

**SARASWATHY COLLEGE OF ENGINEERING AND TECHNOLOGY**

**OLLAKUR, TINDIVANAM - 604305**



**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**BACHELOR OF ENGINEERING**

**2024-2025**

**FIFTH SEMESTER**

**SALESFORCE DEVELOPER**

**TEAM MEMBERS :**

<b>K.LOGESHWARI</b>	<b>421822104028</b>
<b>G.KIRAN</b>	<b>421822104027</b>
<b>K.KEERTHANA</b>	<b>421822104026</b>
<b>P.KARANRAJ</b>	<b>421822104025</b>

**SARASWATHY COLLEGE OF ENGINEERING AND TECHNOLOGY**

**OLLAKUR,TINDIVANAM - 604305**



**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**BACHELOR OF ENGINEERING**

**2024-2025**

certified that this is a bonafide record of work done by

Name : Kiran.G

NM ID : 543026783AC0FA2A75E4FBB8BEE4898C

Universite Reg.no : 421822104027

Semester : 5

Branch : CSE

Year : 2024-2025

**Staff-in-Charge**

**Head of the Department**

Submitted for the\_\_\_\_\_

Practical Examination held on\_\_\_\_\_

**Internal Examiner**

**External Examiner**

# SALES AUTOMOBILE USING SALESFORCE CRM

## PROJECT VIEW:

This project is focused on creating a comprehensive Salesforce Automobile Information CRM, designed to address the challenges of managing and tracking automotive sales, inventory, and customer relationships in the automobile industry. The goal is to deliver a solution that leverages Salesforce CRM, cloud technologies, and automation to improve the management of customer interactions, streamline inventory tracking, and optimize sales processes. Through this project, we aim to enhance operational efficiency, customer experience, and data accuracy, while supporting the long-term goals of the automobile dealership or sales organization.

## OBJECTIVES:

### Business Goals:

- Streamline the management of customer data, vehicle inventory, and sales processes.
- Enhance customer engagement through personalized service and targeted marketing.
- Improve operational efficiency by automating routine tasks and workflows.

### Specific Outcomes:

- Develop an integrated system for tracking vehicle inventory, customer inquiries, sales leads, and follow-ups.
- Implement automated workflows for sales, service, and customer support processes.
- Enable detailed reporting and analytics for performance monitoring and decision-making.
- Ensure seamless integration with other systems like financial and service management tools.

### Salesforce Key Features and Concepts Utilized:

This section highlights the main Salesforce functionalities applied in this CRM system:

- **Salesforce Sales Cloud:** Used for managing leads, opportunities, and sales pipelines.
  - **Salesforce Service Cloud:** Provides customer service functionality, including case management and customer support.
  - **Custom Objects and Fields:** Created to track vehicle information (make, model, year, VIN, etc.) and manage inventory.
  - **Apex Triggers and Classes:** Used to automate processes such as updating inventory status, sending notifications, or creating follow-up tasks for sales representatives.
- Reports and Dashboards:** For real-time insights into sales performance, customer activity, and inventory levels.
- **Lightning Web Components (LWC):** Custom user interfaces for improved user experience and interaction with the CRM.

## Detailed Steps to Solution Design:

### Data Models:

Define custom objects like Vehicle Inventory, Customer, and Sales Opportunity to store all relevant data.

Establish relationships between objects such as Customer to Sales Opportunity and Vehicle Inventory to Sales Opportunity.

Design custom fields in each object (e.g., Vehicle Color, Model Year, Price, Customer Preferences).

### User Interface Design:

- Create custom Lightning pages for sales reps, service agents, and management to easily access and update information.
- Utilize Lightning App Builder to create a user-friendly dashboard that displays key metrics such as active sales leads, available inventory, and customer interactions.

### Business Logic:

- Define automation rules for sales and service processes (e.g., automating lead assignment, setting up reminders for follow-up, and creating tasks based on specific triggers).
- Use Flow and Process Builder to streamline sales workflows, such as sending automated emails to customers after a purchase or sending alerts when inventory is running low.

## AUTOMOBILE INFORMATION OBJECT:

### What Is an Object?

Salesforce objects are database tables that permit you to store data that is specific to an organization. What are the types of Salesforce objects

### Salesforce objects are of two types:

1. **Standard Objects:** Standard objects are the kind of objects that are provided by salesforce.com such as users, contracts, reports, dashboards, etc.
2. **Custom Objects:** Custom objects are those objects that are created by users. They supply information that is unique and essential to their organization. They are the heart of any application and provide a structure for sharing data.

## Create Automobile Information Object

1. Download and open [this spreadsheet](#), save it as AutomobileInformation.csv.

Web Tabs are custom tabs that display web content or applications embedded in the salesforce.com window. Web tabs make it easier for your users to quickly access content and applications they frequently use without leaving

the salesforce.com application.

### Visualforce Tabs

Visualforce Tabs are custom tabs that display a Visualforce page. Visualforce tabs look and behave like standard salesforce.com tabs such as accounts, contacts, and opportunities.

### Lightning Component Tabs

Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app.

### Lightning Page Tabs

Lightning Page Tabs let you add Lightning Pages to the mobile app navigation menu.

Lightning Page tabs don't work like other custom tabs. Once created, they don't show up on the All Tabs page when you click the Plus icon that appears to the right of your current tabs. Lightning Page tabs also don't show up in the Available Tabs list when you customize the tabs for your apps.

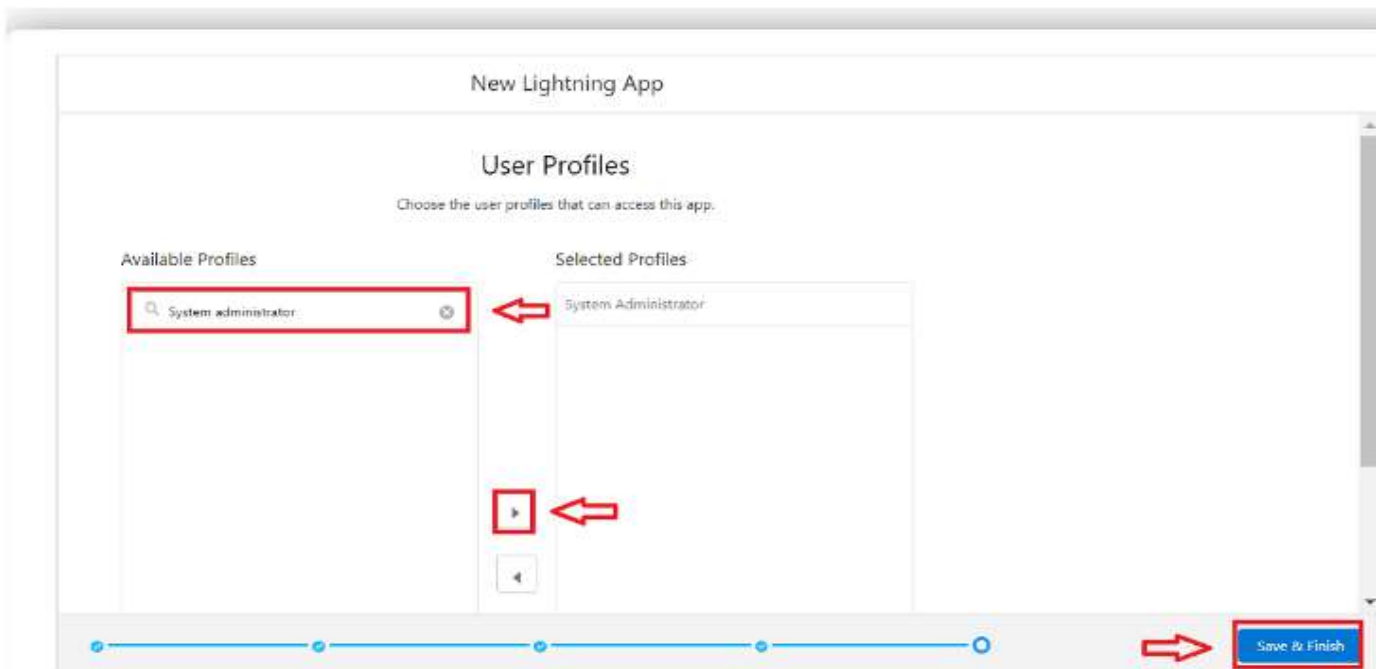
## Creating a Custom Tab:



## LIGHTNING APPS:

An app is a collection of items that work together to serve a particular function. In Lightning Experience, Lightning apps gives users access to sets of objects, tabs, and other items all in one convenient bund navigation bar. Lightning apps let you brand your apps with a custom color and logo. You can even include a utility bar and Lightning page tabs in your Lightning app. Members of your org can work more efficiently by easily switching between apps.

## Create a Lightning App:



1. Fill the app name in app details and branding as follow
  - a. App Name :Sales Automobile Using Salesforce CRM
  - b. Developer Name : this will auto populated
  - c. Description : Give a meaningful description
  - d. Image : optional (if you want to give any image you can otherwise not mandatory)
  - e. Primary color hex value : keep this default

### STEPS:

Search the items in the search bar(Account,Contact ,Opportunities,Automobile Information,Opportunity Automobile,Invoice, Reports, Dashboard) from the search bar and move it using the arrow button ? Next.  
 Note: select asset the custom object which we have created in the previous activity.

Search profiles (System administrator) in the search bar >> click on the arrow button >> save & finish.

## Fields & Relationships

When we talk about Salesforce, Fields represent the data stored in the columns of a relational database. It can hold any valuable information that you require for a specific object. Hence, the overall searching, deletion, and editing of the records become simpler and quicker.

### Types of Fields

1. Standard Fields
2. Custom Fields

As the name suggests, the Standard Fields are the predefined fields in Salesforce that perform a standard task. The main point is that you can't simply delete a Standard Field until it is a non-required standard field. Otherwise, users have the option to delete them at any point from the application freely. Moreover, we have some fields that you will find common in every Salesforce application. They are,

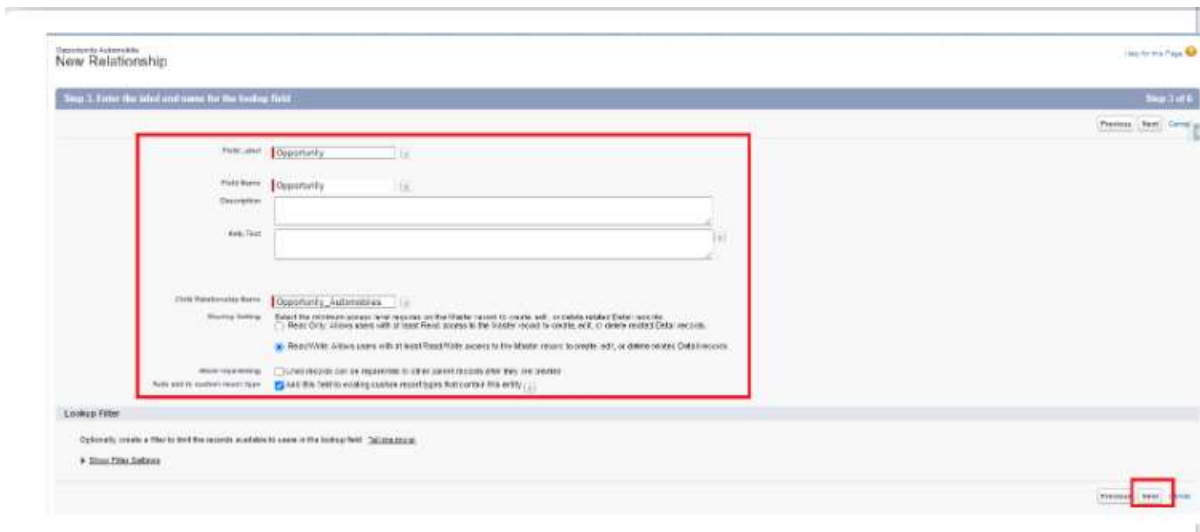
- >> Created By
- >> Owner
- >> Last Modified
- >> Field Made During object Creation

### Custom Fields:

On the other side of the coin, Custom Fields are highly flexible, and users can change them according to requirements. Moreover, each organizer or company can use them if necessary. It means you need not always include them in the records, unlike Standard fields. Hence, the final decision depends on the user, and he can add/remove Custom Fields of any given form.

## Creating Opportunity Master Detail Relationship Field in Opportunity

### AutoMobile Object:

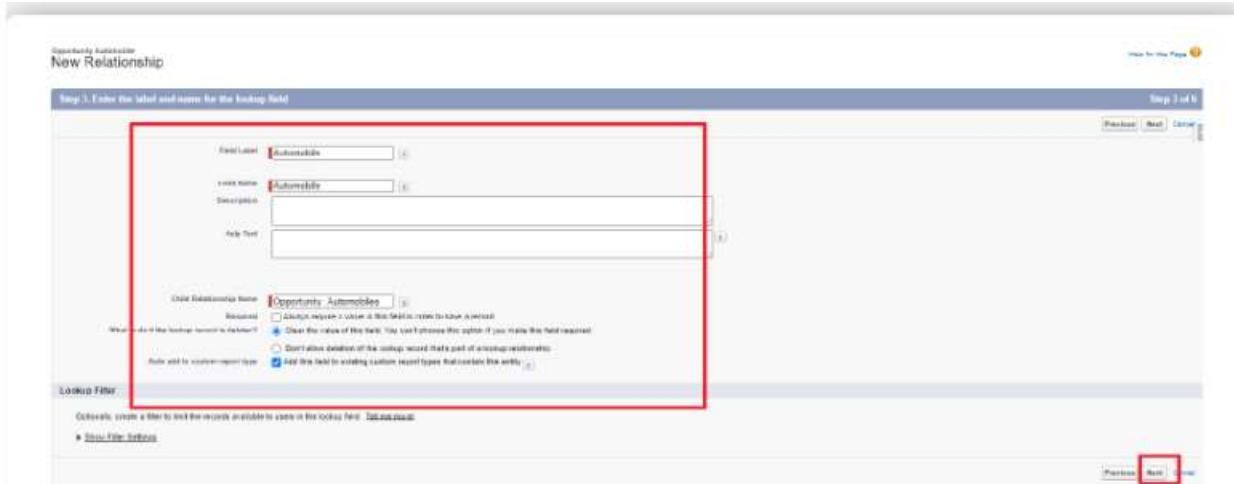


The screenshot displays the 'New Relationship' configuration page in Salesforce. The 'Master Label' and 'Master Name' fields are both set to 'Opportunity'. The 'Description' and 'Web Text' fields are empty. The 'Master Relationship Name' is 'Opportunity-AutoMobile'. Under 'Sharing Settings', the 'Read/Write' checkbox is selected, indicating that users with at least Read/Write access to the Master record can create, edit, or delete related Detail records. The 'Lookup Filter' section at the bottom allows for creating a filter to limit records available in the lookup field, with a 'Show Filter Settings' link.

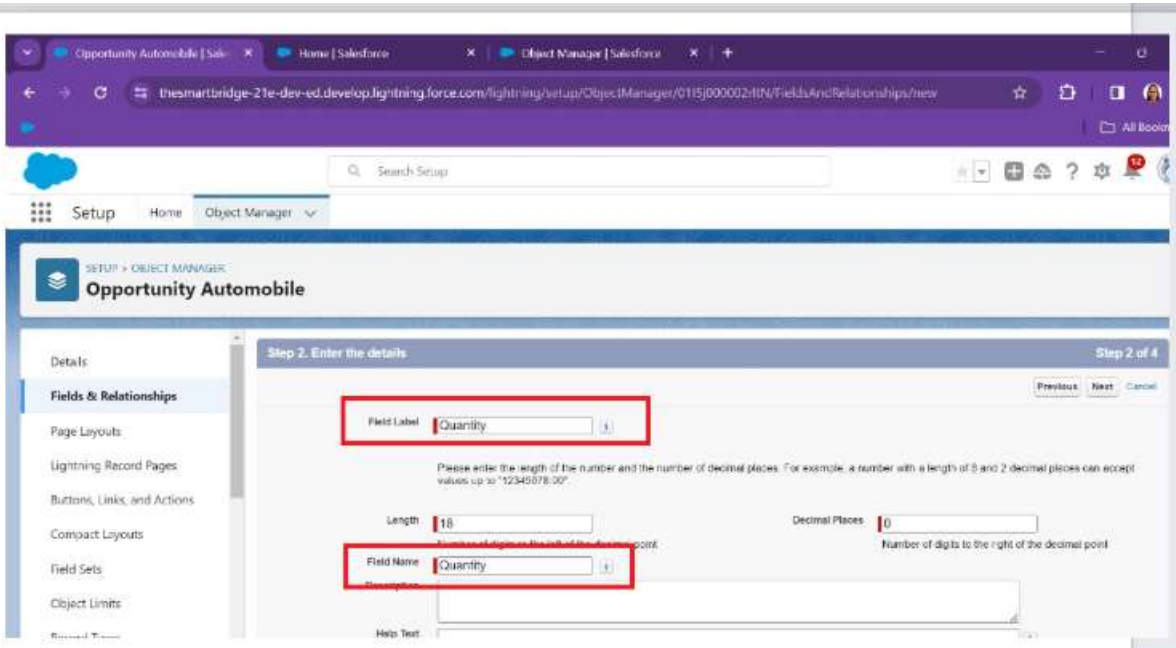


## Creating the AutoMobile Information Lookup Field in Opportunity

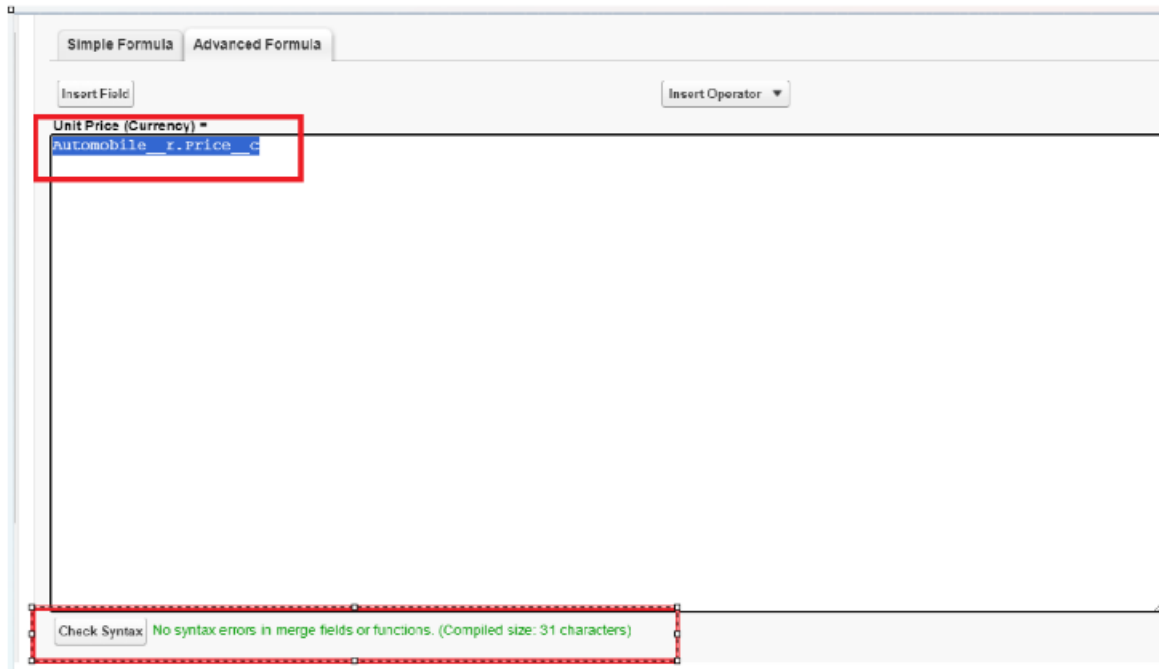
### Automobile Object:



## Creating Quantity Number Field in Opportunity Automobile Object




## Creating Formula Field in Opportunity Automobile Object



The screenshot shows the Salesforce Formula Editor interface. The 'Simple Formula' tab is selected. The formula bar contains the text 'Unit Price (Currency) =' followed by a red box highlighting the field 'Automobile r.Price\_c'. Below the formula bar, a red box highlights the 'Check Syntax' button and the message 'No syntax errors in merge fields or functions. (Compiled size: 31 characters)'.

## Updating field in Invoice Object



The screenshot shows the Salesforce Setup - Object Manager - Invoice Field page. The 'Fields & Relationships' tab is selected. The 'Record Name' field is highlighted with a red box and contains the text 'Invoice ID'. Below it, the 'Data Type' is set to 'Auto Number'. The 'Display Format' field is highlighted with a red box and contains the text '1-0000'. The 'Starting Number' field is highlighted with a red box and contains the text '1'. To the right of the form, there is a table titled 'Recent Accounts'.

Account Name	City
Acme	New York
Global Media	Toronto
salesforce.com	San Francisco

## Creating Remaining Fields in Objects

s.no	Object name	Fields
		Field Name
		Opportunity
1	Invoice	Data type
		Master Detail relationship Object : Opportunity

## Page Layouts:

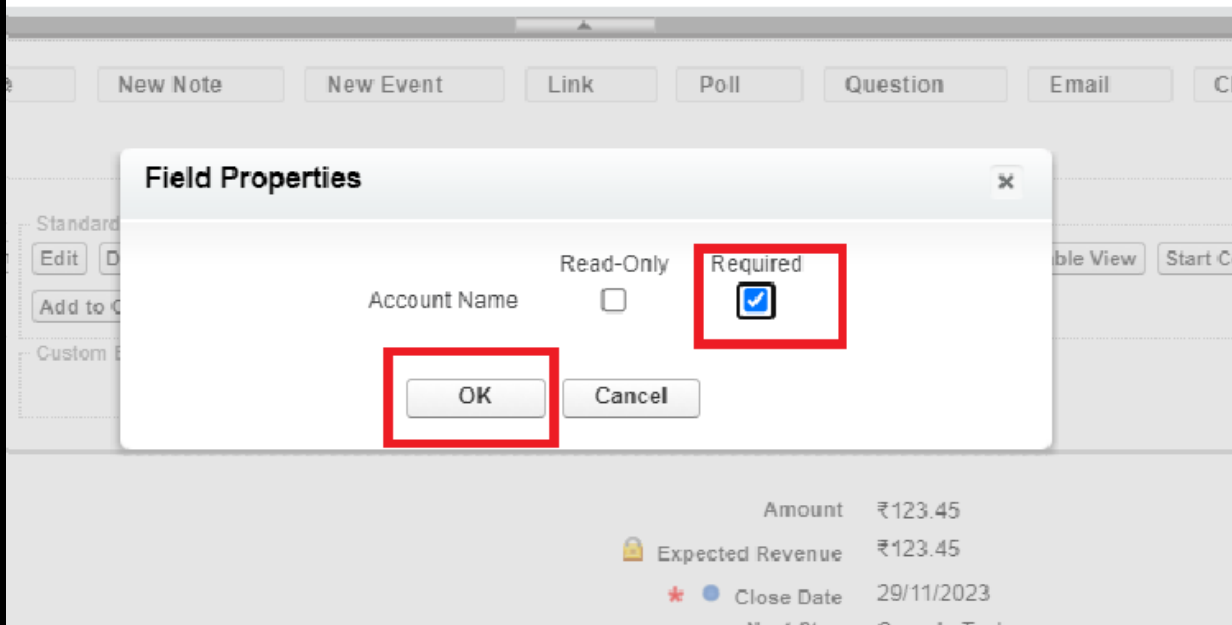
Page Layout in Salesforce allows us to customize the design and organize detail and edit pages of records in Salesforce. Page layouts can be used to control the appearance of fields, related lists, and custom links on standard and custom objects' detail and edit pages.

## Edit the Page layout for Opportunity Object

Step 1: Go to Setup >> Click on Object Manager >> On the search bar, select Opportunity Layout. You can notice Page Layouts on the left panel

Step 2: Click on Page Layouts, Click on 'Opportunity Layouts'.

Step 3: In the Opportunity Detail Section, you can see various fields. Go on Account And Click on that Properties icon of Account name Field.



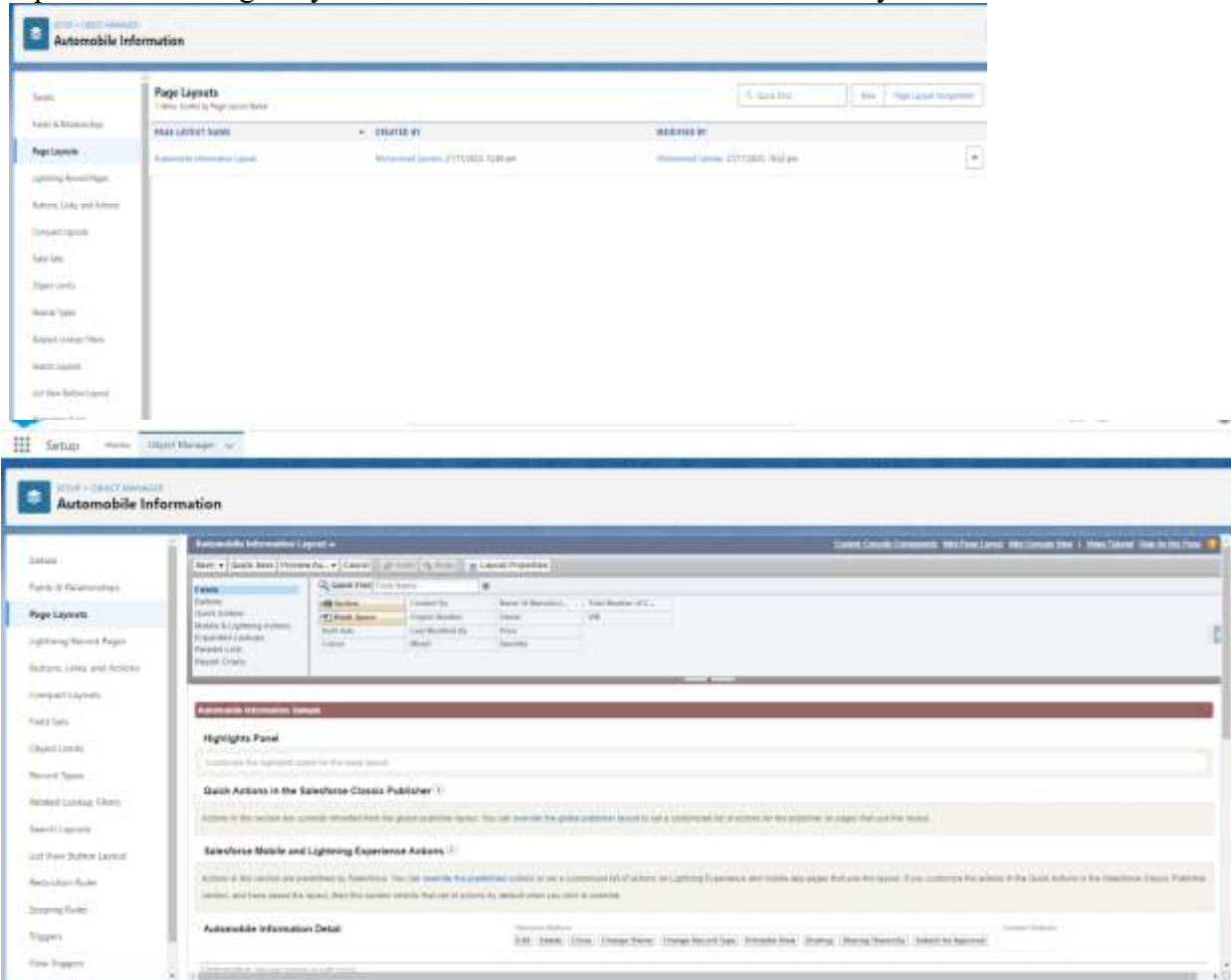
Step 4: check the Required box for Account name and click on Ok.

Step 5: Click on Save.

## Edit the Page layout for Automobiles Information

Step 1: Go to Setup >> Click on Object Manager >> On the search bar, select Automobile Information. You can notice Page Layouts on the left panel

Step 2: Click on Page Layouts. Click on 'Automobile Information Layout'.

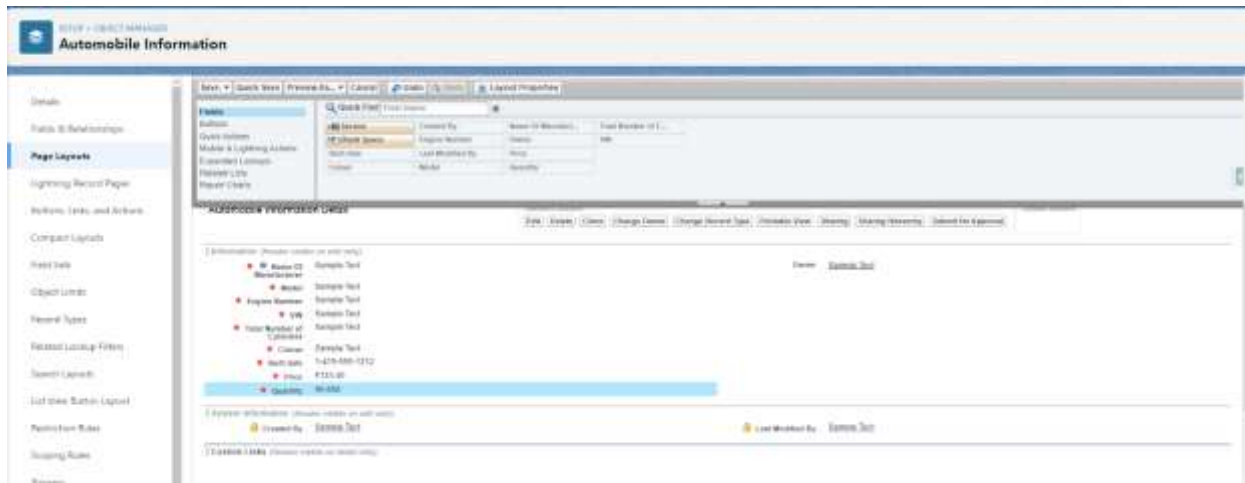


The screenshot shows the Salesforce Setup interface for the 'Automobile Information' object. The left sidebar lists various setup options, with 'Page Layouts' selected. The main area displays the 'Automobile Information Layout' with a table of fields and their properties.

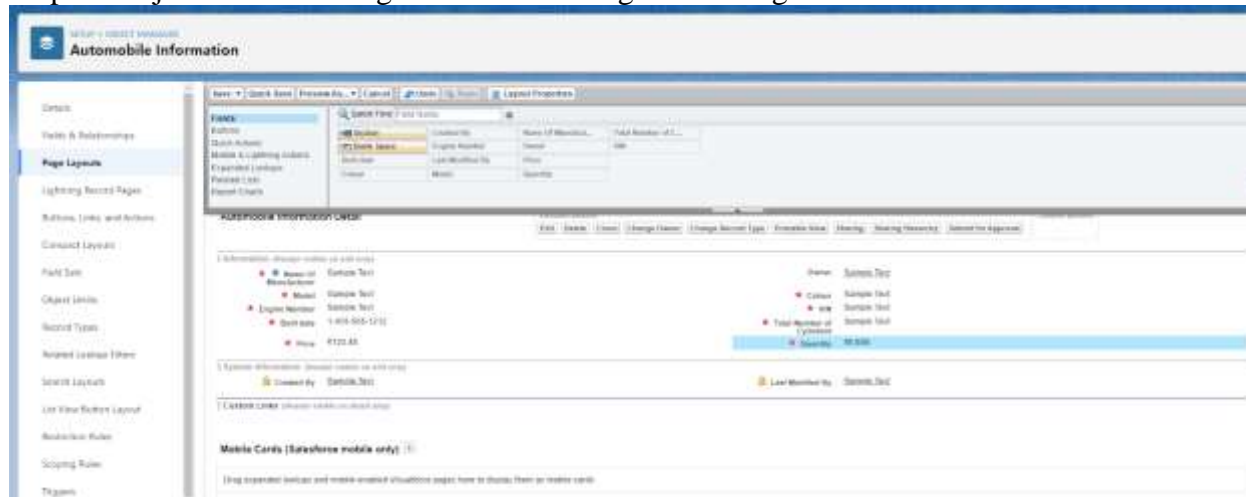
Field	Field Type	Field Label	Field Name	Field Type	Field Label	Field Name	Field Type	Field Label	Field Name
Account	Lookup	Account	Account	Text	Account	Account	Text	Account	Account
Quick Action	Text	Quick Action	Quick Action	Text	Quick Action	Quick Action	Text	Quick Action	Quick Action
Lightning Experience	Text	Lightning Experience	Lightning Experience	Text	Lightning Experience	Lightning Experience	Text	Lightning Experience	Lightning Experience
Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts
Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts
Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts	Text	Predefined Layouts	Predefined Layouts

The screenshot also shows the 'Automobile Information Layout' with a table of fields and their properties. The table includes columns for Field, Field Type, Field Label, Field Name, Field Type, Field Label, Field Name, Field Type, Field Label, and Field Name. The table lists various fields such as Account, Quick Action, Lightning Experience, Predefined Layouts, and Predefined Layouts.

Step 3: Just Go for each one field of Automobile Information Object, Click on Gear Icon and mark as Required just as Done for Above Account Object. After required is done it will show the red color as given in below image.



Step 4 : Adjust the Fields as given below for A good looking view.



Step 5 : Click on Save.

## Apex Trigger

Apex can be invoked by using triggers. Apex triggers enable you to perform custom actions before or after changes to Salesforce records, such as insertions, updates, or deletions.

A trigger is Apex code that executes before or after the following types of operations:

- insert
- update
- delete
- merge
- upsert
- undelete

For example, you can have a trigger run before an object's records are inserted into the database, after records have been deleted, or even after a record is restored from the Recycle Bin.

You can define triggers for top-level standard objects that support triggers, such as a Contact or an Account, some standard child objects, such as a CaseComment, and custom objects. To define a trigger, from the object management settings for the object whose triggers you want to access, go to Triggers.

There are primarily two types of Apex Triggers:

**Before Trigger:** This type of trigger in Salesforce is used either to update or validate the values of a record before they can be saved into the database. So, basically, the before trigger validates the record first and then saves it. Some criteria or code can be set to check data before it gets ready to be inserted into the database.

**After Trigger:** This type of trigger in Salesforce is used to access the field values set by the system and affect any change in the record. In other words, the after trigger makes changes to the value from the data inserted in some other record.

## Opportunity Automobile quantity

### Code:

```
public class OpportunityHandlerClass {

    public static void opportunityAutomobileQuantity(List<Opportunity> LstOpportunity, Map<Id,Opportunity>
    OldMapOpportunity){
        set<Id> opportunityIds = new set<Id>();
        for(Opportunity opp : LstOpportunity){
            if(opp.StageName == 'Closed Won' ){
                opportunityIds.add(opp.Id);
            }
        }
        Map<Id,Opportunity_Automobile__c> lstOpportunityAutomobile =new
        Map<Id,Opportunity_Automobile__c>([SELECT Id, Opportunity__c, Automobile__c, Quantity__c,
        Unit_Price__c, Total_Price__c FROM Opportunity_Automobile__c Where Opportunity__c IN:
        opportunityIds]);

        set<Id> AutoInformationIds = new set<Id>();
        for(Opportunity_Automobile__c OppAuto: lstOpportunityAutomobile.values()){
            if(OppAuto.Automobile__c != null){
                AutoInformationIds.add(OppAuto.Automobile__c);
            }
        }
        List<Automobile_Information__c> lstAutomobileInfomation = new List<Automobile_Information__c>();
        Map<Id,Automobile_Information__c> MapAutomobileInformation = New
        Map<Id,Automobile_Information__c>([SELECT Quantity__c, Price__c, Name, Id FROM
        Automobile_Information__c WHERE Id IN: AutoInformationIds]);
        For(Opportunity_Automobile__c AutoOpp : lstOpportunityAutomobile.Values()){
            decimal num = 0;
            if(AutoOpp.Automobile__c == MapAutomobileInformation.get(AutoOpp.Automobile__c).Id &&
            OldMapOpportunity.get(AutoOpp.Opportunity__c).stagename != 'Closed Won'){

                num = MapAutomobileInformation.get(AutoOpp.Automobile__c).Quantity__c- AutoOpp.Quantity__c;
```

```
MapAutomobileInformation.get(AutoOpp.Automobile__c).quantity__c = num;
    lstAutomobileInfomation.add(MapAutomobileInformation.get(AutoOpp.Automobile__c));
}
}
If(!lstAutomobileInfomation.IsEmpty()){
    update lstAutomobileInfomation;
}

}
```

### Trigger Handler :

```
trigger OpportunityTrigger on Opportunity (before update, After Update) {
    if(trigger.isbefore && trigger.isUpdate){
        OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
    }
}
```

### Opportunity-Automobile Error

```
public class OpportunityAutomobileHandler {
    public static void quantityErrorOnAutomobileInformation(List<Opportunity_Automobile__c>
lstOpportunityAutomobile){
        set<Id> AutomobileIds = new Set<Id>();
        For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
            if(oppAutomobile.Automobile__c != null){
                AutomobileIds.add(oppAutomobile.Automobile__c);
            }
        }
        Map<Id,Automobile_Information__c> lstAutomobileInformation = new
map<Id,Automobile_Information__c>([SELECT Id, CreatedById, Quantity__c, Price__c FROM
Automobile_Information__c WHERE Id IN: AutomobileIds]);
        For(Opportunity_Automobile__c OppAutomobile : lstOpportunityAutomobile){
            If(OppAutomobile.Automobile__c == lstAutomobileInformation.get(OppAutomobile.Automobile__c).Id
&& lstAutomobileInformation.get(OppAutomobile.Automobile__c).Quantity__c <
OppAutomobile.Quantity__c){
                OppAutomobile.addError('the Number of Automobile u want are not Available !! the Automobile are
Available Count is ' + .get(OppAutomobile.Automobile__c).Quantity__c );
            }
        }
    }
}
```

## Trigger Handler :

```
trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before Update) {
    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){
        OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);
    }
}
```

## Invoice Creation Trigger

```
public class InvoiceCreation {
    public static void OpportunityClosedwonInvoiceGeneration(List<Opportunity> lstOpportunity,
    Map<Id,Opportunity>OldMapOpportunity){
        set<Id> oppIds = new Set<Id>();
        For(Opportunity opp : lstOpportunity){
            if(opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName !=
opp.StageName){
                oppIds.add(opp.Id);
            }
        }
        List<Opportunity_Automobile__c> lstOpportunityAutomobile = [SELECT Unit_Price__c, Total_Price__c,
Automobile__c, Quantity__c,Opportunity__c, Id FROM Opportunity_Automobile__c WHERE Opportunity__c
IN: oppIds];
        List<Invoice__c> lstInvoice = new List<Invoice__c>();
        For(Opportunity_Automobile__c oppAuto : lstOpportunityAutomobile){
            Invoice__c i = new Invoice__c();
            i.Quantity__c = oppAuto.Quantity__c;
            i.Unit_Price__c = oppAuto.Unit_Price__c;
            i.Total_Price__c = oppAuto.Total_Price__c;
            i.Purchase_Date__c = date.today();
            i.Opportunity__c = oppAuto.Opportunity__c;
            lstInvoice.add(i);
        }
        if(!lstInvoice.isEmpty()){
            insert lstInvoice;
        }
    }
}
```

## Trigger Handler :

```
trigger OpportunityTrigger on Opportunity (before update, After Update) {
    if(trigger.isbefore && trigger.isUpdate){
        OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
    }
    IF(trigger.isafter && trigger.isupdate){
        InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
    }
}
```



## Check contact role:

### Trigger:

```
public class ContactRoleCheck {
    public static void CheckcontactRoleonOpportunity(List<Opportunity> lstOpportunity,
    Map<Id,Opportunity>OldMapOpportunity){
        List<OpportunityContactRole> lstContactRole = [SELECT Id From OpportunityContactRole WHERE
    OpportunityId IN: OldMapOpportunity.keySet()];
        For(Opportunity opp : lstOpportunity){
            if(Opp.StageName == 'Closed Won' && OldMapOpportunity.get(opp.Id).StageName != opp.StageName){
                If(lstContactRole.isEmpty()){
                    opp.adderror('Please add contact Role on opportunity whenever Opportunity is Going to Closed
    Won.');
```

### Trigger Handler :

```
trigger OpportunityTrigger on Opportunity (before update, After Update) {
    if(trigger.isbefore && trigger.isUpdate){
        OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);
        ContactRoleCheck.CheckcontactRoleonOpportunity(trigger.new, trigger.oldMap);
    }
    IF(trigger.isafter && trigger.isupdate){
        InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);
    }
}
```

## LWC Component:

### Create Apex Class to Get Invoices:

```
public class OpportunityInvoiceswithLWC {
    @AuraEnabled(cacheable=true)
    public static List<Invoice__c> getInvoices(string OpportunityId){
        return [SELECT Id, Quantity__c, Purchase_Date__c, Opportunity__c, Unit_Price__c, Total_Price__c, Name
    FROM Invoice__c WHERE Opportunity__c =: OpportunityId];
    }
}
```

### 3 Install Salesforce CLI:

```
C:\Users\ navee>sfdx
Salesforce CLI

VERSION
sfdx-cli/7.182.1 win32-x64 node-v18.12.1

USAGE
$ sfdx [COMMAND]

TOPICS
alias      manage username aliases
auth       authorize an org for use with the Salesforce CLI
config     configure the Salesforce CLI
force      tools for the Salesforce developer
info       access cli info from the command line
plugins    add/remove/create CLI plug-ins
version
```

[codekiat.com](https://codekiat.com)

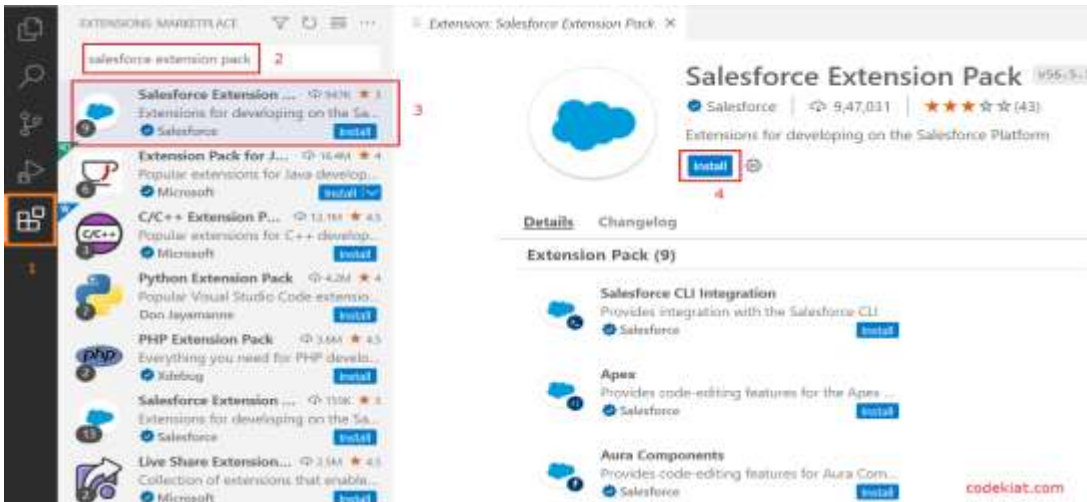
### Install Microsoft VS Code:

VS Code, or Visual Studio Code, is a free, open-source code editor developed by Microsoft. It is a lightweight, cross-platform code editor that provides features such as debugging, Git integration, and support for a wide range of programming languages.

[Download the version of the software](#) that is compatible with your operating system and install it.


The following instructions are for Windows OS. Other operating systems may have slightly different steps.

### Install the Salesforce Extension Pack:



The screenshot shows the Visual Studio Code interface with the Extensions Marketplace open. The search bar contains "salesforce extension pack". The search results list several extensions, with the "Salesforce Extension Pack" by Salesforce highlighted. The details panel on the right shows the "Salesforce Extension Pack" version 956.5.1, with a download count of 9,47,031 and 43 reviews. The "Install" button is visible. Below the main extension, a list of included components is shown: "Salesforce CLI Integration", "Apex", and "Aura Components", each with its own "Install" button. The source "codekiat.com" is noted at the bottom right.

Extension: Salesforce Extension Pack X



## Salesforce Extension Pack v56.5.1

Salesforce | 9,47,031 | ★★★★★ (43)

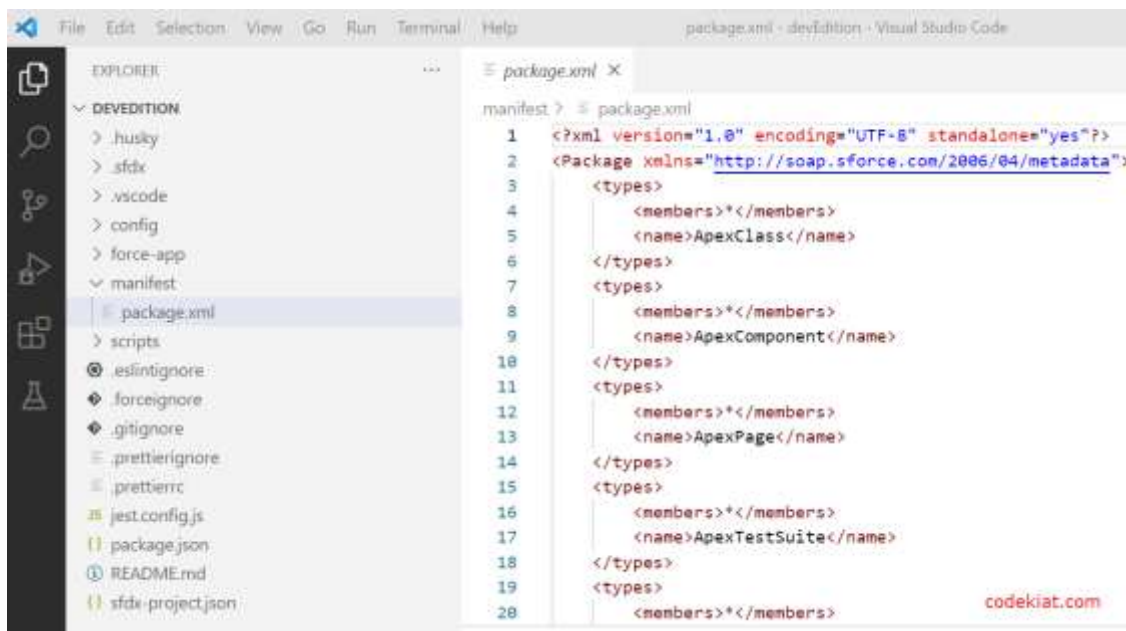
Extensions for developing on the Salesforce Platform

[Disable](#) [Uninstall](#) ⚙️

This extension is enabled globally.

[codekiat.com](https://codekiat.com)

### Create a project in VS Code:



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure of a project, including files like .husky, .sfdx, .vscode, config, force-app, manifest, package.xml, scripts, .eslintrc, .forceignore, .gitignore, .prettierrc, .prettierrc, jest.config.js, package.json, README.md, and sfdx-project.json. The 'package.xml' file is selected and highlighted. On the right, the Editor pane shows the content of the 'package.xml' file, which is an XML manifest for a Salesforce package. The XML content is as follows:

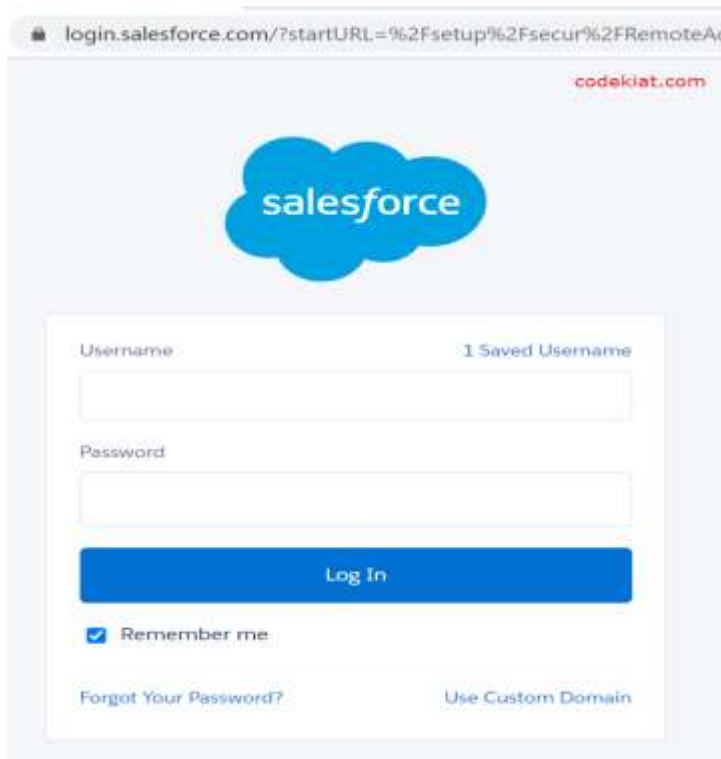
```

1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3   <types>
4     <members>*</members>
5     <name>ApexClass</name>
6   </types>
7   <types>
8     <members>*</members>
9     <name>ApexComponent</name>
10  </types>
11  <types>
12    <members>*</members>
13    <name>ApexPage</name>
14  </types>
15  <types>
16    <members>*</members>
17    <name>ApexTestSuite</name>
18  </types>
19  <types>
20    <members>*</members>

```

The 'codekiat.com' watermark is visible in the bottom right corner of the editor area.

### Authorize an org:



## Create Lightning Web Component:

### XML File 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>
```

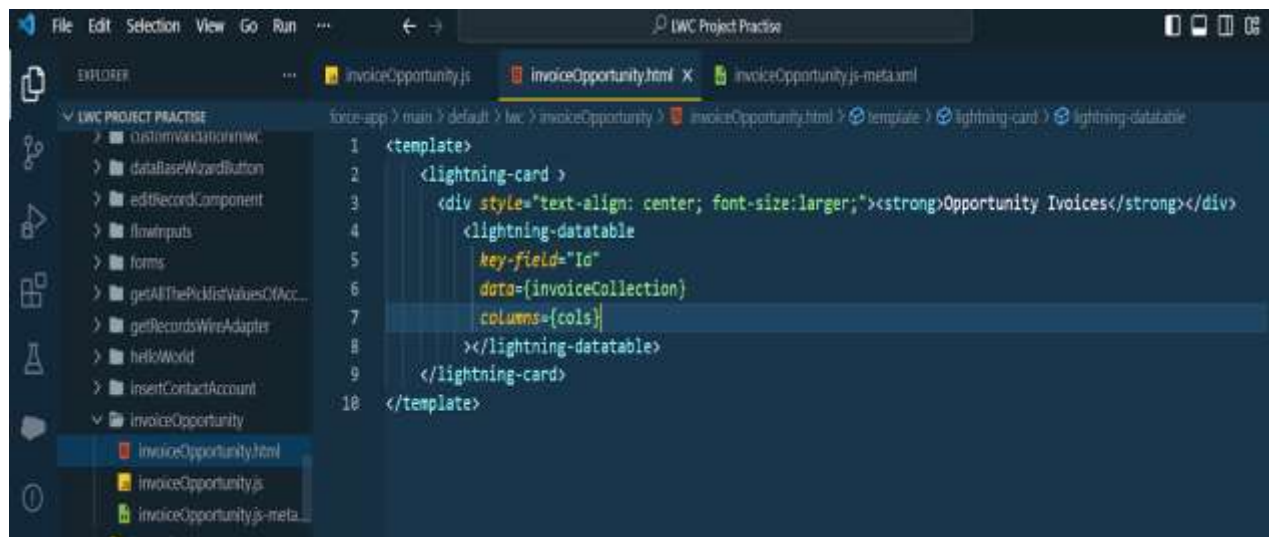
### JS File Code :

```
import { LightningElement, api, track, wire } from 'lwc';
import getInvoices from '@salesforce/apex/OpportunityInvoiceswithLWC.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:"Total Price" , fieldName:'Total_Price__c'},
    {label:"Purchase Date" , fieldName:'Purchase_Date__c'}
  ]
}
```

```
@wire(getInvoices,{ OpportunityId:'$recordId'})
invoicefunction({ data,error}){
  console.log(this.recordId +'this is record Id');
  if(data){
    console.log(data);
    this.invoiceCollection = data
  }if(error){
    console.log('this is error')
    console.log('error');
  }
}
```

## HTML File :

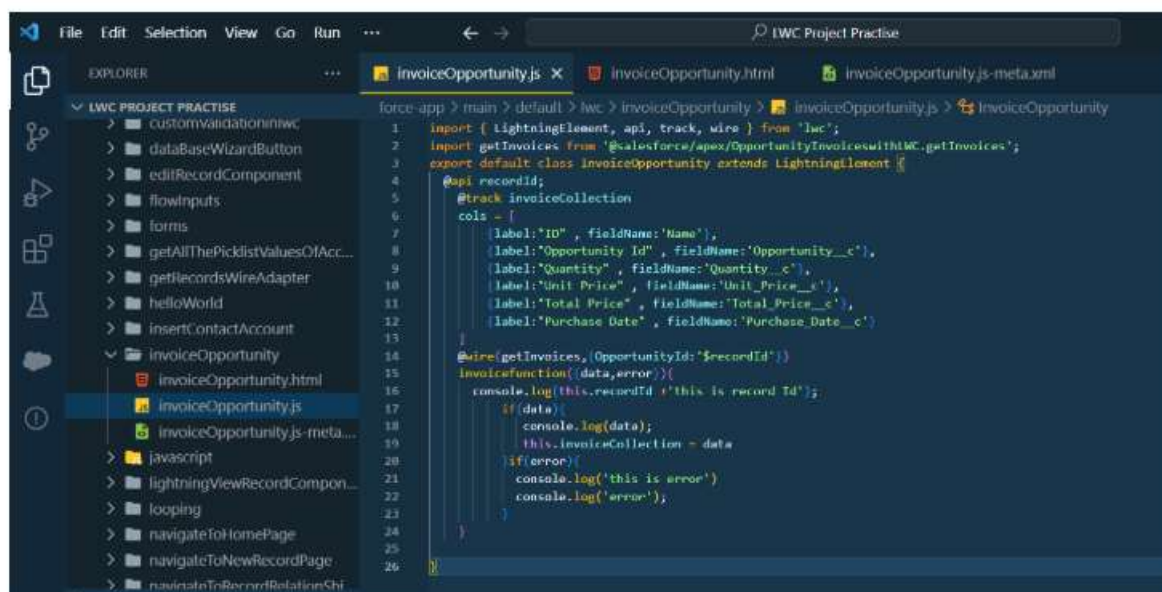


```
<template>
<lightning-card >
  <div style="text-align: center; font-size:larger;"><strong>Opportunity Ivoices</strong></div>
  <lightning-datatable
    key-field="Id"
    data={ invoiceCollection}
    columns={ cols }
  ></lightning-datatable>
</lightning-card>
</template>
```

## Create Lightning Web Component:

### JS File Code :

```
import { LightningElement, api, track, wire } from 'lwc';
import getInvoices from '@salesforce/apex/OpportunityInvoiceswithLWC.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'}
]
@wire(getInvoices,{OpportunityId:$recordId})
invoicefunction({data,error}){
  console.log(this.recordId +'this is record Id');
  if(data){
    console.log(data);
    this.invoiceCollection = data
  }if(error){
    console.log('this is error')
    console.log('error');
  }
}
```



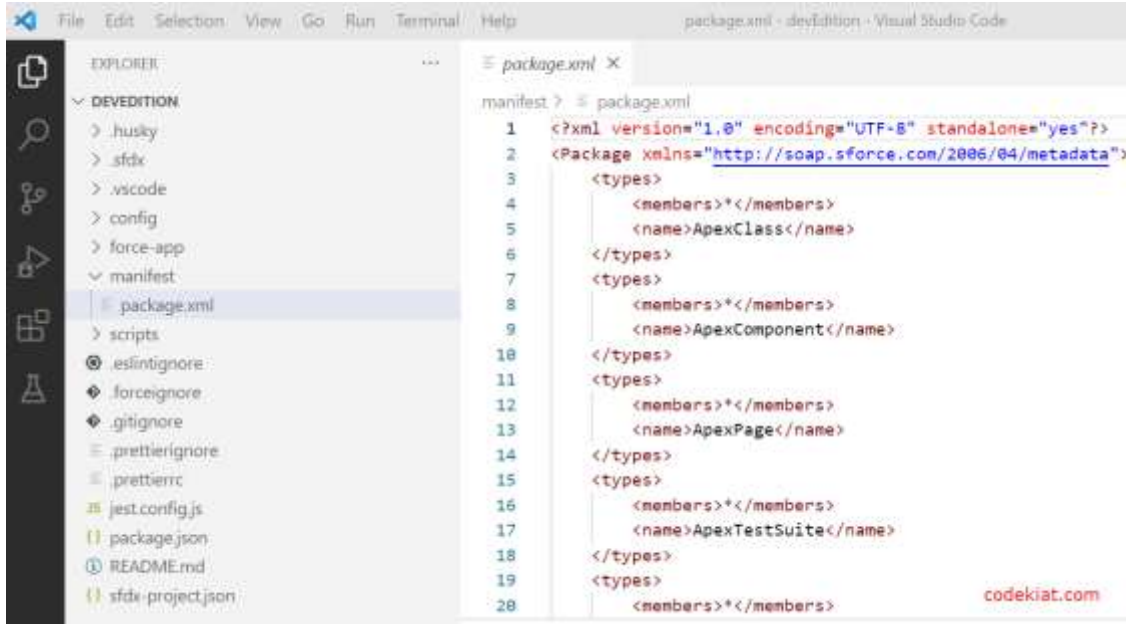


## Create Button to Add on Opportunity

Select the InvoiceOpportunity component

Label :- Invoices

Name :- Invoices



```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Package xmlns="http://soap.sforce.com/2006/04/metadata">
3   <types>
4     <members>*</members>
5     <name>ApexClass</name>
6   </types>
7   <types>
8     <members>*</members>
9     <name>ApexComponent</name>
10  </types>
11  <types>
12    <members>*</members>
13    <name>ApexPage</name>
14  </types>
15  <types>
16    <members>*</members>
17    <name>ApexTestSuite</name>
18  </types>
19  <types>
20    <members>*</members>
```

## Delete opportunity Schedule Class

### Objective :

Through this schedulable class, we can see all the Closed Lost Opportunities.

We can delete all the Closed lost Opportunities by this Scheduled method on every monday as weekly.

1. Login to the respective account and navigate to the gear icon in the top right corner.
2. Click on the Developer console. Now you will see a new console window.
3. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
4. Name the class as "DeleteClosedLostOpportunities "

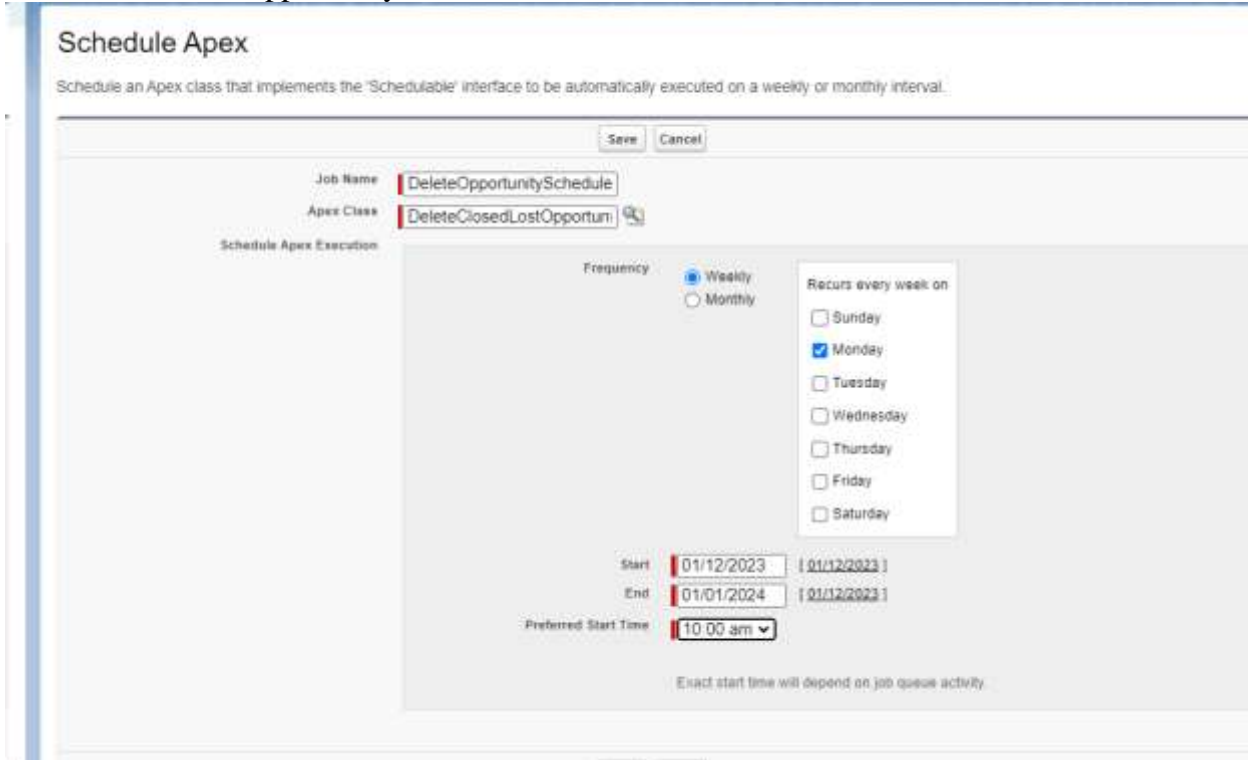
### CODE SNIPPET :

[illegible]



Click on Schedule Apex and enter the Job name.

- Job Name : DeleteOpportunitySchedule



**Schedule Apex**

Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly interval.

Save Cancel

Job Name: DeleteOpportunitySchedule

Apex Class: DeleteClosedLostOpportun

Schedule Apex Execution

Frequency: ☒ Weekly ☐ Monthly

Recurs every week on:

- ☐ Sunday
- ☒ Monday
- ☐ Tuesday
- ☐ Wednesday
- ☐ Thursday
- ☐ Friday
- ☐ Saturday

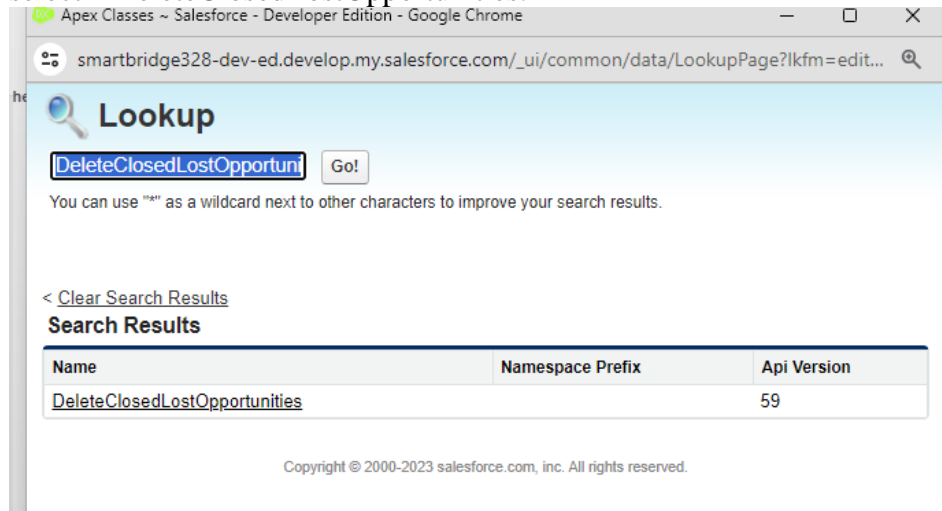
Start: 01/12/2023

End: 01/01/2024

Preferred Start Time: 10:00 am

Exact start time will depend on job queue activity.

Now click on the search icon present near the Apex class : Goto the Lookup icon beside ? click on it ? select DeleteClosedLostOpportunities.



Apex Classes ~ Salesforce - Developer Edition - Google Chrome

smartbridge328-dev-ed.develop.my.salesforce.com/\_ui/common/data/LookupPage?lkfm=edit...

**Lookup**

DeleteClosedLostOpportun Go!

You can use "\*" as a wildcard next to other characters to improve your search results.

< Clear Search Results

**Search Results**

Name	Namespace Prefix	Api Version
DeleteClosedLostOpportunities		59

Copyright © 2000-2023 salesforce.com, inc. All rights reserved.

In the Schedule Apex section , select weekly and select Monday mentioned and preferred time as 10:00 AM.

Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly interval.

Save

Cancel

Job Name

DeleteOpportunitySchedule

Apex Class

DeleteClosedLostOpportunum

Schedule Apex Execution

Frequency

☒ Weekly
 ☐ Monthly

Start

01/12/2023

01/12/2023

End

01/01/2024

01/12/2023

Preferred Start Time

10 00 am

Recurs every week on

☐ Sunday
 ☒ Monday
 ☐ Tuesday
 ☐ Wednesday
 ☐ Thursday
 ☐ Friday
 ☐ Saturday

Exact start time will depend on job queue activity.

Click on Save. Now enter Apex in the search box and select Apex jobs.

The screenshot shows the 'Scheduled Jobs' section of the SAP S/4HANA Cloud Setup Assistant. The left sidebar contains navigation links such as 'Quick Start', 'Run Home', 'Book Setup Assistant', 'Overview Setup Center', 'Get Factor Authentication Assistant', 'Performance Assistant', 'Recent Updates', 'Planning Extension Transition Center', 'Software Module App', and 'Planning Change'. The main area displays the title 'All Scheduled Jobs' with a subtitle explaining that it lists all jobs scheduled by each user and that multiple job types may display on this page. Below this is a table listing scheduled jobs.

Action	Job Name	Scheduled By	Scheduled On	Status	Next Scheduled Run	Type
<a href="#">Manage</a>	System Function Controller	System Administrator	01/07/2021, 01:07 pm	Success	04/07/2021, 10:30 am	Automatic sleep
<a href="#">Go</a>	Maintenance Data Loader Job for Org - 40000000000000000000000000000000	User Administrator	02/07/2021, 2:21 pm	In Progress	03/07/2021, 10:30 pm	Automated Data Loader job

You can see that the batch job is in queue and will run whenever the day mentioned comes.

## 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.

## Types of Reports in Salesforce

1. Tabular
2. Summary
3. Matrix
4. Joined Reports

## Create Report on Opportunity:

Opportunities

Showing a limited number of records. Run the report to see everything.

Account Name	Opportunity Name	Owner Role	Opportunity Owner	Stage	Next Step	Lead Source	Type
Subtotal							
Burlington Industries Corp of America (1)	Burlington Textiles Weaving Plant Services		Personal Services	Closed Won		Web	Existing Customer - Upgrade
Subtotal							
Downsouth (1)	Downsouth Middle Elementary		Personal Services	Qualification		Purchased List	New Customer
Subtotal							
Edge Communications (3)	Edge Emergency Car Wash		Personal Services	Closed Won		Word of mouth	New Customer
	Edge Installation		Personal Services	Closed Won		Word of mouth	Existing Customer - Upgrade
	Edge USA		Personal Services	Closed Won		Word of mouth	Existing Customer - Upgrade
Subtotal							
Grand Hotels & Resorts Ltd (5)	Grand Hotels Kitchen Renovation		Personal Services	In Decision Making			Existing Customer - Upgrade
	Grand Hotels Glass Portable Containers		Personal Services	Value Proposition		Employee Referral	Existing Customer - Upgrade
	Grand Hotels Generator Installation		Personal Services	Closed Won		Editorial Referral	Existing Customer - Upgrade
	Grand Hotels S.A.		Personal Services	Closed Won		Editorial Referral	Existing Customer - Upgrade
Subtotal							
	Grand Hotels Emergency Car Washes		Personal Services	Closed Won		Editorial Referral	New Customer

## Create Report on Automobile Information:

Filters:-

Sales Automobile U...

Accounts ▾ Currencies ▾ Opportunities ▾ Automobile Information ▾ Opportunity Automobiles ▾ Invoices ▾ Reports ▾ Dashboards ▾

REPORT ▾

Automobile Information Report ▾ Automobile Information

Showing a limited number of records. Run the report to see everything.

Filters

Automobile Information

Automobile Information	Name Of Manufacturer	Model	Built Date	Total Number Of Cylinders	Colour	Quantity	Price	VIN
1	Toyota	Corolla	12-18-2022	4	Red	10	\$10,000	1001000000000000
2	Old	Mustang	12-18-2022	8	Blue	10	\$10,000	1001000000000000
3	Subaru	Outback	14-10-1975	4	Green	10	\$10,000	1001000000000000
4	Hyundai	Tucson	10-26-2022	4	Red	10	\$10,000	1001000000000000
5	Jeep	Wrangler	10-26-2022	4	Blue	10	\$10,000	1001000000000000
6	Audi	A8	10-26-2022	4	Red	10	\$10,000	1001000000000000
7	Mercedes-Benz	C-Class	10-26-2022	4	Grey	10	\$10,000	1001000000000000
8	BMW	3 Series	10-26-2022	4	White	10	\$10,000	1001000000000000
9	Chrysler	Pacifica	10-26-2022	4	Black	10	\$10,000	1001000000000000
10						100	\$10,000	

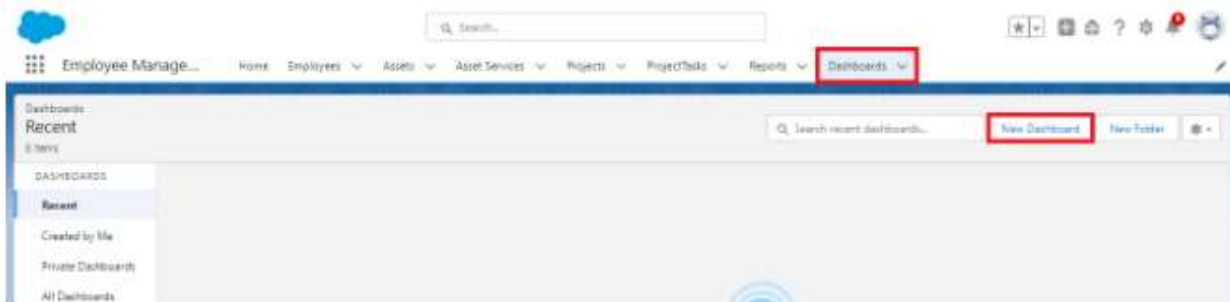
## Dashboard:

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. Before building, reading, and sharing dashboards, review these dashboard basics.

## Sales Dashboard:

### Create Dashboard

1. Go to the app ? click on the Dashboards tabs.



2. Give a Name and click on Create.

### New Dashboard

**\* Name**

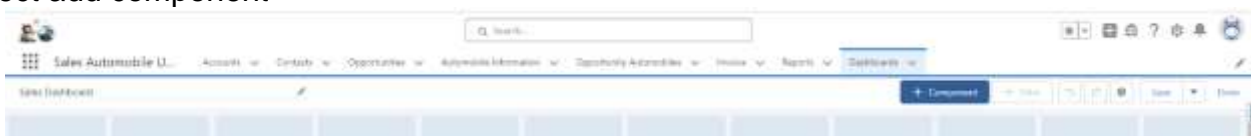
**Description**

**Folder**

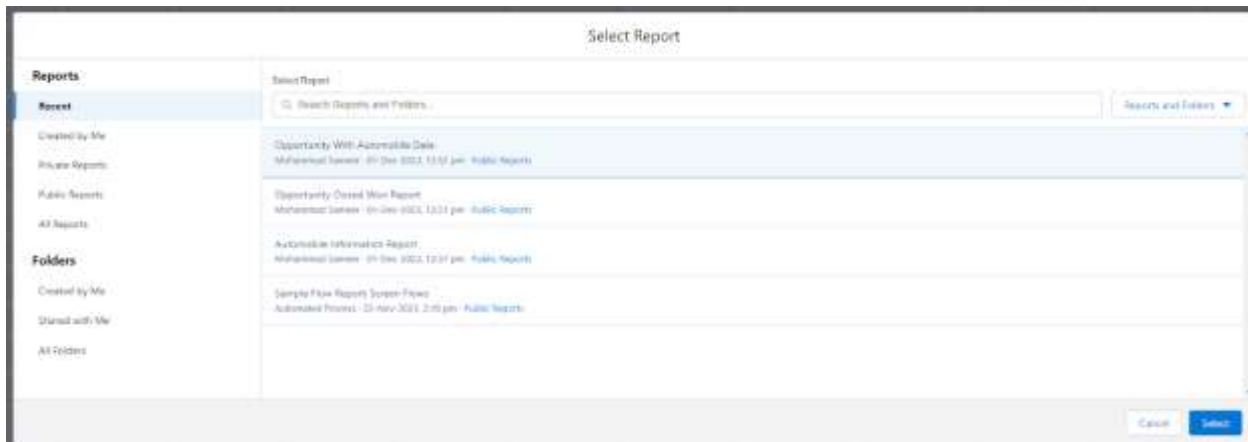


Name : Automobile Sales

3. Select add component



4. Select a Report and click on select.



5. Click Add then click on Save and then click on Done.

The Created Dashboard will look like this.



## Conclusion:

### Summary of Achievements:

The Salesforce Automobile Information CRM project has successfully integrated sales and service functionalities into a single platform, improving both internal operations and customer satisfaction. The CRM system now allows seamless tracking of inventory, sales opportunities, and customer interactions while automating key processes to boost efficiency. With enhanced reporting and analytics, the organization can make data-driven decisions to grow their business and improve the customer experience. The solution is scalable, customizable, and aligns with the long-term goals of the automobile dealership, setting a foundation for future growth and digital transformation.