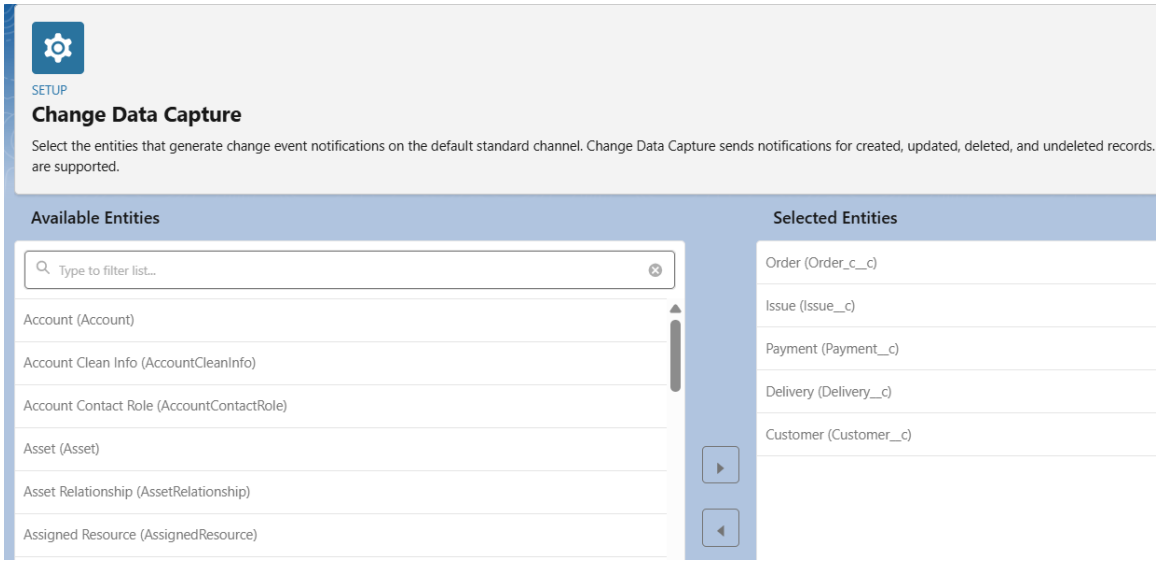
Http http = new Http(); HttpResponse res = http.send(req); System.debug(res.getBody());

20. Platform Events



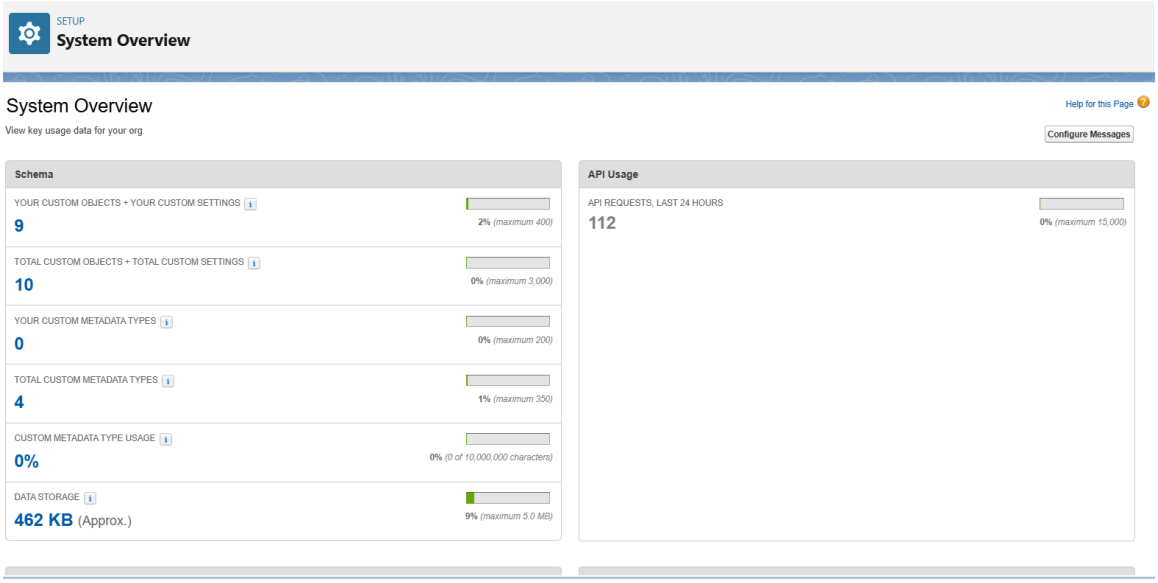
- **Event Created:** OrderEvent__e
- **Type:** High Volume
- **Publish Behavior:** Publish Immediately
- **Status:** Implemented
- **Purpose:** Used to notify external systems or trigger processes in real-time when orders are created or updated.

21. Change Data Capture



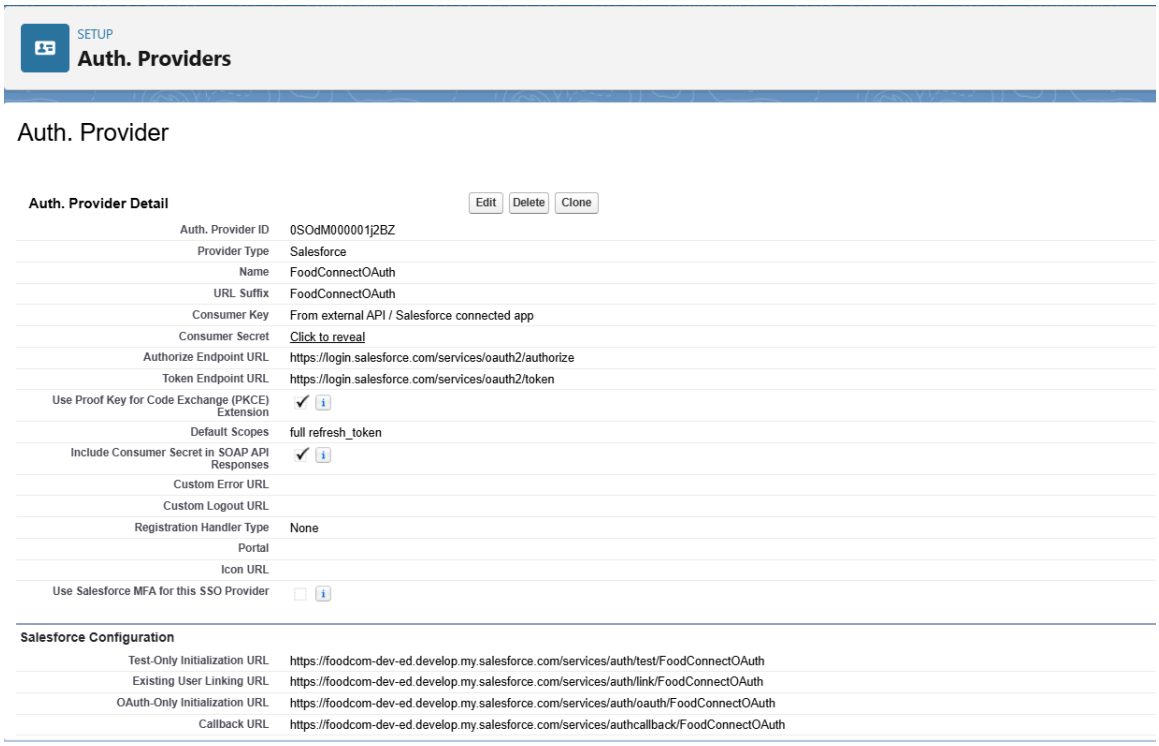
- **Objects Enabled:** Order__c
- **Purpose:** Automatically tracks Create, Update, Delete, Undelete operations and publishes events.
- **Status:** Implemented
- **Note:** Used for real-time integration with external systems.

22. API Limits



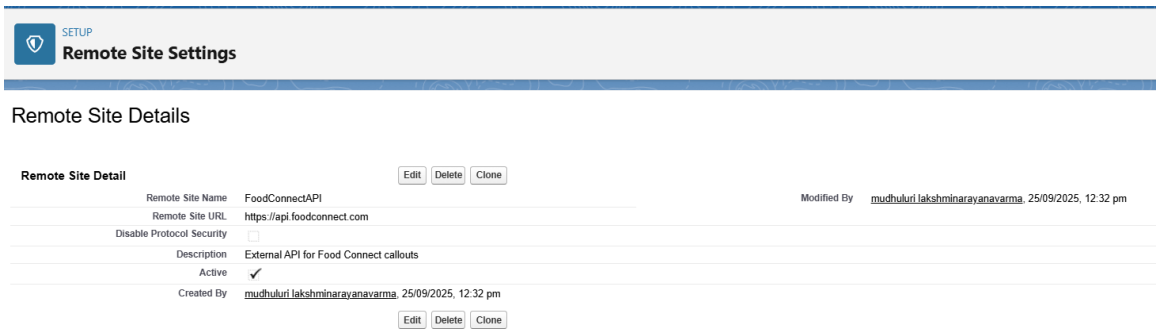
- **Current Usage:** 112 requests in last 24 hours (0% of 15,000 daily limit)
- **Status:** Checked
- **Notes:** Project is within safe API usage limits; monitoring only.

23. OAuth & Authentication



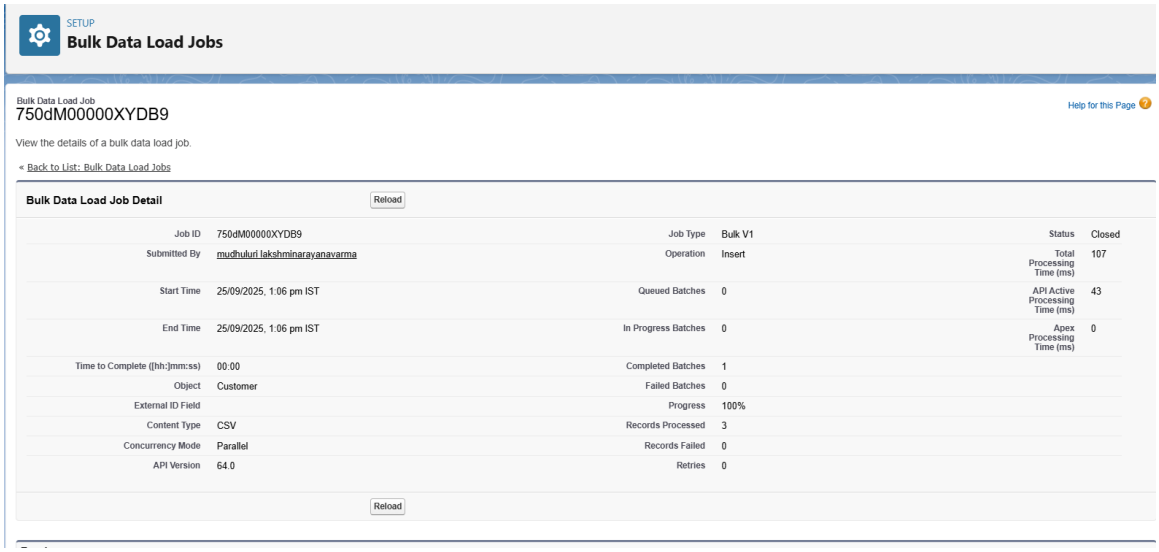
- **Auth Provider Created:** FoodConnectOAuth (Salesforce)
- **Named Credential:** Configured for OAuth 2.0 authentication
- **Status:** Implemented
- **Notes:** OAuth 2.0 authentication implemented for secure API access.

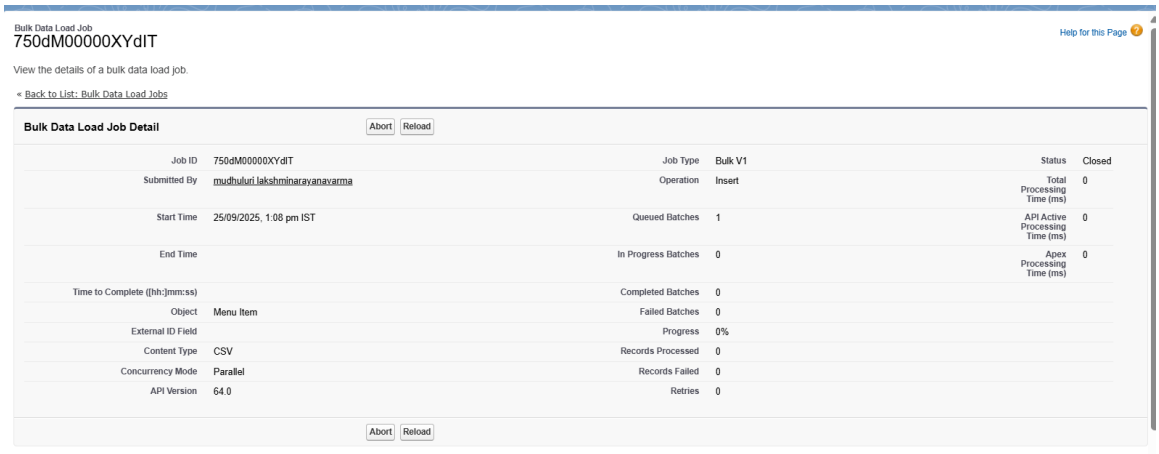
23. Remote Site Settings



- **Status:** Configured
- **Notes:** Additional remote site settings configured for backup callout endpoints and external service integrations.

24.Data Import Wizard





Purpose:

The Data Import Wizard is a Salesforce tool designed to simplify the process of importing data from external sources like CSV files into Salesforce standard and custom objects. It is particularly helpful for admins and users needing to upload data without scripting or programming knowledge. The wizard supports objects such as Customers, Orders, Payments, Restaurants, and more.

Detailed Steps Implemented:

- **Accessing the Wizard:** The process begins by navigating to Salesforce Setup and typing "Data Import Wizard" in the Quick Find box. Selecting the Data Import Wizard from the results launches the import interface.
- **Object Selection:** Users choose the object they want to import data into. For FoodieConnect, both standard and custom objects such as Customers were selected.

- **File Upload:** A CSV file containing the data (e.g., customer details) is uploaded via the wizard.
- **Field Mapping:** The wizard attempts automatic mapping of CSV file headers to Salesforce object fields. Example mappings for Customer data include:
    - CSV First_Name → Salesforce field First Name
    - CSV Last_Name → Salesforce field Last Name
    - CSV Email → Salesforce field Email
    - CSV Phone → Salesforce field Phone
    - CSV Address → Salesforce field Address
    - CSV Loyalty_Points → Salesforce field Loyalty Points
    - Any unmapped fields can be manually corrected before proceeding.
- **Validation & Import:** The wizard verifies the mapping, allowing corrections before starting the import process.
- **Import Success:** Successfully imported records are now available in the Salesforce CRM, ensuring data integrity and completeness.

Outcome:
The import process was smooth and accurate, resulting in a fully populated Customer database in Salesforce that reflects the source CSV data.

---

25. Data Loader

Purpose:
The Data Loader is a more powerful Salesforce client application ideal for handling large-scale data operations. It supports bulk data insertions, updates, upserts, and exports – especially useful for objects and volumes not supported by the Data Import Wizard.

Detailed Steps Implemented:

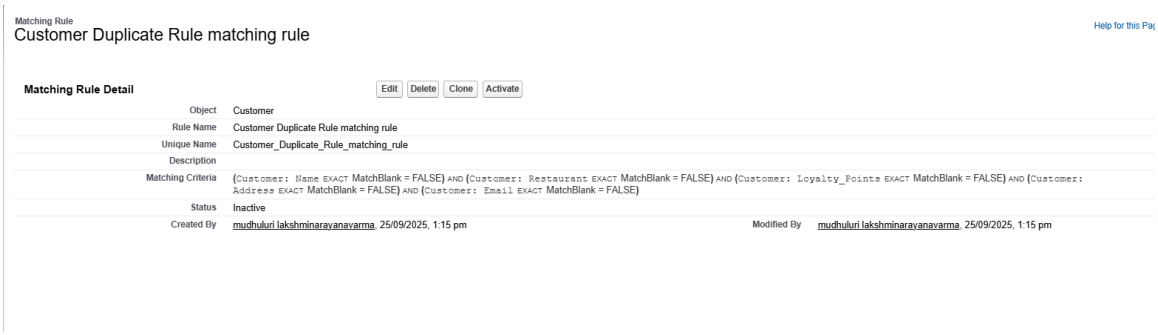
- **Installation:** The Data Loader application was downloaded and installed locally.
- **Connection Setup:** Configured connection credentials including Salesforce username, password, and security token to authenticate API access.
- **Bulk Operations:** Used to import large datasets for Orders and Payments which exceeded the Data Import Wizard's practical limits.

- **Data Integrity:** Care was taken to maintain proper data relationships (such as lookup and master-detail relationships) to ensure data consistency across related objects.
- **Error Handling:** The Data Loader's detailed error reports helped quickly identify and fix any issues during upload.

Outcome:
The Data Loader enabled efficient handling of large datasets, significantly supporting testing and validating CRM workflows from order placement to payment and delivery.

---

26. Duplicate Rules



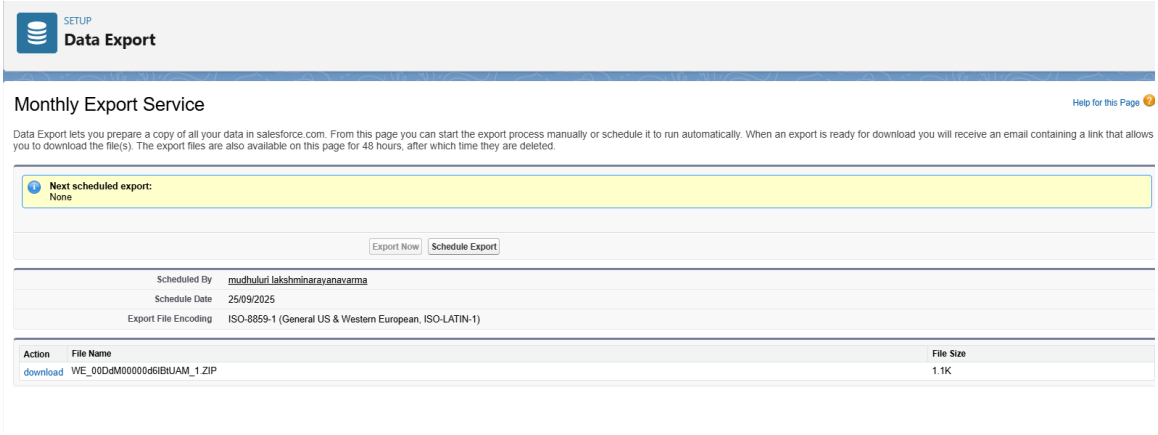
Purpose:
To maintain a clean and reliable customer database, Duplicate Rules are essential. They help detect, prevent, and manage duplicate records, reducing inconsistencies and improving user trust in the data.

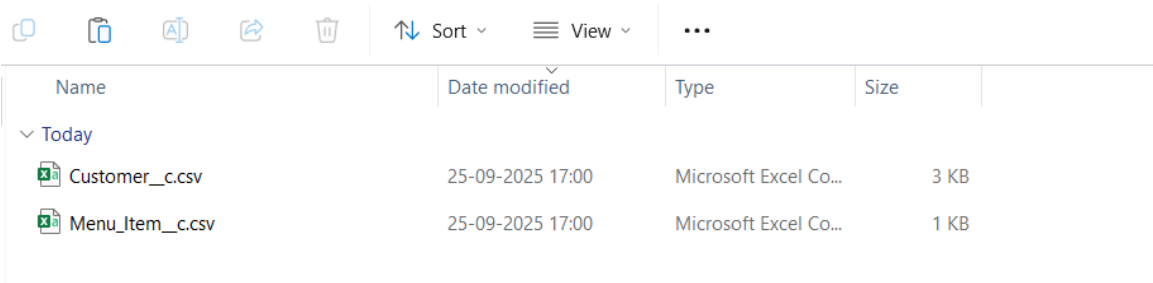
Detailed Steps Implemented:

- **Setup Navigation:** Accessed through Setup under Duplicate Management.
- **Rule Creation:** A Duplicate Rule for the Customer object was created and configured.
- **Matching Logic:** The rule matches records based on critical criteria like Customer Name, Restaurant affiliation, Loyalty Points, Address, and Email.
- **User Experience:** When duplicates are detected during record creation or edits, users are alerted with options for resolution.
- **Reporting:** Duplicate detection is logged for administrative oversight.

Outcome:
Duplicate entries are minimized, which enhances CRM data quality and operational efficiency.

27. Data Export & Backup





Purpose:
Regular data backups are critical for disaster recovery, compliance, and offline access. Salesforce's Data Export feature automates this process.

Detailed Steps Implemented:

- **Access:** Located in Setup under Data Export.
- **Scheduling:** Configured manual and recurring export schedules to cover all relevant Salesforce objects including Customers, Orders, Payments, etc.
- **Export Format:** Data was exported as CSV files packaged inside ZIP archives, using the ISO-8859-1 encoding to accommodate special characters.
- **Download & Storage:** Exported ZIP files downloaded and safely stored outside Salesforce for redundancy.
- **Restore Readiness:** Data is ready for import back into Salesforce or migration to other environments if needed.

Outcome:
A reliable backup strategy is in place ensuring data safety and availability.

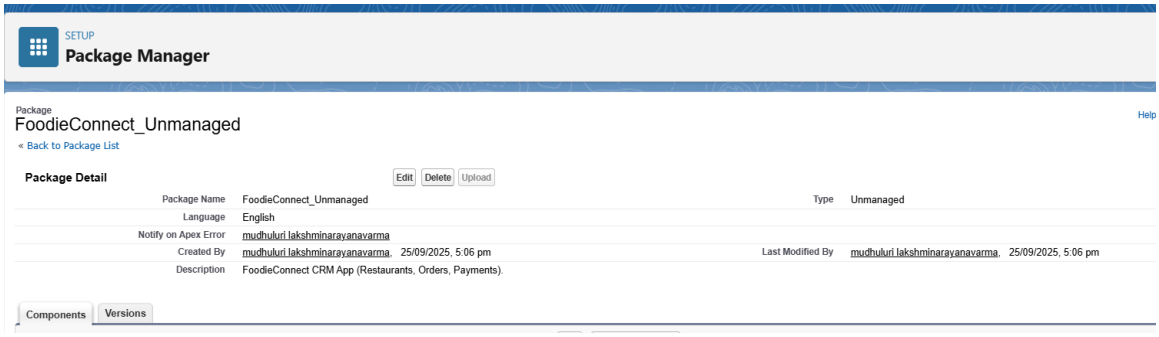
28. Change Sets

Purpose:
Change Sets simplify deployment by transferring metadata changes (like new fields or triggers) between Salesforce orgs, typically from sandbox to production environments.

Implementation Status:
Although Change Sets were noted, they were not visible or actively used in this FoodieConnect org because all work was done within a single environment, making Change Sets optional.

Outcome:
No deployment disruptions from Change Sets; future deployments can consider using Change Sets if multi-org rollout is introduced.

29. Unmanaged vs Managed Packages



Action	Component Name	Parent Object	Type	Included By
	Address	Customer	Custom Field	Customer
	All	Payment	List View	Payment
	All	Delivery	List View	Delivery
	All	Restaurant	List View	Restaurant
	All	Customer	List View	Customer
	All	Menu Item	List View	Menu Item
	All	Issue	List View	Issue
	All	Order	List View	Order
	Amount	Payment	Custom Field	Payment Payment_Amount_Positive Payment_Highlights
	Assigned To	Issue	Custom Field	Issue
	Availability	Menu Item	Custom Field	Menu Item Menu_Item_Highlights
	Category	Menu Item	Custom Field	Menu Item Menu_Item_Highlights
	Contact Number	Restaurant	Custom Field	Restaurant
	Cuisine Type	Restaurant	Custom Field	Restaurant Restaurant_Highlights
	Customer		Tab	FoodieConnect
	Customer		Custom Object	Customer All OrderHandle
	Customer	Issue	Custom Field	Issue Issue_Highlights
	Customer	Order	Custom Field	Order Order_Highlights OrderHandle
	Customer Layout	Customer	Page Layout	Customer
	Customer_Highlights	Customer	Compact Layout	Customer
	Delivery		Tab	FoodieConnect
	Delivery		Custom Object	Delivery All
	Delivery Date	Delivery	Custom Field	Delivery
	Delivery Layout	Delivery	Page Layout	Delivery
	Delivery Number	Delivery	Custom Field	Delivery Delivery_Highlights
	Delivery Status	Delivery	Custom Field	Delivery Delivery_Highlights

Purpose:
Packaging bundles all necessary components (objects, fields, validation rules, Apex code) for reuse or distribution in other Salesforce orgs.

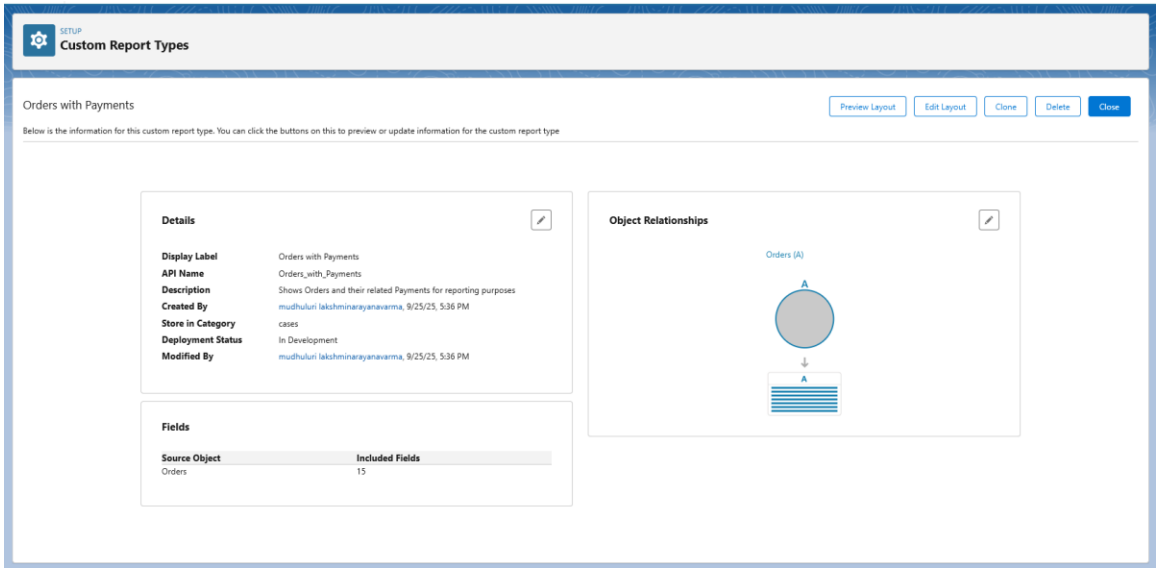
Detailed Steps Implemented:

- Created an Unmanaged Package named FoodieConnect_Unmanaged containing:
    - Core Objects: Customer, Order, Payment, Delivery, Restaurant, Menu Item, Issue.
    - Custom fields and validation rules enforcing business logic (e.g., positive order totals).
    - User interface elements: Page layouts, tabs, Lightning pages, and utility bar components.
    - Apex classes and triggers for business automation.
- Attempted to upload the package to the AppExchange or another org but failed due to missing dependent components (fields/layouts referenced in triggers and validation rules).

Reports

- **Tabular Reports:** Simple lists for Customers, Orders, Payments[reports.png].
- **Summary Reports:** Orders grouped by Status, Deliveries grouped by Rider[reports.png].
- **Matrix Reports:** Orders vs Restaurants matrix with total sales[reports.png].
- **Joined Reports:** Combined Customer & Order reports for cross-object insights[reports.png].

Outcome: **Management efficiently tracks performance, deliveries, and payments.**



30. Report Types

Custom Report Types were created for advanced cross-object insights:

- Customer with Orders
- Orders with Payments
- Delivery with Orders

Purpose: Enable complex reporting beyond standard Salesforce objects.
Status: Types deployed and available to all users.

---

31. Dashboards

Custom dashboards provide instant KPIs:

- **FoodieConnect Overview:** Orders, Payments, Customer Growth[reports.png].
- **Delivery Performance:** Deliveries by status and rider[reports.png].

Components: Charts, tables, metrics.
Outcome: Executives and managers get at-a-glance insights.

---

32. Dynamic Dashboards

- Displays data specific to the logged-in user's role.

- Example: Delivery staff see assigned deliveries; Admin views all data.

Benefit: Personalized, role-specific analytics without duplicating dashboards.

---

33. Sharing Settings

Organization-Wide Defaults (OWD):

- Customer: Public Read/Write
- Delivery: Public Read/Write
- Issue: Public Read/Write
- Menu Item: Public Read/Write
- Order: Controlled by Parent
- Payment: Public Read/Write
- Restaurant: Public Read/Write

Sharing Rules:

- Customers → Support Agents (Read/Write)
- Orders → Managers & Admins (Full Access)
- Payments → Managers (Full Access)

Outcome: Proper access and security across user roles.

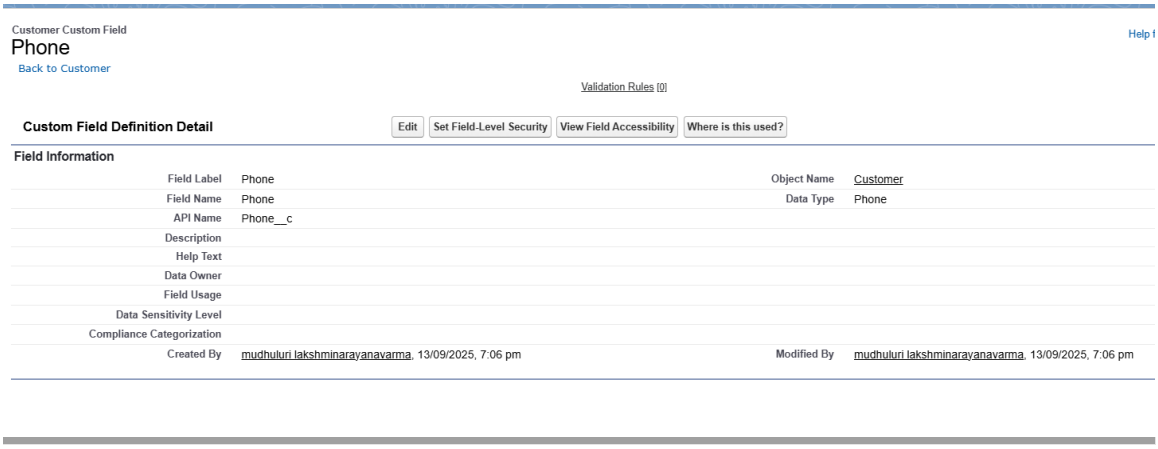
Action	Criteria	Shared With	Access Level
Edit | Del	Owner in Role: Rider	Role: food connect CEO	Read Only

---

34. Field Level Security

- **Admin:** Full field access
- **Support:** Customer & Issue fields
- **Riders:** Delivery and Order fields only
- **Customers:** Own customer record and order history

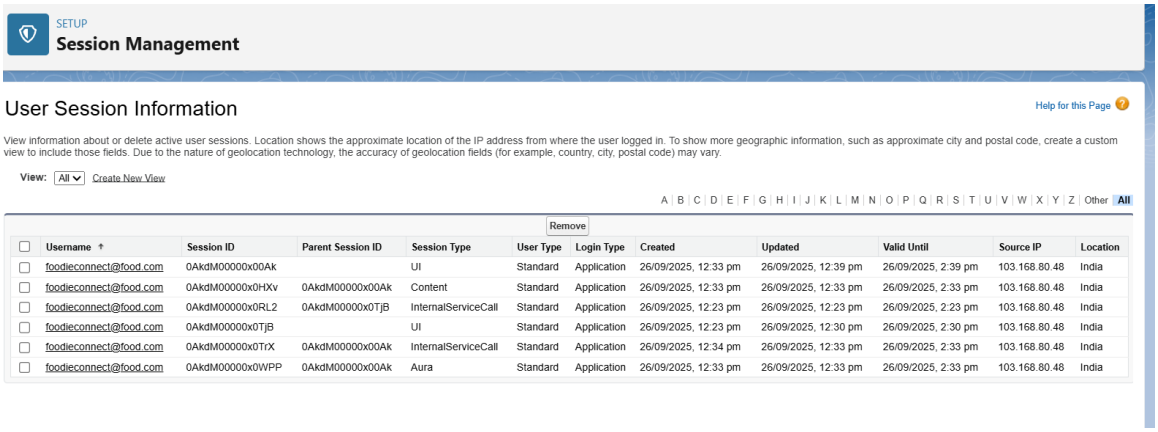
Sensitive fields (Payment info, Loyalty Points) locked down.



35. Session Settings

- Session Timeout: 2 hours
- Force logout on timeout: Enabled
- Lock sessions to domain/IP: Admins only
- Lightning Web Security: Enabled
- Multi-Factor Authentication: Admin & Managers (high assurance)

Benefit: Secures logins and prevents unauthorized access.



36. Login IP Ranges

- **Admins:** Restricted to office IP range (*e.g.*, 103.168.80.48)
- **Delivery & Customers:** No IP restrictions (mobile access supported)

Admin logins protected; other users can access anywhere.

37. Audit Trail

- Tracks changes across all setup activities:
    - Object creation (Customer, Order, Payment, Delivery, Issue)
    - Validation, Sharing, Duplicate Rules
    - Login IP range updates
- Audit Trail export available in CSV for reporting.

Full transparency of config changes and user actions.

38. Deployment

A custom Lightning App named FoodieConnect has been created with access to Restaurants, Menu Items, Customers, Orders, Deliveries, Payments, and Issues.