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 non-approved, 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
    a. Add, update, and delete client details.
    b. Track client preferences, budget, and location interests.
    c. Maintain contact details and communication history.

2. Property Management
    a. Manage property listings with details like type, price, location, and features.
    b. Track properties available for sale, rent, or lease.
    c. Upload photos and documents for properties.

3. Requirement Matching
    a. Match client requirements with available properties using filters.
    b. Notify clients about new properties that fit their criteria.

4. Lead Tracking
    a. Manage inquiries and follow up with potential clients.
    b. Schedule meetings and site visits.
    c. Assign leads to specific team members.

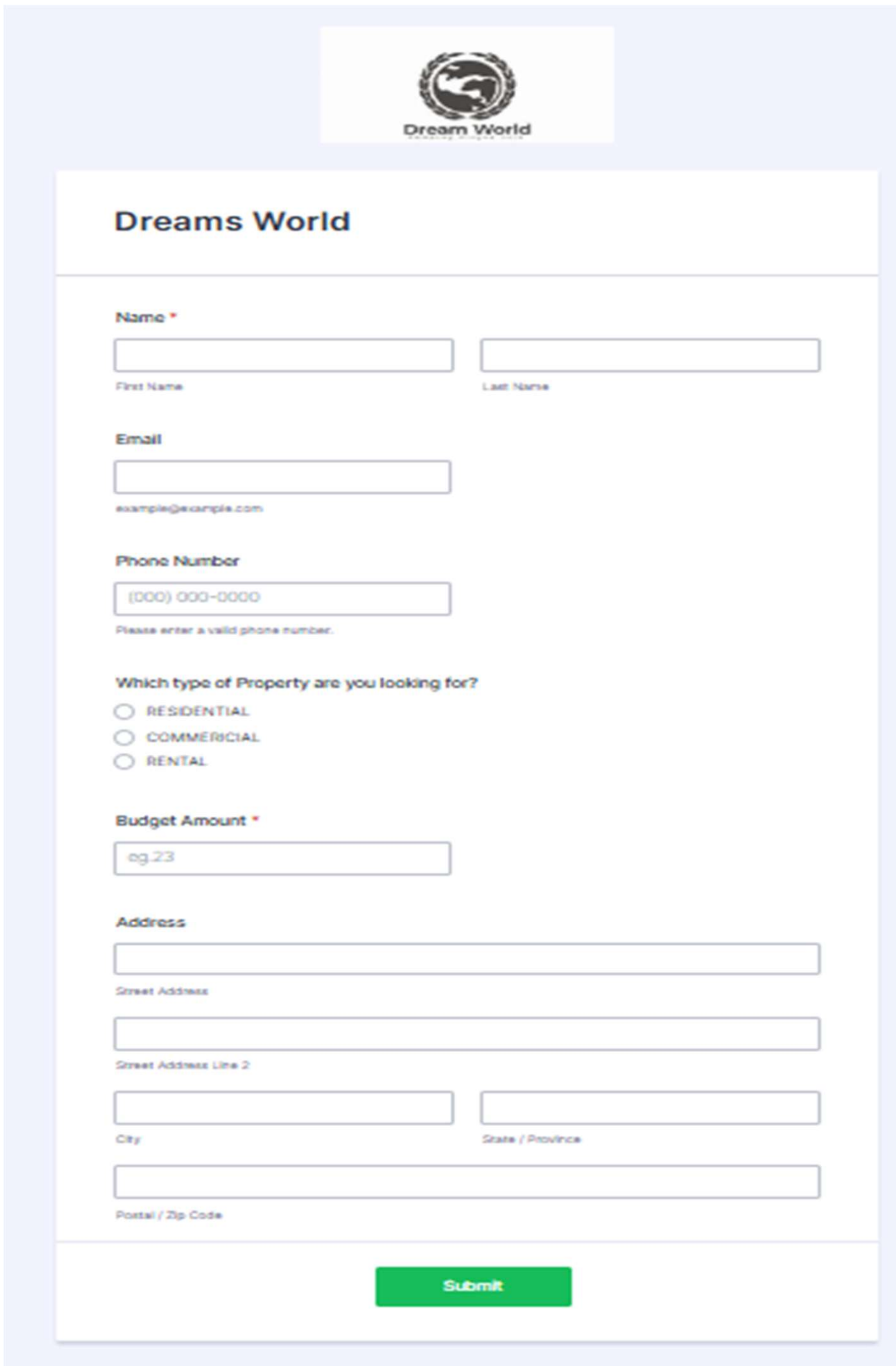
Milestone 1: Create a Jotform and integrate it with the org to create a record of customers automatically.

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.

Activity1

Open your browser and search for jotform and log in.

1. After login click on create form and click on start from scratch
2. Now create a form to get the customer details like Name, Phone,Email, Address and type of property the customer is interested in.
3. Once the form is created, publish it by clicking on publish.
4. form link :-https://form.jotform.com/21501a0534/SweetHome



Dream World

Dreams World

Name *

First Name | Last Name

Email

example@example.com

Phone Number

(000) 000-0000

Please enter a valid phone number.

Which type of Property are you looking for?

- RESIDENTIAL
- COMMERICIAL
- RENTAL

Budget Amount *

eg.23

Address

Street Address

Street Address Line 2

City | State / Province

Postal / Zip Code

Submit

Create Objects from Spreadsheet

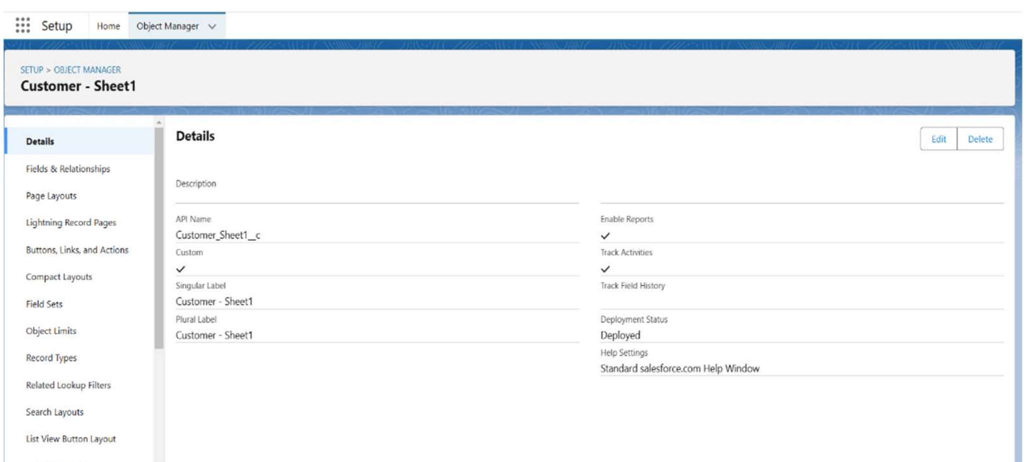
Directly Creating Objects from Spreadsheet in Salesforce

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	kamdi	kathora	Amravati	444805	checked

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

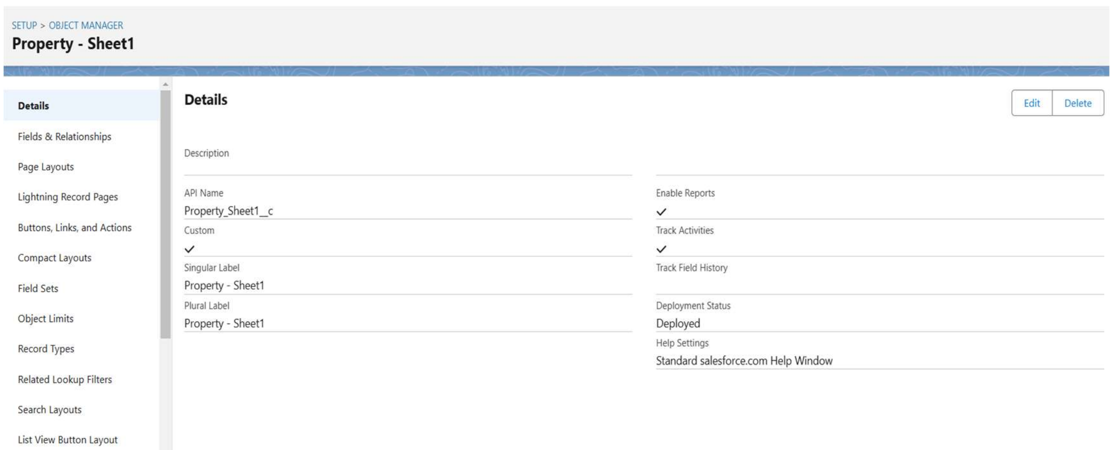


Creating Property Object :

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 pl	Commercial	Amravati	uchecked
3 Bhk fkat at st	rental	Jubliee hill Hyd	Checked

After downloading, upload the file, map the fields and upload to create an object. the fileds as follows



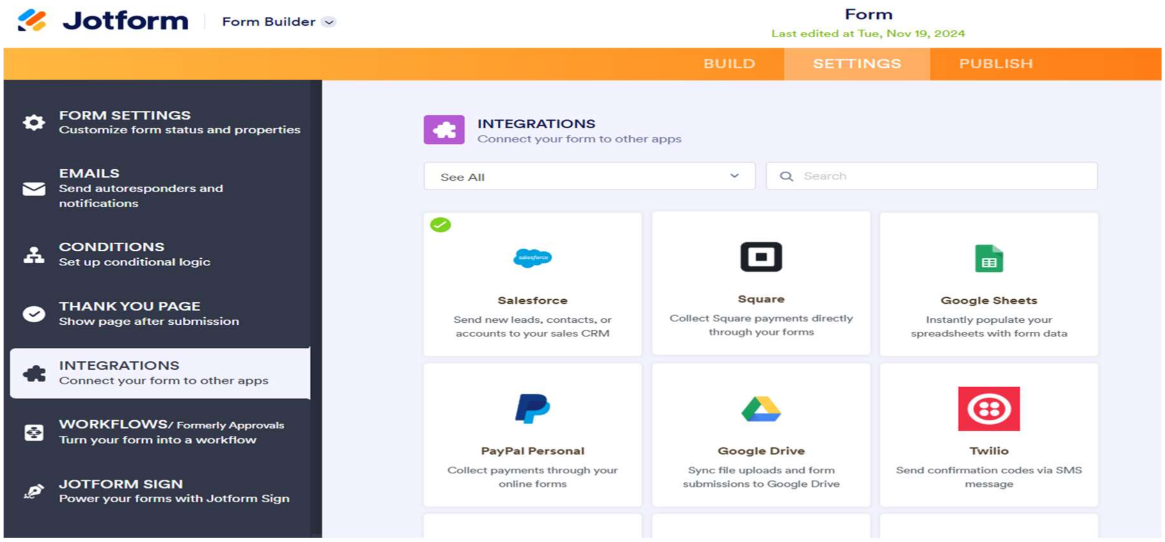
Integrate Jotform with Salesforce Platform

In this Milestone we are going to integrate jotform with Salesforce

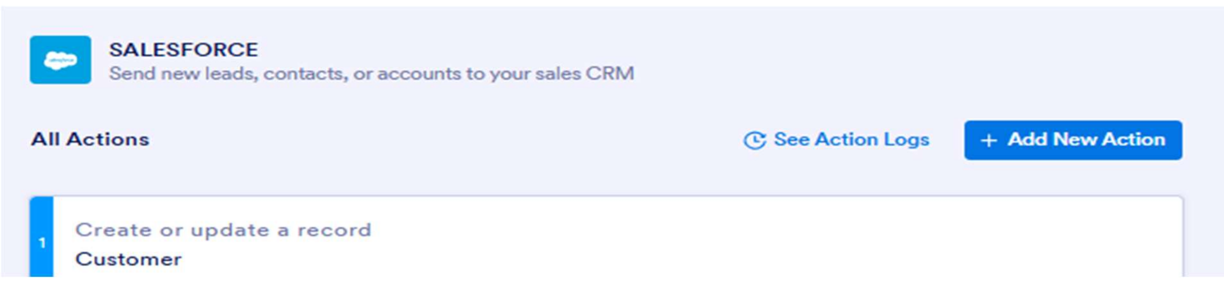
Activity

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 requirenment

Activity:- 1

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

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

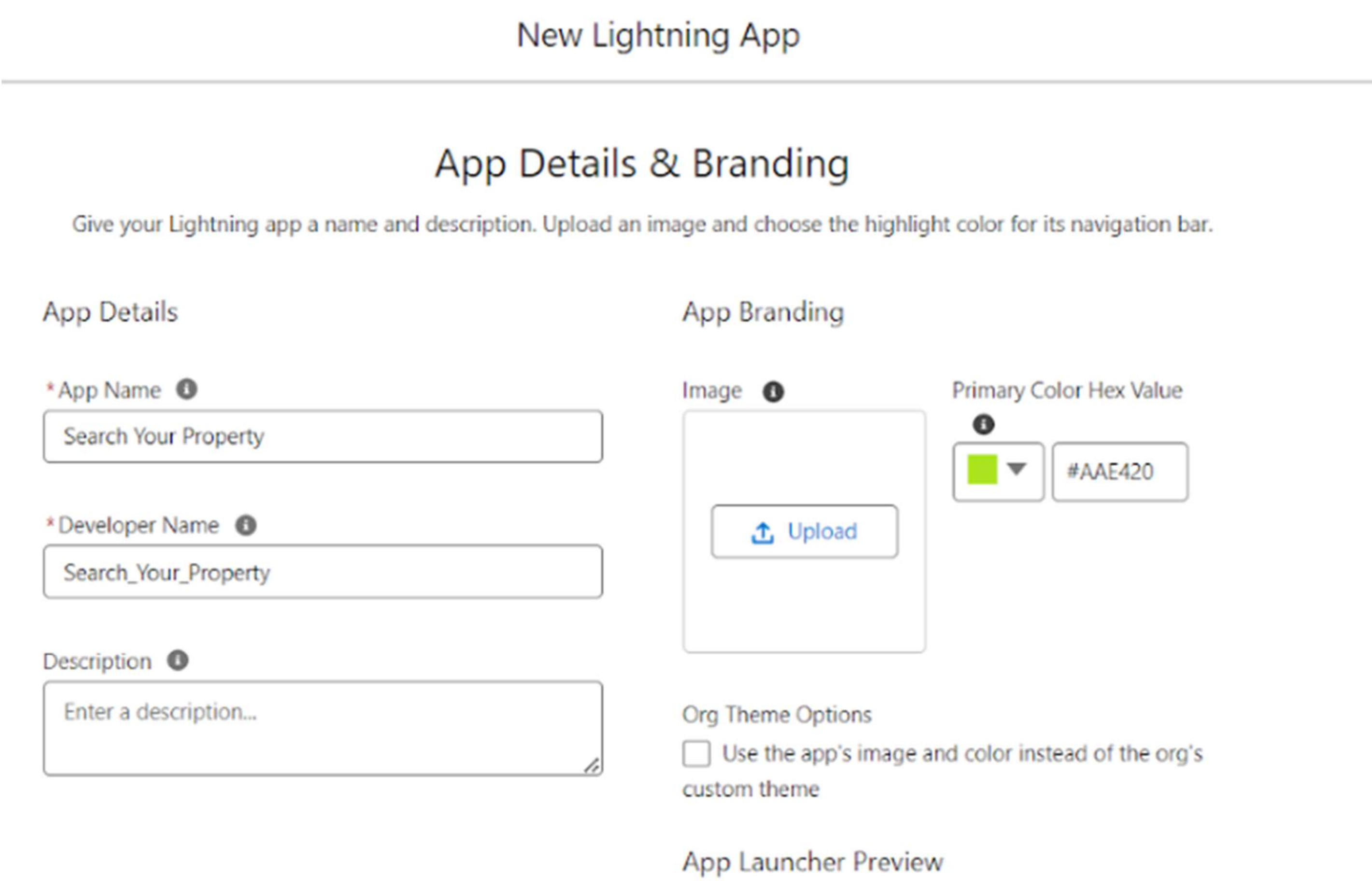
An App where the objects will be displayed

Activity1

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.



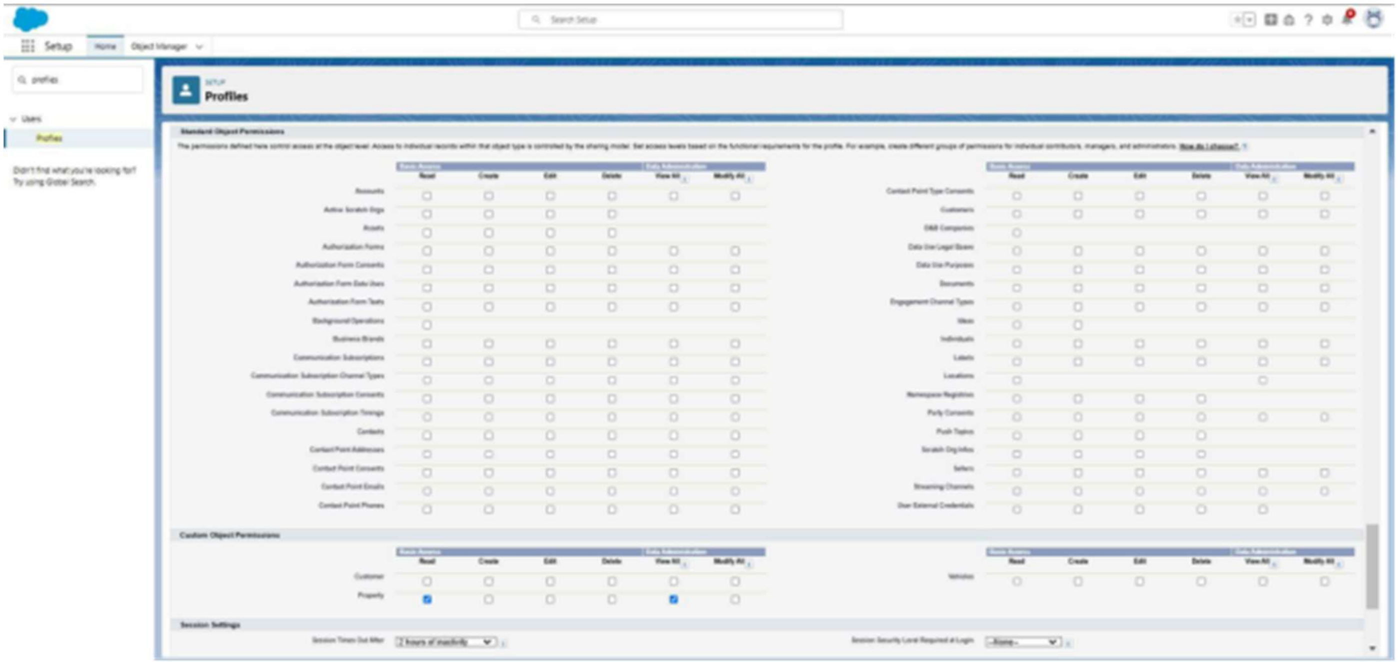
Create Profiles

Create profiles as per business requirement

Creating Customer Profiles

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. so 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 Profiles :-

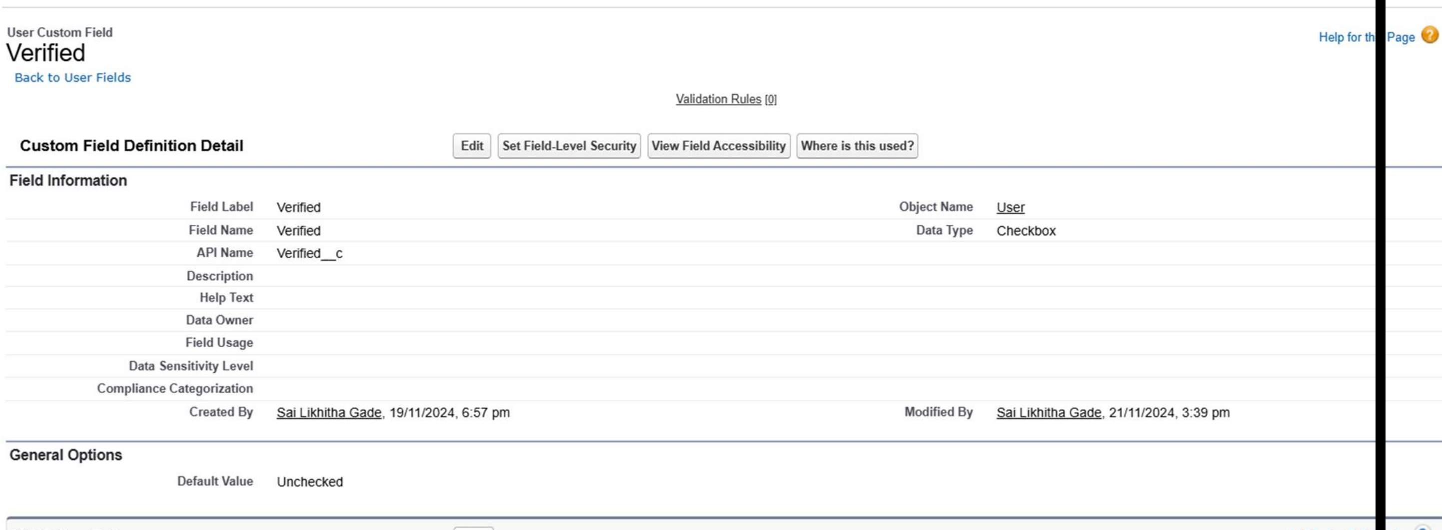
1. From Setup » Go to Profiles and Clone Salesforce Platform User and Name it “Manager”.
2. Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
3. Also Remove all the Standard Object Permissions.
4. Uncheck all the Custom Object Permissions and check only “modify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



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

User : 2

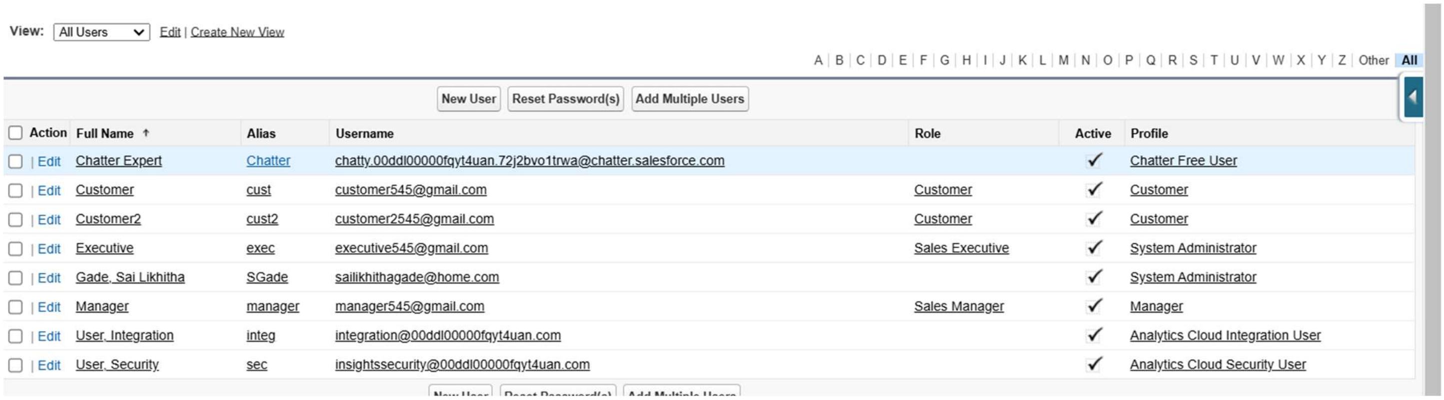
1. Go to Setup >Administration >› Users ›> New User
2. Last Name >> Manager
3. Role >> Sales Manager
4. License >> Salesforce Platform
5. Profile >> Manager
6. Save

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 : 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


View: All Users | Edit | Create New View

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

New User | Reset Password(s) | Add Multiple Users

Action	Full Name ↑	Alias	Username	Role	Active	Profile
Edit	Chatter Expert	Chatter	chatty.00ddl00000fqyt4uan.72j2bvo1trwa@chatter.salesforce.com		✓	Chatter Free User
Edit	Customer	cust	customer545@gmail.com	Customer	✓	Customer
Edit	Customer2	cust2	customer2545@gmail.com	Customer	✓	Customer
Edit	Executive	exec	executive545@gmail.com	Sales Executive	✓	System Administrator
Edit	Gade, Sai Likhitha	SGade	sailikhithagade@home.com		✓	System Administrator
Edit	Manager	manager	manager545@gmail.com	Sales Manager	✓	Manager
Edit	User, Integration	integ	integration@00ddl00000fqyt4uan.com		✓	Analytics Cloud Integration User
Edit	User, Security	sec	insightssecurity@00ddl00000fqyt4uan.com		✓	Analytics Cloud Security User

Create an Approval Process for Property Object

An Approval process to approve or reject the records as according

Activity1

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

1. Click Next and Select the initial Submiters »
2. Owner >> Property Owner
3. Roles >> Sales Manager
4. Save.

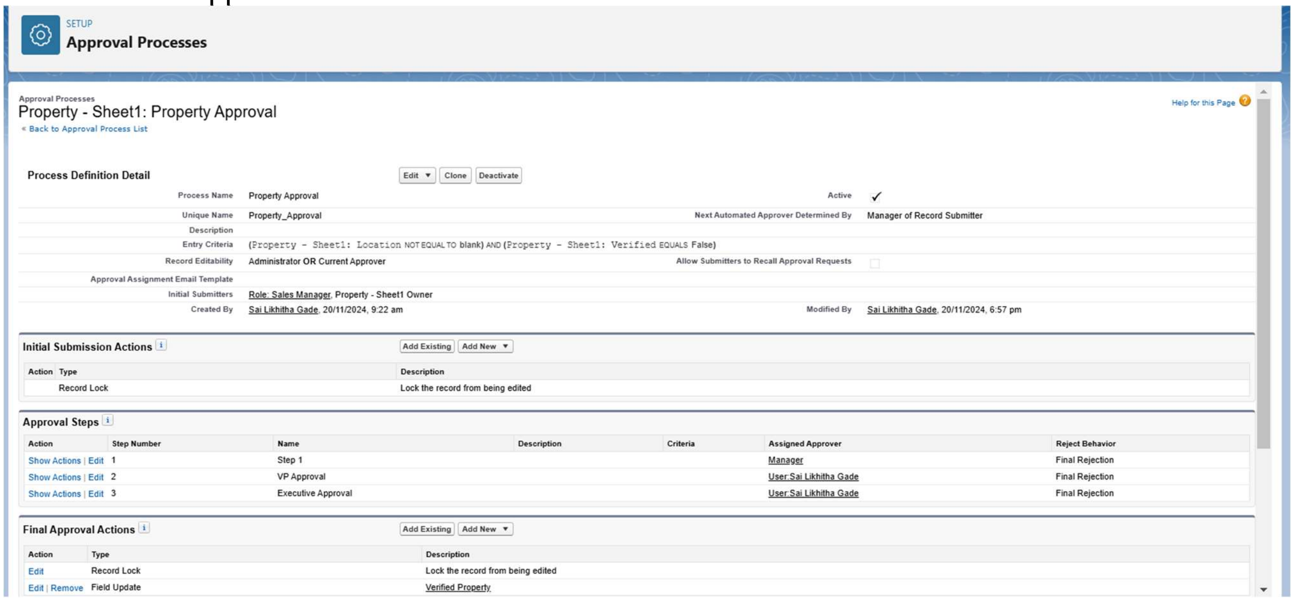
after saving we are directed to approval steps and we need to do as follows Add an approval step name "Executive Approval "
click next and select the Approver as "Sales Executive "and "Save" Add One field Update as "Verified Property"

1. Select Object »Property
2. Field to Update >> Verified
3. Field Data Type >» CheckBox
4. Select CheckBox Option as "True"
5. Save.

Add One field Update as "UnVerified Property"

1. Select Object » Property
2. Field to Update >» Verified
3. Field Data Type >» CheckBox
4. Select CheckBox Option as "False"
5. Save.

Activate the Approval Process.



Create a Record trigger ñow to submit the Approval Process Automatically

A flow that can submit the records directly for approval

Activity1

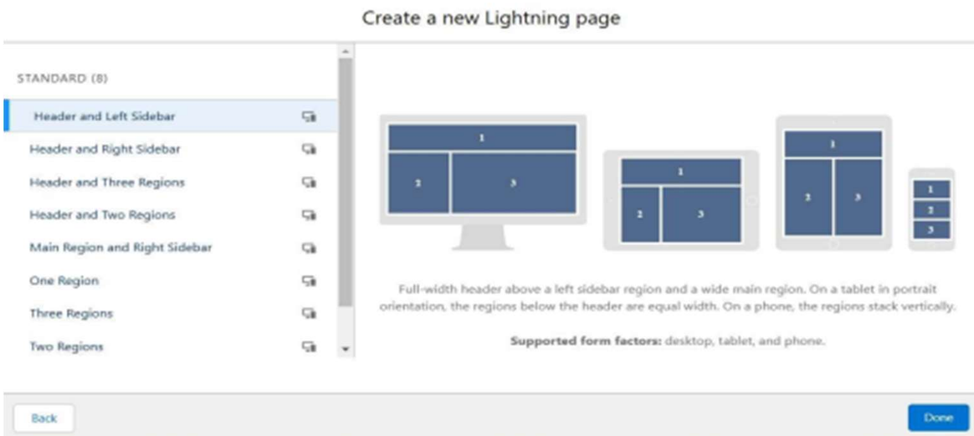
a. From Setup >> Search for Flows >> Click On New and Select “Record Trigger Flow”.
b. Select Object >> Property
c. Select “Trigger the flow when” >> “A record is created”
d. Set Entry Conditions >> “None”
e. Add a “Action” >> “Submit for Approval”
f. Give Label >> Approval for property
g. Record Id >> (!SRecord.Id)
h. Done

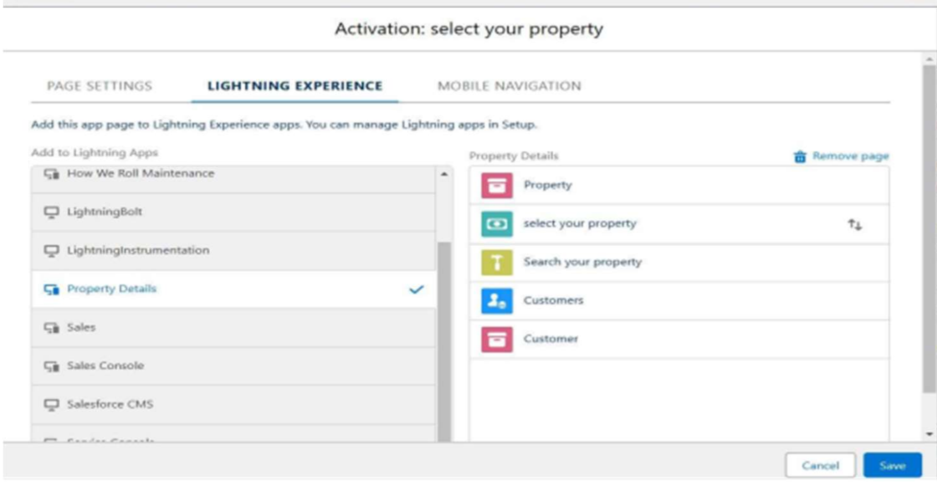
Save the Flow and Give label as “Property Approval” and “Activate”

Create an App Page on the Property details Object named as “Search Your Property”

Activity1

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

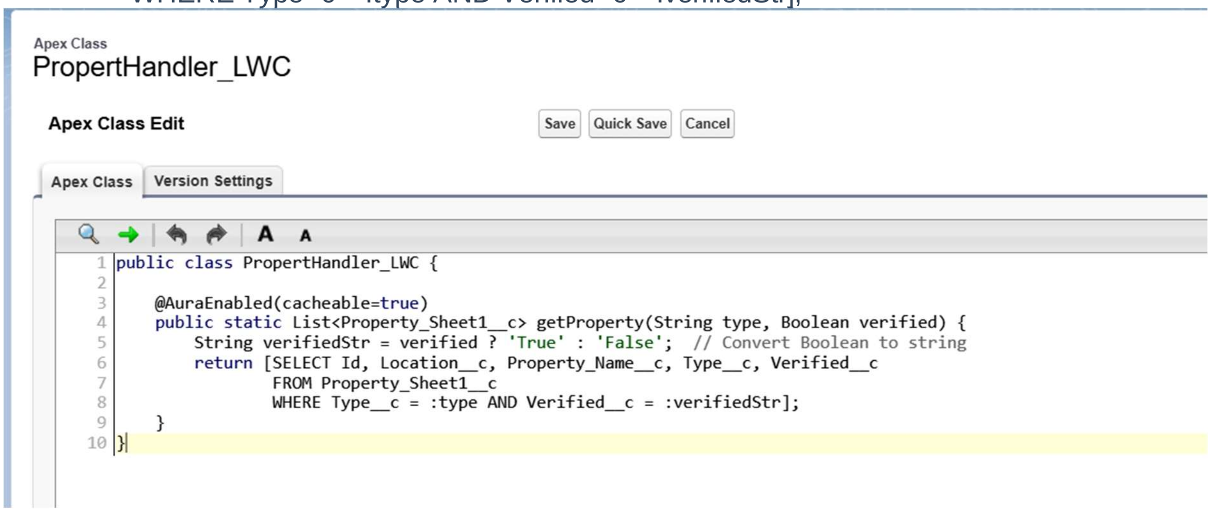
Create an LWC Component for the customers so that only verified customers can access the verified properties and non Verified customers can access non verified properties, and deploy it on “Search your Property Page”

Activity1

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

Code: -

```
public class PropertHandler_LWC (
@AuraEnabled(cacheable=true)
public static List<Propertyc> getProperty(String type, Boolean verified) ( String verifiedstr =
verified ? 'true' : 'false' // Convert boolean to string return [SELECT Id, Location  c,
Property_Namec, Type  c, Verified  c
    FROM Property  c
    WHERE Type  c = :type AND Verified  c = :verifiedStr];
```



1. Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.
2. Enter your login id and password to authorize your org.

3. Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to. (Example -
4. 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=(istrue)>
     <div class="slds-box">
          <lightning-datatable key-field="id" data=(propertylist} columns=(columns}></lightning-
datatable>
     </div>
    </templates
    <template if:false=(isfalse}>
     <div class="slds-box">
      <div style="font-size: 15px;"><b>No properties Are Found !!</b></div>
     </div>
    </template›
   </lightning-card>
</templates>
```

```
<template>
    <lightning-card title="Properties" icon-name="standard:account">
      <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">
            <!-- Combobox for selecting Property Type -->
            <div class="slds-col slds-size_5-of-6">
              <lightning-combobox
                name="Type"
                label="Property Type"
                value={typevar}
                placeholder="Select Property Type"
                options={propertyOptions}
                onchange={changeHandler}>
              </lightning-combobox>
            </div>
```

```
        <div class="slds-col slds-size_1-of-6">
          <br>
          <lightning-button-icon
            variant="neutral"
            icon-name="utility:search"
            alternative-text="Search"
            onclick={handleClick}>
          </lightning-button-icon>
        </div>
      </div>
    </div>
  </div>

  <!-- Conditional Rendering -->
  <template if:true={istrue}>
    <div class="slds-box">
      <lightning-datatable
        key-field="id"
        data={propertyList}
        columns={columns}>
      </lightning-datatable>
    </div>
  </template>
```

```
  <template if:false={istrue}>
    <div class="slds-box">
      <div style="font-size: 15px;"><b>No properties are found!</b></div>
    </div>
  </template>
</lightning-card>
/template>
```

1. In Your Js File Write this code : -

Code :-

import (LightningElement, api, track, wire) from 'lwc';

import getProperty from "@salesforce/apex/PropertHandler_LWC.getProperty"

import (getRecord } from 'lightning/uiRecordApi'; import USER ID from

'@salesforce/user/Id';

export default class C 01_Property_M anagement extends LightningElement (@api recordId

userId = USER_ID; verifiedvar typevar

isfalse = true; istrue = false;

@track propertylist = [];

columns = [
(label: 'Property Name', fieldName: 'Property_Name c'), (label: 'Property

Type', fieldName: 'Type c'),

(label: 'Property Location', fieldName: 'Locationc'), (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)
```

```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from "@salesforce/apex/PropertHandler_LWC.getProperty";
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';

export default class MyFirstComponent extends LightningElement {
    @api recordId;
    userId = USER_ID;
    verifiedVar;  // Correctly renamed to camelCase
    typeVar;  // Correctly renamed to camelCase
    isFalse = true;  // Correctly renamed to camelCase
    isTrue = false;  // Correctly renamed to camelCase

    @track propertyList = [];  // Correctly renamed to camelCase
    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" }
    ];

    propertyOptions = [  // Correctly renamed to camelCase
        { label: "Commercial", value: "Commercial" },
        { label: "Residential", value: "Residential" },
        { label: "Rental", value: "Rental" }
    ];

    // Wire user record to retrieve Verified__c field
    @wire(getRecord, { recordId: "$userId", fields: ['User.Verified__c'] })
    recordFunction({ data, error }) {
```

```
recordFunction({ data, error }) {
    if (data) {
        this.verifiedVar = data.fields.Verified__c.value;  // Set verifiedVar reactively
    } else {
        console.error(error);
    }
}

// Handles changes in the combobox (property type)
changeHandler(event) {
    this.typeVar = event.target.value;
}

// Handles the button click to fetch properties
handleClick() {
    getProperty({ type: this.typeVar, verified: this.verifiedVar })
        .then((result) => {
            this.isFalse = true;
            if (result != null && result.length !== 0) {
                this.isTrue = true;
                this.propertyList = result;
            } else {
                this.isFalse = false;
                this.isTrue = false;
            }
        })
        .catch((error) => {
            console.log(error);
        });
```

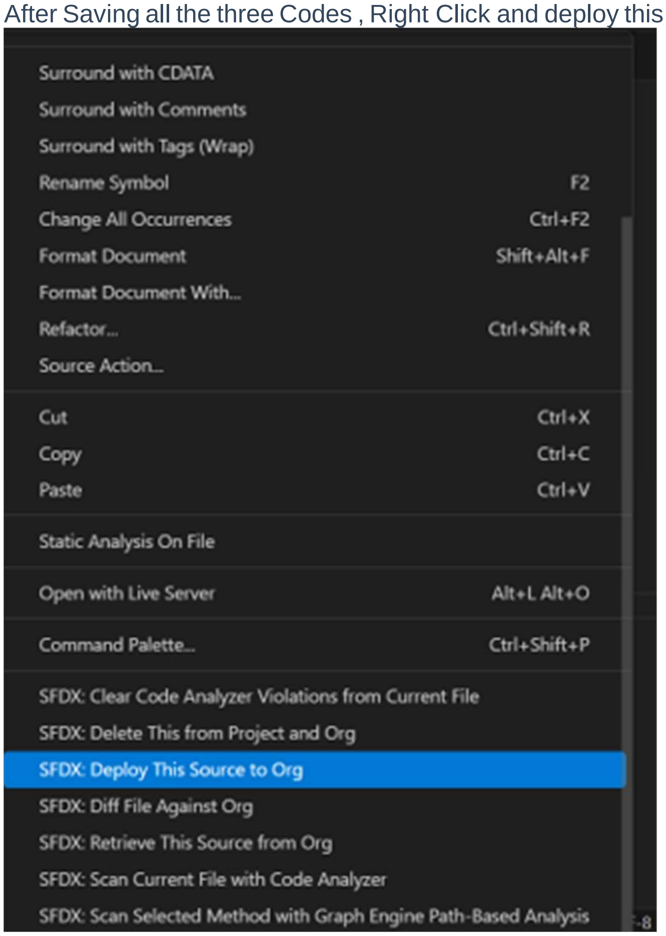
1. In Your metafile give your targets to deploy the component.

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</targets
      <target>lightning__AppPage</targets
      <target>lightning__HomePage</targets
         </targets>
</LightningComponentBundle>
```

```xml
<?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>
```

After Saving all the three Codes , Right Click and deploy this component to the org

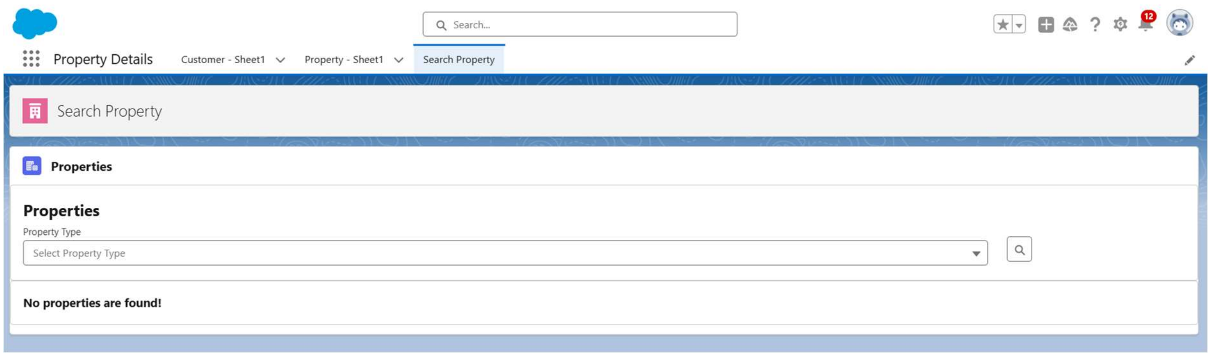


Drag this Component to your App Page

Adding the Component to your Page

Activity1

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 bne directed to app pages then
3. 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.

1. Activity1

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