AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
(IOIT)
ADDING VALUE TO ENGINEERING
An Autonomous Institute Affiliated to Savitribai Phule Pune University
Approved by AICTE, New Delhi and Recognised by Govt. of Maharashtra
Accredited by NAAC with "A+" Grade | NBA - 5 UG Programmes

Department of Computer Engineering

A Mini Project Report (Cloud Computing Lab)

"Restaurant Billing System"

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AISSMS IOIT

T. Y. BTech Engineering

SUBMITTED BY

Student Roll No	Student Name
B-84	M Venkata Soodarshan

सत्याला मरण नाही

AISSMS
INSTITUTE OF INFORMATION TECHNOLOGY
(IOIT)
ADDING VALUE TO ENGINEERING
An Autonomous Institute Affiliated to Savitribai Phule Pune University
Approved by AICTE, New Delhi and Recognised by Govt. of Maharashtra
Accredited by NAAC with "A+" Grade | NBA - 5 UG Programmes

Department of Computer Engineering

CERTIFICATE

This is to certify that the project report

“Restaurant Billing System”

Student Roll No	Student Name
B-84	M.Venkata Soodarshan

is a bonafide student of this institute and the work has been carried out by him/her under the supervision of **Mrs. Veena Bhende** and it is approved for the partial fulfillment of the Department of Computer Engineering AISSMS IOIT.

(Mrs. VEENA BHENDE)
Guide
Department of Computer Engineering
Place : Pune

(Dr. S. N. ZAWARE)
Head
Department of Computer Engineering
Date

Procedure:

Step 1: Define Requirements

Before you start coding, gather the key requirements for your Restaurant Billing System:

Users (e.g., restaurant staff) should be able to:

- Create new bills for customer orders.
- View existing bills and their details.
- Update bills for modifications (e.g., adding or removing items).
- Delete bills when necessary (e.g., cancelled orders).

Step 2: Create Custom Objects

1. **Restaurant Object:**
    - Go to Setup > Object Manager > Create Object.
    - Create a new object called Restaurant Orders with the following fields:
        - Item Quantities (Text)
        - Selected Items (Text)
        - Menu Items (Long Text Area)
        - Total Price (Currency)

Step 3: Create Fields

Add fields to the Restaurant object as described:

- Go to Object Manager > Restaurant_Orders > Fields & Relationships.
- Create the necessary fields:
    - Item Quantities (Text)
    - Selected Items (Text)
    - Menu Items (Long Text Area)
    - Total Price (Currency)

Step 4: Create Apex Class for Restaurant Billing System

Create an Apex class that will handle the creation, update, and deletion of orders. Here's a basic structure for the Restaurant Billing System class:

```
public with sharing class RestaurantOrder {
    // Field declarations
    public Map<String, Integer> itemQuantities { get; set; } // Store quantities of selected items using item names as keys
    public Decimal totalPrice { get; set; } // Store the total price of the selected items
    public Map<String, Decimal> menuItems { get; set; } // Menu items with their prices
```

```
// Constructor to initialize the menu
public RestaurantOrder() {
    // Initialize the menu with item names and prices in Rupees
    menuItems = new Map<String, Decimal>();
    menuItems.put('Pizza', 150); // Price in Rupees
    menuItems.put('Burger', 100);
    menuItems.put('Pasta', 80);
    menuItems.put('Dosa', 60);
    menuItems.put('Soda', 20);

    itemQuantities = new Map<String, Integer>(); // Initialize empty quantity map
    totalPrice = 0; // Initialize the total price
}

// Method to calculate the total price based on selected items and their quantities
public void calculateTotalPrice() {
    totalPrice = 0; // Reset total price for new calculation

    // Loop through selected items and calculate the total price
    for (String itemName : itemQuantities.keySet()) {
        Integer quantity = itemQuantities.get(itemName);
        if (menuItems.containsKey(itemName)) {
            totalPrice += menuItems.get(itemName) * quantity;
        } else {
            System.debug('Invalid item name: ' + itemName);
        }
    }

    System.debug('Total Price: ₹' + totalPrice); // Displaying total price in Rupees
}

// Method to save the restaurant order to the custom object
public void saveOrder() {
    Restaurant_Order__c orderRecord = new Restaurant_Order__c();
    orderRecord.Selected_Items__c = mapToSelectedItemsString(itemQuantities);
    orderRecord.Total_Price__c = totalPrice;
     orderRecord.Item_Quantities__c = mapToString(itemQuantities); // Convert the map to string for storage
        orderRecord.Menu_Items__c = mapToMenuItemsString(menuItems); // Convert the menu to long text area for storage

    try {
        insert orderRecord;
        System.debug('Order saved successfully: ' + orderRecord.Id);
    } catch (DmlException e) {
        System.debug('Error saving order: ' + e.getMessage());
    }
}
// Helper method to convert item quantities map to a string format
private String mapToString(Map<String, Integer> itemMap) {
    List<String> itemList = new List<String>();
    for (String key : itemMap.keySet()) {
        itemList.add(key + ':' + itemMap.get(key));
    }
    return String.join(itemList, ',');
}

    // Helper method to convert item quantities to selected item names
(Selected_Items__c field)
private String mapToSelectedItemsString(Map<String, Integer> itemMap) {
    List<String> selectedItemsList = new List<String>();
    for (String key : itemMap.keySet()) {
        selectedItemsList.add(key);
    }
    return String.join(selectedItemsList, ',');
}

  // Helper method to convert menu items map to a string format for the
Menu_Items__c field
private String mapToMenuItemsString(Map<String, Decimal> menuMap) {
    List<String> menuList = new List<String>();
    for (String key : menuMap.keySet()) {
        menuList.add(key + ' - ₹' + menuMap.get(key)); // Display prices in Rupees
    }
    return String.join(menuList, '\n');
}

// Method to display the menu to the user
public String displayMenu() {
    String menu = 'Restaurant Menu:\n';
    for (String itemName : menuItems.keySet()) {
        menu += itemName + ' - ₹' + menuItems.get(itemName) + '\n'; // Display
prices in Rupees
    }
    return menu;
}

// Method to add an item and its quantity to the order
public void addItemToOrder(String itemName, Integer quantity) {
    if (menuItems.containsKey(itemName)) {
        if (itemQuantities.containsKey(itemName)) {
            itemQuantities.put(itemName, itemQuantities.get(itemName) + quantity);
// Add to existing quantity
        } else {
            itemQuantities.put(itemName, quantity); // Add new item with quantity
        }
        System.debug('Added ' + quantity + ' of ' + itemName + ' to the order.');
    } else {
```

```
        System.debug('Invalid item name: ' + itemName);
    }
}

// Method to remove an item from the order
public void removeItemFromOrder(String itemName) {
    if (itemQuantities.containsKey(itemName)) {
        itemQuantities.remove(itemName);
        System.debug('Removed item ' + itemName + ' from the order.');
    } else {
        System.debug('Item not found in order: ' + itemName);
    }
}

// Method to reset the entire order
public void resetOrder() {
    itemQuantities.clear(); // Clear all items from the order
    totalPrice = 0; // Reset the total price
    System.debug('Order has been reset.');
}

// Method to display an order summary
public String displayOrderSummary() {
    String summary = 'Order Summary:\n';
    for (String itemName : itemQuantities.keySet()) {
            summary += 'Item: ' + itemName + ', Quantity: ' + itemQuantities.get(itemName) + '\n';
        }
        summary += 'Total Price: ₹' + totalPrice + '\n'; // Display total price in Rupees
        return summary;
    }
}
```

Step 6: Lightning Component

If you want a UI for your Restaurant Billing System, you can create either a Visualforce Page or a Lightning Component. Here's a sample code for a basic

Visualforce page:

```
<apex:page controller="RestaurantOrder">
  <h1>Restaurant Billing System</h1>

  <h2>Menu</h2>
  <pre>{!displayMenu()}</pre>

  <h2>Add Item to Order</h2>
  <apex:form>
    <apex:inputText value="{!itemName}" placeholder="Item Name" />
    <apex:inputText value="{!quantity}" placeholder="Quantity" />
```

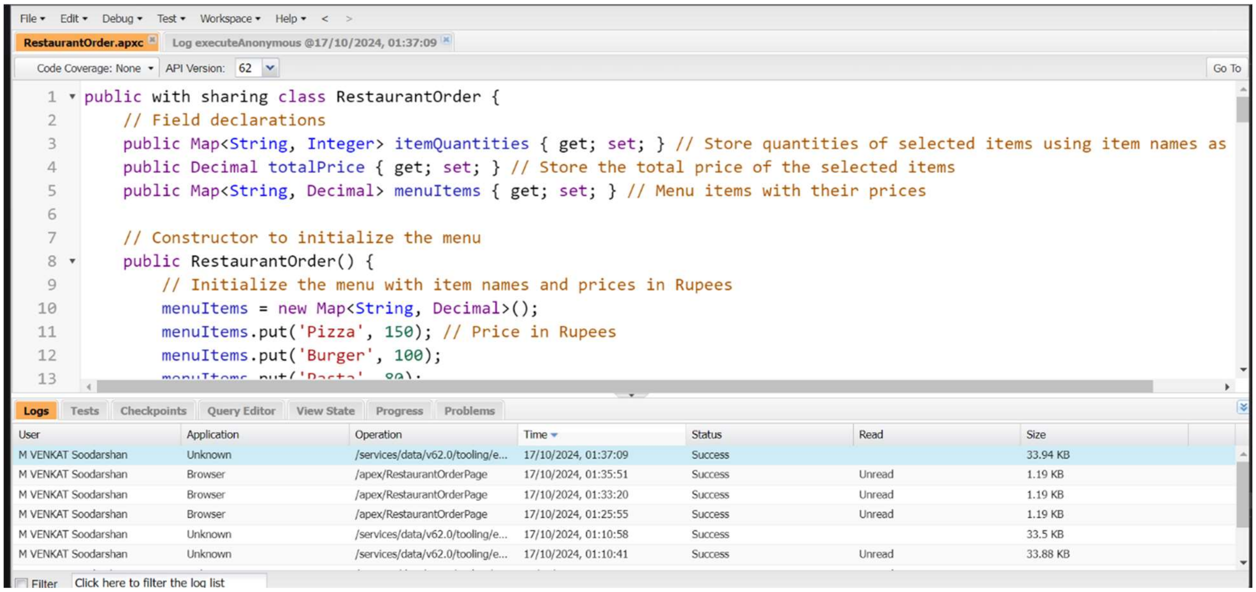
```
        <apex:commandButton value="Add Item" action="{!addItemToOrder}" />
    </apex:form>

    <h2>Order Summary</h2>
    <pre>{!displayOrderSummary()}</pre>

    <apex:commandButton value="Save Order" action="{!saveOrder}" />
    <apex:commandButton value="Reset Order" action="{!resetOrder}" />
</apex:page>
```

Step 7: Deploy and Test

- Deploy your Apex code to your Salesforce org.
- Test the functionality by adding, updating, and deleting Orders.
- Ensure that the data is stored correctly and appears in the UI if applicable.

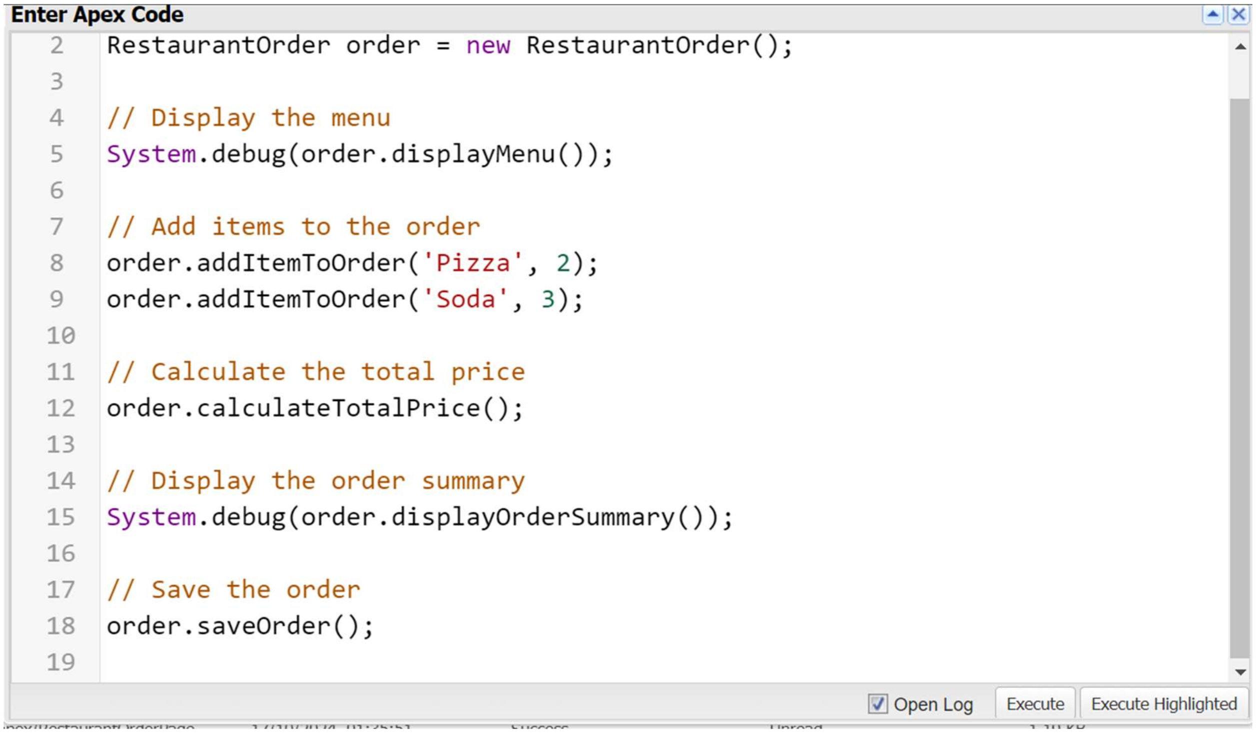
By following these steps, you will have a functional Restaurant_Order__c built in Salesforce using Apex. You can further enhance it by adding validation rules, triggers, reports, and dashboards.

PROJECT OUTPUT:

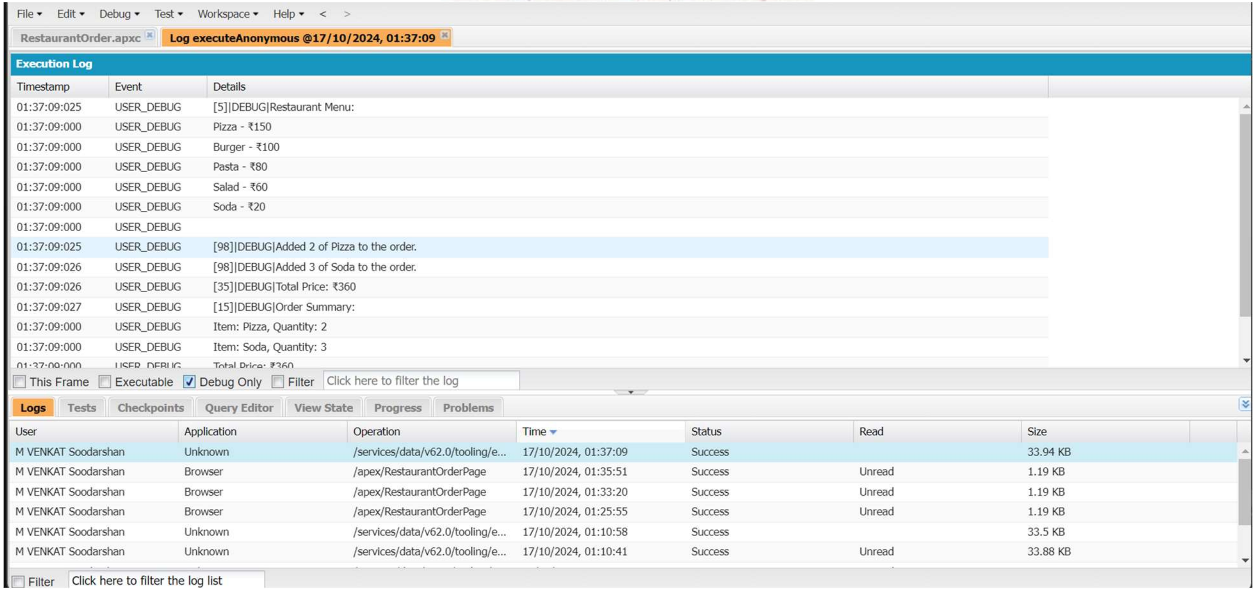


```
public with sharing class RestaurantOrder {
    // Field declarations
    public Map<String, Integer> itemQuantities { get; set; } // Store quantities of selected items using item names as
    public Decimal totalPrice { get; set; } // Store the total price of the selected items
    public Map<String, Decimal> menuItems { get; set; } // Menu items with their prices

    // Constructor to initialize the menu
    public RestaurantOrder() {
        // Initialize the menu with item names and prices in Rupees
        menuItems = new Map<String, Decimal>();
        menuItems.put('Pizza', 150); // Price in Rupees
        menuItems.put('Burger', 100);
        menuItems.put('Pasta', 80);
```

User	Application	Operation	Time ▾	Status	Read	Size
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:37:09	Success		33.94 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:35:51	Success	Unread	1.19 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:33:20	Success	Unread	1.19 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:25:55	Success	Unread	1.19 KB
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:10:58	Success		33.5 KB
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:10:41	Success	Unread	33.88 KB



```
RestaurantOrder order = new RestaurantOrder();

// Display the menu
System.debug(order.displayMenu());

// Add items to the order
order.addItemToOrder('Pizza', 2);
order.addItemToOrder('Soda', 3);

// Calculate the total price
order.calculateTotalPrice();

// Display the order summary
System.debug(order.displayOrderSummary());

// Save the order
order.saveOrder();

```


File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
RestaurantOrder.apxc
Log executeAnonymous @17/10/2024, 01:37:09

Execution Log

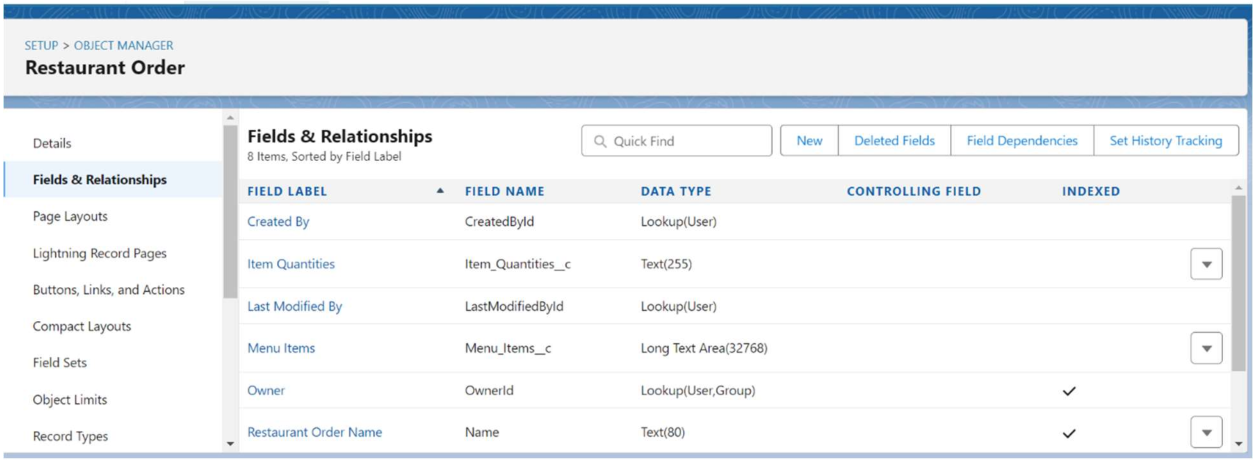
Timestamp	Event	Details
01:37:09:025	USER_DEBUG	[5]|DEBUG|Restaurant Menu:
01:37:09:000	USER_DEBUG	Pizza - ₹150
01:37:09:000	USER_DEBUG	Burger - ₹100
01:37:09:000	USER_DEBUG	Pasta - ₹80
01:37:09:000	USER_DEBUG	Salad - ₹60
01:37:09:000	USER_DEBUG	Soda - ₹20
01:37:09:000	USER_DEBUG	
01:37:09:025	USER_DEBUG	[98]|DEBUG|Added 2 of Pizza to the order.
01:37:09:026	USER_DEBUG	[98]|DEBUG|Added 3 of Soda to the order.
01:37:09:026	USER_DEBUG	[35]|DEBUG|Total Price: ₹360
01:37:09:027	USER_DEBUG	[15]|DEBUG|Order Summary:
01:37:09:000	USER_DEBUG	Item: Pizza, Quantity: 2
01:37:09:000	USER_DEBUG	Item: Soda, Quantity: 3
01:37:09:000	USER_DEBUG	Total Price: ₹360

☐ This Frame ☐ Executable ☑ Debug Only ☐ Filter Click here to filter the log

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:37:09	Success		33.94 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:35:51	Success	Unread	1.19 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:33:20	Success	Unread	1.19 KB
M VENKAT Soodarshan	Browser	/apex/RestaurantOrderPage	17/10/2024, 01:25:55	Success	Unread	1.19 KB
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:10:58	Success		33.5 KB
M VENKAT Soodarshan	Unknown	/services/data/v62.0/tooling/e...	17/10/2024, 01:10:41	Success	Unread	33.88 KB

☐ Filter Click here to filter the log list

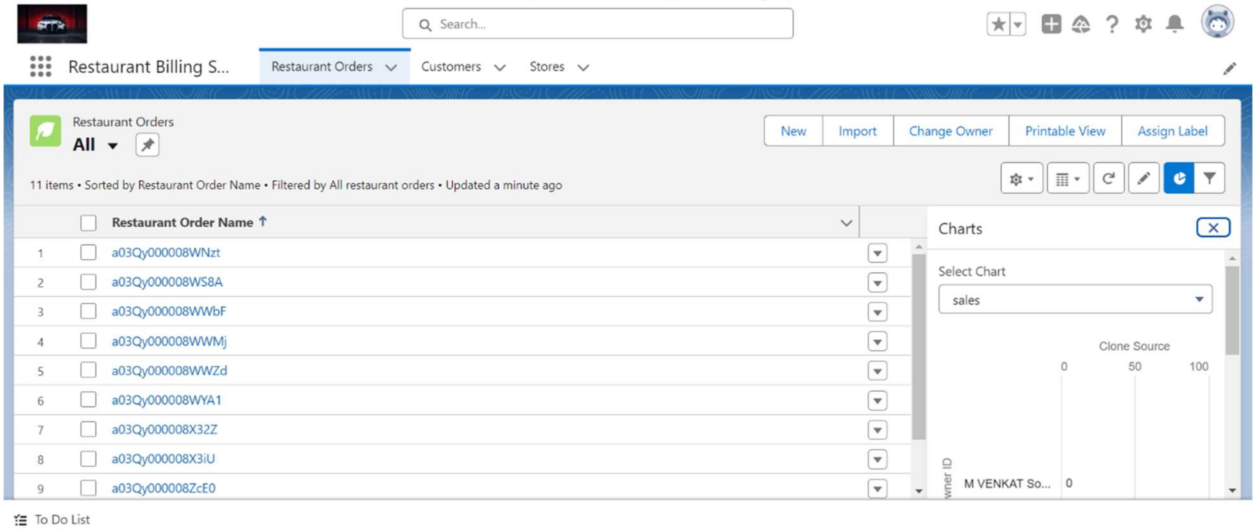


Restaurant Order

Fields & Relationships

8 Items, Sorted by Field Label

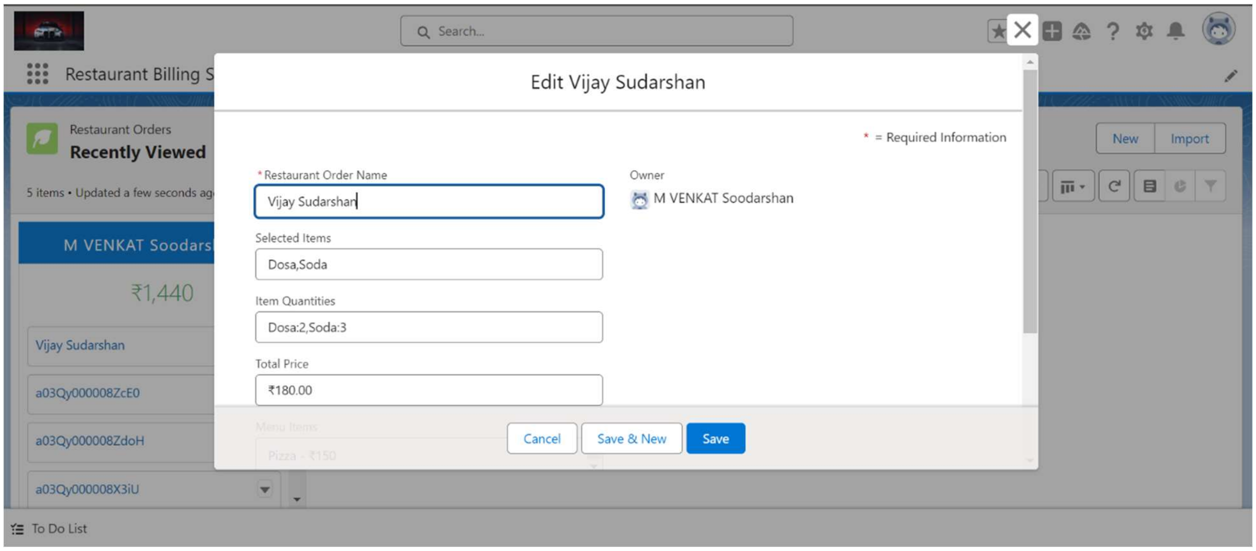
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Item Quantities	Item_Quantities__c	Text(255)		
Last Modified By	LastModifiedById	Lookup(User)		
Menu Items	Menu_Items__c	Long Text Area(32768)		
Owner	OwnerId	Lookup(User,Group)		✓
Restaurant Order Name	Name	Text(80)		✓



Restaurant Orders
All

11 items • Sorted by Restaurant Order Name • Filtered by All restaurant orders • Updated a minute ago

	Restaurant Order Name
1	a03Qy000008WNzt
2	a03Qy000008WS8A
3	a03Qy000008WWbF
4	a03Qy000008WWMj
5	a03Qy000008WWZd
6	a03Qy000008WYA1
7	a03Qy000008X32Z
8	a03Qy000008X3iU
9	a03Qy000008ZcE0



Edit Vijay Sudarshan

* = Required Information

*Restaurant Order Name: Vijay Sudarshan

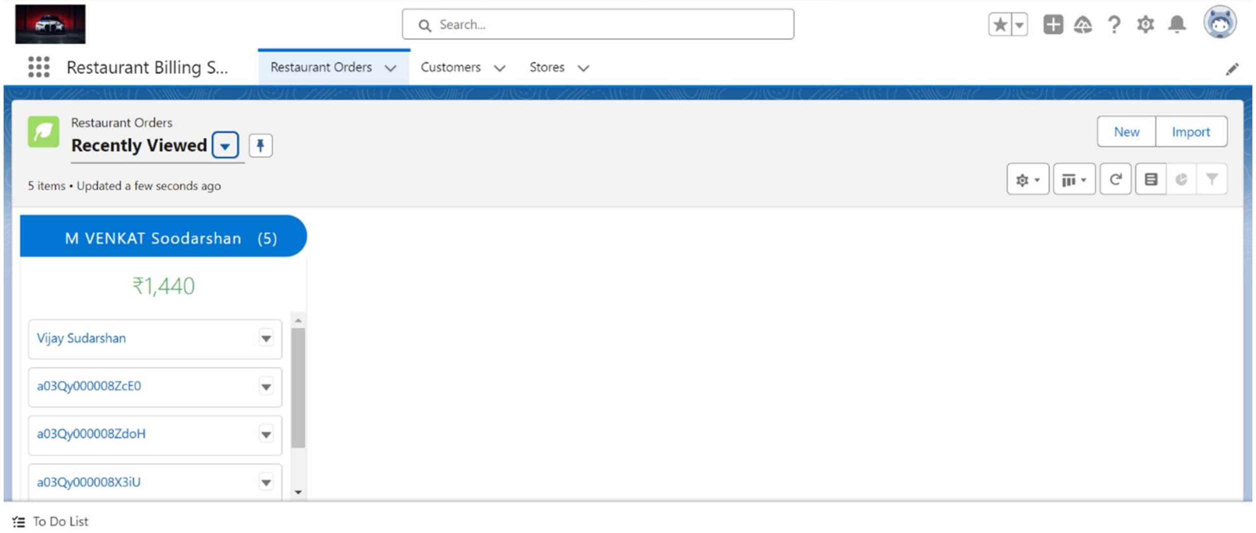
Owner: M VENKAT Soodarshan

Selected Items: Dosa,Soda

Item Quantities: Dosa:2,Soda:3

Total Price: ₹180.00

Menu Items: Pizza - ₹150



Restaurant Orders
Recently Viewed

5 items • Updated a few seconds ago

M VENKAT Soodarshan (5)

₹1,440

Vijay Sudarshan

a03Qy000008ZcE0

a03Qy000008ZdoH

a03Qy000008X3iU