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

College Name: Arunai Engineering College

College code: 5104

Department: Computer Science And Engineering

Team Leader:

Name: Rasiga Priya A G Register No: 510422104082

Team Members:

Name: Monica G Register No: 510422104064

Name: Nivetha B Register No: 510422104071

Name: Salma Bee A Register No: 510422104088

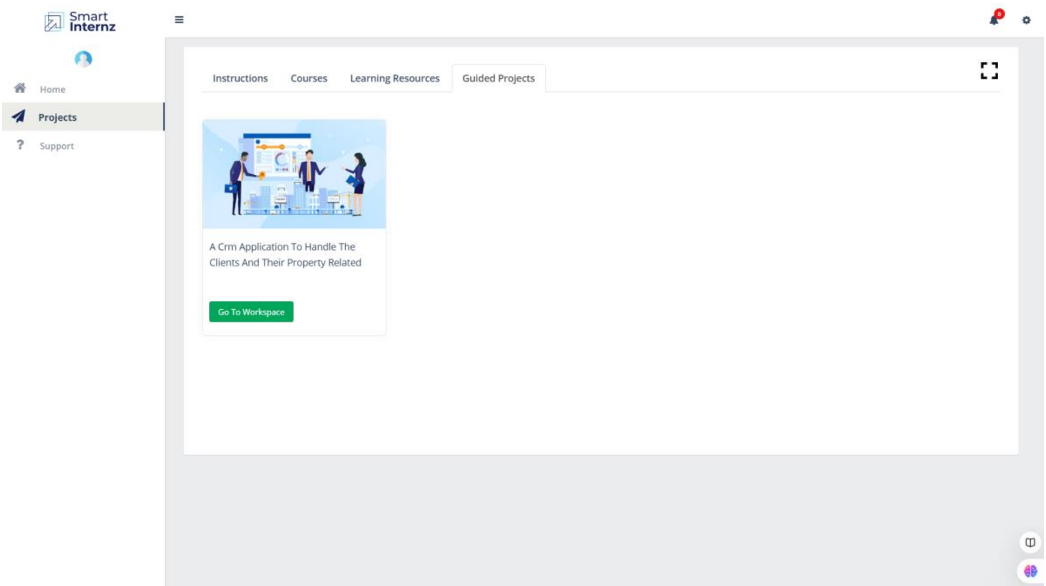
GitHub link: https://github.com/Rasiga-priya/NM.git

Registered Email id: rasigapriya771@gmail.com

A CRM APPLICATION TO HANDLE THE CLIENTS AND THEIR PROPERTY RELATED REQUIREMENTS

1. PROJECT OVERVIEW

This project is focused on a CRM application to handle the clients and their property related requirements, designed to address the challenge of managing customer interactions efficiently and delivering personalized property recommendations based on user preferences. The goal is to deliver a comprehensive solution by leveraging Salesforce to automatically create and categorize customer records based on website engagement, capturing essential details and preferences. Users are categorized as either approved or non-approved, allowing approved users access to tailored property recommendations and broader listings. Through this project, we aim to enhance This integration enhances user experience and operational efficiency, providing personalized property recommendations and optimizing customer engagement and support the long-term goals of Dreams World Properties by facilitating growth in the real estate market and enhancing customer satisfaction.



2. OBJECTIVES

The specific, measurable goals the project intends to achieve.

Business Goals:

- Streamline customer relationship management by automating record creation and categorization.
- Enhance customer engagement and satisfaction through tailored property recommendations.
- Support business growth in the real estate market by optimizing operational efficiency.

Specific Outcomes:

- Automated creation and categorization of customer records in Salesforce based on website interactions.
- Delivery of personalized property selections to approved users, enhancing user experience.
- Improved data management, resulting in better insights into customer preferences and behaviours.

3. SALESFORCE KEY FEATURES AND CONCEPTS UTILIZED

The main functionalities and concepts applied within the Salesforce project:

1. **Automated Record Creation**: Integrated with JotForm to capture customer details from website interactions and automatically create records in Salesforce.
2. **User Categorization**: Salesforce categorizes users as approved or non-approved, enabling targeted property recommendations based on user status.
3. **Data Management and Segmentation**: Organizes and segments customer data efficiently within Salesforce, allowing easy access to user information for personalized interactions.
4. **Customized Recommendations**: Uses customer data to provide tailored property suggestions to approved users, enhancing engagement and relevance.

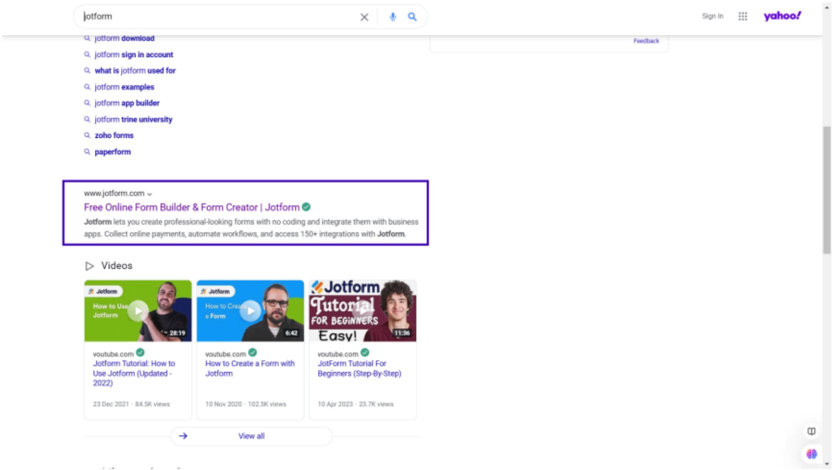
5. **Integration and Automation**: Seamlessly integrates JotForm with Salesforce, creating an automated workflow that reduces manual data entry and optimizes operations.
6. **Real-Time Customer Insights**: Delivers insights into customer preferences and behaviors, supporting strategic decision-making.
7. **Deployment via Visual Studio Code (VS Code)**: The project was deployed using VS Code, enabling streamlined development, version control, and deployment processes.
8. **Apex Programming**: Utilized Apex to implement custom business logic, enhancing automation, data handling, and tailored functionalities within Salesforce.
9. **Lightning Web Components (LWC)**: Built interactive and responsive UI components using LWC, improving user experience and enabling a more dynamic interface.

4. DETAILED STEPS TO SOLUTION DESIGN

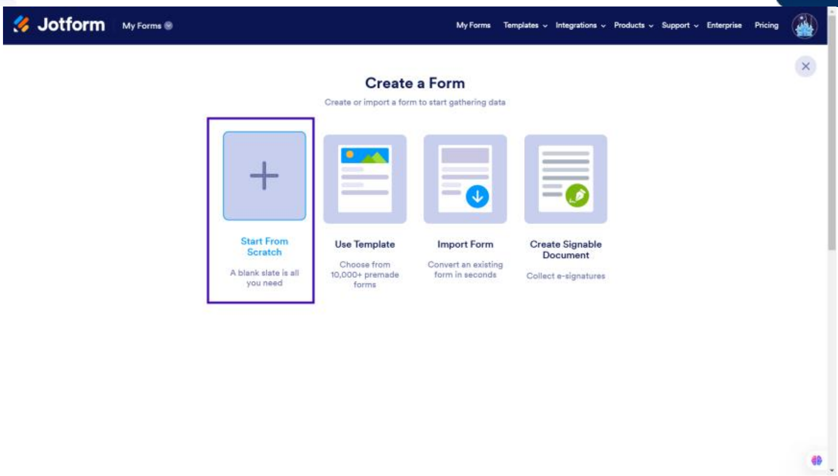
Milestone 1 :- Create a Jotform and integrate it with the org to create a record of customers automatically.

Activity 1:

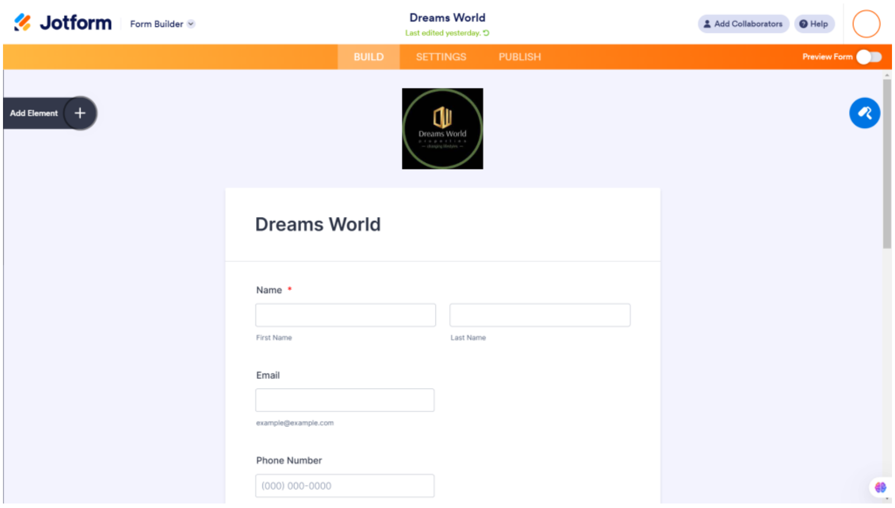
1. Open your browser, search for JotForm, go to the official website, and log in.

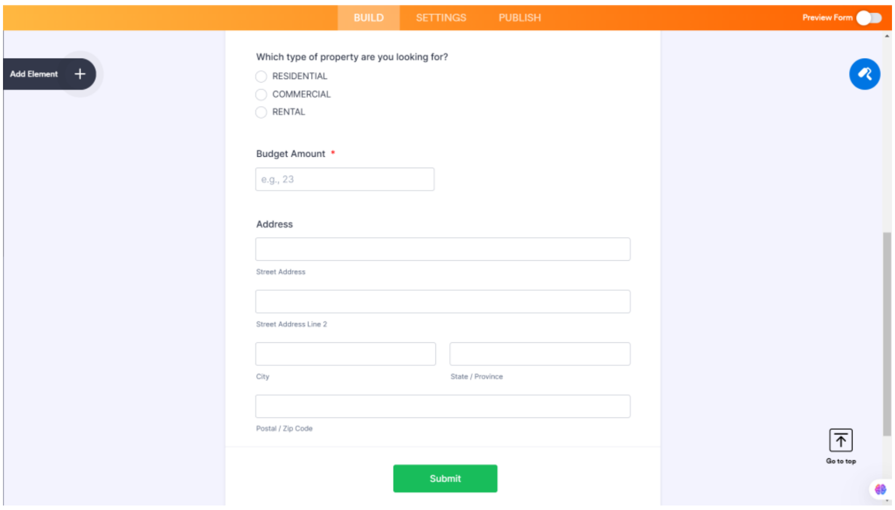


2. After login click on create form and click on start from scratch

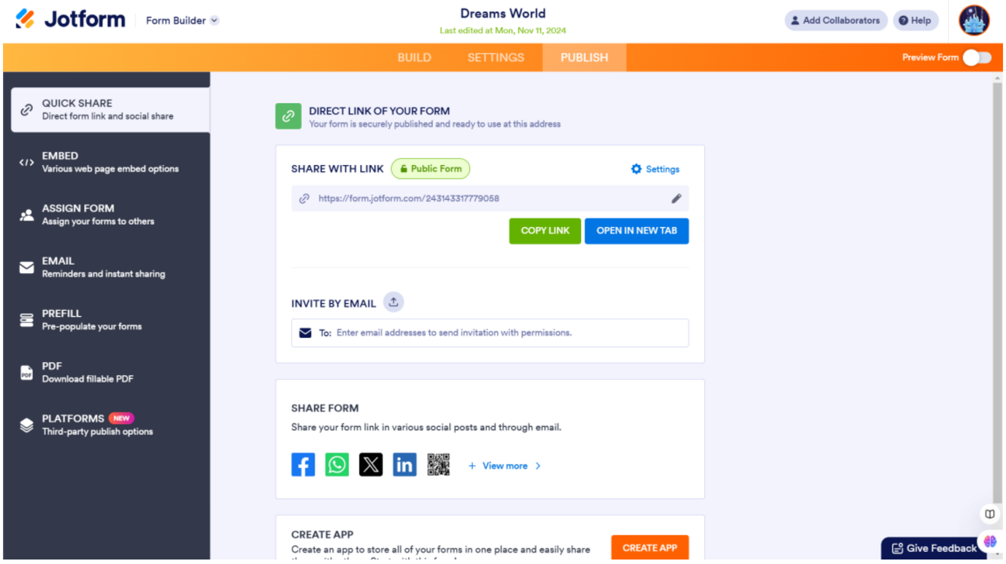


3. Now create a form to get the customer details like Name, Phone,Email, Address and type of property the customer is interested in.





4. Once the form is created, publish it by clicking on publish.

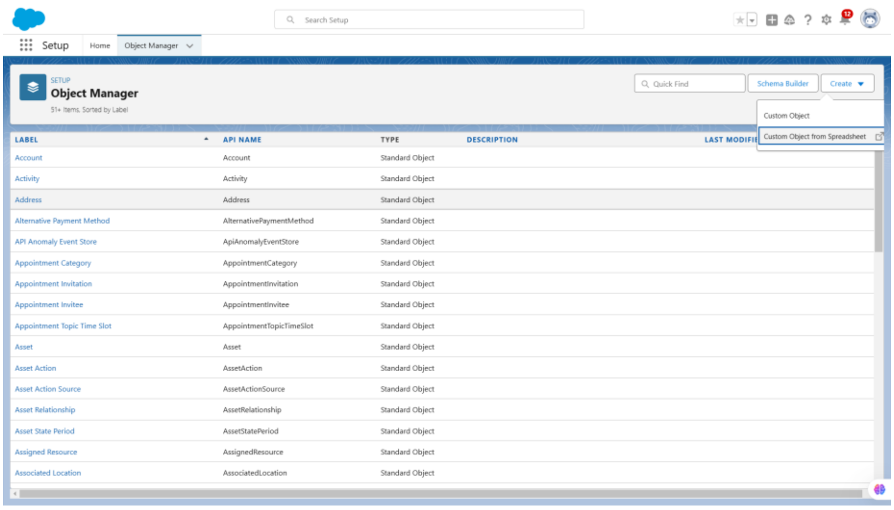


The jotform link: https://form.jotform.com/243143317779058

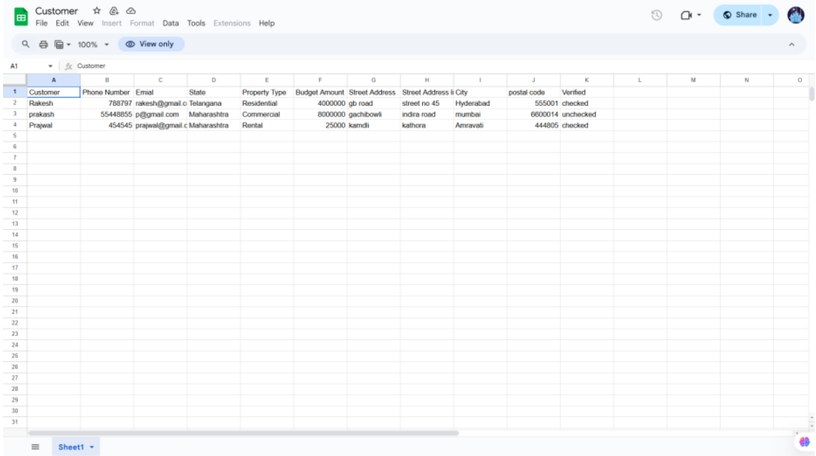
Create Objects from Spreadsheet.

<u>Create Customer object</u>

1. Go to your object manager and and click on create object from spreadsheet.

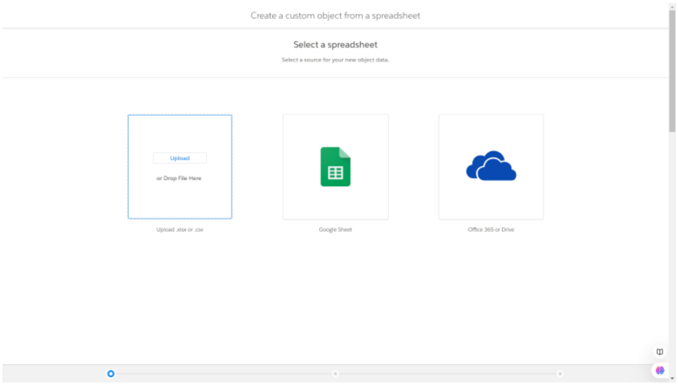


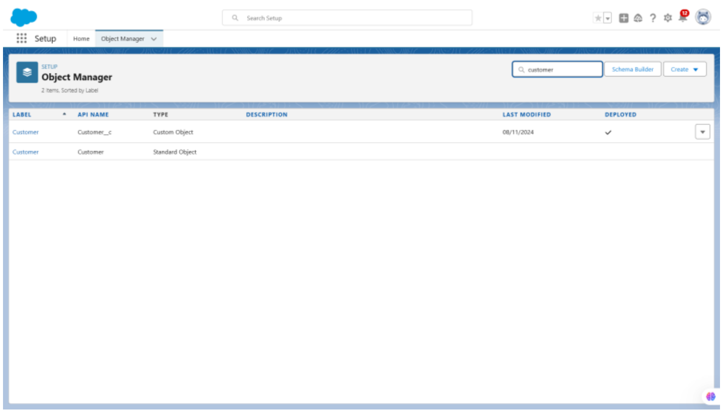
2. Click on the link to get the spreadsheet



3. Downloading of customer spreadsheet

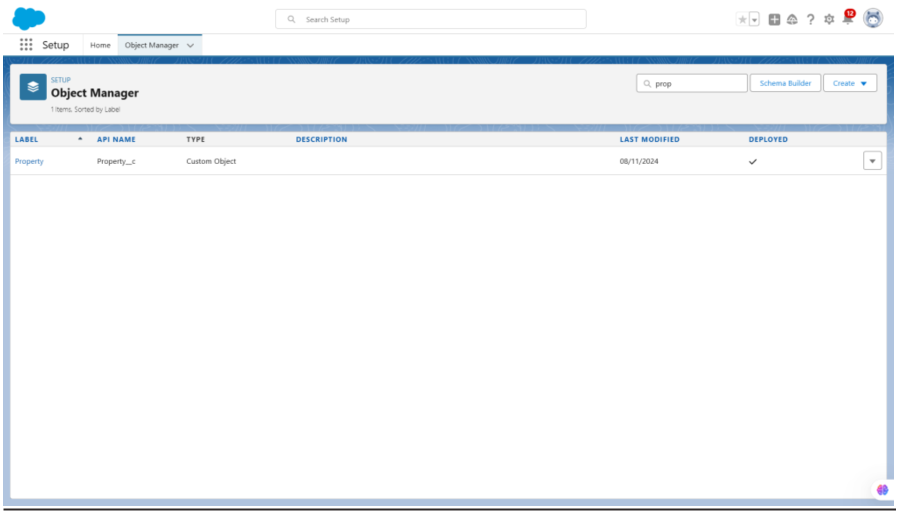
4. After downloading, upload the file, map the fields and upload to create an object.





Create Property object

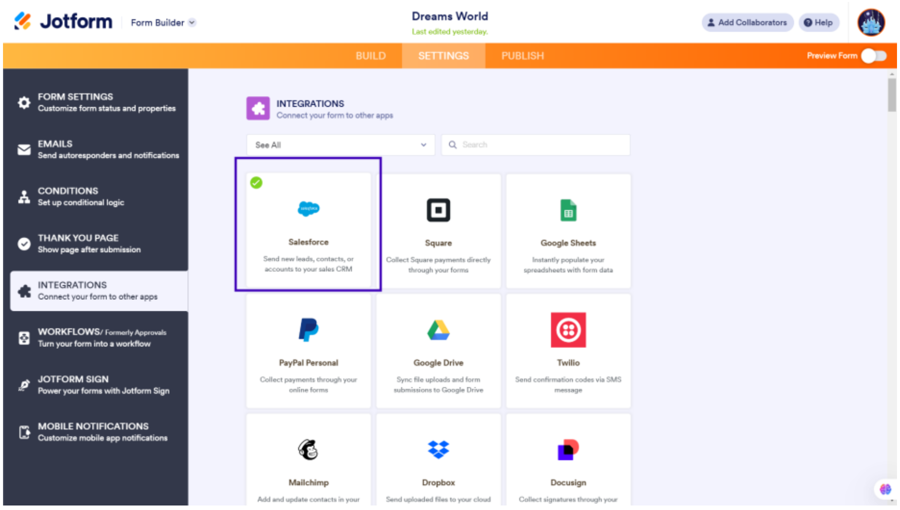
1. Follow the same from the customer object to create the Property Object



Integrate Jotform with Salesforce Platform

Activity 1

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

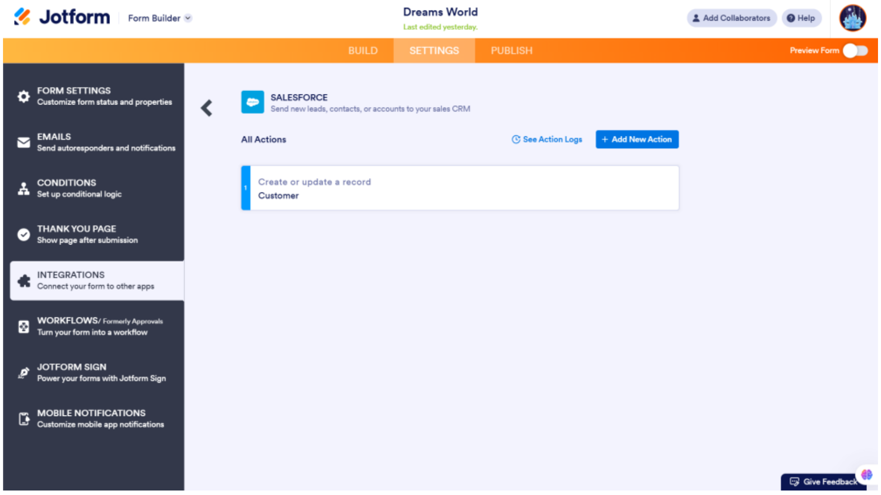


2. Click on User Integration and choose “Add to From”

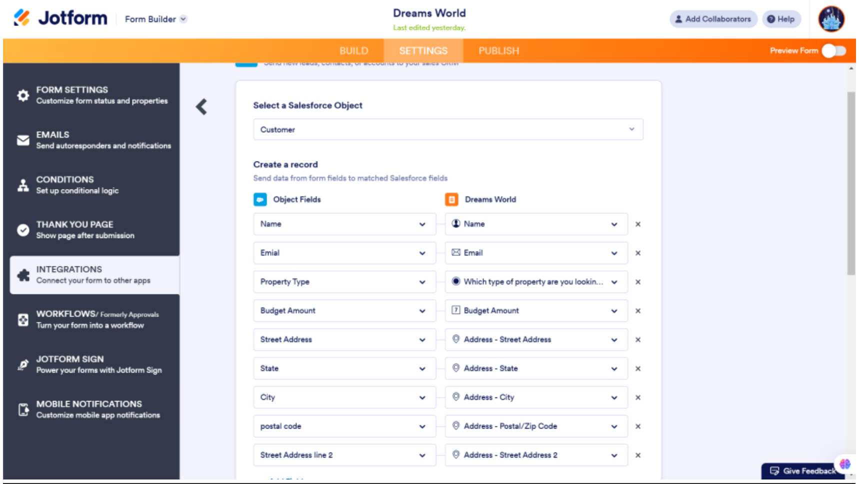
3. Select the Org with which you want to Integrate your jotform with.

4. Select an Action - Create a record.

Select a Salesforce Object : - Customer



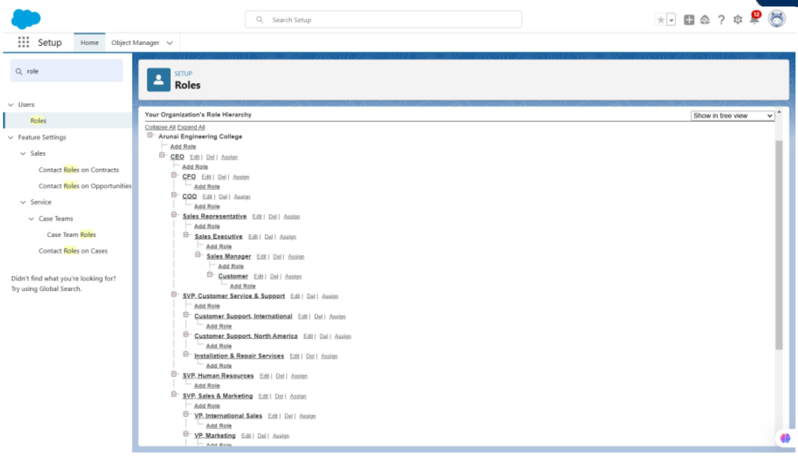
5. Map Each and every field on the Object with the fields on the form and "Save Action"
6. Then "Save the Integration" and "Finish".



Create Roles

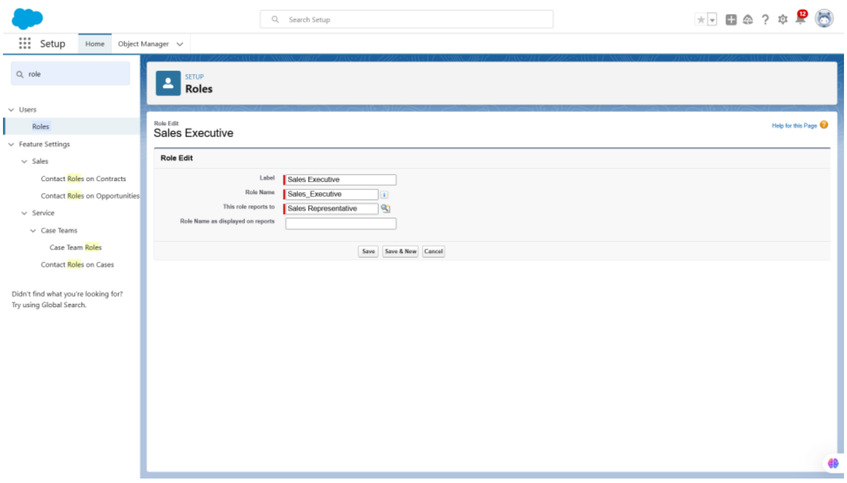
<u>Sales Executive Role</u>

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

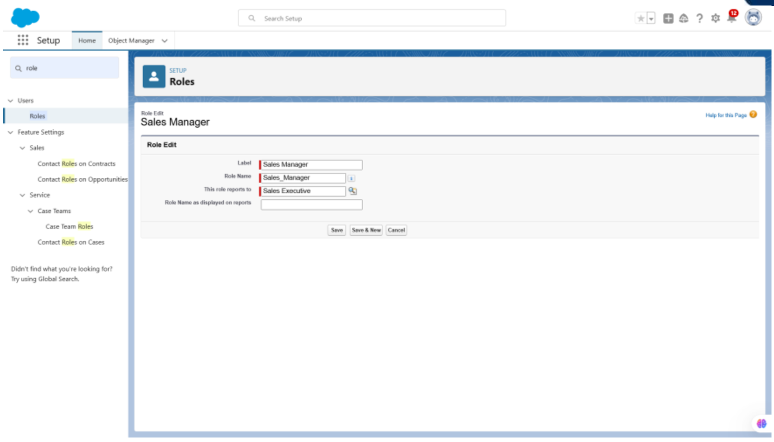


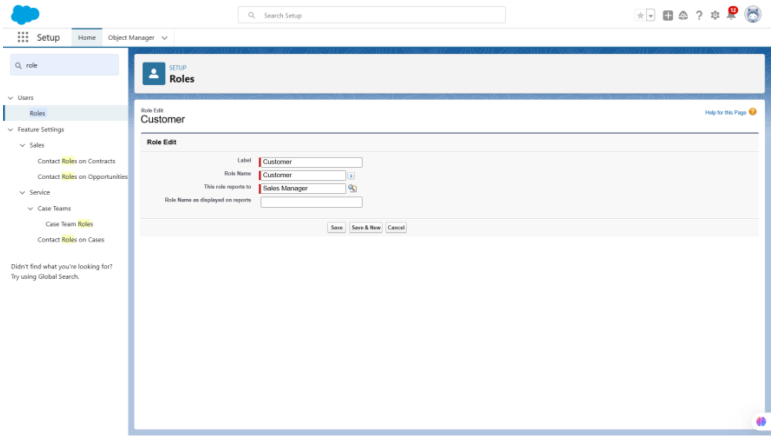
2. Label - Sales Executive

   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.

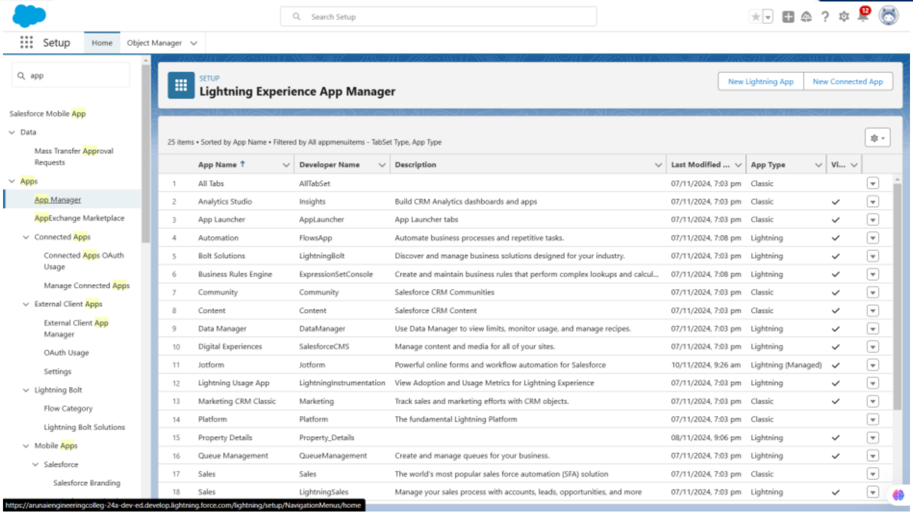


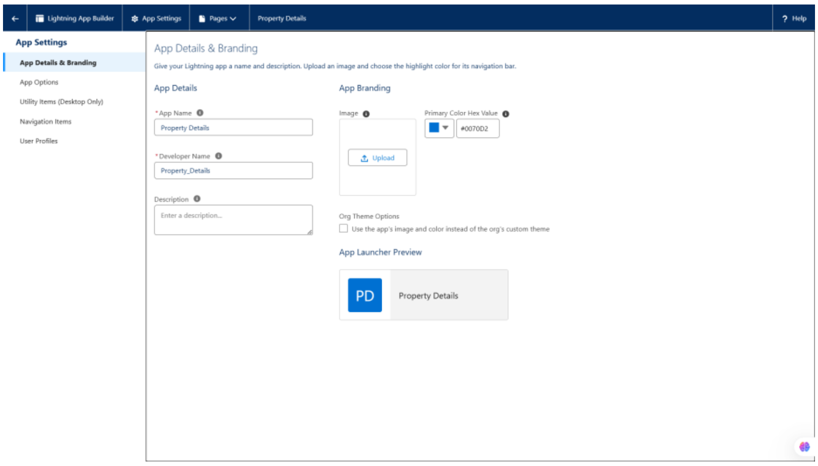


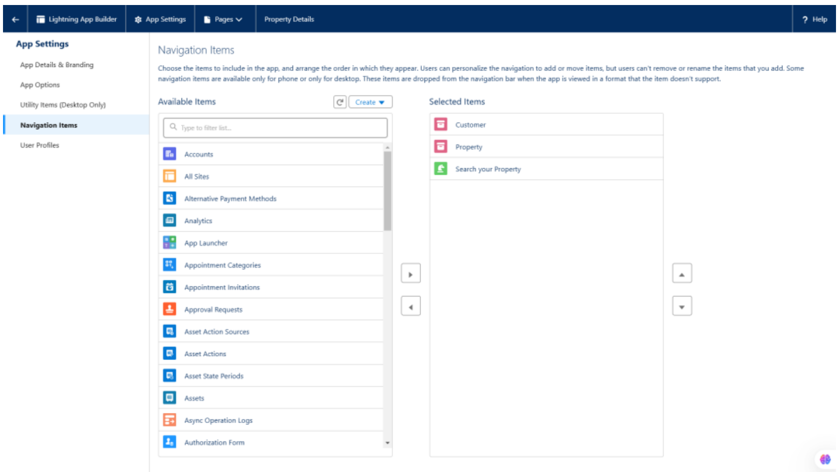
Create a Property Details App

Activity 1

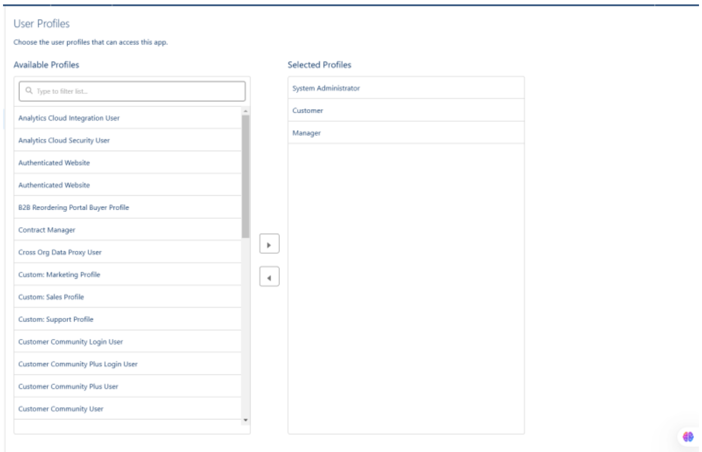
1. From Setup>> Go to App Manager and click on New Lightning App and Name it as “Property Details” and add “Customer” and “Property” Object.







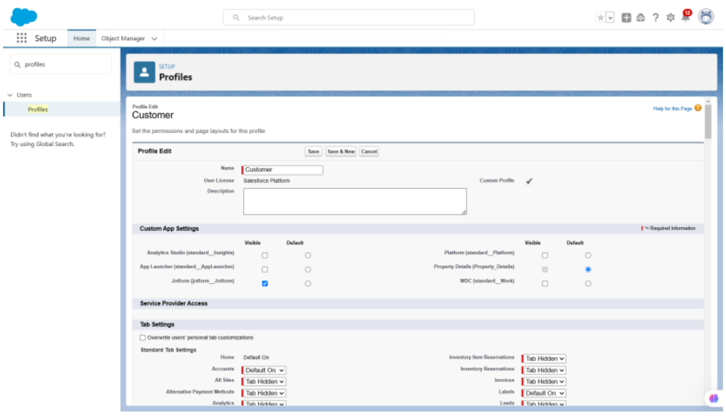
2. Click Next >> Next >> Save and Add “System Admin ”Profile.



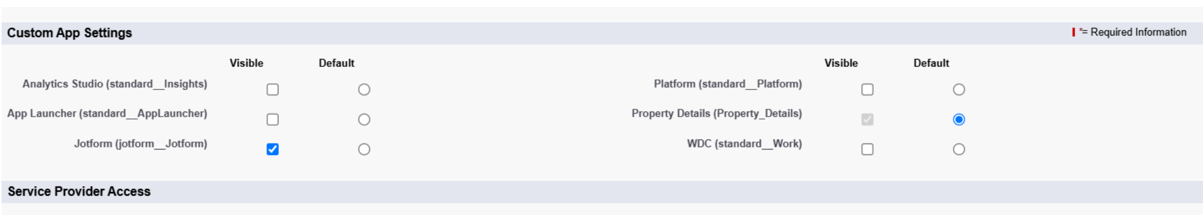
Create Profiles

<u>Customer</u>

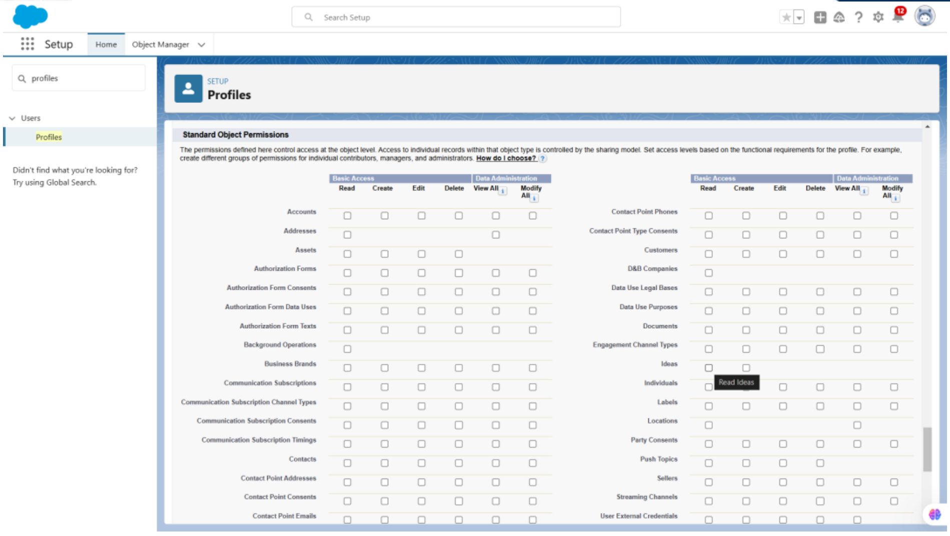
1. From Setup >> Go to Profiles and Clone Salesforce Platform User and Name it "Customer".



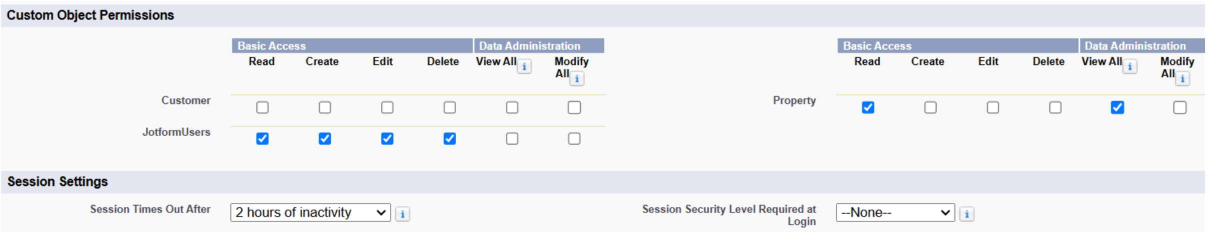
2. Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.



3. Also Remove all the Standard Object Permissions.



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



Custom Object Permissions

	Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All
Customer	☐	☐	☐	☐	☐	☐
JotformUsers	☑	☑	☑	☑	☐	☐

	Basic Access				Data Administration	
	Read	Create	Edit	Delete	View All	Modify All
Property	☑	☐	☐	☐	☑	☐

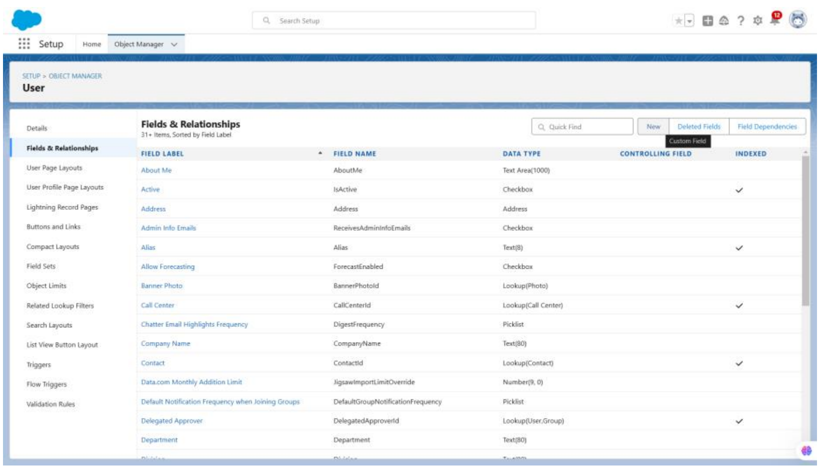
Session Settings

Session Times Out After	2 hours of inactivity	Session Security Level Required at Login	--None--

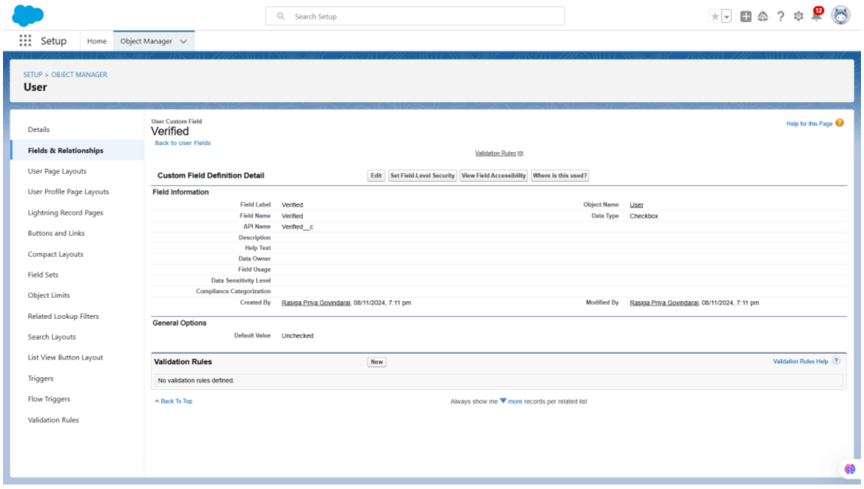
Create a Check Box field on user

Activity 1

1. Setup >> Object Manager >> Search for User >> Fields and Relationships



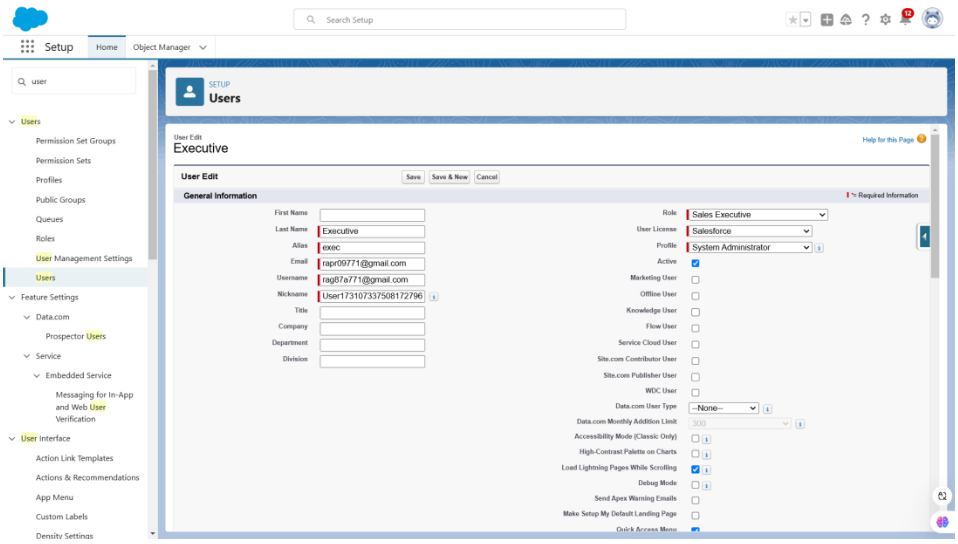
2. Create new Field Named as "Verified" as Data type "Check Box"



Create Users

User 1

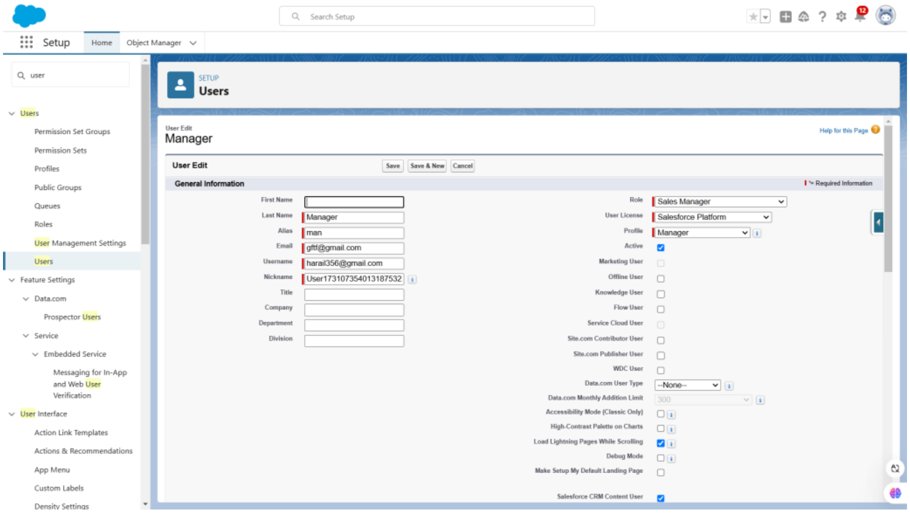
1. Go to Setup --> Administration --> Users --> New User
2. Last Name - Executive
3. Role - Sales Executive
4. License - Salesforce
5. Profile - System Administrator
6. Save



User 2

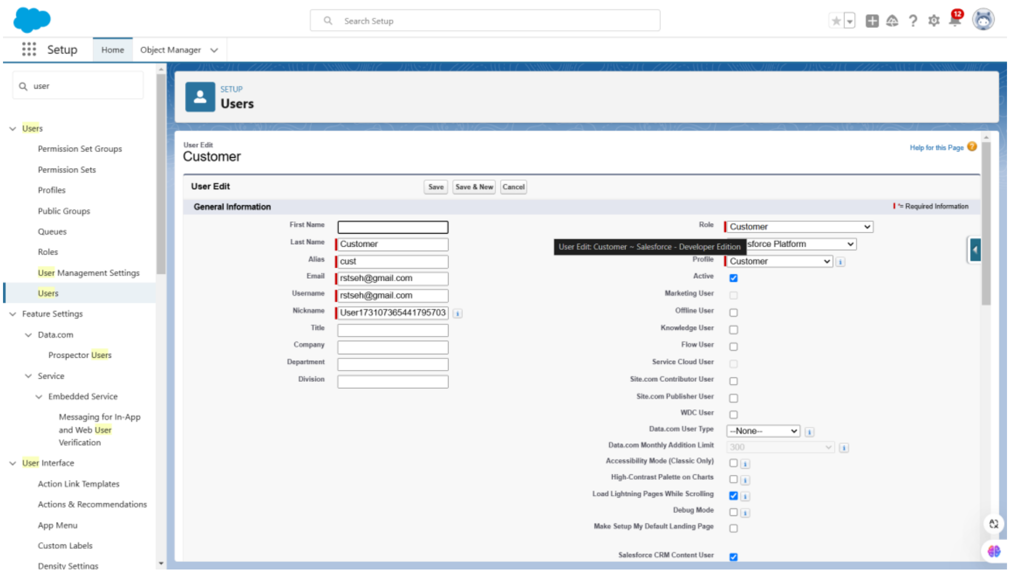
1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Manager
3. Role >> Sales Manager

4. License >> Salesforce Platform
5. Profile >> Manager
6. Save



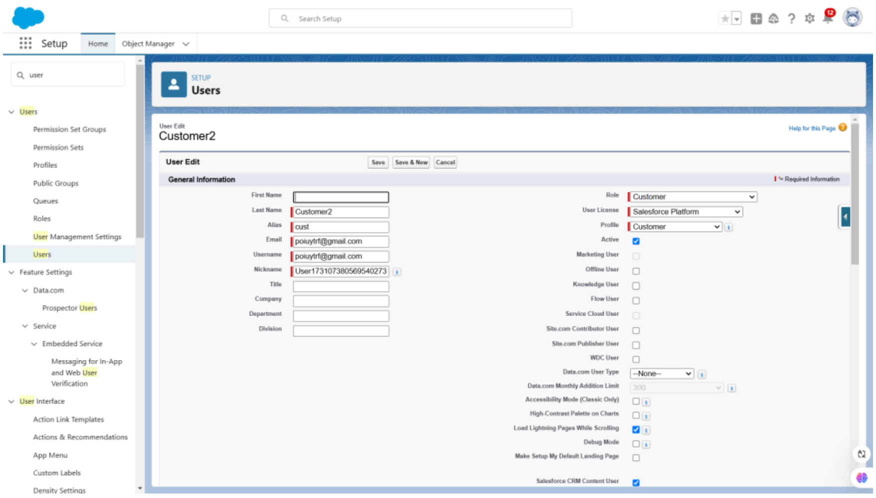
User 3

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

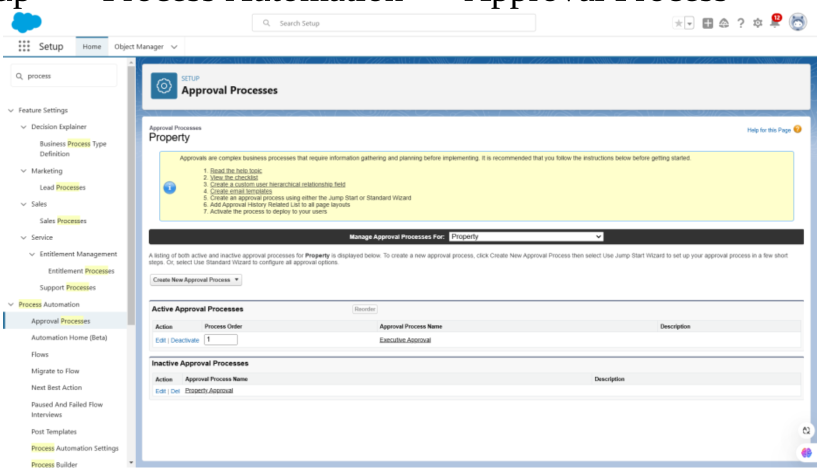
1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Customer2
3. Role >> Customer
4. License >> Salesforce Platform
5. Profile >> Customer
6. Make Sure the verified check box is “checked”
7. Save



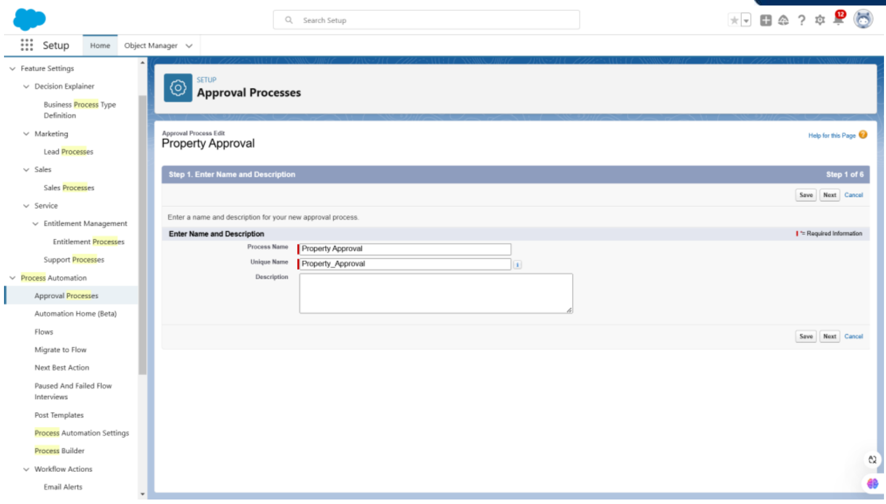
Create an Approval Process for Property Object

<u>Activity 1</u>

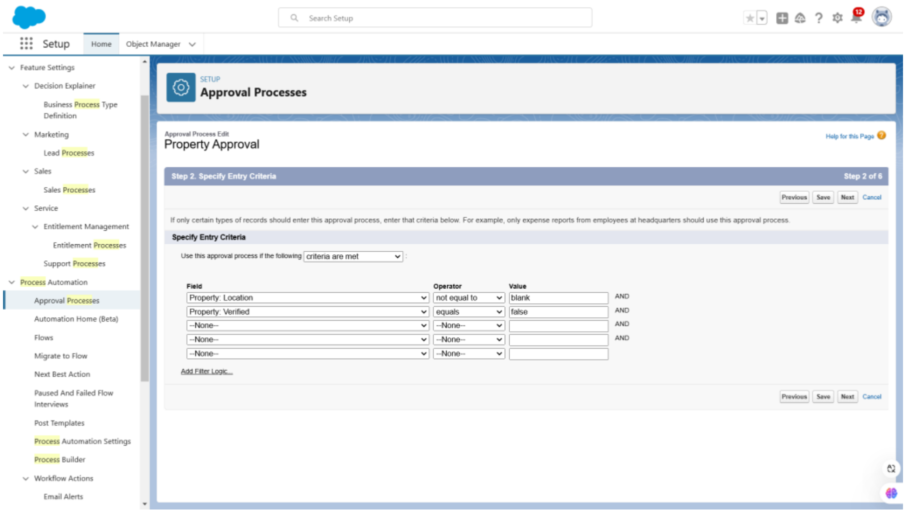
1. From Setup >> Process Automation >> Approval Process



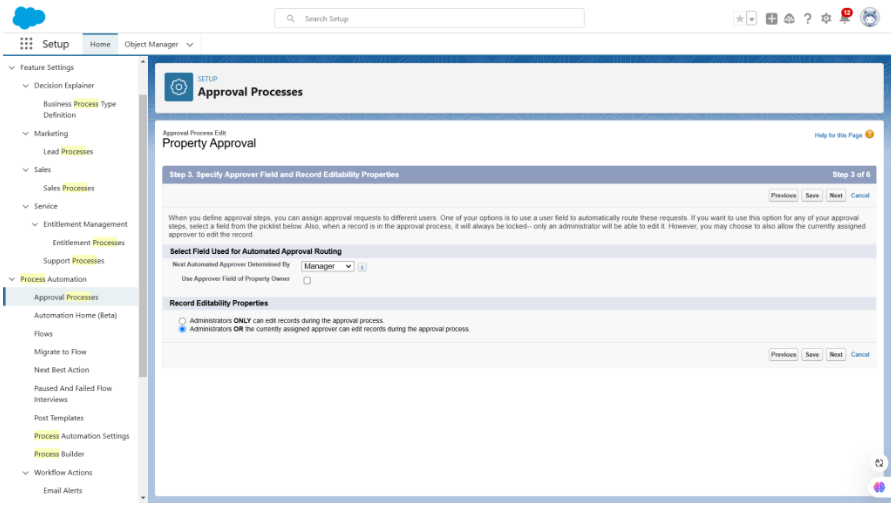
2. Process Name - Property Approval



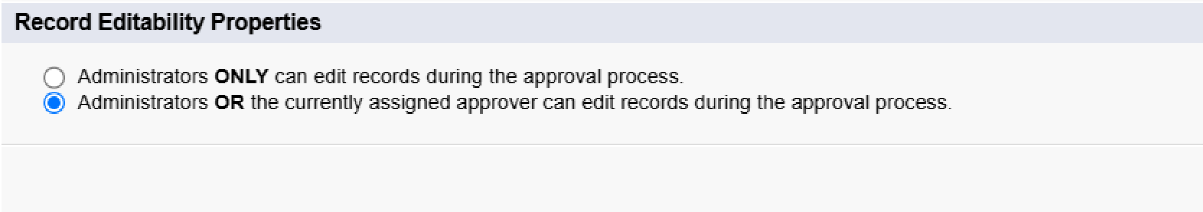
3. Give 2 criteria -Location is not equal to blank,Verified Equals false



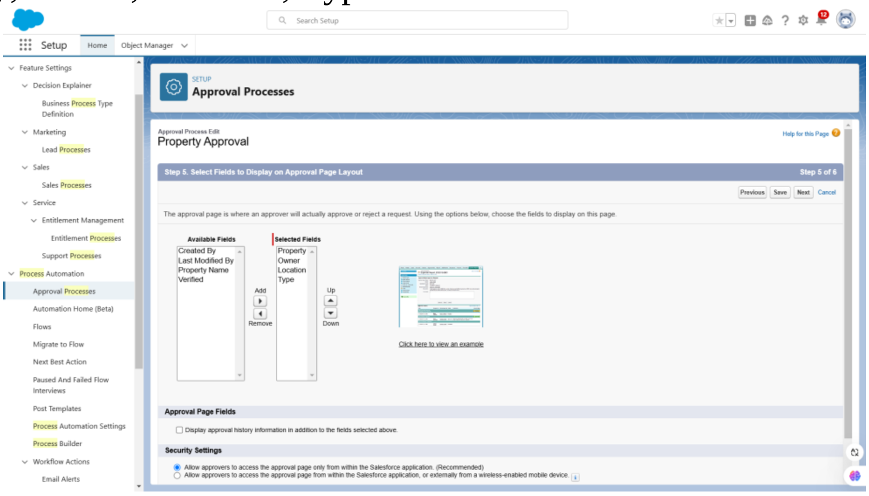
4. Click next and "Next Automated Approver Determined By" Select Manager



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.

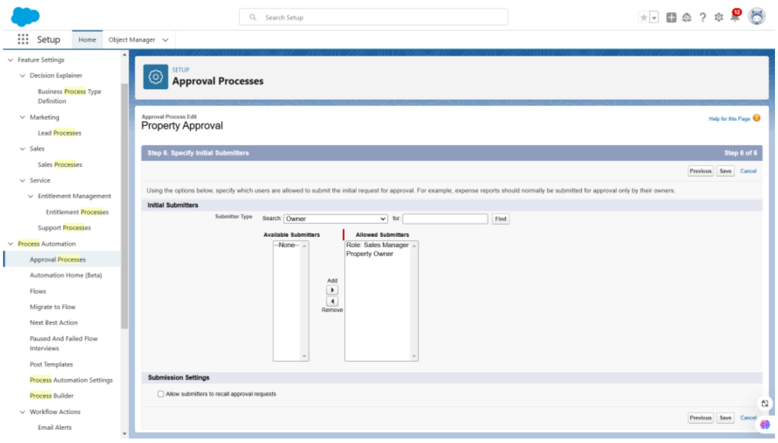


7. Click Next and Select the initial Submiters >>

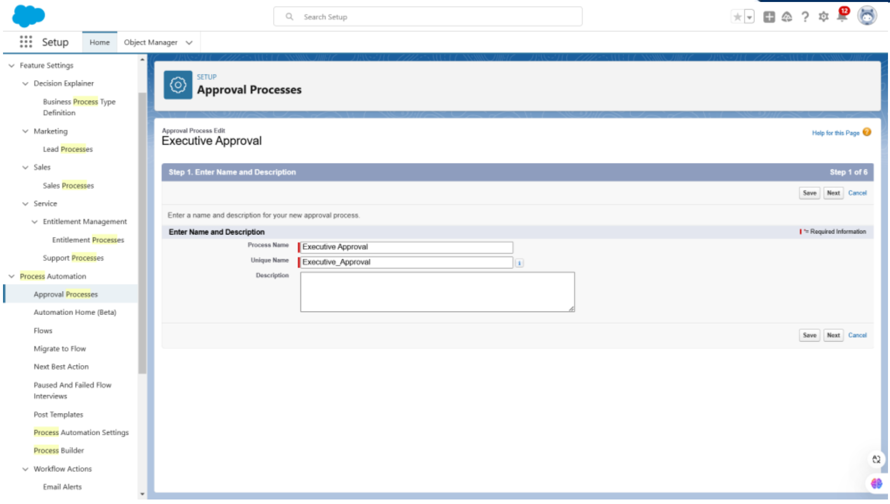
   a) Owner >> Property Owner

   b) Roles >> Sales Manager

8. Save.

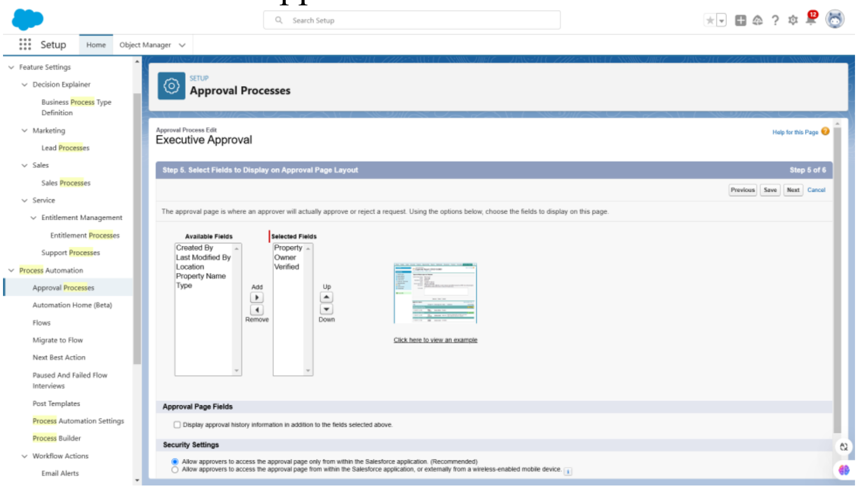


9. Add an approval step name “Executive Approval ”



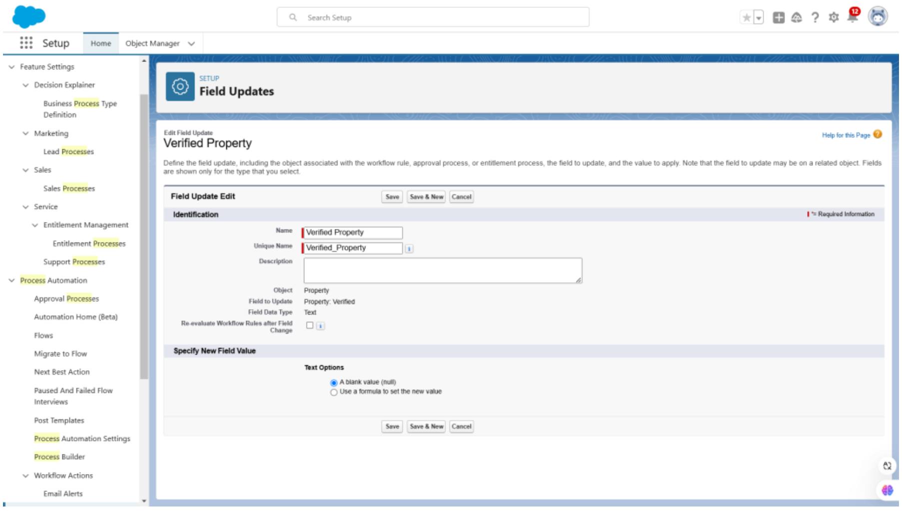
10.specify the Criteria >> All record should enter

11. click next and select the Approver as " Sales Executive " and "Save"



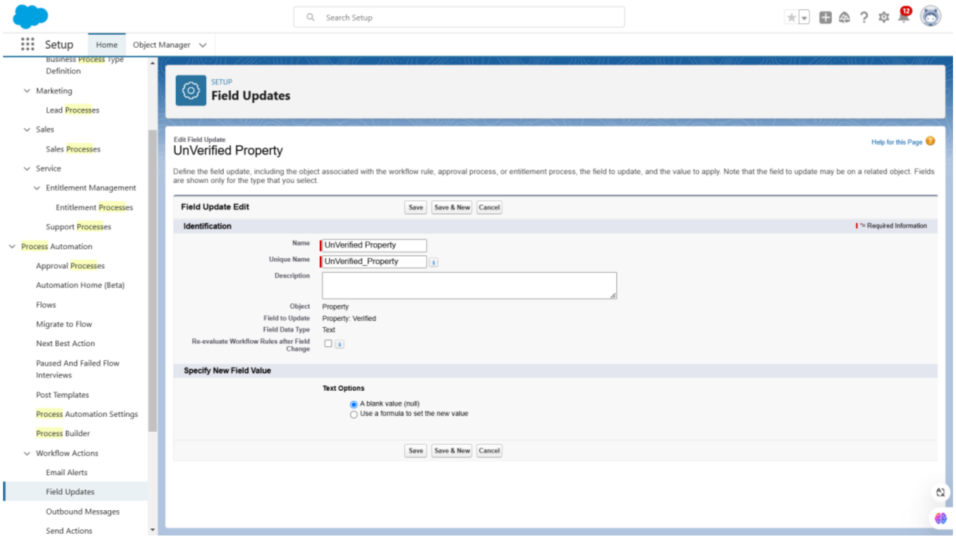
12.Add One field Update as "Verified Property"

a. Select Object >> Property
b. Field to Update >> Verified
c. Field Data Type >> CheckBox
d. Select CheckBox Option as "True"
e. Save.

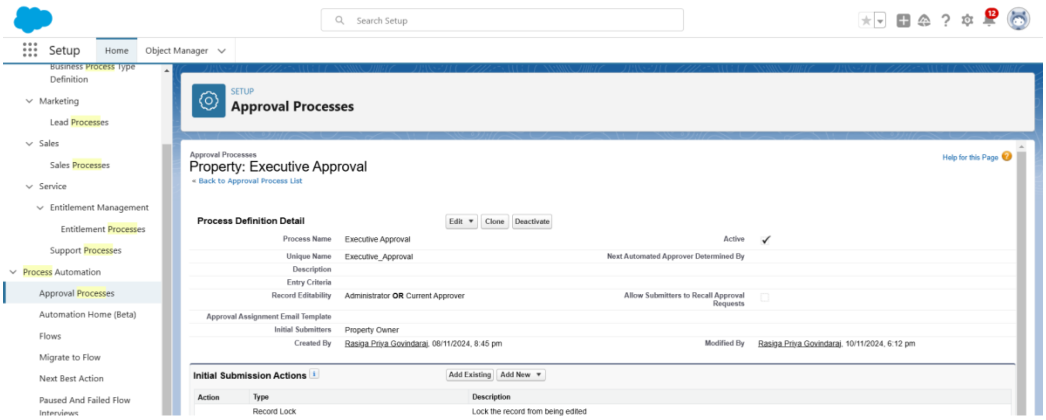


13. Add One field Update as “UnVerified Property”

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



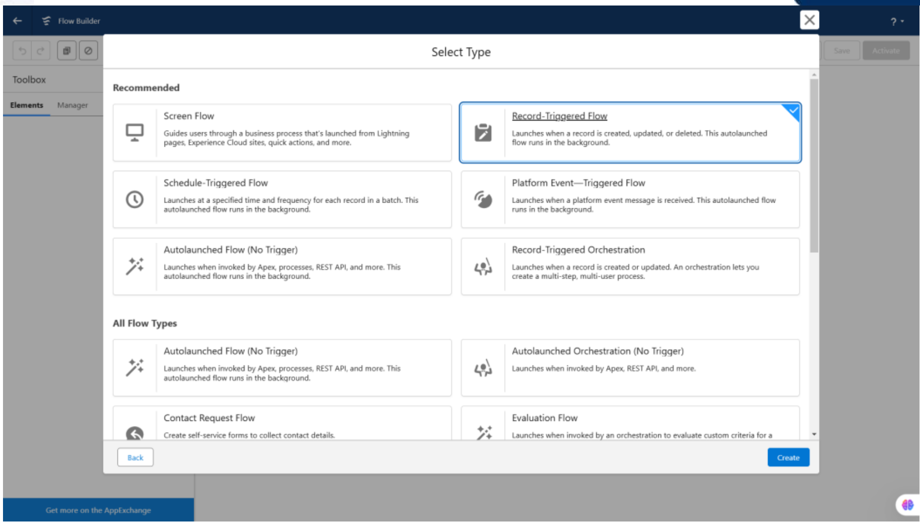
14. Activate the Approval Process.



Create a Record trigger flow to submit the Approval Process Automatically.

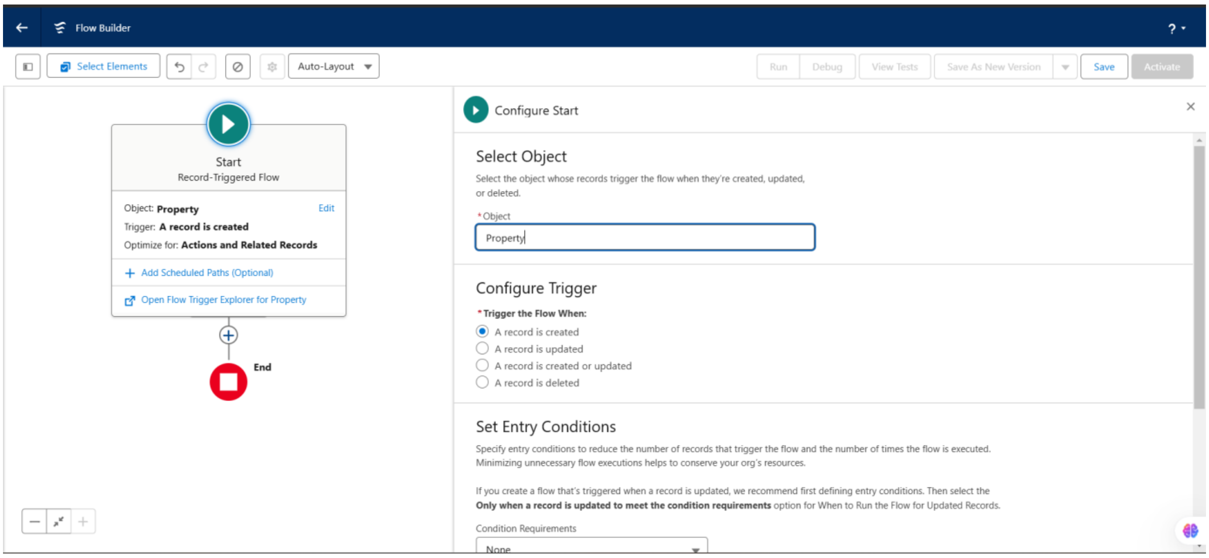
<u>Activity 1</u>

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

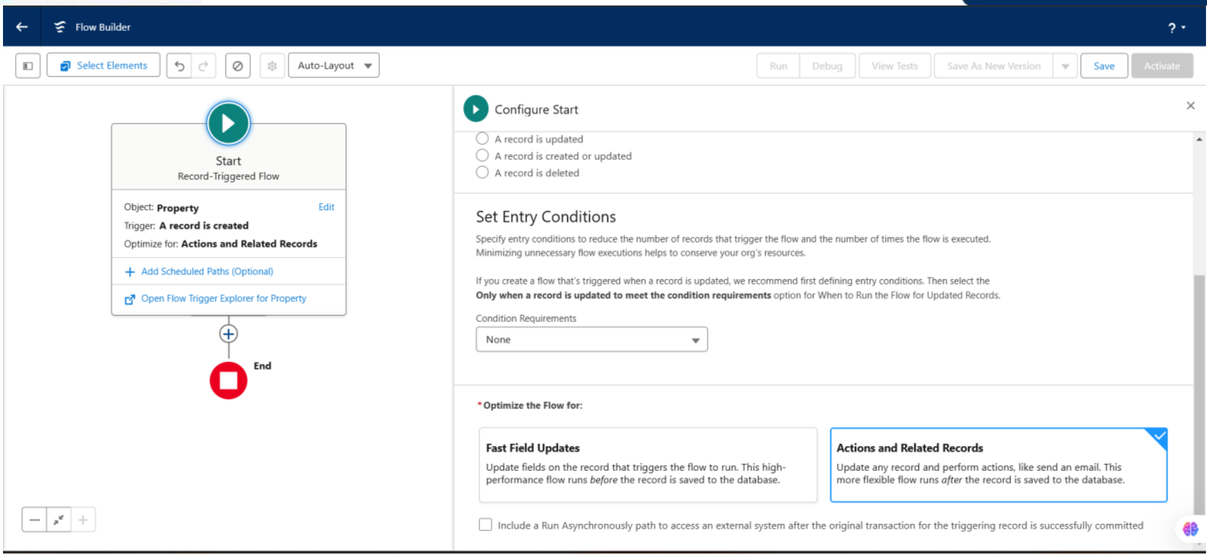


2. Select Object >> Property

3. Select “Trigger the flow when” >> “A record is created”

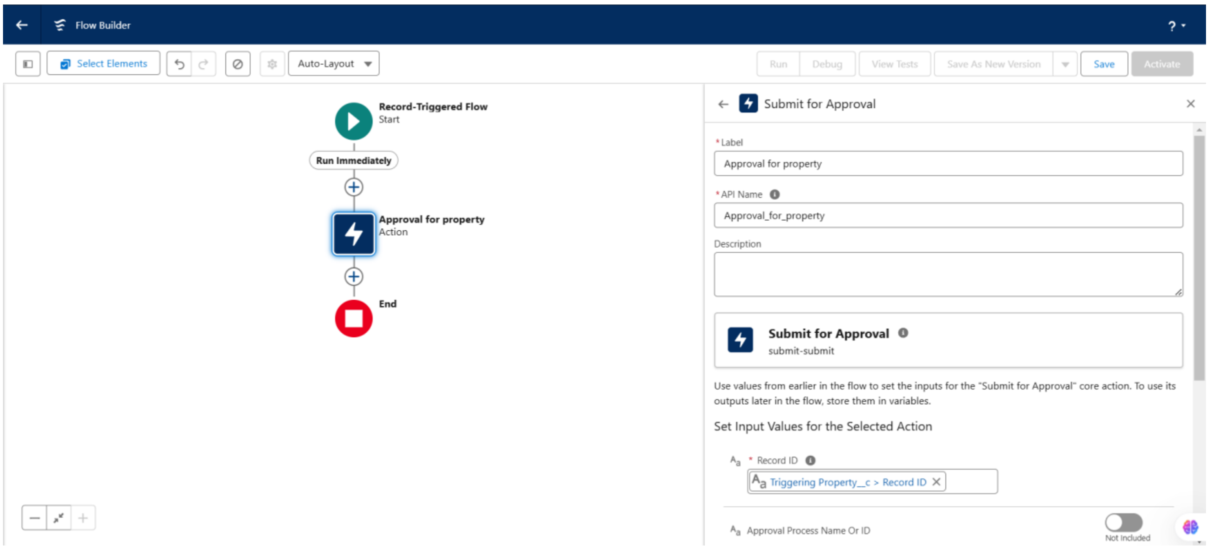


4. Set Entry Conditions >> “None”
5. Add a “Action” >> “Submit for Approval”



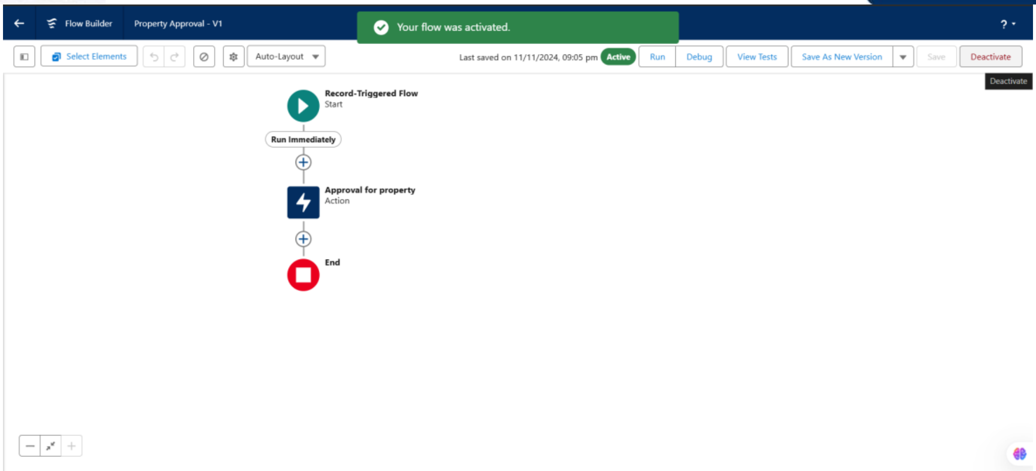
6. Give Label >> Approval for property

7. Record Id >> {!$Record.Id}



8. Done

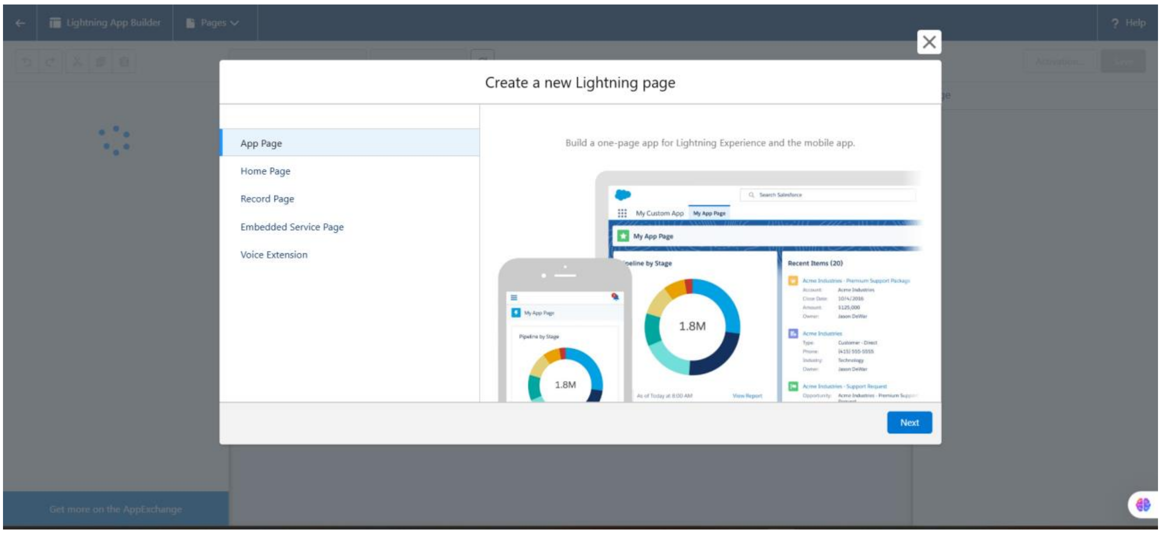
9. Save the Flow and Give label as "Property Approval" and "Activate"



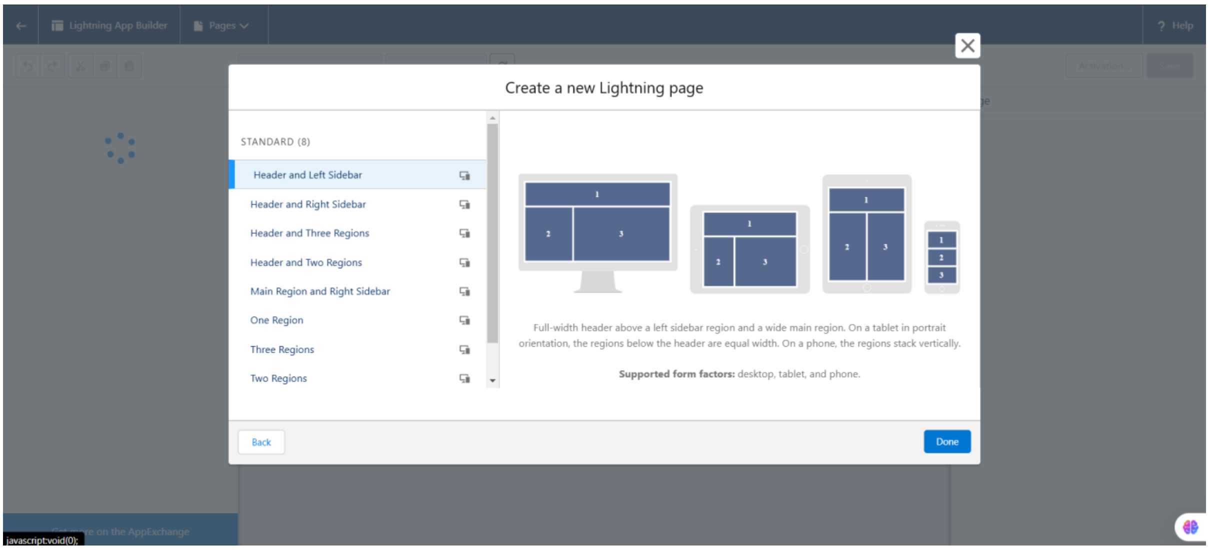
Create an App Page

Activity 1

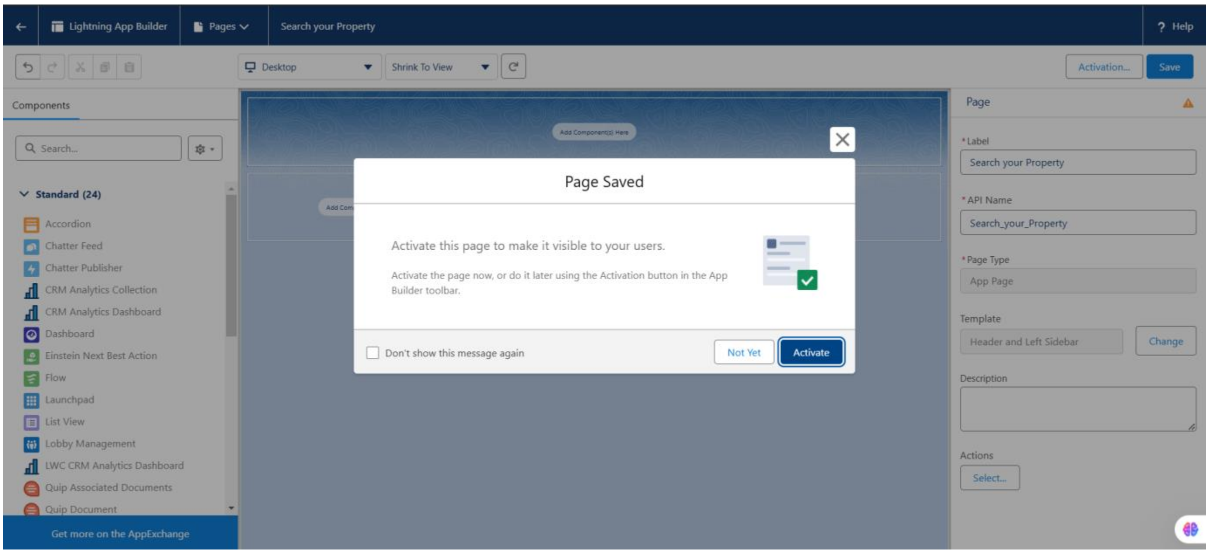
1. From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and
   Click on Next.



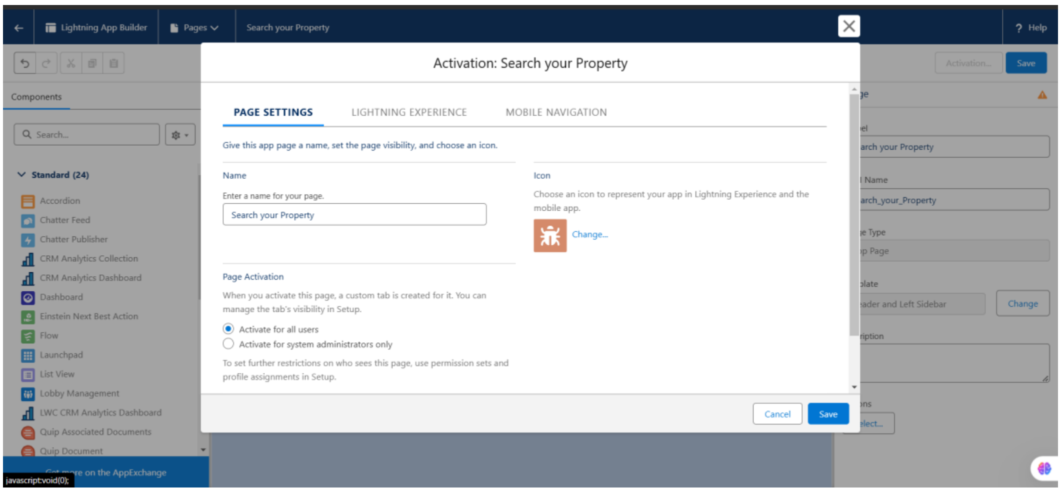
2. Give Label as “Search your Property” click “Next”.
3. Click “header and Left Sidebar” and Click on “Done”



4. Click on “Save ” and then click on “Activate”.

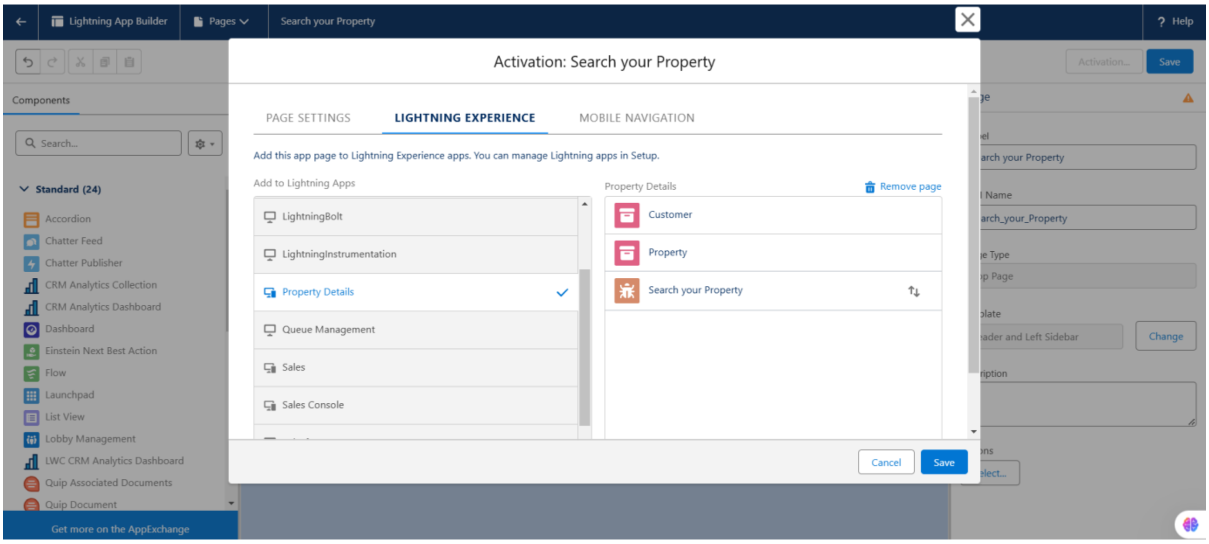


5. From Page Setting select page activation as “Activate for all Users”.



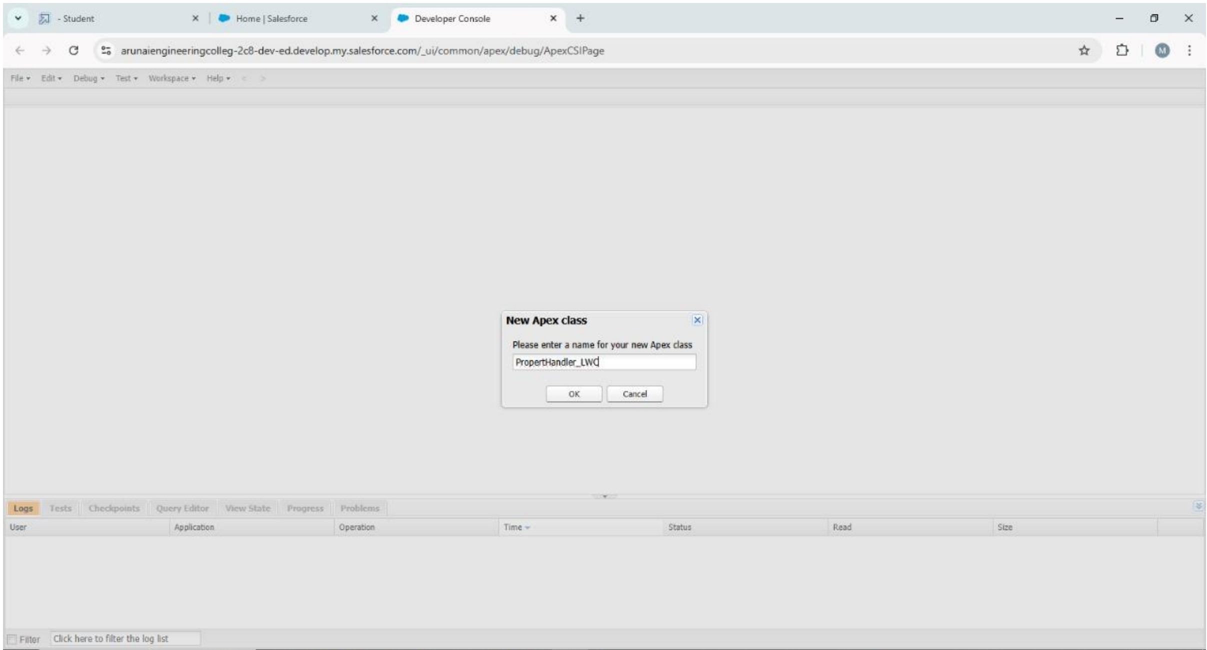
6. From Lightning Experience Click on “Property Details” and click on Add Page“.

7. Then Click on “Save”



Create a LWC Component

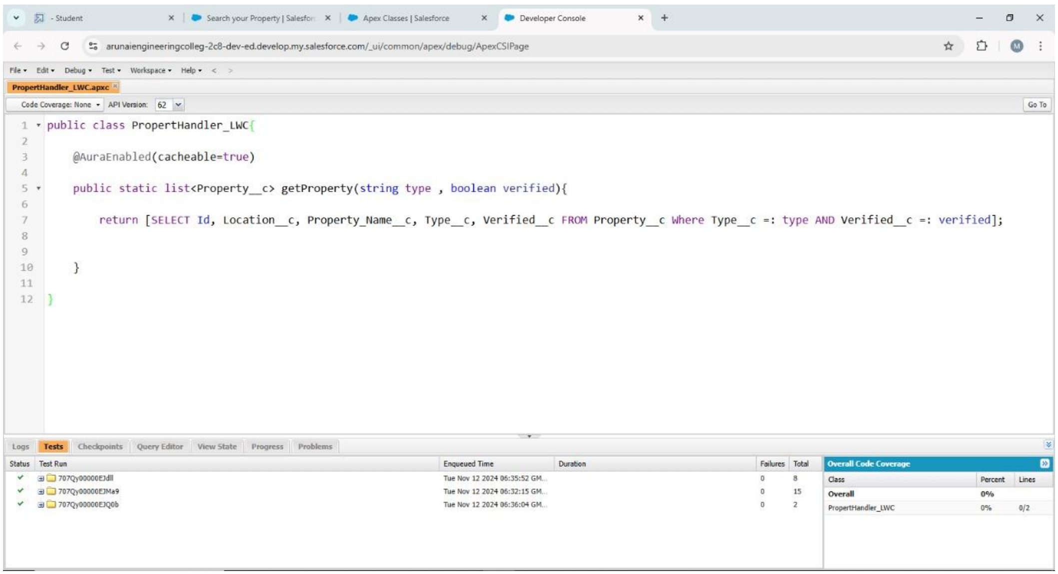
1. Create an Apex Class and make it aura enabled and name it “PropertHandler_LWC”



2.After creating the apex class paste the code.

CODE:

```
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];
   }
}
```

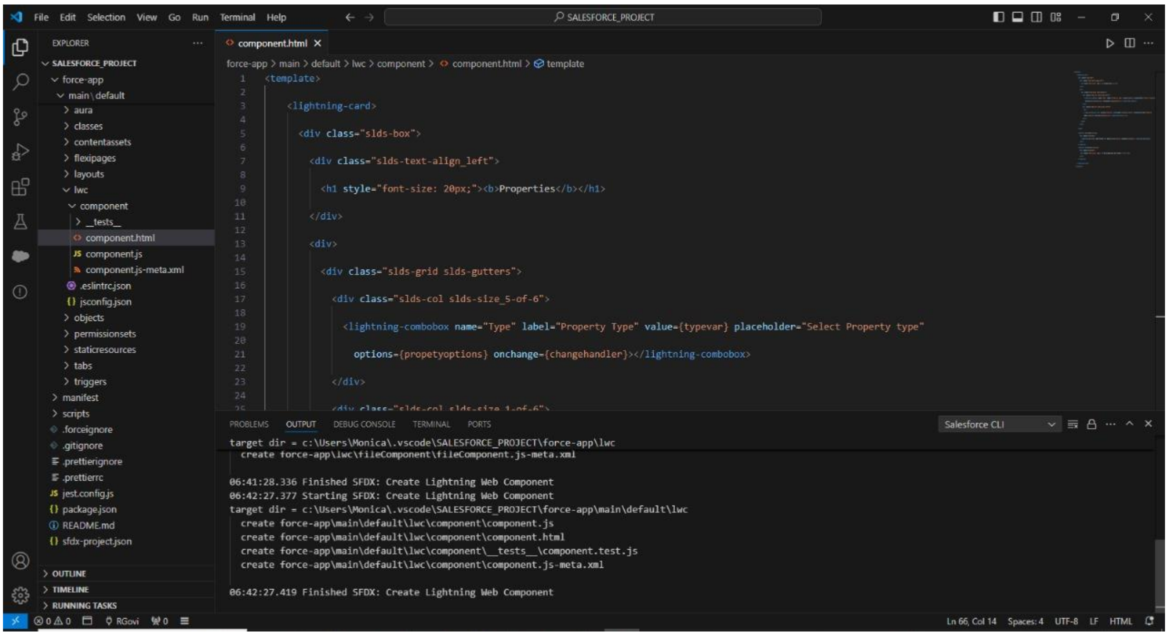


2. Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.
3. Enter your login id and password to authorize your org.
4. Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to. (Example -)
5. In your Html File Write this code

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>
```

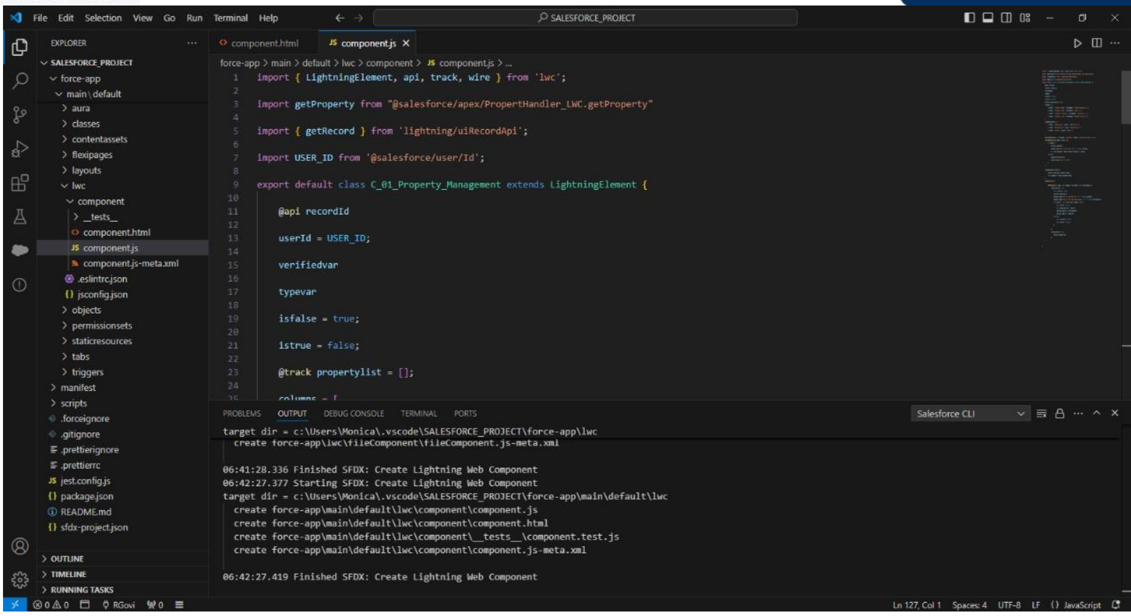


6. In Your Js File Write this code

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" },
```

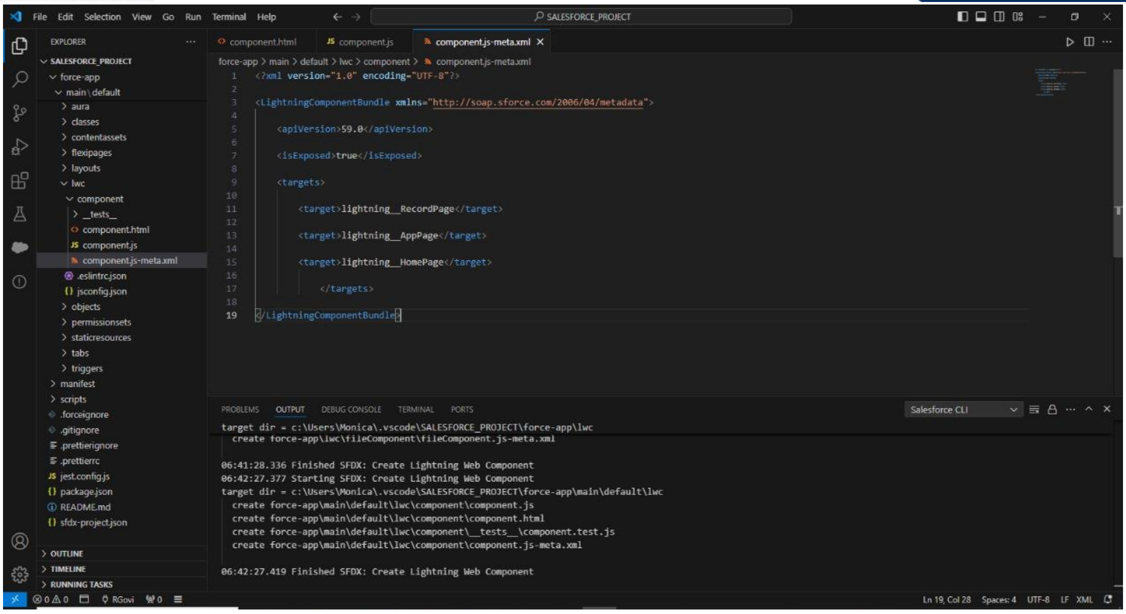
```
        { label: "rental", value: "rental" }
    ]
    @wire(getRecord, { recordId: "$userId", fields: ['User.Verified__c'] })
    recordFunction({ data, error }) {
        if (data) {
            console.log(data)
            console.log("This is the User Id ---> "+this.userId);
            this.verifiedvar = 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)
            })
    }
}
```



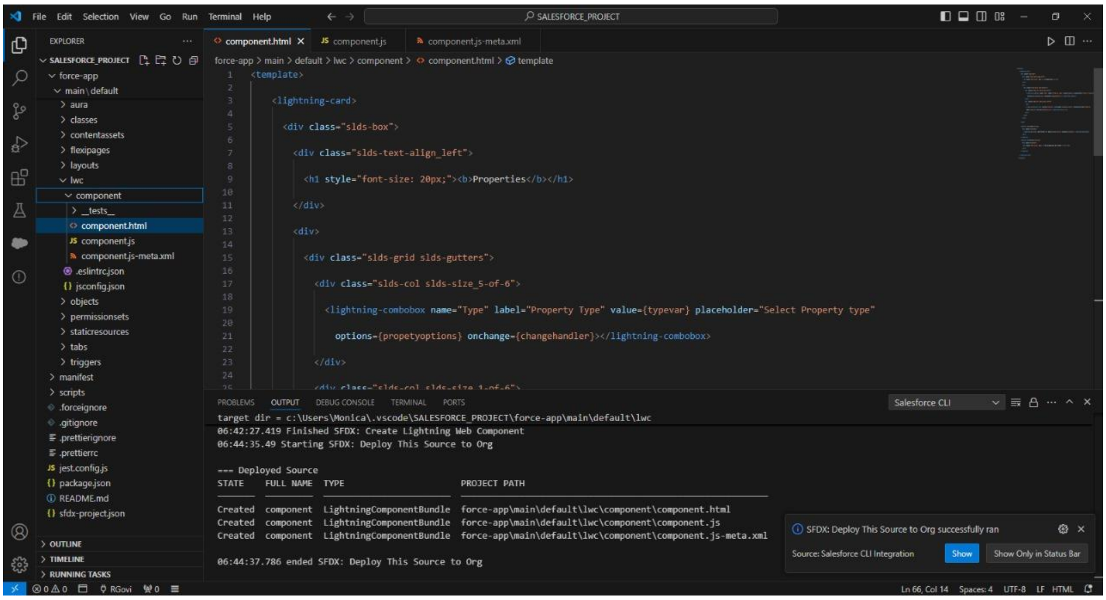
7. In Your metafile give your targets to deploy the component.

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>
```



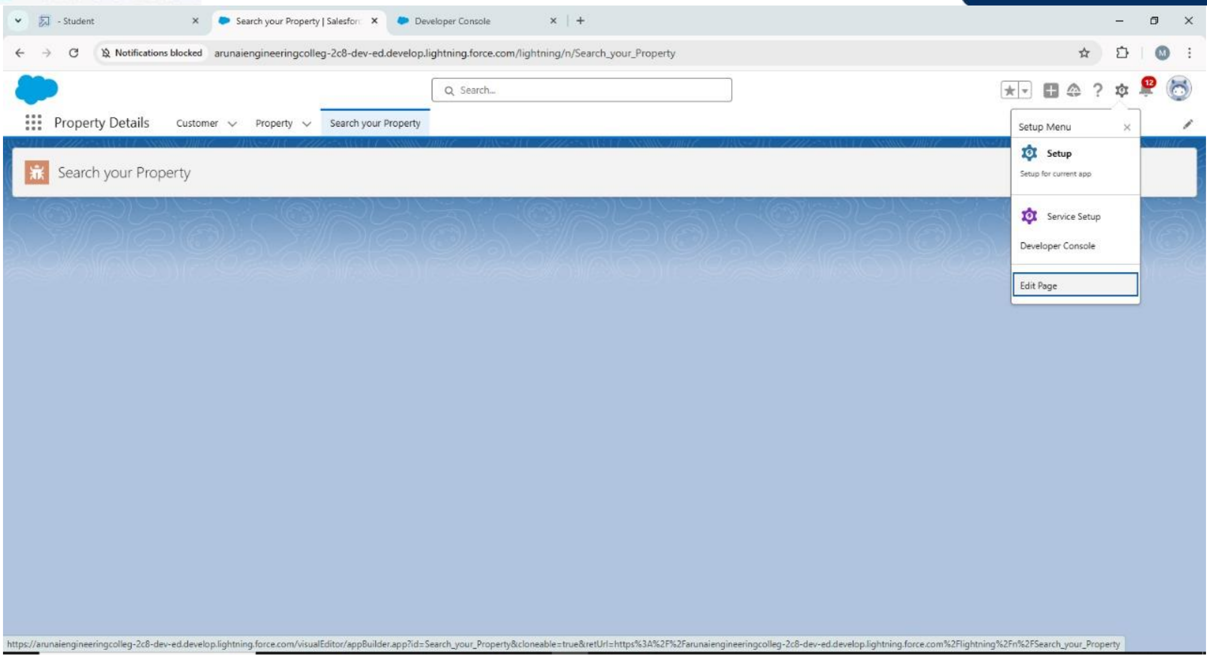
8. After Saving all the three Codes , Right Click and deploy this component to the org.



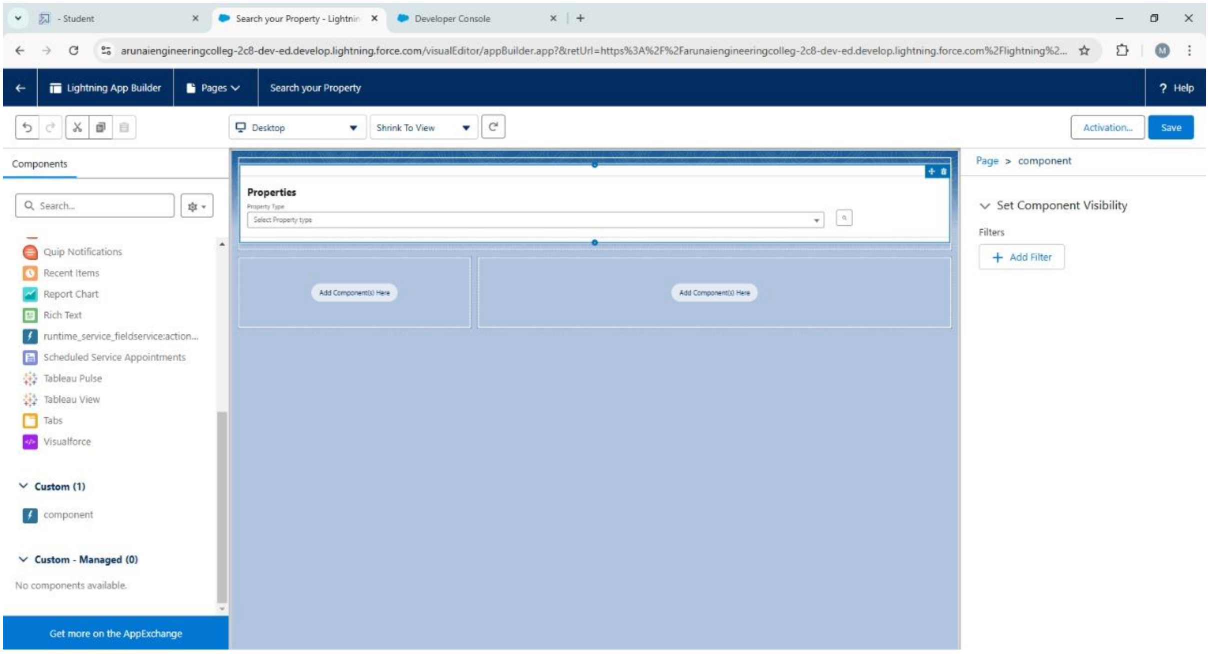
Drag this Component to your App Page

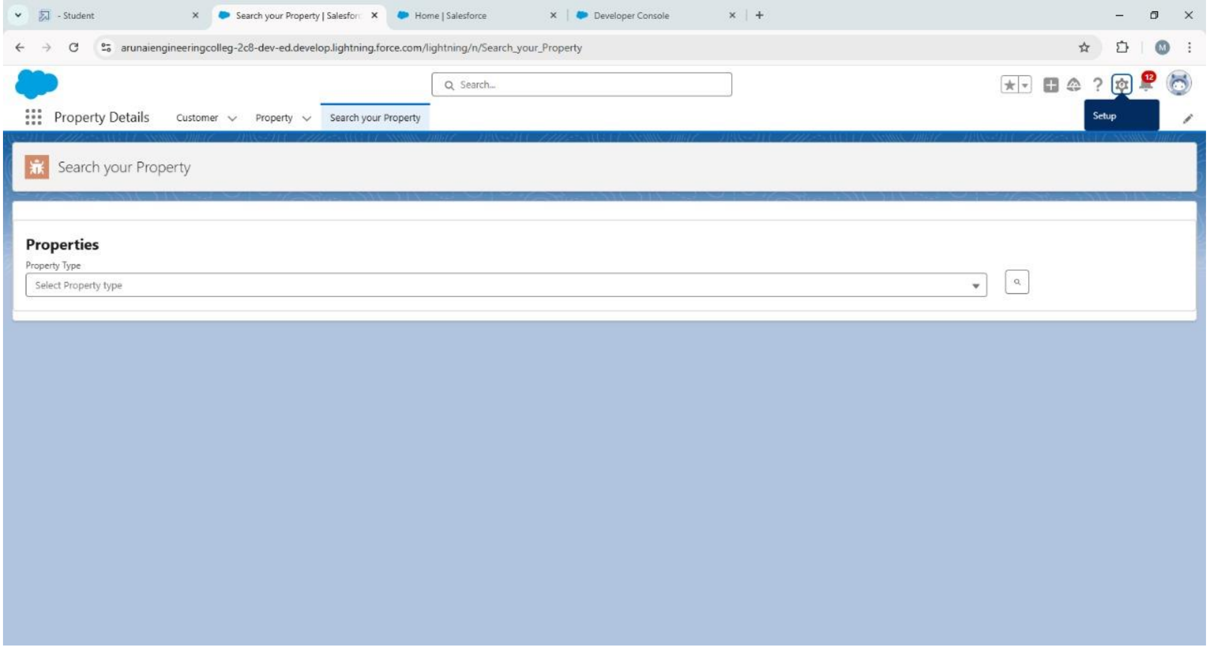
Activity 1

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. Drag the Component to your App Page and Save the Page.

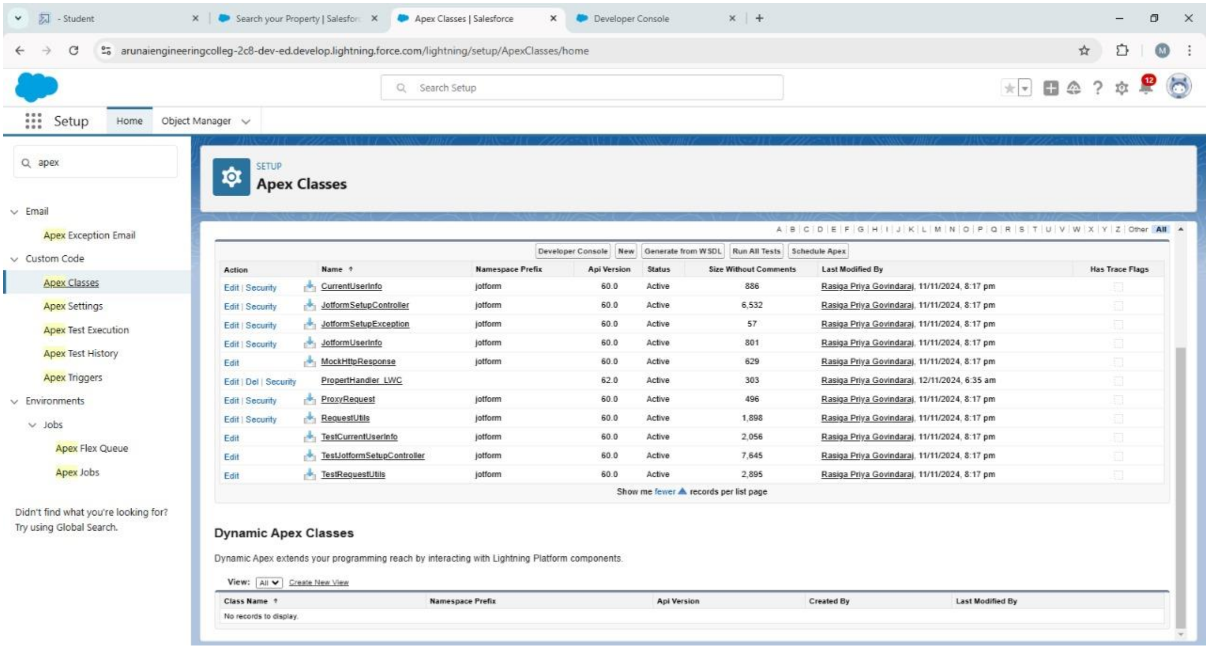




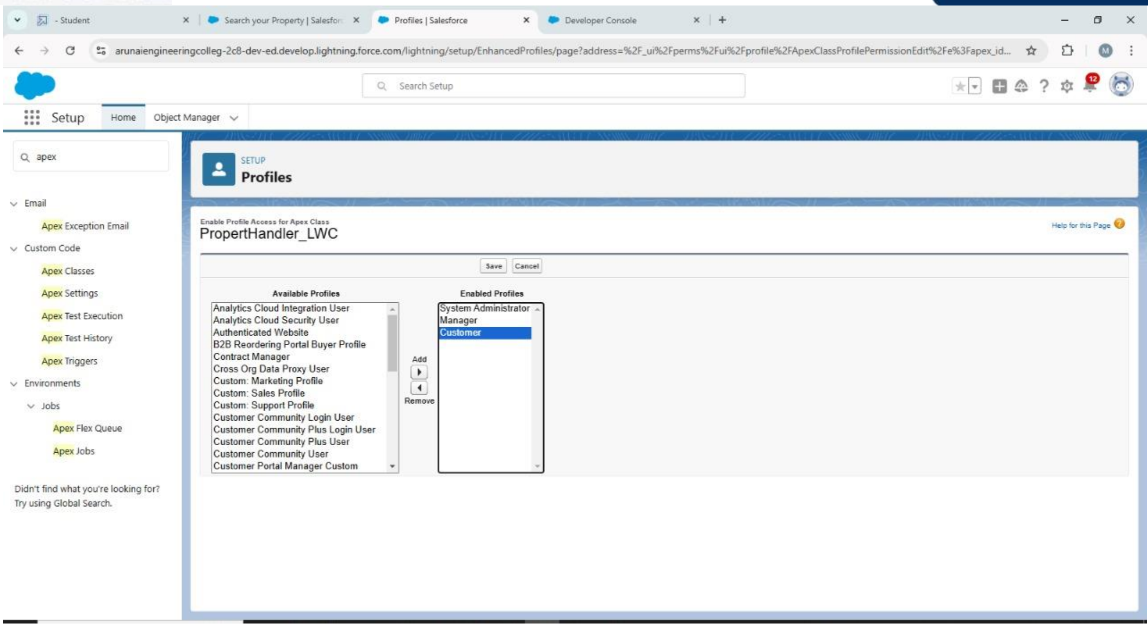
Give Access of Apex Classes to Profiles

Activity 1

1. From Setup >> Search For Apex Classes >> Click on “Security” behind “PropertyHandler__LWC”.



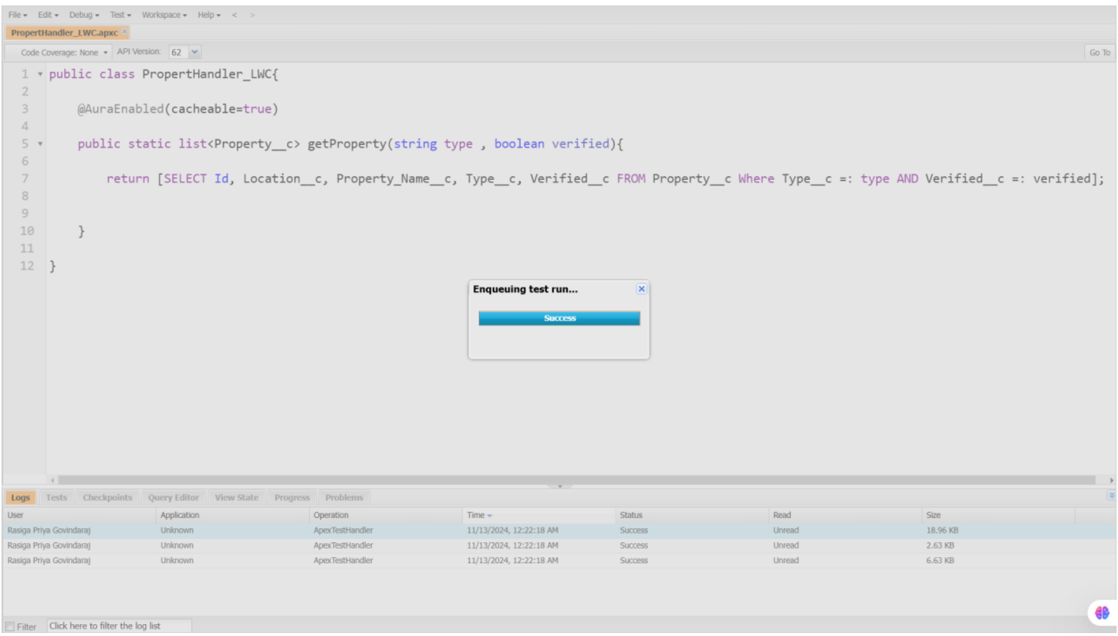
2. From Profiles Add “Manager” and “Customer” and “Save”

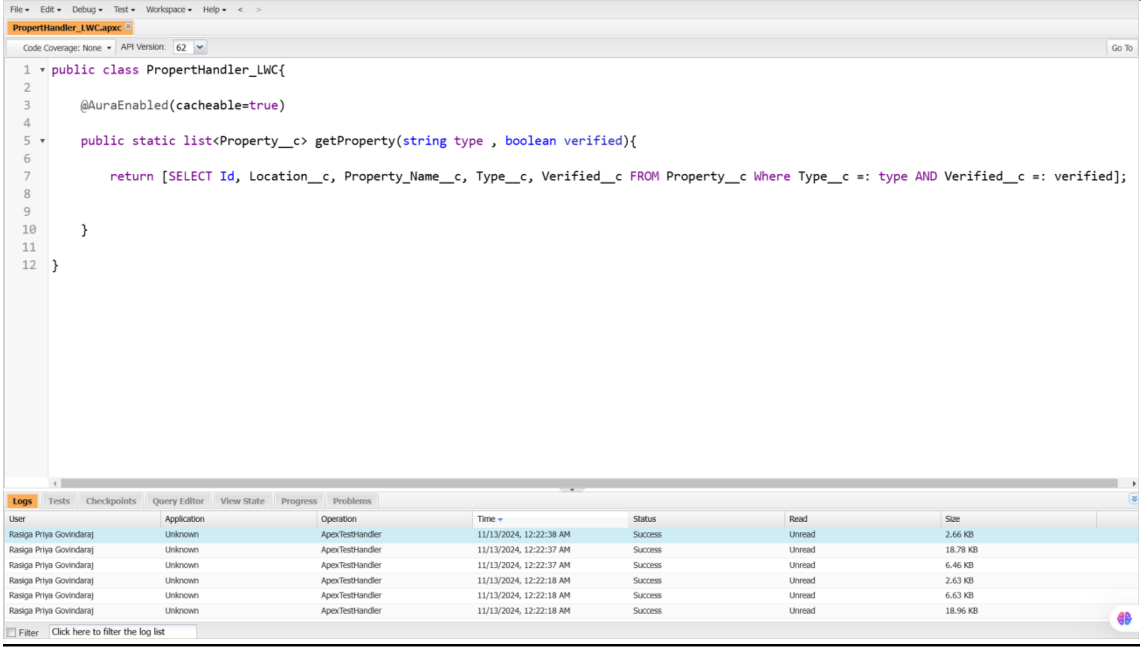


5. TESTING AND VALIDATION

1. <u>Unit testing(apex classes, Triggers)</u>

Unit testing in Salesforce with Aura involves verifying the functionality of individual Lightning components and their JavaScript controllers, helpers, and handlers. Using the **Lightning Testing Service (LTS)** and **Jest**, developers can create test cases to simulate user actions, test component logic, and verify data flows between the client and server-side controllers. Testing in Aura also helps catch errors early, especially when components interact with Apex or other Lightning components.



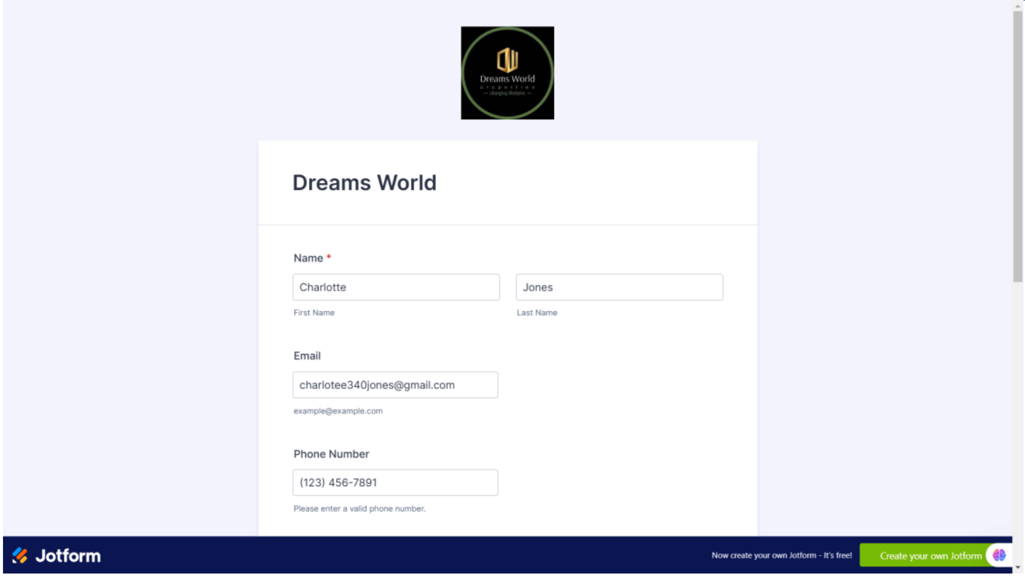


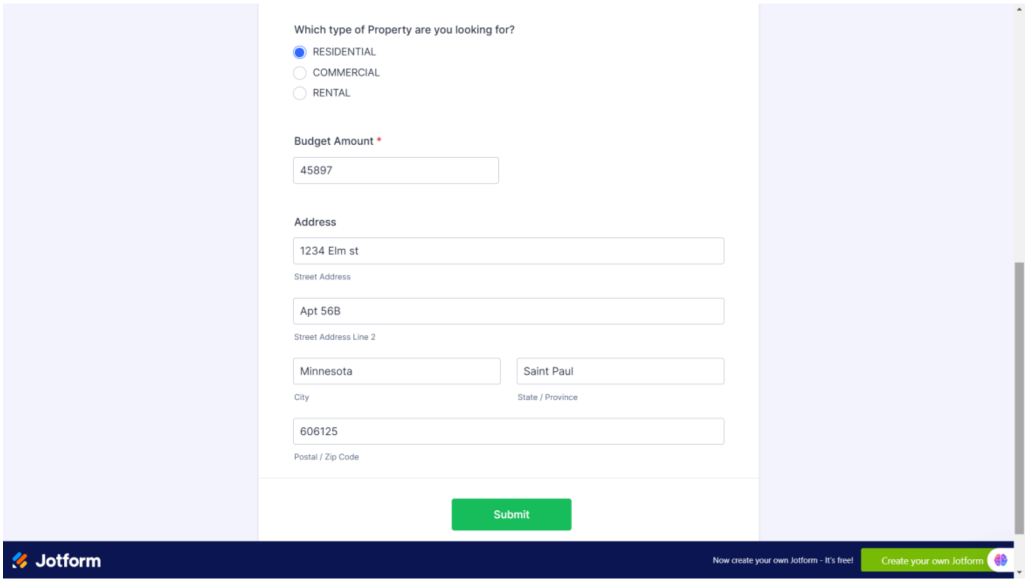
2. User Interface Testing

UI testing for a JotForm integrated with Salesforce CRM ensures the form is visually consistent across devices, validates input fields for correctness, and verifies that submitted data maps accurately to Salesforce fields. It also checks error handling for invalid inputs and tests end-to-end functionality to confirm smooth data flow from the form to Salesforce, ensuring proper synchronization and triggering of workflows. This process ensures a user-friendly experience and reliable data integration.

Here are the steps for testing the process where a JotForm is submitted, data is visible in the Property Detail tab in Salesforce, and also available on the JotForm website:

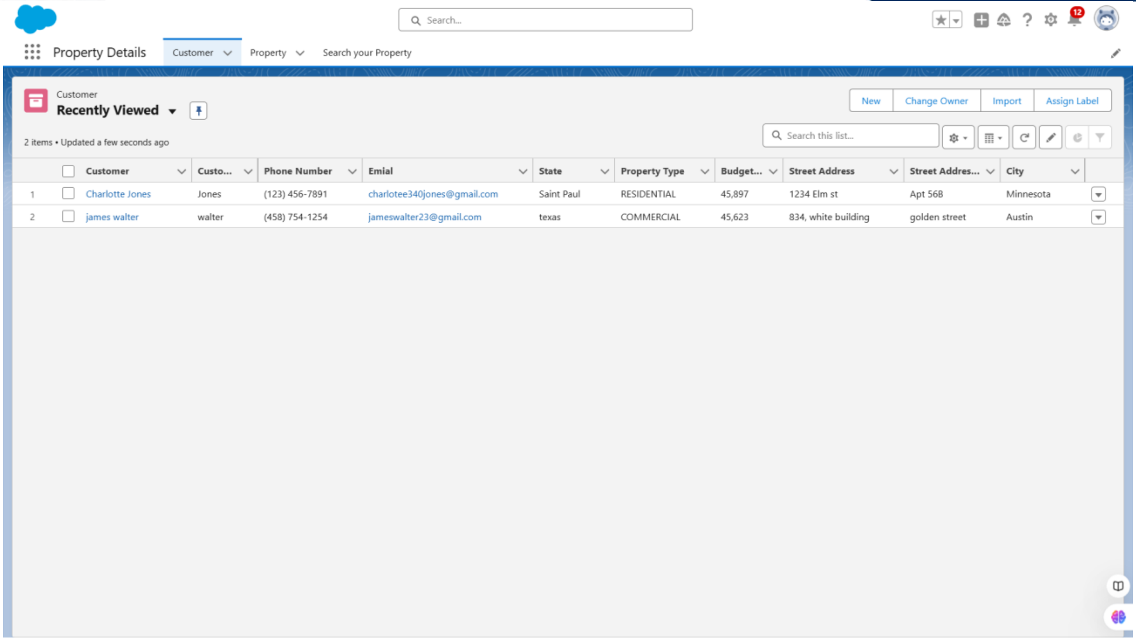
1. **Form Submission**:
   - A user submits the JotForm, filling in the relevant information.
   - Ensure that all form fields (such as property details) are correctly filled out and validated before submission.

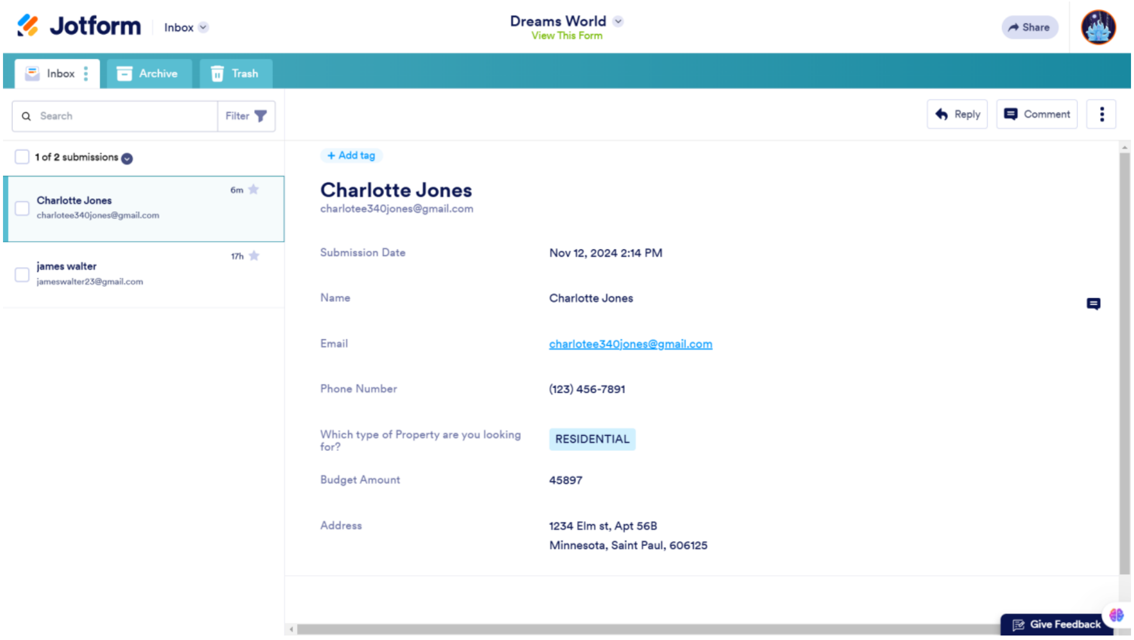




Data Sync to Salesforce:

- Verify that upon submission, the data is transferred from JotForm to Salesforce.
- Check that the information appears in the correct Property Detail tab within Salesforce, ensuring that each field maps to the appropriate Salesforce CRM fields.
- Confirm that any workflows, such as notifications or updates, are triggered based on the form submission


Search...
Property Details
Customer
Property
Search your Property
Customer
Recently Viewed
New
Change Owner
Import
Assign Label
2 items • Updated a few seconds ago
Search this list...
Customer | Custo... | Phone Number | Emial | State | Property Type | Budget... | Street Address | Street Addres... | City
1 Charlotte Jones | Jones | (123) 456-7891 | charlotee340jones@gmail.com | Saint Paul | RESIDENTIAL | 45,897 | 1234 Elm st | Apt 56B | Minnesota
2 james walter | walter | (458) 754-1254 | jameswalter23@gmail.com | texas | COMMERCIAL | 45,623 | 834, white building | golden street | Austin


Jotform
Inbox
Dreams World
View This Form
Share
Inbox
Archive
Trash
Search
Filter
Reply
Comment
1 of 2 submissions
Charlotte Jones
charlotee340jones@gmail.com
6m
james walter
jameswalter23@gmail.com
17h
+ Add tag
Charlotte Jones
charlotee340jones@gmail.com
Submission Date
Nov 12, 2024 2:14 PM
Name
Charlotte Jones
Email
charlotee340jones@gmail.com
Phone Number
(123) 456-7891
Which type of Property are you looking for?
RESIDENTIAL
Budget Amount
45897
Address
1234 Elm st, Apt 56B
Minnesota, Saint Paul, 606125
Give Feedback

6. CONCLUSION

In conclusion, our project for Dreams World Properties successfully integrates Salesforce to enhance and streamline customer interactions. By linking website engagement with Salesforce's CRM capabilities, customer details and preferences are automatically captured, creating a comprehensive record of each client. This integration categorizes users into approved and non-approved groups, allowing approved users to receive tailored property recommendations, enhancing their experience with relevant and expanded listings.

The seamless data flow between systems optimizes operations, improves customer engagement, and provides a more efficient and personalized user journey. This solution ultimately positions Dreams World Properties for growth, making them more competitive and responsive in the real estate market.