

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

## **TEAM MEMBERS:**

- K Raghu Prasanth
- T Mathanesh
- S Arun Kumar
- A Surya Daniel

## **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.

## **Objectives**

### **Business Goals:**

- **Improve Client Management:** Maintain a comprehensive record of client information, preferences, and interactions.
- **Enhance Customer Satisfaction:** Provide personalized service by quickly addressing client property requirements.
- **Streamline Property Management:** Efficiently handle property listings, viewings, and transactions.
- **Boost Sales Efficiency:** Enable sales teams to track and manage leads and opportunities more effectively.
- **Data-Driven Insights:** Generate reports and insights to understand market trends and client needs.

### **Specific Outcomes:**

- **Faster Client Response Times:** Reduced response times to client inquiries.
- **Increased Conversion Rates:** More effective lead tracking and follow-ups.
- **Improved Customer Retention:** Enhanced relationships with personalized interactions.
- **Increased Productivity:** Sales and service teams spend less time on manual data entry and more on client engagement.
- **Better Market Understanding:** Enhanced analytics for market trends and demand prediction.

## Salesforce Key Features and Concepts Utilized:

- **Sales Cloud:** Manage leads, opportunities, and accounts.
- **Service Cloud:** Track customer inquiries and provide support.
- **Property Management Objects:** Custom objects for properties, listings, and client requirements.
- **Reports and Dashboards:** Visualize key metrics for property and client performance.
- **Mobile Accessibility:** Manage CRM from any device for on-the-go updates.
- **Automation and Workflow Rules:** Automate notifications and follow-up reminders.

## Detailed Steps to Solution Design

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

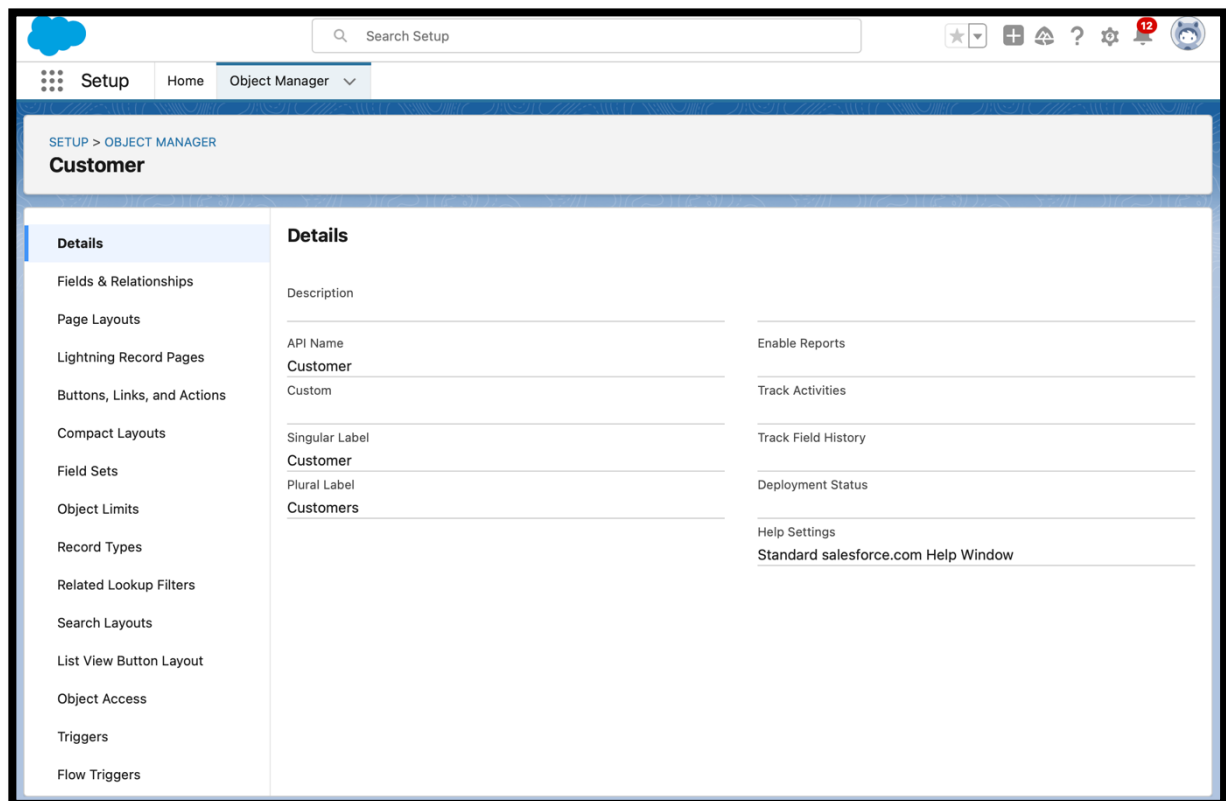
- Open your browser and search for jotform and log in.
- After login click on create form and click on start from scratch
- Now create a form to get the customer details like Name, Phone, Email, Address and type of property the customer is interested in.
- Once the form is created, publish it by clicking on publish.

The screenshot displays the Jotform Form Builder interface. The top navigation bar includes the Jotform logo, a 'Form Builder' dropdown, and a 'Form' title. Below this is an orange bar with 'BUILD', 'SETTINGS', and 'PUBLISH' tabs, and a 'Preview Form' toggle. The left sidebar contains various sharing and integration options: QUICK SHARE, EMBED, ASSIGN FORM, EMAIL, PREFILL, PDF, and PLATFORMS. The main content area shows the 'DIRECT LINK OF YOUR FORM' section, which includes a 'SHARE WITH LINK' button, a public form status, and a URL: <https://form.jotform.com/243101034249040>. Below this are buttons for 'COPY LINK' and 'OPEN IN NEW TAB'. There is also an 'INVITE BY EMAIL' section with a text input field for email addresses. At the bottom, there is a 'SHARE FORM' section with social media icons for Facebook, WhatsApp, X, LinkedIn, and a QR code, along with a '+ View more' link. A 'Give Feedback' button is located in the bottom right corner.

## Create Objects from Spreadsheet.

### Create Customer Object

- Go to your object manager and click on create object from spreadsheet.
- Click on the link to get the spreadsheet
- customer
- After downloading, upload the file, map the fields and upload to create an object.



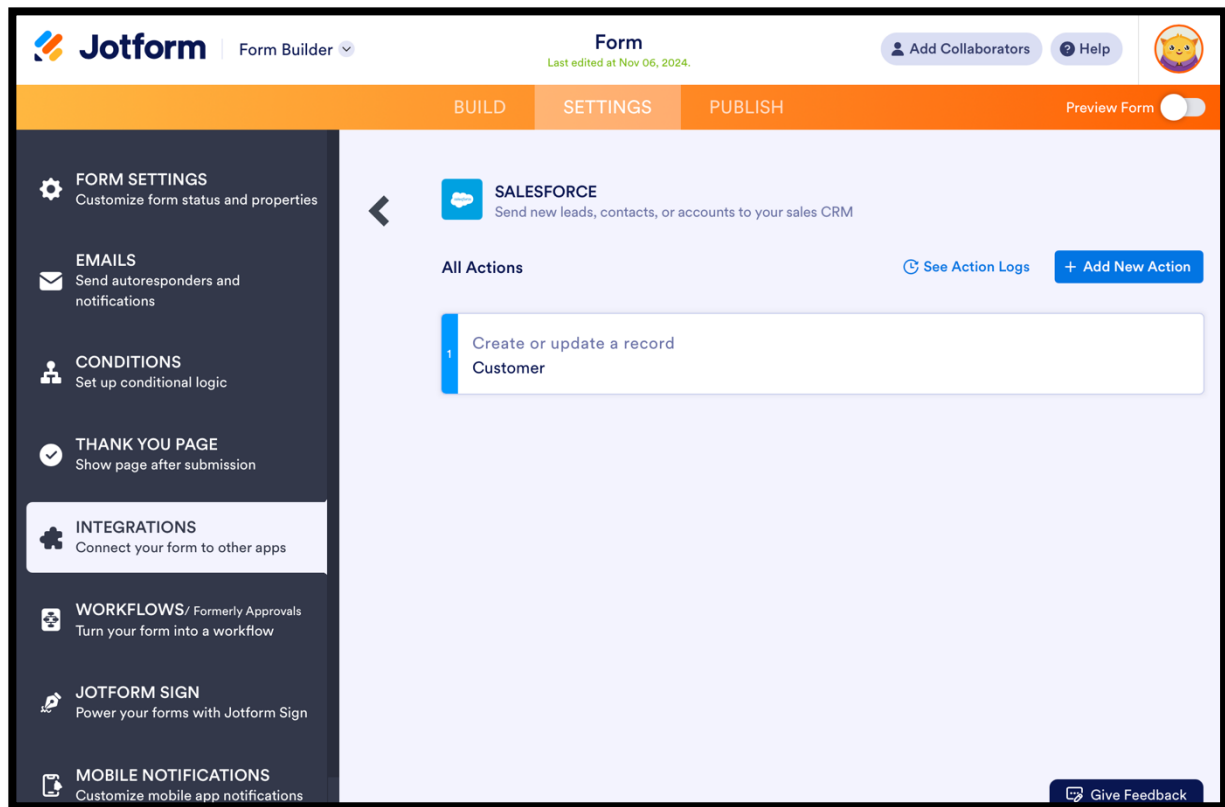
## Create Property Object

- Follow the same from the customer object to create the Property Object
- Property

The screenshot shows the Salesforce Object Manager interface for the 'Property12' object. The breadcrumb trail at the top reads 'SETUP > OBJECT MANAGER'. The page title is 'Property12'. On the left, a sidebar lists various configuration options: Details (selected), Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Restriction Rules, Scoping Rules, Object Access, Triggers, and Flow Triggers. The main content area is titled 'Details' and contains several fields: Description (empty), API Name (Property12\_\_c), Custom (checked), Singular Label (Property12), and Plural Label (Properties). On the right side of the main content area, there are checkboxes for 'Enable Reports', 'Track Activities', 'Track Field History', and 'Deployment Status' (which is set to 'Deployed'). At the bottom right, there are links for 'Help Settings' and 'Standard salesforce.com Help Window'. In the top right corner of the main content area, there are 'Edit' and 'Delete' buttons.

## Integrate Jotform With Salesforce Platform

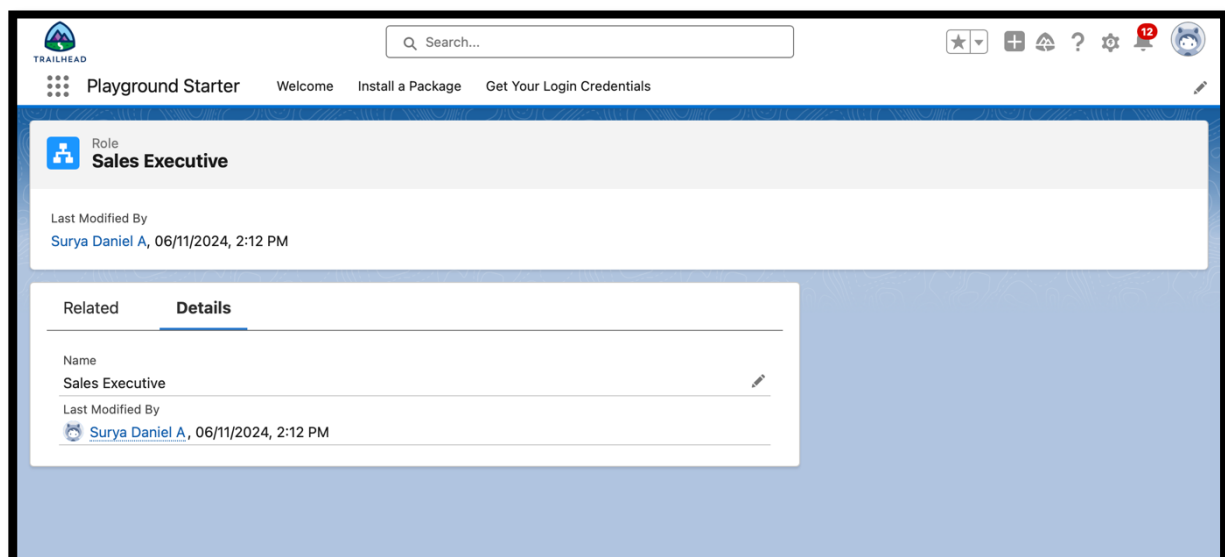
- On the Jotform Platform, Click on Integration and choose Salesforce.
- Click on User Integration and choose “Add to From”.
- Select the Org with which you want to Integrate your jotform with.
- Select an Action - Create a record.
- Select a Salesforce Object : - Customer
- Map Each and every field on the Object with the fields on the form and “Save Action”.
- Then “Save the Integration” and “Finish”.



## Create Roles

### Sales Executive Role

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

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

The screenshot shows the 'App Details & Branding' configuration page in the Lightning App Builder. The left sidebar lists 'App Settings' with sub-items: 'App Details & Branding' (selected), 'App Options', 'Utility Items (Desktop Only)', 'Navigation Items', and 'User Profiles'. The main content area is titled 'App Details & Branding' and includes a subtitle: 'Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.'

**App Details**

- \* App Name**: Property Details
- \* Developer Name**: Property\_Details
- Description**: Enter a description...

**App Branding**

- Image**: Upload button
- Primary Color Hex Value**: #0070D2
- Org Theme Options**: ☐ Use the app's image and color instead of the org's custom theme

**App Launcher Preview**

PD Property Details

## Create Profiles

- From Setup >> Go to Profiles and Clone Salesforce Platform User and Name it “Customer”..
- Uncheck all the Custom Objects and Check only Property Details From Custom App Settings.
- Also Remove all the Standard Object Permissions.
- Uncheck all the Custom Object Permissions and check read and view all in “Property”

The screenshot shows the Salesforce 'Profile Edit' page for a profile named 'Customer'. The page is titled 'Profile Edit' and 'Customer'. Below the title, it says 'Set the permissions and page layouts for this profile.' There are buttons for 'Save', 'Save & New', and 'Cancel'. The 'Name' field is 'Customer', 'User License' is 'Salesforce Platform', and 'Custom Profile' is checked. The 'Description' field is empty. Below this, there are sections for 'Custom App Settings', 'Connected App Access', 'Service Provider Access', and 'Tab Settings'. The 'Custom App Settings' section has a table with columns for 'Visible' and 'Default' for various apps. The 'Connected App Access' section has a table with columns for 'Visible' and 'Default' for various connected apps. The 'Service Provider Access' section is empty. The 'Tab Settings' section is empty.

**Profile Edit** [Save] [Save & New] [Cancel]

Name:

User License: Salesforce Platform

Description:

Custom Profile: ☒

**Custom App Settings** ⓘ Required Information

	Visible	Default		Visible	Default
Analytics Studio (standard__Insights)	<input type="checkbox"/>	<input type="radio"/>	Playground Starter (trihdtpa__Playground_Starter)	<input type="checkbox"/>	<input type="radio"/>
App Launcher (standard__AppLauncher)	<input type="checkbox"/>	<input type="radio"/>	Property Details (Property_Details)	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>
Platform (standard__Platform)	<input type="checkbox"/>	<input type="radio"/>	WDC (standard__Work)	<input type="checkbox"/>	<input type="radio"/>

**Connected App Access**

	Visible	Default
Ant Migration Tool	<input type="checkbox"/>	
Dataloader Bulk	<input type="checkbox"/>	
Dataloader Partner	<input type="checkbox"/>	
Force.com IDE	<input type="checkbox"/>	
Salesforce for Outlook	<input type="checkbox"/>	
Salesforce Mobile Dashboards	<input type="checkbox"/>	
Salesforce Touch	<input type="checkbox"/>	
Workbench	<input type="checkbox"/>	

**Service Provider Access**

**Tab Settings**

## Manager

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

The screenshot shows the Salesforce 'Profile Edit' page for a profile named 'Manager'. The page is titled 'Profile Edit Manager' and includes a sub-header 'Set the permissions and page layouts for this profile.' Below this, there are buttons for 'Save', 'Save & New', and 'Cancel'. The 'Name' field is set to 'Manager', the 'User License' is 'Salesforce Platform', and the 'Custom Profile' checkbox is checked. The 'Description' field is empty. The page is divided into several sections: 'Custom App Settings', 'Connected App Access', 'Service Provider Access', and 'Tab Settings'. The 'Custom App Settings' section contains two tables of permissions. The first table lists standard objects with 'Visible' and 'Default' checkboxes. The second table lists custom objects with 'Visible' and 'Default' checkboxes. The 'Connected App Access' section lists various Salesforce integrations with checkboxes. The 'Service Provider Access' section is currently empty. The 'Tab Settings' section is also empty. The left sidebar contains navigation links for 'Home', 'Administer', 'Release Updates', 'Manage Users', 'Profiles', 'Public Groups', 'Queues', 'Login History', 'Identity Provider Event Log', and 'Identity Verification History'. The top of the page features a banner for 'It's Better in Lightning' and a search bar.

**Profile Edit**  
**Manager**  
Set the permissions and page layouts for this profile.

**Profile Edit** [Save] [Save & New] [Cancel]

Name:   
User License: Salesforce Platform  
Description:   
Custom Profile: ☒

**Custom App Settings** ⓘ Required Information

	Visible	Default		Visible	Default
Analytics Studio (standard__Insights)	<input type="checkbox"/>	<input type="radio"/>	Playground Starter (trihdtips__Playground_Starter)	<input type="checkbox"/>	<input type="radio"/>
App Launcher (standard__AppLauncher)	<input type="checkbox"/>	<input type="radio"/>	Property Details (Property_Details)	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>
Platform (standard__Platform)	<input type="checkbox"/>	<input type="radio"/>	WDC (standard__Work)	<input type="checkbox"/>	<input type="radio"/>

**Connected App Access**

Ant Migration Tool	<input type="checkbox"/>	Salesforce for Outlook	<input type="checkbox"/>
Dataloader Bulk	<input type="checkbox"/>	Salesforce Mobile Dashboards	<input type="checkbox"/>
Dataloader Partner	<input type="checkbox"/>	Salesforce Touch	<input type="checkbox"/>
Force.com IDE	<input type="checkbox"/>	Workbench	<input type="checkbox"/>

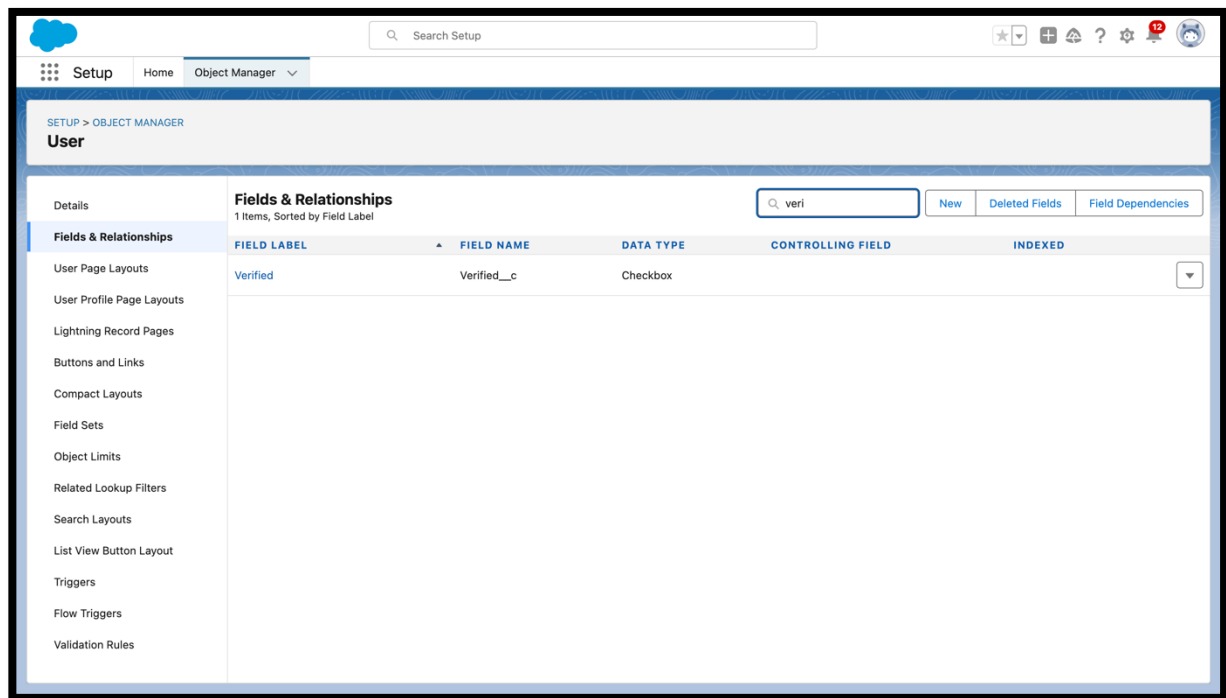
**Service Provider Access**

**Tab Settings**



## Create A Check Box Field On User

- Setup >> Object Manager >> Search for User >> Fields and Relationships
- Create new Field Named as “Verified” as Data type “Check Box”



## Create Users

### User 1

- Go to Setup --> Administration --> Users --> New User
- Last Name - Executive
- Role - Sales Executive
- License - Salesforce
- Profile - System Administrator
- Save

### User 2

- Go to Setup >> Administration >> Users >> New User
- Last Name >> Manager
- Role >> Sales Manager
- License >> Salesforce Platform
- Profile >> Manager

- Save

### User 3

- Go to Setup >> Administration >> Users >> New User
- Last Name >> Customer
- Role >> Customer
- License >> Salesforce Platform
- Profile >> Customer
- Make Sure the verified check box is “Unchecked”
- Save

### User 4

- Go to Setup >> Administration >> Users >> New User
- Last Name >> Customer2
- Role >> Customer
- License >> Salesforce Platform
- Profile >> Customer
- Make Sure the verified check box is “checked”
- Save

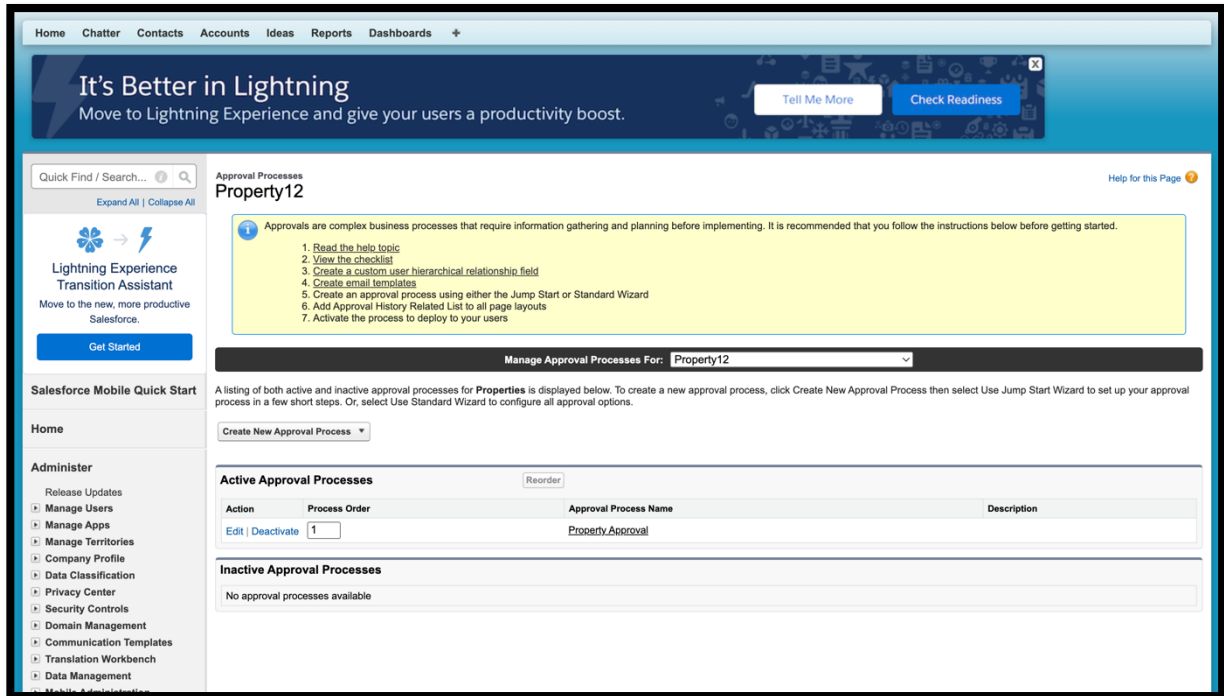
The screenshot displays the Salesforce 'All Users' page. The top navigation bar includes links for Home, Chatter, Contacts, Accounts, Ideas, Reports, and Dashboards. A banner at the top promotes 'It's Better in Lightning' with a 'Check Readiness' button. The left sidebar contains a 'Quick Find / Search...' bar and a 'Lightning Experience Transition Assistant' section. The main content area is titled 'All Users' and includes a 'View: All Users' dropdown and a 'Create New View' link. Below this is a table of users with columns for Action, Full Name, Alias, Username, Role, Active, and Profile. The table lists several users, including 'Surya Daniel' (SA), 'Chatter Expert', 'Customer', 'Customer2', 'Executive', 'Manager', 'User Integration', and 'User Security'. The 'Active' column has checkboxes, and the 'Profile' column lists various roles. The bottom of the page shows a 'New User' button and a 'Reset Password(s)' button.

Action	Full Name	Alias	Username	Role	Active	Profile
<input type="checkbox"/> Edit	A. Surya Daniel	SA	asuryadaniel21@creative-bear-br2crv.com		<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/> Edit	Chatter Expert	Chatter	chatty.00dns000005ohb72ag.fhupf9mymzo@chatter.salesforce.com		<input checked="" type="checkbox"/>	Chatter Free User
<input type="checkbox"/> Edit	Customer	cust	cust123kqcd56@gmail.com	Customer	<input checked="" type="checkbox"/>	Customer
<input type="checkbox"/> Edit	Customer2	cust	cust245drfg@gmail.com	Customer	<input checked="" type="checkbox"/>	Customer
<input type="checkbox"/> Edit	Executive	exec	exejhghg@gmail.com	Sales Executive	<input checked="" type="checkbox"/>	System Administrator
<input type="checkbox"/> Edit	Manager	mana	manager178u23@gmail.com	Sales Manager	<input checked="" type="checkbox"/>	Manager
<input type="checkbox"/> Edit	User Integration	integ	integration@00dns000005ohb72ag.com		<input checked="" type="checkbox"/>	Analytics Cloud Integration User
<input type="checkbox"/> Edit	User Security	sec	insightssecurity@00dns000005ohb72ag.com		<input checked="" type="checkbox"/>	Analytics Cloud Security User

## Create An Approval Process For Property Object

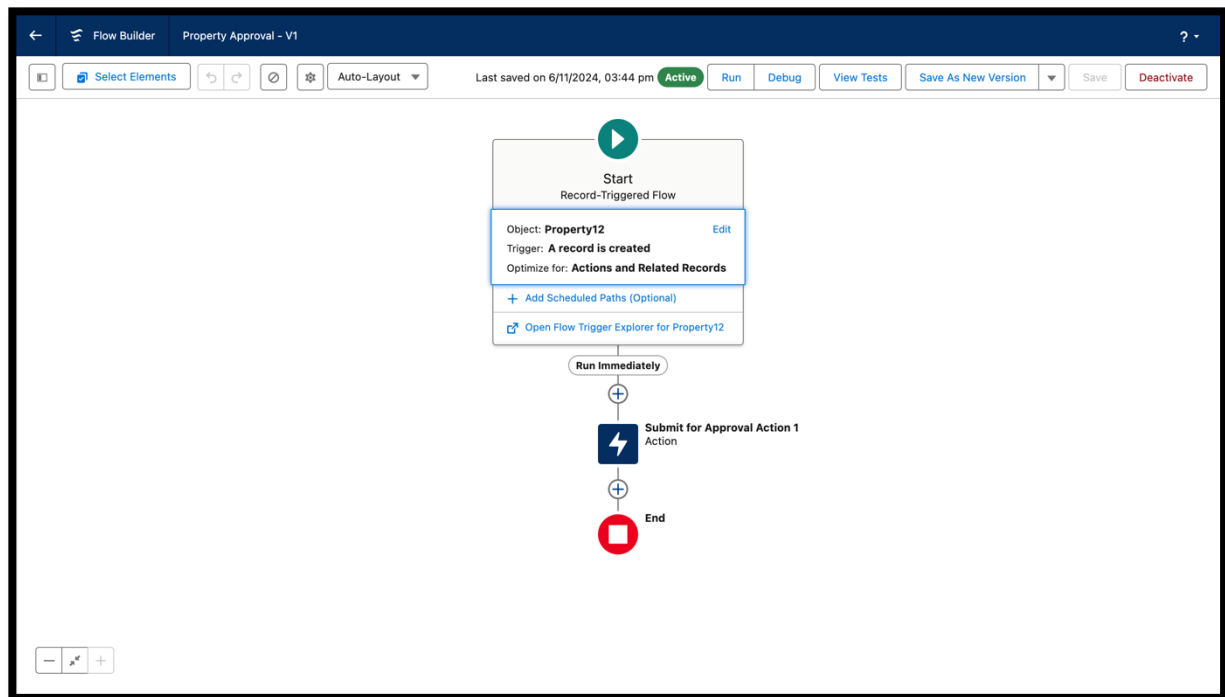
- From Setup >> Process Automation >> Approval Process
- Process Name - Property Approval
- Give 2 criteria –
- Location is not equal to blank,
- Verified Equals false.
- Click next and “Next Automated Approver Determined By” Select Manager
- From Record Editability Properties >> Click on Administrators OR the currently assigned approver can edit records during the approval process.
- From Step 5. Select Fields to Display on Approval Page Layout select Property, Owner, Location, Type.
- Click Next and Select the initial Submitters >>
- Owner >> Property Owner
- Roles >> Sales Manager
- Save.
- Add an approval step name “Executive Approval ”
- specify the Criteria >> All record should enter
- click next and select the Approver as “ Sales Executive “ and “Save”
- Add One field Update as “Verified Property”
- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as “True”
- Save
- Add One field Update as “UnVerified Property”
- Select Object >> Property
- Field to Update >> Verified
- Field Data Type >> CheckBox

- Select CheckBox Option as “False”
- Save.
- Activate the Approval Process.



## Create a Record Trigger Flow To Submit The Approval process automatically

- From Setup >> Search for Flows >> Click On New and Select “Record Trigger Flow”.
- Select Object >> Property
- Select “Trigger the flow when” >> “A record is created”
- Set Entry Conditions >> “None”
- Add a “Action” >> “Submit for Approval”
- Give Label >> Approval for property
- Record Id >> {!\$Record.Id}
- Done
- Save the Flow and Give label as “Property Approval” and “Activate”



## Create An App Page

- From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and Click on Next.
- Give Label as “Search your Property” click “Next”.
- Click “header and Left Sidebar” and Click on “Done”
- Click on “Save ” and then click on “Activate”.
- From Page Setting select page activation as “Activate for all Users”.
- From Lightning Experience Click on “Property Details” and click on Add Page“.
- Then Click on “Save”

**Lightning App Builder** Help for this Page ?

The Lightning App Builder provides an easy to use graphical interface for creating custom Lightning pages for Salesforce Lightning Experience and mobile app. Lightning pages are built using Lightning components—compact, configurable, and reusable elements that you can drag and drop into regions of the page in the Lightning App Builder.

View: All 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**

Lightning Pages							
Action	Label ↑	Name	Namespace Prefix	Description	Type	Created By	Last Modified By
<a href="#">Edit</a>   <a href="#">Clone</a>   <a href="#">Del</a>	<a href="#">Search your Property</a>	Search_your_Property			App Page	SA, 06/11/2024, 3:46 pm	SA, 06/11/2024, 3:46 pm

## Create A LWC Component

- 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<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];

    }

}
```



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

Code :-

<template>

<lightning-card>

```

<div class="slds-box">

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

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

  </div>

  <div>

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

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

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

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

      </div>

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

        <br>

        <lightning-button-icon    variant="neutral"    icon-name="standard:search"    alternative-
text="Search"

        label="Search" onclick={handleClick}></lightning-button-icon>

      </div>

    </div>

  </div>

</div>

<template if:true={istru}>

  <div class="slds-box">

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

  </div>

</template>

<template if:false={isfalse}>

  <div class="slds-box">

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

  </div>

```

</template>

</lightning-card>

</template>

- **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)

})

}

}

```

- 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>

```

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

Drag this Component to Your App page

- From Setup >> Go to App Launcher >> Search for Property Details

