### Project Name: Sales Automobile Using Salesforce CRM

Team ID : LTVIP2024TMID11571

Team Leader : R JAHNAVI

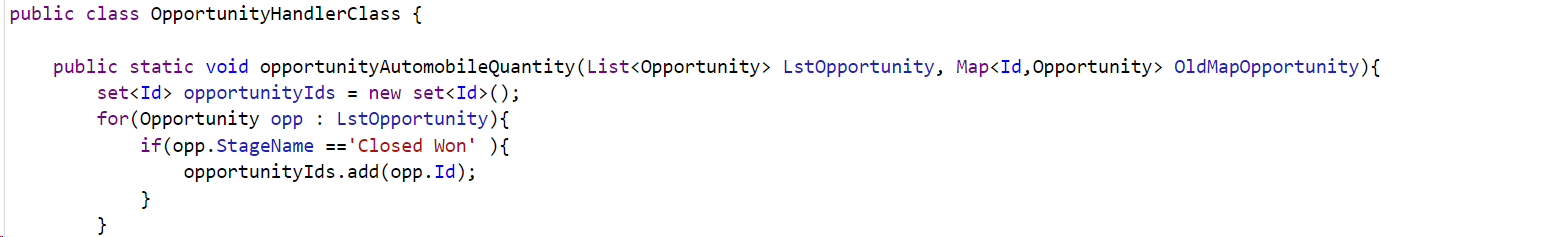
Team member : CHENDRAYAPPAGARI PAVANI

### Apex Trigger

**Opportunity Automobile Quantity**

**UseCase : Whenever Opportunity Closed won Than Neglect / Minus the Quantity From Automobile Information on the Bases of Opportunity Automobile quantity.**

1. Login to the respective trailhead 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 “OpportunityHandlerClass ”.





**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;

        }

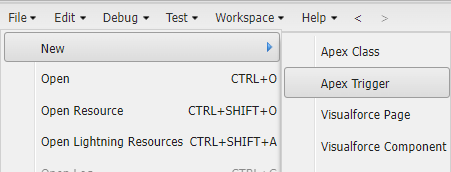
    }

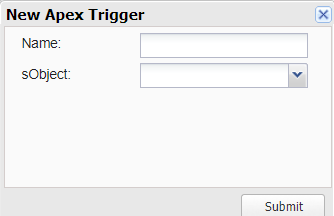
}

**Trigger Handler :**

How to create a new trigger :

1. While still in the account, navigate to the gear icon in the top right corner.
2. Click on developer console and you will be navigated to a new console window.
3. Click on the File menu in the toolbar, and click on new? Trigger.
4. Enter the trigger name and the object to be triggered.
5. Name  : OpportunityTrigger
6. sObject : Opportunity





Syntax For creating trigger :

The syntax for creating trigger is :

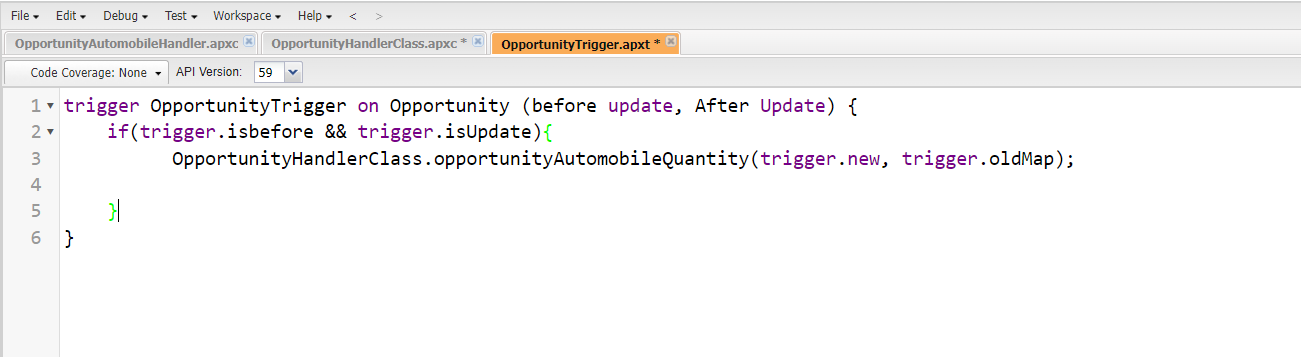
Trigger [trigger name] on [object name]( Before/After event){

//block of code

}

In this project , trigger is called whenever the particular records sum exceed the threshold i.e minimum business requirement value. Then the code in the trigger will get executed.

1. Trigger for Opportunity Object.



**Code:**

trigger OpportunityTrigger on Opportunity (before update, After Update) {

    if(trigger.isbefore && trigger.isUpdate){

          OpportunityHandlerClass.opportunityAutomobileQuantity(trigger.new, trigger.oldMap);

    }

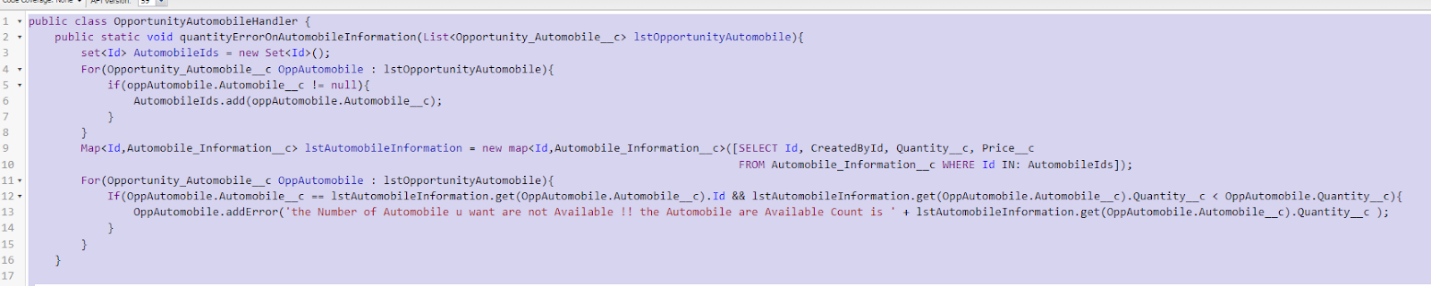
}

### Opportunity-Automobile Error

**UseCase : If Quantity of Automobile is Zero or Less than The Quantity from The Opportunity-Automobile Than Throw an error .**

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.
2. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
3. Name the class as “OpportunityAutomobileHandler ”.



**Code:**

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 ' + lstAutomobileInformation.get(OppAutomobile.Automobile\_\_c).Quantity\_\_c );

            }

        }

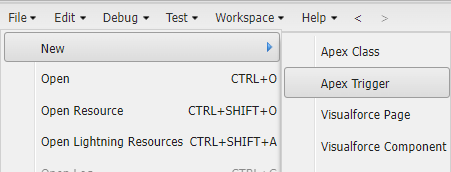
    }

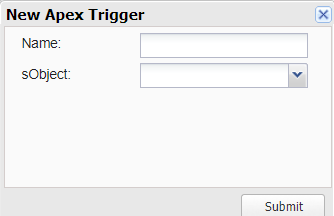
}

**Trigger Handler :**

How to create a new trigger :

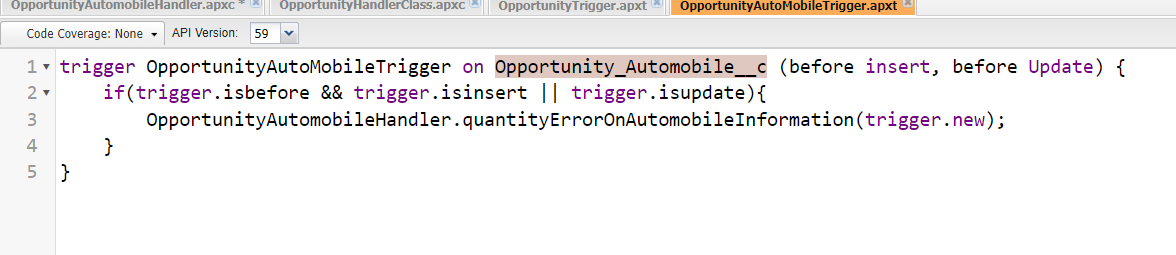
1. While still in the trailhead account, navigate to the gear icon in the top right corner.
2. Click on developer console and you will be navigated to a new console window.
3. Click on the File menu in the toolbar, and click on new? Trigger.
4. Enter the trigger name and the object to be triggered.
5. Name  : OpportunityAutoMobileTrigger
6. sObject : Opportunity\_Automobile\_\_c





**Trigger :**

Handler for the Opportunity\_Automobile\_\_c  Object



**Code:**

trigger OpportunityAutoMobileTrigger on Opportunity\_Automobile\_\_c (before insert, before Update) {

    if(trigger.isbefore && trigger.isinsert || trigger.isupdate){

        OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(trigger.new);

    }

}

### Invoice Creation Trigger

**UseCase : Whenever an opportunity is Closed won then create the Invoice on the Bases of Opportunity Automobile Data.**

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.
2. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
3. Name the class as “InvoiceCreation”.  
   

**Code:**

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;

        }

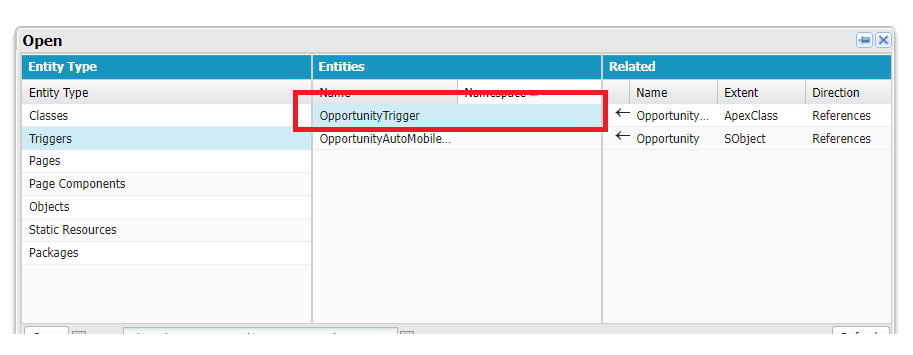
    }

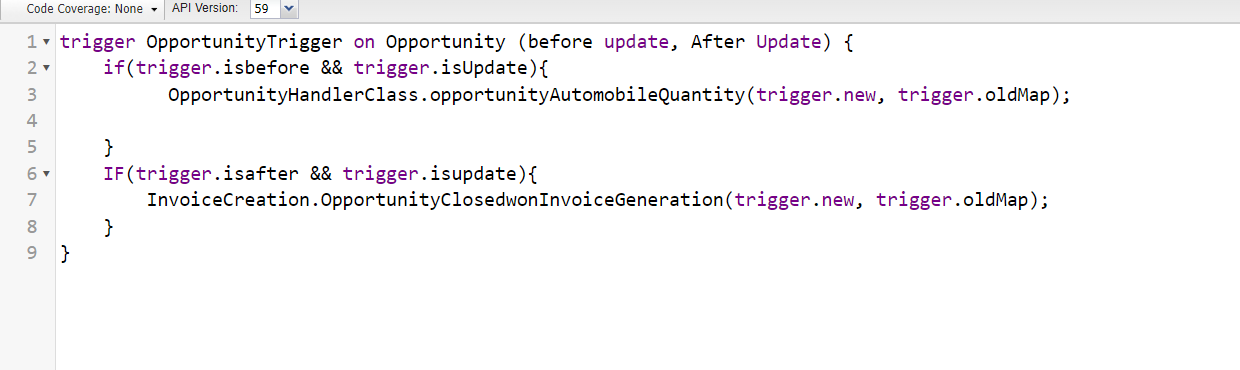
}

**Trigger Handler :**

For this class we don’t need to create any trigger, we will call this Code in “Opportunity Trigger”.

1. Go on files and click on open.
2. Click on triggers.
3. Double click on OpportunityTrigger.





**Trigger:**

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);

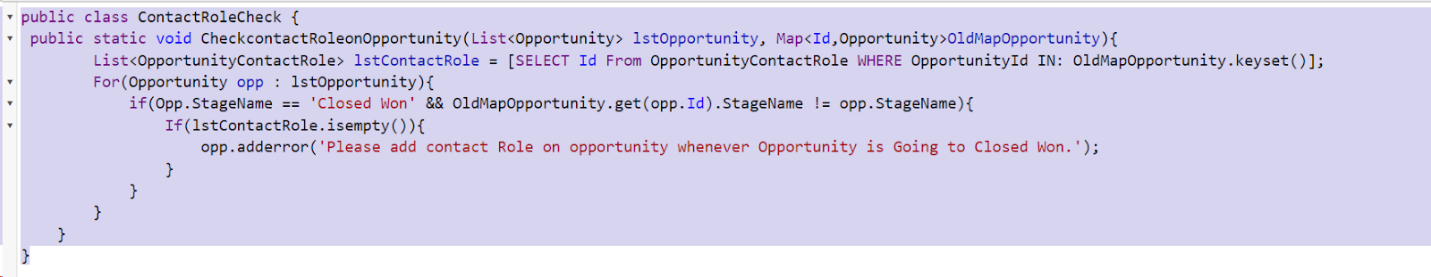
    }

}

### Check Contact Role

**UseCase : Whenever an opportunity is Going to Closed won then check it has the contact role or Not.**

Login to the respective trailhead account and navigate to the gear icon in the top right corner.

1. Click on the Developer console. Now you will see a new console window.
2. In the toolbar, you can see FILE. Click on it and navigate to new and create New apex class.
3. Name the class as “ContactRoleCheck ”.  
   

**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.');

                }

            }

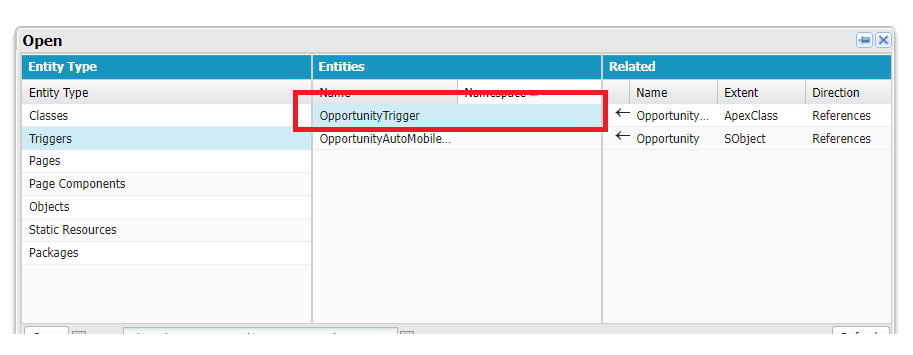
        }

    }

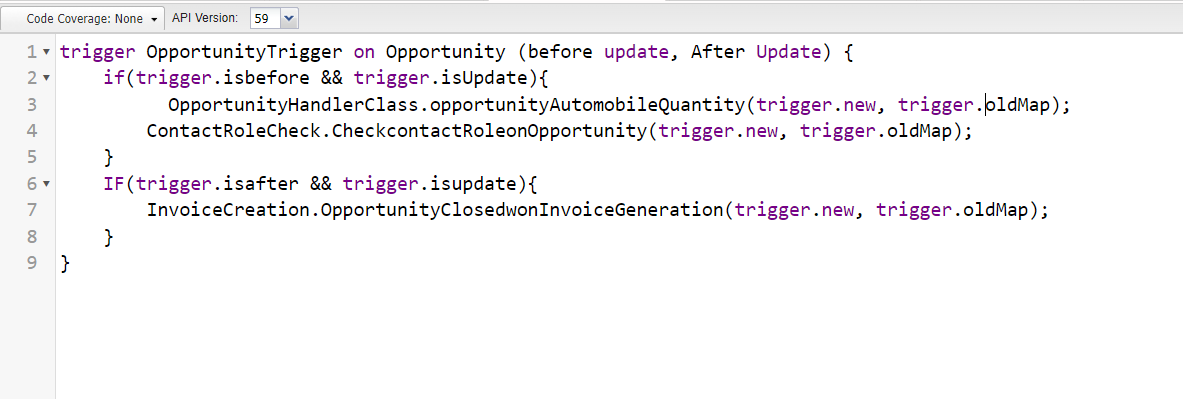
}

**Trigger Handler :**

For this class we don’t need to create any trigger, we will call this Code in “Opportunity Trigger”.

1. Go on files and click on open.
2. Click on triggers.
3. Double click on OpportunityTrigger.  
   

Trigger Code  :



Trigger:

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){

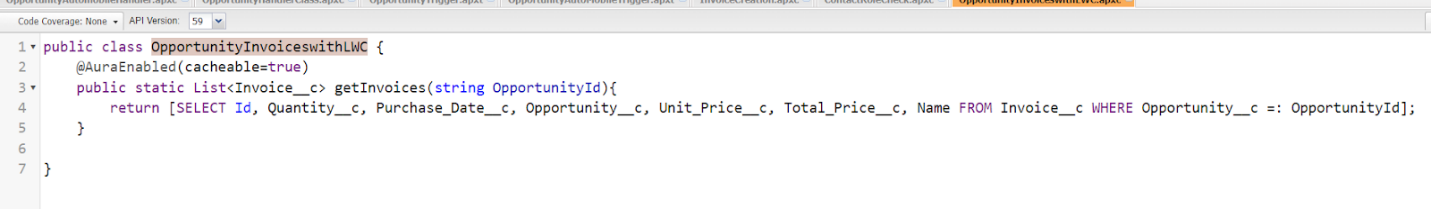
        InvoiceCreation.OpportunityClosedwonInvoiceGeneration(trigger.new, trigger.oldMap);

    }

}

### LWC Component:

**Create Apex Class To Get Invoices**

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

**Code:**

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];

    }

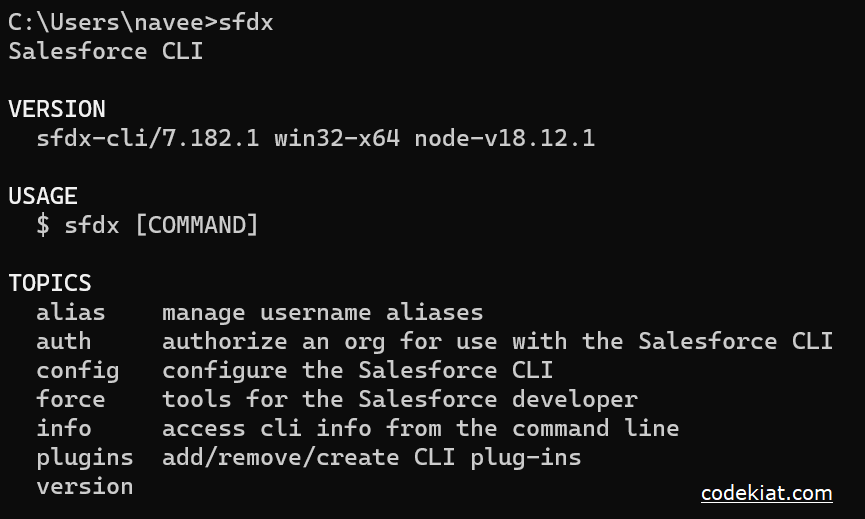
}

### Install Salesforce CLI

The Salesforce CLI is a powerful command line interface that simplifies development and build automation when working with your Salesforce org.

[Download and install Salesforce CLI](https://developer.salesforce.com/tools/sfdxcli)

To confirm that the Salesforce CLI is installed and working correctly, you can open a command prompt and type sfdx. This will display the version number of the Salesforce CLI that is currently installed on your system.



### 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](https://code.visualstudio.com/download) 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

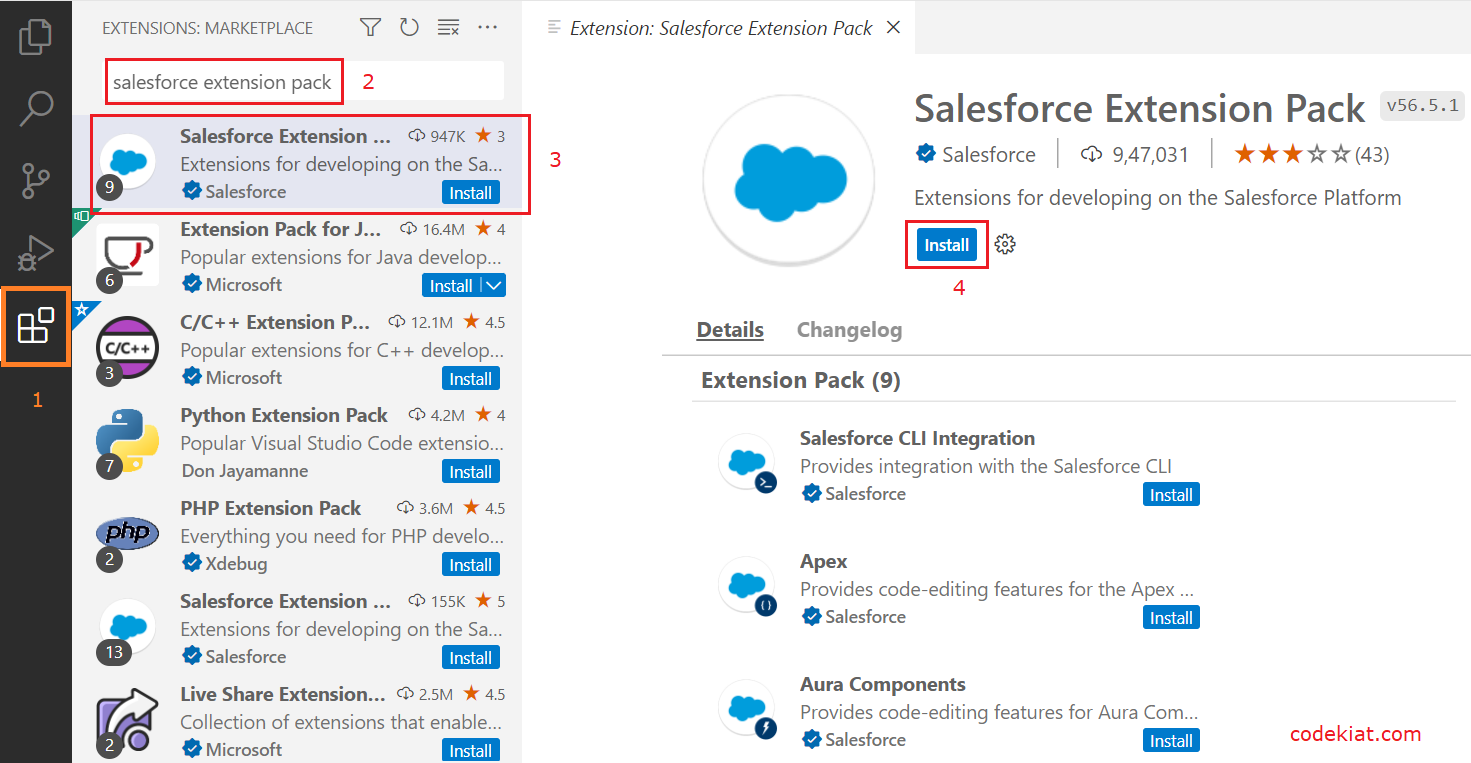
**In the VS Code,**

1. go to extensions (1) as shown in the image below.

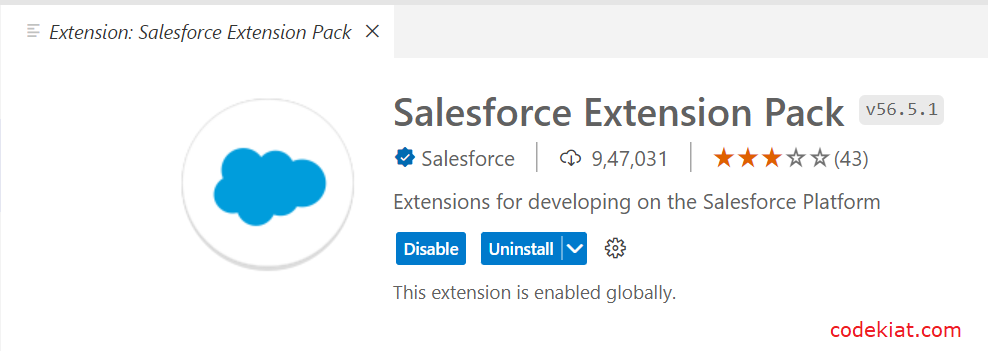
2. Search with the Salesforce extension pack (2) as shown in the image below.

3. select Salesforce Extension Pack from the list (3) as shown in the image below.

4.  Click the Install button (4) as shown in the image below.

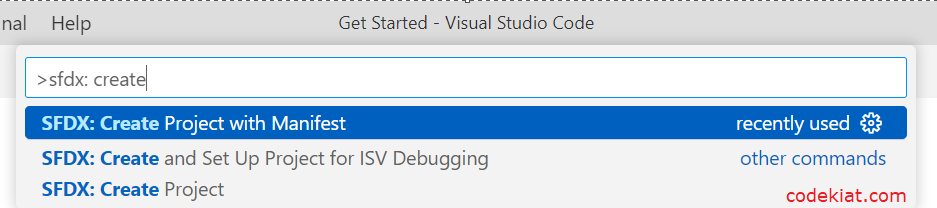
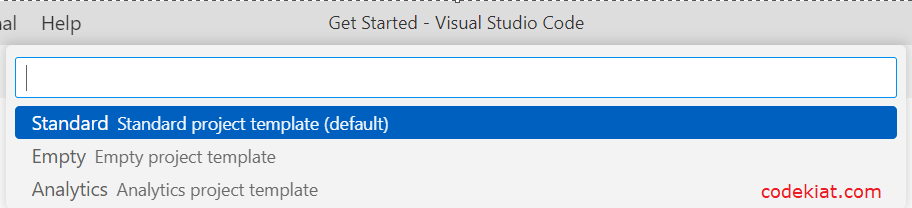
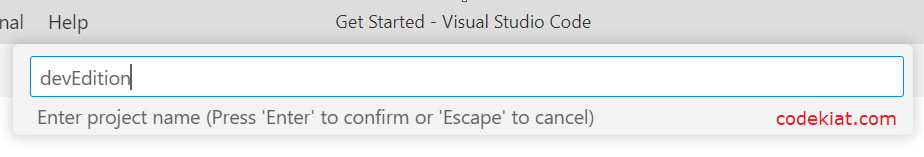
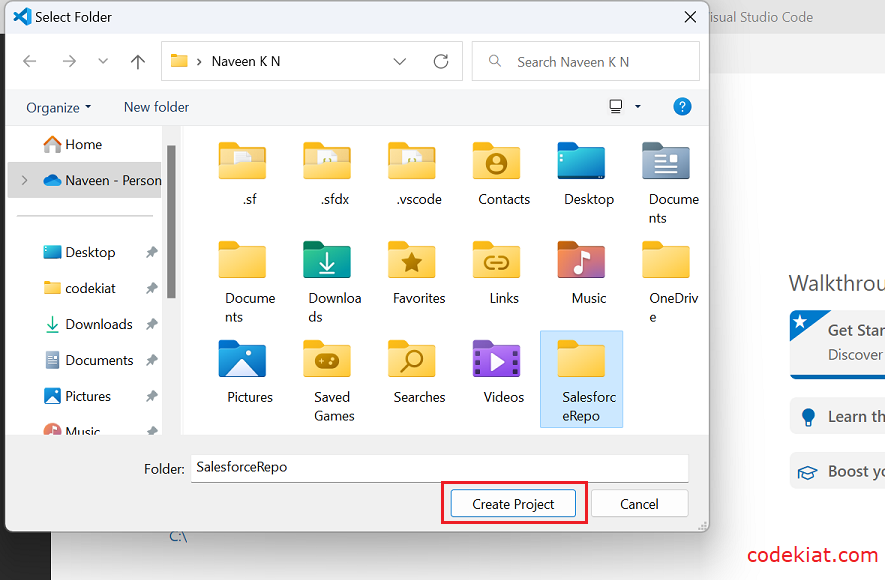
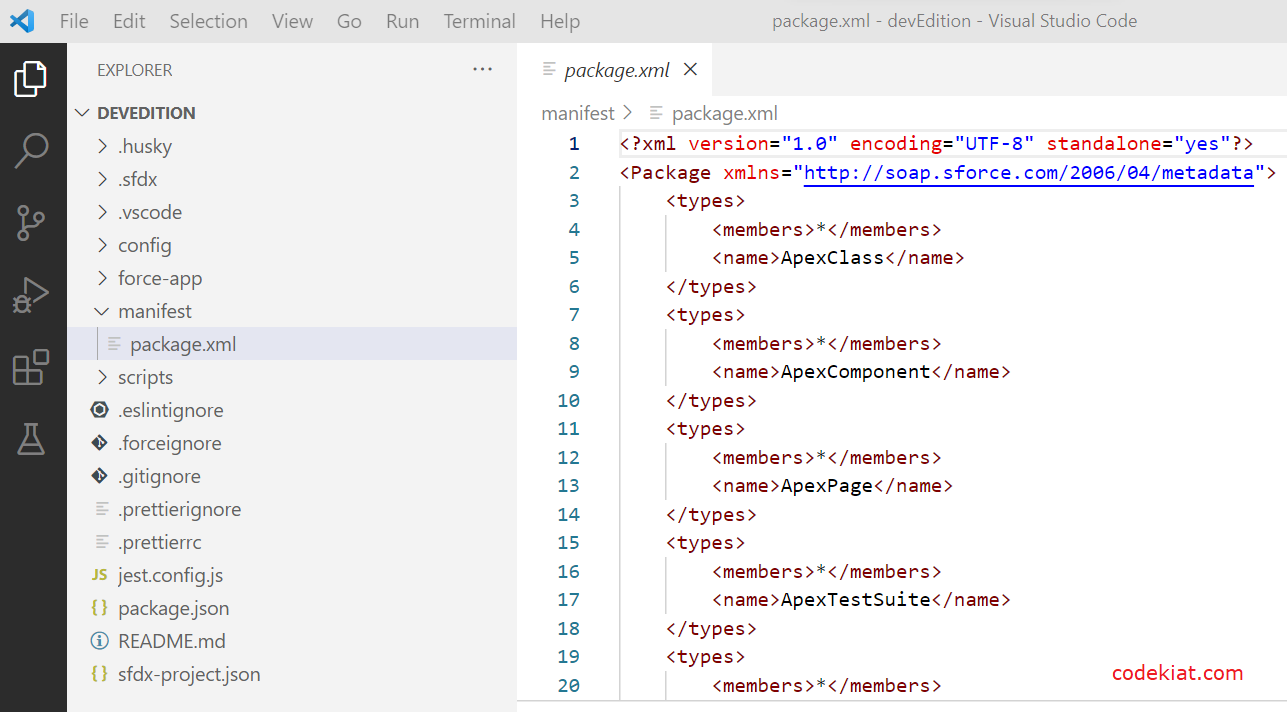


The extension pack is installed successfully



 Install the Salesforce Extension Pack

### Create A Project In VS Code

1. Press CTRL + SHIFT + P, type sfdx: create
2. select SFDX: Create Project with Manifest  
   
3. Select the Standard project template  
   
4. Type a project name and Click Enter.  
   
5. Select the folder (create a new folder if required) and click Create Project  
   
6. The new project is created with package.xml  
   

Default Package.xml contains various metadata types. I have updated Package.xml as shown below as we deal only with LWC in this article.

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<Package xmlns="http://soap.sforce.com/2006/04/metadata">

    <types>

        <members>\*</members>

        <name>LightningComponentBundle</name>

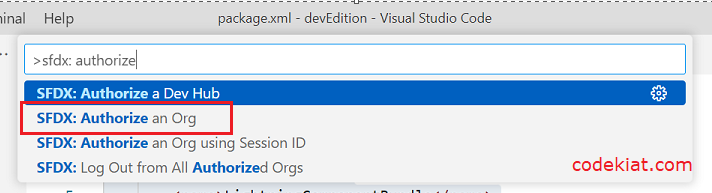
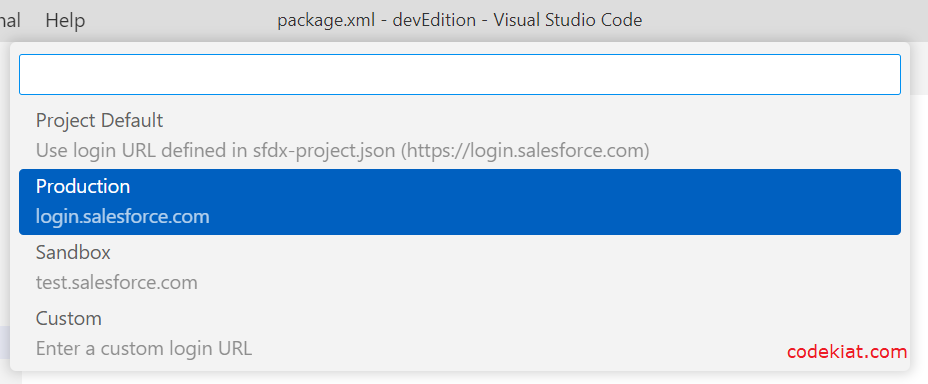
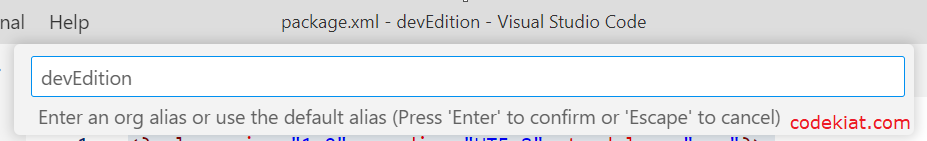
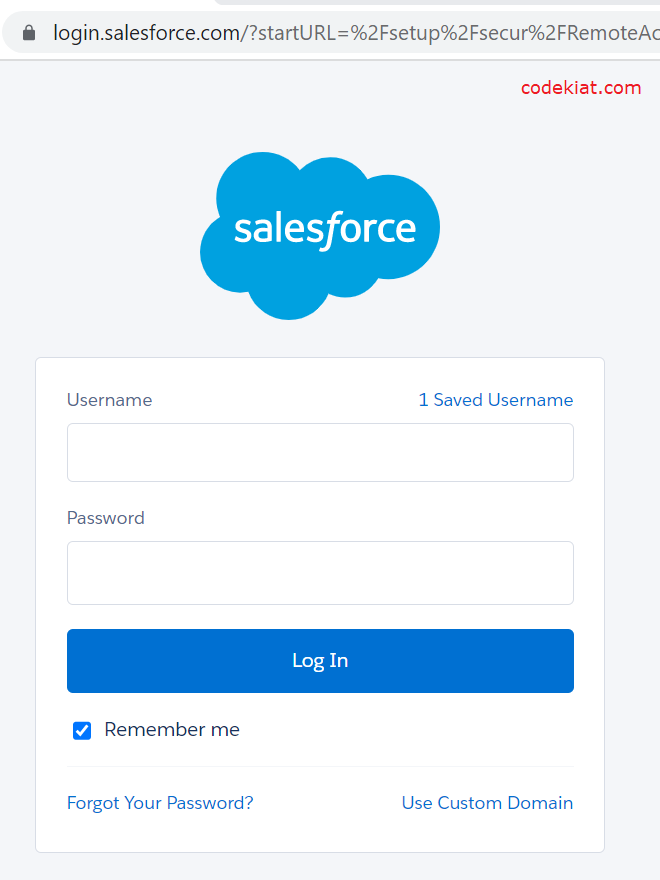
    </types>

    <version>55.0</version>

</Package>

### Authorize An Org

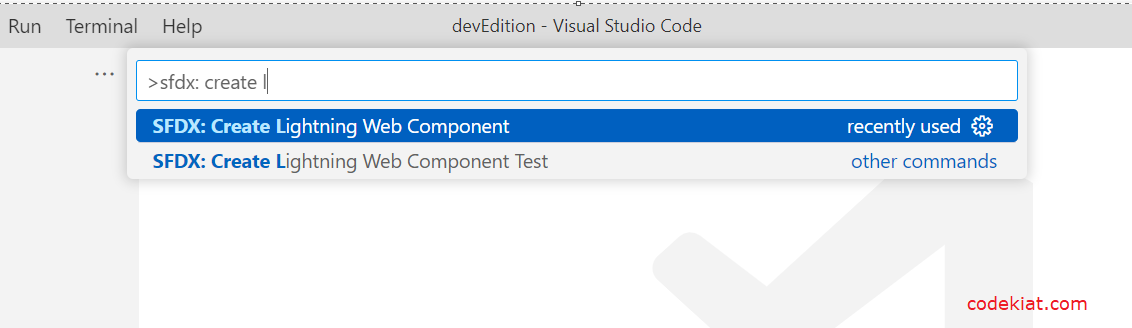
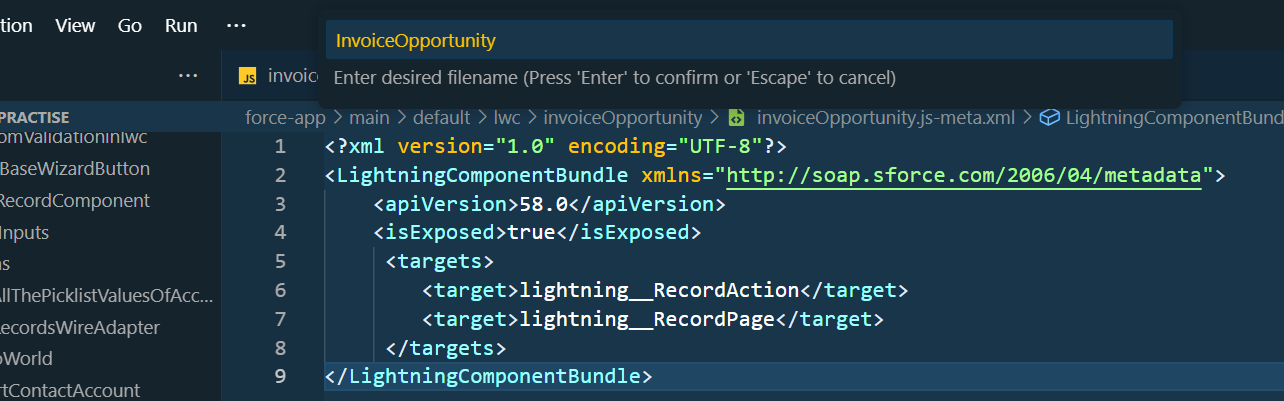
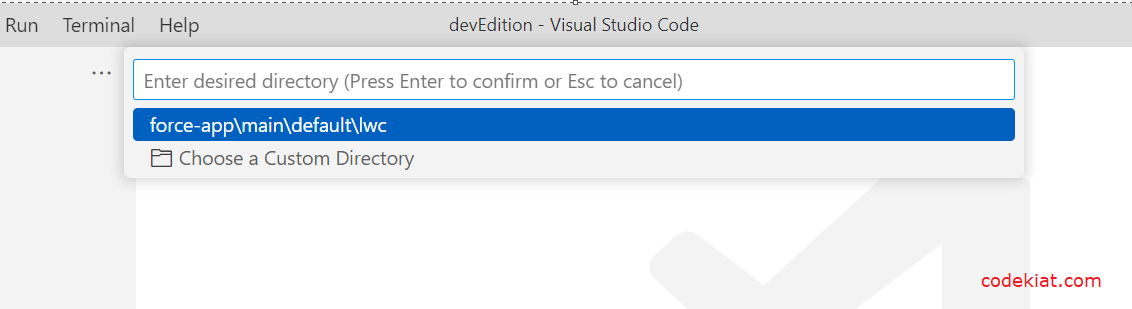
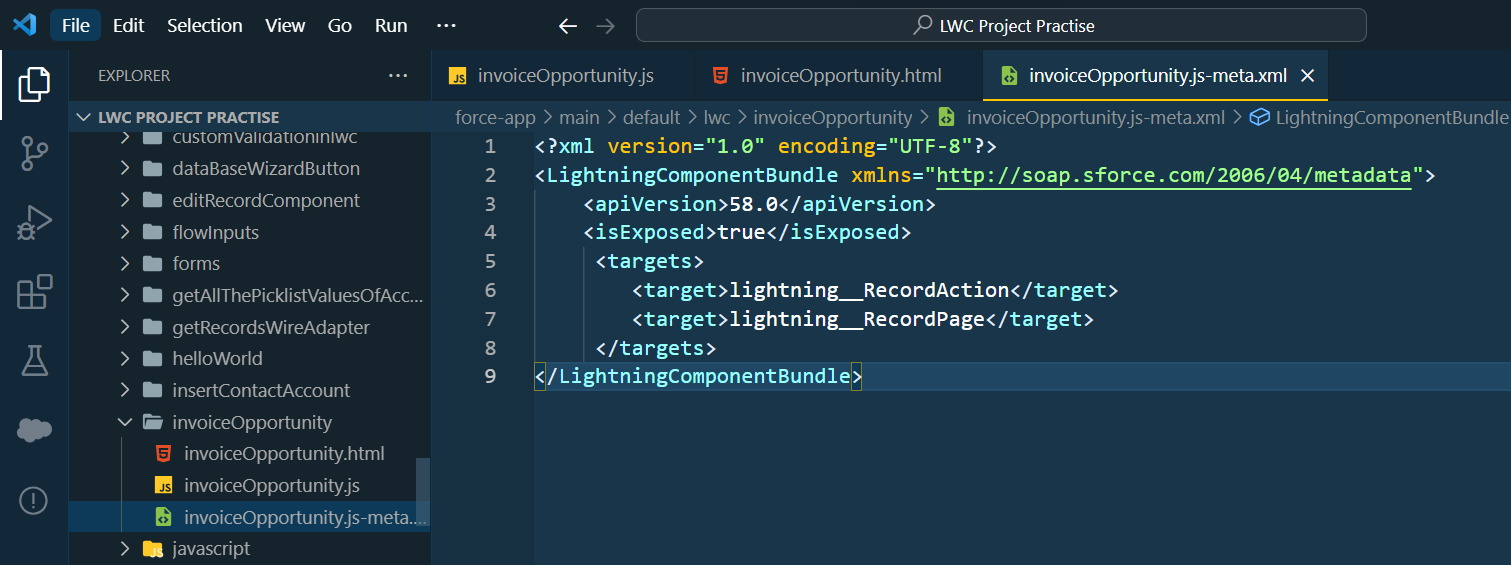
**Establish a connection between the local project and the Salesforce instance to retrieve and deploy the components.**

1. Press CTRL + SHIFT + P, type sfdx: authorize.
2. select SFDX: Authorize an Org from the list  
   
3. Choose your Salesforce instance.  
   For developer edition and production instances select Production. 
4. For this demonstration, I used the developer edition, hence it is Production.
5. Give a project name and press Enter  
   
6. The Salesforce login page opens in the browser.
7. Enter the credentials and click Log In  
   

1. It will be successfully authorized.

**Create Lightning Web Component**

**XML File :**

1. In the VS Code, press CTRL + SHIFT + P, type sfdx: create lightning in the search bar, and select SFDX: Create Lightning Web Component  
   
2. Give the name “InvoiceOpportunity” and press Enter.  
   
3. Choose the directory.  
   
4. LWC is created successfully.  
   
5. Copy and paste the below-mentioned code in the InvoiceOpportunity.js-meta.xml and update the apiVersion tag with the latest API version.

**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>

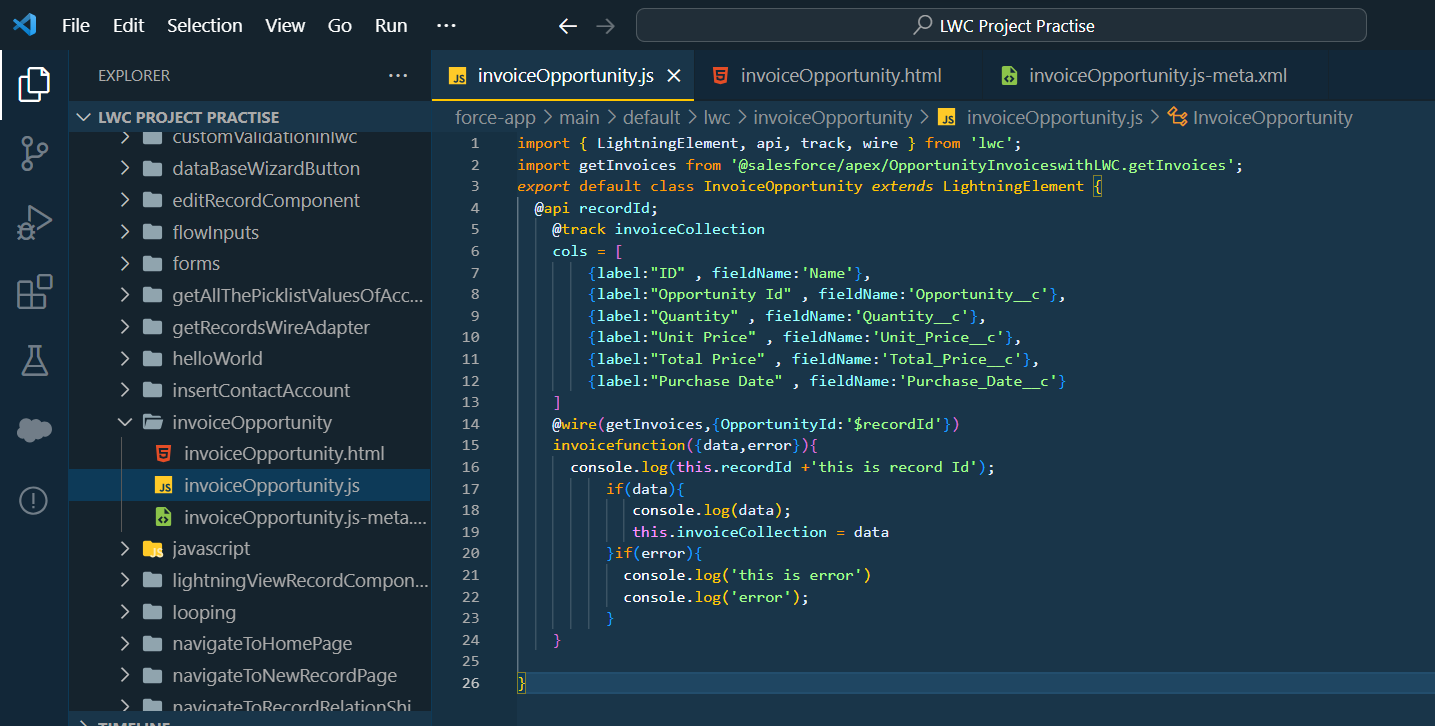
        <target>lightning\_\_RecordAction</target>

        <target>lightning\_\_RecordPage</target>

     </targets>

</LightningComponentBundle>

**JS File :**

1. Copy and paste the below-mentioned code in the InvoiceOpportunity.js  and update the apiVersion tag with the latest API version.  
   

**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')

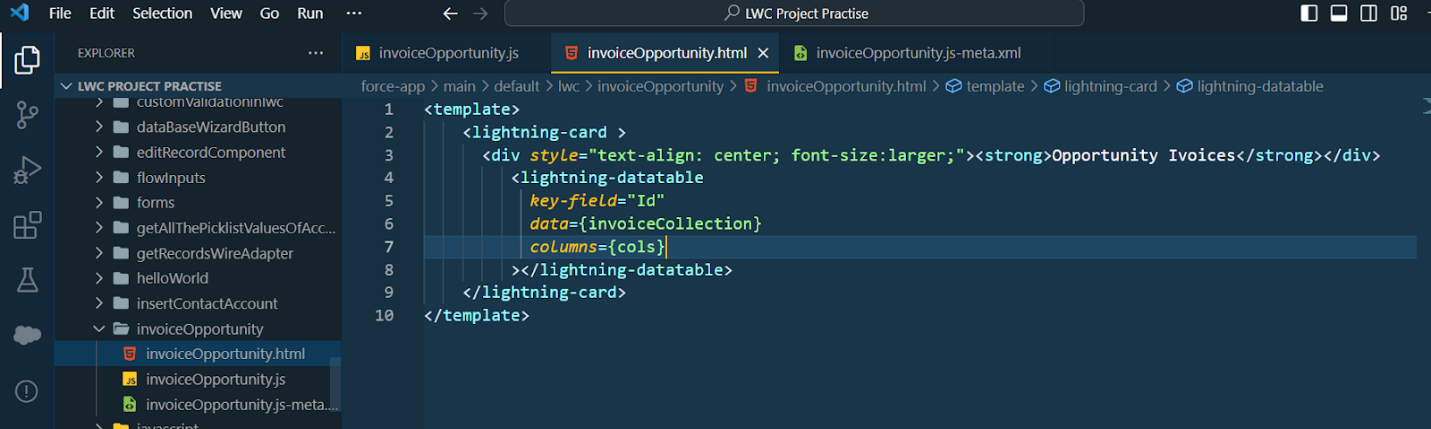
            console.log('error');

          }

    }

}

**HTML File :**

1. Copy and paste the below-mentioned code in the InvoiceOpportunity.html  and update the apiVersion tag with the latest API version.  
   

**HTML File Code:**

<template>

    <lightning-card >

      <div style="text-align: center; font-size:larger;"><strong>Opportunity Ivoices</strong></div>

         <lightning-datatable

           key-field="Id"

           data={invoiceCollection}

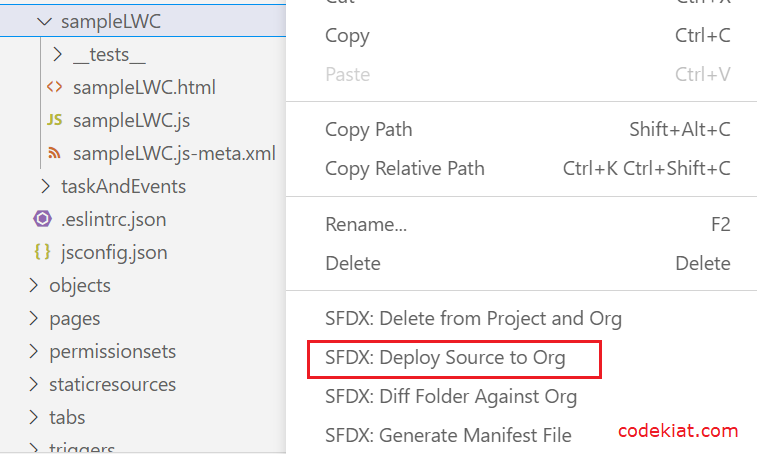
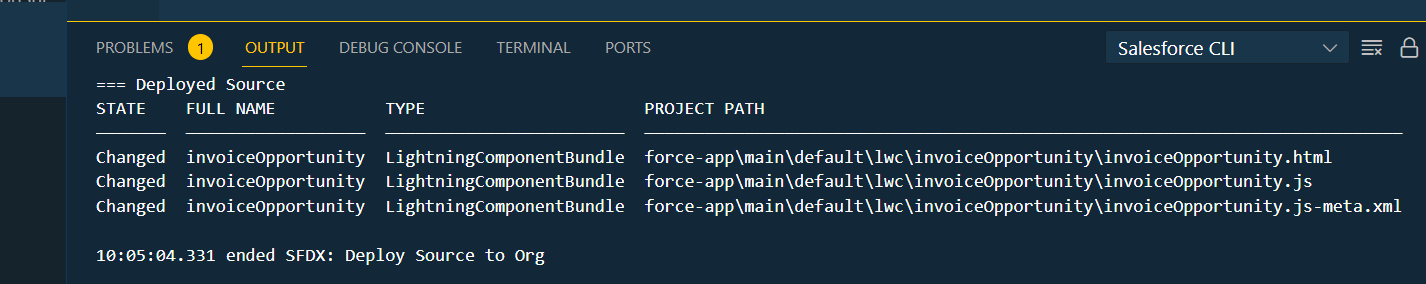
           columns={cols}

         ></lightning-datatable>

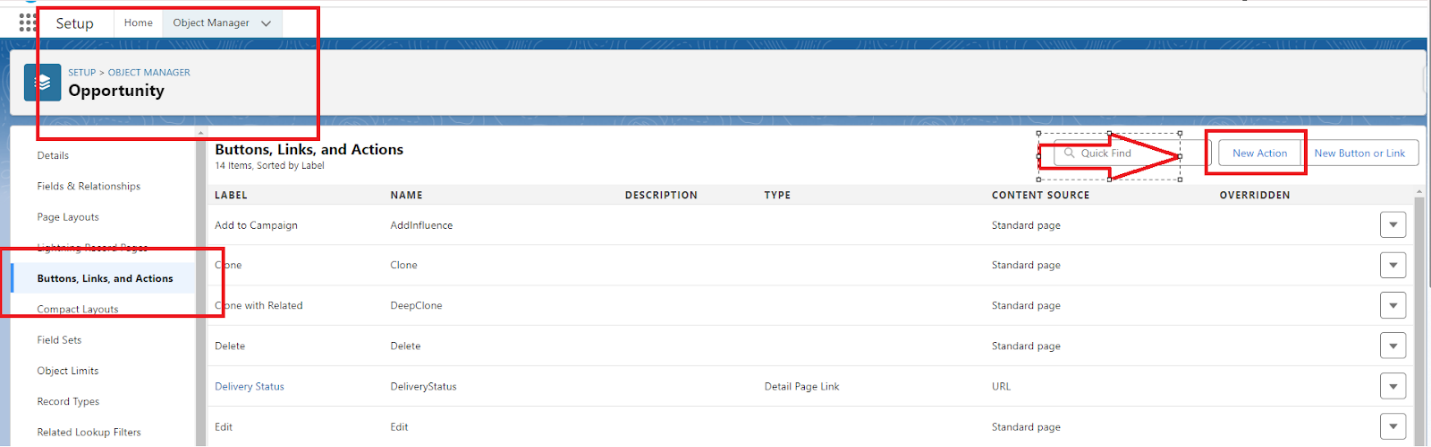
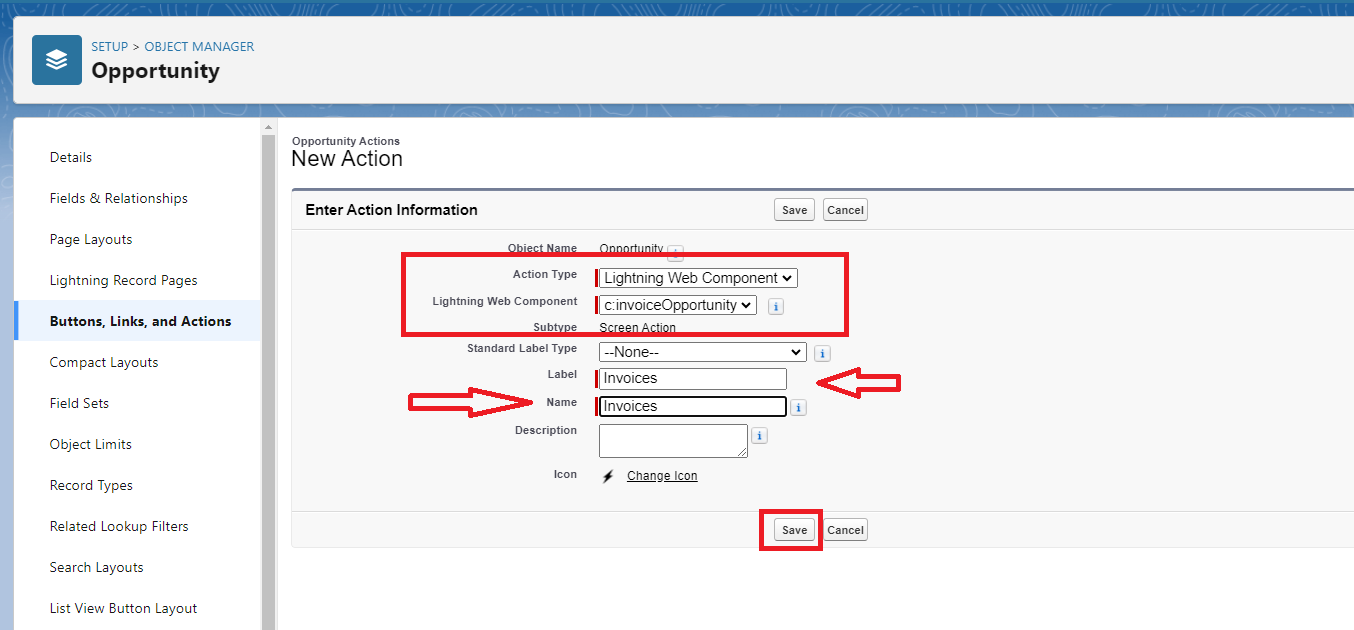
    </lightning-card>

</template>

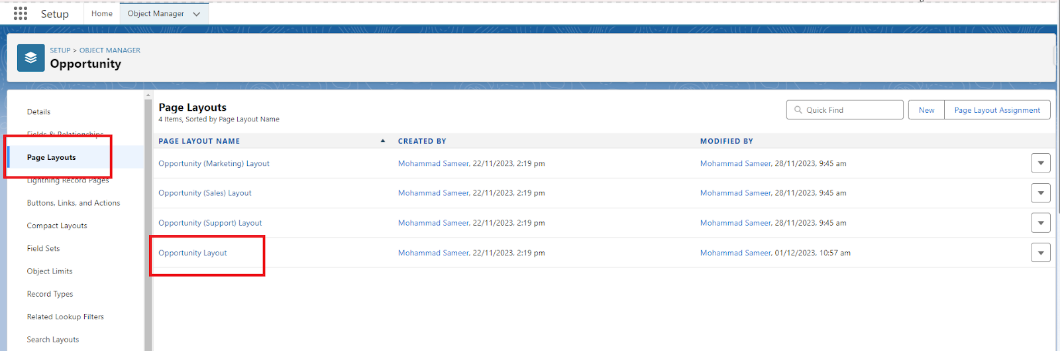
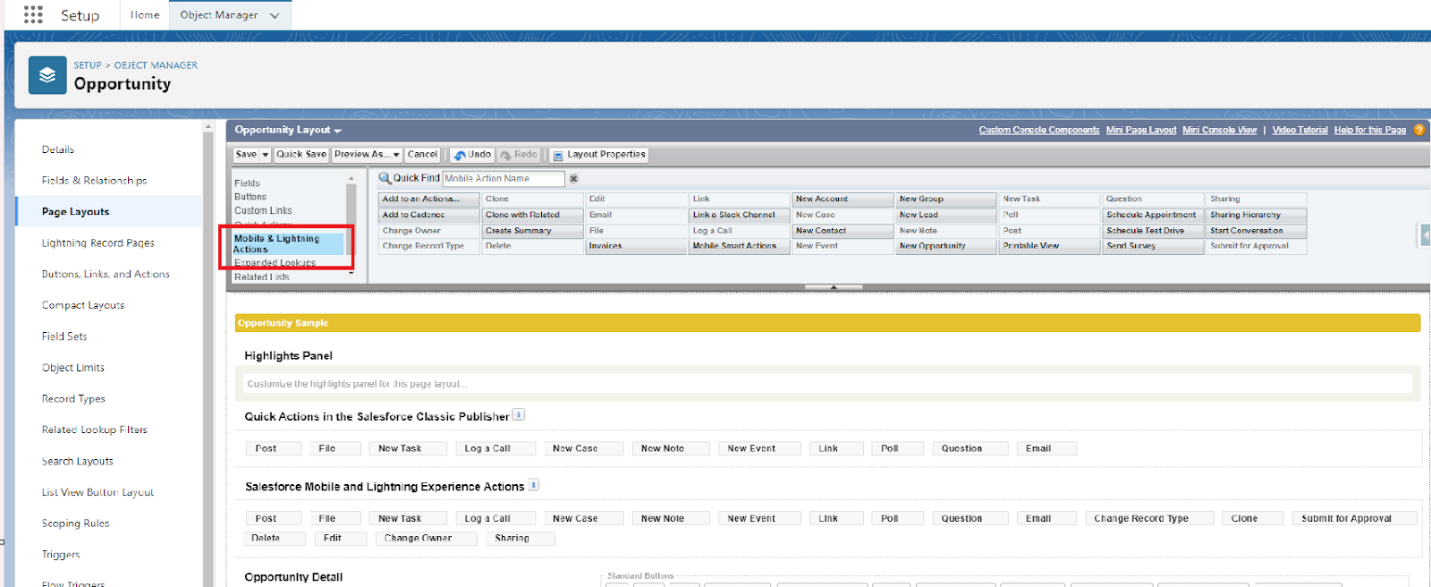
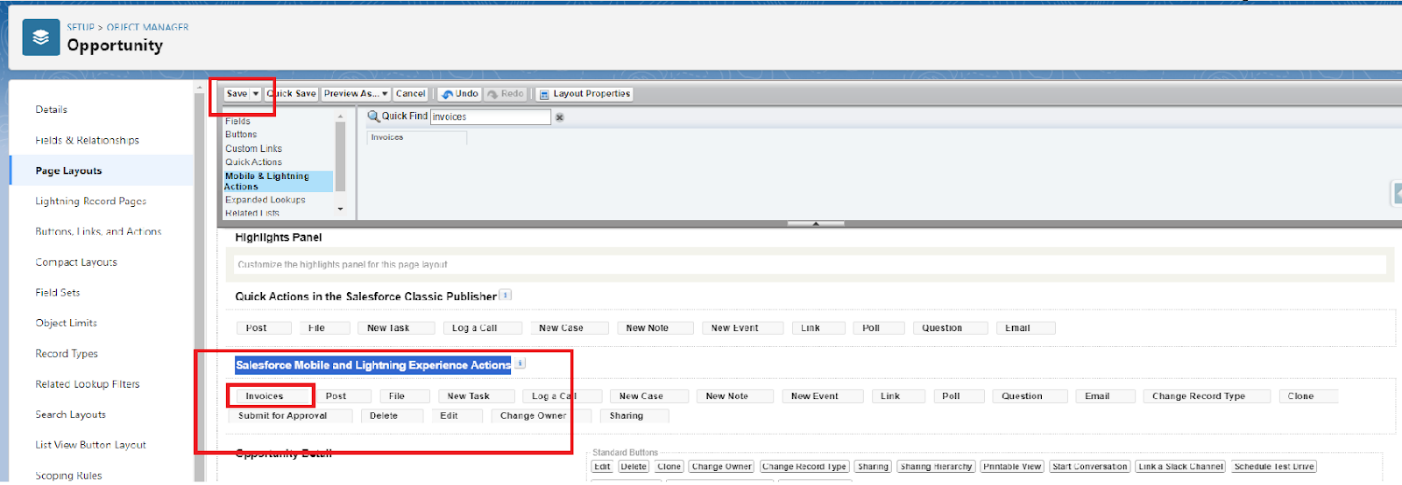
**Deploy Component:**

1. Right-click on the component folder, and select SFDX: Deploy Source to Org to deploy the component to the org.  
   
2. Once the deployment is complete, you will see the below-highlighted message in the output tab  
   

### Create Button To Add On Opportunity

1. To add the newly created component to the view, Go to Salesforce Setup
2. Click on Object Manager
3. Search  Opportunity and Click on it .
4. click on Button Links and Action.
5. click on the New Action.  
   
6. Select Action type as Lightning Web Component
7. Select the InvoiceOpportunity component
   1. Label :- Invoices
   2. Name :- Invoices
8. As given on below image  
   
9. Click on Save and your action Button is Ready.

**Add InvoiceOpportunity Into Opportunity Record Page**

1. On Opportunity Object Manager Click on Page layout.
2. Click on OpportunityLayout.
4. Click on Mobile And Lightning Action as show on below Image 
5. Search for invoice on Quick Find.
6. Drag and Drop the Invoice into Salesforce Mobile and Lightning Experience Actions.

Click on Save.