

Sales Automobile Using Salesforce CRM

Team Members:

Naan Mudhalvan ID:

BHARANIDHARAN A : 2B1E45E457ADE2A89A1488431B43342C

GOWTHAM K : 16ECA332D8C58E9B7084E9B9D33B6344

SANTHANARAJ S : D54A378E7F671458B18460C1089C725C

PARISUGAVELAN S : 1DD33415CD1074CB82E4BEF24D129F06

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 realtime 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).



Q Search Setup

☆

+

🔗

?

⚙️


🔔

👤

Setup

Home

Object Manager ▾

 **Object Manager**
2 Items, Sorted by Label


Q autom

Schema Builder


Create ▾

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED	
Automobile Information	Automobile_Information__c	Custom Object		16/11/2024	✓	▾
Opportunity Automobile	Opportunity_Automobile__c	Custom Object		16/11/2024	✓	▾

Q Search



ENG IN 14:29 18-11-2024



Q Search Setup

☆

+

🔗

?

⚙️


🔔

👤

Setup

Home

Object Manager ▾

 **Object Manager**
5 Items, Sorted by Label

Q invoice

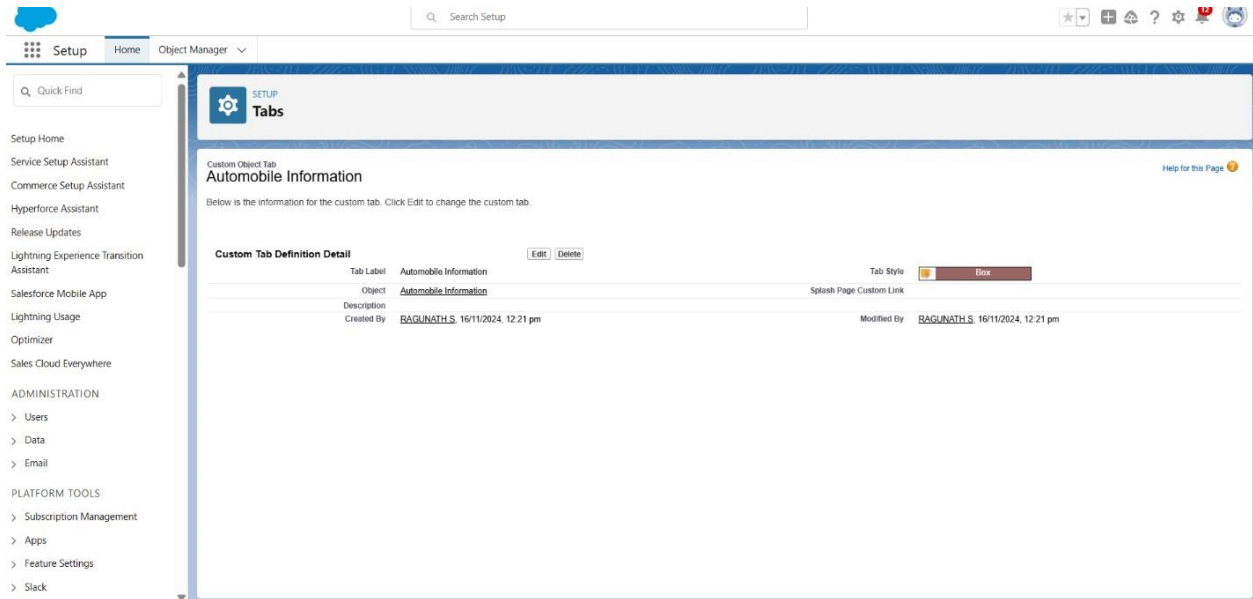
Schema Builder

Create ▾

LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED	
Credit Memo Invoice Application	CreditMemoInvApplication	Standard Object				
Invoice	Invoice__c	Custom Object		16/11/2024	✓	▾
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.



○ Lightning App

- Create a Custom Lightning App that integrates the following components:
 - Opportunity Records
 - Automobile Information records
 - Invoices related to OpportunitiesThe app should include:
 - Navigation to all relevant objects (Opportunities, Automobiles, Invoices).

A dashboard to visualize Total Sales, Invoices due, Opportunity stage.

○ Creating Fields and Relationships:

The screenshot shows the Salesforce Setup interface for the 'Opportunity Automobile' object. The left sidebar contains a navigation menu with options like 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, Object Access, and Triggers. The 'Fields & Relationships' section is active, displaying a table of 8 fields. The table has columns for FIELD LABEL, FIELD NAME, DATA TYPE, CONTROLLING FIELD, and INDEXED. The fields listed are: Automobile (Lookup(Automobile Information)), Created By (Lookup(User)), Last Modified By (Lookup(User)), Opportunity (Master-Detail(Opportunity)), Opportunity Automobile Id (Auto Number), Quantity (Number(18, 0)), Total Price (Formula (Currency)), and Unit Price (Formula (Currency)).

SETUP > OBJECT MANAGER
Opportunity Automobile

Details
Fields & Relationships 8 Items. Sorted by Field Label

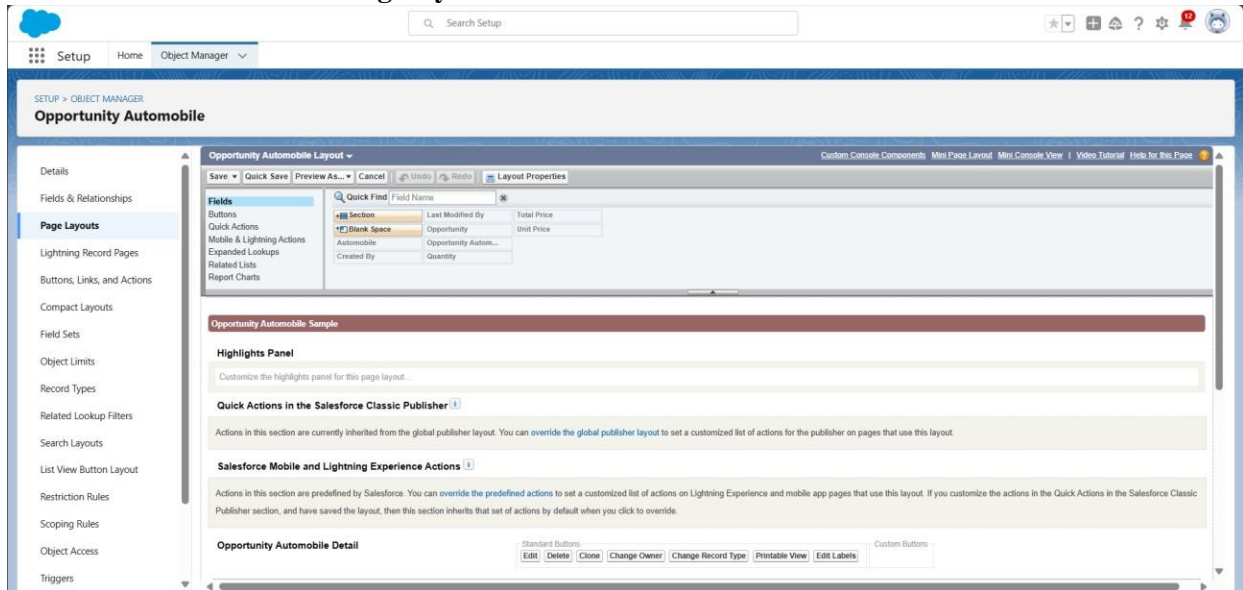
Quick Find New Deleted Fields Field Dependencies Set History Tracking

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.

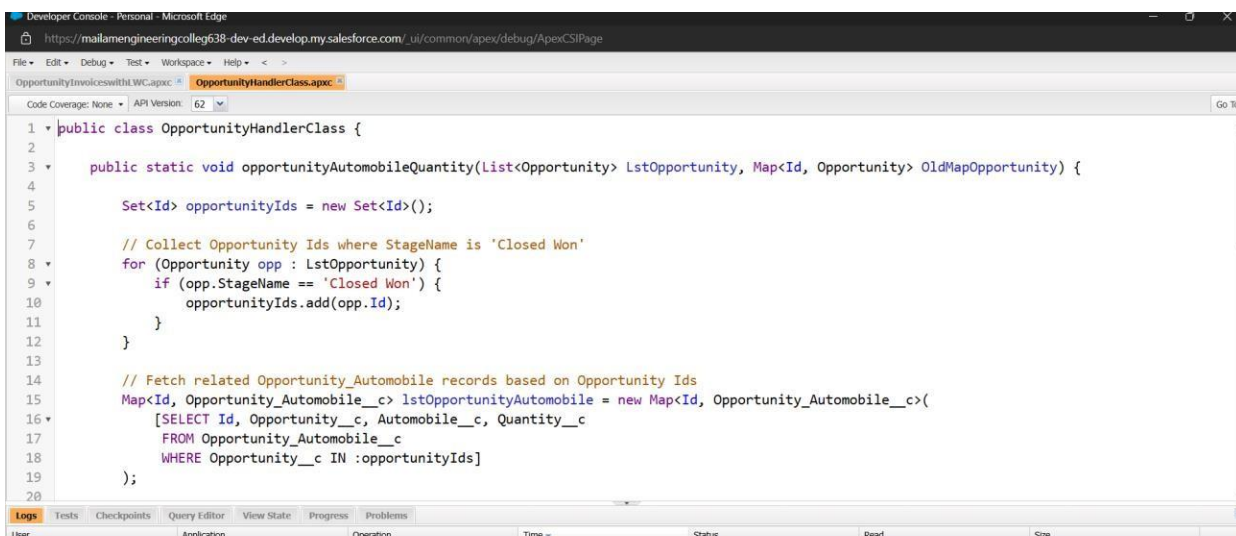
Edit the Page layout for Automobiles Information



○ 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



Creating OpportunityAutomobileHandlerClass

```

1 public class OpportunityAutomobileHandler {
2
3     public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c> lstOpportunityAutomobile){
4
5         Set<Id> AutomobileIds = new Set<Id>();
6
7         For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
8
9             if(oppAutomobile.Automobile__c != null){
10
11                 AutomobileIds.add(oppAutomobile.Automobile__c);
12
13             }
14
15         }
16
17         Map<Id, Automobile_Information__c> lstAutomobileInformation = new map<Id, Automobile_Information__c>([SELECT Id, CreatedById, Quantity__c, Pr
18
19         For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
20

```

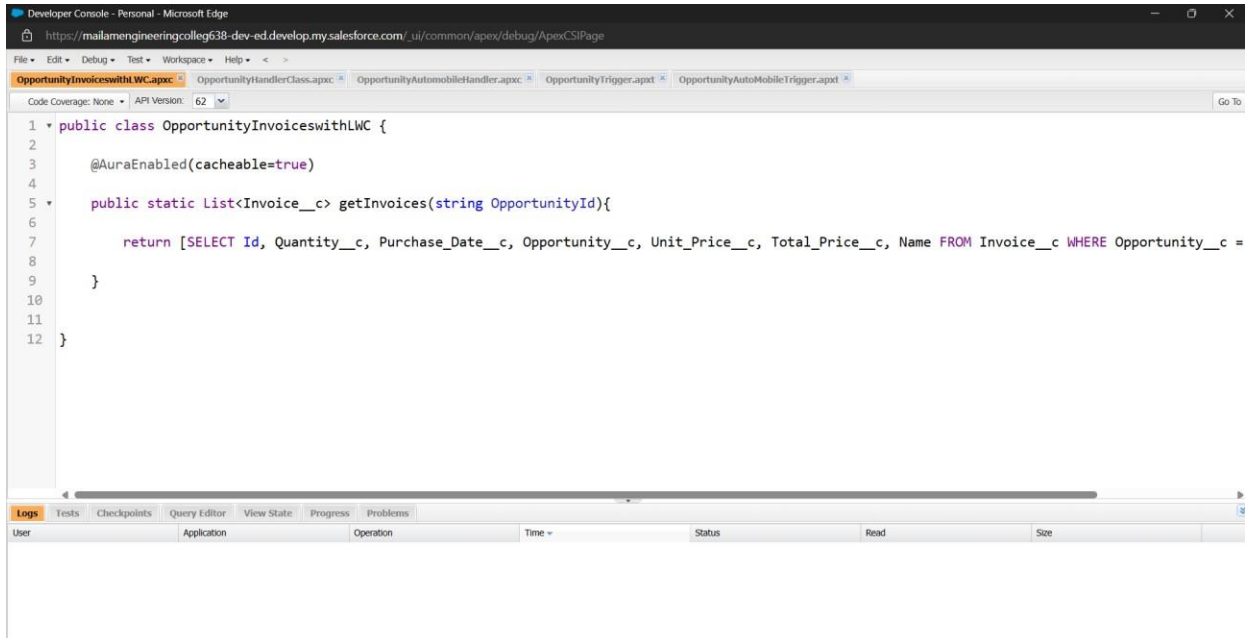
Creating OpportunityAutoMobileTrigger

```

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 }

```

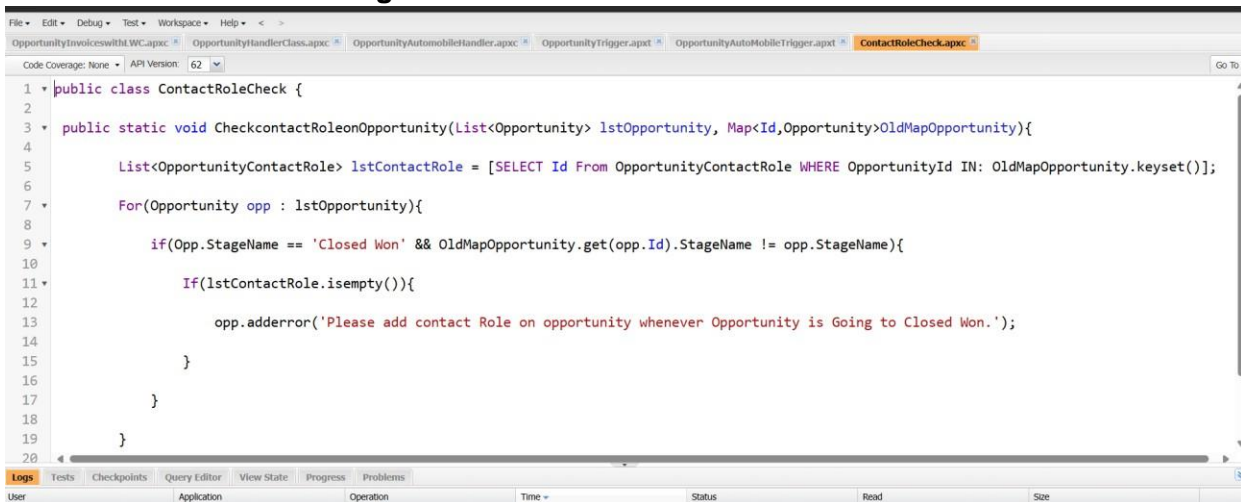
Creating InvoiceCreation class



```

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 }
    
```

Creating ContactRoleCheck class

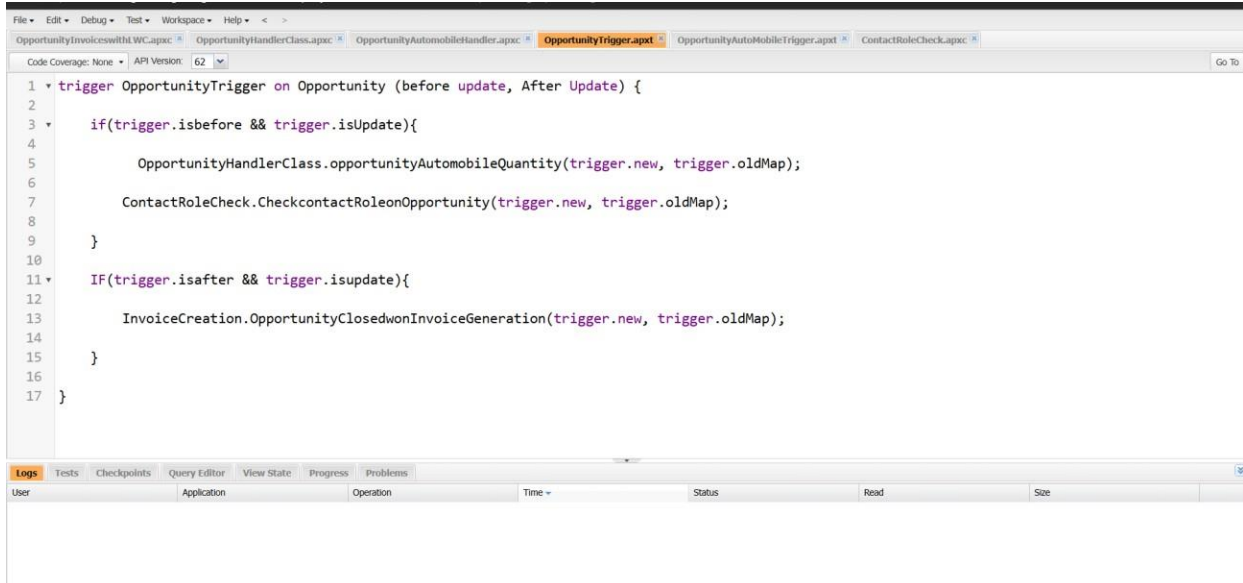


```

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 }

```

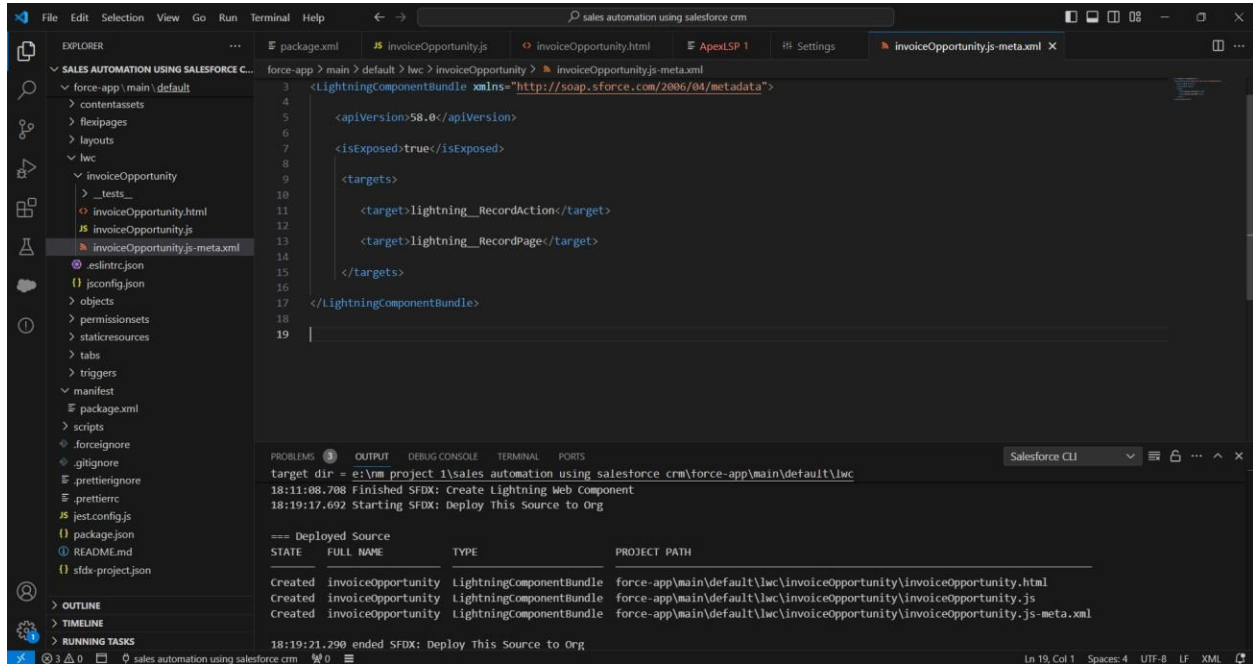
o LWC Component:

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 Lightning Web Component

The lwc component name is InvoiceOpportunity.

XML File



The screenshot shows the VS Code interface with the following details:

- Explorer:** The file tree on the left shows the project structure. The file `invoiceOpportunity.js-meta.xml` is selected.
- Editor:** The main editor displays the XML content of `invoiceOpportunity.js-meta.xml`:


```

1  <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
2
3  <apiVersion>58.0</apiVersion>
4
5  <isExposed>true</isExposed>
6
7  <targets>
8
9    <target>lightning__RecordAction</target>
10
11    <target>lightning__RecordPage</target>
12
13  </targets>
14
15 </LightningComponentBundle>
16
17
18
19

```
- Terminal:** The bottom terminal shows the output of the SFDX command:

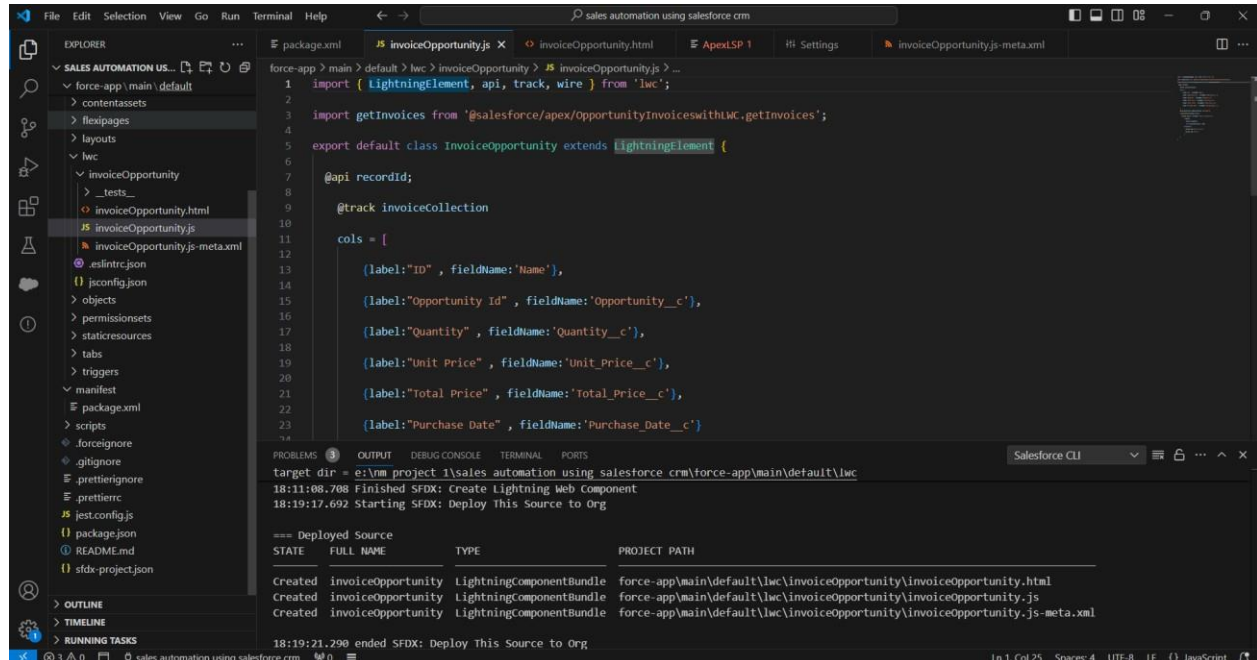

```

target dir = e:\vm project 1\sales automation using salesforce crm\force-app\main\default\lwc
18:11:08.708 Finished SFDX: Create Lightning Web Component
18:19:17.692 Starting SFDX: Deploy this Source to Org

== Deployed Source ==
STATE      FULL NAME                                TYPE                                PROJECT PATH
-----
Created    invoiceOpportunity                       LightningComponentBundle           force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.html
Created    invoiceOpportunity                       LightningComponentBundle           force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js
Created    invoiceOpportunity                       LightningComponentBundle           force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js-meta.xml
18:19:21.290 ended SFDX: Deploy this Source to Org

```

JS File



The image shows a VS Code editor window with the file explorer on the left displaying the project structure. The main editor area shows the `invoiceOpportunity.js` file. The code defines a Lightning component class `InvoiceOpportunity` that extends `LightningElement`. It includes a `@api` recordId and a `@track` invoiceCollection. The `cols` array defines the columns for the `lightning-datatable`.

```

1 import { LightningElement, api, track, wire } from 'lwc';
2
3 import getInvoices from '@salesforce/apex/OpportunityInvoicesWithWC.getInvoices';
4
5 export default class InvoiceOpportunity extends LightningElement {
6
7     @api recordId;
8
9     @track invoiceCollection
10
11     cols = [
12         {label:'ID', fieldName:'Name'},
13         {label:'Opportunity Id', fieldName:'Opportunity__c'},
14         {label:'Quantity', fieldName:'Quantity__c'},
15         {label:'Unit Price', fieldName:'Unit_Price__c'},
16         {label:'Total Price', fieldName:'Total_Price__c'},
17         {label:'Purchase Date', fieldName:'Purchase_Date__c'}
18     ];
19 }

```

The terminal at the bottom shows the deployment process:

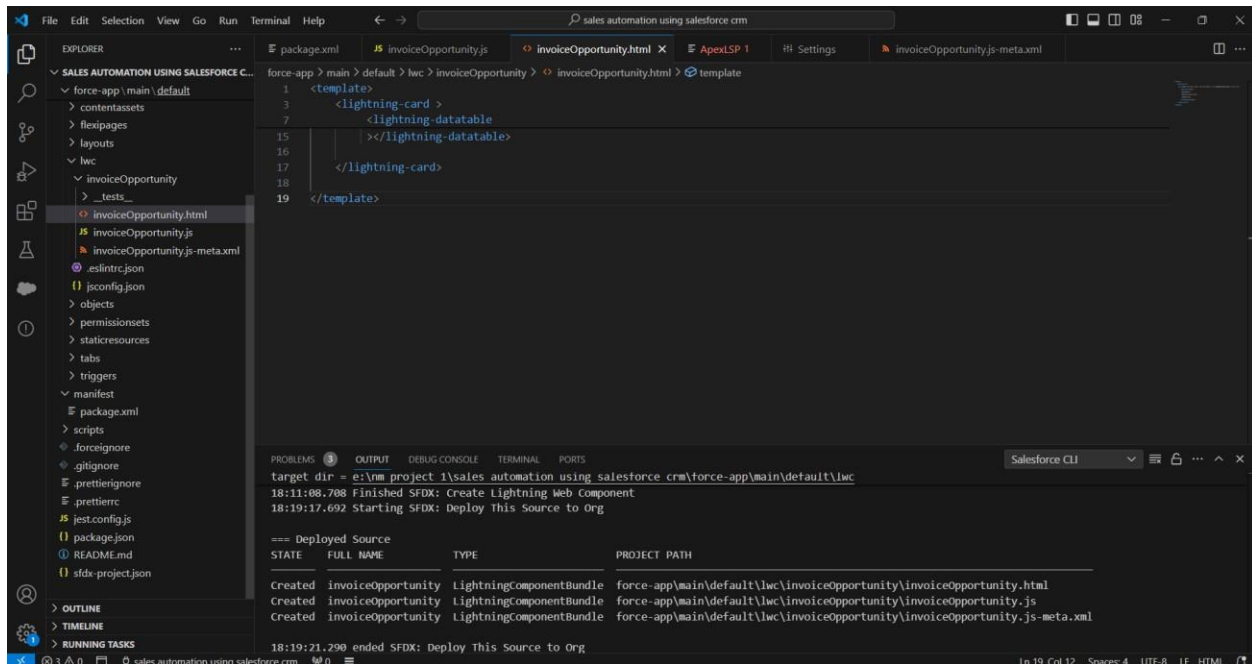
```

target dir = e:\nm project 1\sales automation using salesforce crm\force-app\main\default\lwc
18:11:08.708 Finished SFDX: Create Lightning Web Component
18:19:17.692 Starting SFDX: Deploy This Source to Org
18:19:21.290 ended SFDX: Deploy This Source to Org

```

STATE	FULL NAME	TYPE	PROJECT PATH
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.html
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js-meta.xml

HTML File



The image shows a VS Code editor window with the file explorer on the left displaying the project structure. The main editor area shows the `invoiceOpportunity.html` file. The code defines the HTML structure for the component, including a `template` with a `lightning-card` and a `lightning-datatable`.

```

1 <template>
2
3     <lightning-card>
4
5         <lightning-datatable>
6
7         </lightning-datatable>
8
9     </lightning-card>
10
11 </template>

```

The terminal at the bottom shows the deployment process:

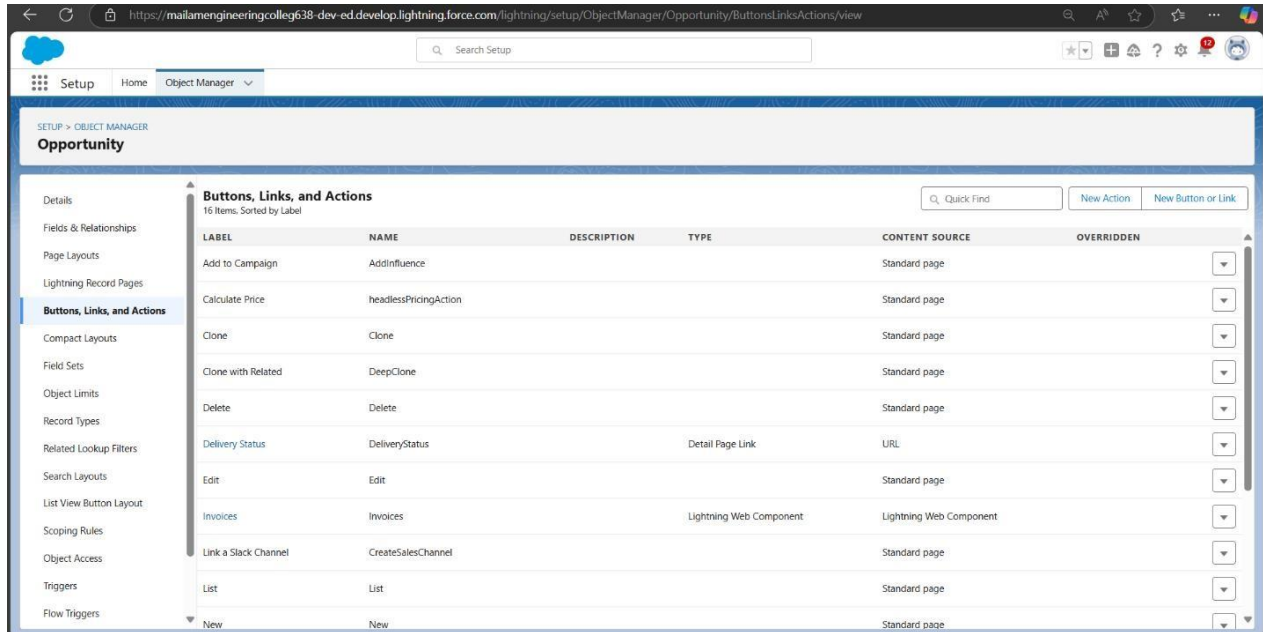
```

target dir = e:\nm project 1\sales automation using salesforce crm\force-app\main\default\lwc
18:11:08.708 Finished SFDX: Create Lightning Web Component
18:19:17.692 Starting SFDX: Deploy This Source to Org
18:19:21.290 ended SFDX: Deploy This Source to Org

```

STATE	FULL NAME	TYPE	PROJECT PATH
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.html
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js
Created	invoiceOpportunity	LightningComponentBundle	force-app\main\default\lwc\invoiceOpportunity\invoiceOpportunity.js-meta.xml

Create Button to add an opportunity

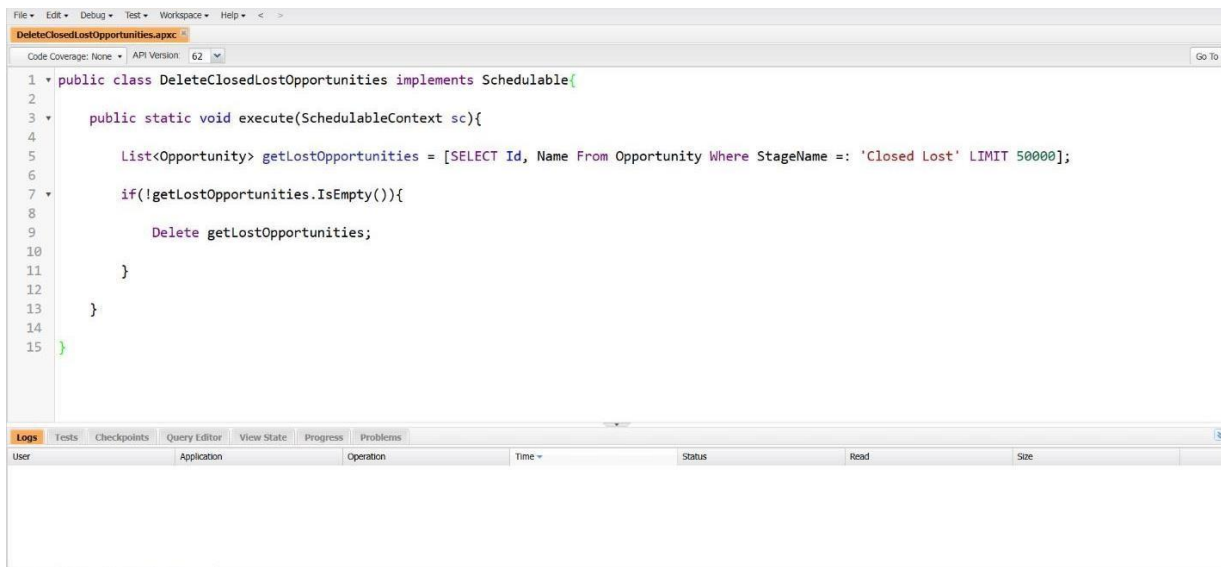


The screenshot shows the Salesforce Setup interface for the Opportunity object. The left sidebar lists various setup options, with 'Buttons, Links, and Actions' selected. The main area displays a table of existing buttons and links.

LABEL	NAME	DESCRIPTION	TYPE	CONTENT SOURCE	OVERRIDDEN
Add to Campaign	AddInfluence			Standard page	
Calculate Price	headlessPricingAction			Standard page	
Clone	Clone			Standard page	
Clone with Related	DeepClone			Standard page	
Delete	Delete			Standard page	
Delivery Status	DeliveryStatus		Detail Page Link	URL	
Edit	Edit			Standard page	
Invoices	Invoices		Lightning Web Component	Lightning Web Component	
Link a Slack Channel	CreateSalesChannel			Standard page	
List	List			Standard page	
New	New			Standard page	

O Apex Schedulers:

Delete opportunity schedule class



```

1 public class DeleteClosedLostOpportunities implements Schedulable{
2
3     public static void execute(SchedulableContext sc){
4
5         List<Opportunity> getLostOpportunities = [SELECT Id, Name From Opportunity Where StageName =: 'Closed Lost' LIMIT 50000];
6
7         if(!getLostOpportunities.isEmpty()){
8
9             Delete getLostOpportunities;
10
11         }
12     }
13 }
14
15

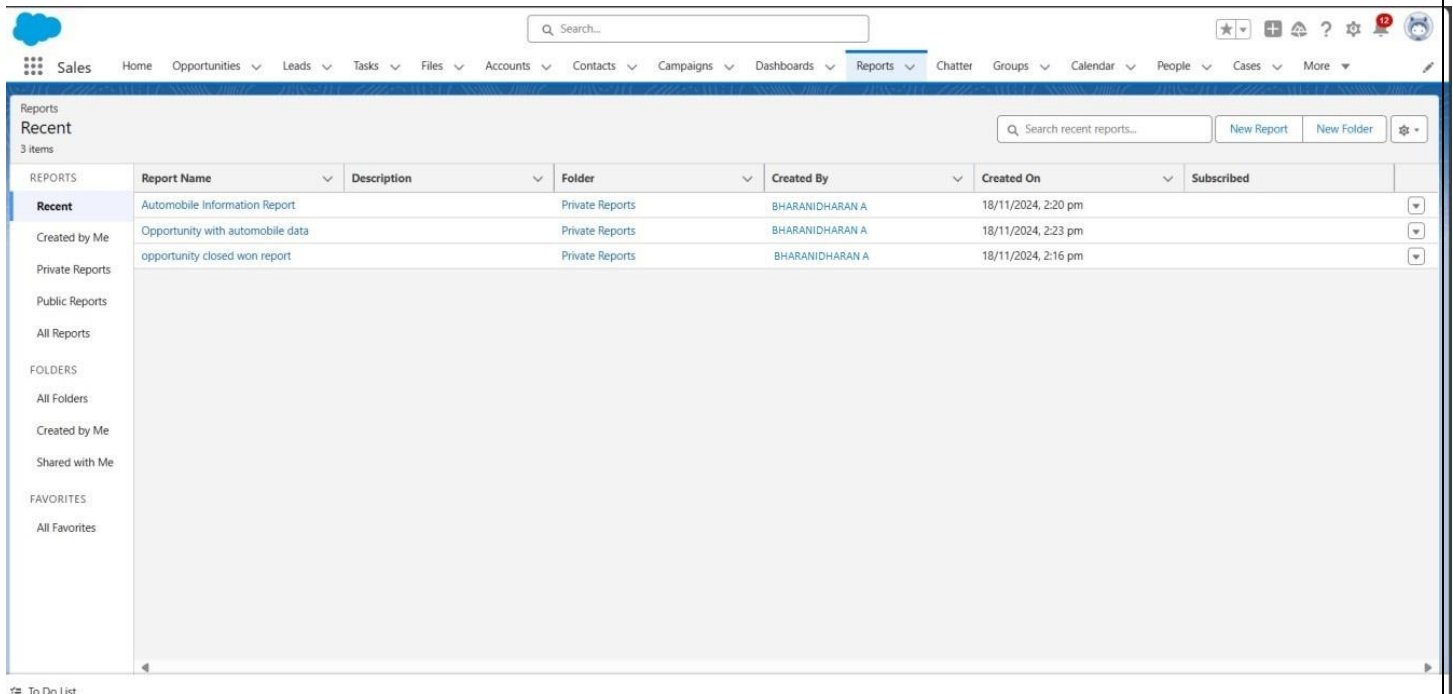
```

The screenshot shows an IDE window titled 'DeleteClosedLostOpportunities.apex'. The code defines a class that implements the Schedulable interface. The execute method queries for closed lost opportunities and deletes them. The bottom of the IDE shows a 'Logs' tab with columns for User, Application, Operation, Time, Status, Read, and Size.

○ 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.



The screenshot shows the Salesforce Reports interface. The top navigation bar includes a search bar and various icons. The left sidebar shows the 'Reports' section with a 'Recent' filter selected. The main content area displays a table of recent reports.

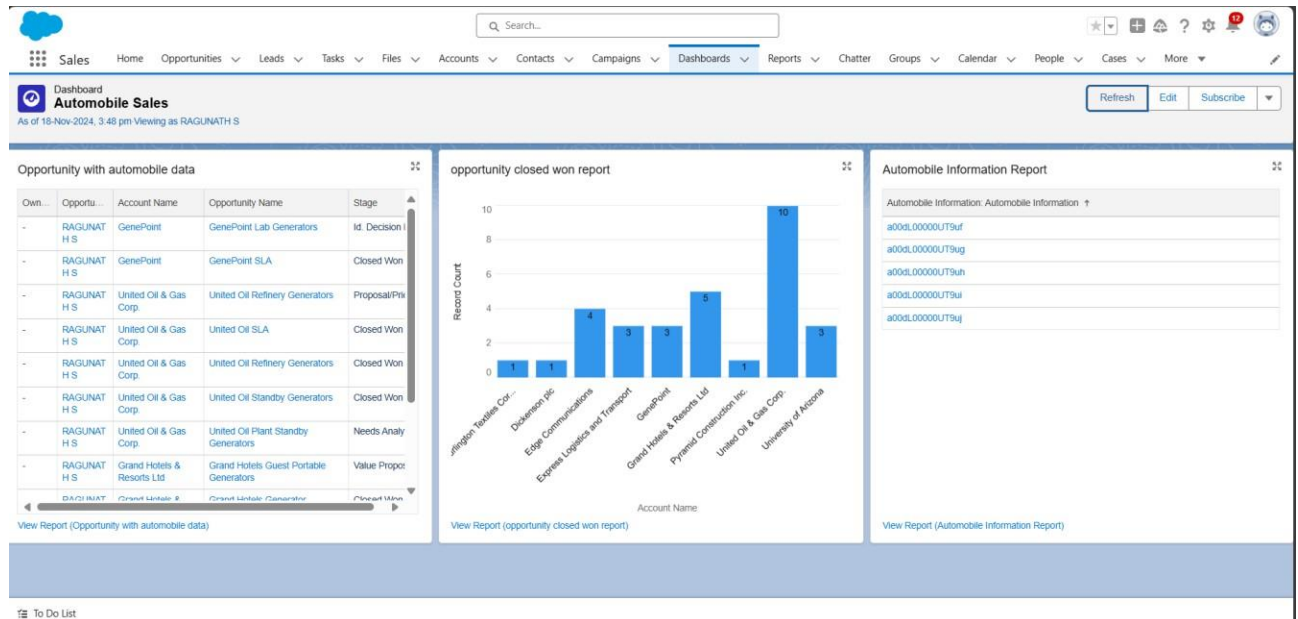
REPORTS	Report Name	Description	Folder	Created By	Created On	Subscribed
Recent	Automobile Information Report		Private Reports	BHARANIDHARAN A	18/11/2024, 2:20 pm	
Created by Me	Opportunity with automobile data		Private Reports	BHARANIDHARAN A	18/11/2024, 2:23 pm	
Private Reports	opportunity closed won report		Private Reports	BHARANIDHARAN A	18/11/2024, 2:16 pm	

The interface also includes a 'FOLDERS' section on the left sidebar with options like 'All Folders', 'Created by Me', and 'Shared with Me'. The bottom of the page shows a 'To Do List' icon.

○ 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 realtime 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.