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

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

2. Property Management

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

3. Requirement Matching

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

4. Lead Tracking

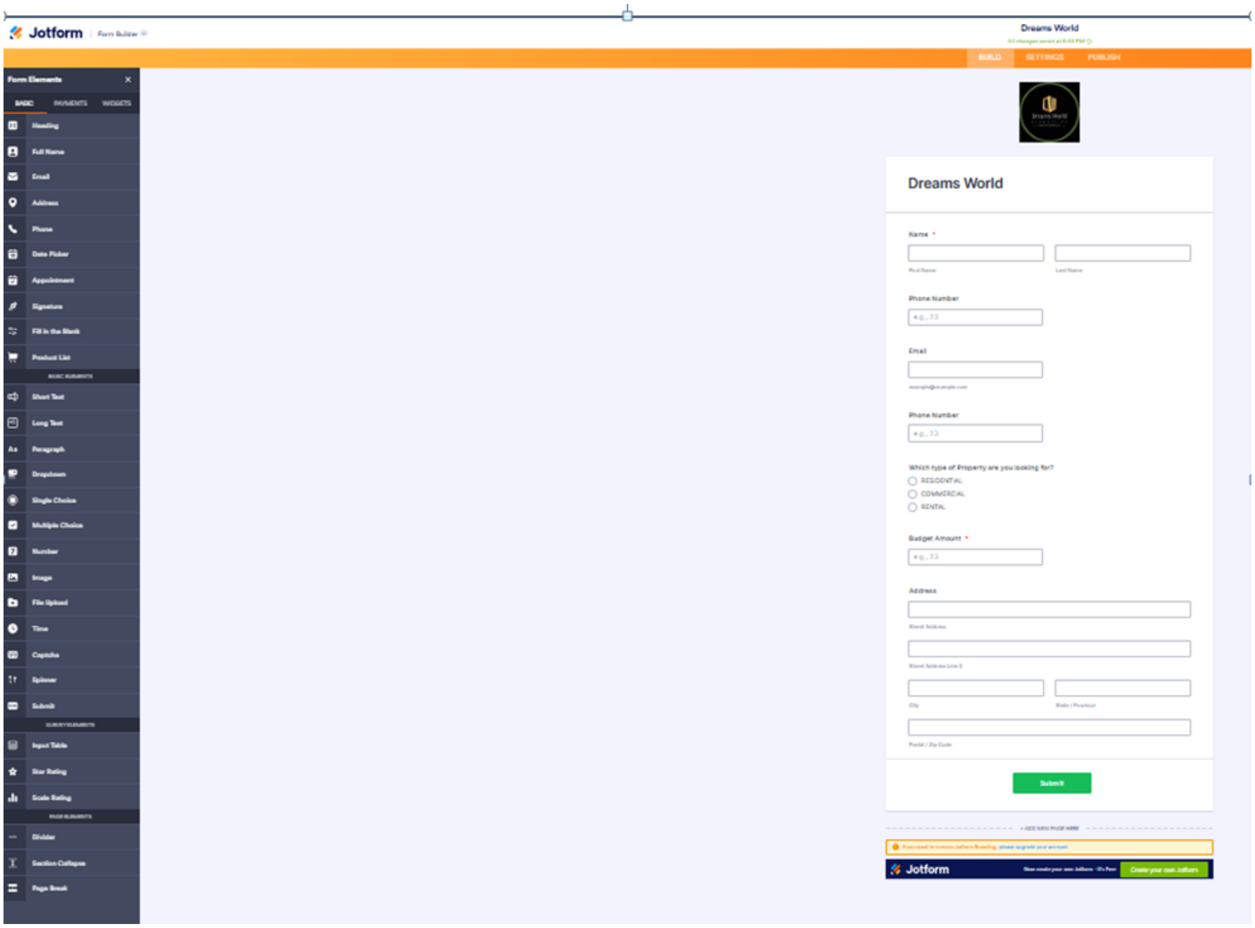
1. Manage inquiries and follow up with potential clients.
2. Schedule meetings and site visits.
3. Assign leads to specific team members.

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/243229403451046



Create Objects from Spreadsheet

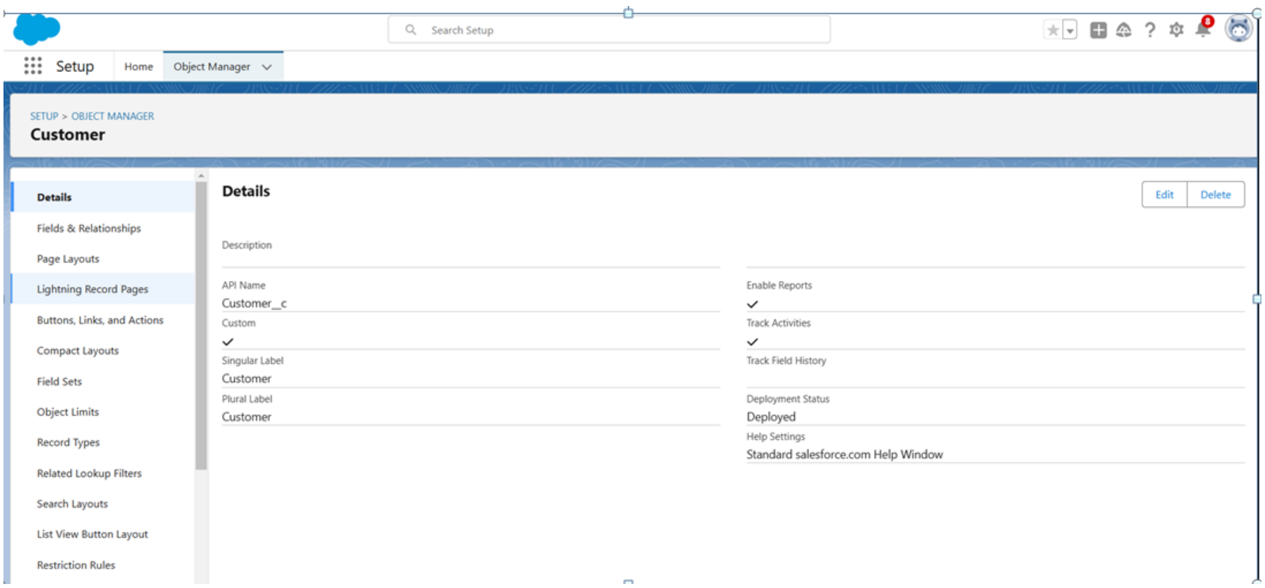
Directly Creating Objects from Spreadsheet in Salesforce

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

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

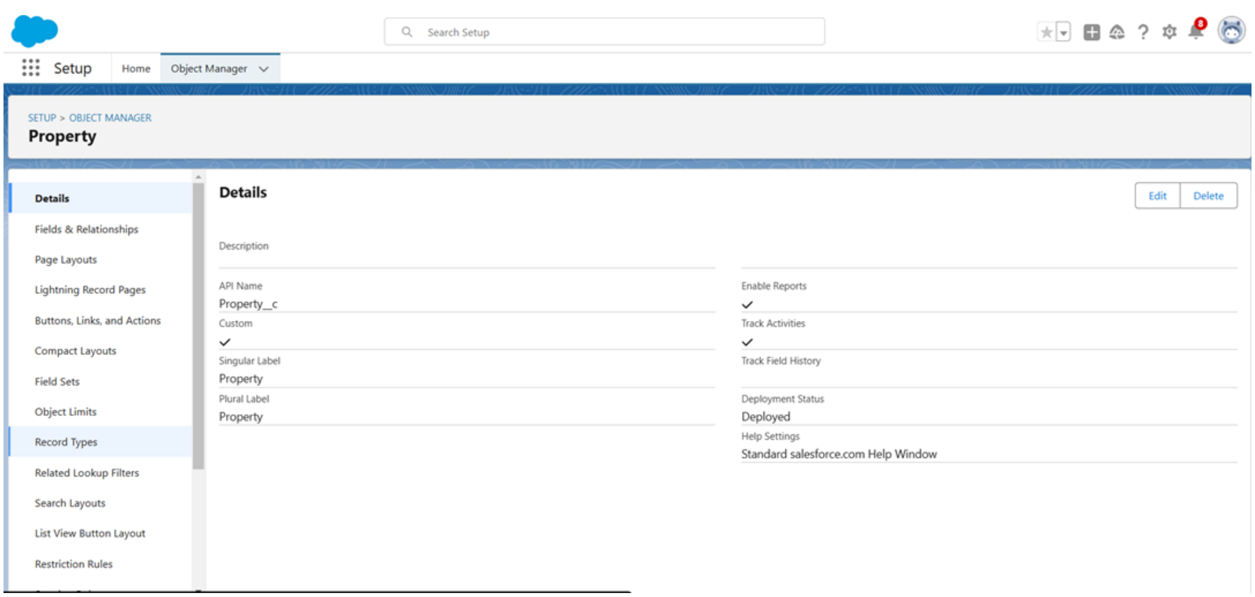


Creating Property Object:

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

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



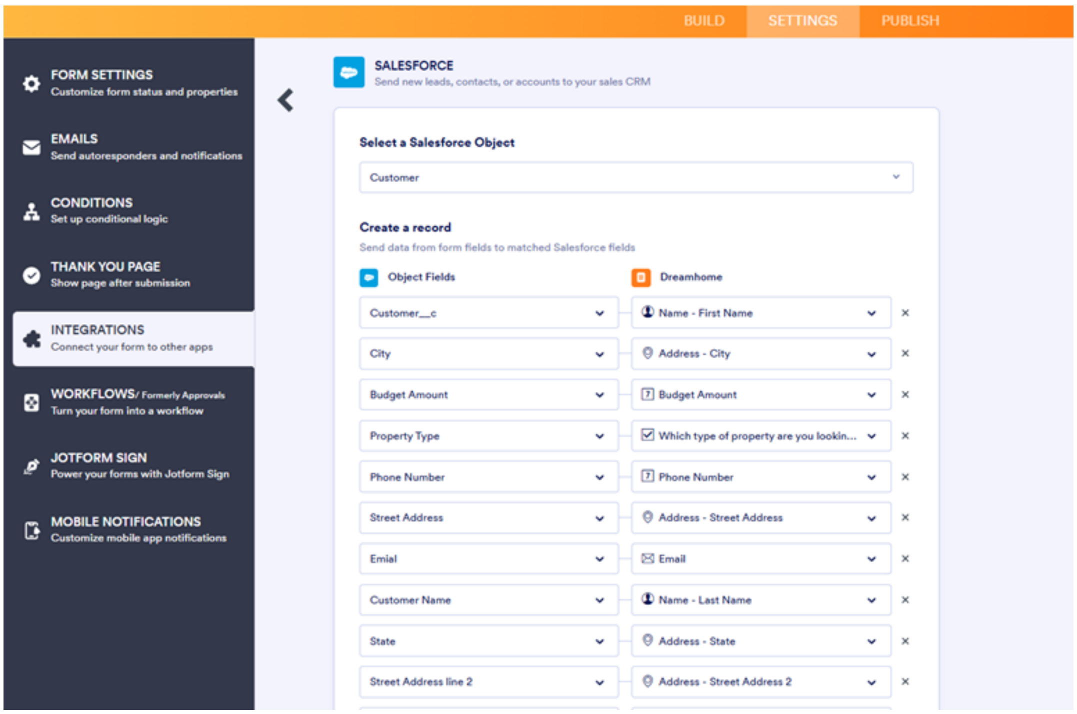
Integrate Jotform with Salesforce Platform

In this Milestone we are going to integrate jotform with Salesforce

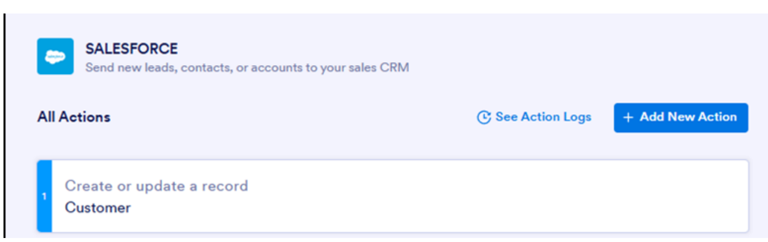
Activity :-1

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".

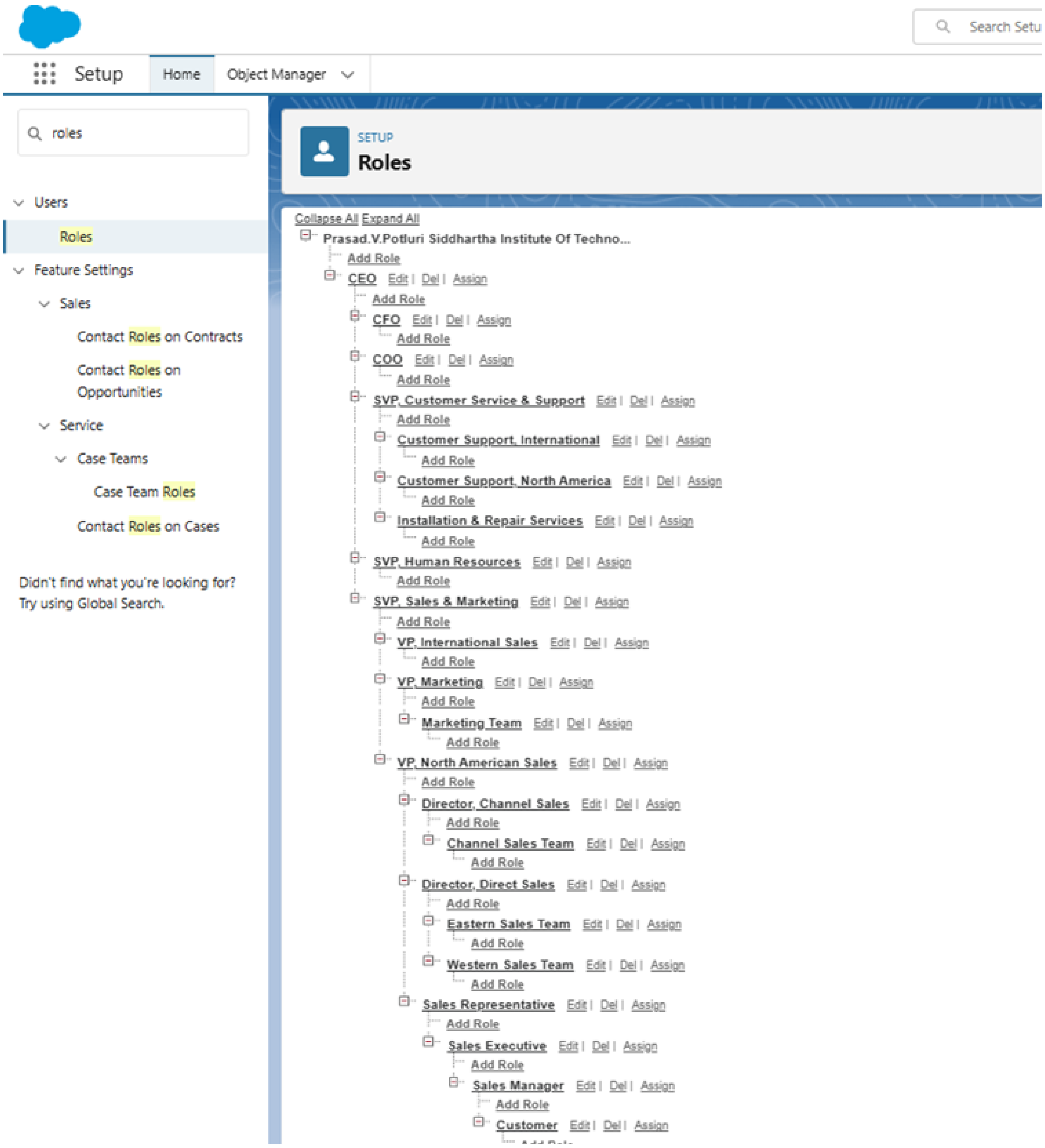


Create Roles

Here we need to Create Roles as per business requirement

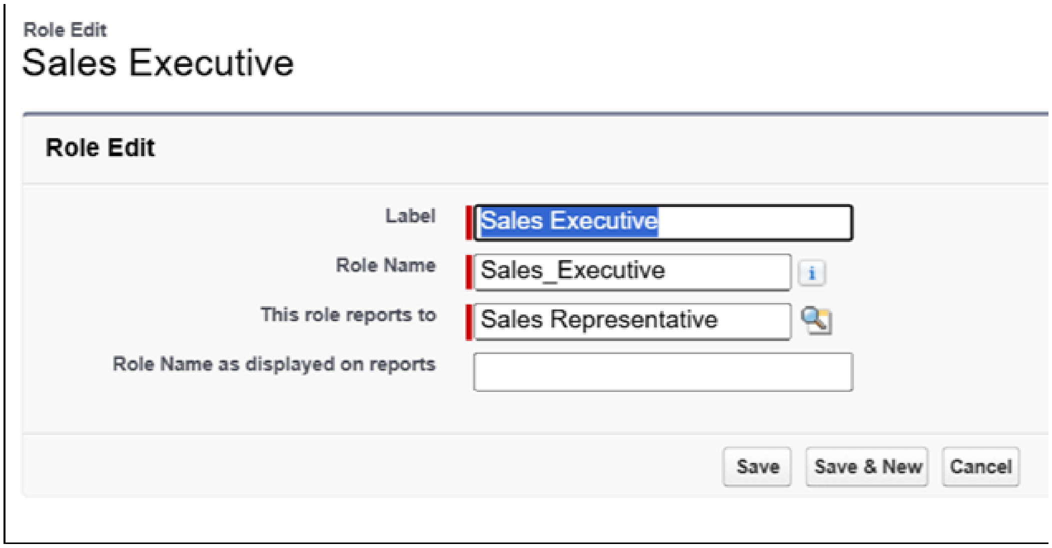
Activity :- 1

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.



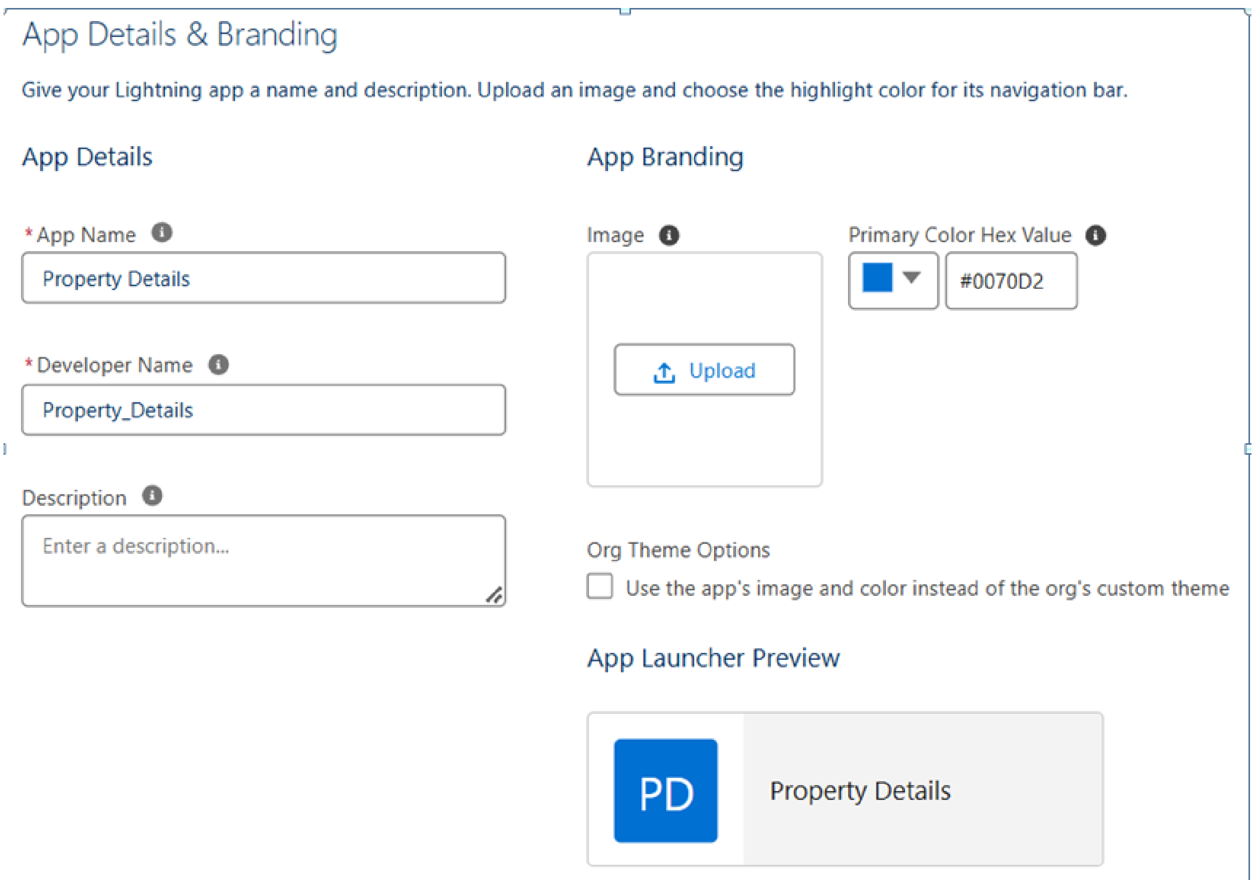
1. Similarly Create a Role Name "Sales Manager" below Sales Executive which reports to Sales Executive, Also Add a Role below Sales Manager labeled as "Customer" which reports to Sales Manager.

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 Narme it as "Property Details" and add "Customer" and "Property" Object.
2. Click Next >> Next >> Save and Add "System Admin "Profile.

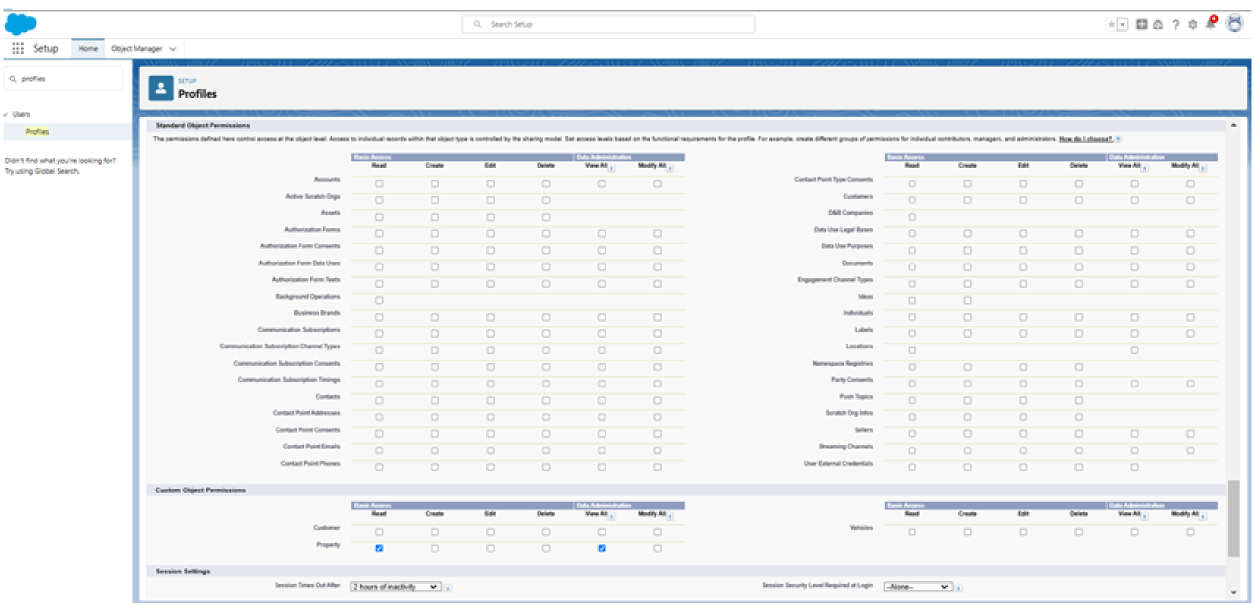


Create Profiles

Create profiles as per business requirement

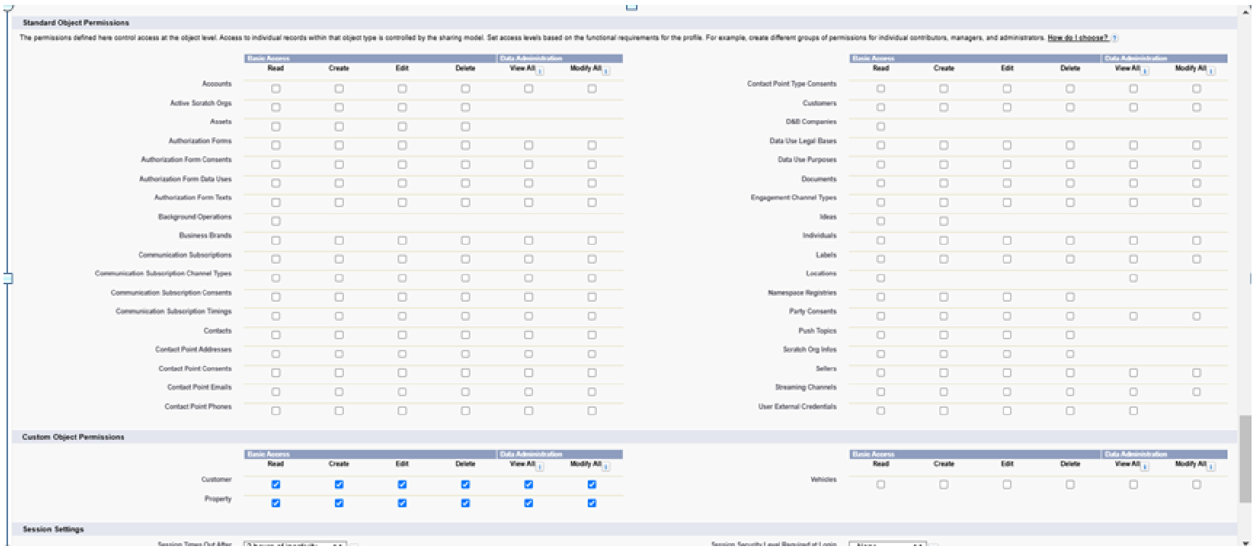
Creating Customer Profile :-

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".

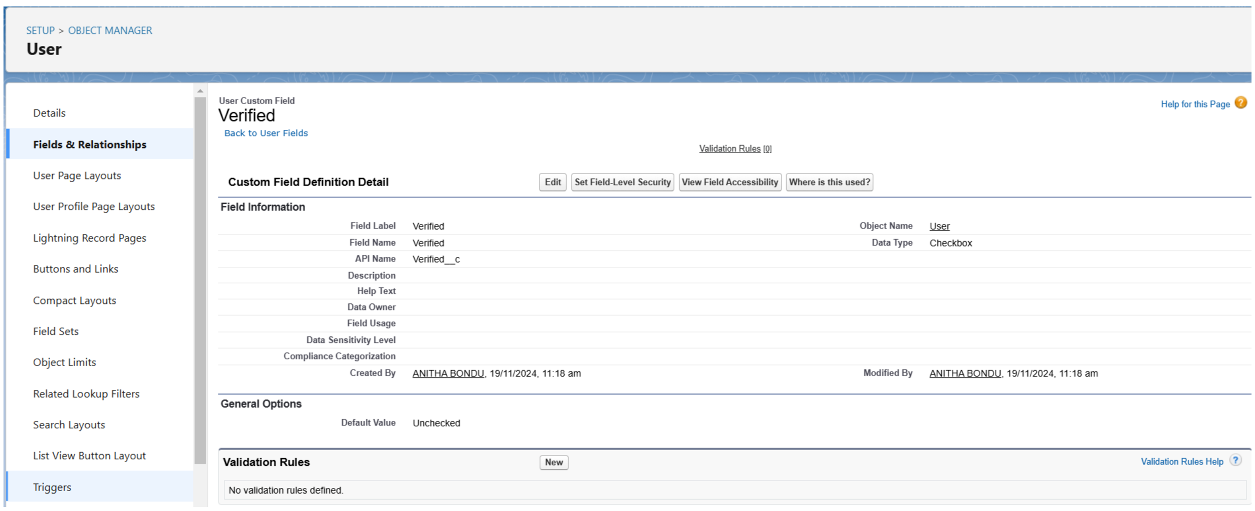


Create a Check Box field on user

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"

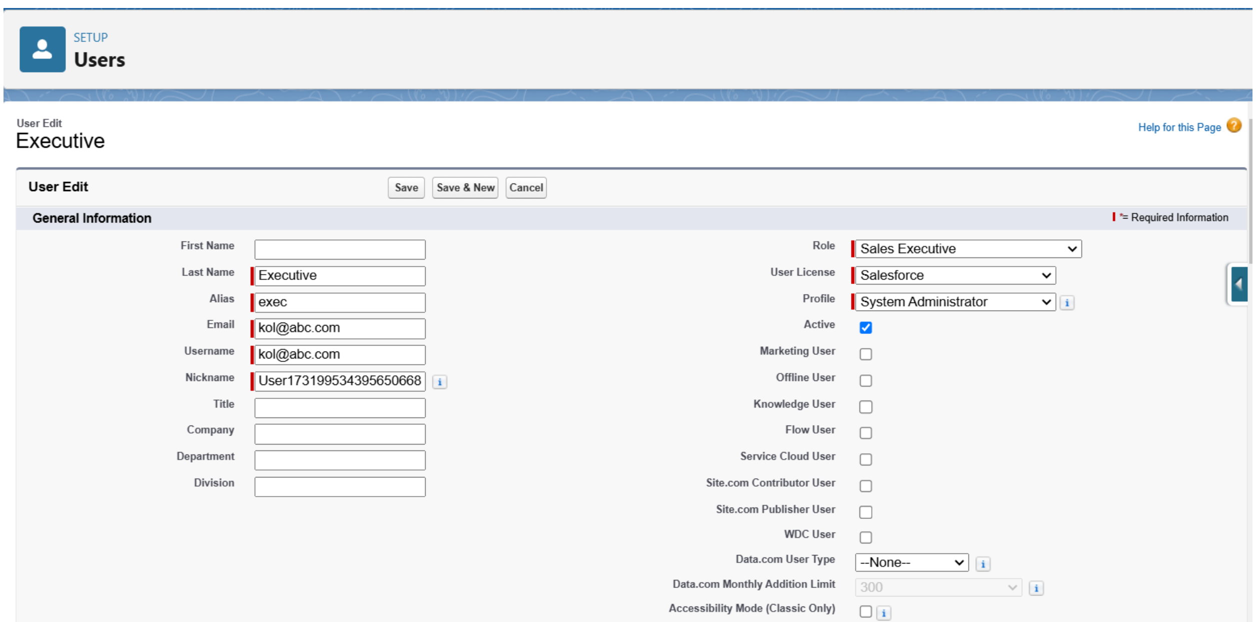


Create Users

Create three different users with three different Roles and profiles as we have mentioned above. Here we are going to create 4 users.

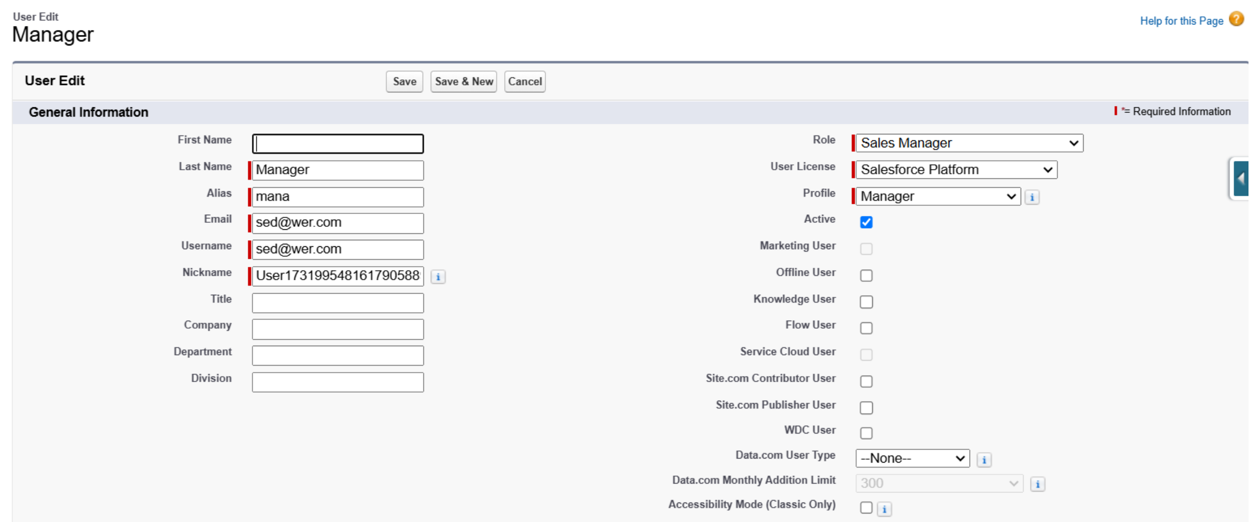
User: 1

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



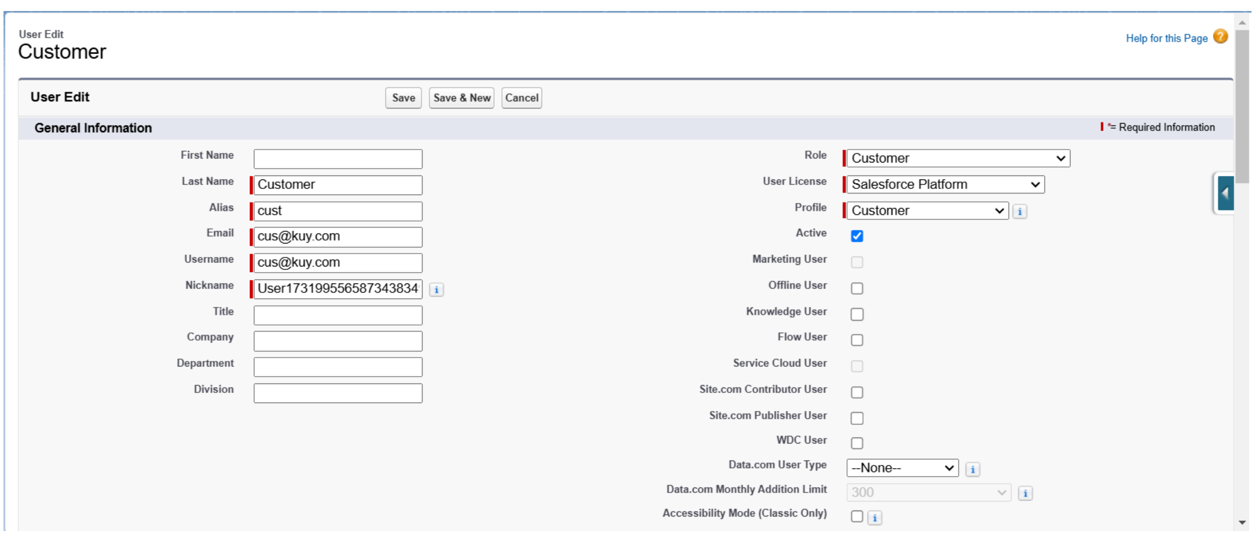
User: 2

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



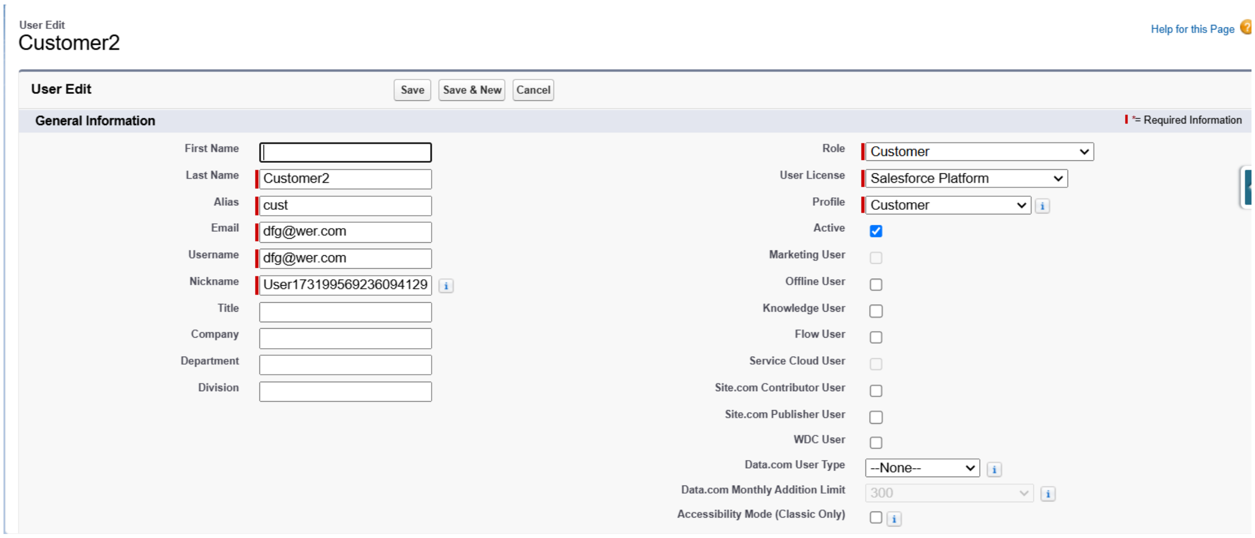
User: 3

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



Create an Approval Process for Property Object

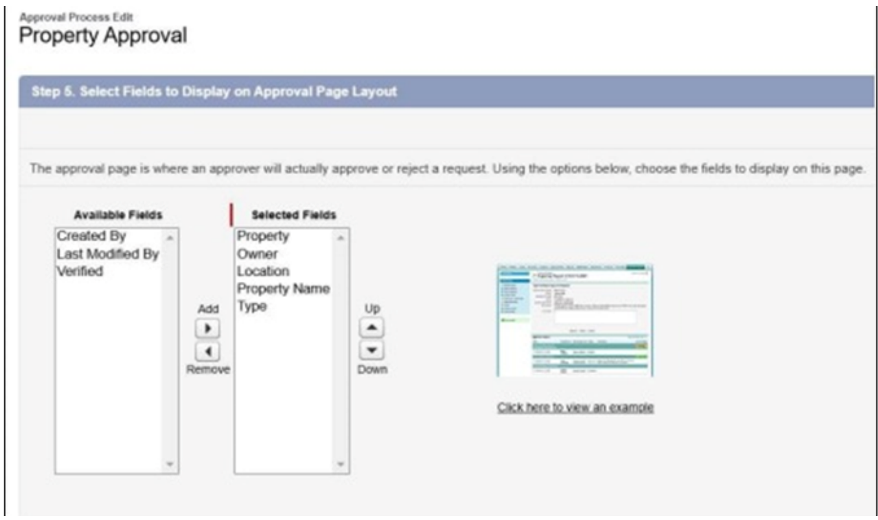
An Approval process to approve or reject the records as according

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 Submiters >>

1. Owner >> Property Owner
2. 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"

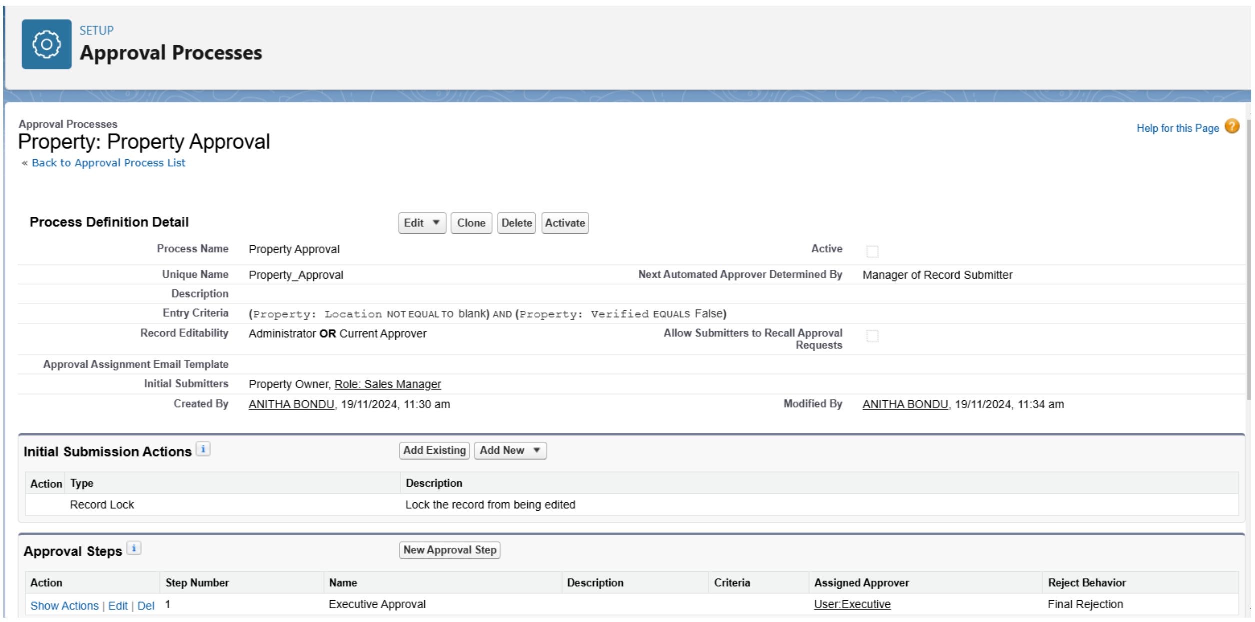
Add One field Update as "Verified Property"

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.



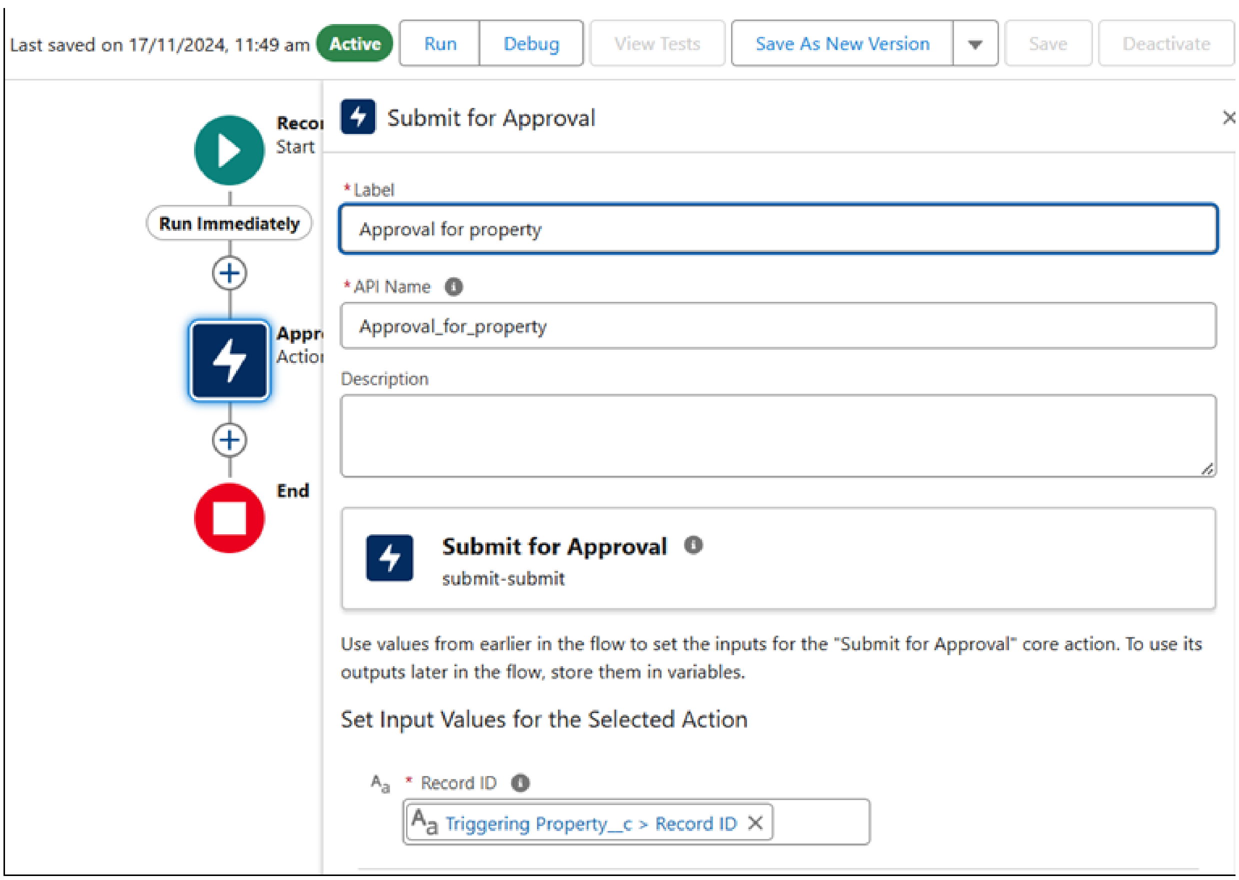
Create a Record trigger flow to submit the Approval Process Automatically

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"



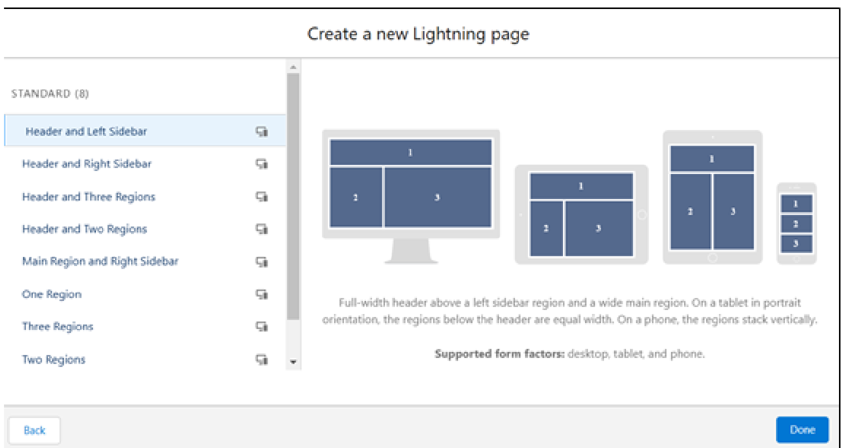
Create an App Page

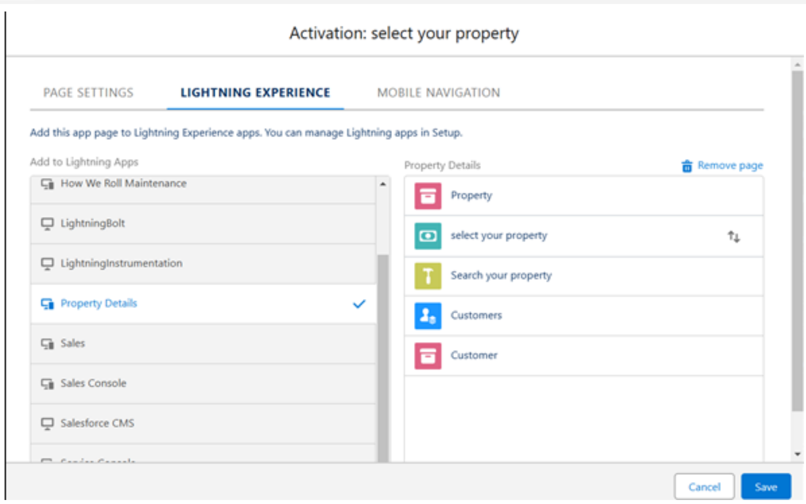
Create an App Page on the Property details Object named as "Search Your Property"

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

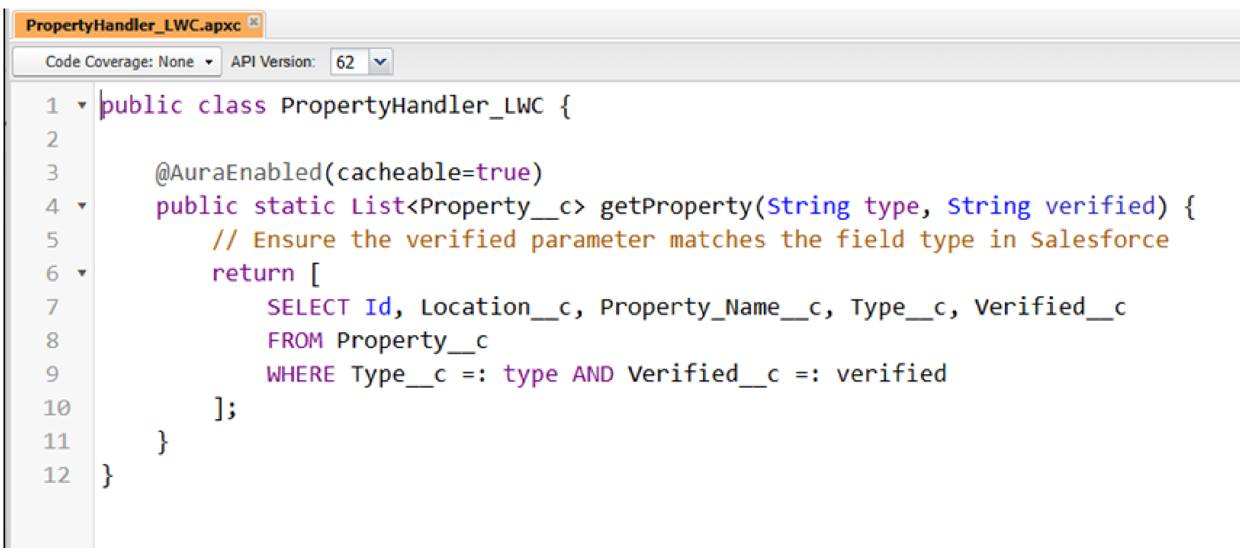
Create an LWC Component for the customers so that only verified customers can access the verified properties and non Verified customers can access non verified properties, and deploy it on "Search your Property Page"

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<Property__c> 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
      ];
   }
}
```



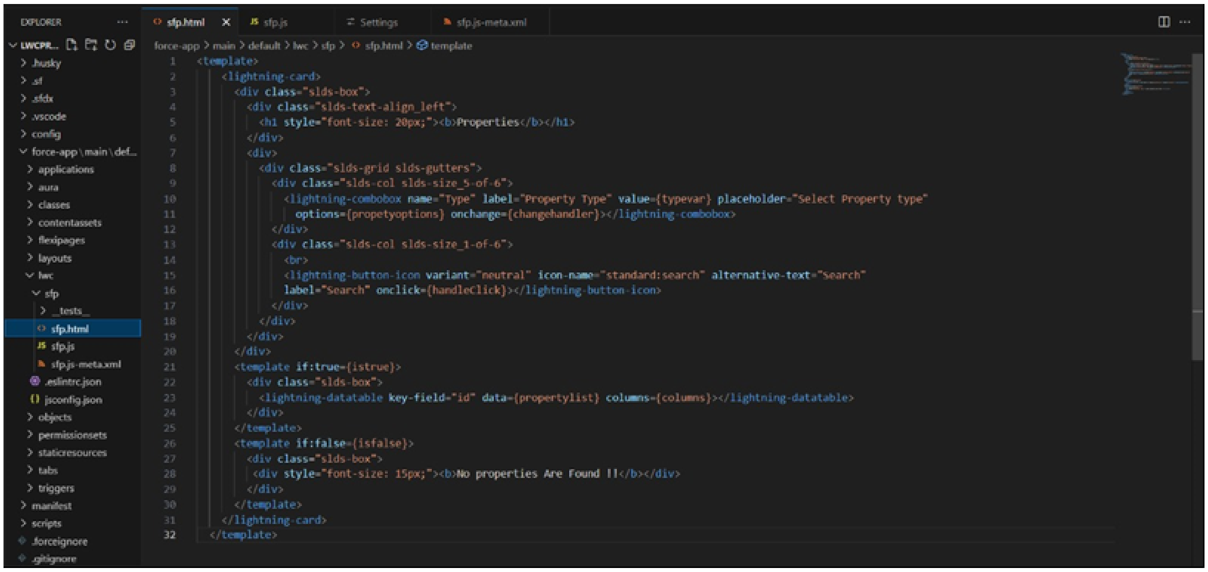

```
public class PropertyHandler_LWC {

    @AuraEnabled(cacheable=true)
    public static List<Property__c> 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
        ];
    }
}
```

1. Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.
2. Enter your login ic and password to authorize your org.
3. Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to.
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>
      </template>
      <template if:false={isfalse}>
        <div class="slds-box">
div style="font-size: 15px;"><b>No properties Are Found !!</b></div>
        </div>
      </template>
    </lightning-card>
 </template>
```

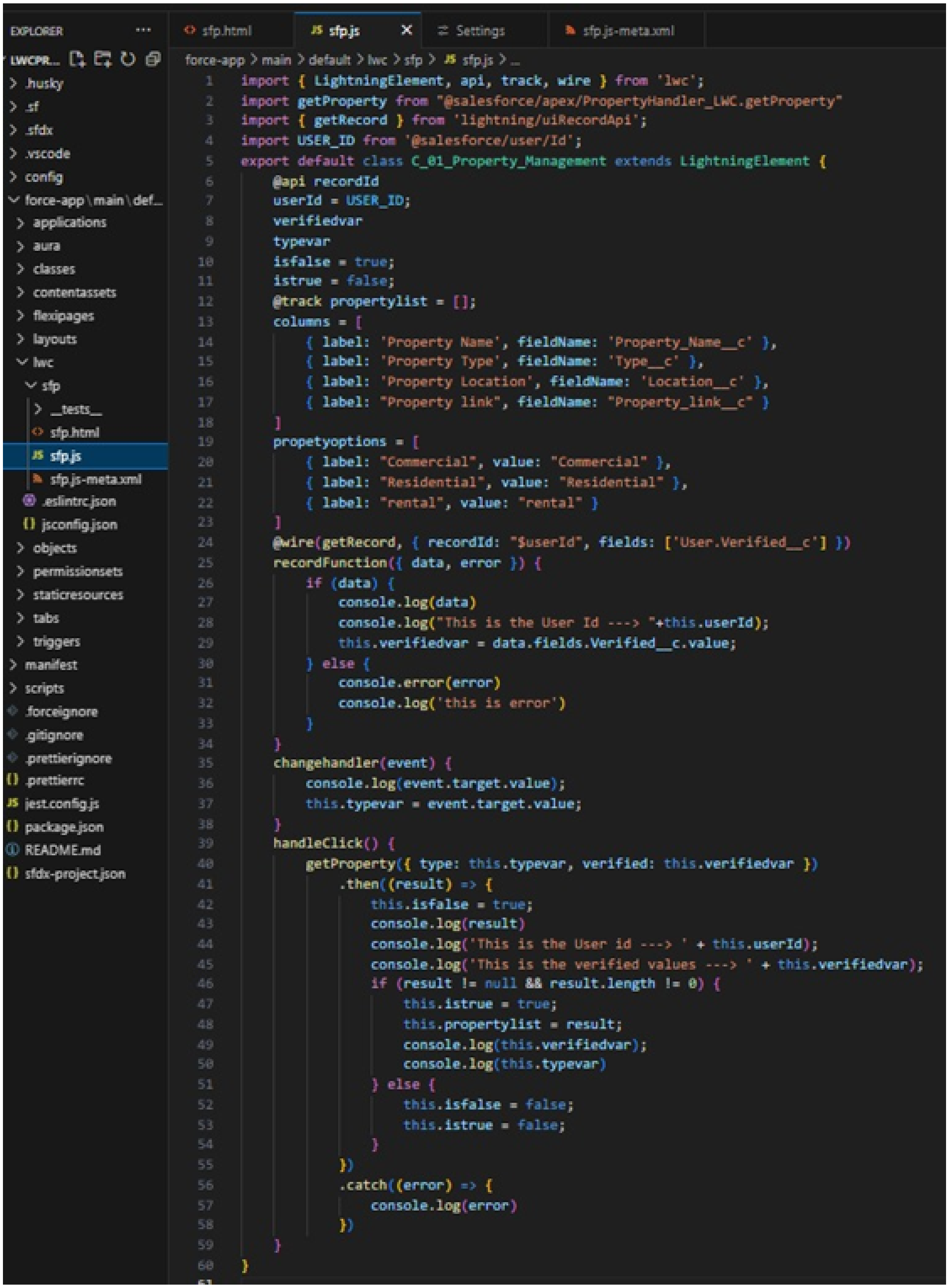


1. In your Js File, write this code

Code:

```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from "@salesforce/apex/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;
    istrue = false;
    @track propertylist = [];
    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)
            })
    }
}
```

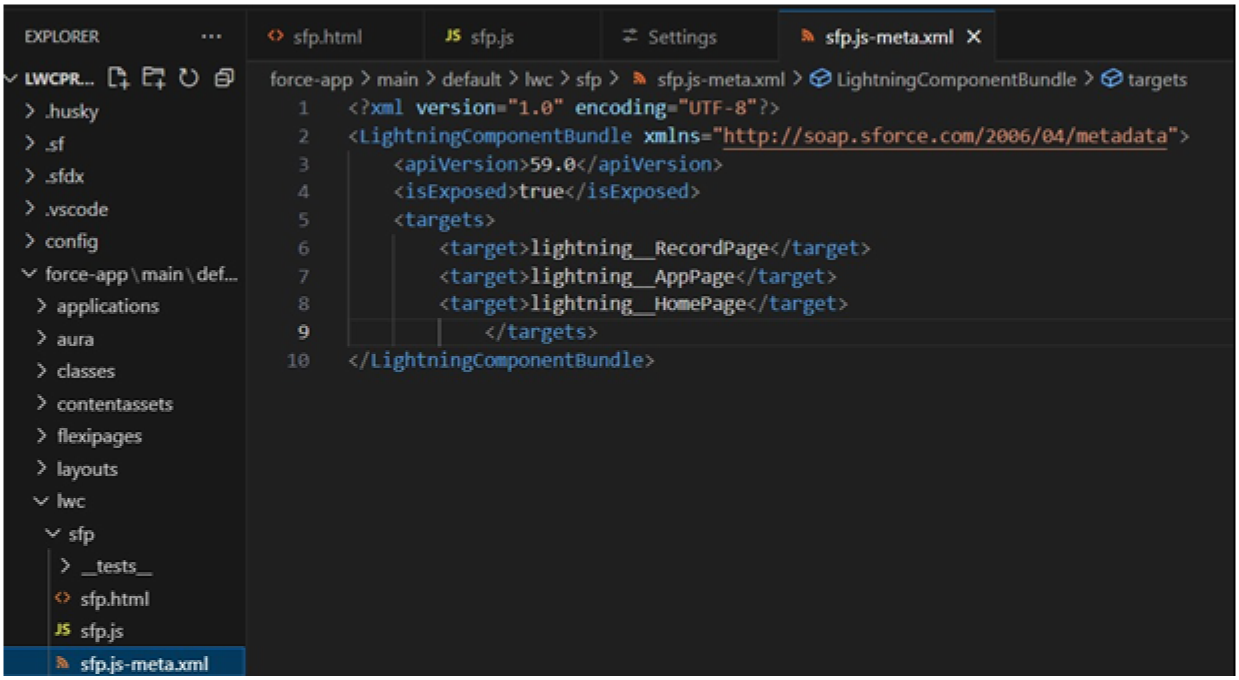



```javascript
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from "@salesforce/apex/PropertyHandler_LWC.getProperty"
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';
export default class C_01_Property_Management extends LightningElement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

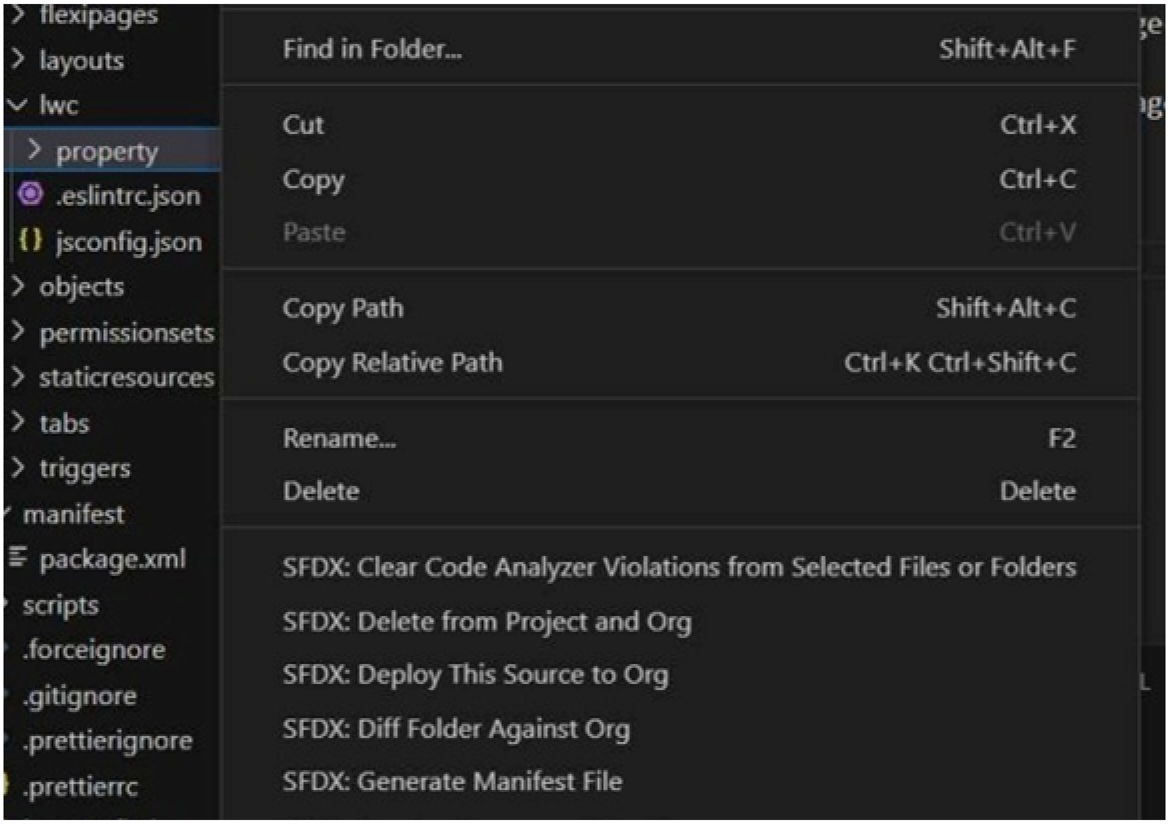
7. In your Metafile, give your targets to deploy the component.

Code:

```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>
```



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

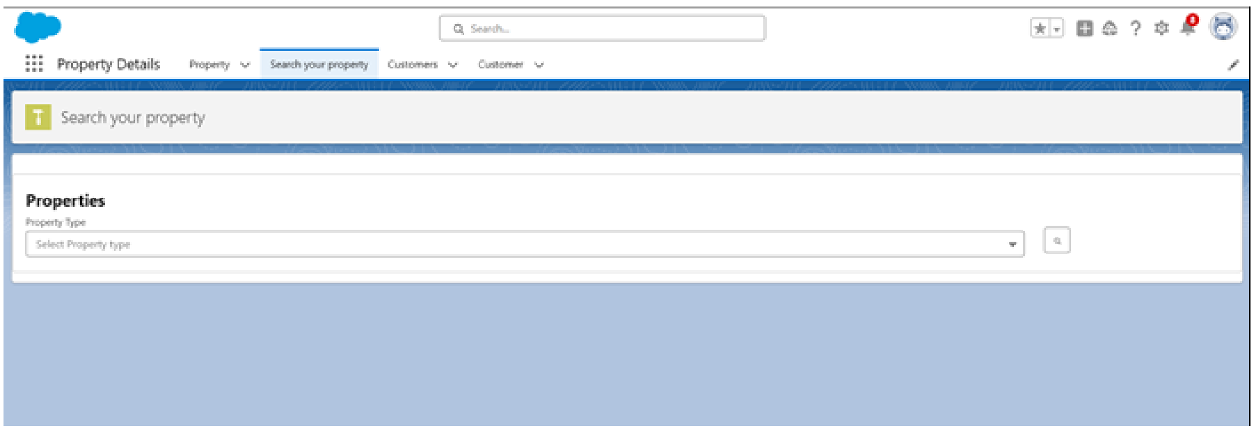

flexipages
layouts
lwc
property
.eslintrc.json
jsconfig.json
objects
permissionsets
staticresources
tabs
triggers
manifest
package.xml
scripts
.forceignore
.gitignore
.prettierignore
.prettierrc
Find in Folder... Shift+Alt+F
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Copy Path Shift+Alt+C
Copy Relative Path Ctrl+K Ctrl+Shift+C
Rename... F2
Delete Delete
SFDX: Clear Code Analyzer Violations from Selected Files or Folders
SFDX: Delete from Project and Org
SFDX: Deploy This Source to Org
SFDX: Diff Folder Against Org
SFDX: Generate Manifest File

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 bne 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".