

Project Report

A CRM Application to Handle Client and Property-Related Requirements

PAMULA NAVEEN KUMAR

21501A05D3@pvpsit.ac.in

Table of Contents

Abstract	3
Introduction	3
Scope	4
Objectives	4
Procedure	5

Technical Implementation	19
Conclusion	21
Future Scope	22
Output	22

Abstract

The "**Estate Masters** " project integrates Salesforce CRM to create a robust and efficient property management solution for the real estate market. By automating customer interaction workflows, the application captures user details, processes their requests, and provides tailored property recommendations. It categorizes users into approved and non-approved groups, ensuring a personalized user experience. This project exemplifies how a Salesforce-powered CRM system can transform operations, improve engagement, and drive growth in real estate.

Introduction

The real estate sector faces challenges in managing client relationships and meeting customer expectations. With numerous inquiries and properties, it becomes essential to have an automated system that handles customer requirements while providing accurate property matches. Estate Masters Time addresses these challenges by integrating Jotform and Salesforce, ensuring seamless interaction between customers and businesses.

This project simplifies client management, property approvals, and data organization, offering real-time property recommendations. The solution also incorporates role-specific access controls to maintain data integrity and secure operations.

Scope

1. Automate the collection of client data.
2. Provide tailored property recommendations based on client preferences.
3. Streamline the approval process for property listings.
4. Offer role-based data access for security and efficiency.
5. Enhance user satisfaction with a clean, interactive interface.

Objectives

The primary goals of this project include:

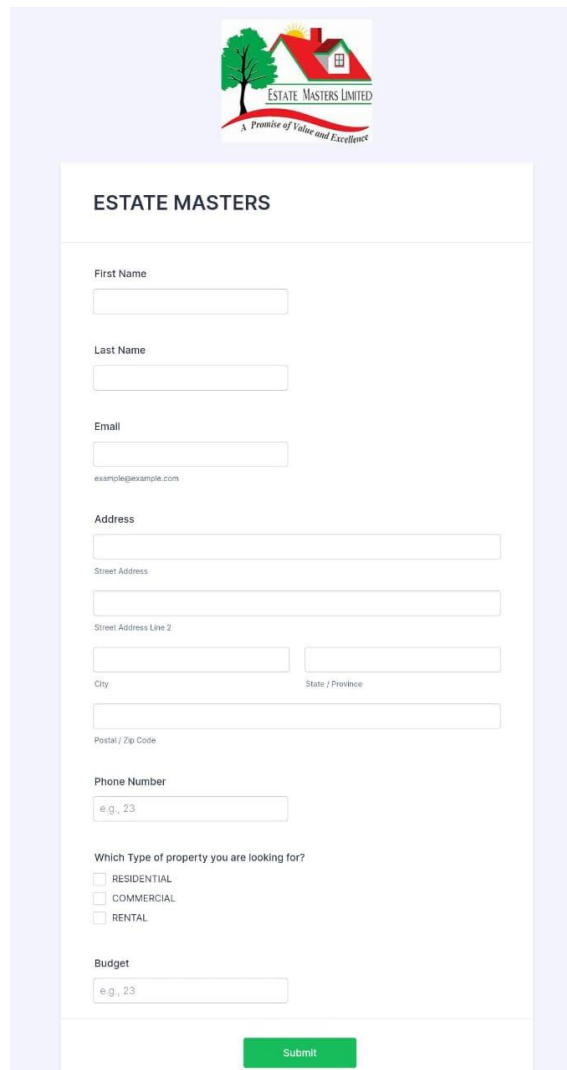
1. **Automation:** Minimize manual intervention by automating client data collection and property recommendation processes.
2. **Efficiency:** Organize client data into actionable categories for better business insights.
3. **User Experience:** Deliver a personalized, user-friendly platform that caters to customer needs.
4. **Security:** Implement role-based access to ensure only authorized personnel can manage sensitive information.


5. **Scalability:** Design the solution to support future expansion and additional features.

Procedure

Step 1: Data Collection and Form Integration

1. **Tool:** Jotform
2. **Implementation:**
 - a. Designed a customer-facing web form for collecting user data such as name, contact details, and property preferences.





ESTATE MASTERS

First Name

Last Name

Email

example@sample.com

Address

Street Address

Street Address Line 2

City
State / Province

Postal / Zip Code

Phone Number

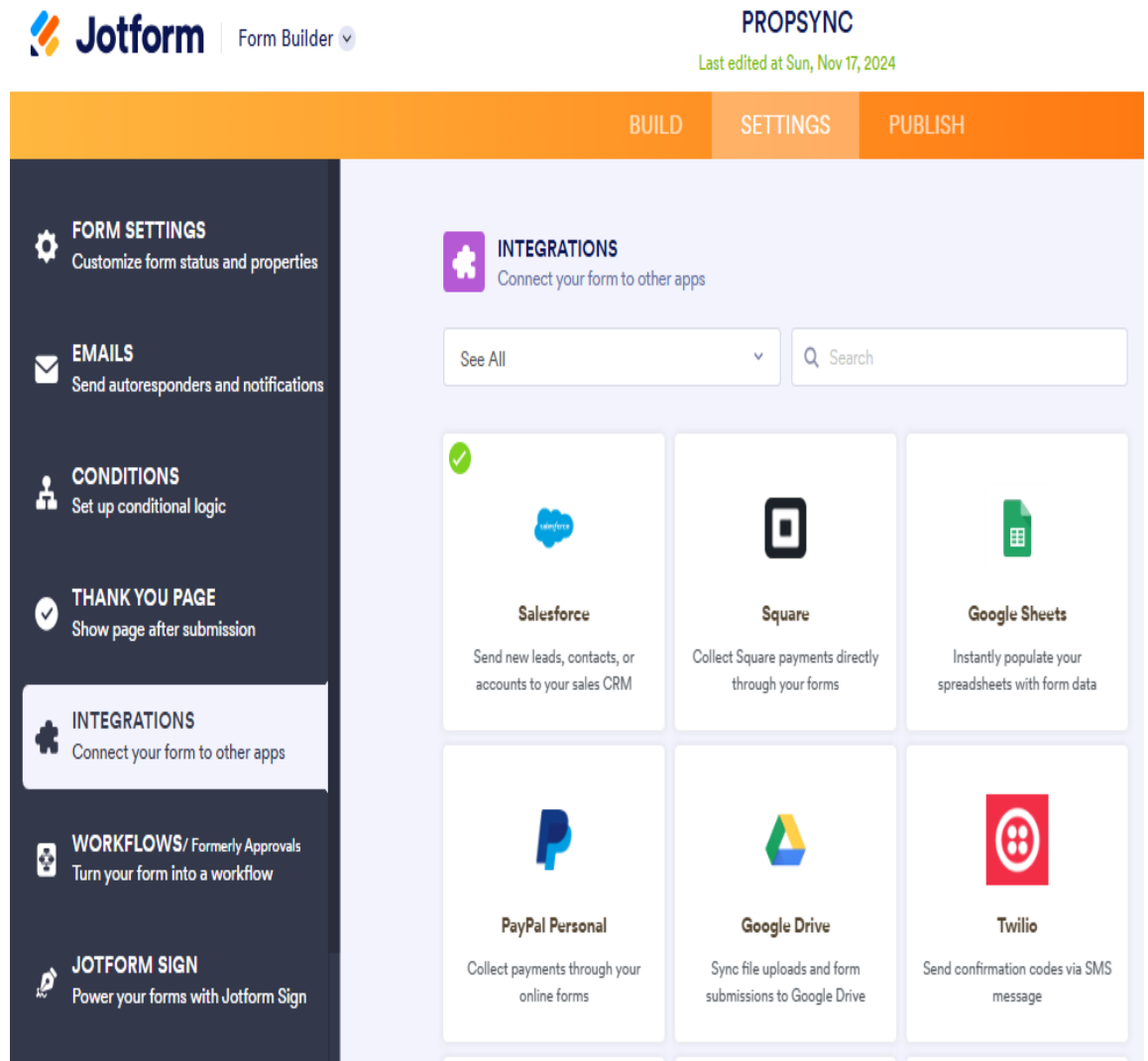
e.g., 23

Which Type of property you are looking for?
☐ RESIDENTIAL
☐ COMMERCIAL
☐ RENTAL

Budget

e.g., 23

- a. Published the form and integrated it with Salesforce for automatic record creation.



- a. Jotform fields were mapped to Salesforce fields to maintain data_integrity.

PROPSYNC
Last edited yesterday.

BUILD **SETTINGS** **PUBLISH**

Find the record that matches with the selected fields

Object Fields	PROPSYNC
Customer__c	Name - First Name
State	Address - State
City	Address - City
Property Type	Which type of Property are you lookin...
Street Address	Address - Street Address
Street Address line 2	Address - Street Address 2
Name	Name - Last Name
postal code	Address - Postal/Zip Code
Emial	Email
Budget Amount	Budget Amount
Phone Number	Phone Number

+ Add Field

Create a record

OFF

1. **Result:** Automated and accurate collection of customer details.

SALESFORCE
Send new leads, contacts, or accounts to your sales CRM

All Actions [See Action Logs](#) [+ Add New Action](#)

1 Find existing record
Customer

Step 2: Custom Object Creation

- 1. **Customer Object:** Captures client information such as contact details and preferences.

Customer

Custom Object Definition Edit

SaveSave & NewCancel

Custom Object Information

The singular and plural labels are used in tabs, page layouts, and reports.
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label

Customer

Example: Account

Plural Label

Customer

Example: Accounts

Starts with vowel sound

☐

The Object Name is used when referencing the object via the API.

Object Name

Customer

Example: Account

Description

Fields & Relationships

15 Items, Sorted by Field Label

FIELD LABEL	FIELD NAME	DATA TYPE
Budget Amount	Budget_Amount__c	Number(18, 0)
City	City__c	Text(255)
Created By	CreatedById	Lookup(User)
Customer	Customer__c	Text(255)
Customer	Name	Text(80)
Email	Email__c	Email
Last Modified By	LastModifiedById	Lookup(User)
Owner	OwnerId	Lookup(User,Group)
Phone Number	Phone_Number__c	Phone
postal code	postal_code__c	Text(255)
Property Type	Property_Type__c	Text(255)
State	State__c	Text(255)
Street Address	Street_Address__c	Text(255)
Street Address line 2	Street_Address_line_2__c	Text(255)
Verified	Verified__c	Checkbox

1. **Property Object:** Maintains property-specific details like location, type, and owner.

Edit Custom Object
Property

Custom Object Definition Edit

SaveSave & NewCancel

Custom Object Information

The singular and plural labels are used in tabs, page layouts, and reports.
Be careful when changing the name or label as it may affect existing integrations and merge templates.

Label

Property

Example: Account

Plural Label

Property

Example: Accounts

Starts with vowel sound

☐

The Object Name is used when referencing the object via the API.

Object Name

Property

Example: Account

Description

Context-Sensitive Help Setting

☒ Open the standard Salesforce.com Help & Training window
☐ Open a window using a Visualforce page

Content Name

—None—

Fields & Relationships

8 Items, Sorted by Field Label

Quick I

FIELD LABEL	FIELD NAME	DATA TYPE
Created By	CreatedById	Lookup(User)
Last Modified By	LastModifiedById	Lookup(User)
Location	Location__c	Text(255)
Owner	OwnerId	Lookup(User,Group)
Property	Name	Text(80)
Property Name	Property_Name__c	Text(255)
Type	Type__c	Text(255)
Verified	Verified__c	Checkbox

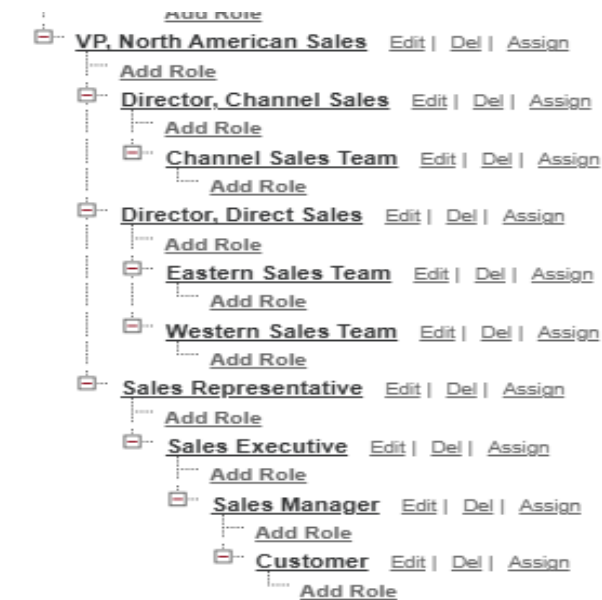
1. Methodology:

- Utilized the "Create Object from Spreadsheet" feature in Salesforce for efficient object creation.
- Mapped data fields between spreadsheets and Salesforce.

Step 3: Role and Profile Setup

1. Roles Created:

- Sales Executive: Handles property listings and interacts with potential customers.
- Sales Manager: Oversees sales executives and approves critical operations.
- Customer: End users who search for properties.



1. Profiles Configured:

- Restricted access to non-relevant objects and ensured each role had permissions tailored to their tasks.

Step 4: Property Approval Workflow

1. Designed an approval process to ensure all property listings are verified before being made public.

The screenshot shows the 'Approval Processes' setup page for 'Property Approval'. The page includes a 'Process Definition Detail' section with fields for Process Name, Unique Name, Description, Entry Criteria, Record Editability, Approval Assignment Email Template, Initial Submitters, and Created By. The 'Entry Criteria' field contains the condition: (Property: Location NOT EQUAL TO blank) AND (Property: Verified EQUALS false). The 'Initial Submitters' field contains the role: Sales Manager, Property Owner. The 'Created By' field shows Naveen Kumar, 23/11/2024, 2:56 am. The 'Modified By' field shows Naveen Kumar, 23/11/2024, 3:23 pm. Below the 'Process Definition Detail' section is the 'Initial Submission Actions' section, which includes a table with columns for Action, Type, and Description. The table contains one row: Record Lock, Lock the record from being edited.

Process Name	Property Approval	Active	✓
Unique Name	Property_Approval	Next Automated Approver Determined By	Manager of Record Submitter
Description			
Entry Criteria	(Property: Location NOT EQUAL TO blank) AND (Property: Verified EQUALS false)		
Record Editability	Administrator OR Current Approver	Allow Submitters to Recall Approval Requests	<input type="checkbox"/>
Approval Assignment Email Template	Group Service Appointments Enrollment Confirmation Email		
Initial Submitters	Role: Sales Manager, Property Owner		
Created By	Naveen Kumar, 23/11/2024, 2:56 am		
Modified By	Naveen Kumar, 23/11/2024, 3:23 pm		

Action	Type	Description
Record Lock		Lock the record from being edited

- a. Criteria for approval included:
 - i. Location field must not be empty.
 - ii. Property must be marked as "unverified."

The screenshot shows the 'Field Update Edit' form for 'Verified Property'. The form includes fields for Name, Unique Name, Description, Object, Field to Update, Field Data Type, and Re-evaluate Workflow Rules after Field Change. The 'Name' field contains 'Verified Property'. The 'Unique Name' field contains 'Verified_Property'. The 'Field to Update' field contains 'Property: Verified'. The 'Field Data Type' field contains 'Checkbox'. The 'Re-evaluate Workflow Rules after Field Change' checkbox is unchecked. Below the 'Field Update Edit' section is the 'Specify New Field Value' section, which includes a 'Checkbox Options' section with radio buttons for 'True' and 'False'. The 'True' radio button is selected.

Name	Verified Property
Unique Name	Verified_Property
Description	
Object	Property
Field to Update	Property: Verified
Field Data Type	Checkbox
Re-evaluate Workflow Rules after Field Change	<input type="checkbox"/>

Specify New Field Value

Checkbox Options

☒ True
☐ False

Field Update Edit

SaveSave & NewCancel

Identification

NameUnVerified Property

Unique NameUnVerified_Property

Description

ObjectProperty

Field to UpdateProperty: Verified

Field Data TypeCheckbox

Re-evaluate Workflow Rules after Field Change

Specify New Field Value

Checkbox Options

True

False

SaveSave & NewCancel

a. Approval hierarchy:

- 1.Sales Executive for initial review.
- 2.Sales Manager for final approval.

Step 5: Application Development

1. **Tool:** Salesforce Lightning App Builder
2. **App Name:** Property Details

New Lightning App

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name ⓘ
Search Your Property

* Developer Name ⓘ
Search_Your_Property

Description ⓘ
Enter a description...

App Branding

Image ⓘ
Upload

Primary Color Hex Value
#AAE420

Org Theme Options
☐ Use the app's image and color instead of the org's custom theme

App Launcher Preview

Next


1. Features:

- Consolidates customer and property objects.
- Provides a user-friendly interface for managing records.

Step 6: Automation Through Flows

1. Record Trigger Flow:

- Automatically submits property records for approval upon creation.
- Simplifies the approval process and reduces delays.

 **SETUP**
Flows

Flow
Property Approval

[Help for this Page](#)

[Back to List: Flows](#)

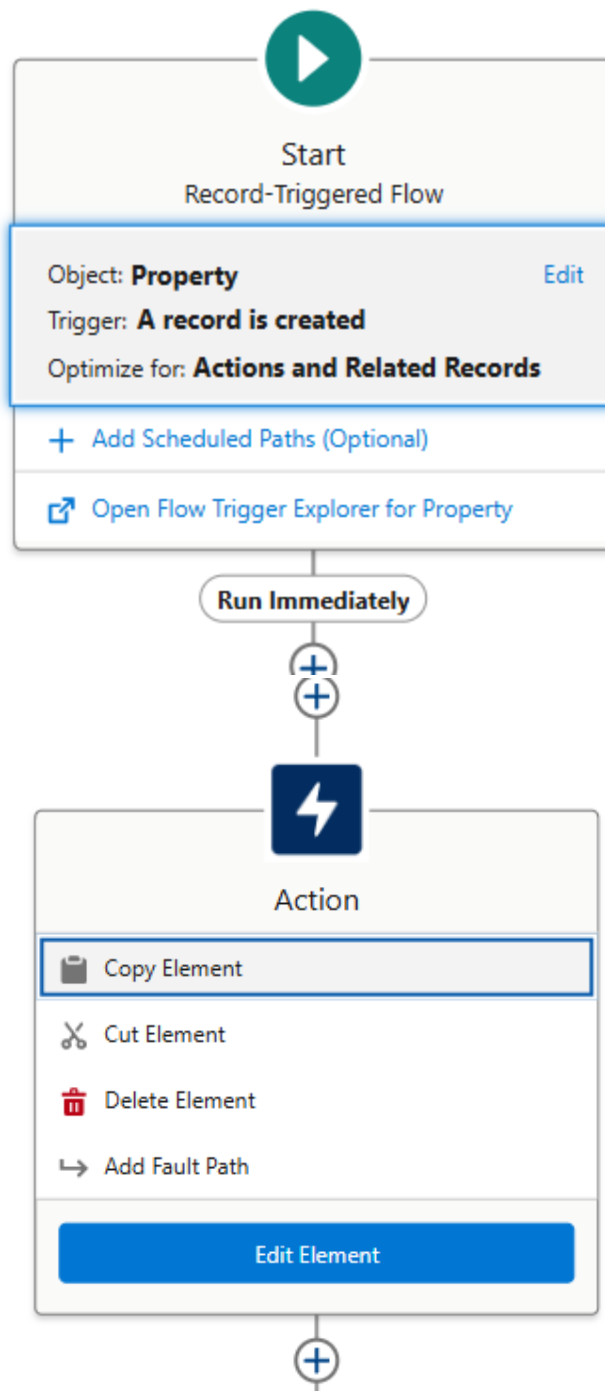
Flow Detail

[Edit](#) [Open](#) [Run](#) [Delete](#)

Flow Label	Property Approval	Flow API Name	Property_Approval
Description		Namespace Prefix	
Environments	Default	Type	Autolaunched Flow
Active Version	1	URL	/flow/Property_Approval
Trigger	Record—Run After Save	Activated/Deactivated By	Naveen Kumar: 23/11/2024, 3:37 pm
Modified By	Naveen Kumar: 23/11/2024, 7:34 pm	Created By	Naveen Kumar: 23/11/2024, 3:37 pm

Flow Versions

Action	Flow Label	Version	Description	Built with	Created Date	Type	Status	Progress Status	Run in Mode	API Version for Running the Flow
Open Run Deactivate	Property Approval	1		Flow Builder	23/11/2024, 3:37 pm	Autolaunched Flow	Active	Activated	Default Mode	62.0



Step 7: Lightning Web Component (LWC)

1. Purpose:

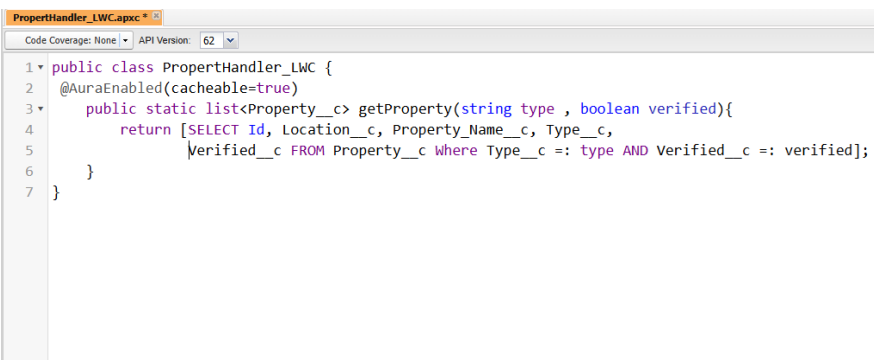
- a. Allows verified customers to access approved properties.
- b. Restricts non-verified customers to view only non-approved properties.

2. Implementation:

- a. Created an Apex class for querying verified properties.

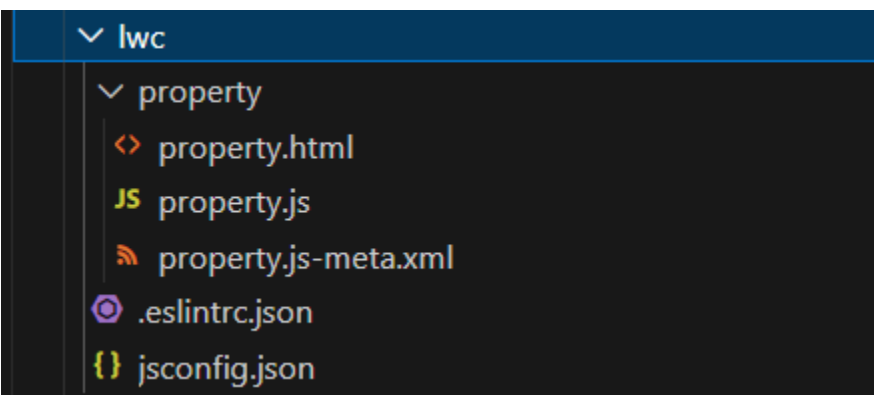
Developed an interactive LWC component with property filters.

a.

The screenshot shows the Salesforce IDE with an Apex class named 'PropertHandler_LWC.apex'. The code is as follows:

```
1 public class PropertHandler_LWC {  
2     @AuraEnabled(cacheable=true)  
3     public static list<Property__c> getProperty(string type , boolean verified){  
4         return [SELECT Id, Location__c, Property_Name__c, Type__c,  
5                 Verified__c FROM Property__c Where Type__c =: type AND Verified__c =: verified];  
6     }  
7 }
```

b.

The screenshot shows the file explorer in the Salesforce IDE. It displays a folder named 'lwc' which contains a sub-folder 'property'. Inside the 'property' folder, there are five files: 'property.html', 'property.js', 'property.js-meta.xml', '.eslintrc.json', and 'jsconfig.json'.

```

<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>
    <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>
    </div>
  </lightning-card>
</template>

```

C.

```

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';

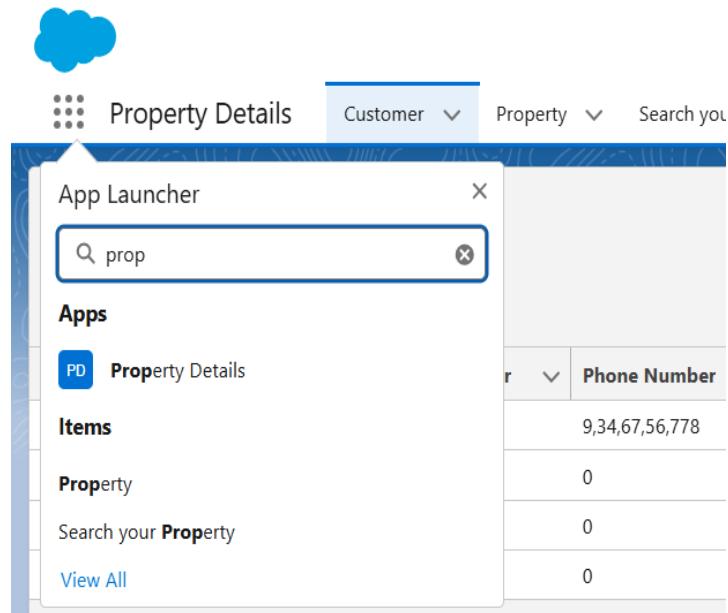
export default class PropSyncComponent extends LightningElement {
    @api recordId;
    userId = USER_ID;
    verifiedvar;
    typevar;
    isfalse = true;
    istrue = false;
    @track propertylist = [];
    columns = [
        { label: 'Property Name', fieldName: 'Property_Name__c' },
        { label: 'Property Type', fieldName: 'Type__c' },
        { label: 'Property Location', fieldName: 'Location__c' },
        { label: 'Property link', fieldName: 'Property_link__c' }
    ];
    propertyoptions = [
        { label: 'Commercial', value: 'Commercial' },
        { label: 'Residential', value: 'Residential' },
        { label: 'rental', value: 'rental' }
    ];
    @wire(getRecord, { recordId: '$recordId', 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);
            })
    }
}

```

d.

1. Deployment:

- a. Added the component to the "Search Your Property" app page for easy access.



Step 8: Deployment and Security

1. Deployment:

- Deployed the app, components, and workflows into the production environment.
- Activated pages and profiles for end-user interaction.

Custom (1)

propSyncComponent

c.

The image shows a Salesforce form titled 'Properties'. At the top, there is a 'Property Type' dropdown menu with a search icon. Below the dropdown, there are two large blue rectangular areas, each containing a button labeled 'Add Component(s) Here'.

1. **Security:**

- a. Configured Apex class access for relevant roles.
- b. Ensured that role-based permissions were strictly enforced.

Technical Implementation

Technologies Used

1. **Salesforce CRM:** For data organization, workflows, and app development.
2. **Jotform:** For creating and integrating customer-facing forms.
3. **Lightning Web Components (LWC):** For creating dynamic user interfaces.
4. **Apex:** For backend logic to support LWC operations.

Integration Details

1. **Jotform Integration:**

- a. Automatically pushes data to Salesforce objects upon form submission.
- b. Reduces manual data entry errors.

2. **Approval Process:**

- a. Configured an automated flow for verifying property details.
- b. Ensures compliance with business requirements.

User Experience

1. Implemented a clean, intuitive interface for property search.
2. Enhanced search options with filters for property type and verification status.

Results

1. **Automation:** Customer data collection and property approvals are fully automated, minimizing human intervention.
2. **Efficiency:** Streamlined workflows reduced turnaround times for property listings.
3. **User Satisfaction:** A tailored property search experience increased engagement.
4. **Business Growth:** Simplified operations allow scalability and attract more clients.

Challenges

1. Complex role and permission configurations for various business requirements.
2. Ensuring the security of customer and property data during integrations.
3. Optimizing workflows to handle large volumes of records efficiently.

Conclusion

The "Estate Masters" project successfully addresses the challenges in real estate management by integrating Salesforce with automated workflows. It enhances the customer experience, optimizes operations, and supports scalability. The solution is a benchmark for leveraging CRM tools in the real estate sector.

Future Scope

1. AI-Driven Recommendations:

- a. Use machine learning to analyze customer preferences and suggest properties.

2. Advanced Reporting:

- a. Add dashboards for insights into sales performance and customer behavior.

3. Mobile Integration:

- a. Develop a mobile-friendly version for better accessibility.

Output

1. **Customer Form Link:** [Jotform Link](#)
2. **Approval Process Screenshots**



ESTATE MASTERS

First Name

Last Name

Email

example@example.com

Address

Street Address

Street Address Line 2

City

State / Province

Postal / Zip Code

Phone Number

e.g., 23

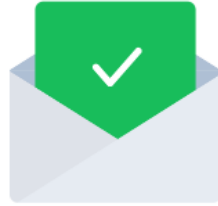
Which Type of property you are looking for?

- ☐ RESIDENTIAL
☐ COMMERCIAL
☐ RENTAL

Budget

e.g., 23

Submit



Thank You!

Your submission has been received.

Now create your own Jotform - It's free!

Create your own Jotform

Customer										
Recently Viewed										
4 items • Updated a few seconds ago										
Q Search this list...										
<input type="checkbox"/>	Customer	Customer	Phone Number	Emial	State	Property Type	Budget ...	Street Address	Street Address line 2	City
1	<input type="checkbox"/>	Naveen	KUMAR	9,34,67,56,778	21501a05d3@pvpsitac.in	RESIDENTIAL	23,000	rajeev nagar	visaalandhra colony	
2	<input type="checkbox"/>	a06NS00000Oyk29	0				0			
3	<input type="checkbox"/>	a06NS00000Oyjqr	0				0			
4	<input type="checkbox"/>	a06NS00000OyjH8	0				0			

Customer

All Records

New

Change Owner

Import

Printable View

Assign Label

Q

Search this list...

7 items • Sorted by Customer • Filtered by All customer • Updated a few seconds ago

	<div><div></div><div>Customer ↑</div></div>	<div><div></div><div>Custo...</div></div>	<div><div></div><div>Phone Num...</div></div>	<div><div></div><div>Emial</div></div>	<div><div></div><div>State</div></div>	<div><div></div><div>Property Ty...</div></div>	<div><div></div><div>Budget ...</div></div>	<div><div></div><div>Street Add...</div></div>	<div><div></div><div>Street Address lin...</div></div>	<div><div></div><div>City</div></div>	<div><div></div><div>postal ...</div></div>	
1	<div><div></div><div>a06NS00000OxmWj</div></div>	Rakesh	7,88,797	rakesh@gmail.com	Telangana	Residential	40,00,000	gb road	street no 45	Hyderabad	555001	<div></div>
2	<div><div></div><div>a06NS00000OxmWk</div></div>	prakash	5,54,48,855	p@gmail.com	Maharashtra	Commercial	80,00,000	gachibowli	indira road	mumbai	6600014	<div></div>
3	<div><div></div><div>a06NS00000OxmWl</div></div>	Prajwal	4,54,545	prajwal@gmail.com	Maharashtra	Rental	25,000	kamdli	kathora	Amravati	444805	<div></div>
4	<div><div></div><div>a06NS00000OyjH8</div></div>		0				0					<div></div>
5	<div><div></div><div>a06NS00000Oyjqr</div></div>		0				0					<div></div>
6	<div><div></div><div>a06NS00000Oyk29</div></div>		0				0					<div></div>
7	<div><div></div><div>Naveen</div></div>	KUMAR	9,34,67,56,778	21501a05d3@pvpsit.ac.in	RESIDENTIAL		23,000	rajeev nagar	visaalandhra colony		520015	<div></div>

Property Details Search your Property Property Customer

Search your Property

Properties

Property Type

Select Property type

Commercial

Residential

rental

Property Details Search your Property Property Customer

Search your Property

Properties

Property Type

Commercial

Property Name Property Type Property Location Property link

500000 sq.ft plot Commercial Amravati

Property Details Search your Property Property Customer

Search your Property

Properties

Property Type

Residential

Property Name Property Type Property Location Property link

Lotus Apartments Residential hyderabad

3. Apex Class 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];

    }
}
```

```
}
```

4. **LWC Component Code** **property.js:**

```
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 property 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)
        })
    }
}

```

property.html:

```
<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>

  </div>

</div>
```

```

        </div>
    </div>
    <template if:true={isttrue}>
        <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>

```

property.js-meta.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">
    <apiVersion>62.0</apiVersion>
    <isExposed>true</isExposed>
    <targets>
        <target>lightning__RecordPage</target>
    </targets>
</LightningComponentBundle>

```

<target>lightning__AppPage</target>

<target>lightning__HomePage</target>

</targets>

</LightningComponentBundle>

5. Role Hierarchy and Profiles Configuration

Sales Representative

Edit | Del | Assign

Add Role

Sales Executive

Edit | Del | Assign

Add Role

Sales Manager

Edit | Del | Assign

Add Role

Customer

Edit | Del | Assign

Add Role

Profile Edit

Customer

Help for this Page

Set the permissions and page layouts for this profile.

Profile Edit

Save Save & New Cancel

Name

Customer

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"/>	Property Details (Property_Details)	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>
App Launcher (standard__AppLauncher)	<input type="checkbox"/>	<input type="radio"/>	WDC (standard__Work)	<input type="checkbox"/>	<input type="radio"/>
Platform (standard__Platform)	<input type="checkbox"/>	<input type="radio"/>			

Profile Edit

Manager

Help for this Page

Set the permissions and page layouts for this profile.

Profile Edit

Save Save & New Cancel

Name

Manager

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"/>	Property Details (Property_Details)	<input checked="" type="checkbox"/>	<input type="radio"/>
App Launcher (standard__AppLauncher)	<input type="checkbox"/>	<input type="radio"/>	WDC (standard__Work)	<input type="checkbox"/>	<input type="radio"/>
Platform (standard__Platform)	<input checked="" type="checkbox"/>	<input checked="" type="radio"/>			