# A CRM Application to Handle the Clients and their property Related Requirements

**Project Overview : -**

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.

**Project Flow:**

Milestone 1 : Create a jotform and integrate it with the org to create records automatically
Milestone 2 : Create Objects from Spreadsheet
Milestone 3 : Integrate Jotform with Salesforce Platform
Milestone 4 : Create Roles
Milestone 5 : Create a Property Details App
Milestone 6 : Create Profiles
Milestone 7 : Create a Checkbox Field on User
Milestone 8 : Create Users
Milestone 9 : Create an Approval Process for Property Object
Milestone 10 : Create a Record trigger flow to submit the Approval Process Automatically.
Milestone 11 : Create an App Page
Milestone 11 : Create an LWC Component
Milestone 11 : Drag this Component To Your App Page

## Requirements : -

### 1) Website Integration Requirements :

Implement a form on the website for users to express interest in property listings.
Ensure the form captures essential details such as name, contact information, preferred property type, location, budget, etc. Set up validation rules to ensure data accuracy and completeness.
Integrate the form submission process with Salesforce.

### 2) Salesforce Configuration Requirements :

Set up Salesforce objects and fields to store customer data. This includes fields for name, contact information, preferences, approval status, etc. Define workflows or processes to automate the creation of records when a user submits the form on the website.
Implement validation rules and data integrity checks to maintain data quality.
Configure Salesforce security

### 3) Approval Process Requirements :

Define criteria for categorizing users as approved or non-approved based on specific parameters such as budget, property preferences, etc. Implement an approval process in Salesforce to review and approve users. Set up email notifications or alerts to notify relevant stakeholders when a user is approved or rejected.

### 4) User Experience Requirements:

Design user interfaces in Salesforce for managing customer records, approval processes, and property listings. Ensure a seamless user experience for both customers and internal users interacting with Salesforce.Provide training and documentation for internal staff on how to use the Salesforce system effectively.

**5) Integration Testing and Quality Assurance Requirements:**

Conduct thorough testing of the integration between the website and Salesforce to ensure data is accurately captured and transferred. Perform end-to-end testing of the approval process to verify that users are categorized correctly and granted appropriate access.
Identify and resolve any issues or bugs encountered during testing.

**6) Documentation and Maintenance Requirements:**

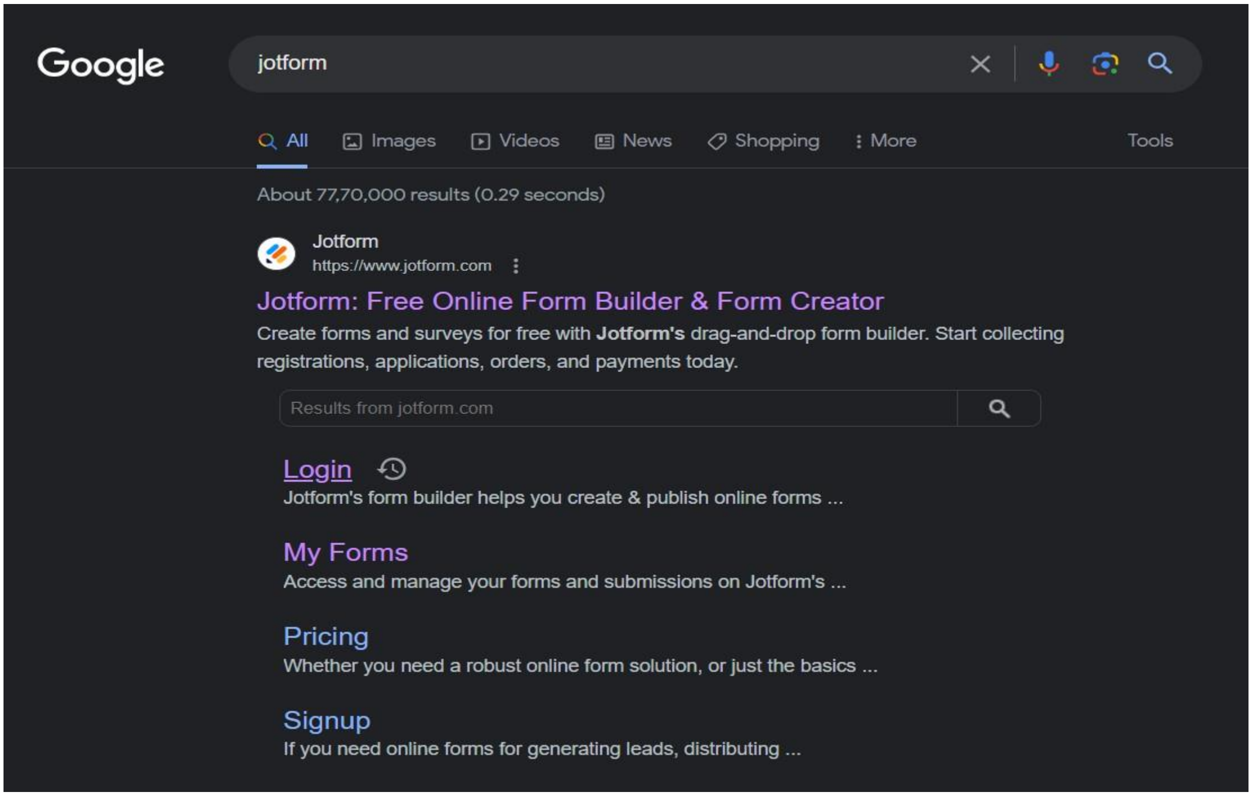
Document the integration architecture, data flows, and configuration details for future reference. Establish procedures for ongoing maintenance and updates to the integration and Salesforce configuration. Provide ongoing support and training for users to address any issues or questions that may arise.

***After Creating the Salesforce org Start with the First Milestone***

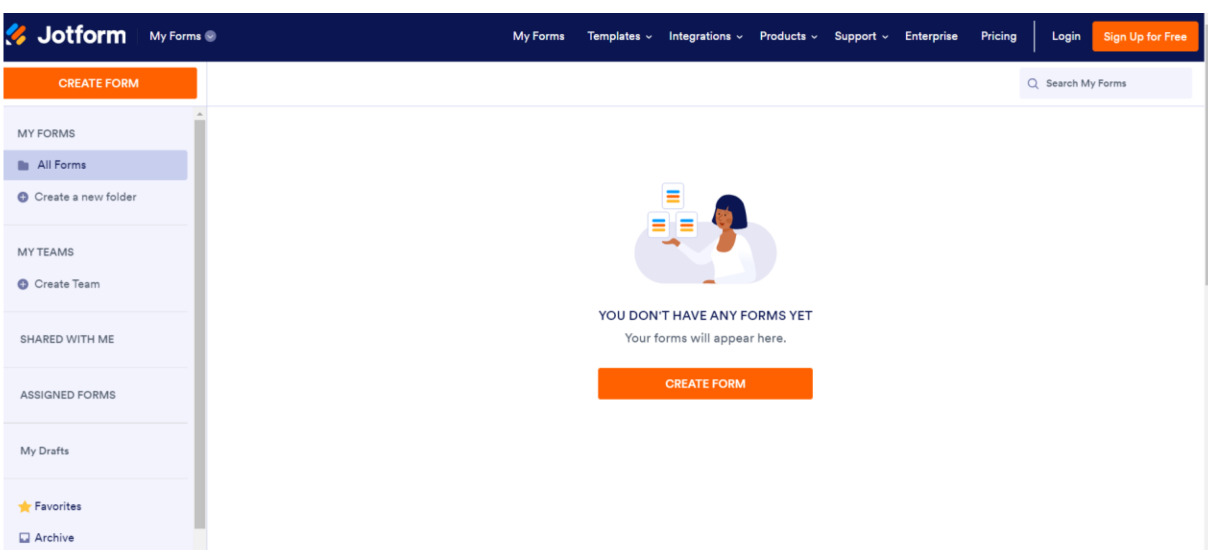
## Milestone 1 :- Create a jotform and integrate it with the org to create a record of customers automatically.

**USE CASE : -** 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.

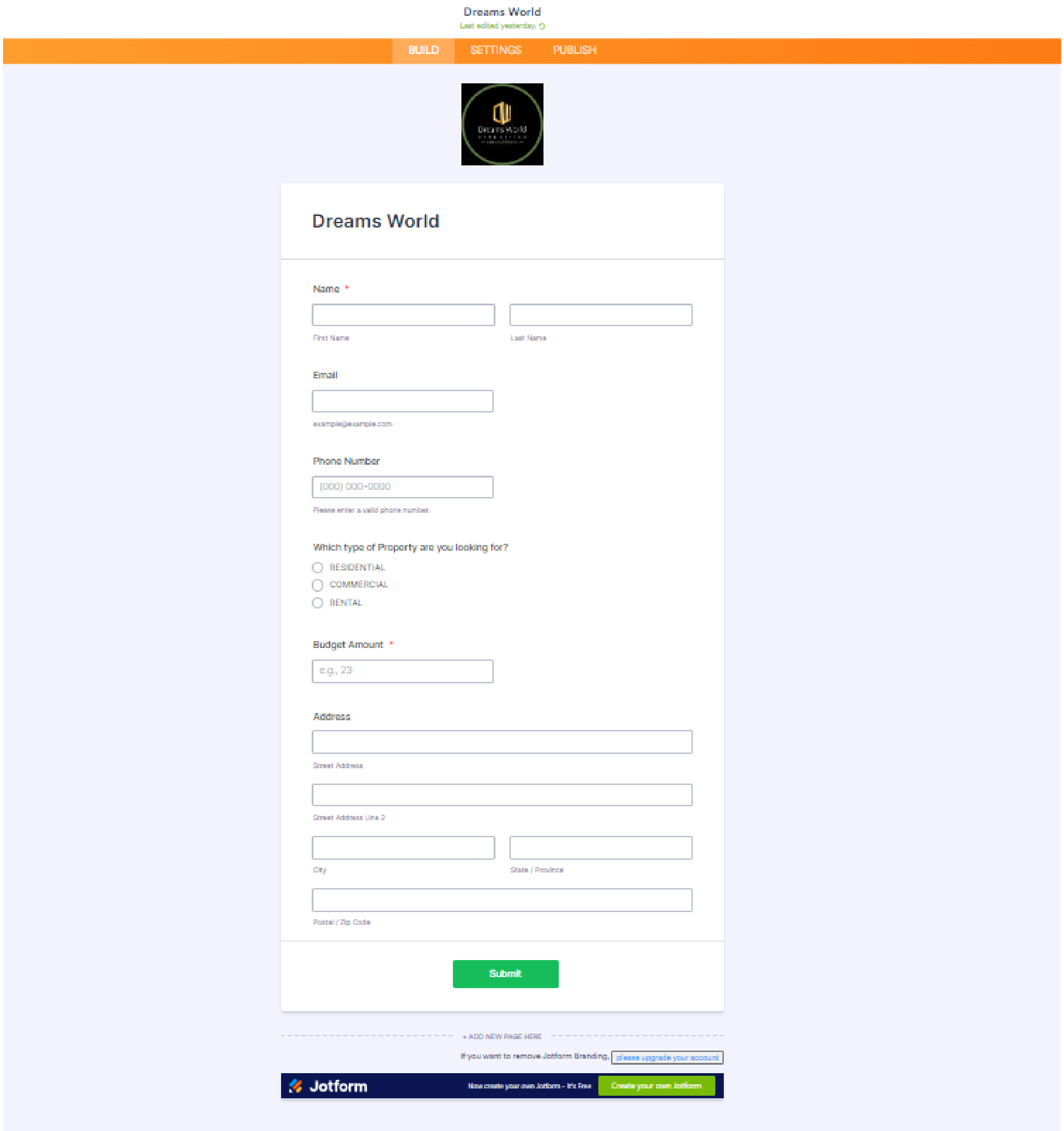
1) Open your browser and search for jotform and log in.




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.

https://www.jotform.com/form/240031134484041

## Milestone 2 :- Create Objects from Spreadsheet.

- **Create 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
    4) After downloading, upload the file, map the fields and upload to create an object.

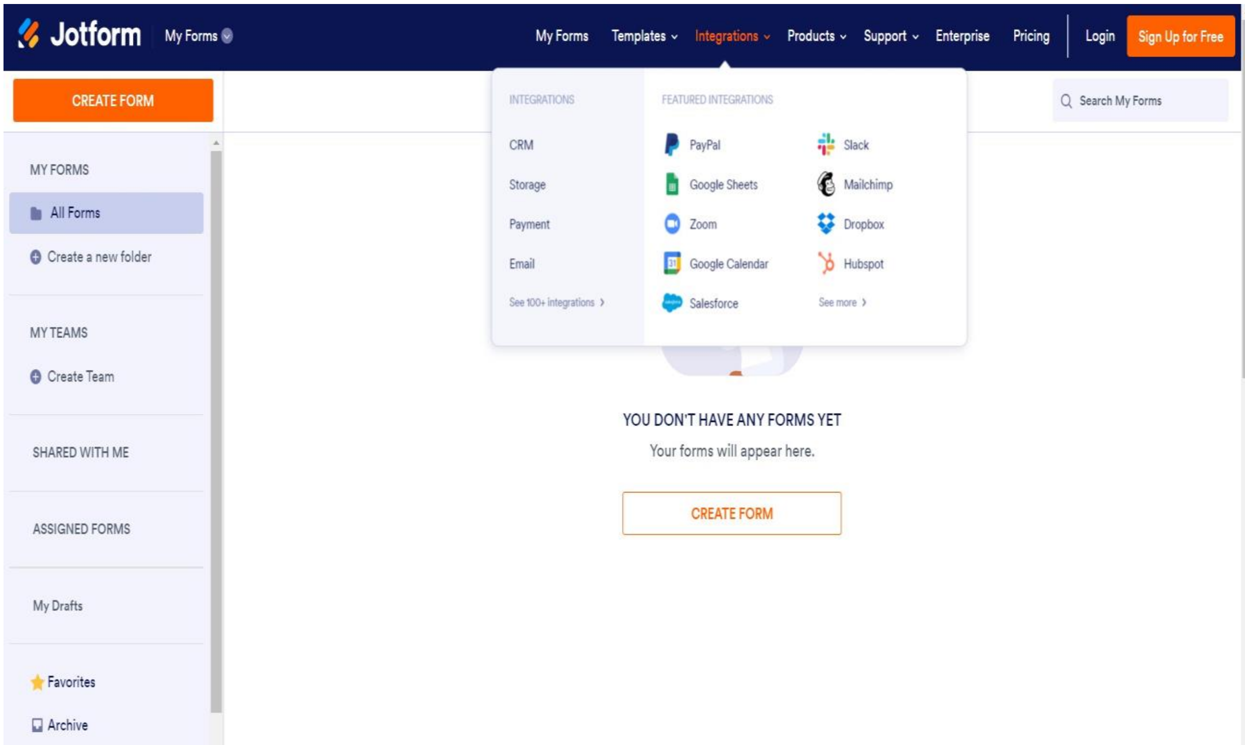
- **Create Property object**

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

## Milestone 3 : - Integrate Jotform with Salesforce Platform

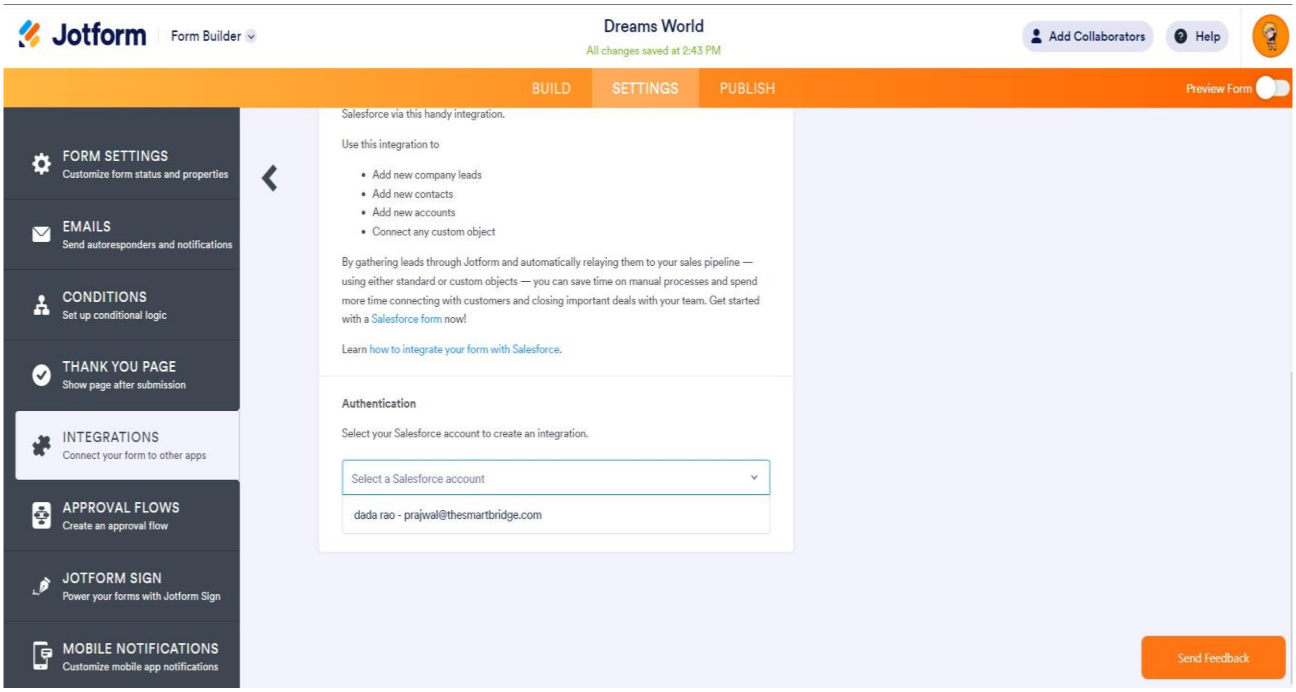
]

1) On the Jotform Platform, Click on Integration and choose Salesforce.

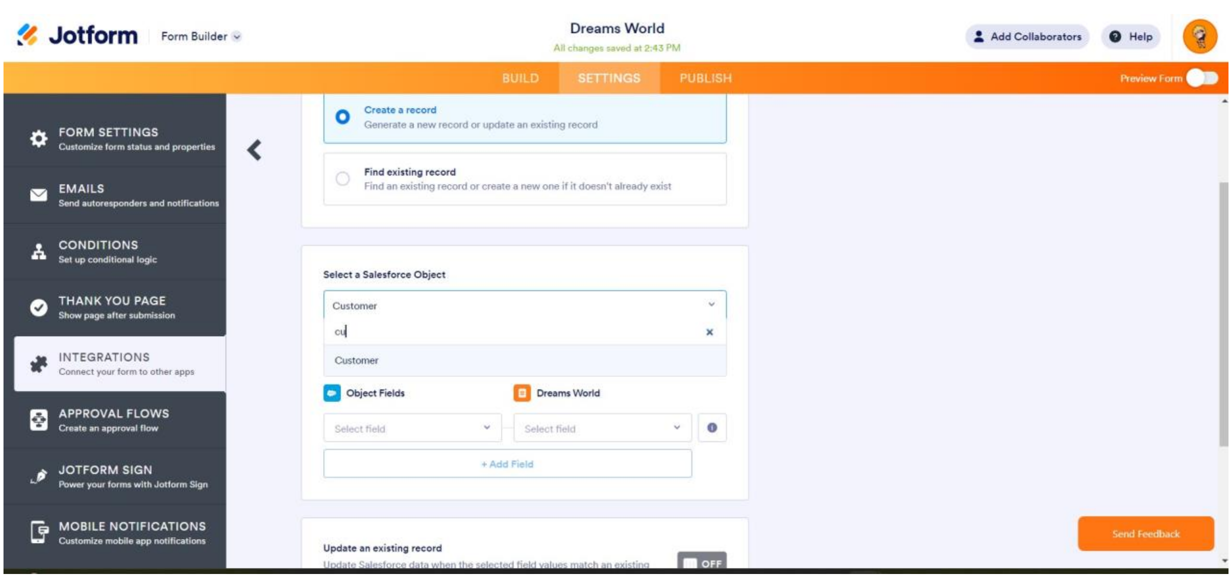



2) Click on User Integration and choose "Add to From".

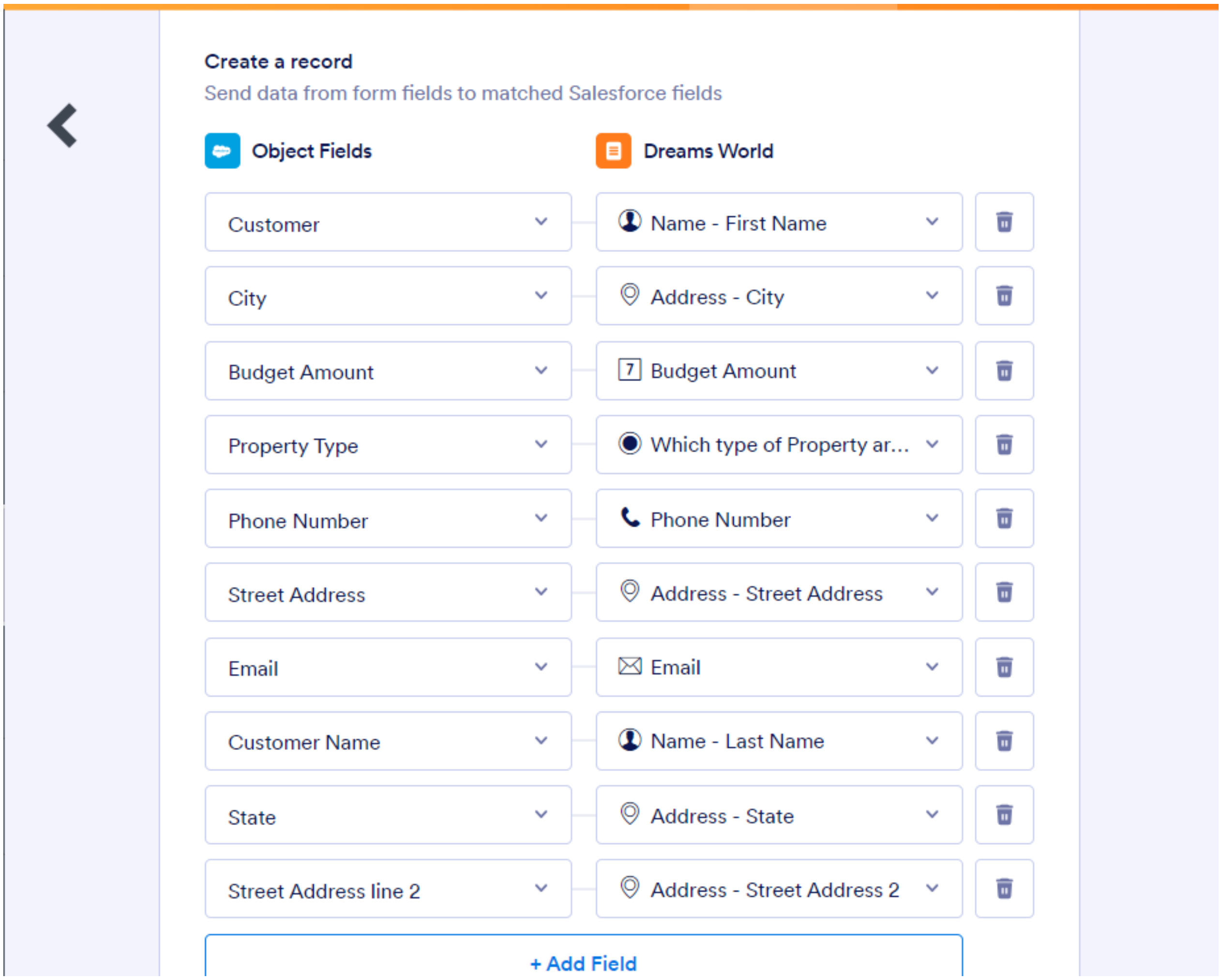
3) Select the Org with which you want to Integrate your jotform with.



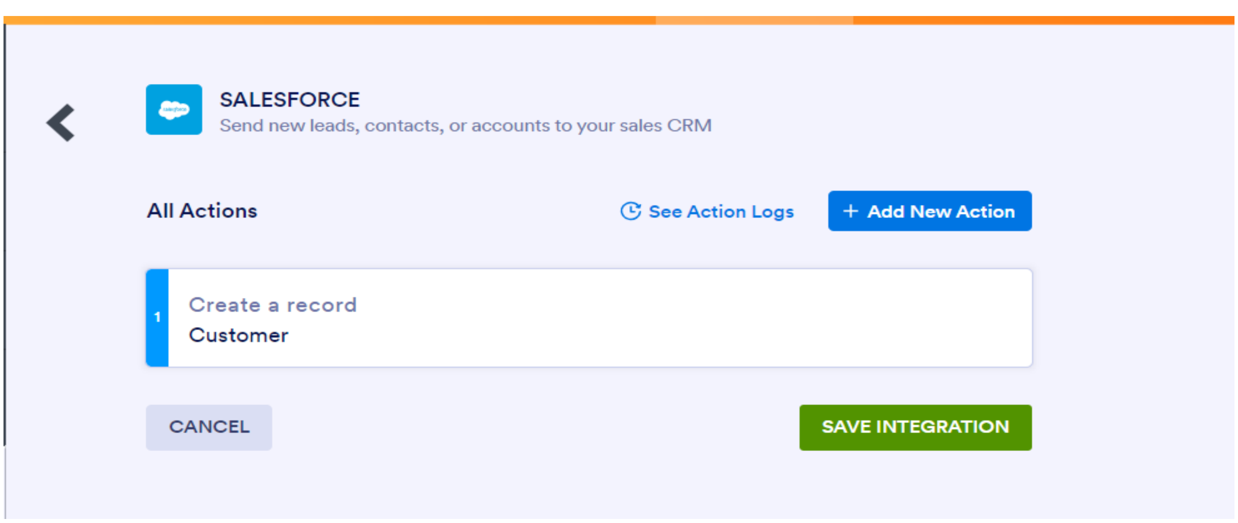

4) Select an Action - Create a record.
Select a Salesforce Object : - Customer



5) Map Each and every field on the Object with the fields on the form and "Save Action".



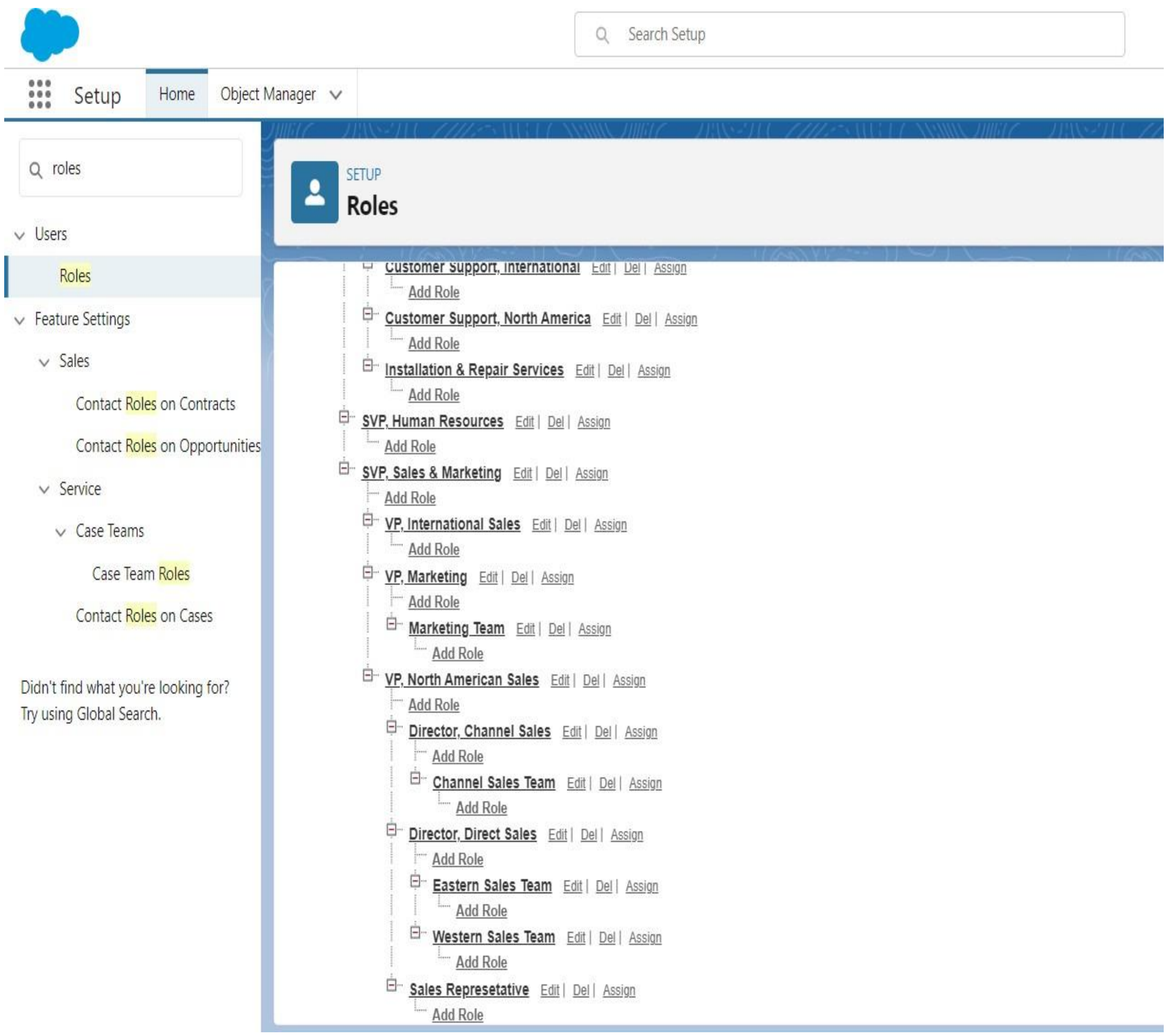

6) Then "Save the Integration" and "Finish".



# Milestone 4 : Create Roles

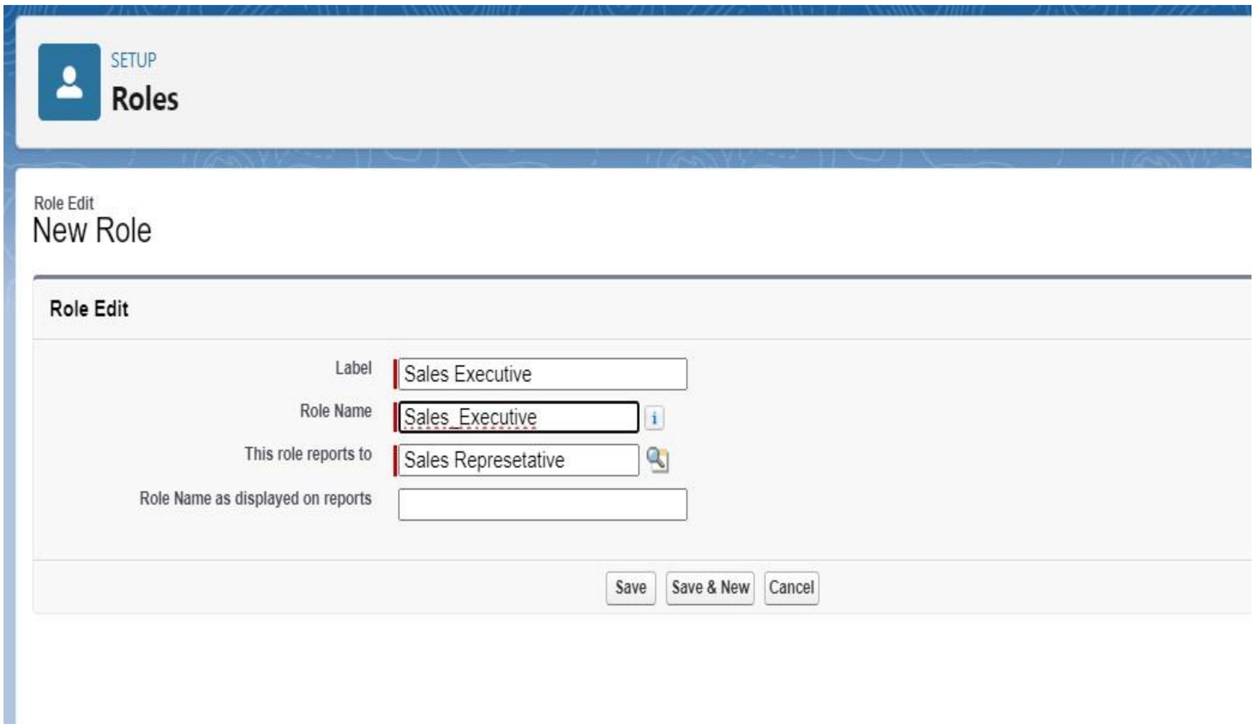
- **Sales Executive Role**

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




* It will use the “System Administrator Profile”.

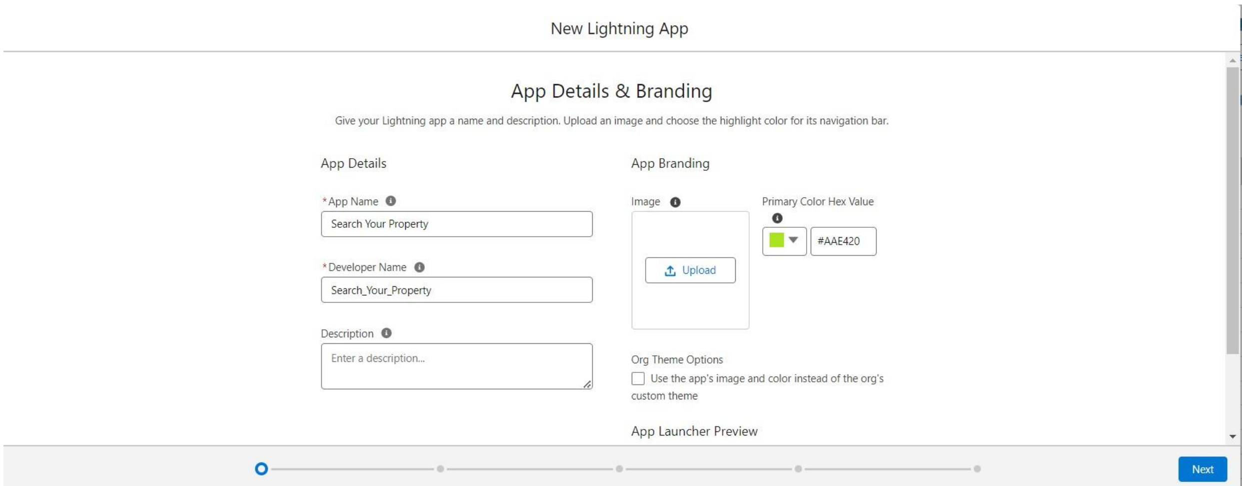
2) Label - Sales Executive
   Reports to - Sales Representative




- *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.*

# Milestone 5 : - Create a Property Details App

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.

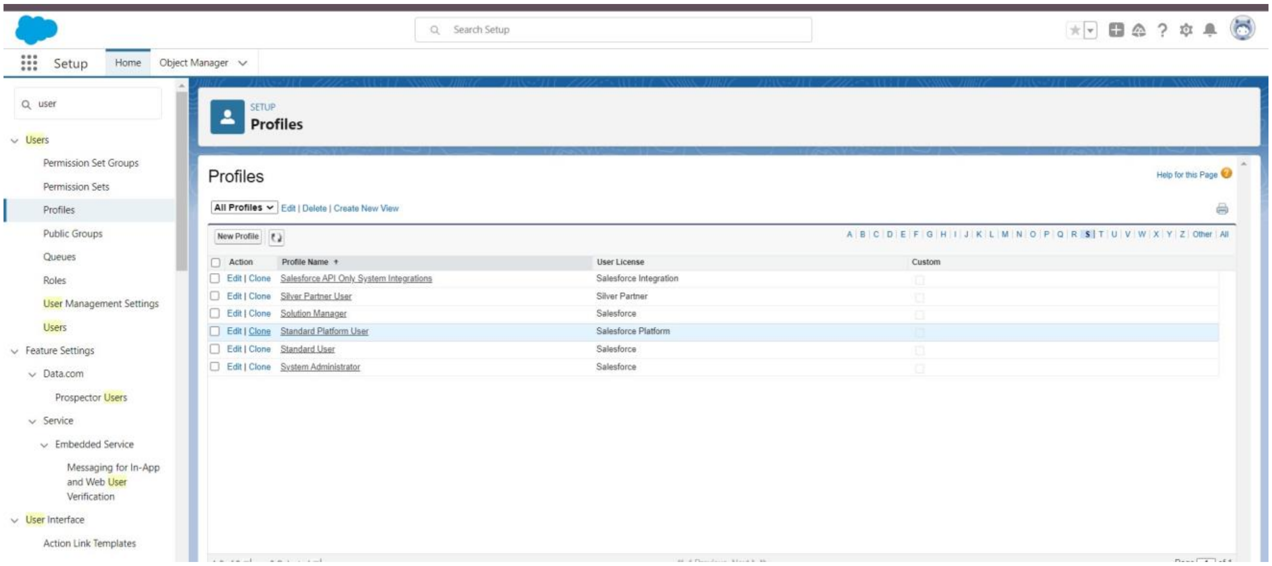



2) Click Next→ Next → Save and Add "System Admin "Profile.

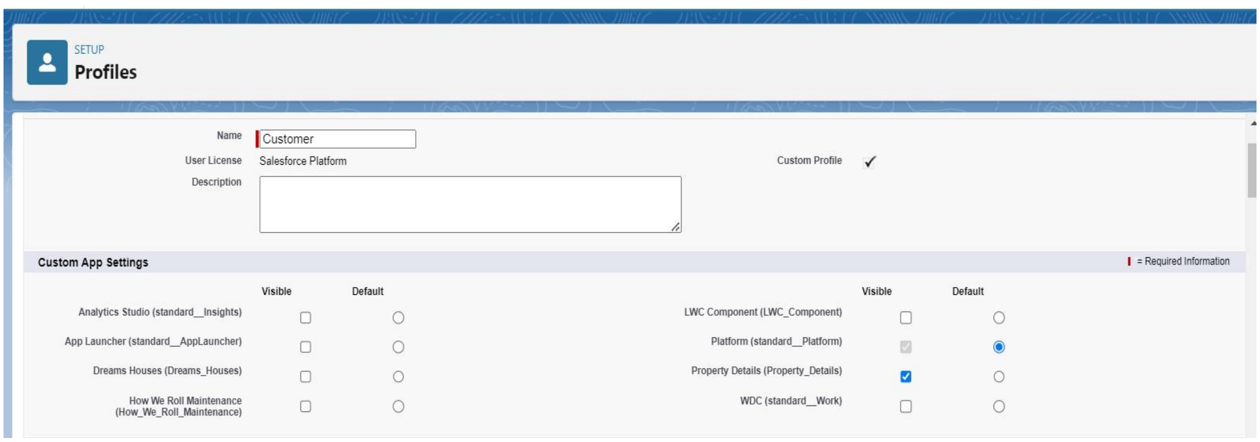
# Milestone 6 : - Create Profiles

- **Customer : -**

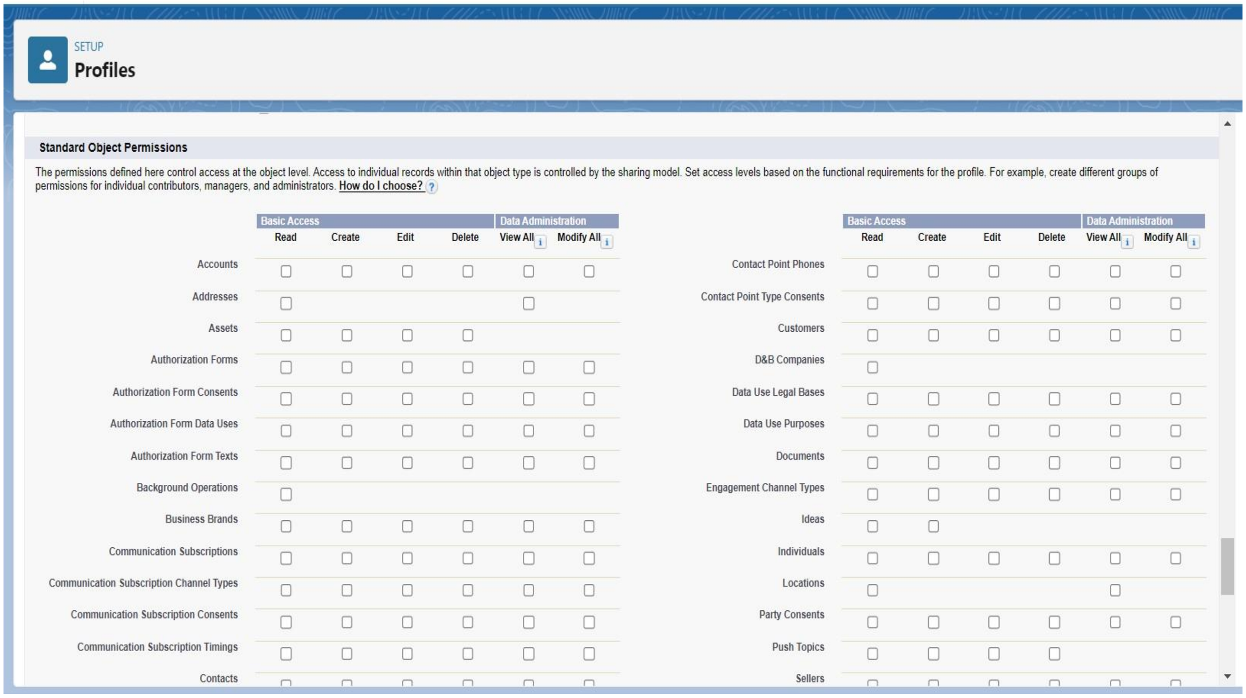
1) From Setup→ Go to Profiles and Clone Salesforce Platform User and Name it "Customer"..



2) Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.



3) Also Remove all the Standard Object Permissions.

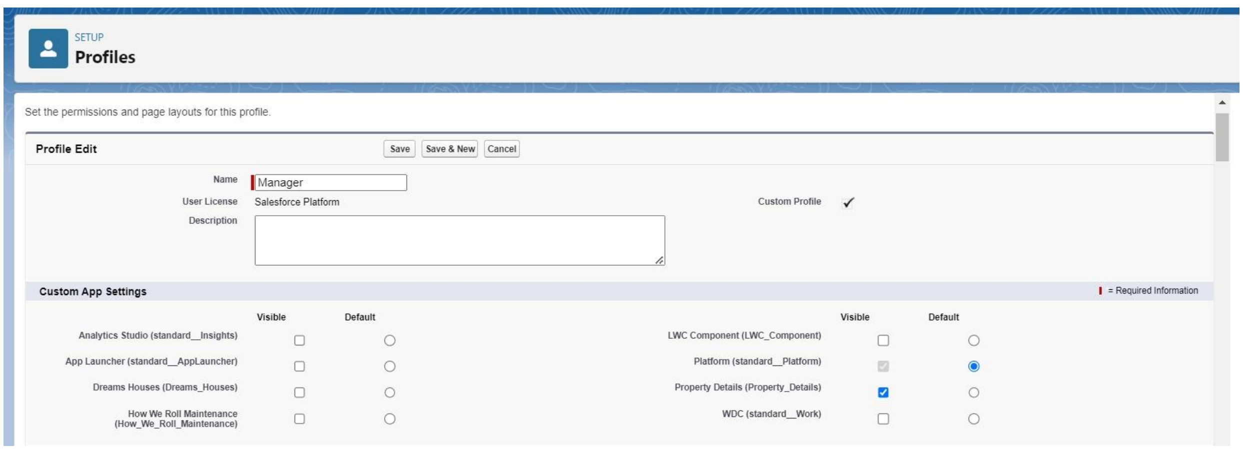



4) Uncheck all the Custom Object Permissions and check read and view all in "Property"

- **Manager : -**

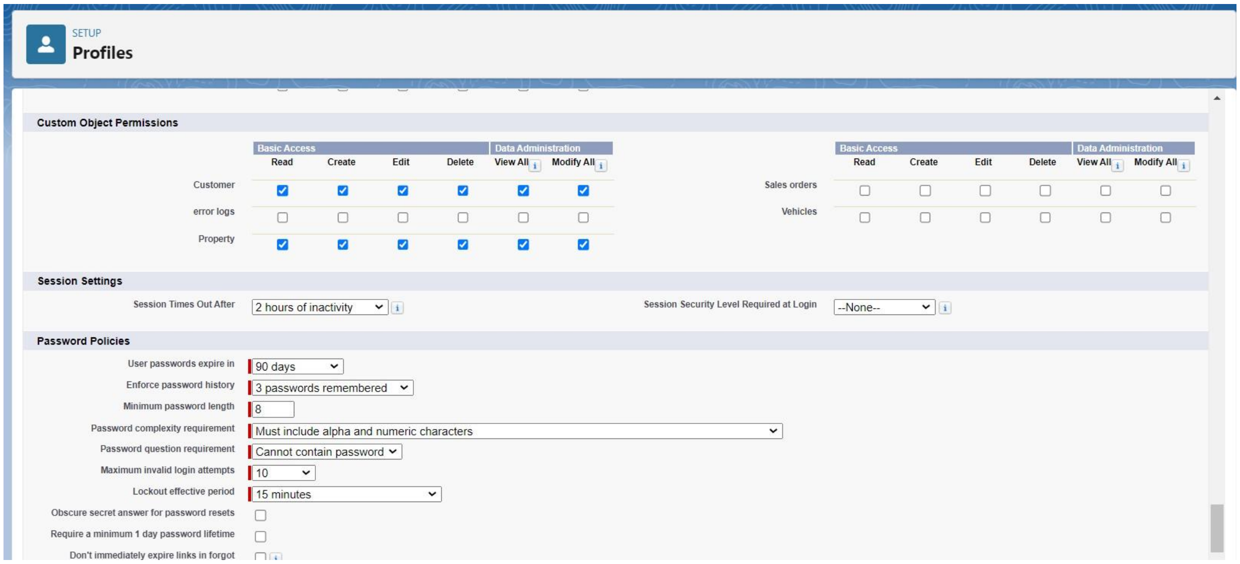
1) From Setup→ Go to Profiles and Clone Salesforce Platform User and Name it “Manager”..

2) Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.




3) Also Remove all the Standard Object Permissions.

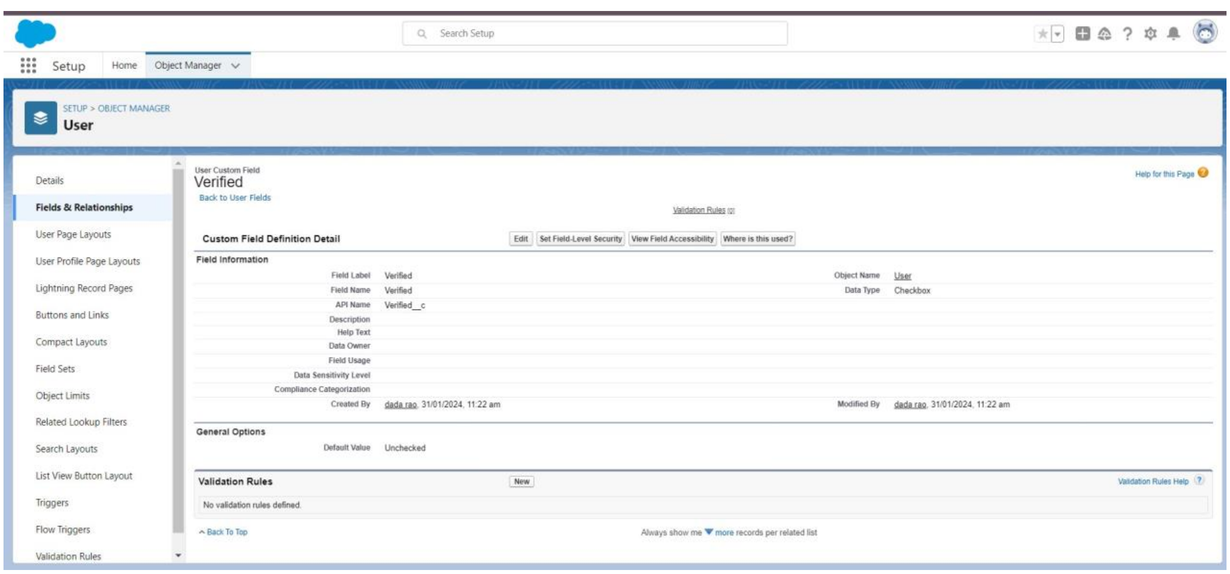
4) Uncheck all the Custom Object Permissions and check only “modify all” from “Property” and “Customer”




# Milestone 7 : - Create a CheckBox field on user

1) Setup → Object Manager → Search for User → Fields and Relationships

2) Create new Field Named as “Verified” as Data type “CheckBox”



## Milestone 8 : - Create Users

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

### User 1 : -

1) Go to Setup → Administration → Users → New User
2) LastName - 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) LastName - 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) LastName - 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) LastName - Customer2

3) Role - Customer

4) License - Salesforce Platform

5) Profile - Customer

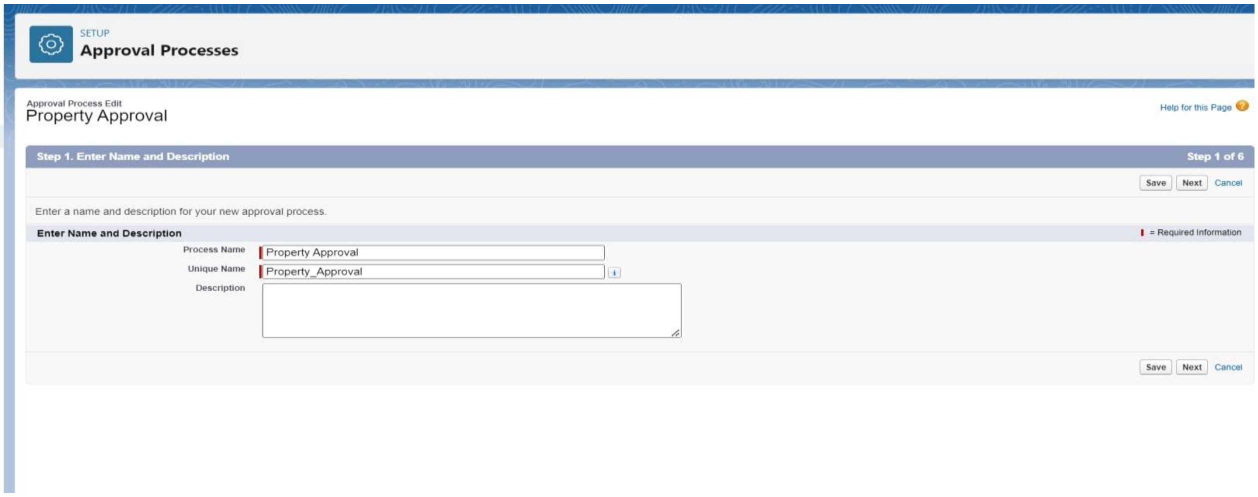
6) Make Sure the verified check box is “checked”

7) Save

## Milestone 9 :- Create an Approval Process for Property Object

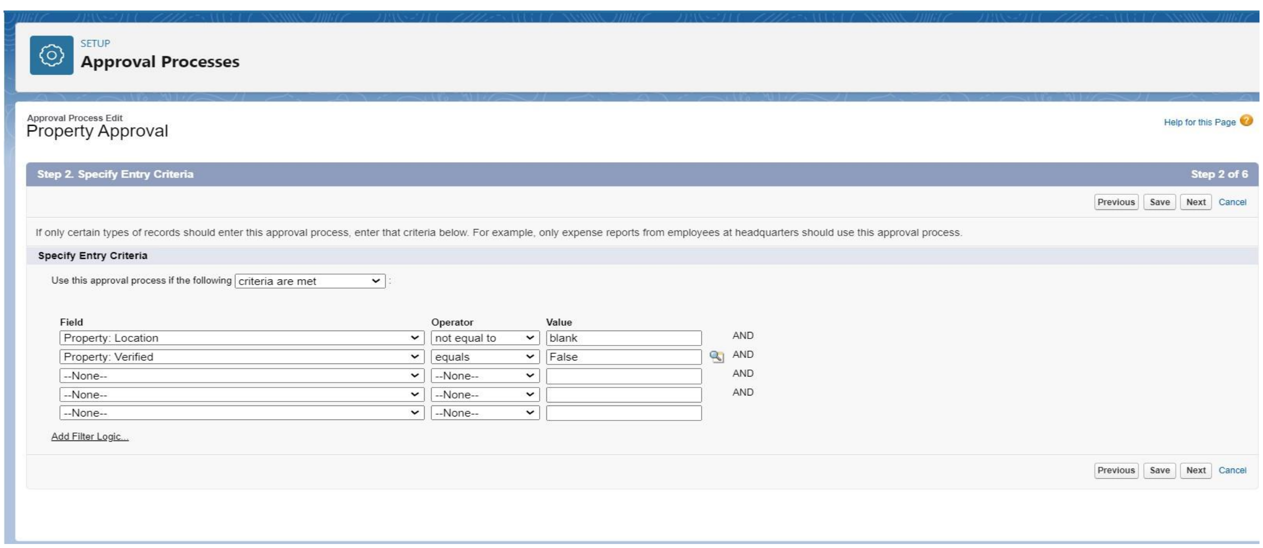
1) From Setup → Process Automation → Approval Process

2) Process Name - Property Approval



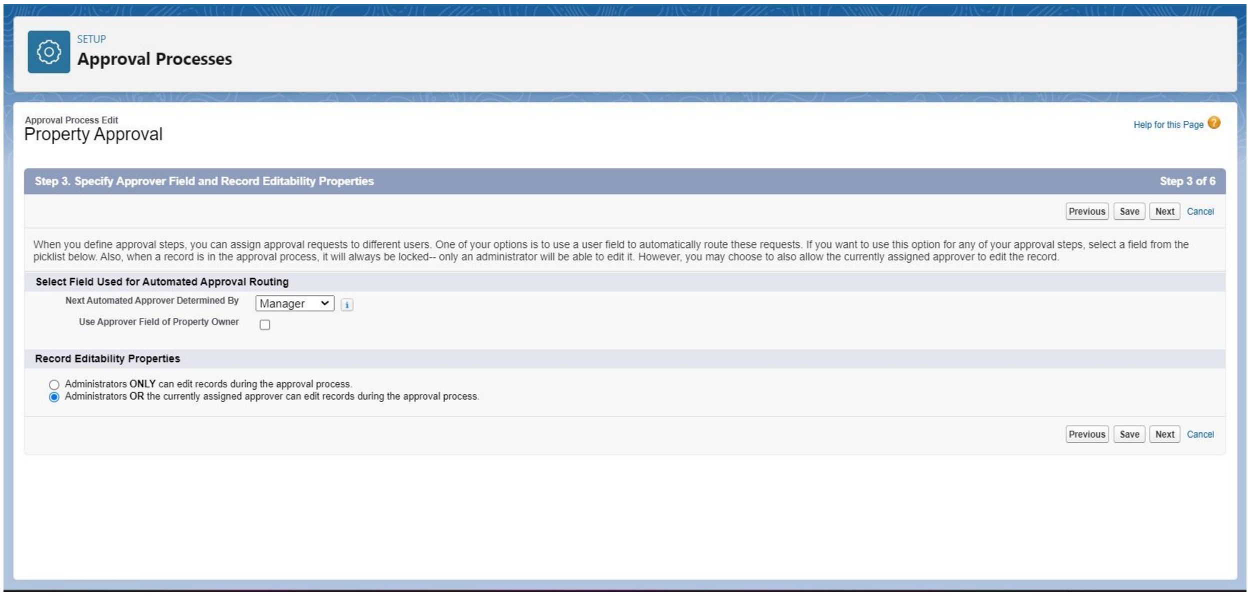

3) Give 2 criteria →

a) Location is not equal to blank,
b) Verified Equals false.

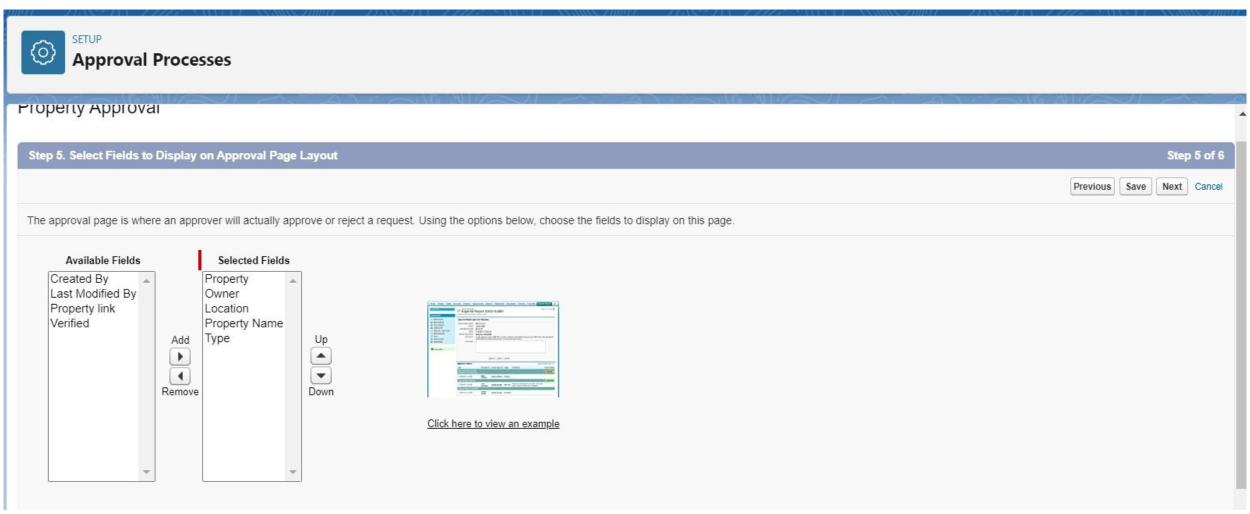



**4)** Click next and "**Next Automated Approver Determined By**" → **Select Manager**

5) From Record Editability Properties → Click on Administrators **OR** the currently assigned approver can edit records during the approval process.




6) From Step 5. Select Fields to Display on Approval Page Layout select Property, Owner, Location, Type.

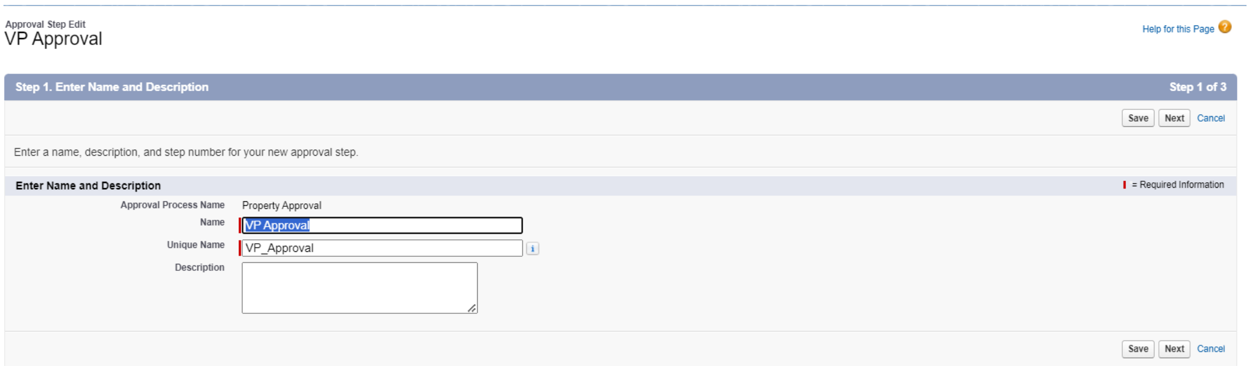



7) Click Next and Select the initial Submiters →

a ) Owner → Property Owner
b) Roles → Sales Manager

8) Save.

9) Add an approval step name “Executive Approval ”

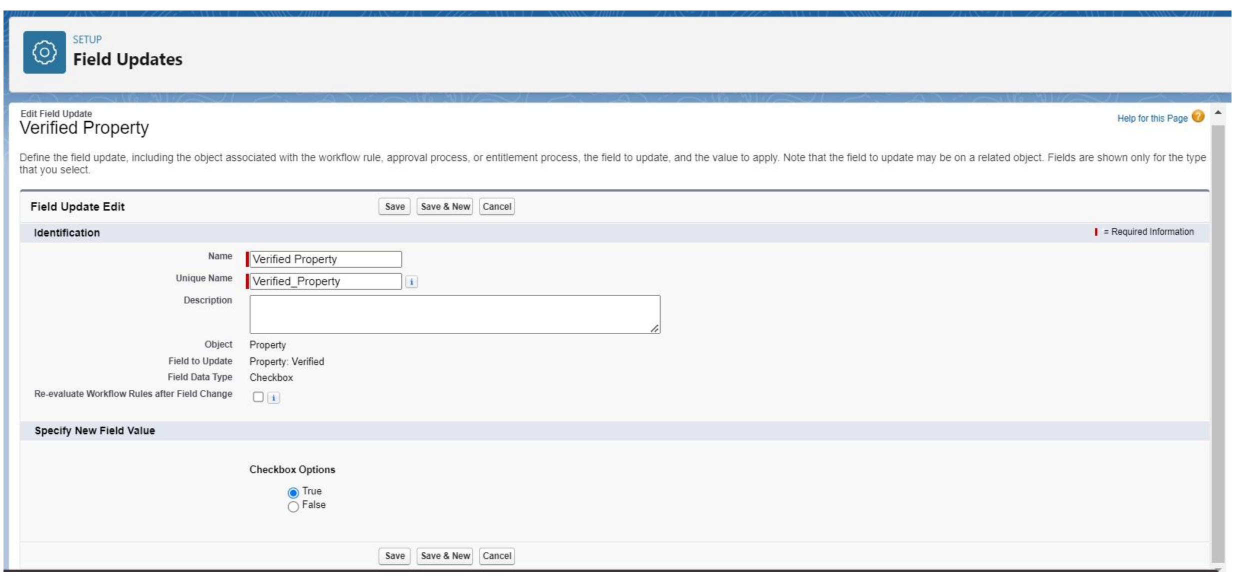



10) specify the Criteria → All record should enter


11) click next and select the Approver as “ Sales Executive “ and “Save”

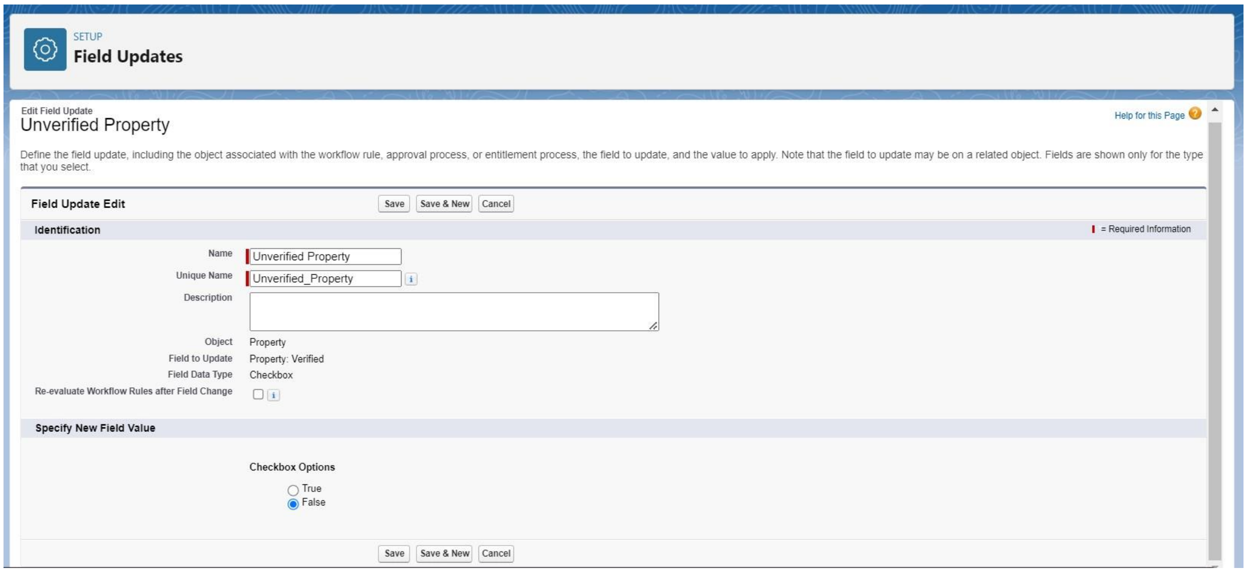

12) Add One field Update as “Verified Property”

a) Select Object → Property
b) Field to Update → Verified
c) Field Data Type → CheckBox
d) Select CheckBox Option as “True”
e) Save.



13) Add One field Update  as “UnVerified Property”

a) Select Object → Property
b) Field to Update → Verified
c) Field Data Type → CheckBox
d) Select CheckBox Option as “False”
e) Save.




14) Activate the Approval Process.

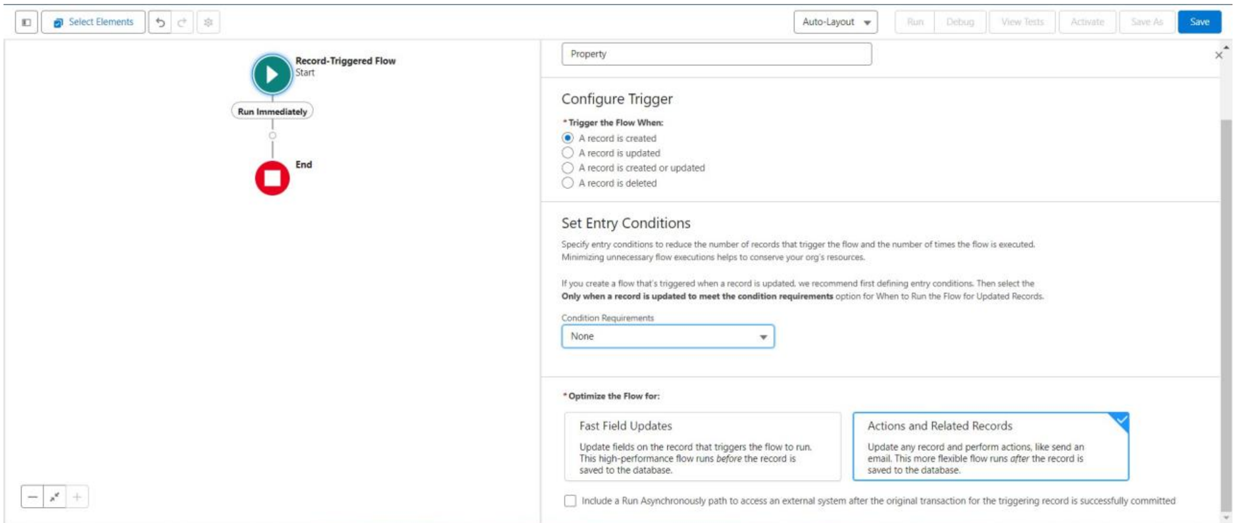
## Milestone 10 : - Create a Record trigger flow to submit the Approval Process Automatically.

1) From Setup → Search for Flows → Click On New and Select “Record Trigger Flow”.

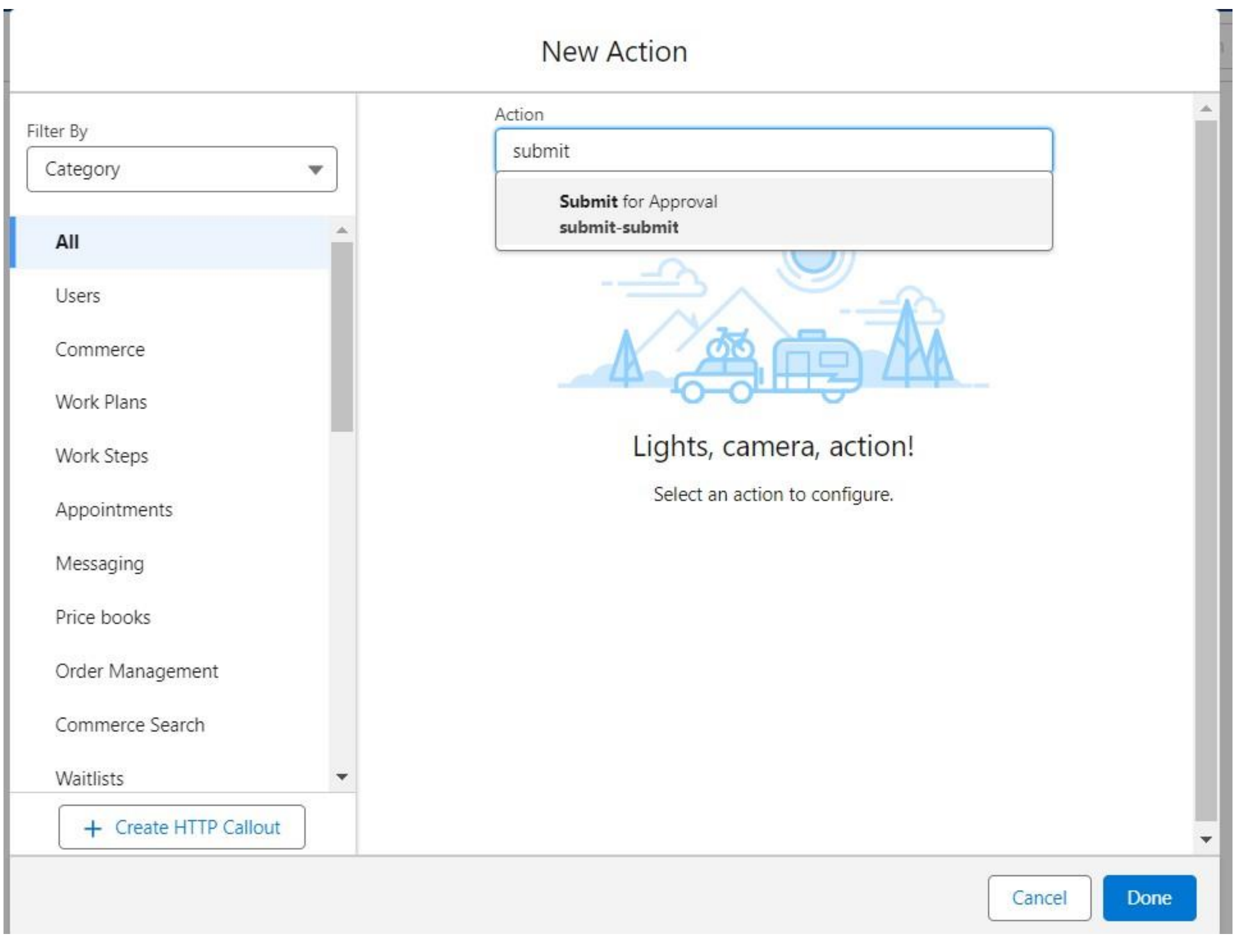
2) Select Object → Property

3) Select “Trigger the flow when” → “A record is created”

4) Set Entry Conditions → “None”



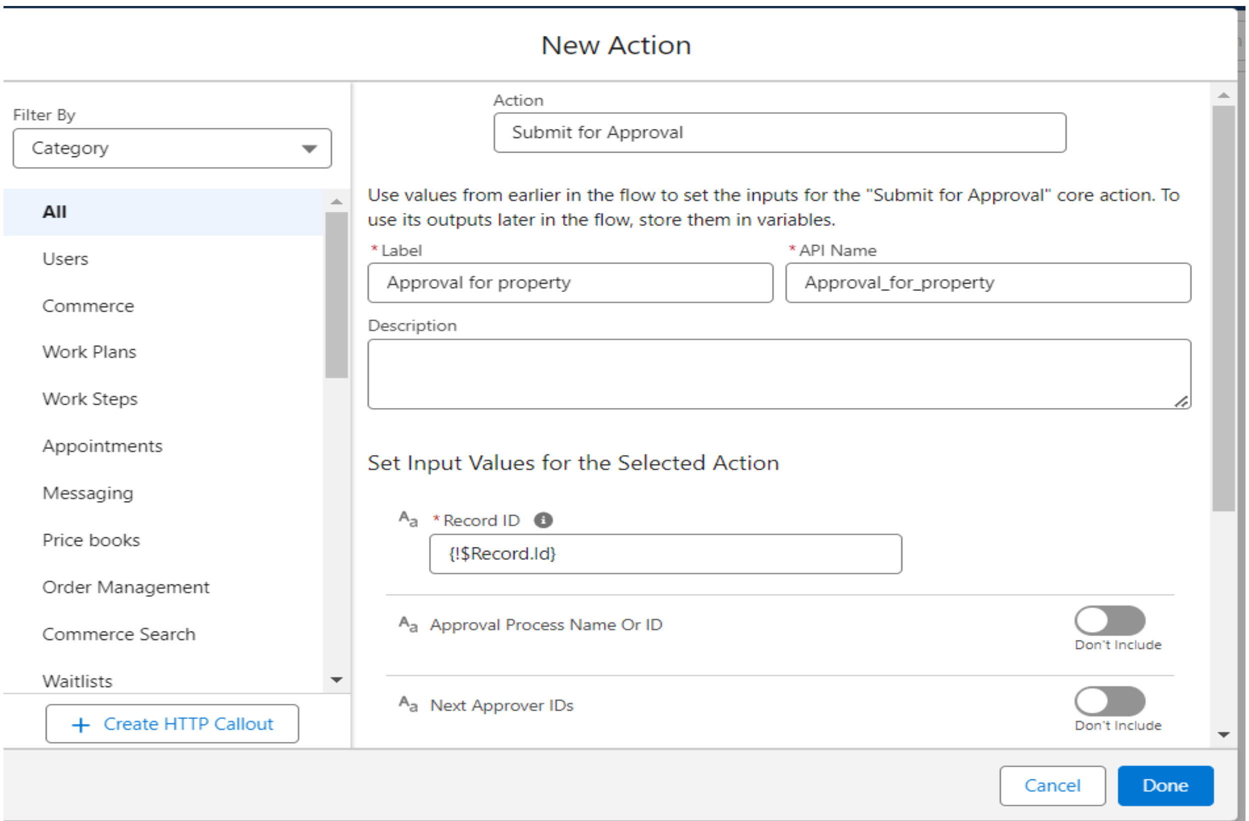

5) Add a "Action" → "Submit for Approval"



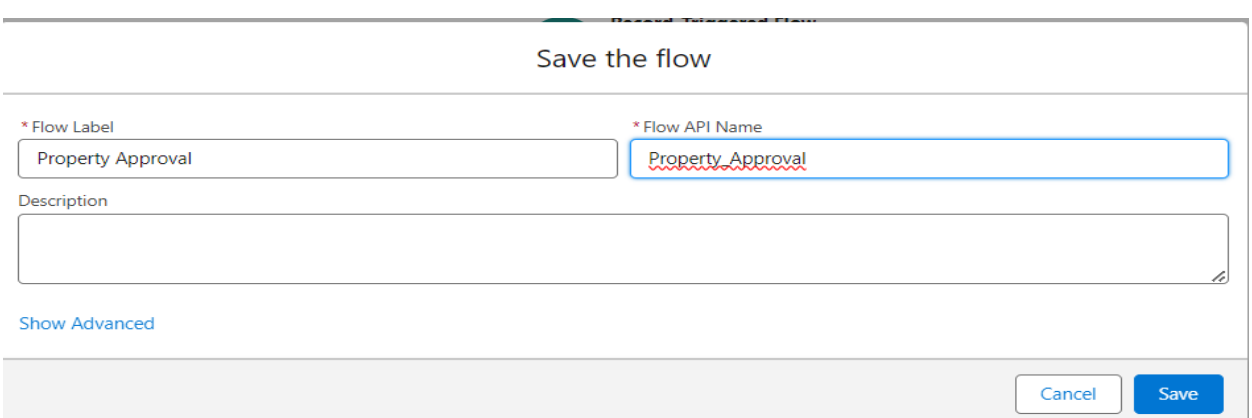
6) Give Label → Approval for property

7) Record Id → {!$Record.Id}

8) Done



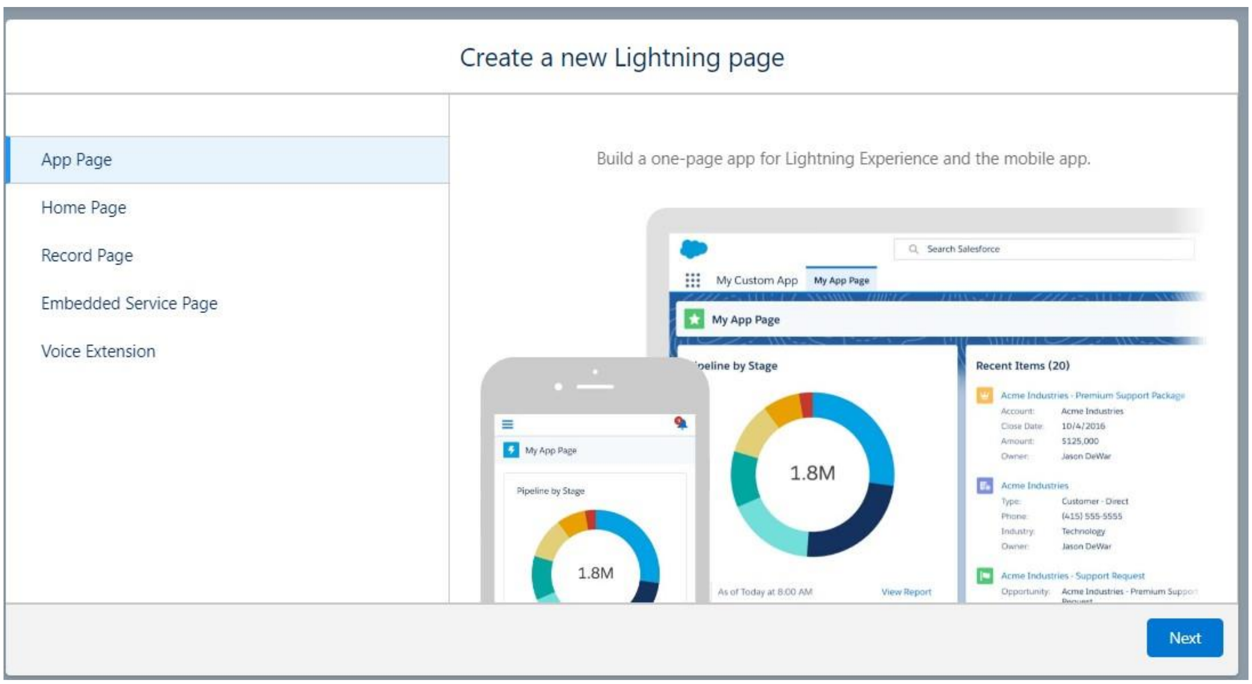

9) Save the Flow and Give label as → "Property Approval" and "Activate"



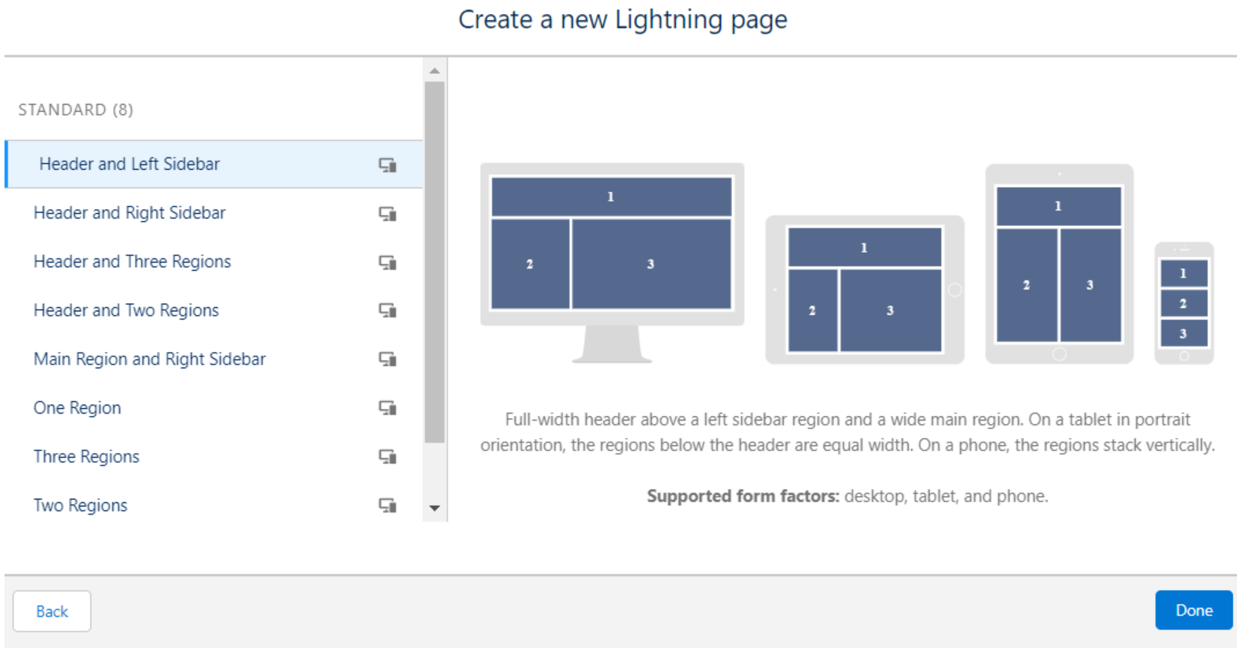
## Milestone 11 :- Create an App Page

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

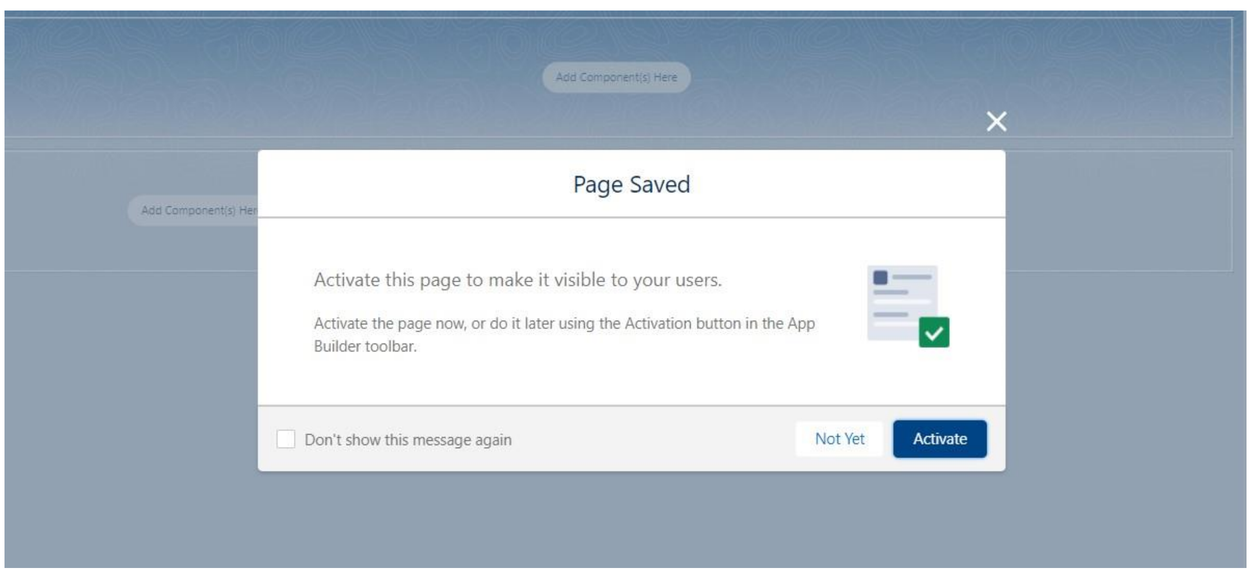
1) From Setup → Go to Lightning App Builder →  Click on New → Select App Page and Click on Next.



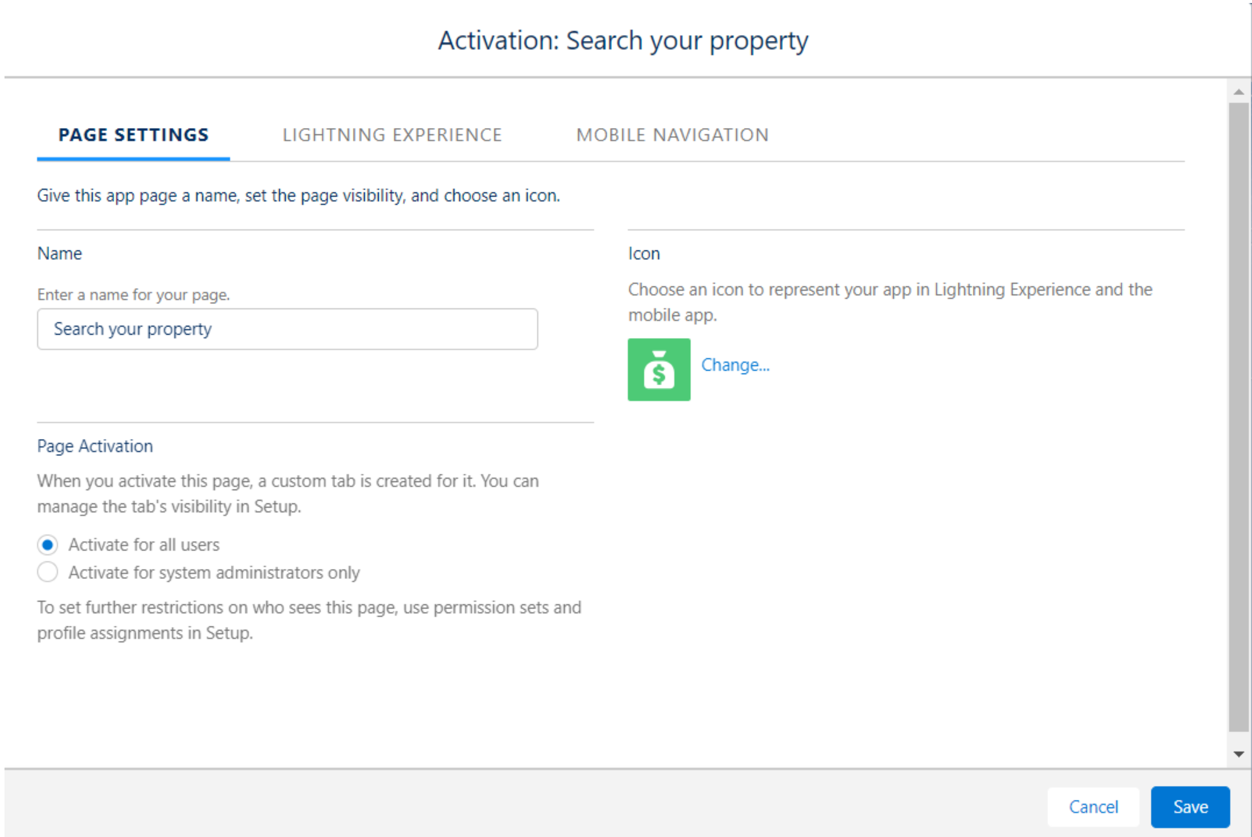

2) Give Label as “Search your Property” click “Next”.
3) Click “header and Left Sidebar”  and Click on “Done”



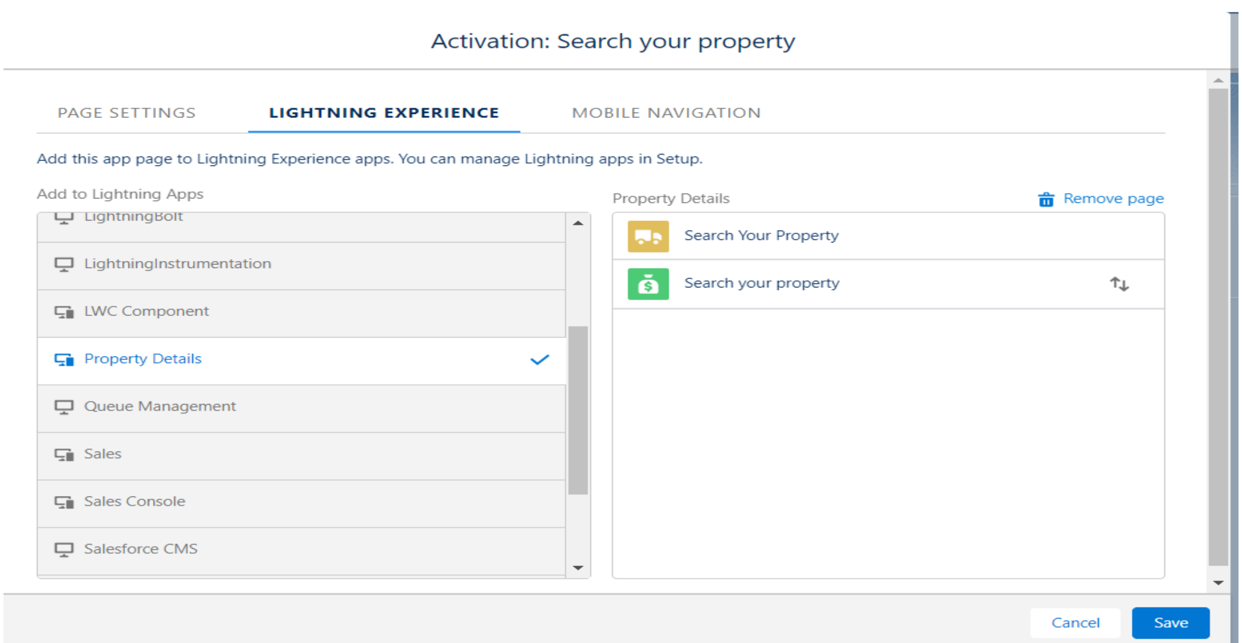

4) Click on "Save " and then click on "Activate".




5) From Page Setting select page activation as "Activate for all Users".




6) From Lightning Experience Click on "Property Details" and click on Add Page".




7) Then Click on "Save"

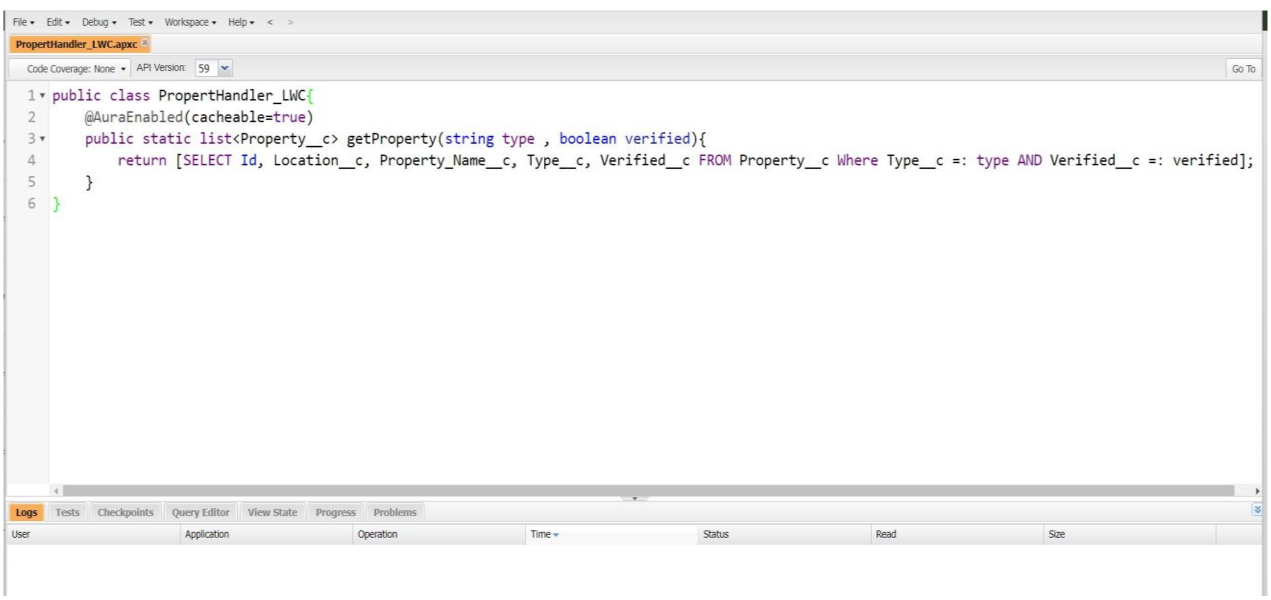
# Milestone 12 :- Create a LWC Component

- 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”

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

**<u>Code: -</u>**

```
public class PropertHandler_LWC{
   @AuraEnabled(cacheable=true)
   public static list<Property__c> getProperty(string type , boolean verified){
      return [SELECT Id, Location_c, Property_Name_c, Type_c, Verified_c FROM
Property_c Where Type_c =: type AND Verified_c =: verified];

   }
}
```




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

3) Enter your login id and password to authorize your org.

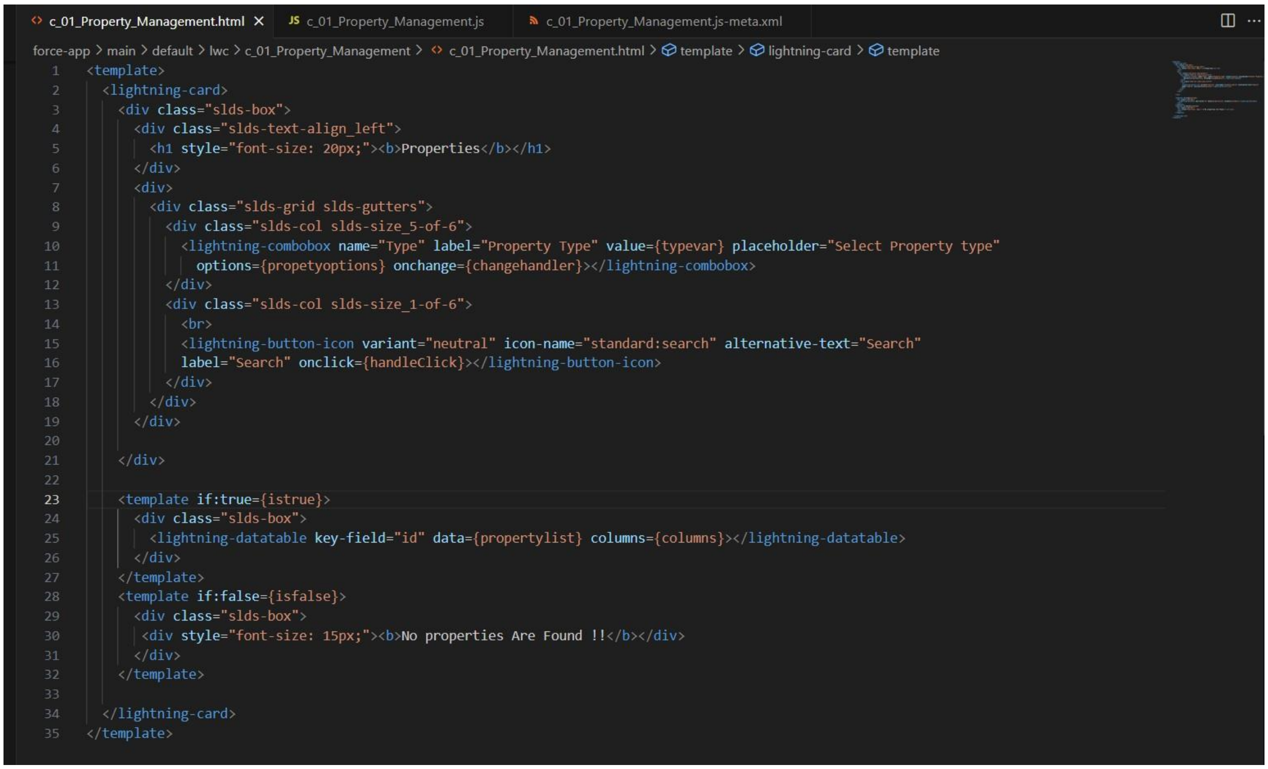
4) Now (ctrl+shift +P) → Create a lightning Web Component and Name it Anything you want to. (Example - )

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={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>
```



```
<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>
```

6) 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_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)
        })
    }


}
```

c_01_Property_Management.html    c_01_Property_Management.js ×    c_01_Property_Management.js-meta.xml

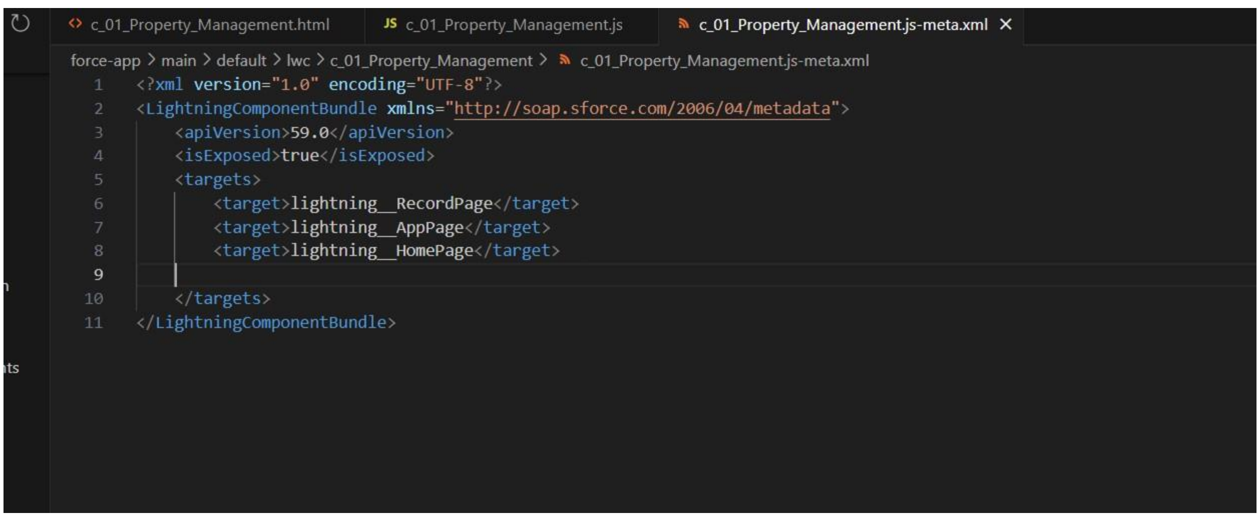
force-app > main > default > lwc > c_01_Property_Management > c_01_Property_Management.js > C_01_Property_Management > propetyoptions

```javascript
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_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 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>
```

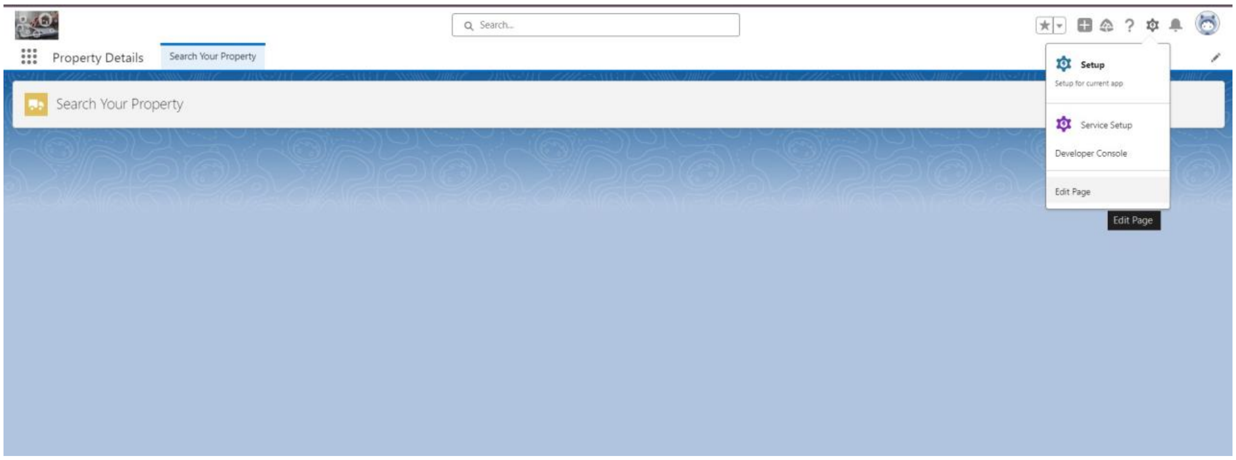



8) After Saving all the three Codes , Right Click and deploy this component to the org.

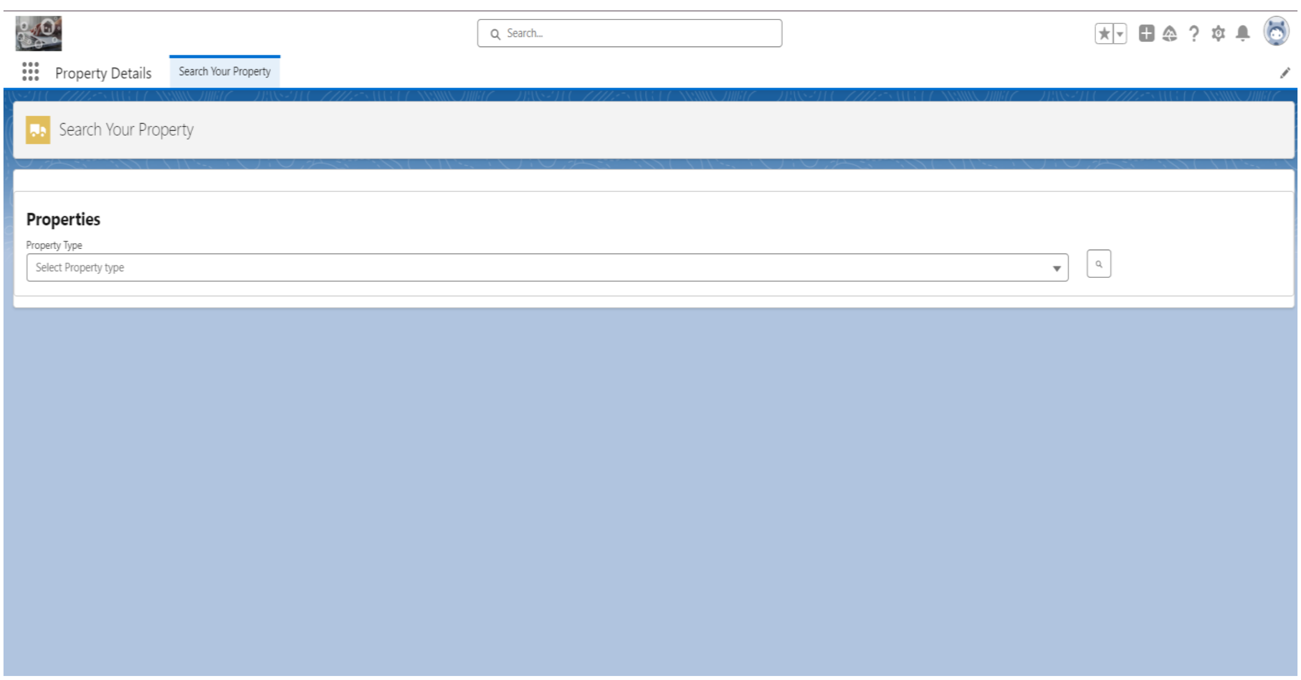
# Milestone 13 : - Drag this Component to your App Page

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) Drag theComponent to your App Page and Save the Page.



## Milestone 14 : - Give Access of Apex Classes to Profiles

1) From Setup → Search For Apex Classes → Click on “Security” behind “PropertyHandler_LWC”.

2) From Profiles Add “Manager” and “Customer” and “Save”.

—-------------------------------------------------------*--------------------------------------------------------

**SUBMITTED BY**

au711221104056
au711221104003
au711221104052
au711221104016