PROJECT REPORT

Naan Mudhalvan – Salesforce Developer

Project Title:

Sales Automobile using Salesfoce CRM

Team Members:

Name	NM ID	32 digit NM ID
Prasanth A (T.L)	au710021104020	8CA0C23B1C5773D835A471A13A561139
Varun Kumar S	au710021104035	1889915CCF711CD8792496C047FE8418
Amarnathan V	au710021104003	B7373C9B3055A0385254F334EDA80F3F
Subash S	aut710021104702	6632D80A341641E84A74FA0C8F8A1BE1

ANNA UNIVERSITY
REGIONAL CAMPUS COIMBATORE

Sales Automobile Using Salesforce CRM

Project Overview:

This project focuses on developing a **Salesforce CRM solution** tailored to streamline and automate the automobile sales process. It addresses the challenges of managing vast amounts of sales data, maintaining accurate inventory records, and automating routine business processes in the automobile industry. The primary goal is to implement a solution that utilizes Salesforce's robust features, such as custom objects, Apex automation, and Lightning Web Components (LWC), to improve sales management.

The solution aims to:

- Enhance **data accuracy** and **sales productivity**.
- Provide **real-time visibility** into sales metrics.
- Improve the **overall user experience** for the sales team.
- Align with the broader objectives of the organization, ensuring better tracking of customer interactions and quicker sales cycle times.

Project Description:

The Salesforce CRM implementation for automobile sales streamlines the entire sales process, enhancing efficiency and customer satisfaction. Through this system, sales teams can manage leads, track customer interactions, and automate follow-ups. It enables comprehensive customer profiling, allowing for personalized marketing strategies and targeted campaigns. The platform facilitates inventory management, ensuring real-time updates on available vehicles and their specifications. Integration with marketing tools enables seamless communication and lead nurturing. Additionally, the system provides insightful analytics, empowering decision-making by identifying sales trends and forecasting demand. Overall, the Salesforce CRM for automobile sales optimizes operations, fosters customer relationships, and drives revenue growth within the automotive industry.

Short Description:

This report is to provide a comprehensive guide on creating and managing an automobile sales database using Salesforce CRM. This involves setting up a developer account, creating and managing objects, and configuring tabs, fields, and relationships to streamline data access and improve sales management in the automobile industry. This process is structured to ensure efficient and systematic handling of automobile sales data within Salesforce.

Objectives

Business Goals

- **Optimize Sales Operations**: Simplify processes to reduce manual tasks and speed up sales.
- **Increase Data Transparency**: Enable clear tracking of customer information, inventory, and invoices.
- **Enhance Sales Insights**: Allow management to gain actionable insights from sales data for strategic planning.
- **Boost Customer Satisfaction**: Improve accuracy and speed of transactions, ensuring a smooth customer experience.

Specific Outcomes

- **Custom Objects for Key Data**: Define custom objects such as **Opportunity Automobile** (tracking each sales opportunity) and **Invoice** to manage payment data.
- **Automation through Apex Triggers**: Set up automated processes for key tasks, such as creating invoices and checking inventory availability.
- **User-Friendly Interface**: Create a **Lightning Web Component** (LWC) for displaying real-time invoice data and streamline user navigation with custom tabs and layouts.
- **Insightful Reports and Dashboards**: Develop reports to analyze sales trends, performance metrics, and inventory levels.

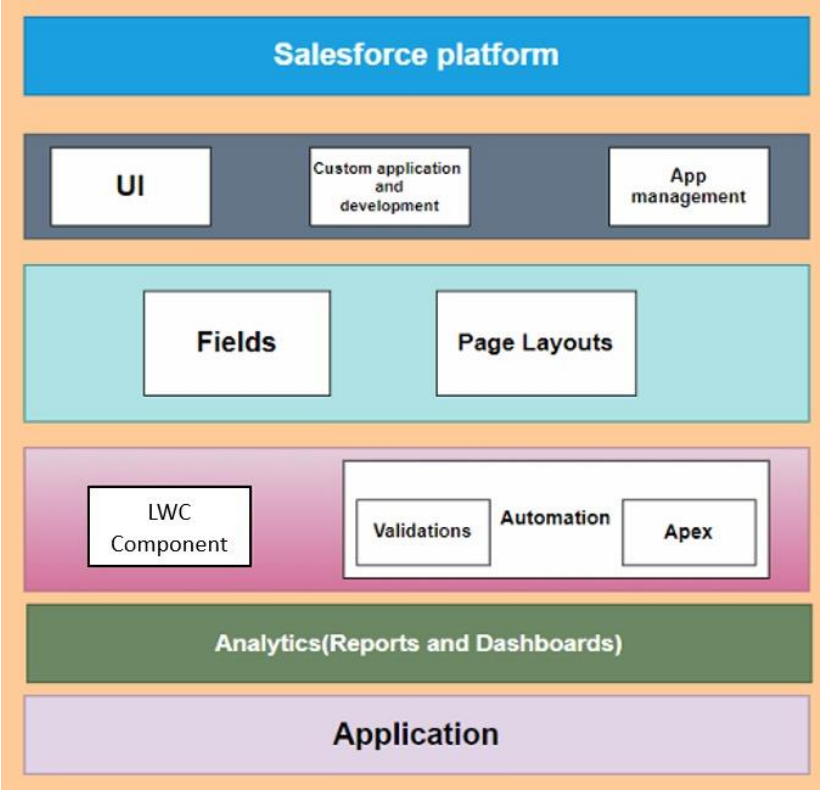
Salesforce Key Features and Concepts Utilized

This project leverages several critical Salesforce functionalities to address the business needs:

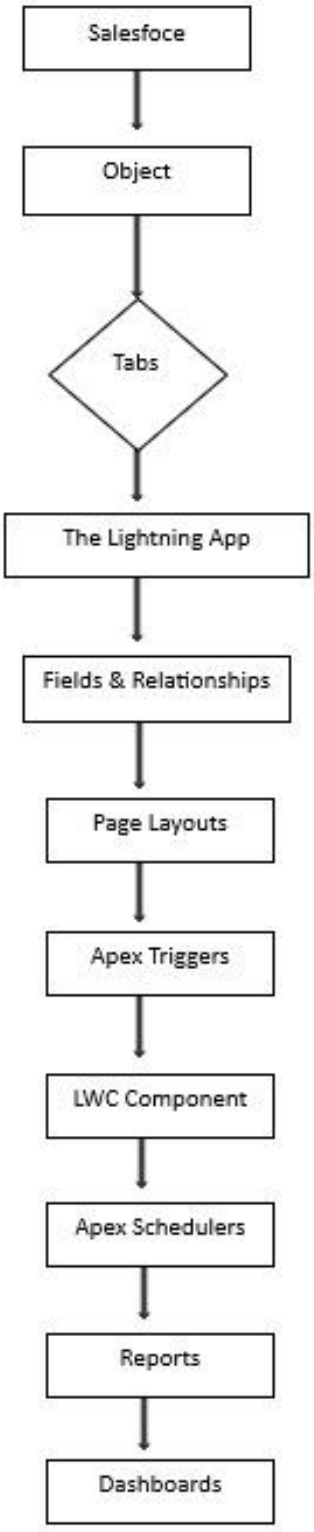
- **Custom Objects**: Custom objects like **Opportunity Automobile** and **Invoice** store and manage essential sales and inventory data. These custom objects help tailor Salesforce to meet the specific requirements of the automobile sales industry.
- **Relationships**: Implemented **master-detail** and **lookup relationships** to connect data meaningfully. For instance, an Opportunity record is related to Automobile Information, ensuring the sales team can view relevant automobile details alongside opportunities.
- **Apex Triggers**: Utilized triggers to automatically update records based on specific conditions, such as generating an invoice when an opportunity is marked as "Closed Won" or deducting stock when an automobile is sold.

- **Lightning Web Components (LWC)**: Developed a custom LWC to display real-time invoice information linked to each opportunity, enhancing data visibility for the sales team.
- **Reports and Dashboards**: Configured Salesforce's reporting tools to create dashboards that track sales performance, opportunity stages, and inventory data. This functionality gives management a holistic view of the sales pipeline and stock levels at any time.

Technical Architecture



Project Flow:



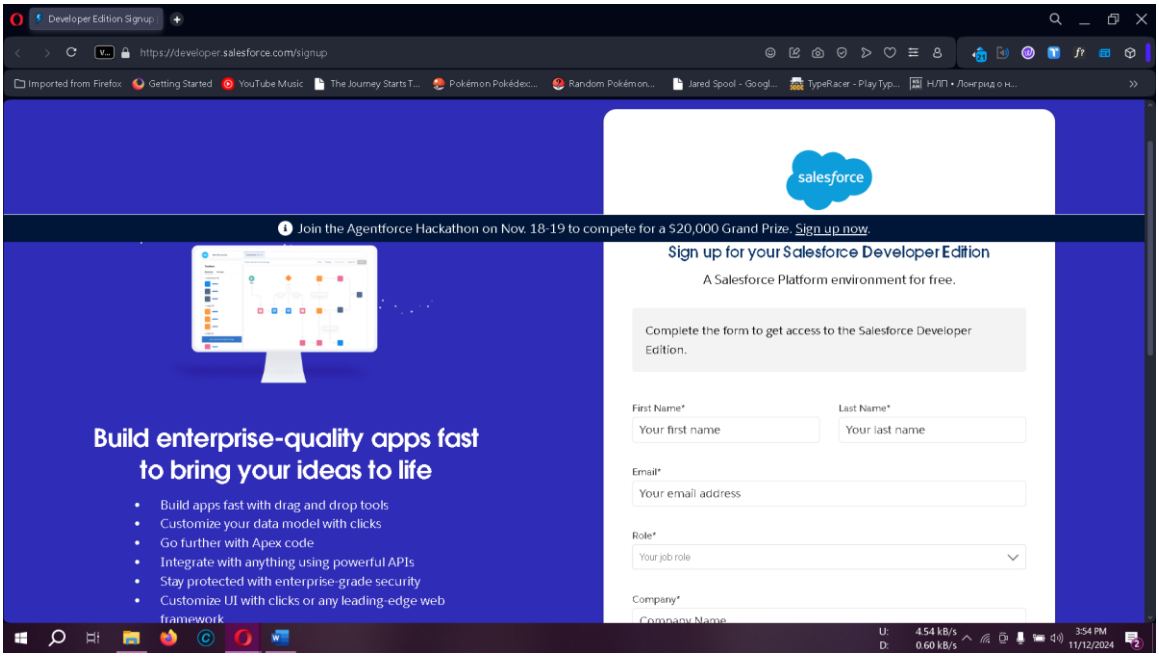
Detailed Steps to Solution Design

Milestone 1- Salesforce

Developer Account Creation

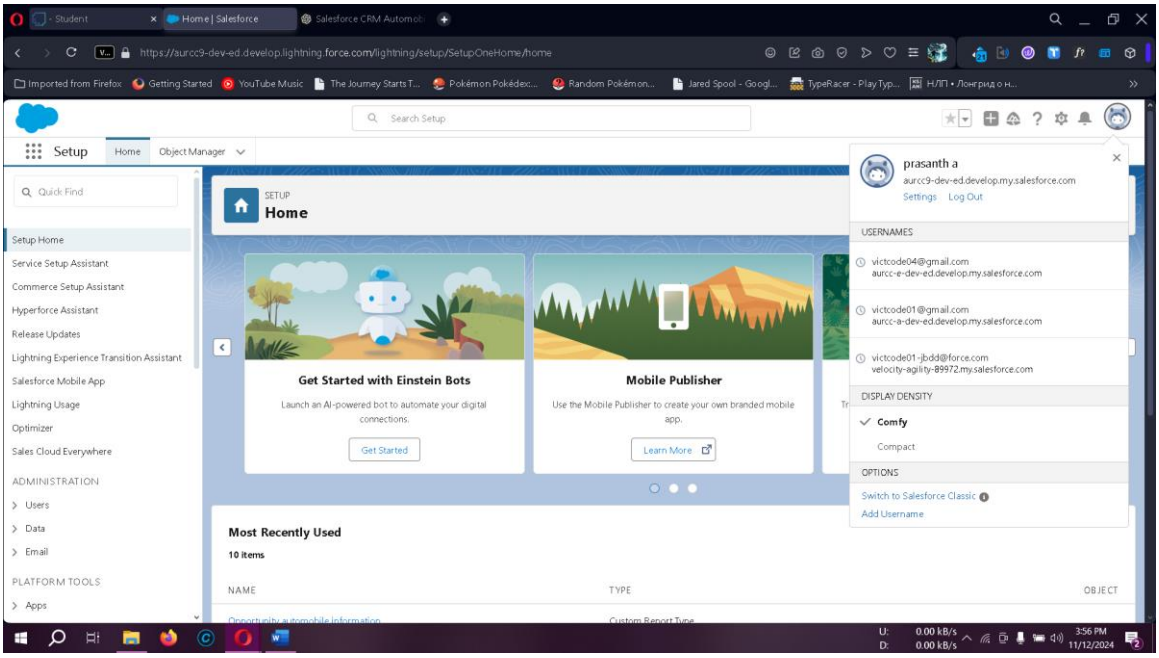
To start working with Salesforce CRM, a developer account is essential. Follow these steps to create an account:

- Go to Salesforce Developer Sign-Up.
- Enter your **First and Last Name**, **Email**, and set **Role** as "Developer."
- Input your **Company** (College Name), **Country** (India), **Postal Code**, and **Username** (formatted as username@organization.com).
- Click **Sign Me Up** after filling out the form.



Account Activation

- Open the inbox of the email used for registration, locate the Salesforce verification email, and click **Verify Account**.
- Set a password, choose a security question, and log into your Salesforce account to access the setup page.



Milestone 2 - Objects in Salesforce

Salesforce objects function as database tables for storing and organizing data relevant to the organization.

- **Standard Objects**: Provided by Salesforce by default (e.g., Accounts, Contacts).
- **Custom Objects**: User-defined objects to store unique organizational data.

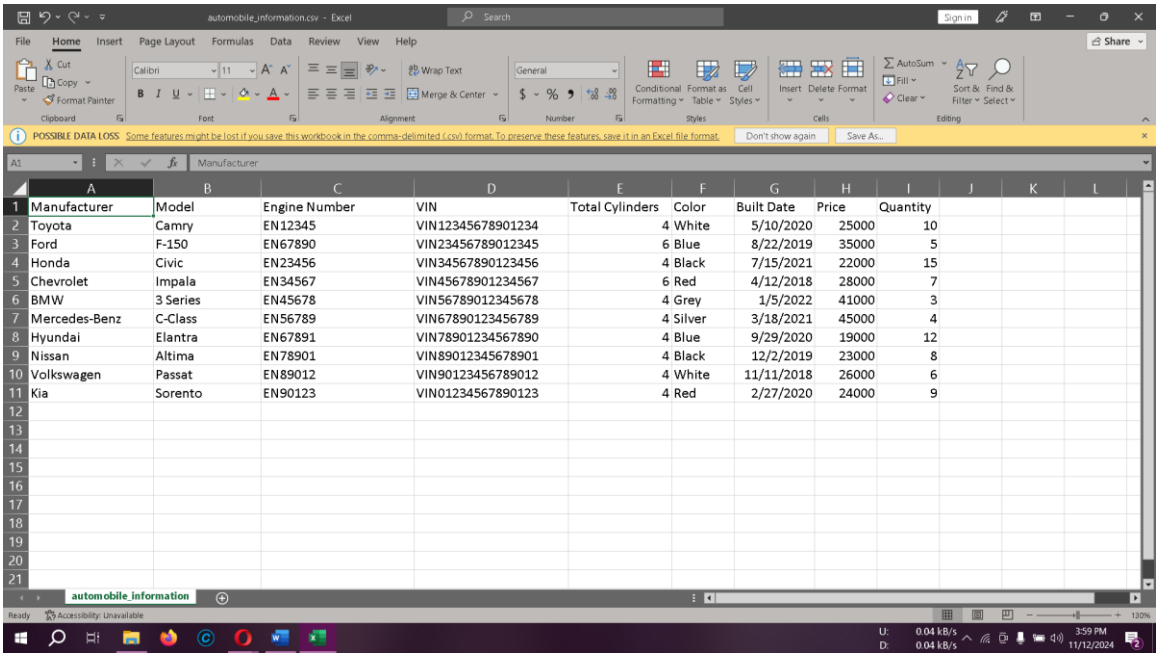
Use Case

Creating custom objects such as Automobile Information and Invoice enables efficient data handling, process automation, and tailored reporting for automobile sales.

Creating Objects

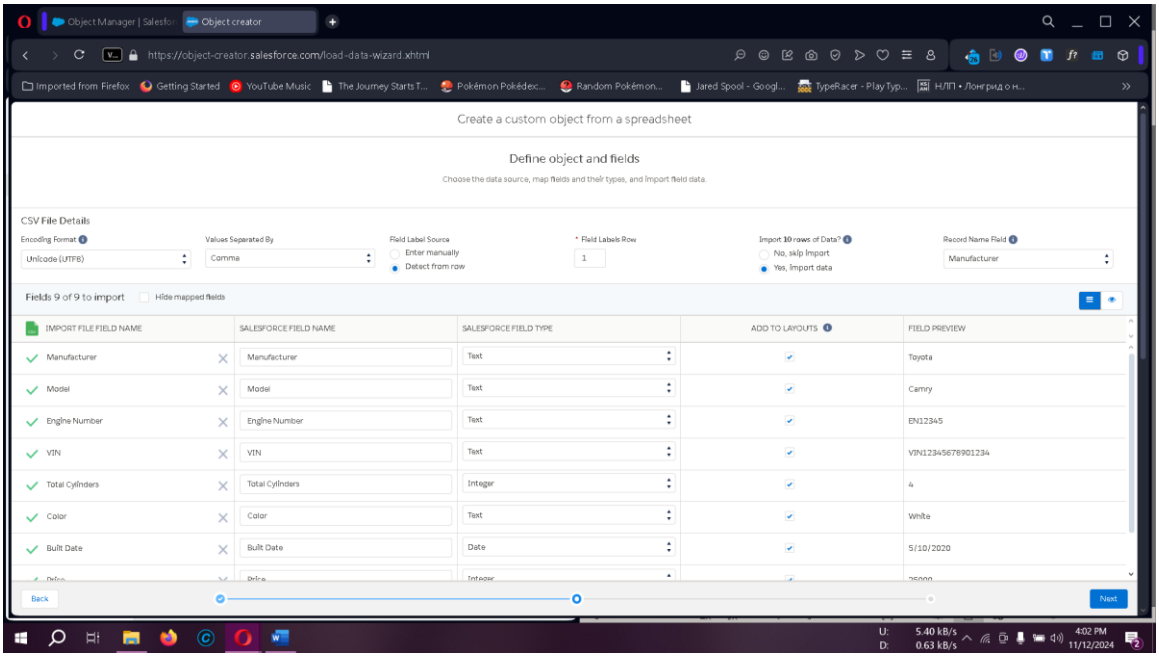
Automobile Information Object

1. Download and open **AutomobileInformation.csv**.
2. Log into Salesforce, navigate to **Setup > Object Manager**.
3. Select **Custom Object from Spreadsheet**, login if prompted, upload the CSV file, and map fields with the correct data types.
4. Follow prompts and finish to complete object creation.



automobile_information.csv - Excel

Manufacturer	Model	Engine Number	VIN	Total Cylinders	Color	Built Date	Price	Quantity
Toyota	Camry	EN12345	VIN12345678901234	4	White	5/10/2020	25000	10
Ford	F-150	EN67890	VIN23456789012345	6	Blue	8/22/2019	35000	5
Honda	Civic	EN23456	VIN34567890123456	4	Black	7/15/2021	22000	15
Chevrolet	Impala	EN34567	VIN45678901234567	6	Red	4/12/2018	28000	7
BMW	3 Series	EN45678	VIN56789012345678	4	Grey	1/5/2022	41000	3
Mercedes-Benz	C-Class	EN56789	VIN67890123456789	4	Silver	3/18/2021	45000	4
Hyundai	Elantra	EN67891	VIN78901234567890	4	Blue	9/29/2020	19000	12
Nissan	Altima	EN78901	VIN89012345678901	4	Black	12/2/2019	23000	8
Volkswagen	Passat	EN89012	VIN90123456789012	4	White	11/11/2018	26000	6
Kia	Sorento	EN90123	VIN01234567890123	4	Red	2/27/2020	24000	9



Create a custom object from a spreadsheet

Define object and fields

Choose the data source, map fields and their types, and import field data.

CSV File Details

Fields 9 of 9 to import

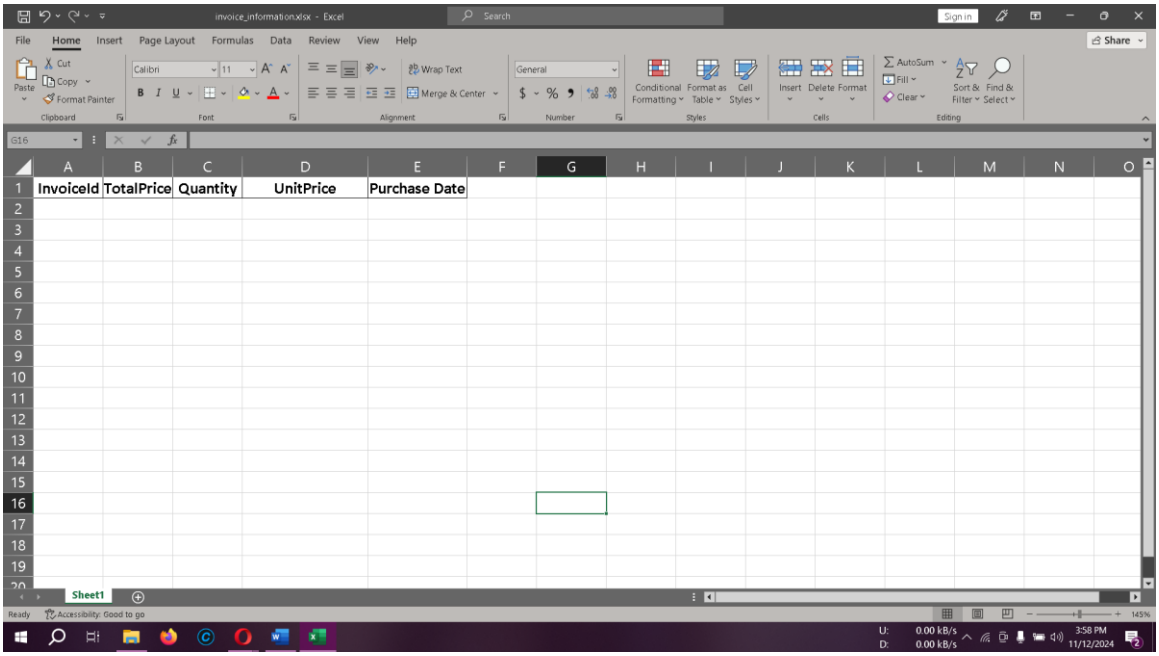
IMPORT FILE FIELD NAME	SALESFORCE FIELD NAME	SALESFORCE FIELD TYPE	ADD TO LAYOUTS	FIELD PREVIEW
Manufacturer	Manufacturer	Text	✓	Toyota
Model	Model	Text	✓	Camry
Engine Number	Engine Number	Text	✓	EN12345
VIN	VIN	Text	✓	VIN12345678901234
Total Cylinders	Total Cylinders	Integer	✓	4
Color	Color	Text	✓	White
Built Date	Built Date	Date	✓	5/10/2020
Price	Price	Integer	✓	25000

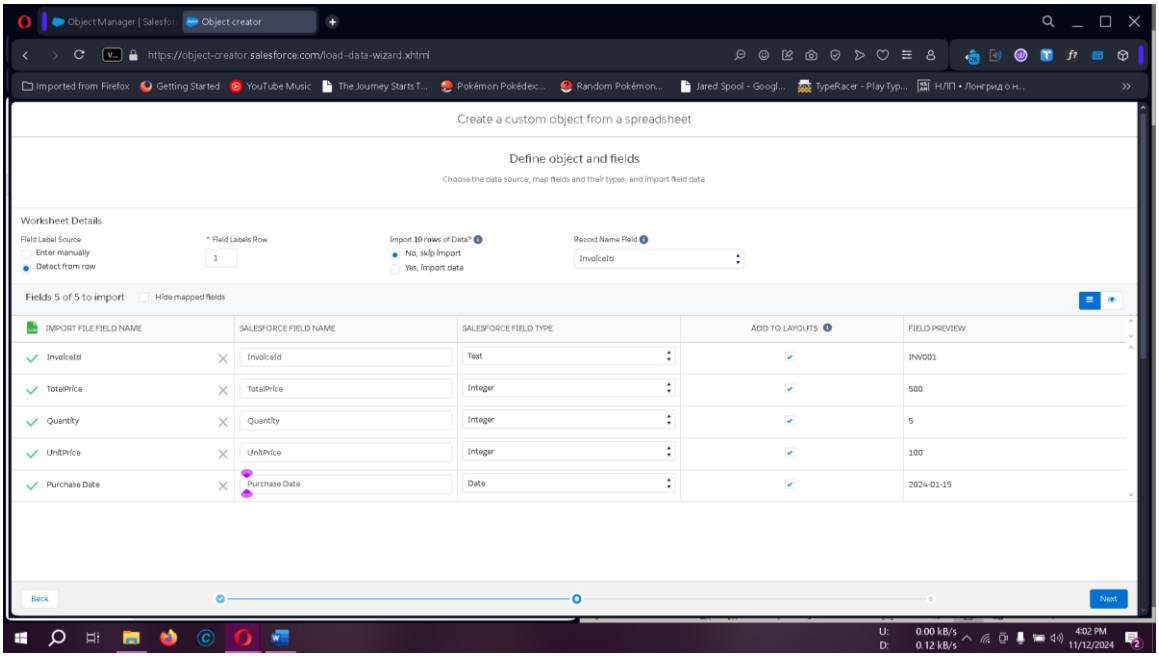
Back

Next

Invoice Object

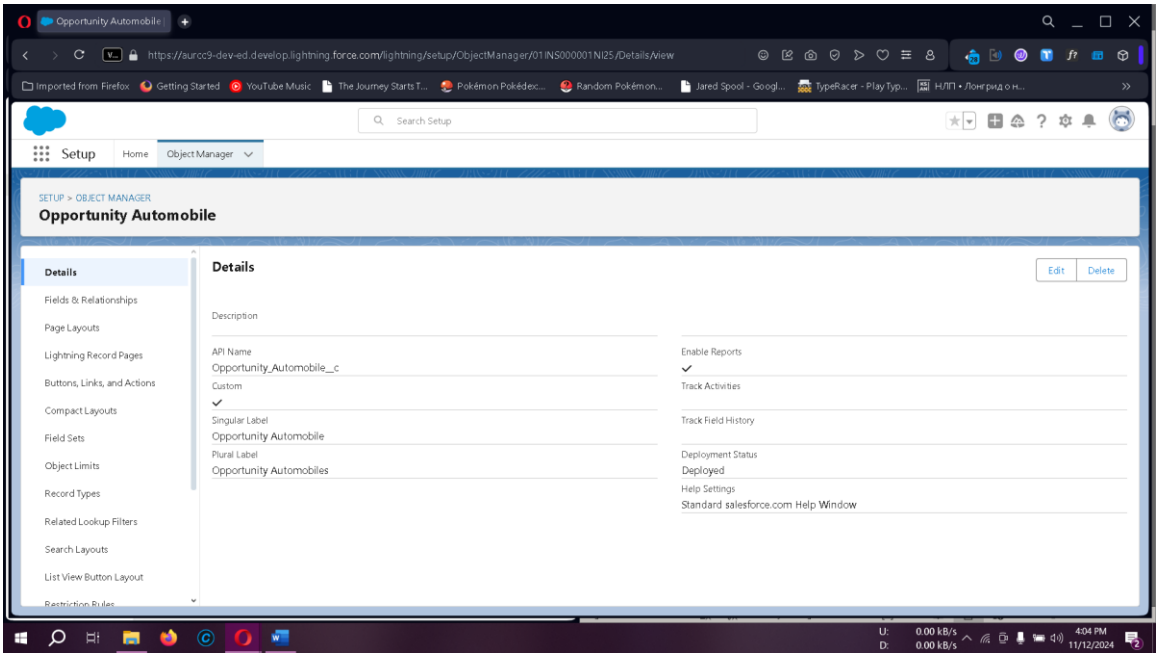
Repeat the steps above with the **Invoice CSV** file, ensuring proper field mapping.





Opportunity Automobile Object

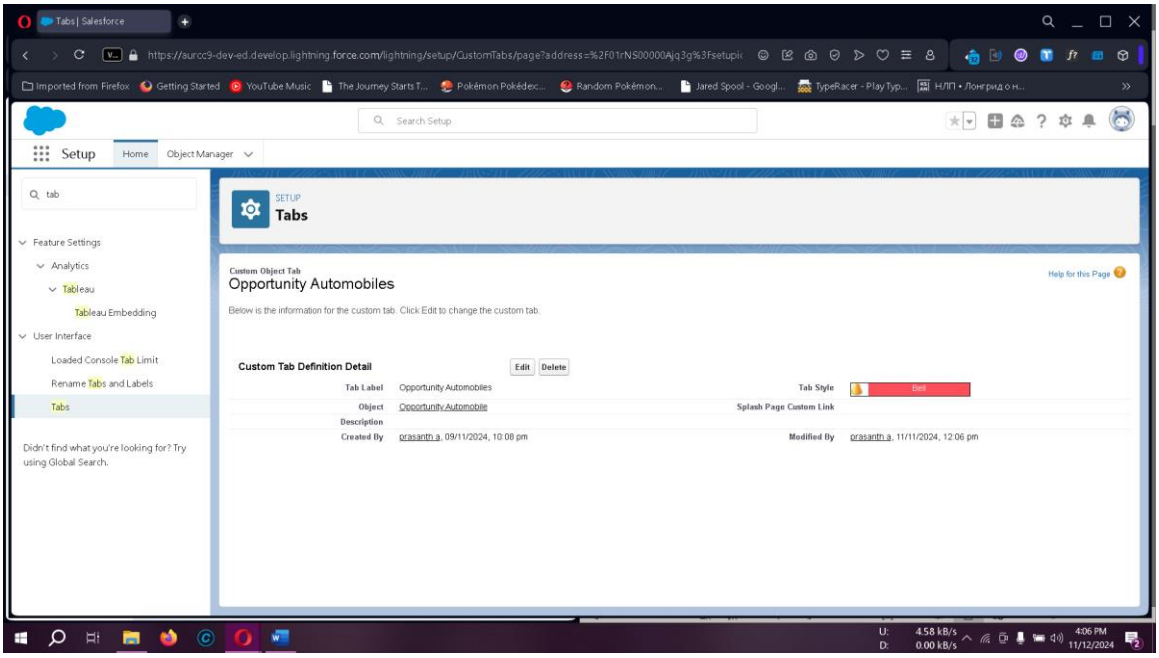
- From **Setup > Object Manager**, select **Create > Custom Object**.
- Configure labels, record names, and data types (e.g., **Auto Number** with display format OA-{0000}, starting at 1).
- Enable reporting and search functionality, then save the object.



Milestone 3 - Tabs

Tabs in Salesforce provide a user interface for managing and viewing records.

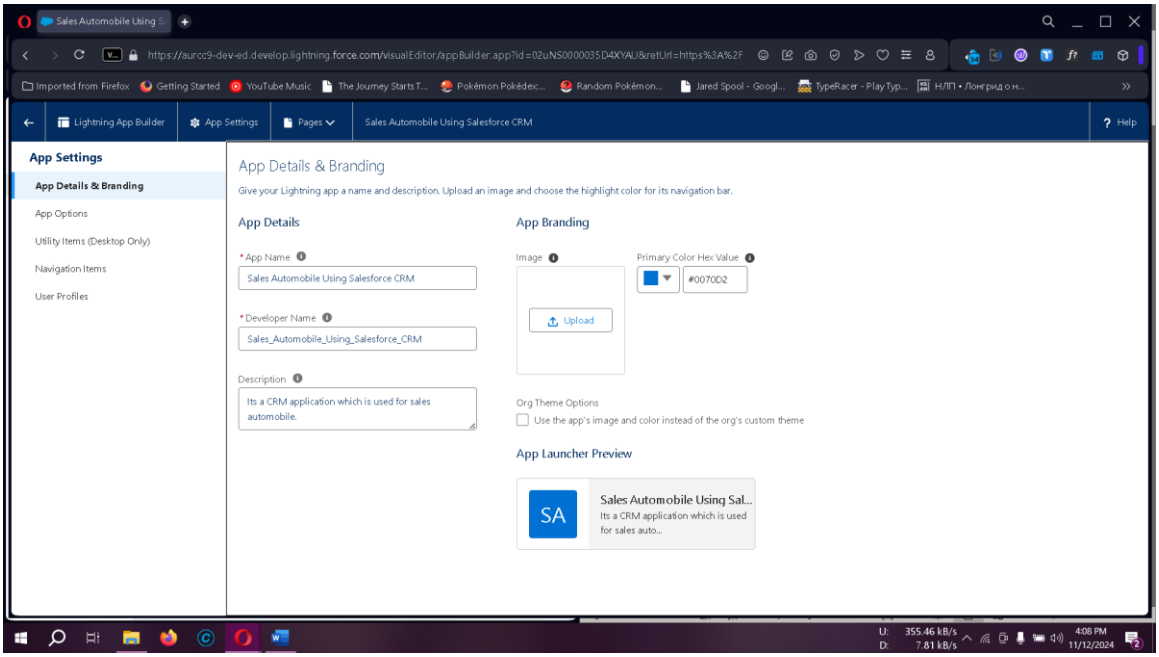
1. **Types of Tabs**:
   - **Custom Tabs**: Specific to custom objects.
   - **Web Tabs**: Display web content.
   - **Visualforce Tabs**: Display Visualforce pages.
   - **Lightning Component Tabs**: Add Lightning components to the navigation.
   - **Lightning Page Tabs**: Add Lightning Pages to mobile app navigation.
2. **Creating a Custom Tab**
   - From **Setup**, search **Tabs** and select **New (Custom Object Tab)**.
   - Choose **Opportunity Automobile** and complete the setup.



Milestone 4 - The Lightning App

Lightning Apps enable organized navigation, grouping objects, tabs, and settings into a bundle.

1. **Creating the Sales Automobile Lightning App**
   - From **Setup**, search **App Manager** and select **New Lightning App**.
   - Provide **App Name** (Sales Automobile Using Salesforce CRM) and branding details.
   - Add navigation items (e.g., Account, Contact, Opportunities, etc.).
   - Assign profiles to complete the setup.



Milestone 5 - Fields & Relationships

Fields in Salesforce store valuable data and support relational database functionalities.

1. **Types of Fields**:
   - **Standard Fields**: Predefined by Salesforce and include basic fields like Created By, Owner, Last Modified.
   - **Custom Fields**: User-defined fields tailored to specific organizational needs.
2. **Field Creation**:
   - Define fields such as Quantity, Unit Price, and Total Price in **Opportunity Automobile** using appropriate data types (e.g., Number, Formula).
   - Use formulas to calculate fields (e.g., Unit_Price__c * Quantity__c for Total Price).

Use Case

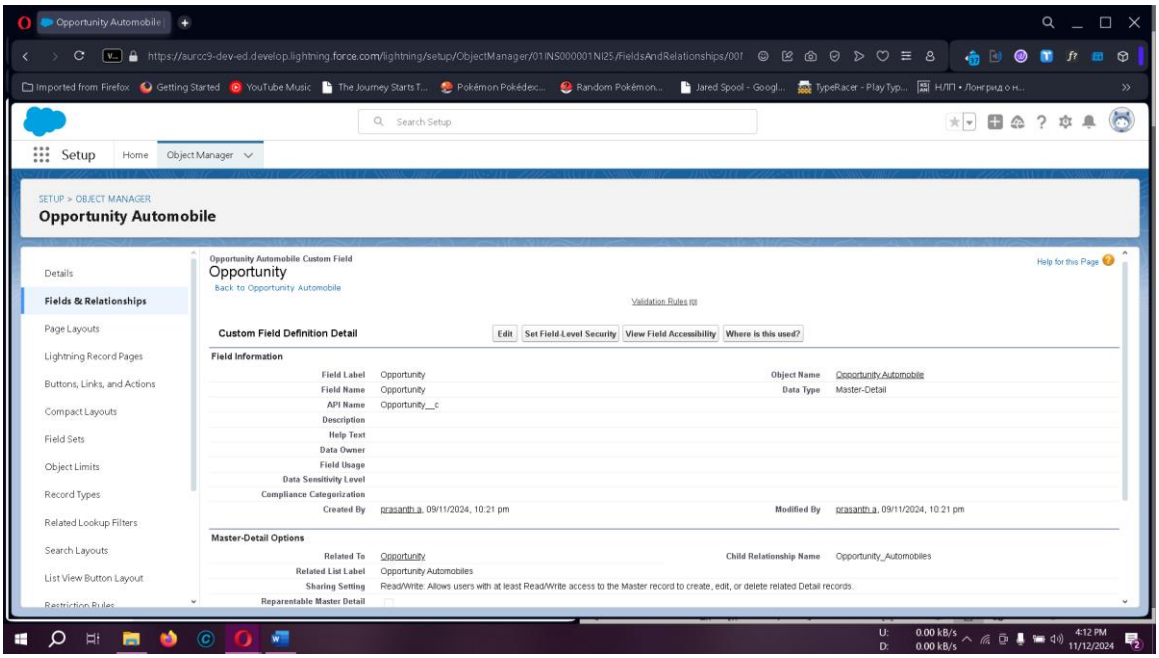
Creating fields for objects allows storing structured information, optimizing record accessibility, and enabling automated data calculations.

Field Relationships

Establishing relationships between fields enables data consistency and organization within related objects.

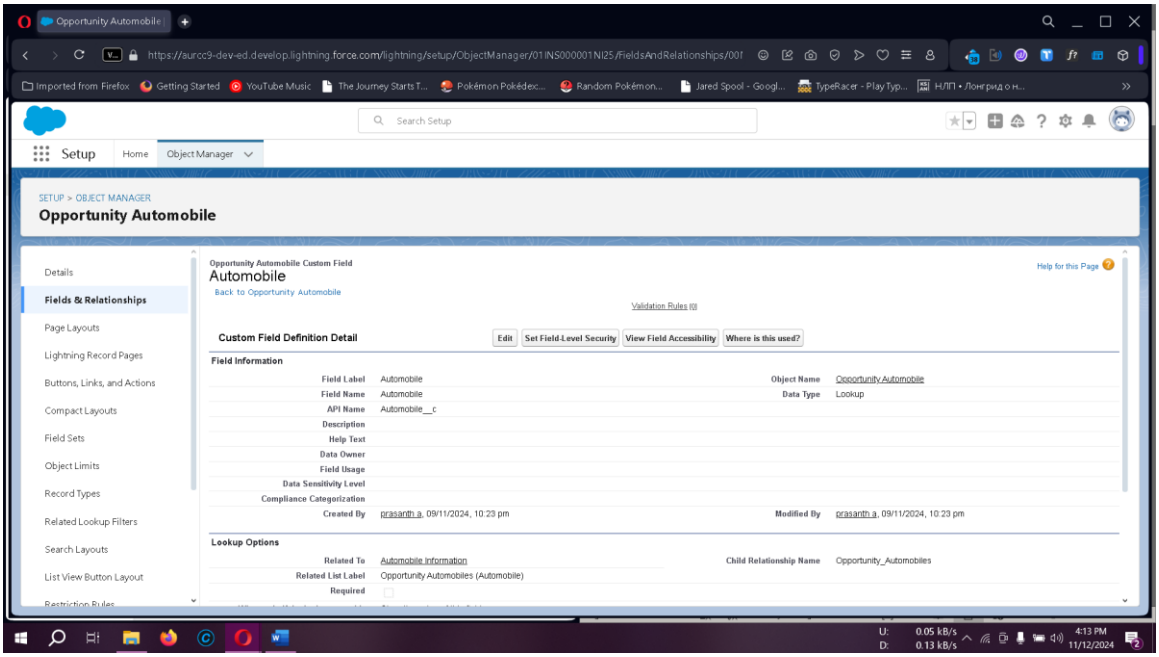
Creating Master Detail Relationship in Opportunity Automobile

- From **Object Manager**, select **Fields & Relationships > New** and choose **Master-Detail Relationship**.
- Link to the **Opportunity** object for data hierarchy.



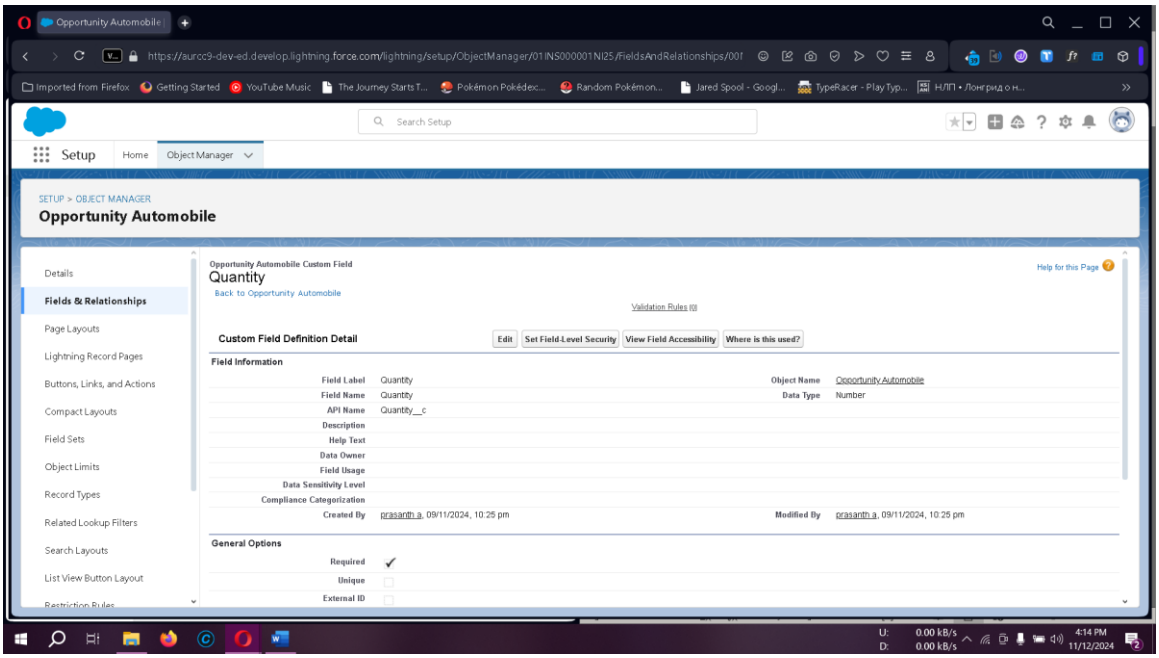
Creating Lookup Field for Automobile

- Use **Lookup Relationship** to connect **Automobile Information** to **Opportunity Automobile**.



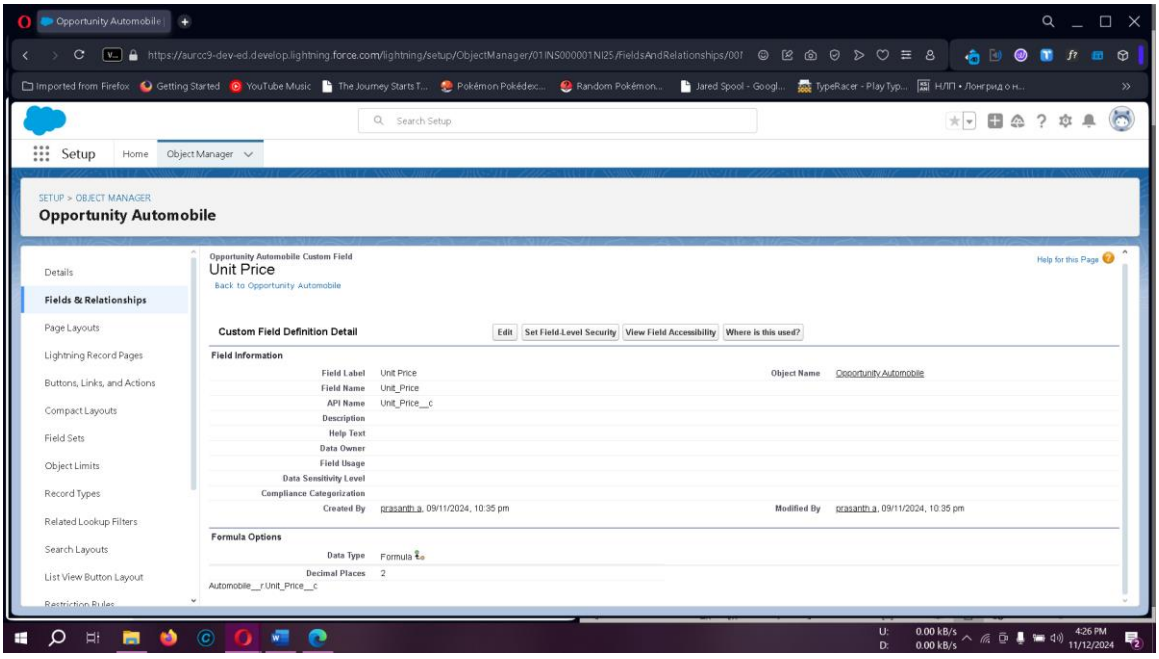
Creating Quantity Number Field in Opportunity Automobile Object

- **Set the** Data Type **to** Number **and proceed with** Next**.**
- **Label the field as** Quantity **with the same Field Name, and mark it as** Required**.**
- **Complete the process by clicking** Next > Next > Save **.**



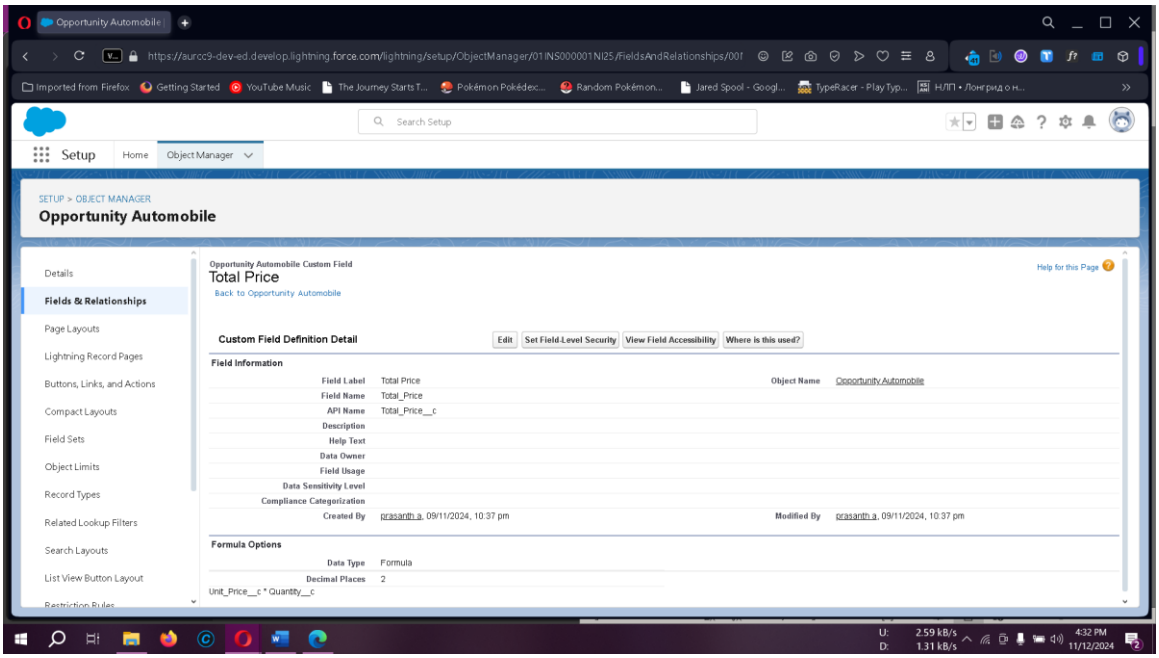
To create a formula field in the Opportunity Automobile object:

- **In** Setup > Object Manager**, locate** Opportunity Automobile**.**
- **Under** Fields & Relationships**, click** New > Data Type: Formula > Next**.**
- **Set** Field Label **and** Field Name **to** Unit Price**, select** Currency **as the formula return type, and set decimal places to two.**
- **Enter** Automobile__r.Price__c **in** Advanced Formula > Check Syntax**.**
- **Click** Next > Next > Save & New**.**



To create a formula field in the Opportunity Automobile object:

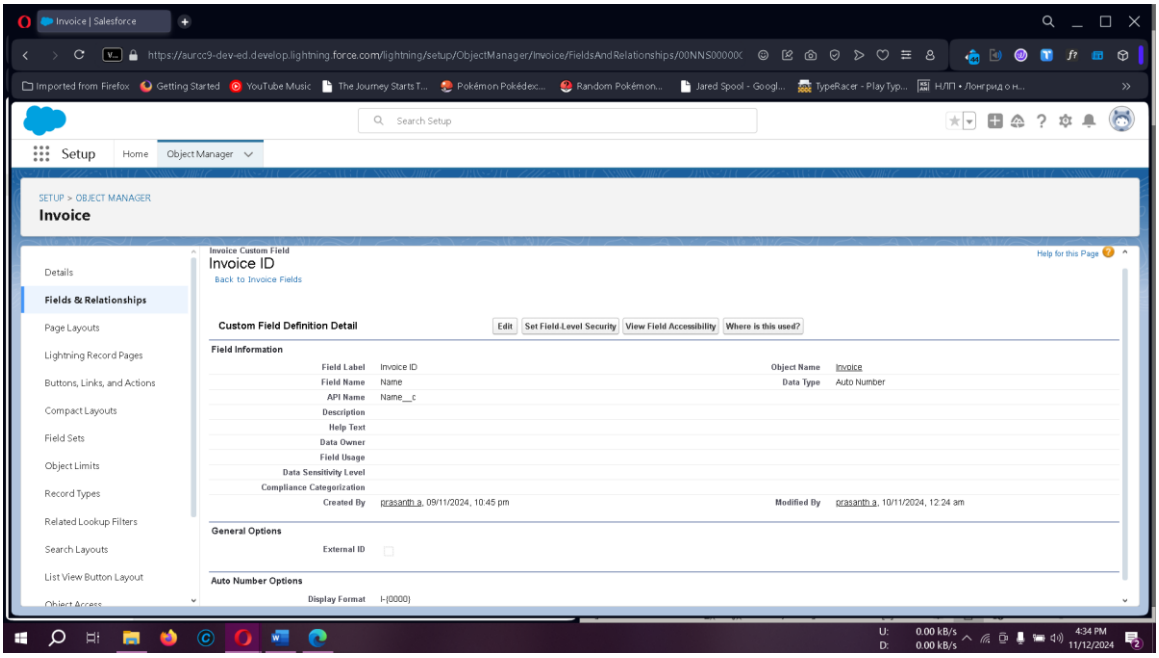
- In Setup > Object Manager, locate Opportunity Automobile.
- Go to Fields & Relationships > New > Data Type: Formula > Next.
- Set Field Label and Field Name to Total Price, select Currency as the return type, and set decimal places to two.
- In Advanced Formula, enter Unit_Price__c * Quantity__c > Check Syntax.
- Click Next > Next > Save.



Updating Fields

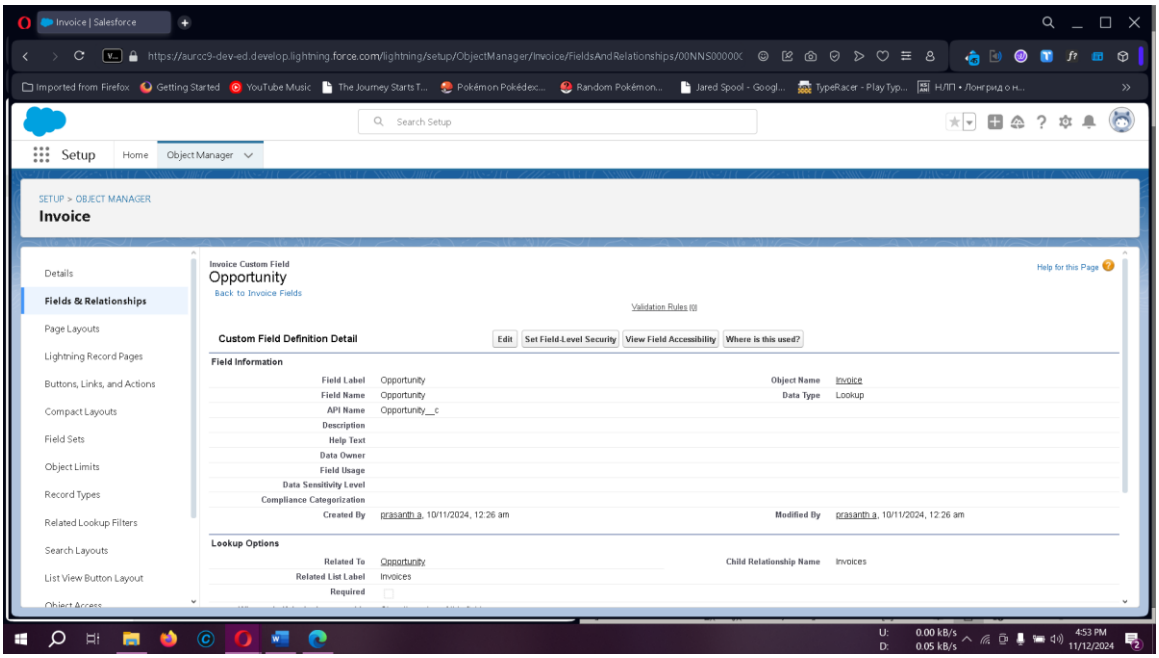
To update fields (e.g., Invoice ID in the Invoice Object):

- Navigate to **Object Manager > Invoice > Fields & Relationships**.
- Set **Data Type** to **Auto Number** with a display format (e.g., I-{0000}).



Creating Master-Detail Relationship Field in the Invoice Object

In the **Invoice** object, create a field named **Opportunity** with a **Master-Detail Relationship** data type, establishing a connection to the **Opportunity** object.



Milestone 6 - Page Layouts in Salesforce

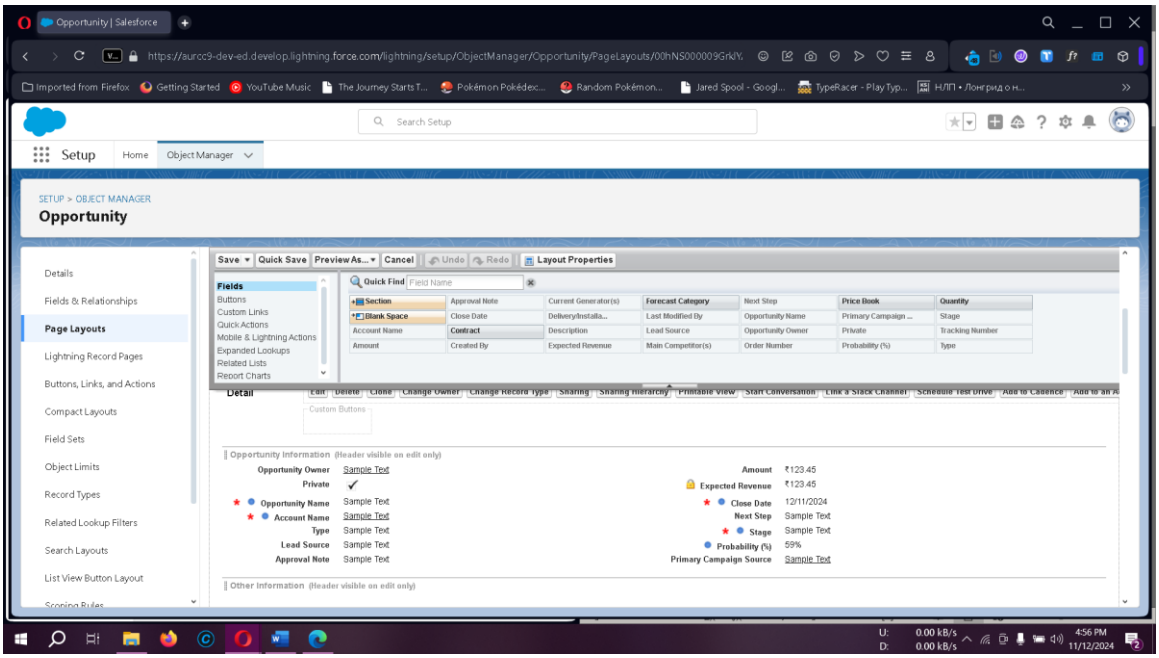
In Salesforce, **Page Layouts** allow you to design and organize detail and edit pages for both standard and custom objects, giving you control over the appearance of fields, related lists, and custom links. They help ensure an organized display of information on pages, enhancing usability and readability for users.

Use Case: Organizing Opportunity and Automobile Information Layouts

After setting up the data model structure, the detail and edit pages for records might appear cluttered. A clean, organized layout improves data accessibility and provides a better user experience. We will edit the page layout for the **Opportunity** and **Automobile Information** objects.

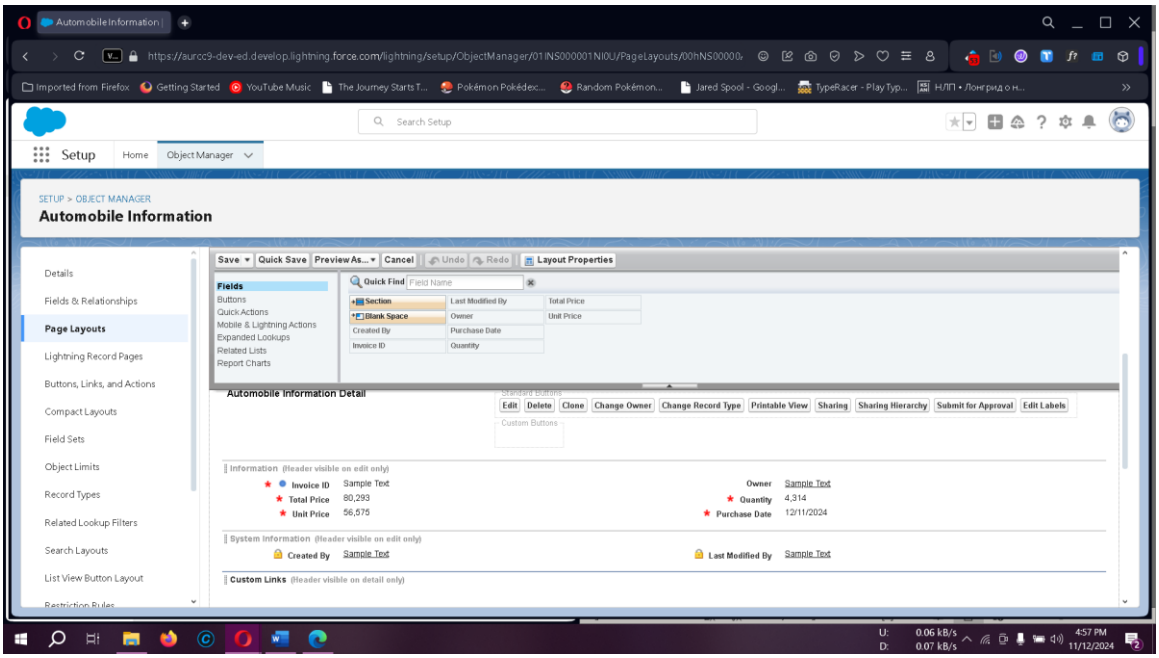
Editing the Page Layout for the Opportunity Object

1. **Navigate to the Setup**:
   - Go to **Setup** > **Object Manager**.
   - In the search bar, select **Opportunity Layout**.
2. **Access Page Layouts**:
   - Click **Page Layouts** in the left panel.
   - Select **Opportunity Layouts**.
3. **Customize Fields**:
   - Locate the **Opportunity Detail** section.
   - Select the **Properties icon** for the **Account Name** field.
   - Check the **Required** box and confirm by clicking **OK**.
4. **Save the Layout**:
   - Click **Save** to apply changes.



Editing the Page Layout for Automobile Information

1. **Navigate to the Setup**:
   - Go to **Setup** > **Object Manager**.
   - Search for **Automobile Information**.
2. **Access Page Layouts**:
   - Click **Page Layouts** and select **Automobile Information Layout**.
3. **Mark Fields as Required**:
   - For each field in the Automobile Information object, mark them as required if needed by clicking the **Gear Icon** next to each field.
4. **Adjust Field Positioning**:
   - Organize the fields in a visually appealing layout for clarity and readability.
5. **Save the Layout**:
   - Click **Save** to finalize changes.



Milestone 7 - Implementing Apex Triggers for Automation

Apex Triggers enable custom actions before or after changes to Salesforce records, helping automate complex business processes. Below are some use cases with trigger implementations for the Opportunity and Automobile Information objects.

Use Case 1: Updating Automobile Quantity upon Opportunity Closure

- **Objective**: When an **Opportunity** is marked as **Closed Won**, deduct the specified **Opportunity Automobile quantity** from the available stock in the **Automobile Information** object.
- **Class and Trigger**: The OpportunityHandlerClass class and OpportunityTrigger trigger manage the logic to update quantities accordingly.

Class 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 Code:

```
trigger OpportunityTrigger on Opportunity (before update, after update) {
    if (Trigger.isBefore && Trigger.isUpdate) {
        OpportunityHandlerClass.opportunityAutomobileQuantity(Trigger.new, Trigger.oldMap);
    }
}
```

Use Case 2: Validating Automobile Quantity Availability

- **Objective**: Ensure that the **quantity requested** in an Opportunity does not exceed the available stock in the **Automobile Information** object.
- **Class and Trigger**: The OpportunityAutomobileHandler class and OpportunityAutoMobileTrigger trigger handle validation.

Class 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 Code:

```
    trigger OpportunityAutoMobileTrigger on Opportunity_Automobile__c (before insert, before
update) {
        if (Trigger.isBefore && (Trigger.isInsert || Trigger.isUpdate)) {
           OpportunityAutomobileHandler.quantityErrorOnAutomobileInformation(Trigger.new);
        }
    }
```

Use Case 3: Invoice Creation upon Opportunity Closure

- **Objective**: When an Opportunity is marked as **Closed Won**, automatically generate an **Invoice** using the data from **Opportunity Automobile** records.
- **Class**: The InvoiceCreation class is called from within the **OpportunityTrigger** to create an invoice without needing a separate trigger.

Class 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 Code:

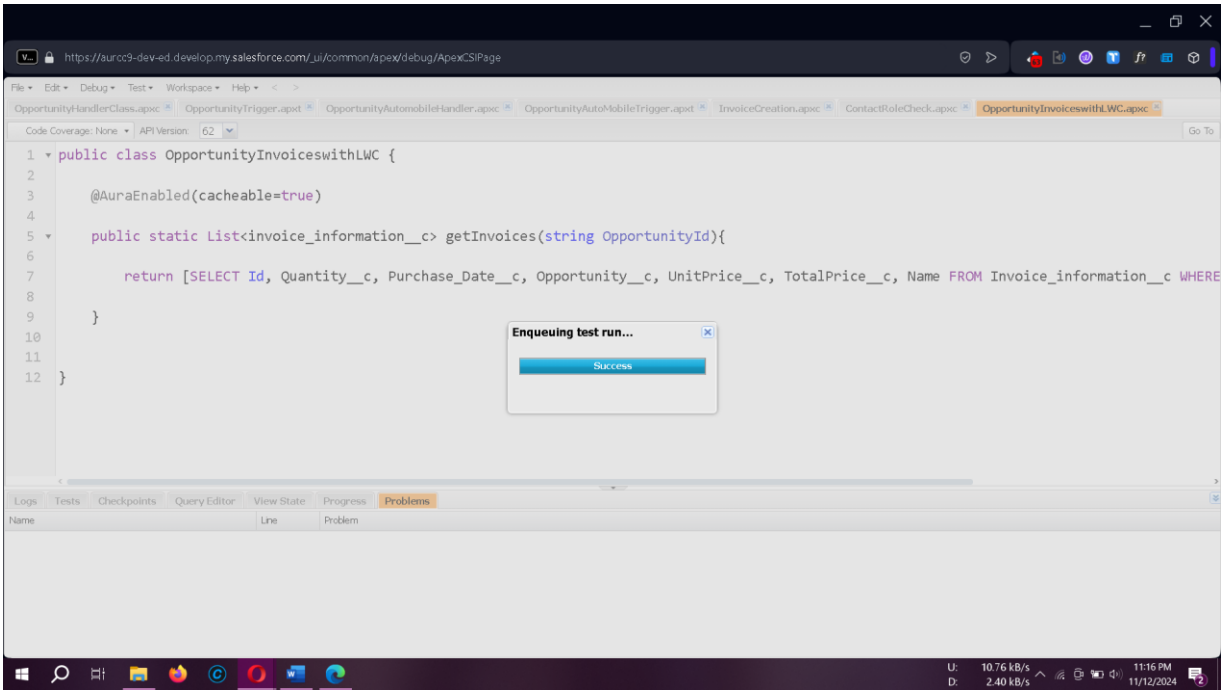
```
trigger OpportunityTrigger on Opportunity (before update, after update) {
    if (Trigger.isBefore && Trigger.isUpdate) {
        OpportunityHandlerClass.opportunityAutomobileQuantity(Trigger.new, Trigger.oldMap);
    }
    if (Trigger.isAfter && Trigger.isUpdate) {
        InvoiceCreation.OpportunityClosedwonInvoiceGeneration(Trigger.new, Trigger.oldMap);
    }
}
```

Use Case 4: Ensuring Contact Role is Assigned

- **Objective**: Before marking an Opportunity as **Closed Won**, check if a **Contact Role** is associated with it.
- **Class**: ContactRoleCheck class handles the logic for this validation, called from within **OpportunityTrigger**.

Trigger Code:

```
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);
    }
}
```

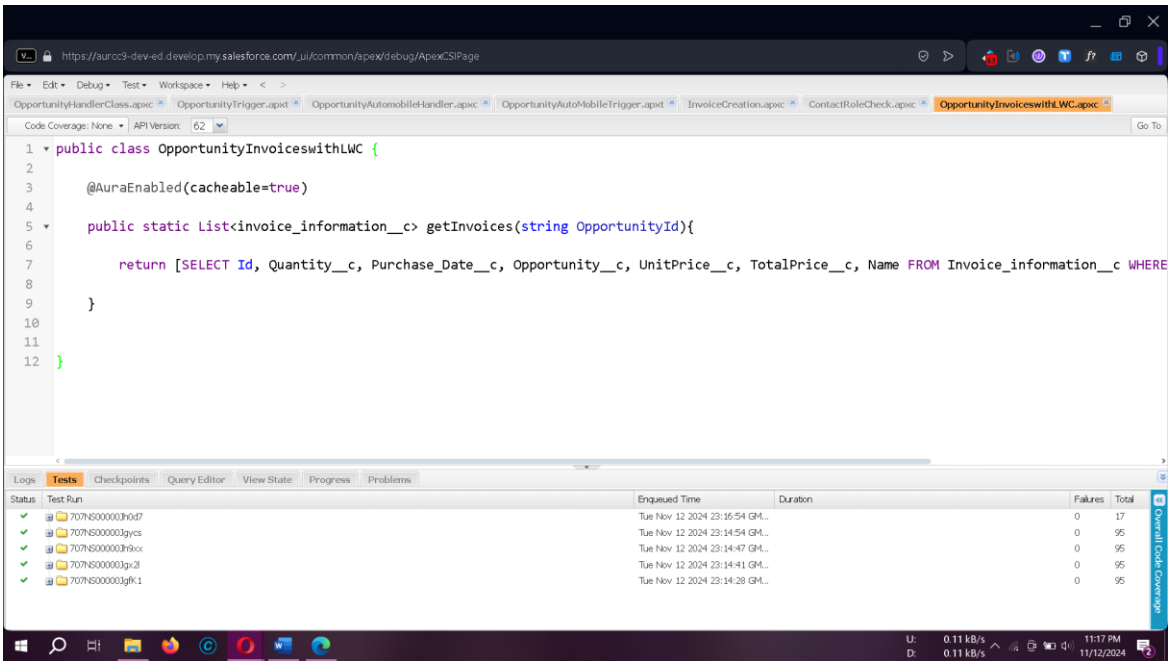


Milestone 8 - LWC Component: Creating an Apex Class to Retrieve Invoices

1. **Log into Salesforce**
   Access your Salesforce account and click the gear icon in the upper-right corner. From the dropdown menu, select **Developer Console** to open a new console window.
2. **Create a New Apex Class**
   In the Developer Console, go to **File > New > Apex Class** and name it OpportunityInvoiceswithLWC.

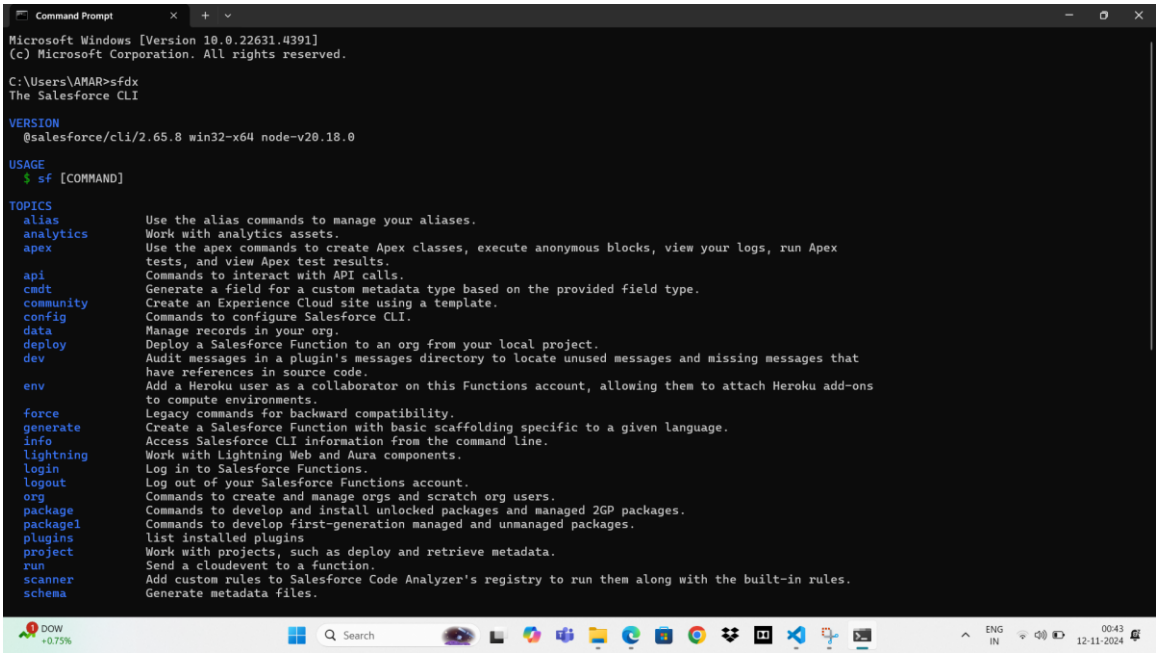
Apex Class Code:

```
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
        ];
    }
}
```



Setting Up the Salesforce CLI

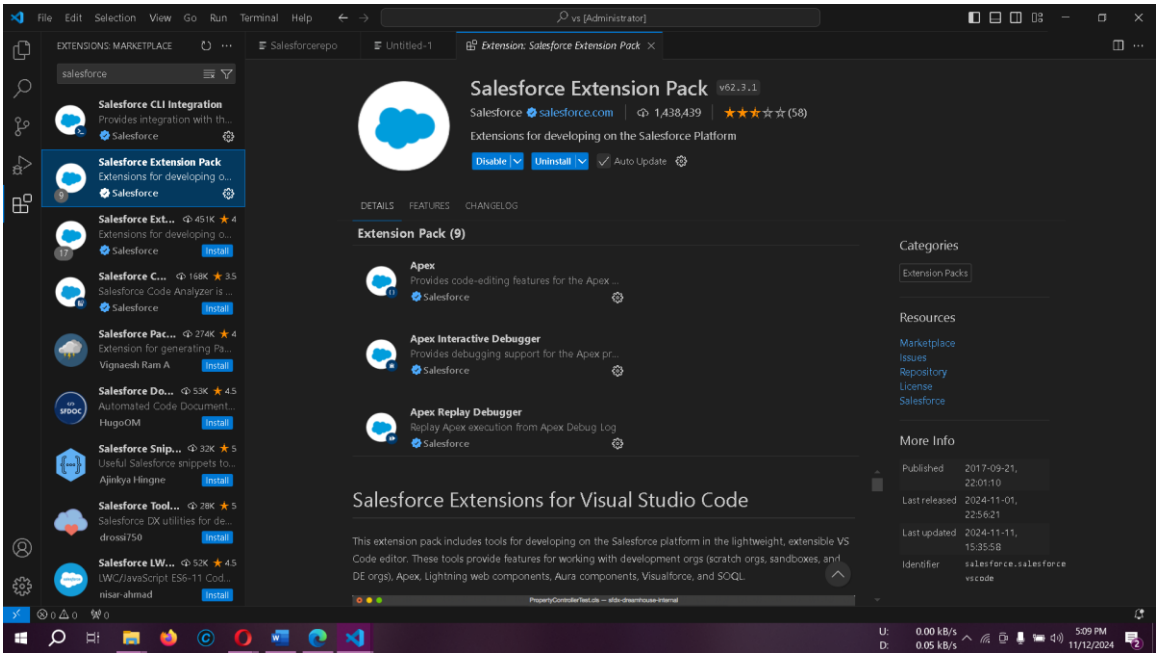
Download and Install
The Salesforce CLI is essential for streamlining development. Download and install it, and verify the installation by typing sfdx in your command prompt.



Installing and Setting Up Microsoft Visual Studio Code (VS Code)

1. **Download and Install VS Code**
   Ensure you download the version compatible with your operating system.

2. **Install the Salesforce Extension Pack**
   Open VS Code, go to Extensions, and search for **Salesforce Extension Pack**. Click **Install**.



Creating a Salesforce Project in VS Code

1. **Create Project**
   Press CTRL + SHIFT + P, type sfdx: create project with manifest, select a project template, and follow the prompts to name and save the project.
2. **Update Package.xml**
   Update your package.xml file with the following code to work with LWC components:

   xml
   Copy code

```
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>*</members>
        <name>LightningComponentBundle</name>
    </types>
    <version>55.0</version>
</Package>
```

3. **Authorize an Org**
   Establish a connection with your Salesforce org by pressing CTRL + SHIFT + P, typing sfdx: authorize an org, and selecting **Production** for the developer edition.

Creating the Lightning Web Component

1. **Create LWC Component**
   Press CTRL + SHIFT + P, type sfdx: create lightning web component, and name it InvoiceOpportunity.
2. **JS File Code for LWC**
   Copy and paste the following code into InvoiceOpportunity.js:

javascript
Copy 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}) {
        if(data) this.invoiceCollection = data;
        if(error) console.log('Error fetching data');
    }
}
```

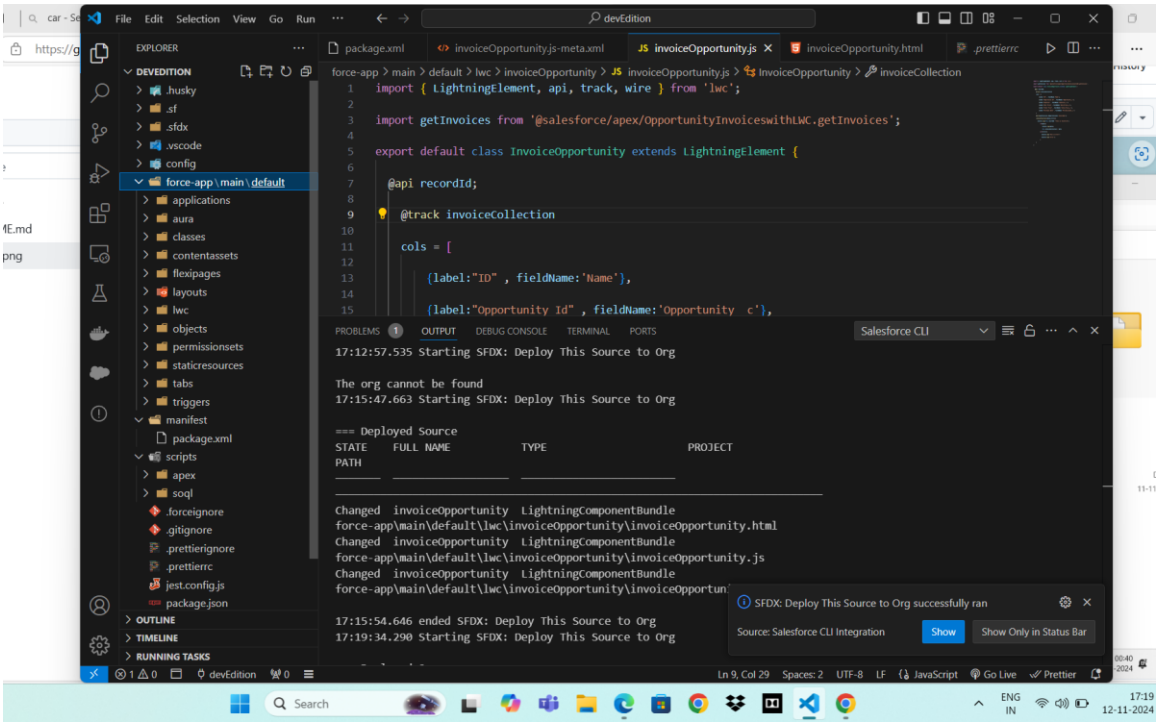
3. **HTML File Code for LWC**

Copy the following code into InvoiceOpportunity.html:

```
html
Copy code
<template>
    <lightning-card title="Opportunity Invoices">
        <lightning-datatable
            key-field="Id"
            data={invoiceCollection}
            columns={cols}>
        </lightning-datatable>
    </lightning-card>
</template>
```

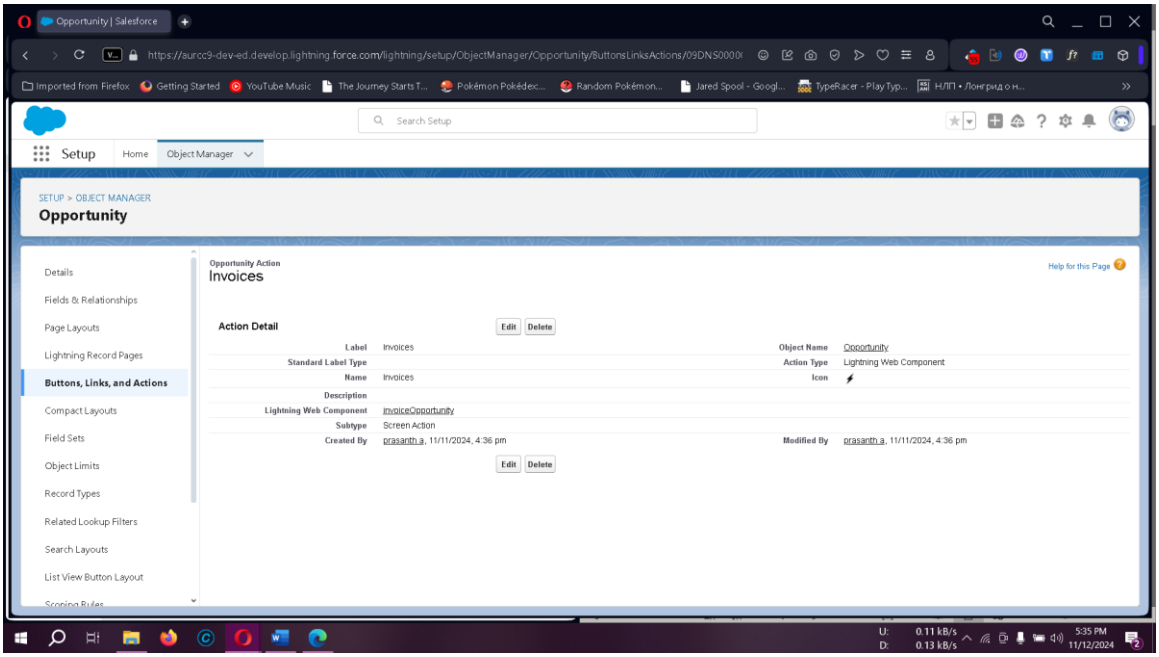
4. **Deploy Component**

Right-click on the component folder and select **SFDX: Deploy Source to Org**.



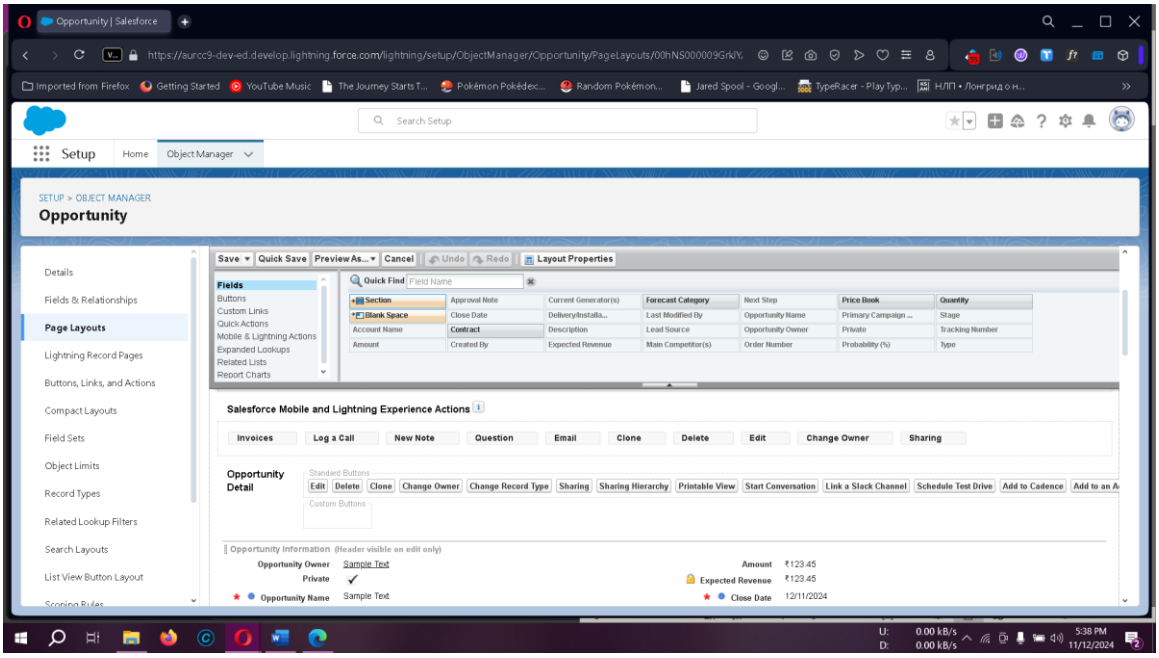
Adding Component to Opportunity Record Page

1. **Add Action Button**
   In Salesforce Setup, go to **Object Manager > Opportunity > Buttons, Links, and Actions**. Click **New Action**, select **Lightning Web Component**, and choose InvoiceOpportunity.



Adding Invoice Opportunity to Opportunity Record Page

1. In **Opportunity Object Manager**, go to **Page Layouts** and select **Opportunity Layout**.
2. Under **Mobile and Lightning Actions**, search for **Invoice** in **Quick Find**.
3. Drag and drop **Invoice** into the Salesforce Mobile and Lightning Experience actions section.
4. Click **Save**.



Milestone 9 - Scheduling Apex for Closed Opportunities

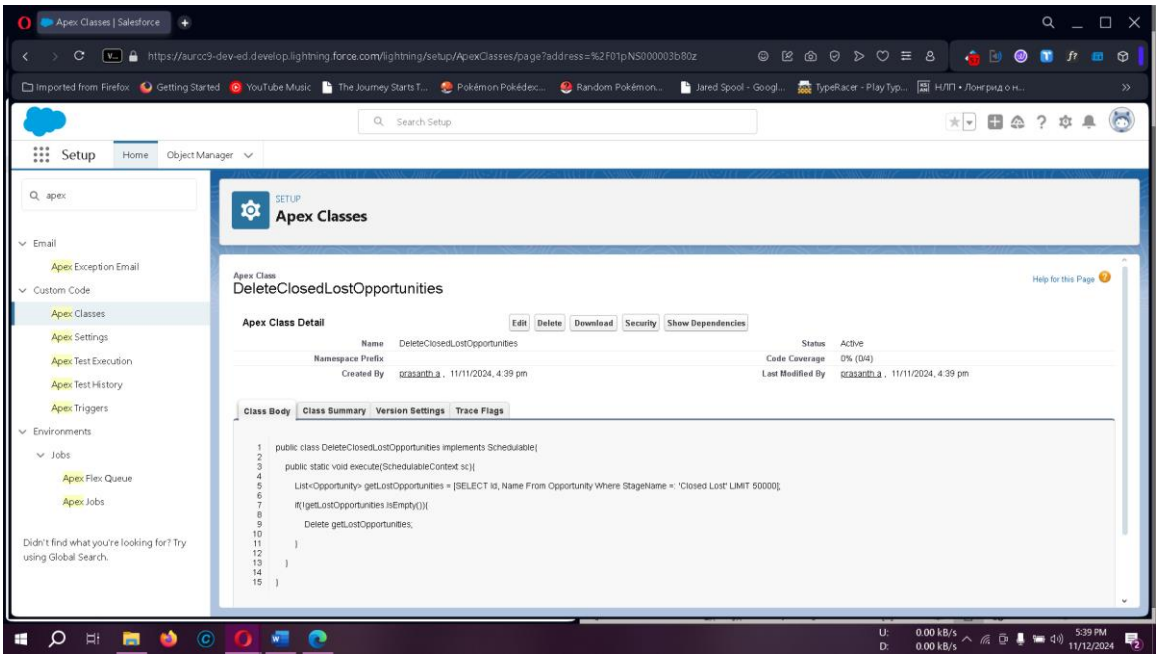
1. **Create Apex Class for Scheduler**
   In Developer Console, create a new Apex class called DeleteClosedLostOpportunities.

   ## Apex Code:

   apex
   Copy code

```
public class DeleteClosedLostOpportunities implements Schedulable {
    public void execute(SchedulableContext sc) {
        List<Opportunity> lostOpps = [
            SELECT Id FROM Opportunity WHERE StageName = 'Closed Lost'
        ];
        delete lostOpps;
    }
}
```

2. **Schedule the Class**
   Go to **Setup > Apex Classes > Schedule Apex** and set it to run every Monday at 10:00 AM.



Milestone 10 - Creating Salesforce 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

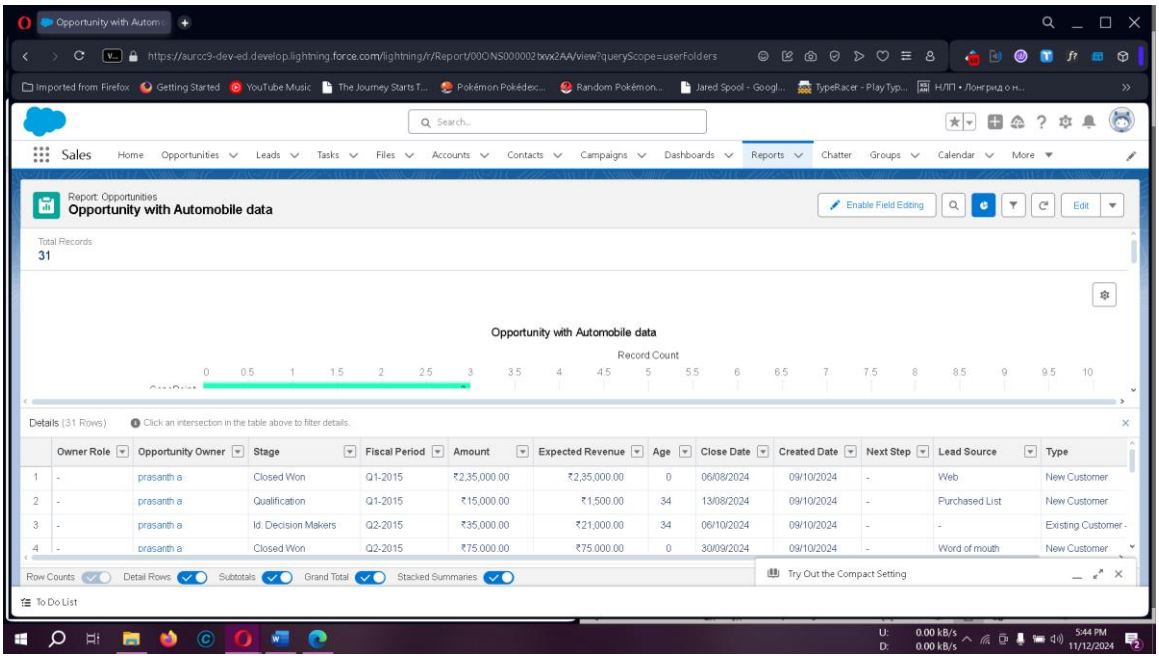
Steps to Create a Report on Opportunity:

1. Navigate to the **Reports** tab and click **New Report**.
2. Select the report type and click **Start Report**.
3. Customize the report by adding fields and applying filters.
4. Save or run the report.

Create a Report on Automobile Information:

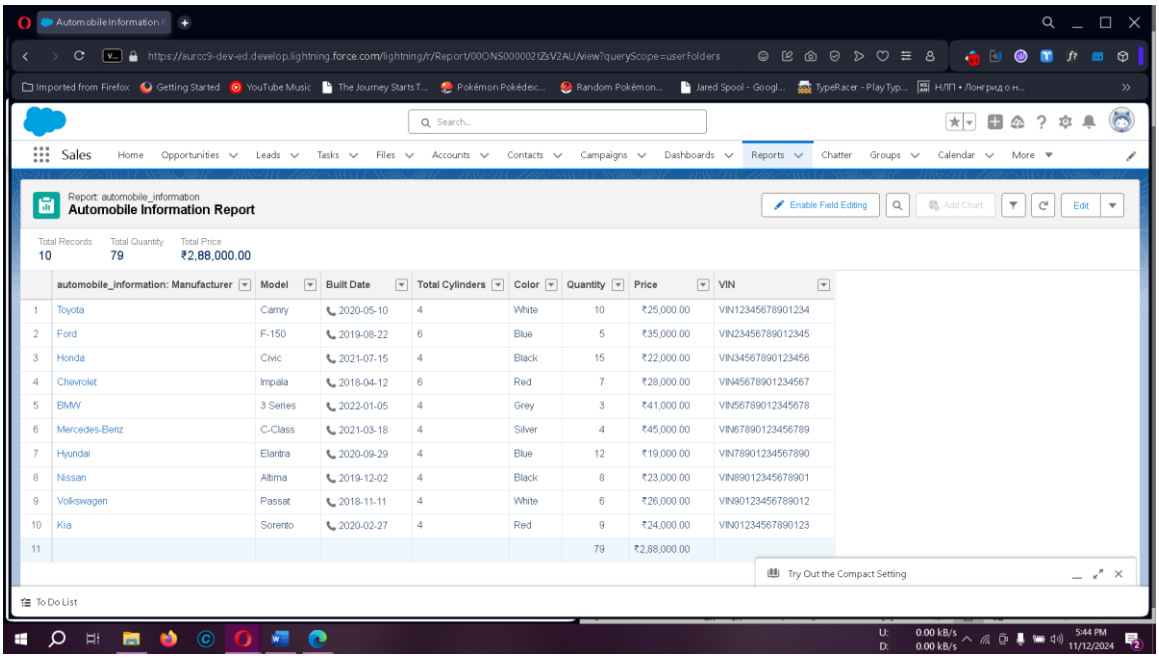
- Choose the **Automobile Information** report type and apply filters.
- **Opportunity Sales Report**

Go to **Reports > New Report**, select **Opportunity** as the report type, and customize the fields and filters.



Create a Report on Opportunity Automobiles:

- Use the **Opportunities with Opportunity Automobiles and Automobile** report type.
- Select the report type: **Automobile Information**.
- Apply filters as needed.



	automobile_information: Manufacturer	Model	Built Date	Total Cylinders	Color	Quantity	Price	VIN
1	Toyota	Camry	2020-05-10	4	White	10	₹25,000.00	VIN12345678901234
2	Ford	F-150	2019-08-22	6	Blue	5	₹35,000.00	VIN23456789012345
3	Honda	Civic	2021-07-15	4	Black	15	₹22,000.00	VIN34567890123456
4	Chevrolet	Impala	2018-04-12	6	Red	7	₹28,000.00	VIN45678901234567
5	BMW	3 Series	2022-01-05	4	Grey	3	₹41,000.00	VIN56789012345678
6	Mercedes-Benz	C-Class	2021-03-18	4	Silver	4	₹45,000.00	VIN67890123456789
7	Hyundai	Elantra	2020-09-29	4	Blue	12	₹19,000.00	VIN78901234567890
8	Nissan	Altima	2019-12-02	4	Black	8	₹23,000.00	VIN89012345678901
9	Volkswagen	Passat	2018-11-11	4	White	6	₹26,000.00	VIN90123456789012
10	Kia	Sorento	2020-02-27	4	Red	9	₹24,000.00	VIN01234567890123
11						79	₹2,88,000.00	

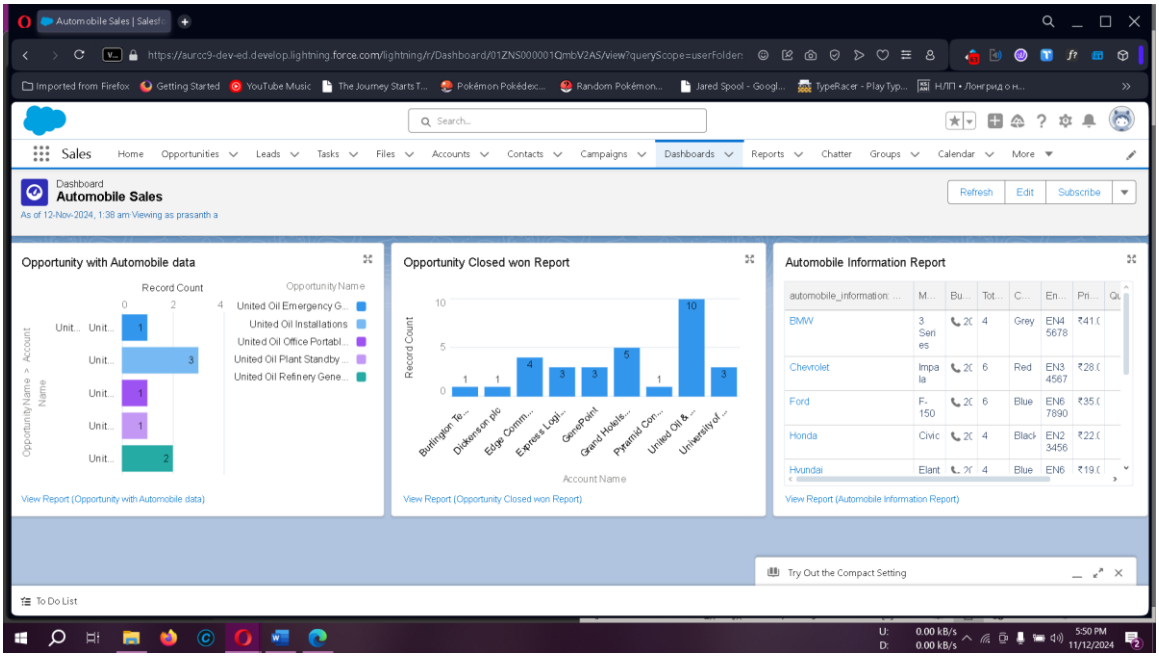
Milestone 11- Creating a Dashboard in Salesforce

Dashboards provide a visual summary of real-time data, enabling users to quickly understand business trends, monitor performance, and make informed decisions. They allow easy access to report data through visual components.

Steps to Create a Sales Dashboard:

1. **Automobile Sales Dashboard**
   Go to **Dashboards > New Dashboard**, name it "Automobile Sales," add components, and save it.

   The created dashboard will display the selected report's data in a visual format.



Key Scenarios Addressed by Salesforce in the Implementation Project

1. **Inventory Management**: Automatically updates automobile stock levels based on sales, reducing manual work and error risks.
2. **Sales Automation**: Generates invoices when an opportunity is closed, providing immediate billing information and tracking payment statuses.
3. **Customer Interaction Tracking**: Logs customer interactions and automates follow-up activities, ensuring timely responses and better customer service.
4. **Performance Insights**: Enables management to view sales and inventory trends through reports and dashboards, supporting data-driven decision-making.

Conclusion

Summary of Achievements

This Salesforce CRM project successfully enhanced the management of automobile sales by implementing a solution that integrates automated processes, custom user interfaces, and real-time reporting. Key achievements include:

- Developed a user-friendly CRM tailored to the automobile sales industry.
- Automated invoicing and inventory tracking, significantly reducing manual input.
- Provided actionable insights into sales performance and inventory trends, supporting better management decisions.

This solution is scalable and adaptable, providing a solid foundation for future enhancements, such as advanced analytics or integration with third-party applications.