Project Report

A CRM Application to Handle Client and Property-Related Requirements

Tumuluri Jaya Surya Sasank

jayasuryasasank@gmail.com

Table of Contents

Abstract

The "**PropSync Properties**" 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. PropSync Properties 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

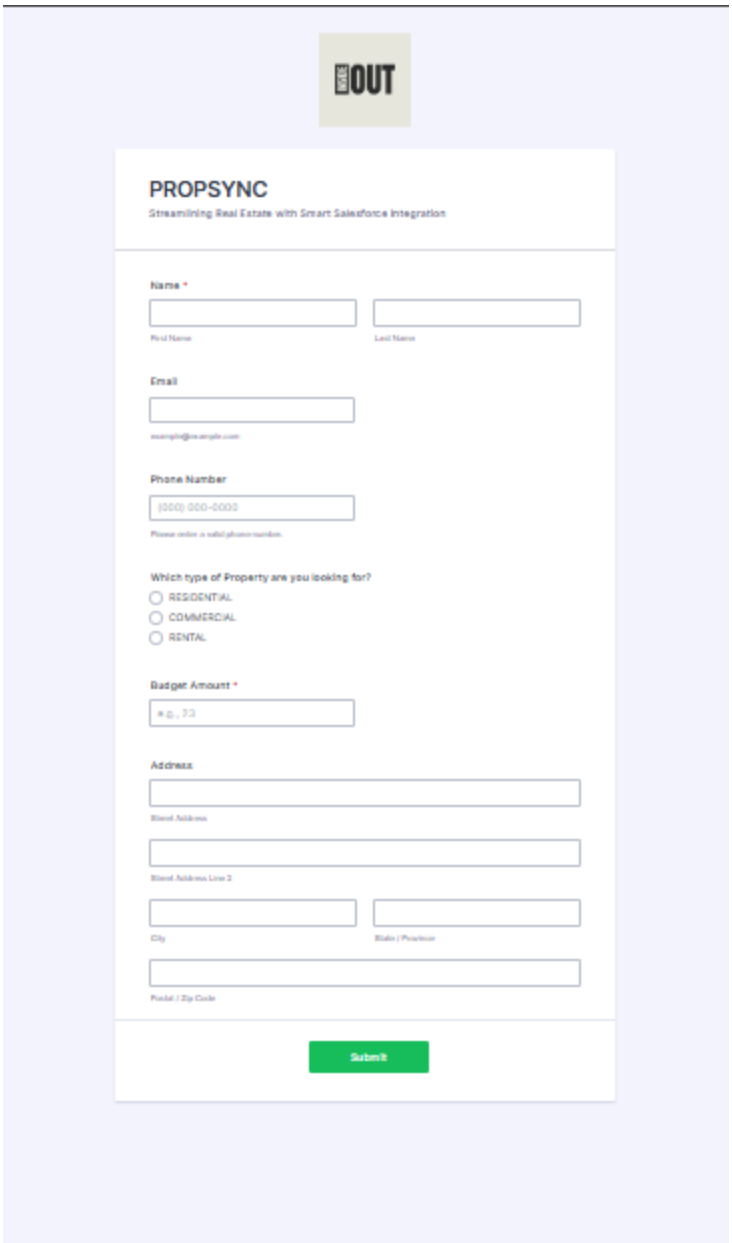
The primary goals of this project include:

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.

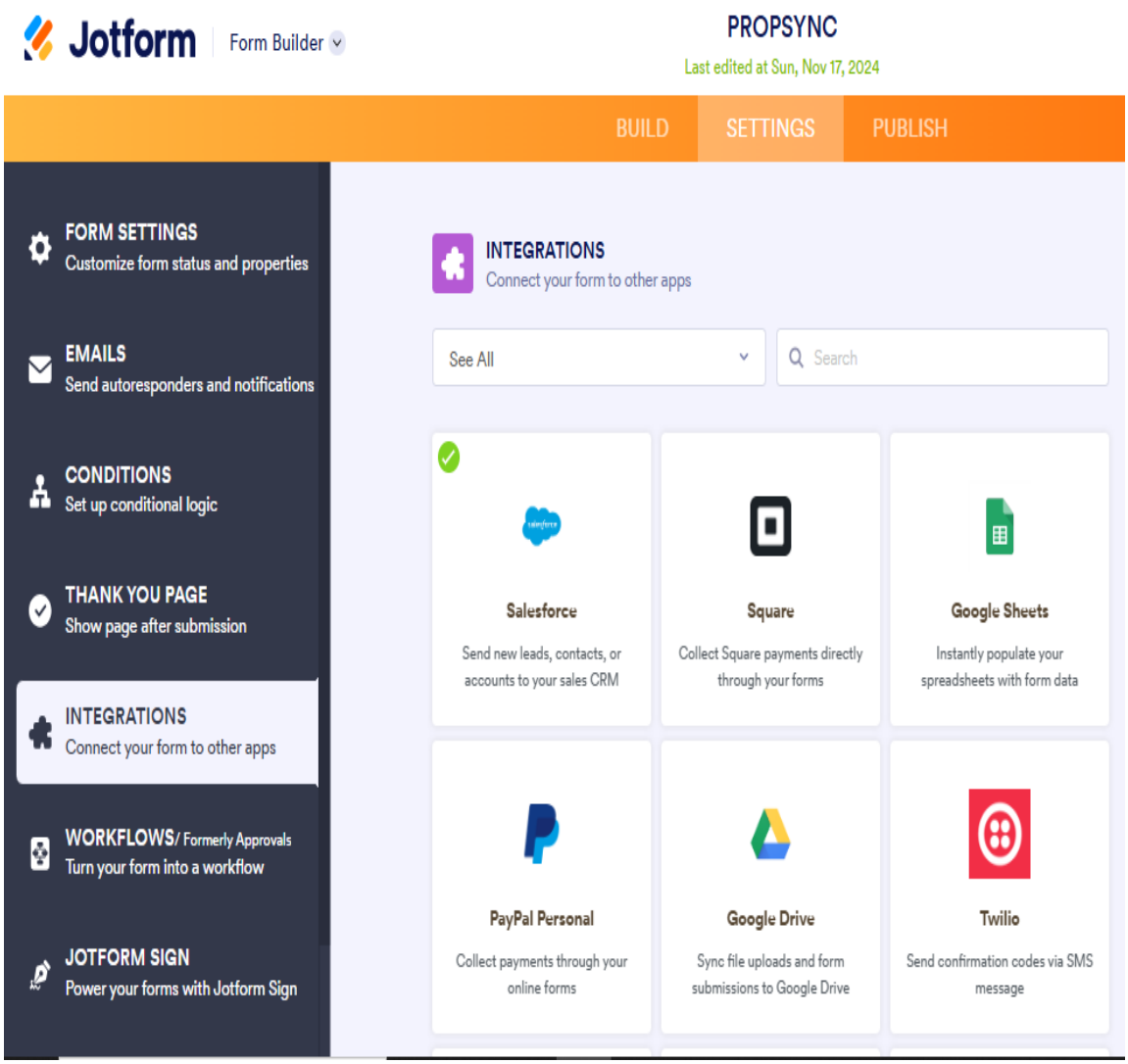
Procedure

Step 1: Data Collection and Form Integration

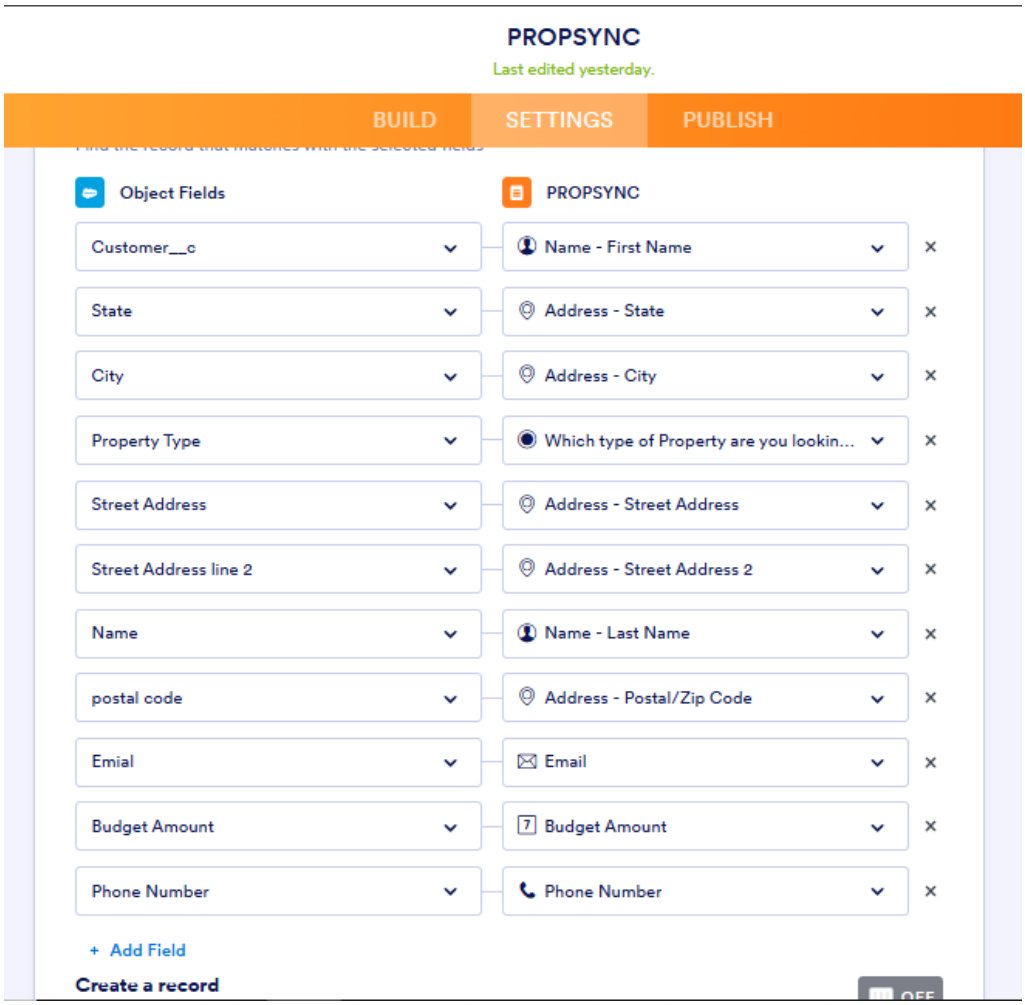
- **Tool**: Jotform
- **Implementation**:
  - Designed a customer-facing web form for collecting user data such as name, contact details, and property preferences.



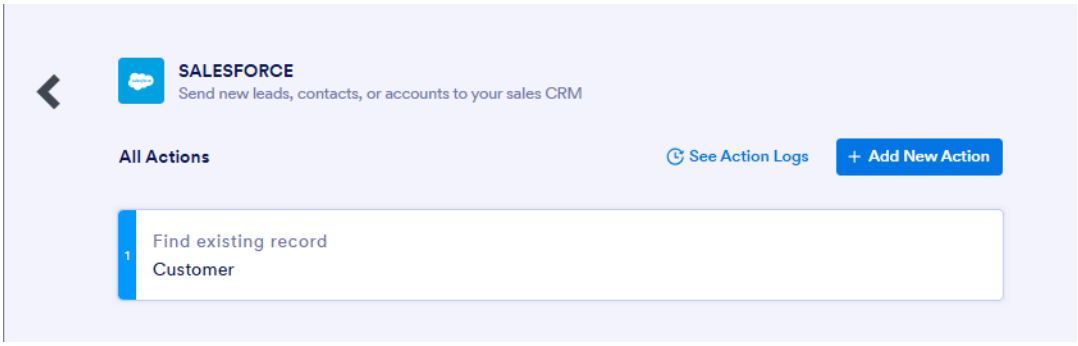
- Published the form and integrated it with Salesforce for automatic record creation.



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

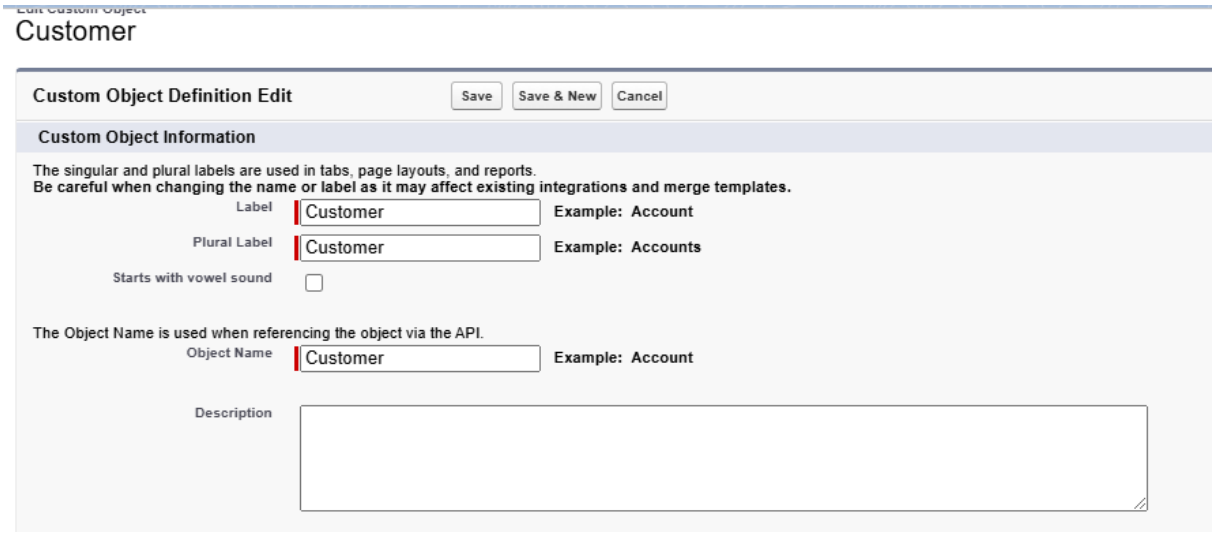


- **Result**: Automated and accurate collection of customer details.



Step 2: Custom Object Creation

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

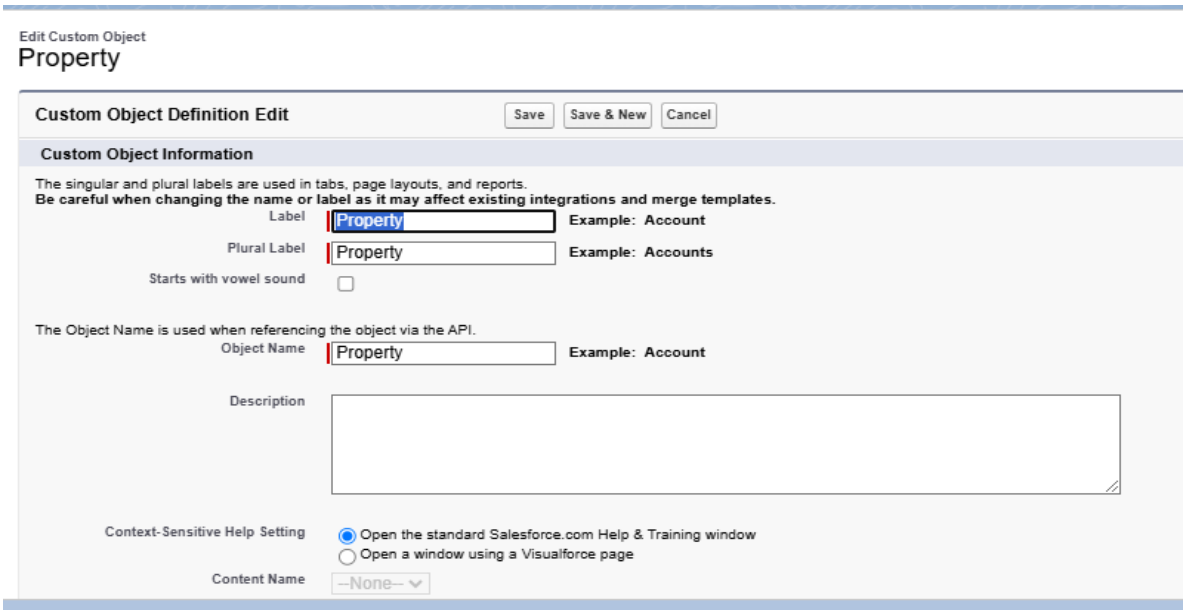


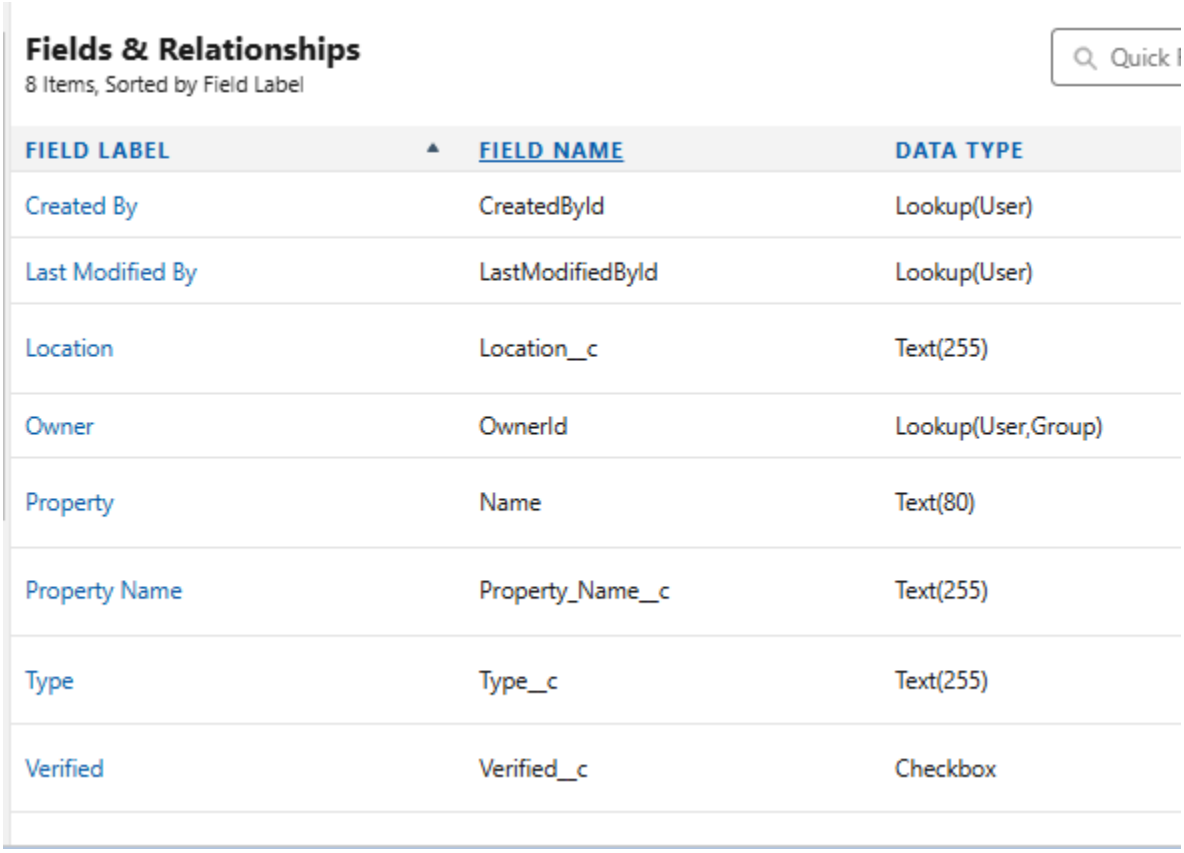
Fields & Relationships

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)
Emial	Emial__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

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





Fields & Relationships

8 Items, Sorted by Field Label

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

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

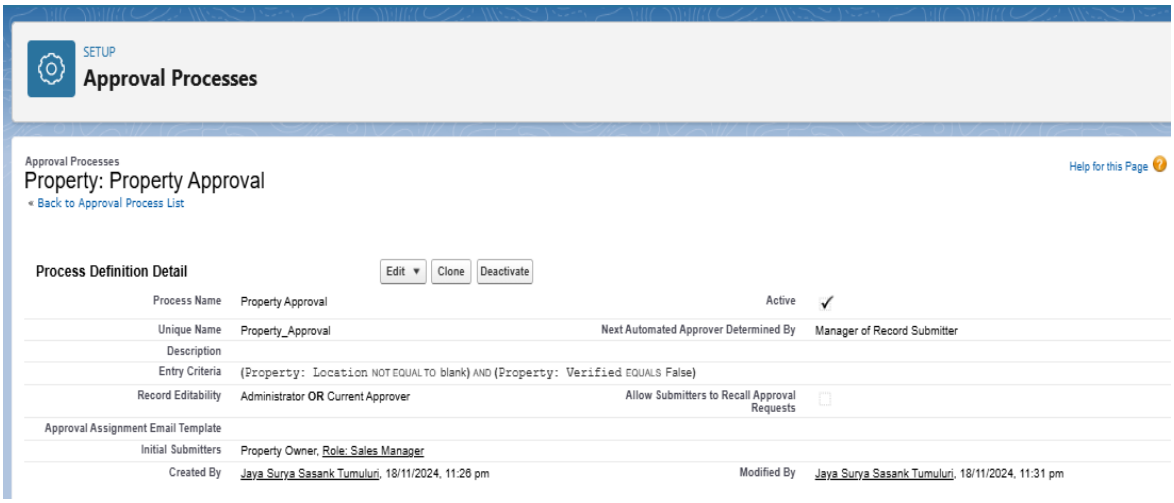
Step 3: Role and Profile Setup

- **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.

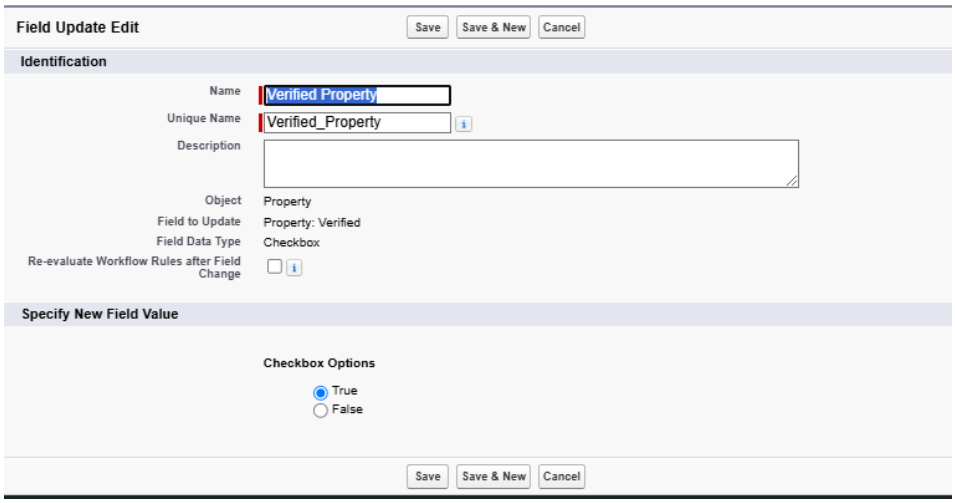
- **Profiles Configured**:
  - Restricted access to non-relevant objects and ensured each role had permissions tailored to their tasks.

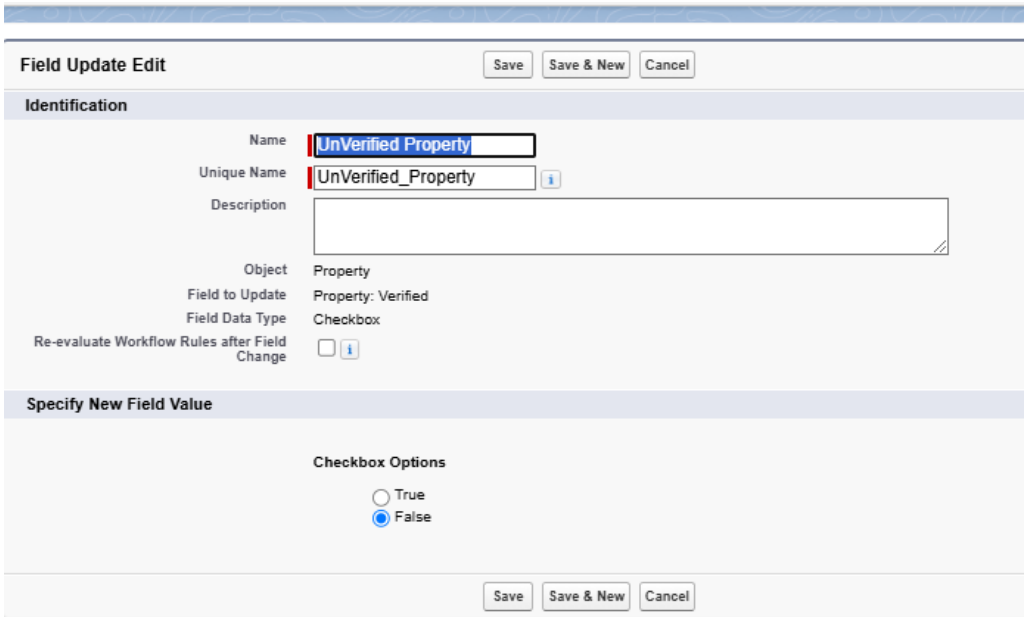
Step 4: Property Approval Workflow

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



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



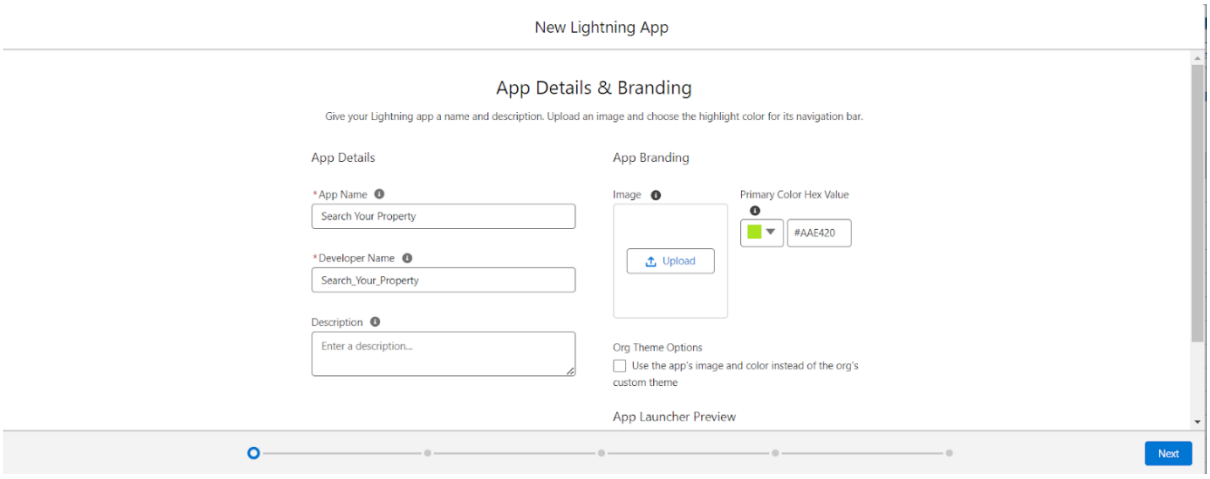


- Approval hierarchy:

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

Step 5: Application Development

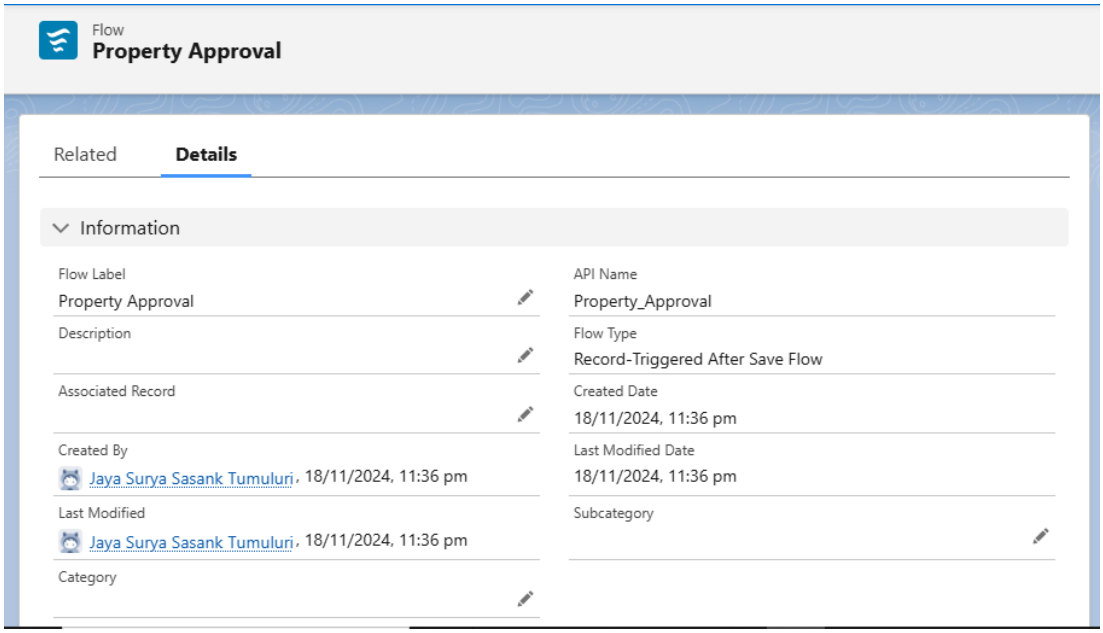
- **Tool**: Salesforce Lightning App Builder
- **App Name**: Property Details

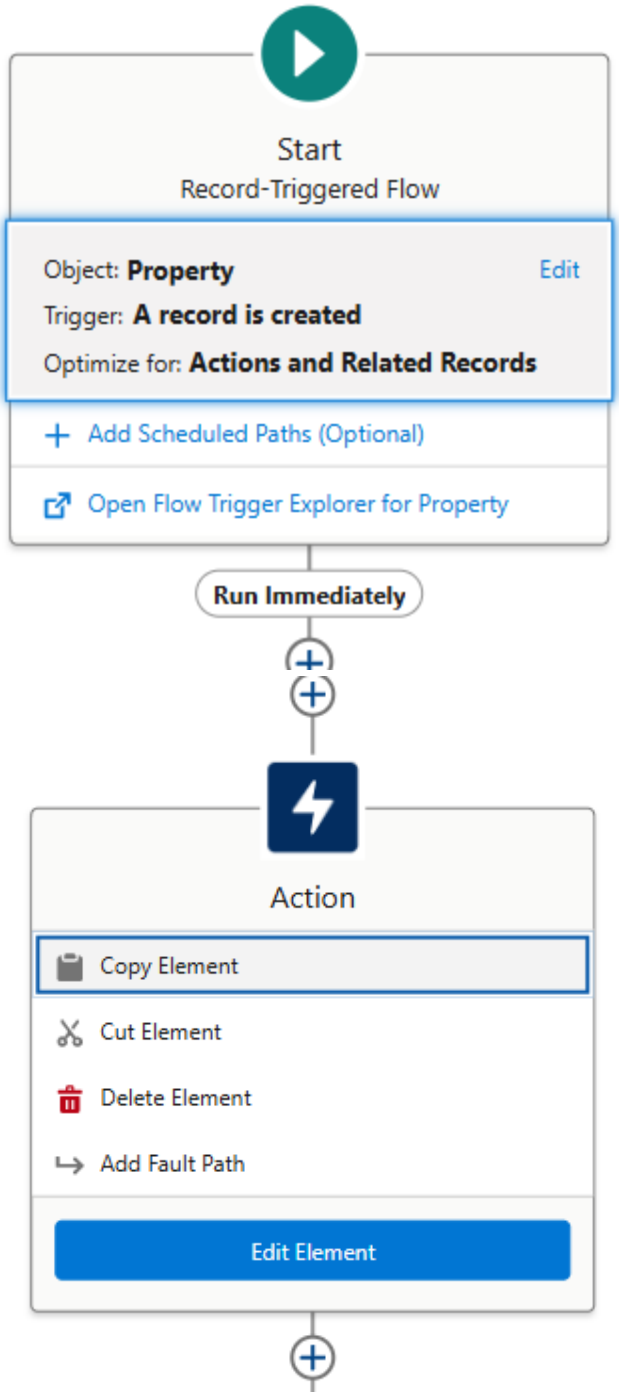


- **Features**:
  - Consolidates customer and property objects.
  - Provides a user-friendly interface for managing records.

Step 6: Automation Through Flows

- **Record Trigger Flow**:
  - Automatically submits property records for approval upon creation.
  - Simplifies the approval process and reduces delays.



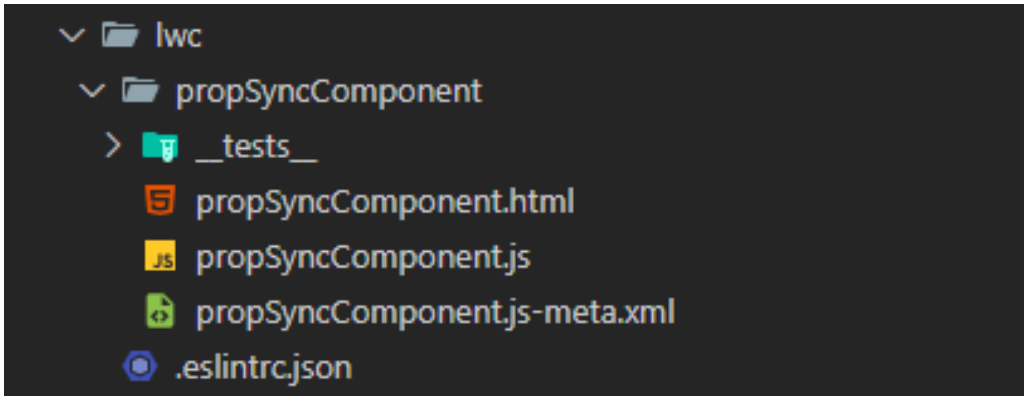

Start
Record-Triggered Flow
Object: Property
Edit
Trigger: A record is created
Optimize for: Actions and Related Records
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Property
Run Immediately
Action
Copy Element
Cut Element
Delete Element
Add Fault Path
Edit Element

Step 7: Lightning Web Component (LWC)

- **Purpose**:
  - Allows verified customers to access approved properties.
  - Restricts non-verified customers to view only non-approved properties.
- **Implementation**:
  - Created an Apex class for querying verified properties.

    Developed an interactive LWC component with property filters.
  -

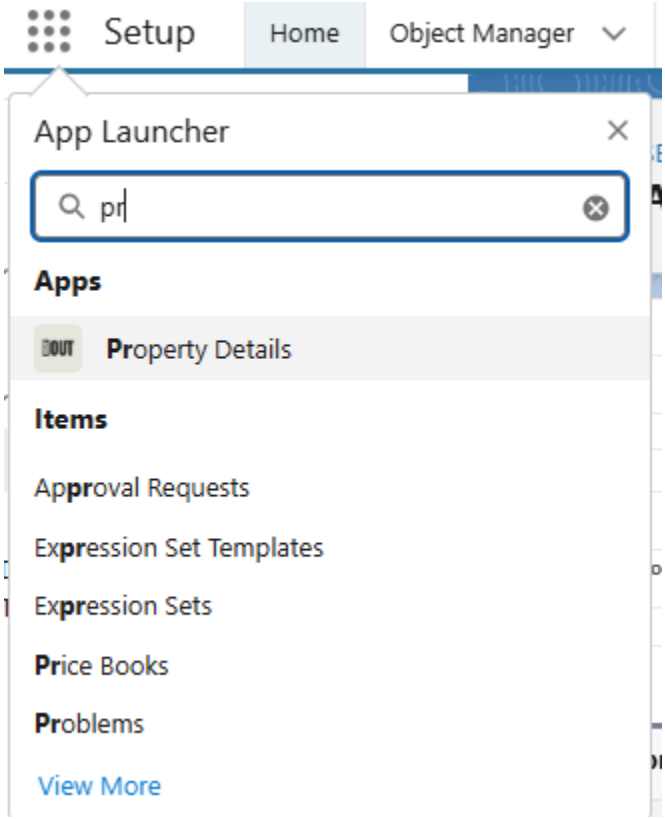
```
PropertHandler_LWC.apxc *
Code Coverage: None   API Version: 62
1 public class PropertHandler_LWC {
2  @AuraEnabled(cacheable=true)
3     public static list<Property__c> getProperty(string type , boolean verified){
4         return [SELECT Id, Location__c, Property_Name__c, Type__c,
5                 Verified__c FROM Property__c Where Type__c =: type AND Verified__c =: verified];
6     }
7 }
```

  - 

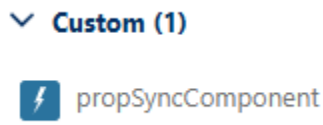
```
<template>
    <lightning-card>
        <div class="slds-box">
          <div class="slds-text-align_left">
            <h1 style="font-size: 20px;"><b>Properties</b></h1>
          </div>
          <div>
            <div class="slds-grid slds-gutters">
              <div class="slds-col slds-size_5-of-6">
                <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>
```

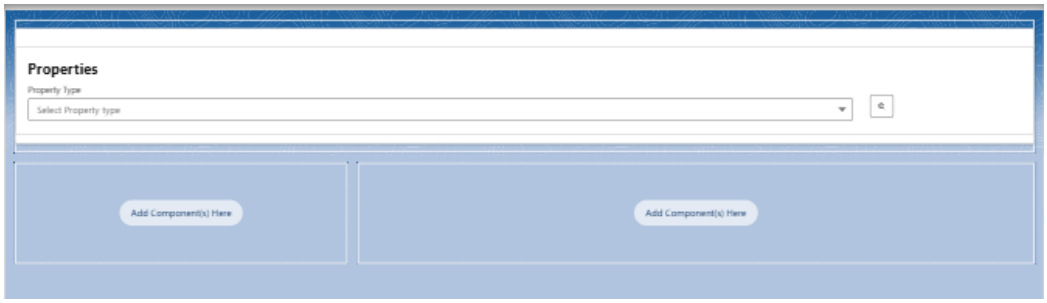
```
import { LightningElement, api, track, wire } from 'lwc';
import getProperty from "@salesforce/apex/PropertHandler_LWC.getProperty"
import { getRecord } from 'lightning/uiRecordApi';
import USER_ID from '@salesforce/user/Id';

export default class 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" }
    ]
    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)
            })
    }
}
```

- **Deployment**:
  - Added the component to the "Search Your Property" app page for easy access.



Step 8: Deployment and Security

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

  - 

- **Security**:
  - Configured Apex class access for relevant roles.
  - Ensured that role-based permissions were strictly enforced.

Technical Implementation

Technologies Used

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

Integration Details

- **Jotform Integration**:
  - Automatically pushes data to Salesforce objects upon form submission.
  - Reduces manual data entry errors.
- **Approval Process**:
  - Configured an automated flow for verifying property details.
  - Ensures compliance with business requirements.

User Experience

- Implemented a clean, intuitive interface for property search.
- 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 "PropSync Properties" 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.

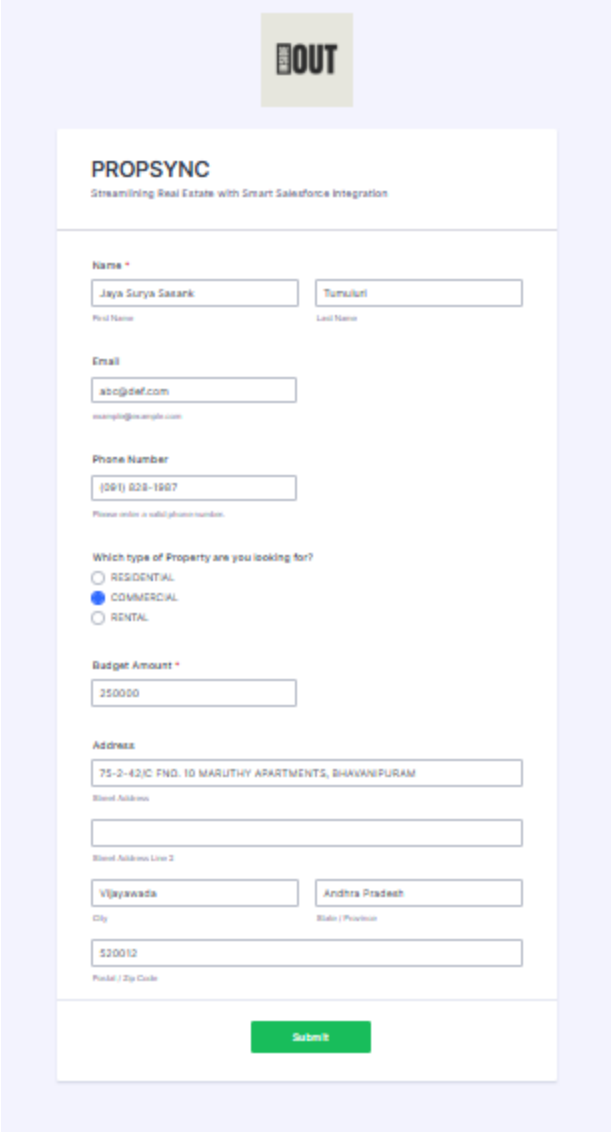
Future Scope

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**



JOUT

PROPSYNC

Streamlining Real Estate with Smart Salesforce Integration

Name *

Jaya Surya Sasank | Tumuluri

First Name | Last Name

Email

abc@def.com

Phone Number

(091) 828-1907

Please enter a valid phone number.

Which type of Property are you looking for?

- RESIDENTIAL
- COMMERCIAL
- RENTAL

Budget Amount *

350000

Address

75-2-42/C FNO. 10 MARUTHY APARTMENTS, BHAVANIPURAM

Street Address

Street Address Line 2

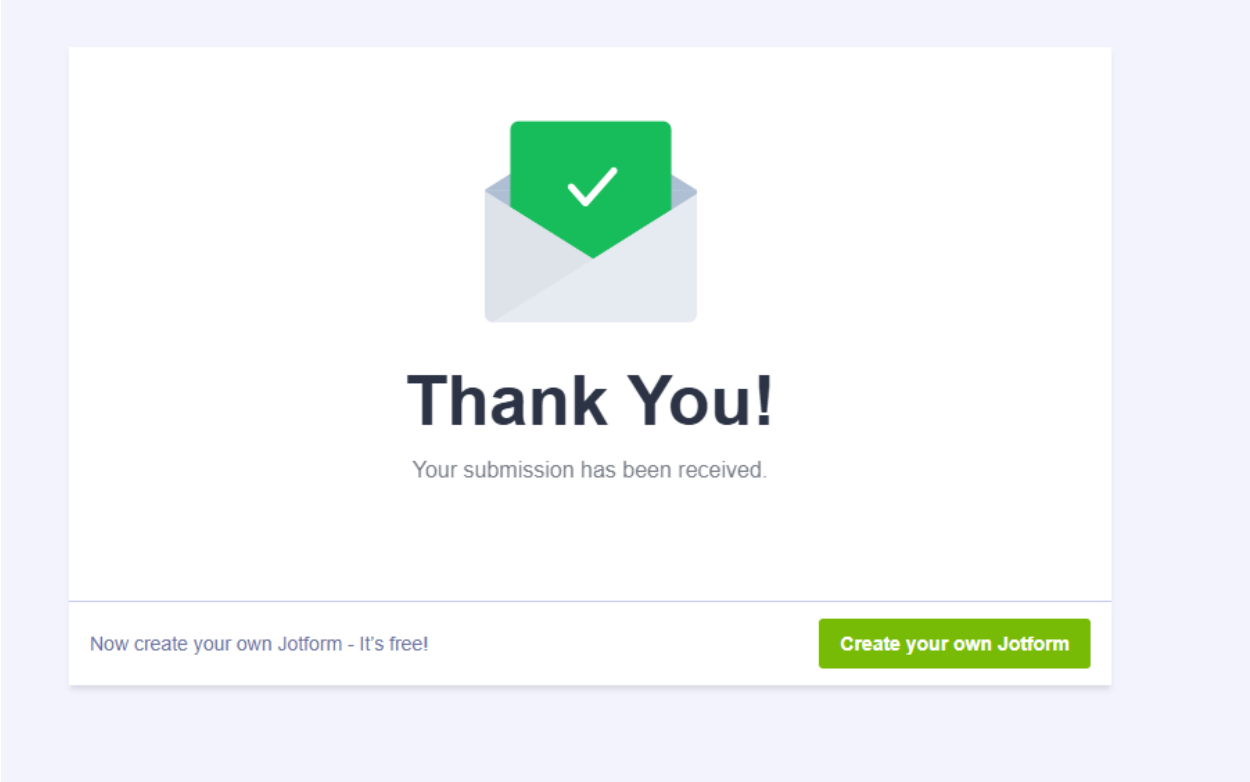
Vijayawada | Andhra Pradesh

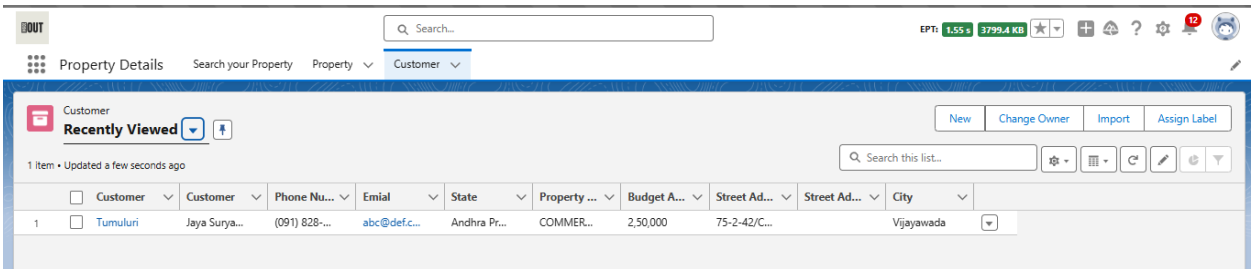
City | State / Province

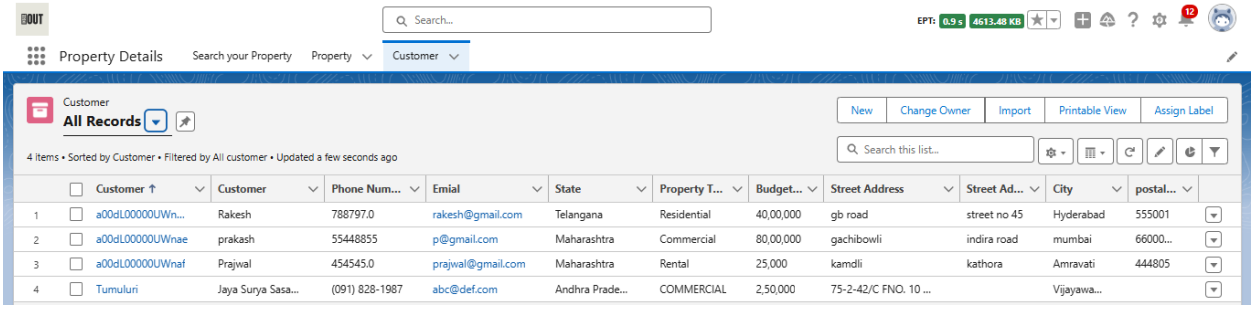
520012

Postal / Zip Code

Submit


Thank You!
Your submission has been received.
Now create your own Jotform - It's free!
Create your own Jotform


Search...
Property Details
Search your Property
Property
Customer
Customer
Recently Viewed
1 item • Updated a few seconds ago
New
Change Owner
Import
Assign Label
Search this list...
Customer
Customer
Phone Nu...
Emial
State
Property ...
Budget A...
Street Ad...
Street Ad...
City
1
Tumuluri
Jaya Surya...
(091) 828-...
abc@def.c...
Andhra Pr...
COMMER...
2,50,000
75-2-42/C...
Vijayawada


Search...
Property Details
Search your Property
Property
Customer
Customer
All Records
4 Items • Sorted by Customer • Filtered by All customer • Updated a few seconds ago
New
Change Owner
Import
Printable View
Assign Label
Search this list...
Customer ↑
Customer
Phone Num...
Emial
State
Property T...
Budget...
Street Address
Street Ad...
City
postal...
1 a00dL00000UWn... Rakesh 788797.0 rakesh@gmail.com Telangana Residential 40,00,000 gb road street no 45 Hyderabad 555001
2 a00dL00000UWnae prakash 55448855 p@gmail.com Maharashtra Commercial 80,00,000 gachibowli indira road mumbai 66000...
3 a00dL00000UWnaf Prajwal 454545.0 prajwal@gmail.com Maharashtra Rental 25,000 kamdli kathora Amravati 444805
4 Tumuluri Jaya Surya Sasa... (091) 828-1987 abc@def.com Andhra Prade... COMMERCIAL 2,50,000 75-2-42/C FNO. 10 ... Vijayawa...

Property Details | Search your Property | Property | Customer

Property
All Records

3 items • Sorted by Property • Filtered by All property • Updated a few seconds ago

New | Change Owner | Import | Printable View | Assign Label

	Property	Property Name	Type	Location	Verified
1	a01dL00000dKJOM	Lotus Appartments	Residential	hydeerabad	
2	a01dL00000dKJON	500000 sq.ft plot	Commercial	Amravati	
3	a01dL00000dKJOO	3 Bhk fkat at stanza	rental	Jubliee hill Hydeeabad	

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 Appartments	Residential	hydeerabad	

Property Details | Search your Property | Property | Customer

Search your Property

Properties

Property Type

rental

Property Name	Property Type	Property Location	Property link
3 Bhk fkat at stanza	rental	Jubliee hill Hydeeabad	

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**

propSyncComponent.js:

```
 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 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" }
    ]
    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)
        })
    }
}
```

propSyncComponent.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>
      <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>
```

propSyncComponent.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>
      <target>lightning__AppPage</target>
      <target>lightning__HomePage</target>
   </targets>
</LightningComponentBundle>
```

5. Role Hierarchy and Profiles Configuration