

Sales Automobile Using Salesforce CRM

1. Project Overview:

This project involves the development and customization of Salesforce CRM for managing the entire automobile sales process. The solution will enable streamlined sales operations, from lead management to the creation of invoices. The main features will include the creation of custom objects for automobile information, invoice management, and the automation of key processes such as opportunity and invoice handling.

The objective is to enhance the sales process by leveraging Salesforce's robust tools, including creating objects, fields, custom tabs, Apex triggers, Lightning Web Components (LWC), and more. This will help to manage customer accounts, automobile data, invoices, and related sales activities efficiently.

2. Objective:

The objectives for the Sales Automobile Using Salesforce CRM project are clearly defined to ensure measurable success and alignment with both business and technical goals. These objectives guide the development process and ensure the solution will meet the needs of the business, sales teams, and customers.

The business goals of the Sales Automobile Using Salesforce CRM project are to streamline and optimize the entire sales process, from lead generation to invoice creation, by leveraging Salesforce's powerful CRM capabilities. The project aims to increase sales efficiency by automating key processes, enhance customer engagement through a 360-degree view of customer data, and improve data accuracy for better decision-making.

Additionally, the solution seeks to optimize inventory management by providing real-time visibility into stock levels and sales trends, accelerate

invoice processing for faster revenue realization, and drive business growth by enabling better tracking, reporting, and forecasting of sales performance.

The specific outcomes of the Sales Automobile Using Salesforce CRM project include the creation of custom objects and fields for managing automobile inventory, sales opportunities, and invoices, which will be fully integrated into the Salesforce platform. The project will deliver automation through Apex triggers and workflows to streamline sales processes, from generating invoices to tracking opportunity quantities. Custom tabs and page layouts will be developed to improve user experience, while the Salesforce Lightning App and Lightning Web Components (LWC) will enhance accessibility and interactivity across devices. Additionally, comprehensive reports and dashboards will be implemented to provide real-time insights into sales performance, inventory levels, and customer interactions. This integration and automation will result in faster, more accurate data entry, improved operational efficiency, and enhanced decision-making capabilities for sales teams and management.

3. Salesforce Key Features and Concepts Utilized:

➤ Custom Objects & Fields:

Custom Objects like Automobile Information and Invoice store specific data related to the sales process, allowing you to track cars, pricing, and invoices in Salesforce.

➤ Page Layouts & Lightning Apps :

Custom Page Layouts for Opportunities and Invoices ensure relevant information is easily accessible, while a Lightning App consolidates the interface for users to manage their workflow efficiently.

➤ Apex Triggers & Classes:

Apex Triggers automate processes such as updating the automobile quantity in an Opportunity and creating an invoice when an Opportunity is closed, reducing manual tasks.

➤ Lightning Web Components (LWC):

A custom LWC component displays related Invoice data on Opportunity pages, enhancing the user interface and providing real-time access to invoice details.

➤ Reports & Dashboards:

Reports and Dashboards provide visual insights into sales performance, inventory levels, and invoice statuses, helping track business health and sales trends.

➤ Salesforce Lightning Experience

The Lightning Experience improves user productivity with an intuitive and responsive interface, allowing users to efficiently navigate and manage automobile sales data

➤ Apex Schedulers

Apex Schedulers automate recurring tasks, such as sending invoice reminders or syncing data, ensuring timely actions without manual intervention.

4. Detailed Steps to Solution Design for Automobile Sales Management in Salesforce:

➤ Automobile Information Object:

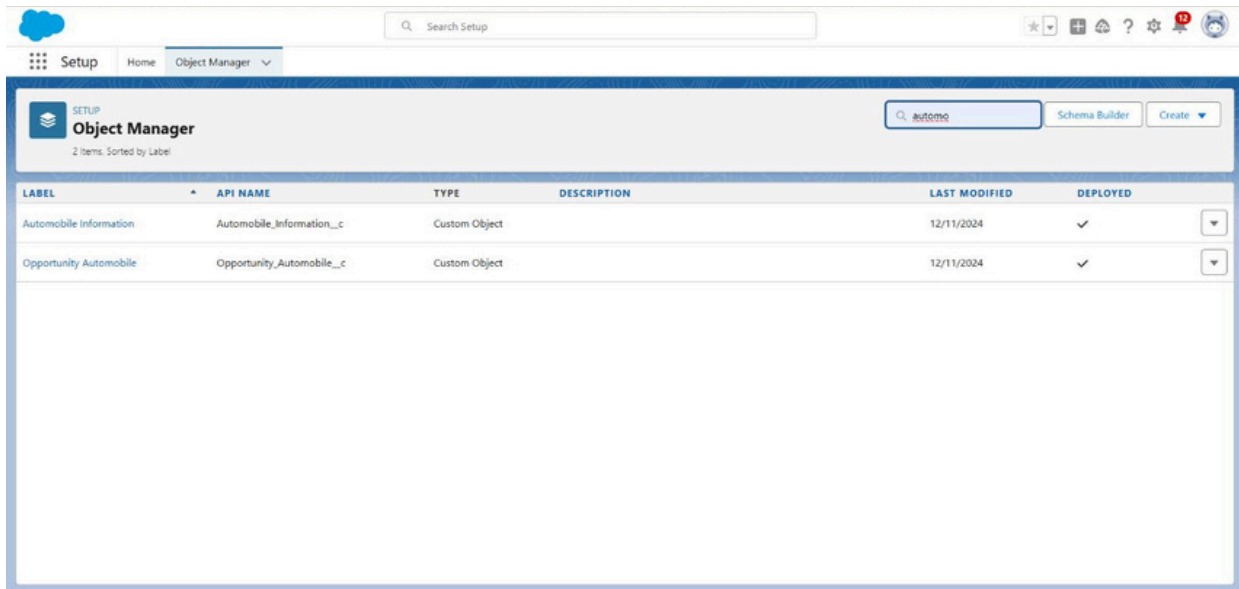
This object stores data related to the automobiles sold, such as make, model, year, price, and VIN.

➤ Invoice Object:

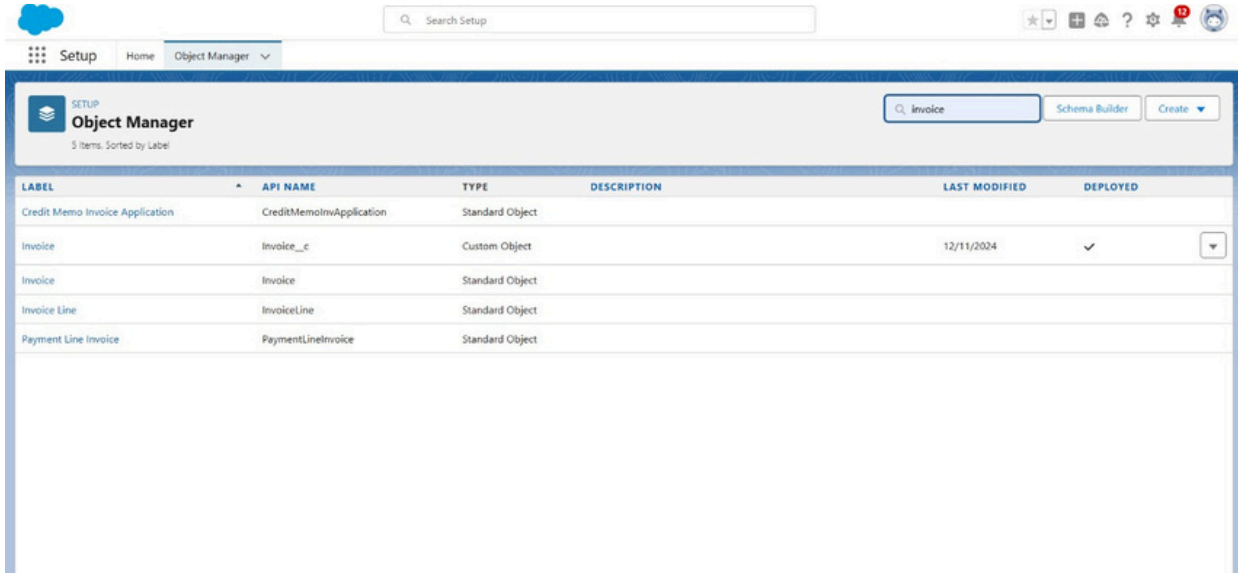
This object holds details about invoices related to Opportunities.

➤ Automobile Object:

This object may track additional attributes related to automobiles or individual transactions (e.g., specific inventory details).



LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Automobile Information	Automobile_Information__c	Custom Object		12/11/2024	✓
Opportunity Automobile	Opportunity_Automobile__c	Custom Object		12/11/2024	✓

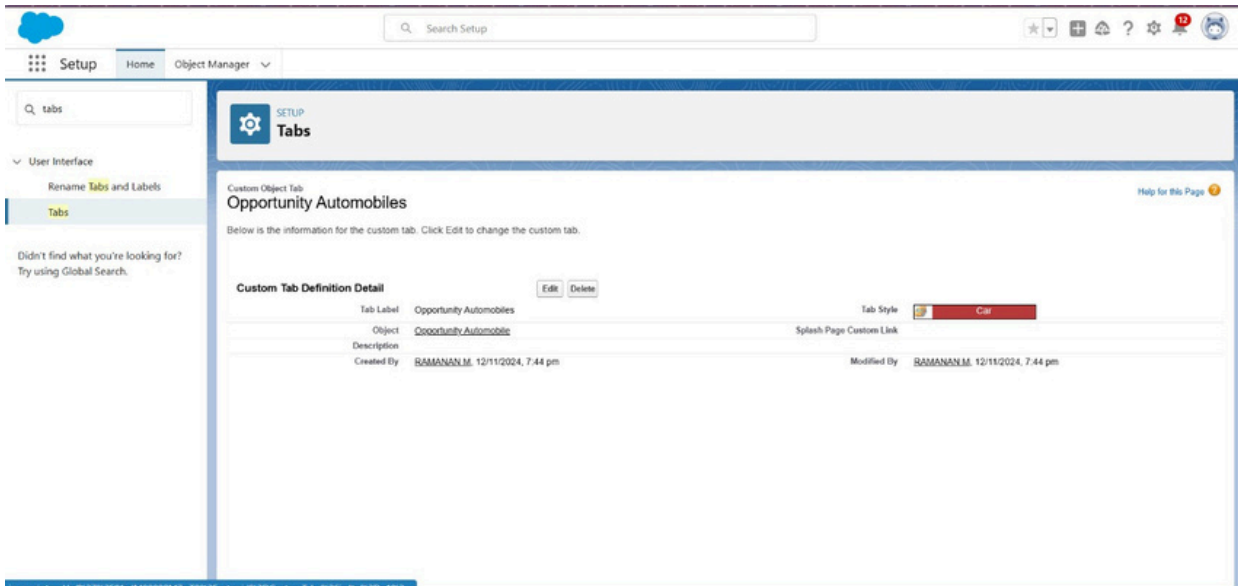


The screenshot shows the Salesforce Object Manager setup page. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main header area has a search bar with 'Invoice' entered, and buttons for 'Schema Builder' and 'Create'. Below the header is a table listing objects:

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Credit Memo Invoice Application	CreditMemoInvApplication	Standard Object			
Invoice	Invoice_c	Custom Object		12/11/2024	✓
Invoice	Invoice	Standard Object			
Invoice Line	InvoiceLine	Standard Object			
Payment Line Invoice	PaymentLineInvoice	Standard Object			

➤ Creating a Custom Tab:

Custom object tabs are the user interface for custom applications that you build in salesforce.com. They look and behave like standard salesforce.com tabs such as accounts, contacts, and opportunities.



The screenshot shows the Salesforce Tabs setup page for a custom object tab named 'Opportunity Automobiles'. The left sidebar shows the 'Setup' menu with 'User Interface' expanded, and 'Tabs' selected. The main content area displays the 'Custom Tab Definition Detail' for 'Opportunity Automobiles'.

Custom Tab Definition Detail		Tab Style
Tab Label	Opportunity Automobiles	Car
Object	Opportunity Automobile	Splash Page Custom Link
Created By	RAMANANJ.M. 12/11/2024, 7:44 pm	Modified By
		RAMANANJ.M. 12/11/2024, 7:44 pm

➤ Lightning App

- Create a Custom Lightning App that integrates the following components:
 - 1.Opportunity Records ,Automobile Information records , Invoices related to Opportunities
- The app should include:
 - Navigation to all relevant objects (Opportunities, Automobiles, Invoices).
 - A dashboard to visualize Total Sales, Invoices due,
 - Opportunity stage.

The screenshot displays the Salesforce Lightning App Builder interface. The top navigation bar includes a back arrow, 'Lightning App Builder', 'App Settings', 'Pages', and the app name 'Sales Automobile Using Salesforce CRM', along with a 'Help' icon. The left sidebar, titled 'App Settings', lists 'App Details & Branding' (selected), 'App Options', 'Utility Items (Desktop Only)', 'Navigation Items', and 'User Profiles'. The main content area is titled 'App Details & Branding' and contains the following sections:

- App Details**: Includes fields for 'App Name' (Sales Automobile Using Salesforce CRM), 'Developer Name' (Sales_Automobile_Using_Salesforce_CRM), and 'Description' (Sales Automobile Using Salesforce CRM).
- App Branding**: Includes an 'Image' upload button, a 'Primary Color Hex Value' dropdown set to '#0070C2', and 'Org Theme Options' with a checkbox 'Use the app's image and color instead of the org's custom theme'.
- App Launcher Preview**: Shows a preview of the app launcher with a blue square icon containing 'SA' and the text 'Sales Automobile Using Sal...' and 'Sales Automobile Using Salesforce CRM'.

Setup Home Object Manager Search Setup

SETUP > OBJECT MANAGER

Opportunity Automobile

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

Restriction Rules

Scoping Rules

Edit Opportunity Automobile Custom Field

Opportunity

Have feedback on lookup filters? Comment on IdealExchange Help for this Page

Custom Field Definition Edit Change Field Type Save Cancel

Field Information

Field Label Opportunity Data Type Master-Detail

Field Name Opportunity

Description

Help Text

Data Owner User

Field Usage --None--

Data Sensitivity Level --None--

Compliance Categorization

Available: PII, HIPAA, GDPR, PCI

Chosen:

Master-Detail Options

➤ Creating Fields and Relationships:

Setup Home Object Manager Search Setup

SETUP > OBJECT MANAGER

Opportunity Automobile

Details

Fields & Relationships

8 Items, Sorted by Field Label

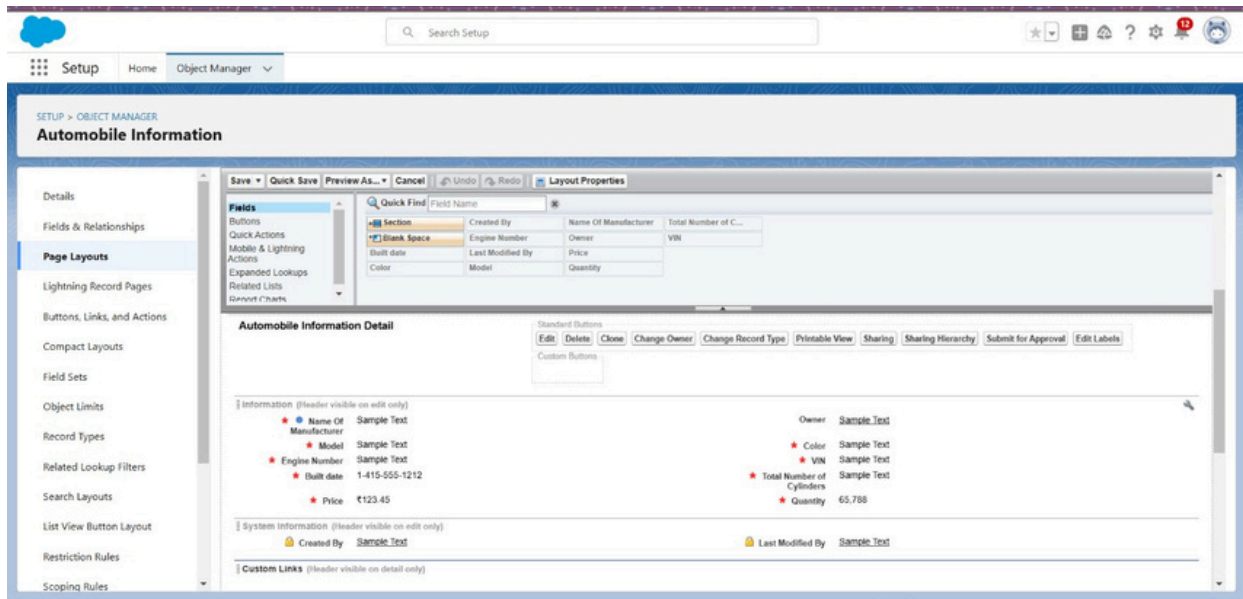
Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Automobile	Automobile__c	Lookup(Automobile Information)		✓
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Opportunity	Opportunity__c	Master-Detail(Opportunity)		✓
Opportunity Automobile Id	Name	Auto Number		✓
Quantity	Quantity__c	Number(18, 0)		
Total Price	Total_Price__c	Formula (Currency)		
Unit Price	Unit_Price__c	Formula (Currency)		

➤ Page Layouts:

Page Layout in Salesforce allows us to customize the design and organize detail and edit pages of records in Salesforce. Page layouts can be used to control the appearance of fields, related lists, and custom links on standard and custom objects' detail and edit pages.

Edit the Page layout for Automobiles Information



➤ Apex Triggers:

An Apex trigger is a set of instructions that execute when certain events occur on a Salesforce object (like when a record is created, updated, deleted, or restored).

Creating OpportunityHandlerClass

```

File • Edit • Debug • Test • Workspace • Help • <
DeleteClosedLostOpportunities.apex • OpportunityHandlerClass.apex
Code Coverage: None • API Version: 62
Go To

1 public class OpportunityHandlerClass {
2
3
4
5 public static void opportunityAutomobileQuantity(List<Opportunity> LstOpportunity, Map<Id,Opportunity> OldMapOpportunity){
6
7     set<Id> opportunityIds = new set<Id>();
8
9     for(Opportunity opp : LstOpportunity){
10
11         if(opp.StageName == 'Closed Won' ){
12
13             opportunityIds.add(opp.Id);
14
15         }
16
17     }
18 }

```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Creating OpportunityAutomobileHandlerClass

```

File • Edit • Debug • Test • Workspace • Help • <
DeleteClosedLostOpportunities.apex • OpportunityAutomobileHandler.apex
Code Coverage: None • API Version: 62
Go To

1 public class OpportunityAutomobileHandler {
2
3
4 public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c> lstOpportunityAutomobile){
5
6     set<Id> AutomobileIds = new Set<Id>();
7
8     For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
9
10         if(oppAutomobile.Automobile__c != null){
11
12             AutomobileIds.add(oppAutomobile.Automobile__c);
13
14         }
15
16     }
17
18     Map<Id,Automobile_Information__c> lstAutomobileInformation = new map<Id,Automobile_Information__c>({SELECT Id, CreatedById, Quantity__c, Pr

```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Filter Click here to filter the log list

Creating OpportunityAutoMobileTrigger

```
File • Edit • Debug • Test • Workspace • Help • < >
DeleteClosedLostOpportunities.apex • OpportunityAutomobileHandler.apex • OpportunityAutoMobileTrigger.apex
Code Coverage: None • API Version: 62 • Go To
```

```
1 trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before Update) {
2
3     if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
4
5         OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);
6
7     }
8
9 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Filter Click here to filter the log list

Creating InvoiceCreation class

```
File • Edit • Debug • Test • Workspace • Help • < >
DeleteClosedLostOpportunities.apex • InvoiceCreation.apex
Code Coverage: None • API Version: 62 • Go To
```

```
1 public class InvoiceCreation {
2
3     public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity> lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
4
5         set<Id> oppIds = new Set<Id>();
6
7         For(Opportunity opp : lstOpportunity){
8
9             if(opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName != opp.StageName){
10
11                 oppIds.add(opp.Id);
12
13             }
14
15         }
16
17         List<Opportunity_Automobile__c> lstOpportunityAutomobile = [SELECT Unit_Price__c, Total_Price__c, Automobile__c, Quantity__c,Opportunity__c
18
19 ]
```

Logs Tests Checkpoints Query Editor View State Progress Problems

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Creating ContactRoleCheck class

```

1 public class ContactRoleCheck {
2
3 public static void CheckcontactRoleonOpportunity(List<Opportunity> lstOpportunity, Map<Id,Opportunity>OldMapOpportunity){
4
5     List<OpportunityContactRole> lstContactRole = [SELECT Id From OpportunityContactRole WHERE OpportunityId IN: OldMapOpportunity.keySet()];
6
7     For(Opportunity opp : lstOpportunity){
8
9         if(opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName != opp.StageName){
10
11             If(lstContactRole.isEmpty()){
12
13                 opp.adderror('Please add contact Role on opportunity whenever Opportunity is Going to Closed Won.');

```

Creating OpportunityTrigger

```

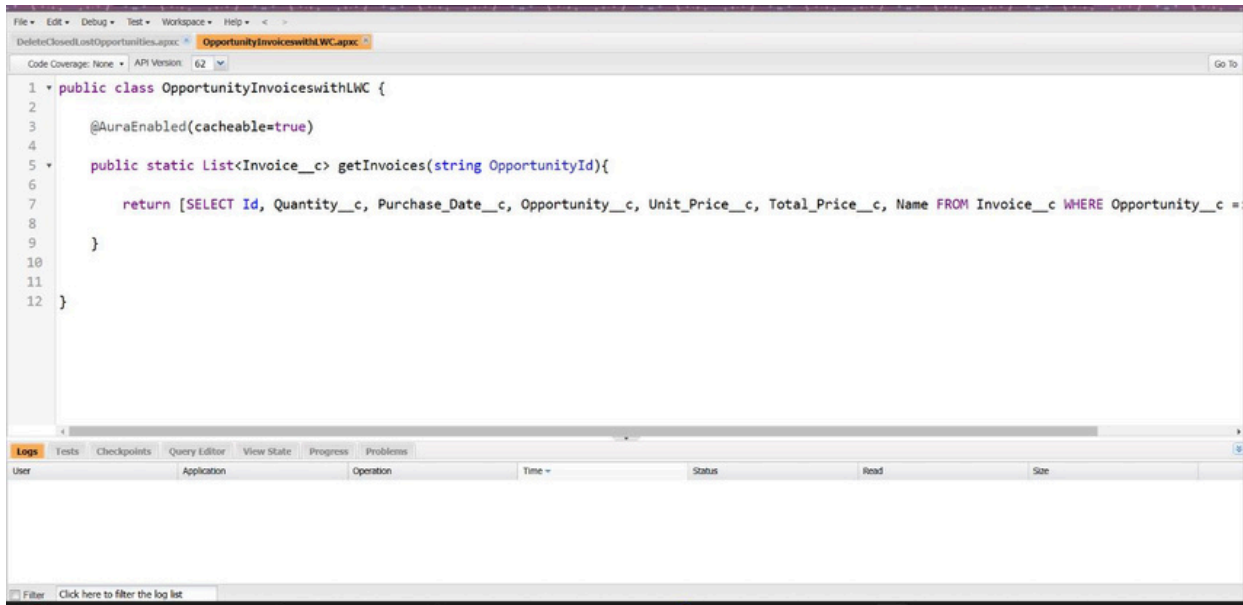
1 trigger OpportunityTrigger on Opportunity (before update, After Update) {
2
3     if(trigger.isbefore && trigger.isUpdate){
4
5         OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
6
7         ContactRoleCheck.CheckcontactRoleonOpportunity(trigger.new, trigger.oldMap);
8
9     }
10
11     IF(trigger.isafter && trigger.isupdate){
12
13         InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
14
15     }
16
17 }

```

➤ LWC Component:

Lightning Web Components (LWC) are a modern framework for building user interfaces in Salesforce. Unlike Aura Components (Salesforce's previous framework), LWC is based on modern web standards, including web components, HTML, CSS, and JavaScript. It offers faster performance, better developer productivity, and is more aligned with web development best practices.

Create apexclass to get invoice



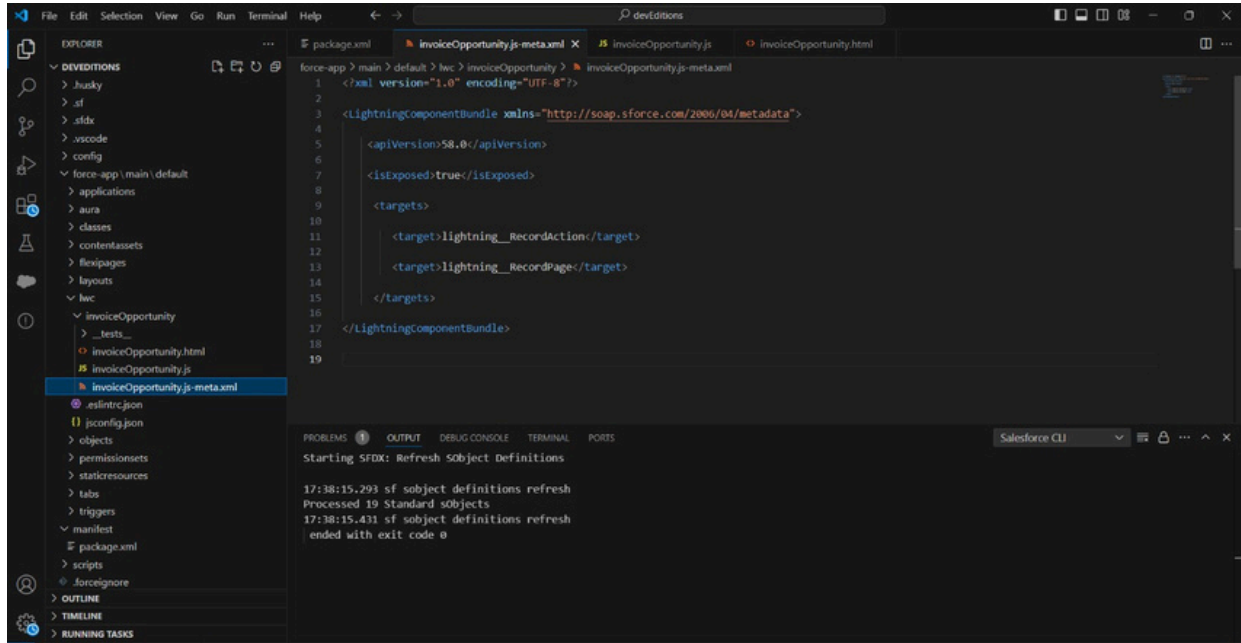
```
1 public class OpportunityInvoiceswithLWC {
2
3     @AuraEnabled(cacheable=true)
4
5     public static List<Invoice__c> getInvoices(string OpportunityId){
6
7         return [SELECT Id, Quantity__c, Purchase_Date__c, Opportunity__c, Unit_Price__c, Total_Price__c, Name FROM Invoice__c WHERE Opportunity__c =:
8
9     }
10
11 }
12 }
```

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Create Lightning Web Component

The lwc component name is InvoiceOpportunity.

XML File

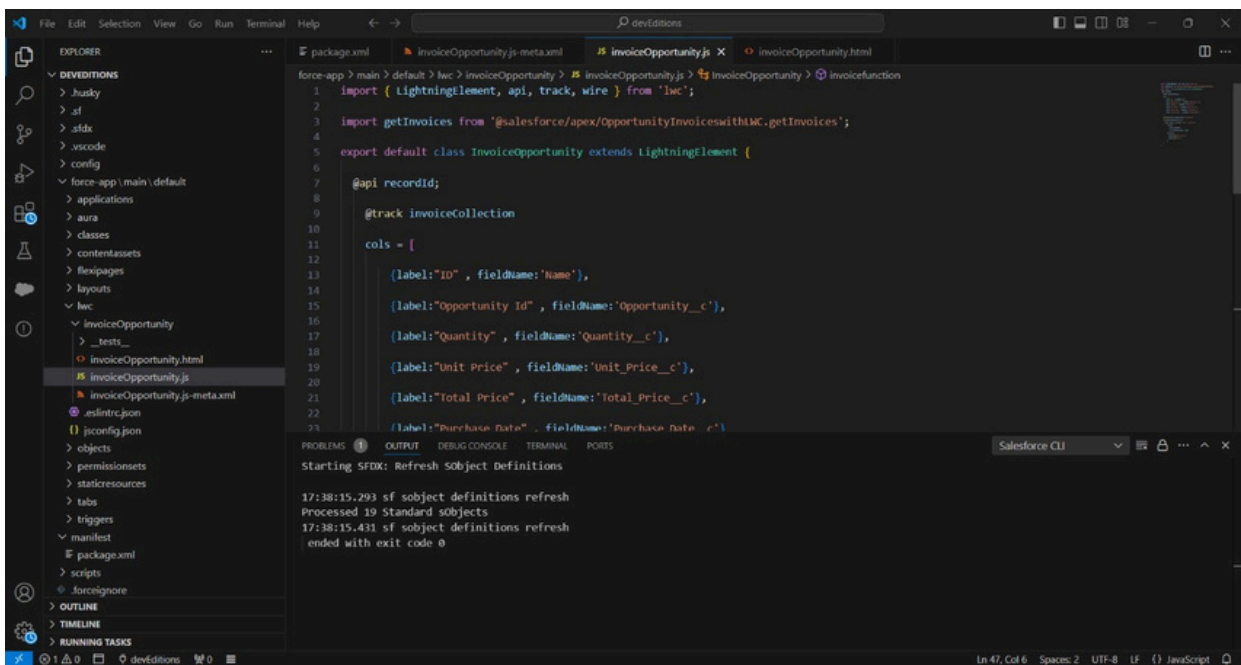


The screenshot shows the VS Code editor with the 'invoiceOpportunity.js-meta.xml' file open. The file contains the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>58.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
    <target>lightning_RecordAction</target>
    <target>lightning_RecordPage</target>
  </targets>
</LightningComponentBundle>
```

The Explorer sidebar on the left shows the project structure, including the 'invoiceOpportunity' folder and its sub-files: 'invoiceOpportunity.html', 'invoiceOpportunity.js', and 'invoiceOpportunity.js-meta.xml' (which is currently selected).

JS File



The screenshot shows the VS Code editor with the 'invoiceOpportunity.js' file open. The file contains the following JavaScript code:

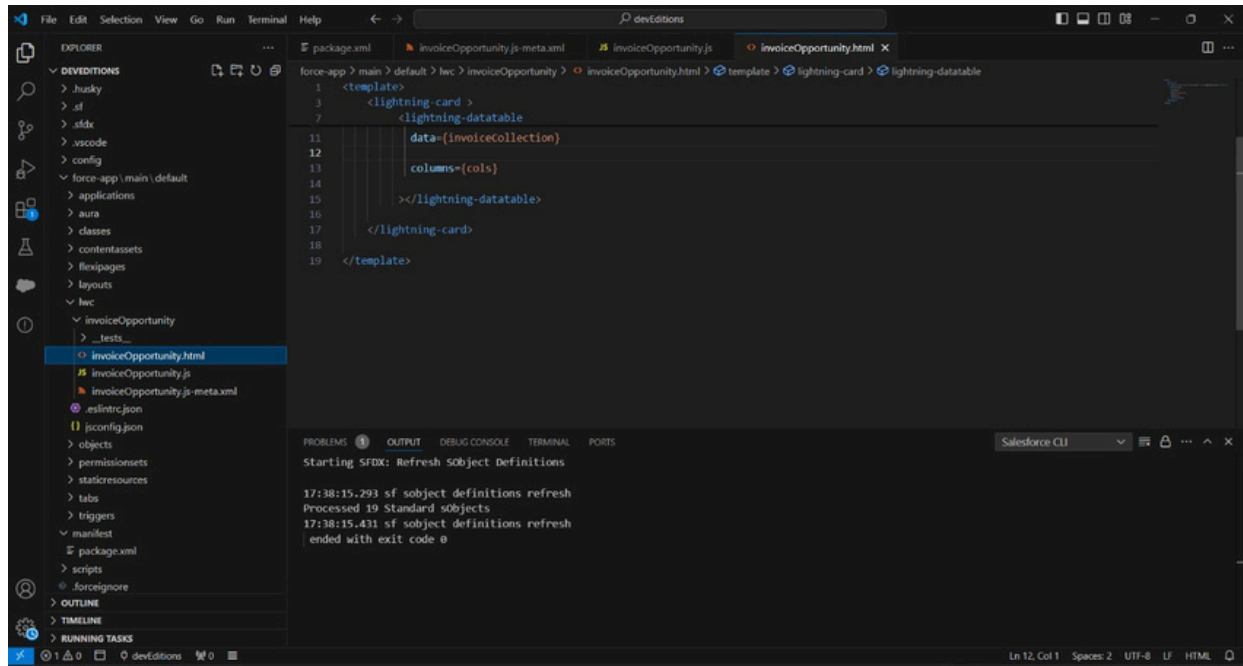
```
import { LightningElement, api, track, wire } from 'lwc';
import getInvoices from '@salesforce/apex/OpportunityInvoicesWithMC.getInvoices';

export default class InvoiceOpportunity extends LightningElement {
  @api recordId;
  @track invoiceCollection;

  cols = [
    {label: 'ID', fieldName: 'Name'},
    {label: 'Opportunity Id', fieldName: 'Opportunity__c'},
    {label: 'Quantity', fieldName: 'Quantity__c'},
    {label: 'Unit Price', fieldName: 'Unit_Price__c'},
    {label: 'Total Price', fieldName: 'Total_Price__c'},
    {label: 'Purchase Date', fieldName: 'Purchase_Date__c'}
  ];
}
```

The Explorer sidebar on the left shows the project structure, including the 'invoiceOpportunity' folder and its sub-files: 'invoiceOpportunity.html', 'invoiceOpportunity.js' (which is currently selected), and 'invoiceOpportunity.js-meta.xml'.

HTML File



```

1 <template>
2   <lightning-card>
3     <lightning-datatable
4       data={invoicecollection}
5       columns={columns}>
6     </lightning-datatable>
7   </lightning-card>
8 </template>

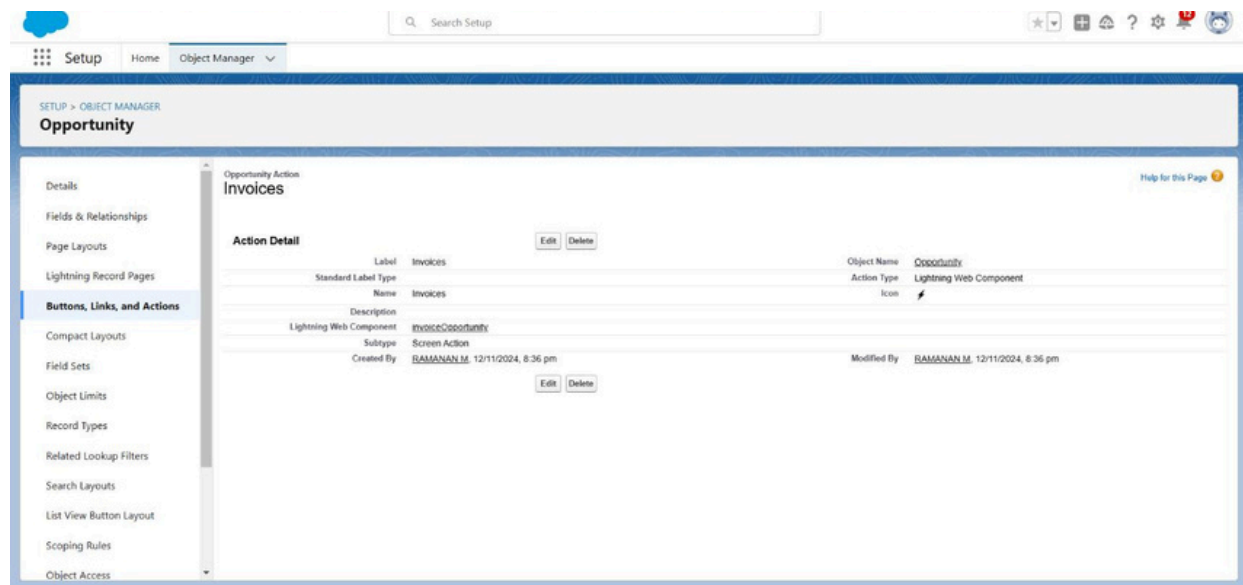
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Starting SFDX: Refresh Object Definitions

17:38:15.293 sf object definitions refresh
 Processed 19 Standard Objects
 17:38:15.411 sf object definitions refresh
 ended with exit code 0

Create Button to add an opportunity



Setup > OBJECT MANAGER

Opportunity

Opportunity Action Invoices

Details

Fields & Relationships

Page Layouts

Lightning Record Pages

Buttons, Links, and Actions

Compact Layouts

Field Sets

Object Limits

Record Types

Related Lookup Filters

Search Layouts

List View Button Layout

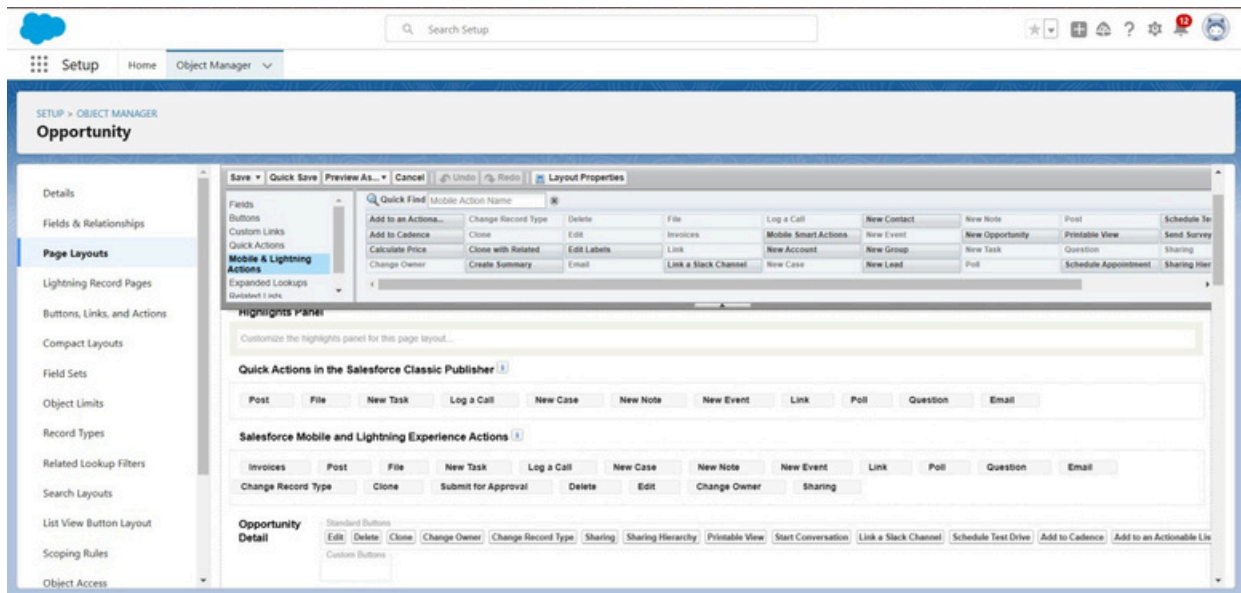
Scoping Rules

Object Access

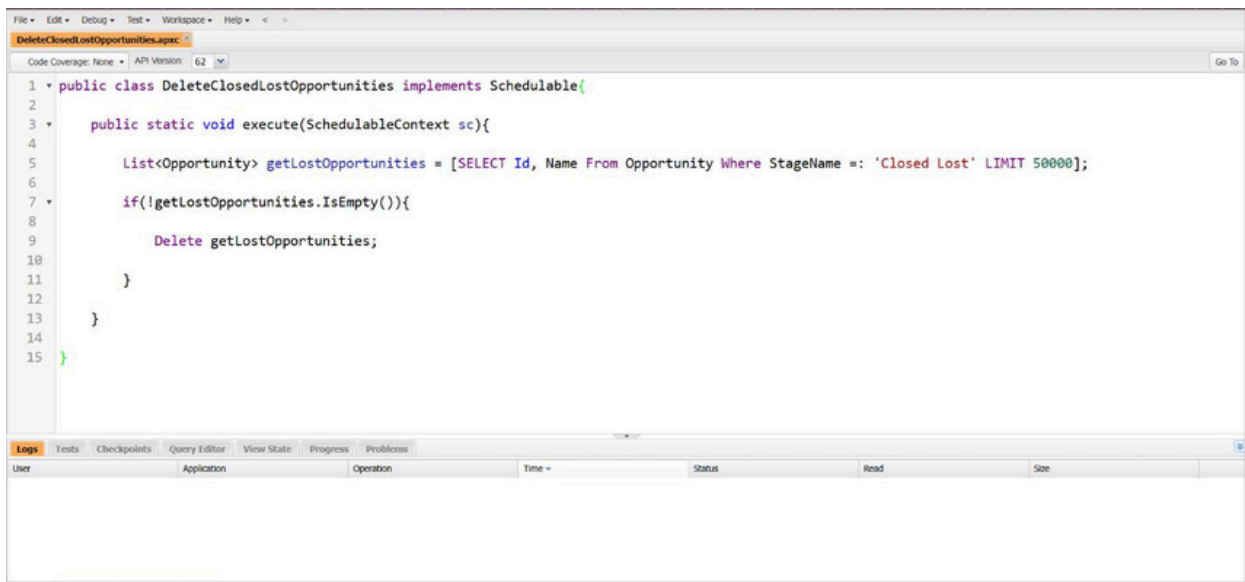
Action Detail

Label	Invoices	Object Name	Opportunity
Standard Label Type		Action Type	Lightning Web Component
Name	Invoices	Icon	
Description	InvoiceOpportunity		
Lightning Web Component	Screen Action		
Subtype			
Created By	RAMANAN M. 12/11/2024, 8:36 pm	Modified By	RAMANAN M. 12/11/2024, 8:36 pm

Add Invoice opportunity into opportunity record page



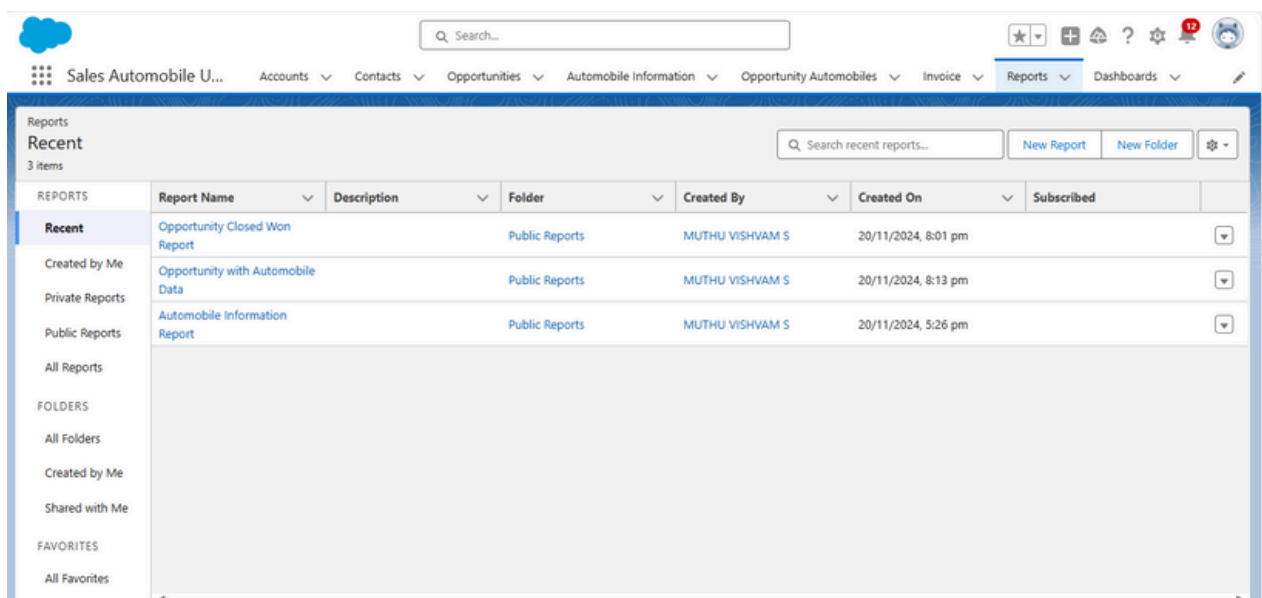
- Apex Schedulers:
Delete opportunity schedule class



➤ Reports:

Reports give you access to your Salesforce data. You can examine your Salesforce data in almost infinite combinations, display it in easy-to-understand formats, and share the resulting insights with others. Before building, reading, and sharing reports, review these reporting basics.

Here we create three reports namely Opportunity with automobile data, opportunity closed won report, and Automobile Information report.

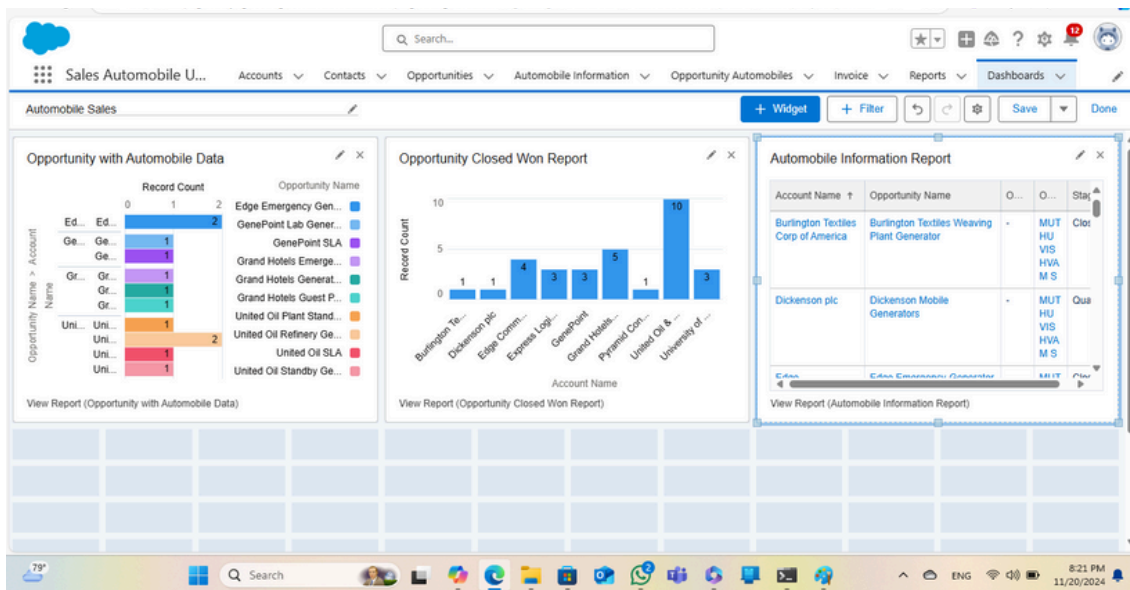


REPORTS	Report Name	Description	Folder	Created By	Created On	Subscribed
Recent	Opportunity Closed Won Report		Public Reports	MUTHU VISHVAM S	20/11/2024, 8:01 pm	<input type="checkbox"/>
Created by Me	Opportunity with Automobile Data		Public Reports	MUTHU VISHVAM S	20/11/2024, 8:13 pm	<input type="checkbox"/>
Private Reports	Automobile Information Report		Public Reports	MUTHU VISHVAM S	20/11/2024, 5:26 pm	<input type="checkbox"/>
Public Reports						
All Reports						
FOLDERS						
All Folders						
Created by Me						
Shared with Me						
FAVORITES						
All Favorites						

➤ Dashboard:

Dashboards help you visually understand changing business conditions so you can make decisions based on the real-time data you've gathered with reports. Use dashboards to help users identify trends, sort out quantities, and measure the impact of their activities.

The Created Dashboard:



5. Testing and Validation for the Automobile Sales CRM Project:

In the Automobile Sales CRM project, testing and validation play a crucial role in ensuring the reliability, accuracy, and overall functionality of the application. The project involves multiple components such as Apex classes, Apex triggers, Lightning Web Components (LWC), and custom objects. For Apex classes and triggers, unit tests are designed to verify business logic, such as calculating automobile prices, updating opportunity quantities, and handling complex workflows. These unit tests simulate real-world scenarios by inserting mock data and validating outcomes with assertions. Additionally, tests cover edge cases, such as handling invalid or missing inputs, ensuring that the system behaves as expected under all conditions, and confirming that bulk processing does not exceed Salesforce governor limits. Apex test classes ensure that the code is fully covered (with at least 75% test coverage) and compliant with Salesforce's deployment requirements.

For the user interface (UI), testing focuses on Lightning Web Components (LWC) to ensure that the system is intuitive, responsive, and user-friendly. The UI testing validates the correct functionality of components such as automobile search, invoice generation, and opportunity management. Tests are implemented to simulate user interactions, such as filtering automobiles by make, adding items to opportunities, and updating quantities. Using Jest for testing LWC, developers verify that components respond correctly to user input, trigger appropriate events, and dynamically update the interface based on real-time data. These tests ensure that the final product offers a seamless user experience, with minimal bugs or UI inconsistencies, and that all components interact smoothly with Salesforce's backend systems.

6. Key Scenarios Addressed by Salesforce in the Implementation Project:

1. Managing Automobile Inventory

Scenario: The business needs to manage a dynamic inventory of automobiles with details such as make, model, price, stock availability, and other relevant attributes. The sales team needs a centralized system to view and update automobile information in real time.

Salesforce Solution:

- Custom Object for Automobile Inventory: A custom object is created to store automobile details (e.g., model, price, stock quantity).
- Lightning Web Component (LWC) is built to enable the sales team to view and search for automobiles in real time.
- The system can auto-update stock quantities based on sales or returns through Apex triggers.

2.Customer Relationship Management (CRM)

Scenario

: The business needs to track detailed customer information, including contact details, previous interactions, and automobile purchases. The system should also allow for effective follow-ups and communication with leads, prospects, and customers.

Salesforce Solution:

- **Contact and Account Management:** Salesforce's Contact and Account objects are customized to track customer information and interactions related to automobile purchases.
- **Lead Management:** Leads are captured through forms or imports and converted into Opportunities when ready for further engagement.
- **Task and Event Tracking:** Salesforce's Task and Event functionalities are used to create reminders, track follow-up calls, and schedule meetings with customers.

7.conclusion:

The Automobile Sales CRM project has successfully implemented a comprehensive solution that streamlines key business processes, enhances sales operations, and improves customer relationship management. By leveraging Salesforce's powerful features, including custom objects, Apex triggers, Lightning Web Components (LWC), and automation tools like Process Builder and Flows, the project has effectively addressed critical use cases such as managing automobile inventory, tracking sales opportunities, generating invoices, and automating workflows. The system enables real-time data updates, detailed reporting and analytics, and seamless integration of sales and customer data, ultimately driving increased sales efficiency and better customer experiences. Through this implementation, the business now has a scalable, flexible CRM solution that supports both operational needs and strategic growth.