

A CRM Application To Handle The Clients And Their Property Related Requirements

Project Description: Dreams World Properties integrates Salesforce to streamline customer interactions. Website engagement triggers automated record creation in Salesforce, capturing customer details and preferences. Salesforce categorizes users as approved or nonapproved, offering tailored property selections to approved users. This enhances user experience and efficiency, providing personalized recommendations and broader listings. Seamless integration optimizes operations, improving customer engagement and facilitating growth in the real estate market.

1. Client Management

- Add, update, and delete client details.
- Track client preferences, budget, and location interests.
- Maintain contact details and communication history.

2. Property Management

- Manage property listings with details like type, price, location, and features.
- Track properties available for sale, rent, or lease.
- Upload photos and documents for properties.

3. Requirement Matching

- Match client requirements with available properties using filters.
- Notify clients about new properties that fit their criteria.

4. Lead Tracking

- Manage inquiries and follow up with potential clients.
- Schedule meetings and site visits.
- Assign leads to specific team members.

Milestone 1:- Create a Jotform and integrate it with the org to create a record of customers automatically.

Client wants a form for the customers to get the details directly into the salesforce so that the admins can create a user in the org. Client wants a form for the customers to get the details directly into the salesforce so that the admins can create a user in the org.

Activity: 1

1. Open your browser and search for jotform and log in.
2. After login click on create form and click on start from scratch
3. Now create a form to get the customer details like Name, Phone, Email, Address and type of property the customer is interested in.
4. Once the form is created, publish it by clicking on publish,
5. form link: <https://form.jotform.com/243219167906057>

Create Objects from Spreadsheet

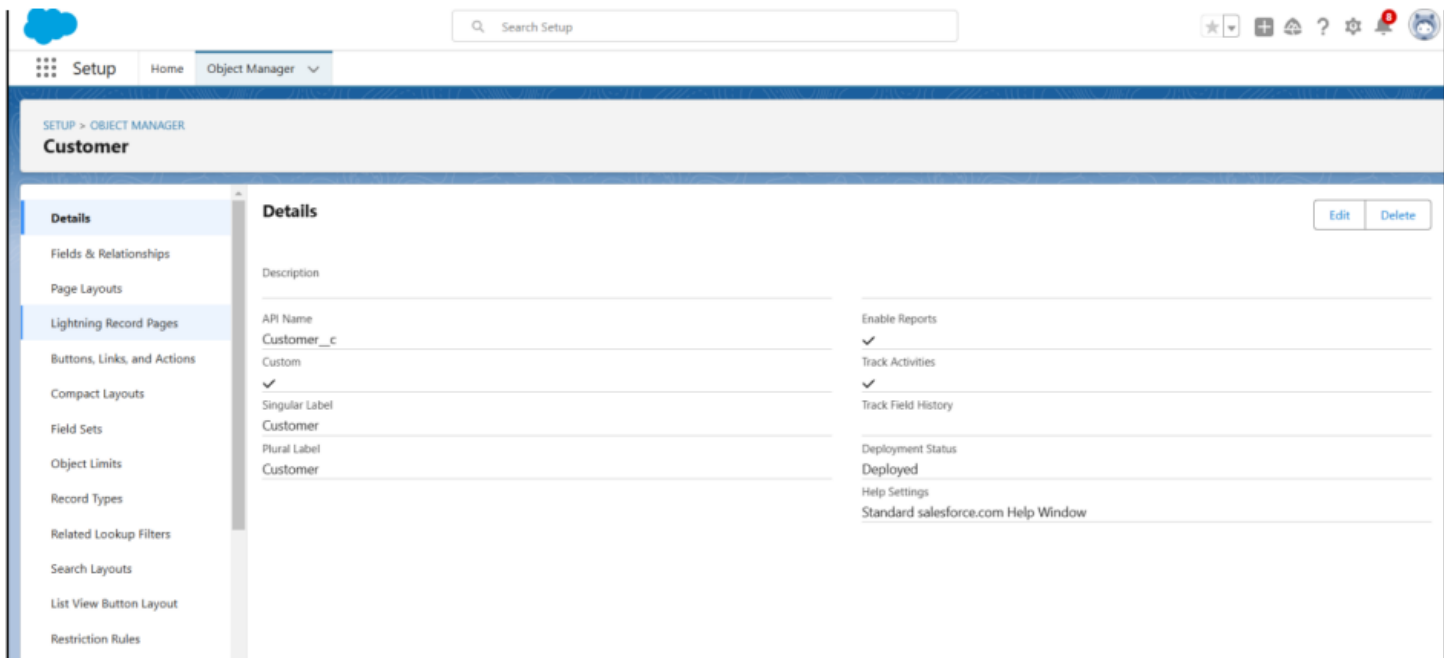
Directly Creating Objects from Spreadsheet in Salesforce

Creating Customer Object:

1. Go to your object manager and and click on create object from spreadsheet
2. Click on the link to get the spreadsheet
3. customer

Customer	Phone Number	Emial	State	Property Type	Budget Amount	Street Address	Street Address	City	postal code	Verified
Rakesh	788797	rakesh@gmail.	Telangana	Residential	4000000	gb road	street no 45	Hyderabad	555001	checked
prakash	55448855	p@gmail.com	Maharashtra	Commercial	8000000	gachibowli	indira road	mumbai	6600014	unchecked
Prajwal	454545	prajwal@gmail.	Maharashtra	Rental	25000	kamdli	kathora	Amravati	444805	checked

4. After downloading, upload the file, map the fields and upload to create an object.

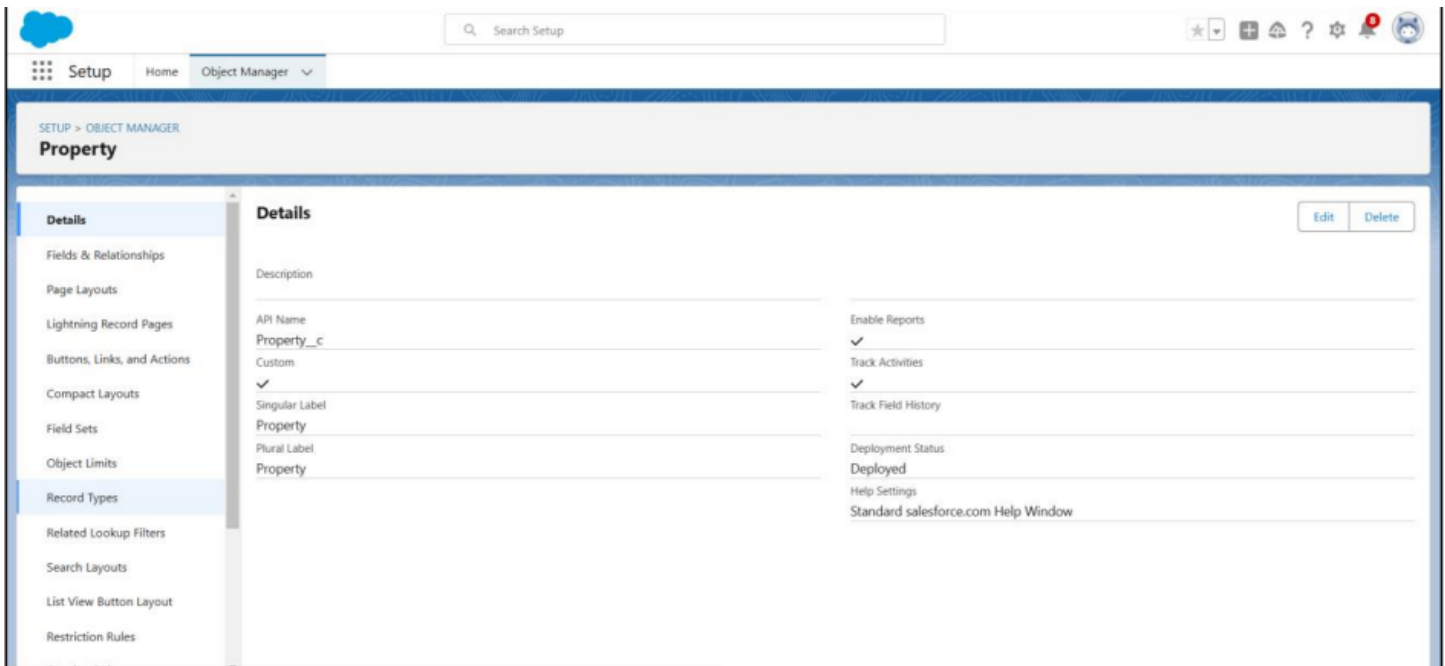


Creating Property Object:

1. Follow the same from the customer object to create the Property Object
2. Property

A	B	C	D
Property Name	Type	Location	Verified
Lotus Appartme	Residential	hydeerabad	checked
500000 sq.ft pk	Commercial	Amravati	unchecked
3 Bhk fkat at st	rental	Jubilee hill Hyd	Checked

3. After downloading, upload the file, map the fie.
4. the fileds as follows.

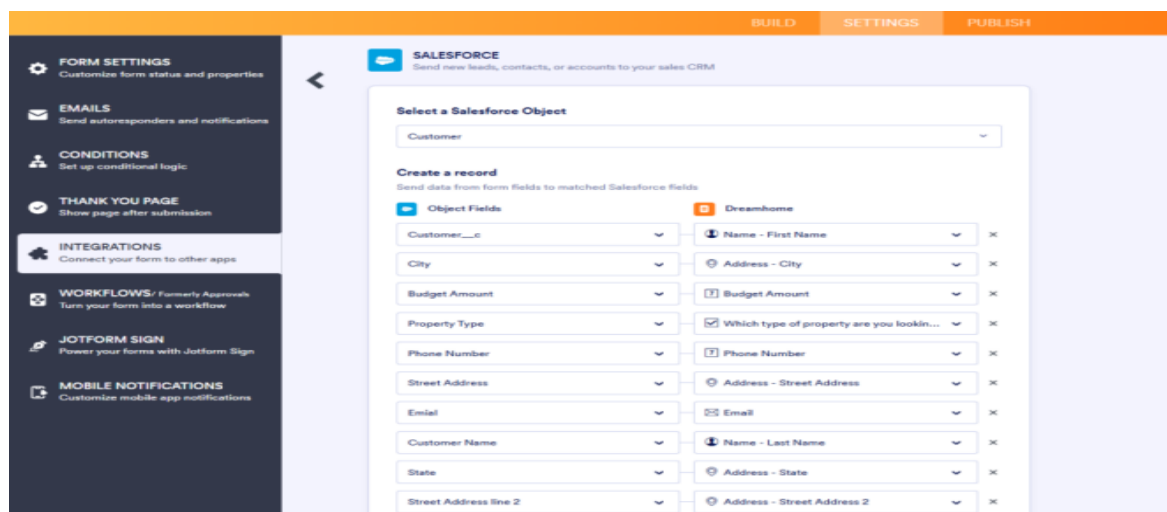


Integrate Jotform with Salesforce Platform

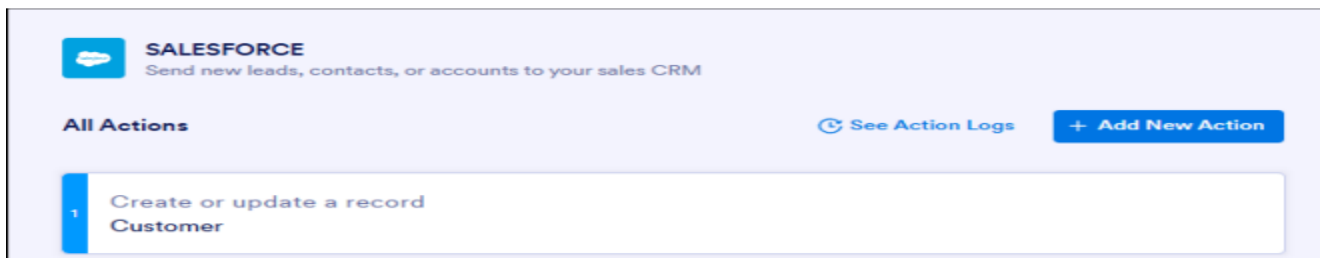
In this Milestone we are going to integrate jotform with Salesforce

Activity :-1

1. On the Jotform Platform, Click on Integration and choose Salesforce
2. Click on User Integration and choose "Add to From"
3. Select the Org with which you want to Integrate your jotform with and select your account
4. Select an Action - Create a record.
5. Select a Salesforce Object: - Customer Map Each and every field on the Object with the fields on the form and "Save Action".



Then "Save the Integration" and "Finish".

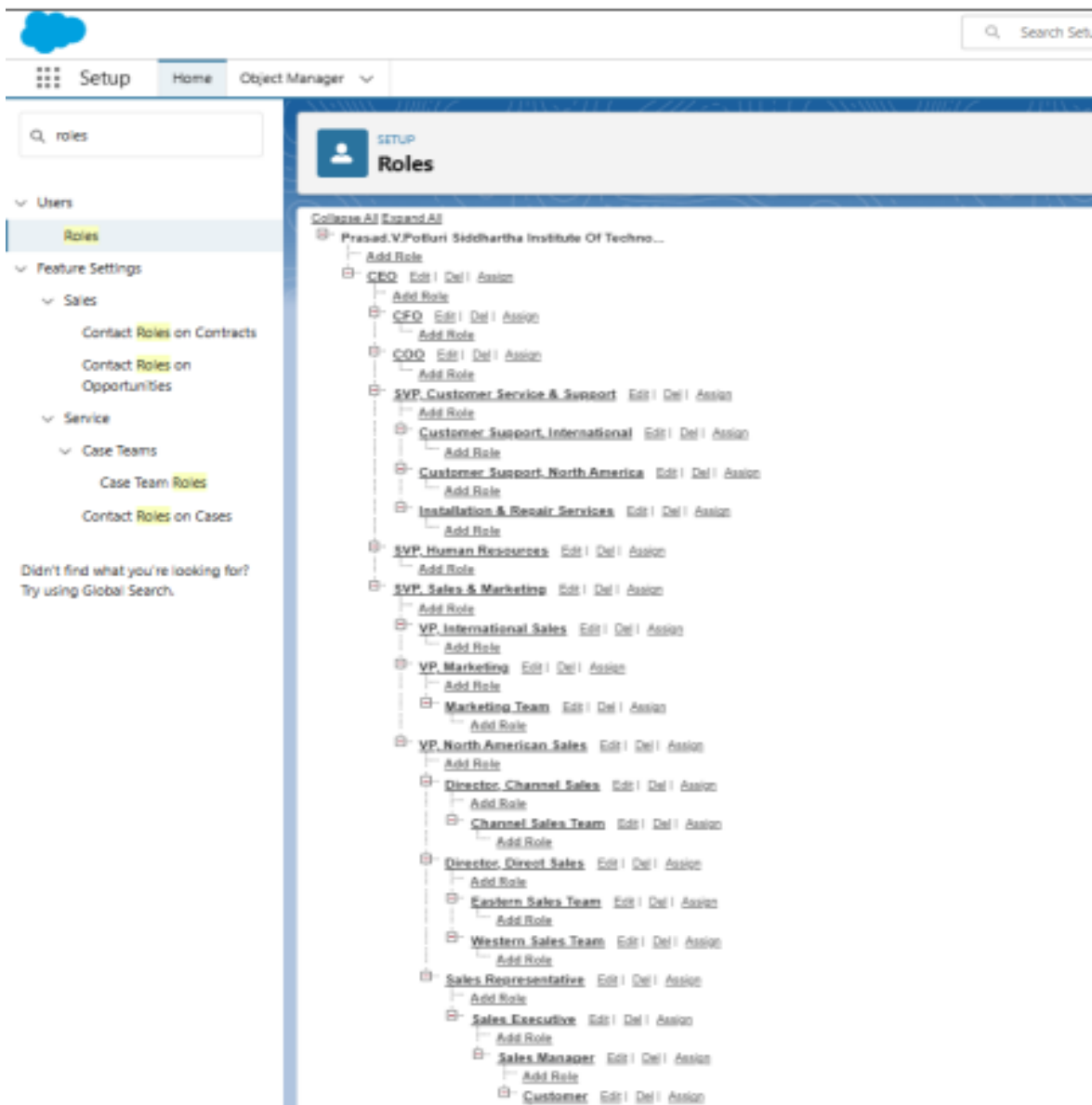


Create Roles

Here we need to Create Roles as per business requirement

Activity :- 1

Go to Setup and Click on Roles, then click on Expand all and Add a Role just below the Sales Representative



1. If we don't find sales representative we need to create it according to the need.
2. It will use the "System Administrator Profile".
3. Label Sales Executive.
4. Reports to - Sales Representative.

The screenshot shows a 'Role Edit' form for a role named 'Sales Executive'. The form has the following fields:

- Label:** A text box containing 'Sales Executive'.
- Role Name:** A text box containing 'Sales_Executive' with an information icon to its right.
- This role reports to:** A dropdown menu showing 'Sales Representative' with a search icon to its right.
- Role Name as displayed on reports:** An empty text box.

At the bottom right of the form are three buttons: 'Save', 'Save & New', and 'Cancel'.

- Similarly Create a Role Name "Sales Manager" below Sales Executive which reports to Sales Executive, Also Add a Role below Sales Manager labeled as "Customer" which reports to Sales Manager.

Create a Property Details App

An App where the objects will be displayed

Activity :- 1

1. From Setup>> Go to App Manager and click on New Lightning App and Name it as "Property Details" and add "Customer" and "Property" Object.
2. Click Next >> Next >> Save and Add "System Admin" Profile.

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name ⓘ

Property Details

* Developer Name ⓘ

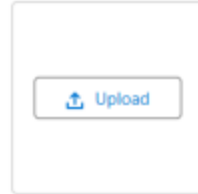
Property_Details

Description ⓘ

Enter a description...

App Branding

Image ⓘ



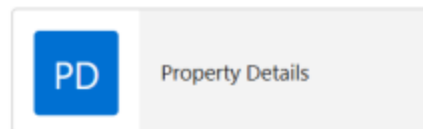
Primary Color Hex Value ⓘ

 #0070D2

Org Theme Options

☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

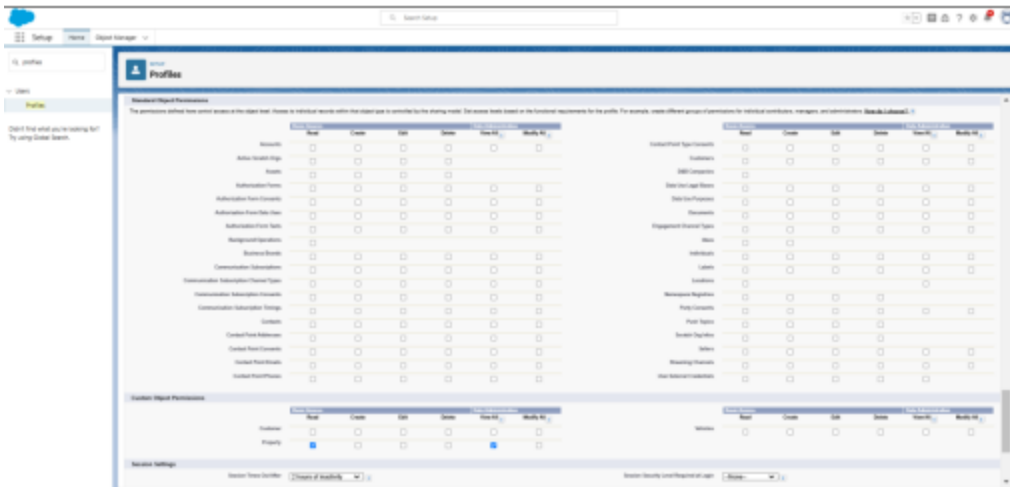


Create Profiles

Create profiles as per business requirement

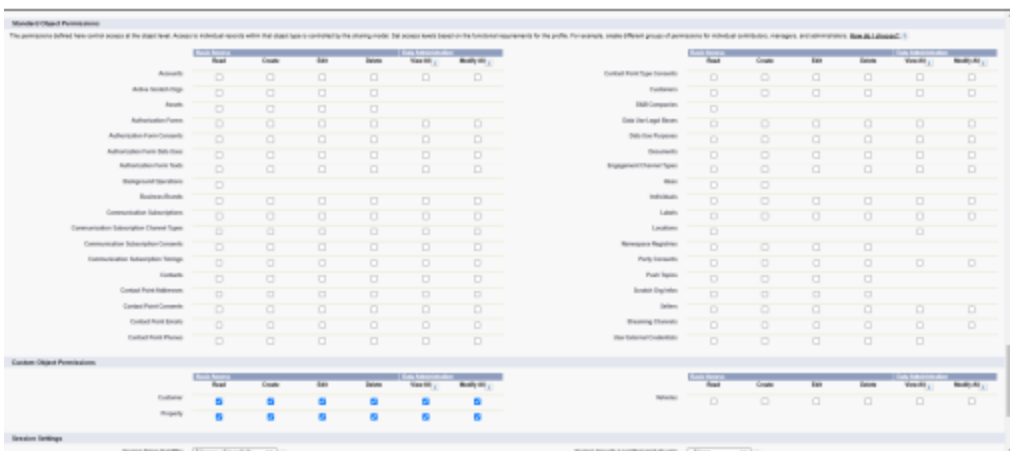
Creating Customer Profile :-

1. From Setup? Go to Profiles and Clone (standard platform) Salesforce Platform User and Name it "Customer".
2. Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
3. Also Remove all the Standard Object Permissions
4. Uncheck all the Custom Object Permissions and check read and view all in "Property"
5. Make sure every submission object permissions are unselected and then save.



Creating Manager Profile :-

1. From Setup >> Go to Profiles and Clone Salesforce Platform User and Name it "Manager".
2. Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
3. Also Remove all the Standard Object Permissions.
4. Uncheck all the Custom Object Permissions and check only "modify all" from "Property" and "Customer".



Create a Check Box field on user

Create Field on the User as per the business requirement,

Activity :- 1

1. Setup >> Object Manager >> Search for User >> Fields and Relationships
2. Select the Data type "Check Box"
3. Create new Field Named as "Verified"

The screenshot displays the Salesforce Object Manager interface for the 'User' object. The left sidebar shows the navigation menu with 'Fields & Relationships' selected. The main content area is titled 'User Custom Field: Verified' and includes a 'Back to User Fields' link. Below the title, there are tabs for 'Field Information', 'Set Field-Level Security', 'View Field Accessibility', and 'Where is this used?'. The 'Field Information' tab is active, showing a table with the following details:

Custom Field Definition Detail	
Field Information	
Field Label	Verified
Field Name	Verified
API Name	Verified__c
Description	
Help Text	
Date Format	
Field Format	
Field Security	
Field Visibility	
Created By	ALBERTO DEVO, 18/11/2024, 1:48 pm
Modified By	ALBERTO DEVO, 18/11/2024, 1:48 pm

Below the table, there is a 'Generate Options' section with a 'Default Value' of 'Undefined'. At the bottom, there is a 'Validation Rules' section with a 'New' button and a message 'No validation rules defined'.

Create Users

Create three different users with three different Roles and profiles as we have mentioned above.

Here we are going to create 4 users.

User: 1

1. Go to Setup -> Administration -> Users -> New User
2. Last Name - Executive

3. Role - Sales Executive
4. License - Salesforce
5. Profile - System Administrator
6. Save

The screenshot shows the 'User Edit' page for a user named 'Executive'. The 'General Information' section includes fields for First Name, Last Name, Alias, Email, Username, Password, Title, Company, Department, and Division. The 'Role' dropdown is set to 'Sales Executive', 'User License' is 'Salesforce', and 'Profile' is 'System Administrator'. The 'Active' checkbox is checked. Other roles like Marketing User, Offline User, Knowledge User, Flow User, Service Cloud User, Site.com Contributor User, Site.com Publisher User, and WDC User are listed with unchecked checkboxes.

User: 2

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Manager
3. Role >> Sales Manager
4. License >> Salesforce Platform
5. Profile >> Manager
6. Save

The screenshot shows the 'User Edit' page for a user named 'Manager'. The 'General Information' section includes fields for First Name, Last Name, Alias, Email, Username, Password, Title, Company, Department, and Division. The 'Role' dropdown is set to 'Sales Manager', 'User License' is 'Salesforce Platform', and 'Profile' is 'Manager'. The 'Active' checkbox is checked. Other roles like Marketing User, Offline User, Knowledge User, Flow User, Service Cloud User, Site.com Contributor User, Site.com Publisher User, and WDC User are listed with unchecked checkboxes. The 'Data.com User Type' dropdown is set to 'None'.

User: 3

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Customer
3. Role >> Customer
4. License >> Salesforce Platform
5. Profile >> Customer
6. Make Sure the verified check box is "Unchecked"
7. Save

User Edit
Customer

Save Save & New Cancel

General Information

First Name:

Last Name:

Alias:

Email:

Username:

Nickname:

Title:

Company:

Department:

Division:

Role:

User License:

Profile:

Active: ☒

Marketing User: ☐

Offline User: ☐

Knowledge User: ☐

Flow User: ☐

Service Cloud User: ☐

Site.com Contributor User: ☐

Site.com Publisher User: ☐

WDC User: ☐

Data.com User Type:

Data.com Weekly Addition Limit:

Accessibility Mode (Classic Only): ☐

High-Contrast Palette on Charts: ☐

User: 4

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Customer2
3. Role >> Customer
4. License >> Salesforce Platform
5. Profile >> Customer
6. Make Sure the verified check box is "checked"
7. Save

User Edit
Customer2

Save Save & New Cancel

General Information

First Name:

Last Name:

Alias:

Email:

Username:

Nickname:

Title:

Company:

Department:

Division:

Role:

User License:

Profile:

Active: ☒

Marketing User: ☐

Offline User: ☐

Knowledge User: ☐

Flow User: ☐

Service Cloud User: ☐

Site.com Contributor User: ☐

Site.com Publisher User: ☐

WDC User: ☐

Data.com User Type:

Data.com Weekly Addition Limit:

Accessibility Mode (Classic Only): ☐

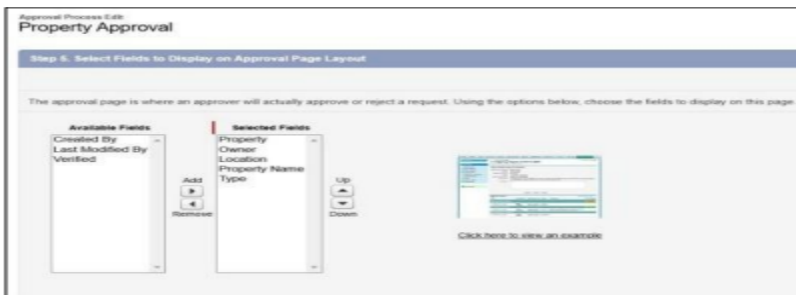
High-Contrast Palette on Charts: ☐

Create an Approval Process for Property Object

An Approval process to approve or reject the records as according

Activity :- 1

1. From Setup >> Process Automation >> Approval Process
2. before proceeding we need to select property in the manage approval process
3. Process Name - Property Approval
4. Select 2 criteria - a. Location - not equal to - blank, b. Verified-Equals- false
5. Click next and "Next Automated Approver Determined By" Select Manager
6. From Record Editability Properties >> Click on Administrators OR the currently assigned approver can edit records during the approval process.
7. From Step 5: Select Fields to Display on Approval Page Layout select Property, Owner, Location, Type.



Click Next and Select the initial Submitters >>

- Owner >> Property Owner
- Roles >> Sales Manager

Save

After saving we are directed to approval steps and we need to do as follows Add an approval step name "Executive Approval "

Click next and select the Approver as" Sales Executive" and "Save"

Add One field Update as "Verified Property"

- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as "True"
- Save. Add One field Update as "UnVerified Property"
- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as "False"

- Save.

Activate the Approval Process.

The screenshot shows the 'Approval Processes' setup page in Salesforce. The process name is 'Property Approval'. It includes fields for 'Unique Name', 'Description', 'Entry Criteria', 'Record Eligibility', 'Approval Assignment Email Template', 'Initial Submitters', and 'Created By'. The 'Active' checkbox is checked. Below the details, there are sections for 'Initial Submission Actions' and 'Approval Steps'. The 'Approval Steps' table shows a single step named 'Executive Approval' with the criteria 'User Executive' and the action 'First Rejection'.

Action	Type	Description
Record Lock	Record Lock	Lock the record from being edited

Action	Step Number	Name	Description	Criteria	Assigned Approver	Repeat Behavior
Executive Approval	1	Executive Approval		User Executive		First Rejection

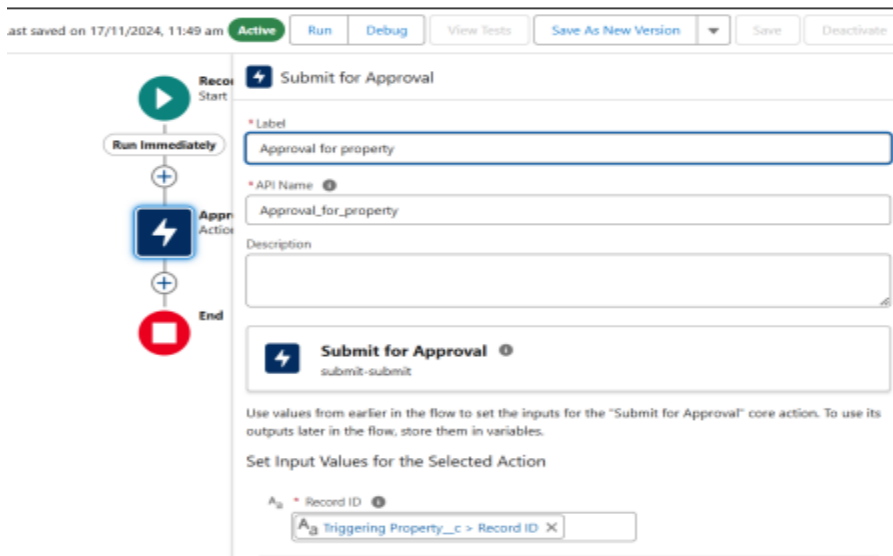
Create a Record trigger flow to submit the Approval Process Automatically

A flow that can submit the records directly for approval

Activity :- 1

1. From Setup >> Search for Flows >> Click On New and Select "Record Trigger Flow".
2. Select Object >> Property
3. Select "Trigger the flow when" >> "A record is created"
4. Set Entry Conditions >> "None"
5. Add a "Action" >> "Submit for Approval"
6. Give Label >> Approval for property
7. Record Id >> {!\$Record.Id}
8. Done

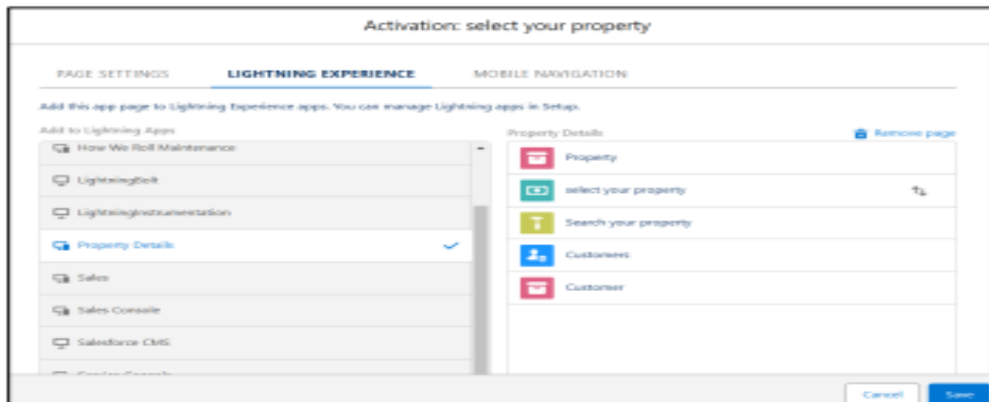
Save the Flow and Give label as "Property Approval" and "Activate"



Create an App Page

Create an App Page on the Property details Object named as "Search Your Property"
Activity :- 1

1. From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and
2. Click on Next
3. Give Label as "Search your Property" click "Next".
4. Click "header and Left Sidebar" and Click on "Done"
5. Click on "Save" and then click on "Activate".
6. From Page Setting select page activation as "Activate for all Users".
7. From Lightning Experience Click on "Property Details" and click on Add Page".
8. Then Click on "Save"



Create a LWC Component

Create an LWC Component for the customers so that only verified customers can access the verified properties and non Verified customers can access non verified properties, and deploy it on "Search your Property Page"

Activity :- 1

1. Create an Apex Class and make it aura enabled and name it "PropertHandler_LWC".

Code:

```
public class PropertyHandler_LWC {
    @AuraEnabled(cacheable=true)
    public static List getProperty(String type, String verified) {
        // Ensure the verified parameter matches the field type in Salesforce
        return [ SELECT Id, Location__c, Property_Name__c, Type__c, Verified__c FROM Property__c WHERE
        Type__c =: type AND Verified__c =: verified ];
    }
}
```

```
PropertyHandler_LWC.apxc
Code Coverage: None API Version: 62
1 public class PropertyHandler_LWC {
2
3     @AuraEnabled(cacheable=true)
4     public static List<Property__c> getProperty(String type, String verified) {
5         // Ensure the verified parameter matches the field type in Salesforce
6         return [
7             SELECT Id, Location__c, Property_Name__c, Type__c, Verified__c
8             FROM Property__c
9             WHERE Type__c =: type AND Verified__c =: verified
10        ];
11    }
12 }
```

2. Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.

3. Enter your login ic and password to authorize your org.

4. Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to.

5. In your Html File, write this code

Code:

```
<template>
```

```
<lightning-card>
```

```
<div class="slds-box">
```

```
<div class="slds-text-align_left">
```

```
<h1 style="font-size: 20px;"><b>Properties</b></h1>
```

```
</div>
```

```
<div>
```

```
<div class="slds-grid slds-gutters">
```

```
<div class="slds-col slds-size_5-of-6">
```

```
<lightning-combobox name="Type" label="Property Type" value={typevar} placeholder="Select
Property type"
```

```
options={propetyoptions} onchange={changehandler}></lightning-combobox>
```

```
</div>
```

```
<div class="slds-col slds-size_1-of-6">
```

```
<br>
```

```
<lightning-button-icon variant="neutral" icon-name="standard:search" alternative-text="Search"
label="Search" onclick={handleClick}></lightning-button-icon>
```

```
</div>
```

```
</div>
```

```
</div>
```


</div>

<template if:true={istru}>

<div class="slds-box">

<lightning-datatable key-field="id" data={propertylist} columns={columns}></lightning-datatable>

</div>

</template>

<template if:false={isfalse}>

<div class="slds-box">

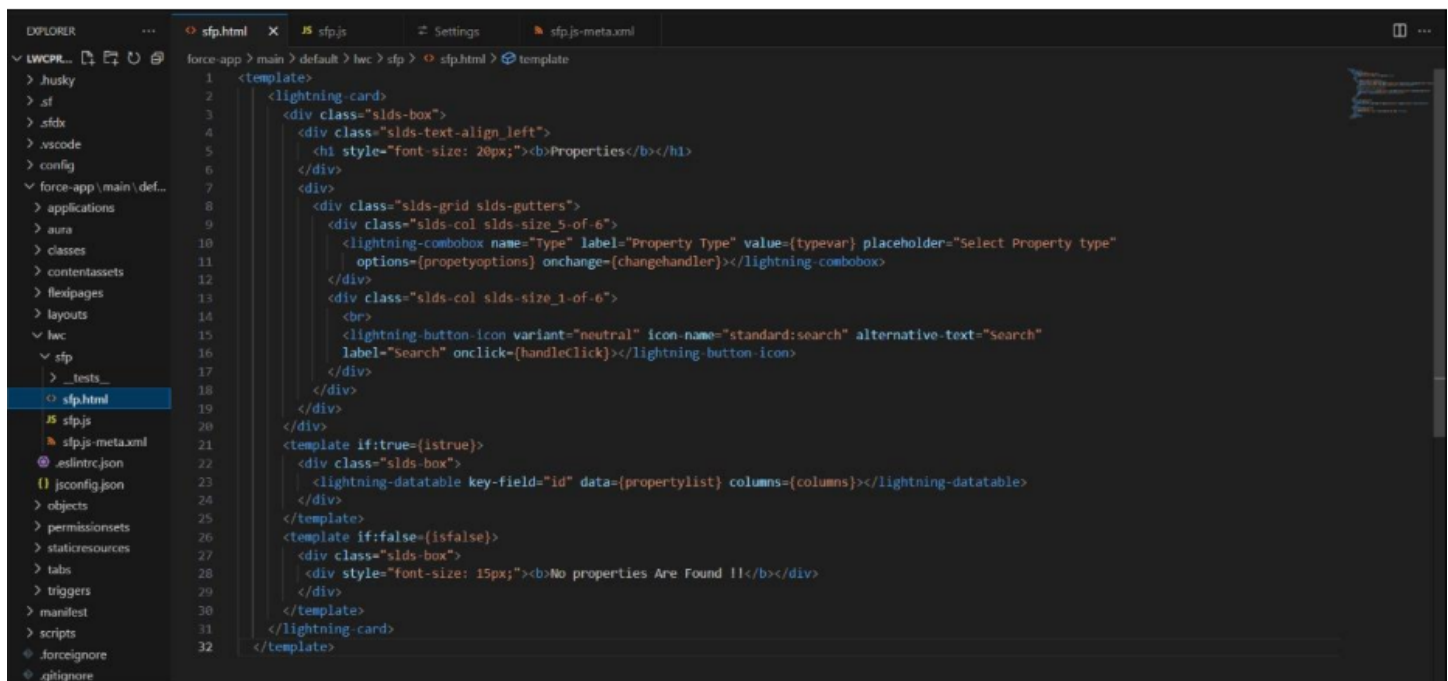
<div style="font-size: 15px;">No properties Are Found !!</div>

</div>

</template>

</lightning-card>

</template>



6. In your Js File, write this code

Code:

```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from '@salesforce/apex/PropertyHandler_LWC.getProperty';
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';
export default class C_01_Property_Management extends LightningElement {
  @api recordId userId = USER_ID;
  verifiedvar typevar isfalse = true;
  istru = false;
  @track property
```

```

list = [];
columns = [ { label: 'Property Name', fieldName: 'Property_Name__c' }, { label: 'Property Type',
fieldName: 'Type__c' }, { label: 'Property Location', fieldName: 'Location__c' }, { label: "Property link",
fieldName: "Property_link__c" } ]
propetyoptions = [ { label: "Commercial", value: "Commercial" }, { label: "Residential", value:
"Residential" }, { label: "rental", value: "rental" } ] @wire(getRecord, { recordId: "$userId", fields:
['User.Verified__c'] })
recordFunction({ data, error }) {
  if (data) {
    console.log(data)
    console.log("This is the User Id --> "+this.userId);
    this.verifiedvar = data.fields.Verified__c.value; }
  else {
    console.error(error)
    console.log('this is error')
  }
}
changehandler(event) {
  console.log(event.target.value);
  this.typevar = event.target.value;
}
handleClick() {
  getProperty({ type: this.typevar, verified: this.verifiedvar })
  .then((result) => {
    this.isfalse = true; console.log(result)
    console.log("This is the User id --> ' + this.userId);
    console.log("This is the verified values --> ' + this.verifiedvar);
    if (result != null && result.length != 0) {
      this.istrue = true; this.propertylist = result;
      console.log(this.verifiedvar);
      console.log(this.typevar)
    } else {
      this.isfalse = false;
      this.istrue = false; } })
  .catch((error) => { console.log(error) }) } }

```

```

1  import { LightningElement, api, track, wire } from 'lwc';
2  import getProperty from "@salesforce/apex/PropertyHandler_LWC.getProperty";
3  import { getRecord } from 'lightning/uiRecordApi';
4  import USER_ID from '@salesforce/user/Id';
5  export default class C_01_Property_Management extends LightningElement {
6
7      @api recordId;
8      userId = USER_ID;
9      verifiedvar;
10     typevar;
11     isfalse = true;
12     istrue = false;
13     @track propertylist = [];
14     columns = [
15         { label: 'Property Name', fieldName: 'Property_Name__c' },
16         { label: 'Property Type', fieldName: 'Type__c' },
17         { label: 'Property Location', fieldName: 'Location__c' },
18         { label: "Property link", fieldName: "Property_link__c" }
19     ]
20     propertyoptions = [
21         { label: "Commercial", value: "Commercial" },
22         { label: "Residential", value: "Residential" },
23         { label: "rental", value: "rental" }
24     ]
25     @wire(getRecord, { recordId: "$recordId", fields: ['User.Verified__c'] })
26     recordFunction({ data, error }) {
27         if (data) {
28             console.log(data);
29             console.log("This is the User Id ---> " + this.userId);
30             this.verifiedvar = data.fields.Verified__c.value;
31         } else {
32             console.error(error);
33             console.log('this is error')
34         }
35     }
36     changehandler(event) {
37         console.log(event.target.value);
38         this.typevar = event.target.value;
39     }
40     handleClick() {
41         getRecord({ type: this.typevar, verified: this.verifiedvar })
42             .then((result) => {
43                 this.isfalse = true;
44                 console.log(result);
45                 console.log('This is the User id ---> ' + this.userId);
46                 console.log('This is the verified values ---> ' + this.verifiedvar);
47                 if (result != null && result.length != 0) {
48                     this.istrue = true;
49                     this.propertylist = result;
50                     console.log(this.verifiedvar);
51                     console.log(this.typevar)
52                 } else {
53                     this.isfalse = false;
54                     this.istrue = false;
55                 }
56             })
57             .catch((error) => {
58                 console.log(error)
59             })
60     }
61 }

```

7. In your Metafile, give your targets to deploy the component.

Code:

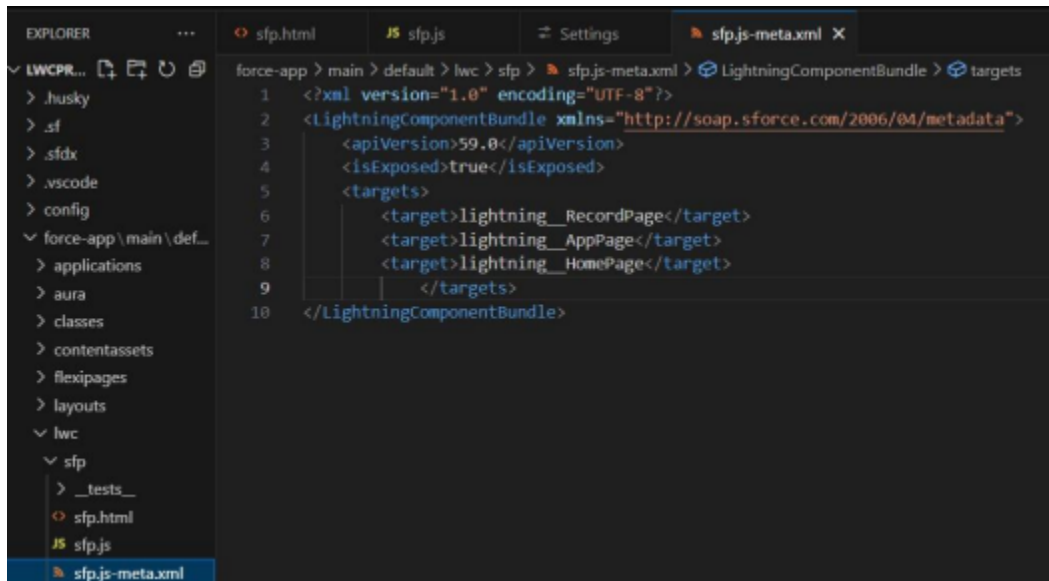
```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
  <apiVersion>59.0</apiVersion>
  <isExposed>true</isExposed>
  <targets>
    <target>lightning__RecordPage</target>
    <target>lightning__AppPage</target>
    <target>lightning__HomePage</target>

```

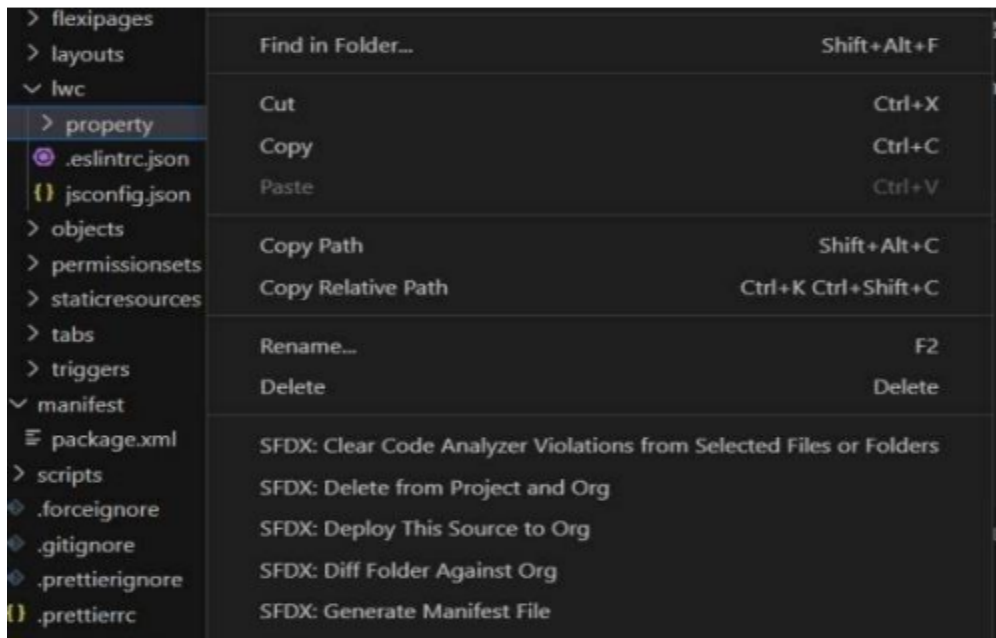
</targets>

</LightningComponentBundle>



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
3   <apiVersion>59.0</apiVersion>
4   <isExposed>true</isExposed>
5   <targets>
6     <target>lightning__RecordPage</target>
7     <target>lightning__AppPage</target>
8     <target>lightning__HomePage</target>
9   </targets>
10 </LightningComponentBundle>
```

8. After saving all the three codes, right click and deploy this component to the org.

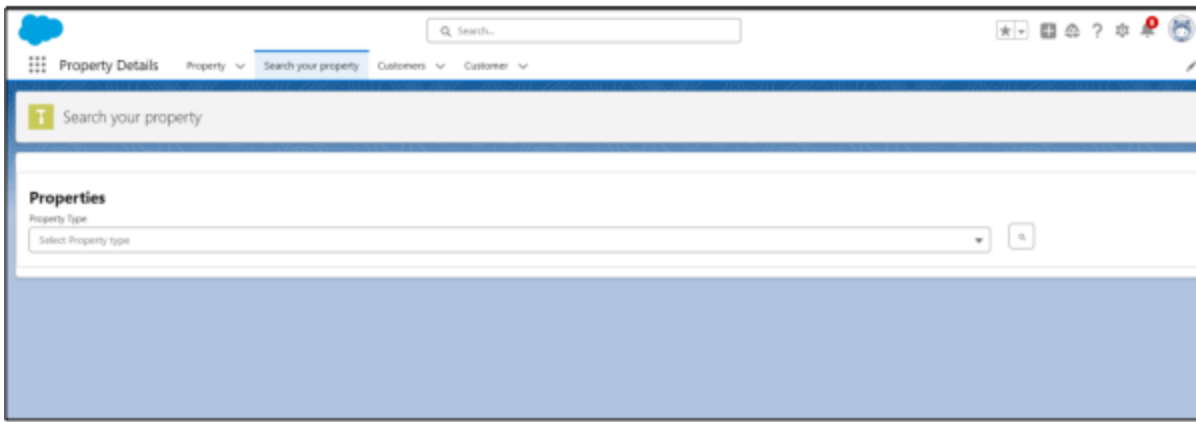


Drag this Component to your App Page

Adding the Component to your Page

Activity :- 1

1. From Setup >> Go to App Launcher >> Search for Property Details
2. On this Page click on gear icon and click on Edit Page
3. After clicking on edit page it will be directed to app pages then
4. Drag the Component(properties) to your App Page and Save the Page.



Give Access of Apex Classes to Profiles

The Apex Class has a Security, Enable the security for the profiles that needs to access this class.

Activity :- 1

1. From Setup >> Search For Apex Classes >> Click on "Security" behind "PropertyHandler_LWC".
2. From Profiles Add "Manager" and "Customer" and "Save".



SETUP

Profiles

Enable Profile Access for Apex Class

PropertyHandler_LWC

Save

Cancel

Available Profiles

Analytics Cloud Integration User
Analytics Cloud Security User
Authenticated Website
B2B Reordering Portal Buyer Profile
Contract Manager
Cross Org Data Proxy User
Custom: Marketing Profile
Custom: Sales Profile
Custom: Support Profile
Customer Community Login User
Customer Community Plus Login User
Customer Community Plus User
Customer Community User
Customer Portal Manager Custom

Add



Remove

Enabled Profiles

Customer
Manager
System Administrator