Project Report

A CRM Application to Handle Client and Property-Related Requirements

Tumuluri Jaya Surya Sasank

jayasuryasasank@gmail.com

**Table of Contents**

[Abstract 2](#_Toc87aqy9z63r55)

[Introduction 3](#_Tocnl4wqj0aa8hj)

[Scope 3](#_Tocu9wdcmndkk4d)

[Objectives 4](#_Toclaclk4gsn59j)

[Procedure 5](#_Toc4b3qhxi2ivlj)

[Technical Implementation 20](#_Toc0zfev2592h7j)

[Conclusion 22](#_Tocesuerphq6aan)

[Future Scope 22](#_Tocjg484knetksn)

[Output 23](#_Tocbpw5id6m5ivx)

# 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

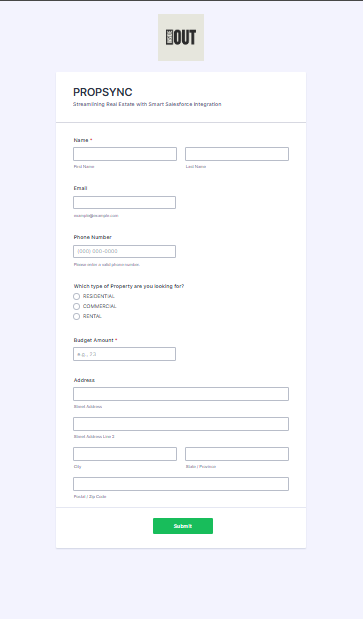
The primary goals of this project include:

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.

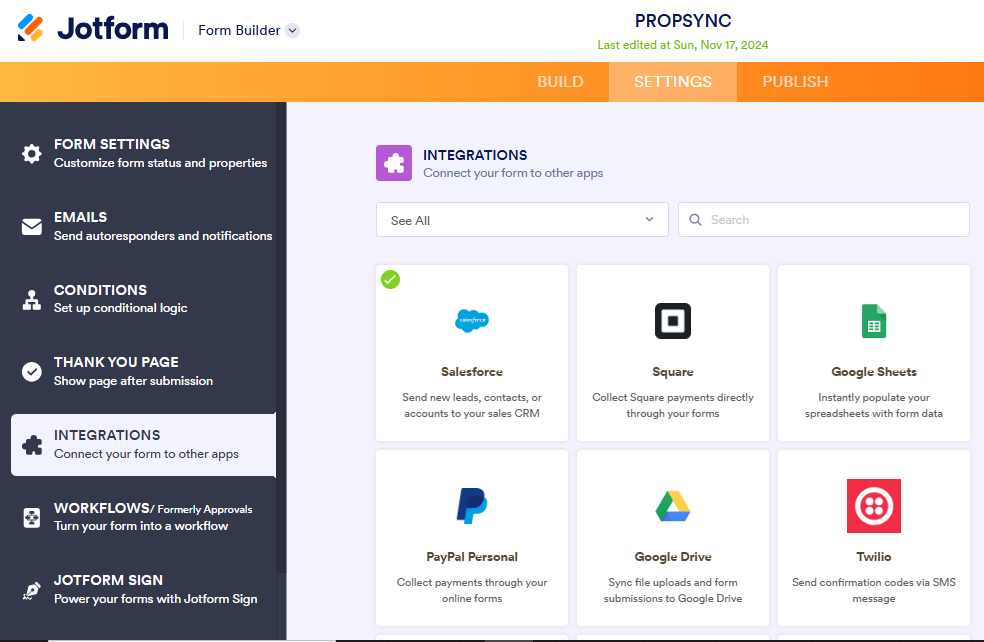
# Procedure

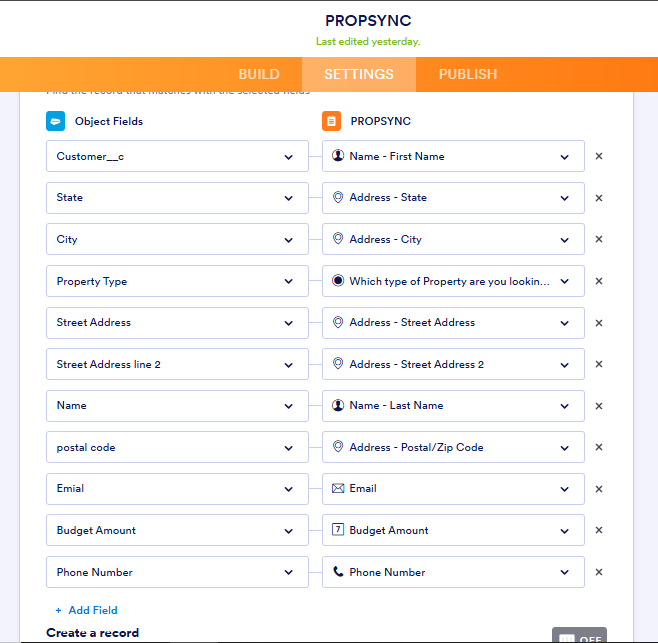
**Step 1: Data Collection and Form Integration**

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

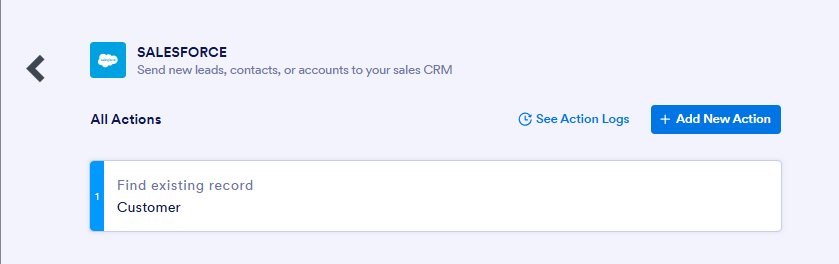


* + Published the form and integrated it with Salesforce for automatic record creation.



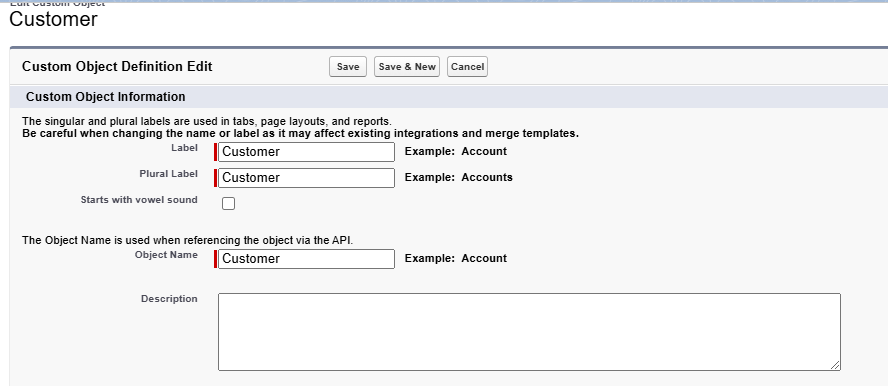
* + Jotform fields were mapped to Salesforce fields to maintain data\_integrity. 

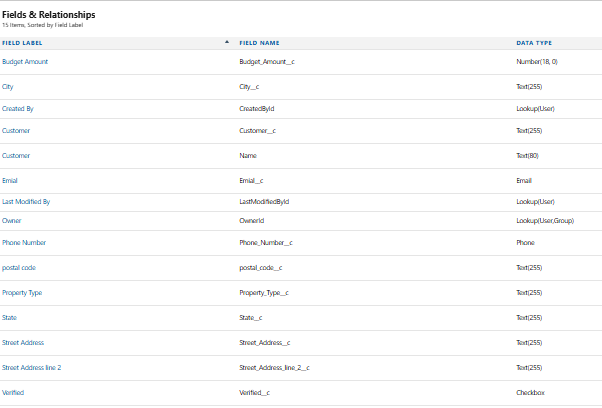
* **Result**: Automated and accurate collection of customer details.



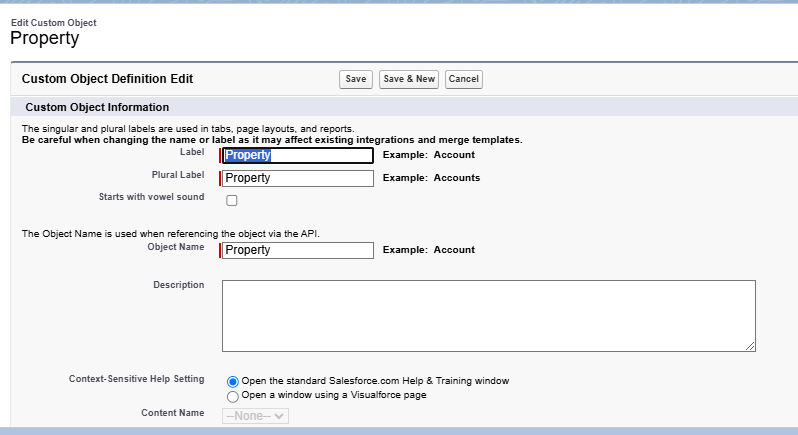
**Step 2: Custom Object Creation**

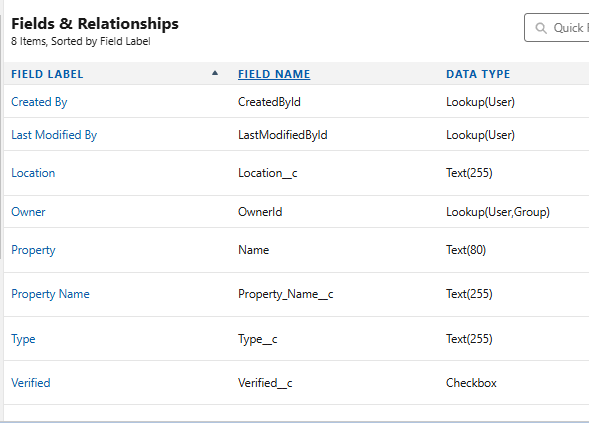
* **Customer Object**: Captures client information such as contact details and preferences.





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





* **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**

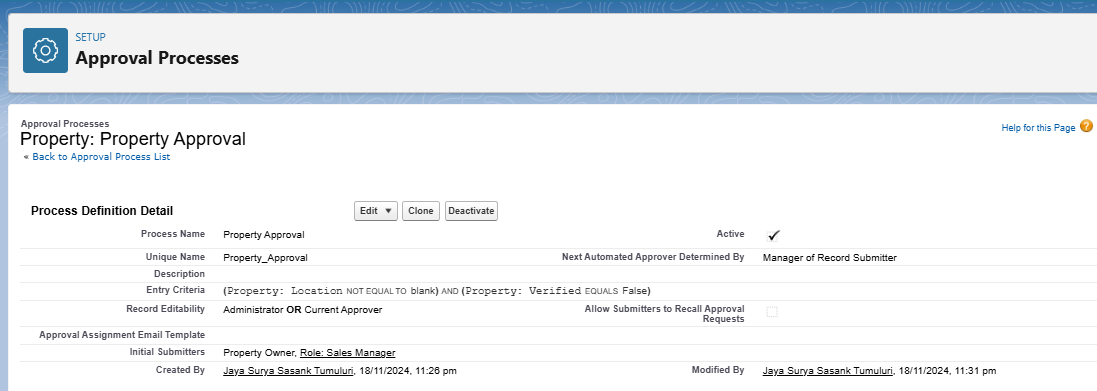
* **Roles Created**:
  + Sales Executive: Handles property listings and interacts with potential customers.
  + Sales Manager: Oversees sales executives and approves critical operations.
  + Customer: End users who search for properties.



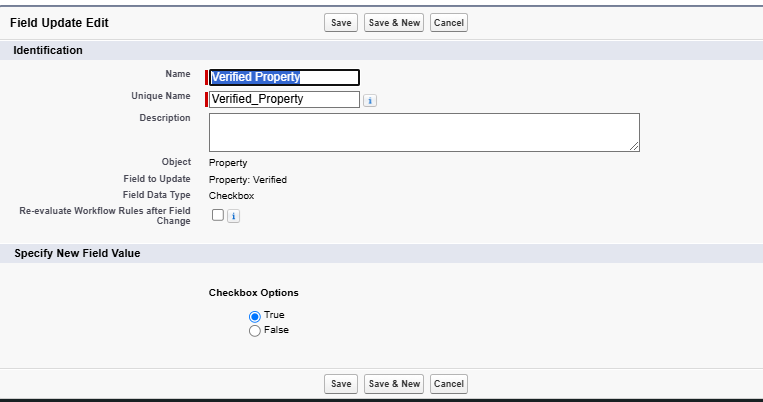
* **Profiles Configured**:
  + Restricted access to non-relevant objects and ensured each role had permissions tailored to their tasks.

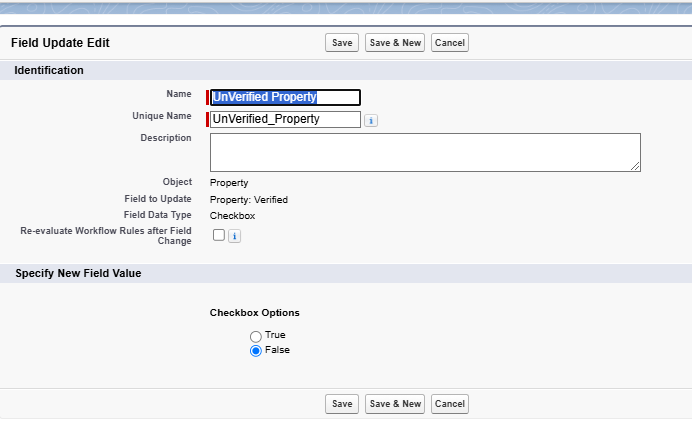
**Step 4: Property Approval Workflow**

* 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."





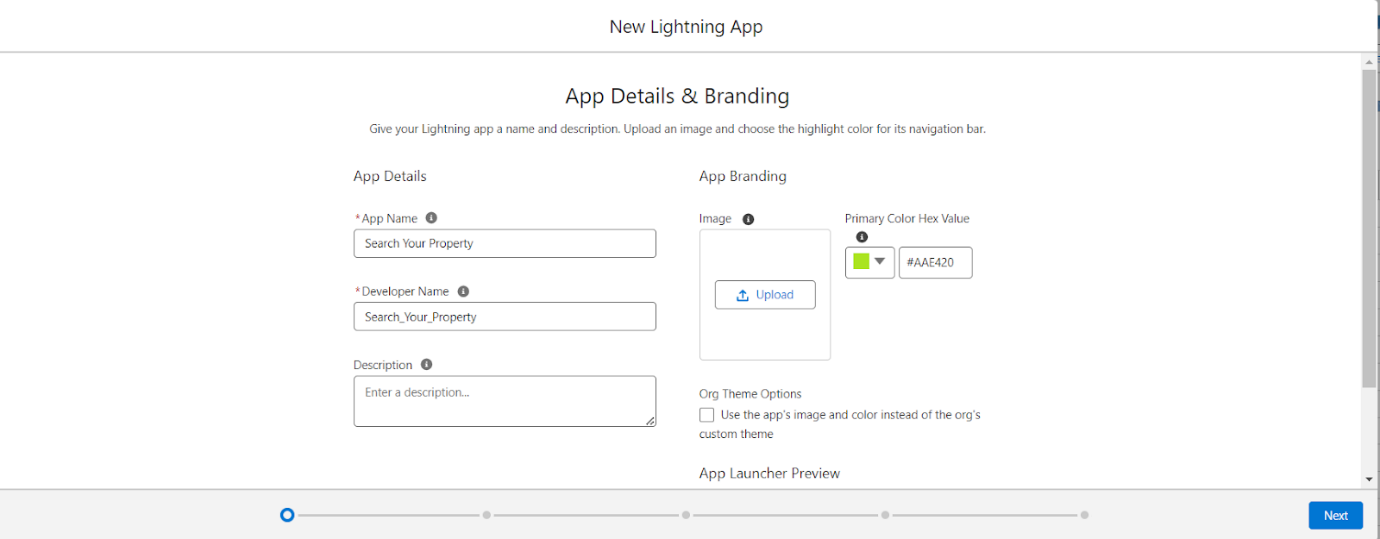
* + Approval hierarchy:

1.Sales Executive for initial review.

2.Sales Manager for final approval.

**Step 5: Application Development**

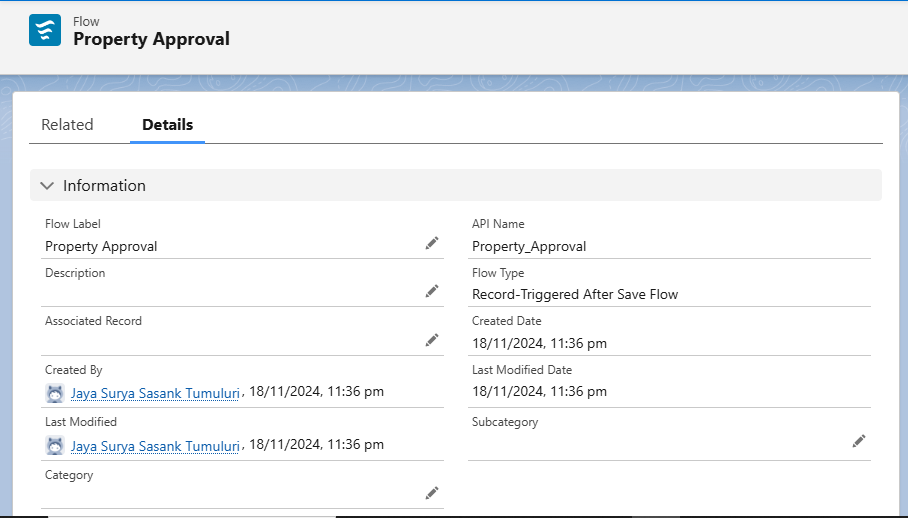
* **Tool**: Salesforce Lightning App Builder
* **App Name**: Property Details

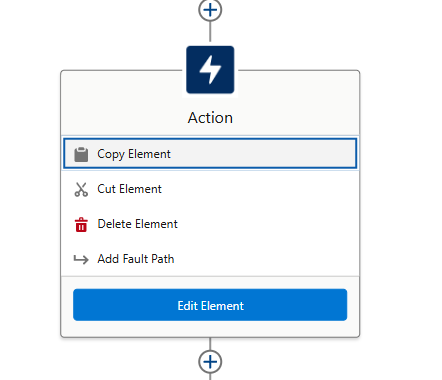
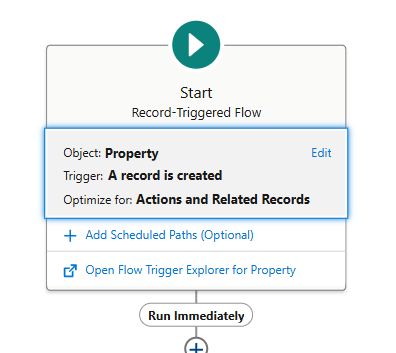


* **Features**:
  + Consolidates customer and property objects.
  + Provides a user-friendly interface for managing records.

**Step 6: Automation Through Flows**

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

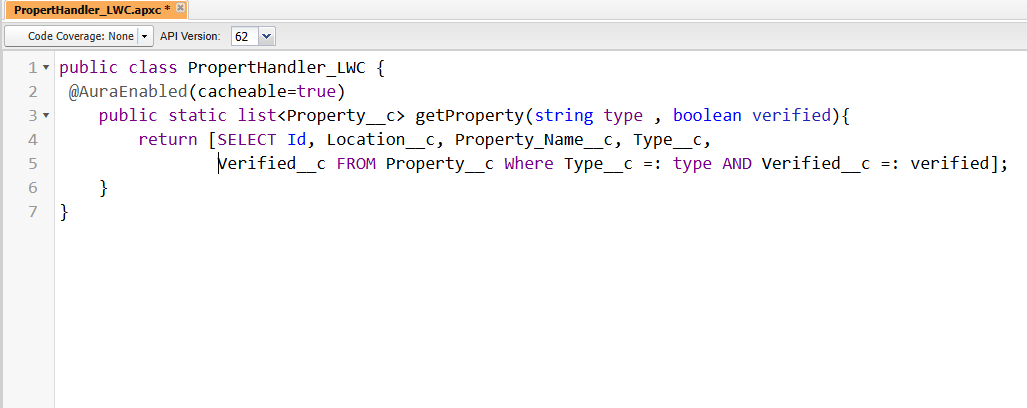
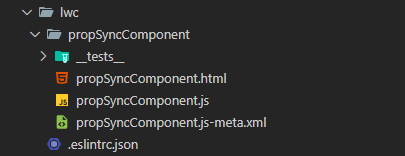
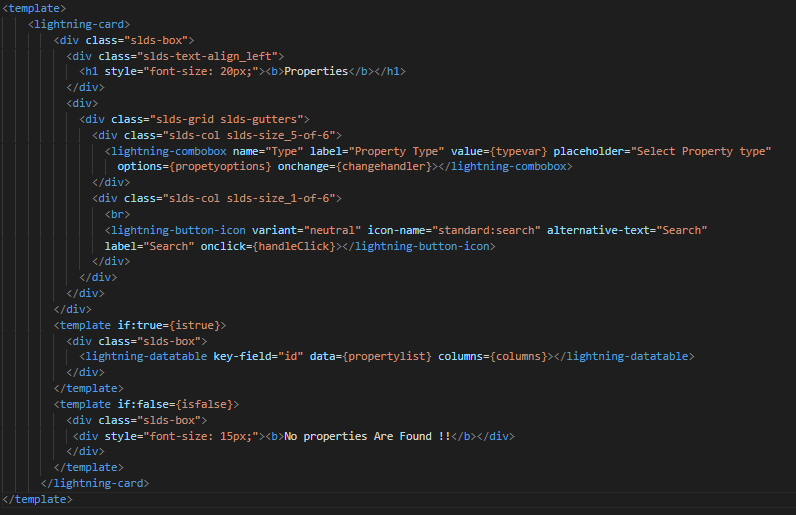
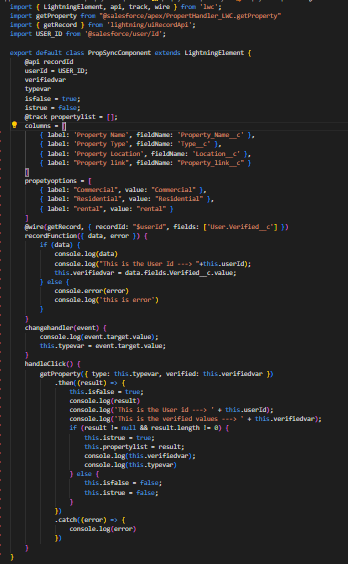




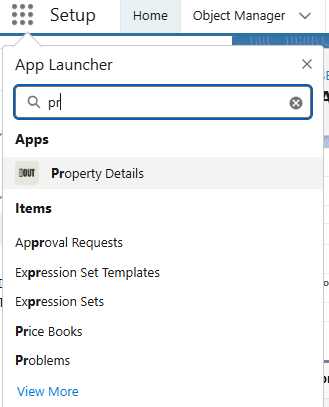
**Step 7: Lightning Web Component (LWC)**

* **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.

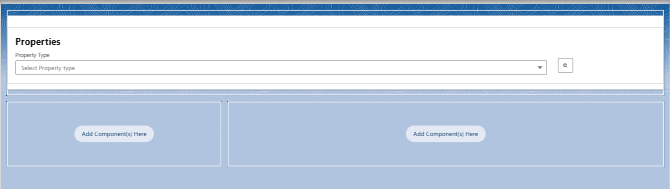
* + 
  + 
  + 
  + 

* **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

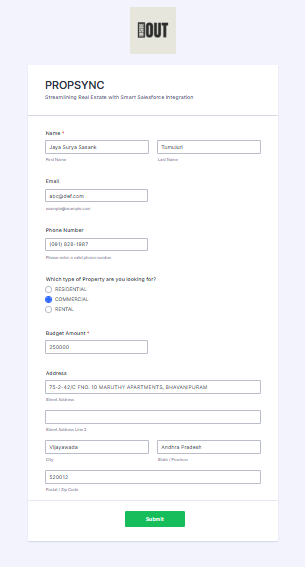
The "PropSync Properties" project successfully addresses the challenges in real estate management by integrating Salesforce with automated workflows. It enhances the customer experience, optimizes operations, and supports scalability. The solution is a benchmark for leveraging CRM tools in the real estate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

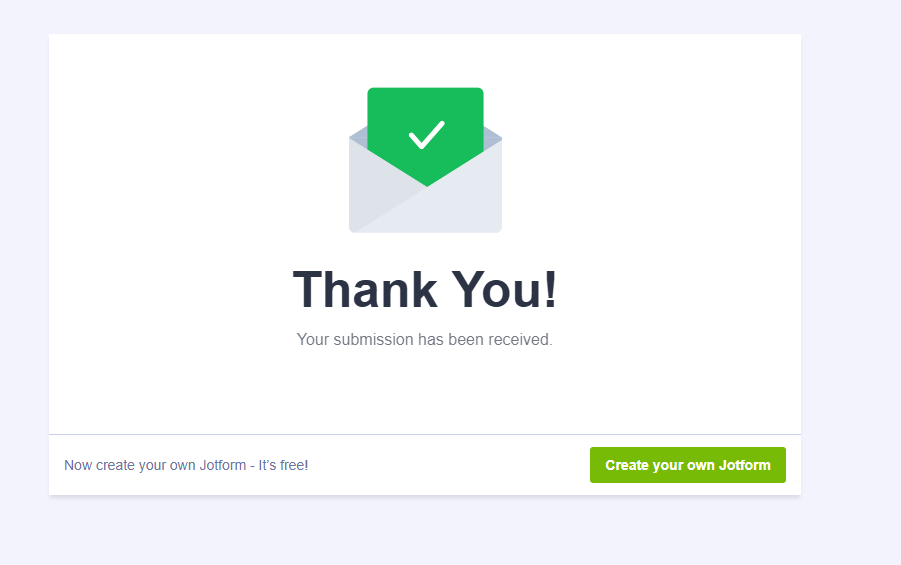
# Future Scope

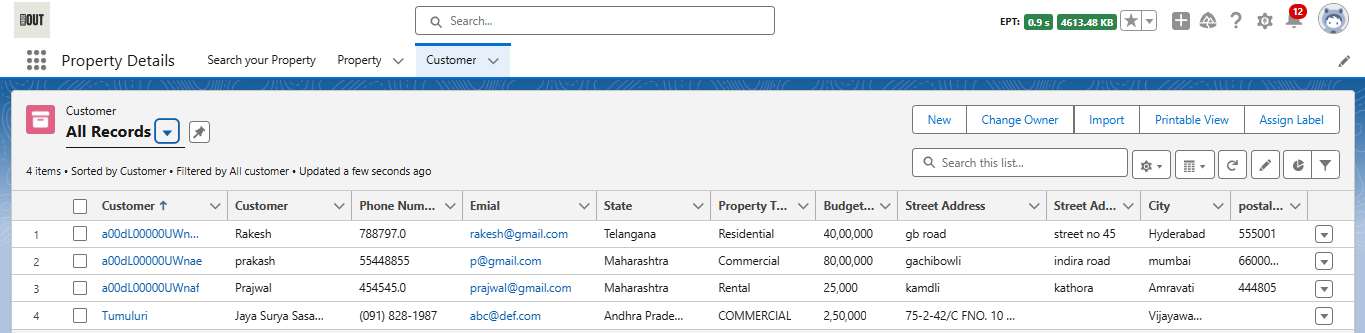
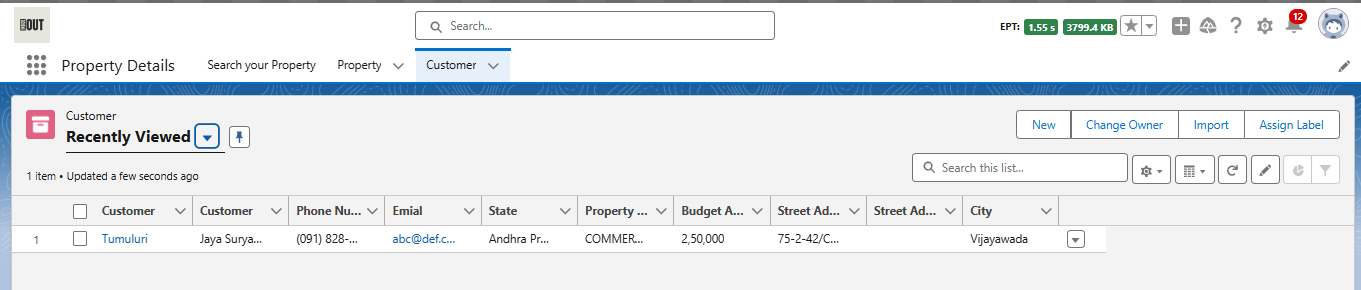
1. **AI-Driven Recommendations**:
   1. Use machine learning to analyze customer preferences and suggest properties.
2. **Advanced Reporting**:
   1. Add dashboards for insights into sales performance and customer behavior.
3. **Mobile Integration**:
   1. Develop a mobile-friendly version for better accessibility.

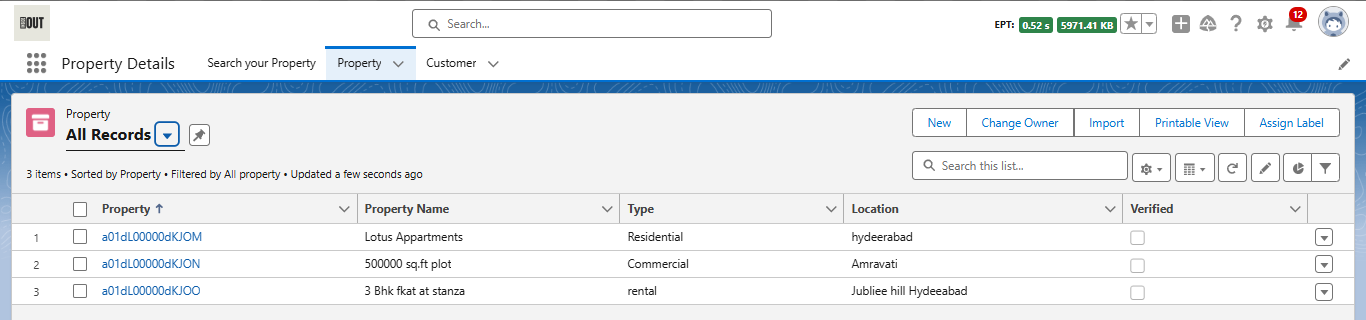
# Output

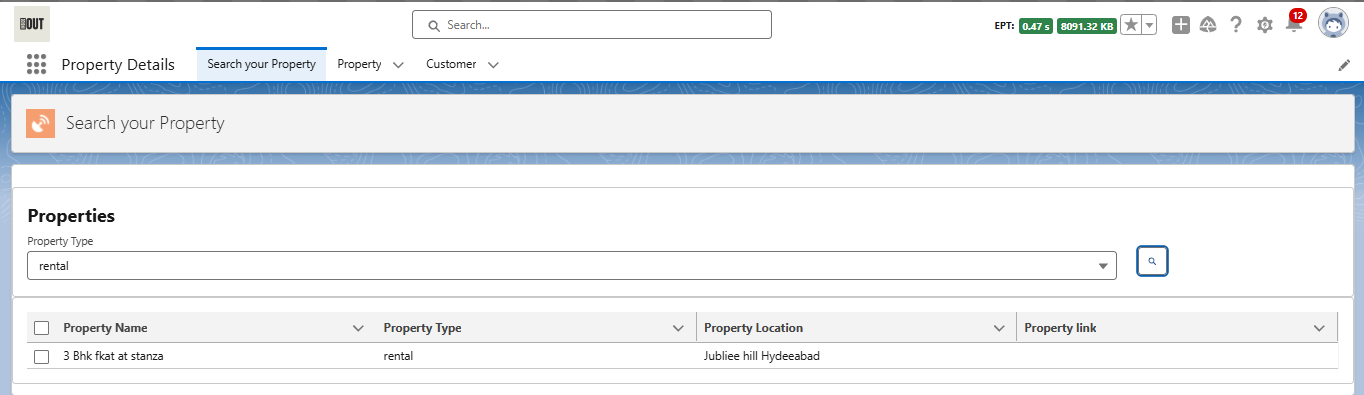
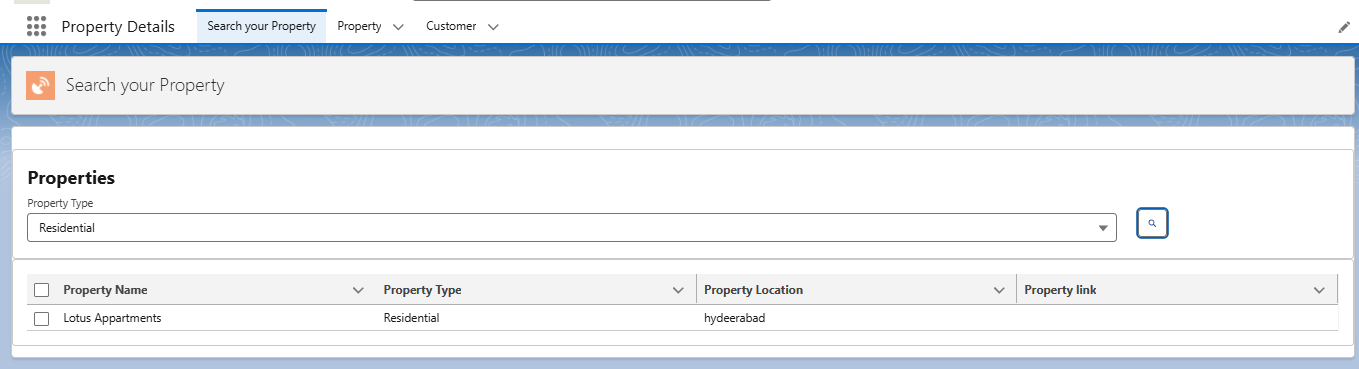
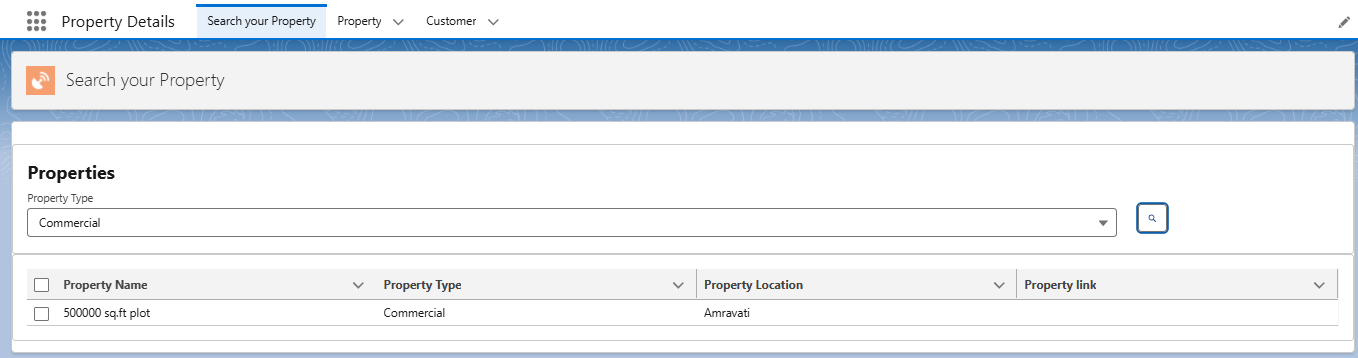
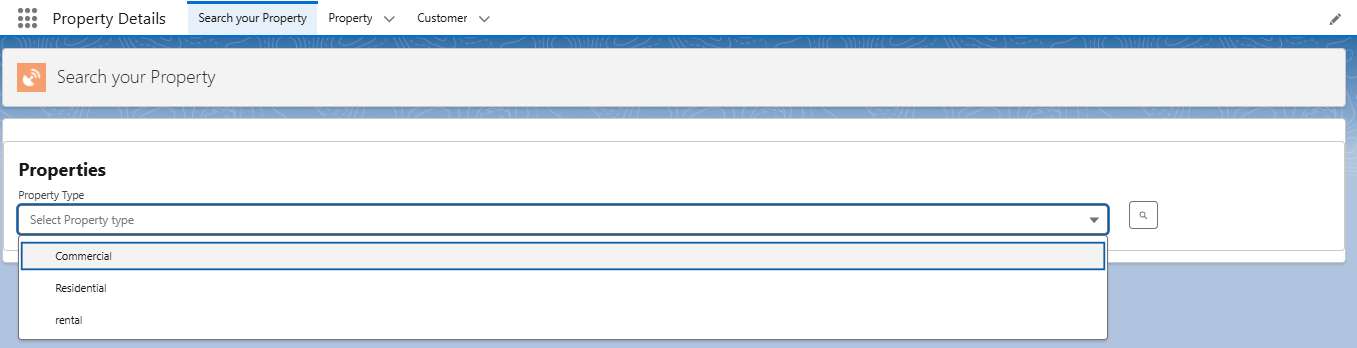
1. **Customer Form Link**: [Jotform Link](https://form.jotform.com/243211297856057)
2. **Approval Process Screenshots**











1. **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];

}

}

1. **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>

1. **Role Hierarchy and Profiles Configuration**

