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

1.Project Overview

The Property Management System (PMS) project developed within Salesforce is designed to automate, streamline, and manage various aspects of property-related operations for a real estate business. The system leverages Salesforce's capabilities to facilitate efficient data management, approval workflows, user interactions, and role-based access control, ensuring the smooth execution of property management tasks across different user groups.

Project Goals:

1. Develop an efficient platform for managing properties and customer information
2. Automate key business processes, including property approval workflows and search functionality.
3. Ensure secure access control tailored to different user types, including Admins, Sales Managers, Sales Executives, and Customers.
4. Address the specific operational needs of each user type, creating a versatile solution adaptable to various roles.
5. Enhance overall system usability and efficiency, providing a seamless experience for property and customer management

2.Objectives

The core objective of this project is to create an efficient platform for managing properties and customer information while automating key business processes such as property approval, search functionality, and secure access for different user types. The system is built to address the specific needs of Admins, Sales Managers, Sales Executives, and Customers, making it a versatile solution for the business.

Business Goals:

This project delivers significant value to the organization by:

Improving Efficiency:
Automating approval workflows, property search, and data entry processes significantly reduces administrative overhead, allowing the team to focus on more critical tasks.

Enhanced User Experience:
A streamlined, user-friendly interface enables both customers and employees to interact with the system more effectively. Customers can easily find properties that match their needs, while employees can quickly process and manage property records.

Data Integrity and Security:
By enforcing role-based access and ensuring that only authorized users can view and modify records, the system ensures the integrity and security of sensitive property and customer data.

Faster Decision-Making: The automated approval process enables quicker approval or rejection of property records, which speeds up decision-making and helps businesses operate more efficiently.

Scalable Infrastructure: With Salesforce as the foundation, the system can scale as the business grows, allowing for more complex workflows, additional users, and new features to be added without significant rework.

Specific outcomes:

1. **Efficient Property and Client Management:** The CRM system allows streamlined handling of clients and their property-related requirements, improving data organization and accessibility.

2. **Automated Approval Workflows:** By automating property approval processes, the system reduces manual intervention, accelerates decision-making, and enhances operational efficiency.

3. **Enhanced Search Functionality:** Advanced search features enable users to quickly locate properties and client information, supporting faster responses and better customer service.

4. **Role-Based Access Control:** Secure access is customized for different user roles—Admins, Sales Managers, Sales Executives, and Customers—ensuring that each user group has the appropriate permissions for their responsibilities.

5. Improved Data Integrity and Security: Salesforce's data management capabilities help maintain accurate and secure records, supporting better compliance and trust with clients.

6. User-Friendly Interface: The system provides a seamless user experience through custom design and Lightning Web Components (LWC), making navigation and data entry straightforward for all user groups.

7. Support for Business Growth: The Property Management System (PMS) is adaptable and scalable, designed to support future business expansion and evolving property management needs.

3.Salesforce Key Features and Concepts Utilized

➢ Customer and Property Objects Creation

Custom Customer and Property objects were created in Salesforce to store key information related to customers and properties.

Customer Object includes fields such as Name, Contact Information, Status, and other details.

Property Object includes fields such as Property Name, Location, Type, Price, Verified Status, and a lookup to the Customer object for the property owner.
The creation of these objects ensures that all necessary property and customer information is stored centrally within Salesforce, making data management and reporting easier.

➢ Jotform Integration for Customer Data Collection

A Jotform was created to collect customer details directly from the website or through other channels.

Integration with Salesforce allows customer data entered into the form to be automatically fed into the Customer Object via the Salesforce API.

This enables admins to view the customer information in Salesforce without manual data entry, increasing efficiency and reducing errors.

➢ Roles and Profiles Configuration

Based on the business requirements, different Roles and Profiles were configured in Salesforce to ensure correct access control.

Roles like Sales Executive, Sales Manager, and Customer were set up for role-based access to records.

Profiles like System Administrator, Manager, and Customer were created to define different levels of permissions on records, apps, and objects in Salesforce.

Profile permissions were set to allow specific access to fields, objects, and data as per user roles.

➢ Approval Process for Property Records

An Approval Process was set up for the Property object to streamline the approval/rejection of property records.

Criteria for Approval: Only properties with a Verified status of false are sent for approval.

Approval Steps: The approval process involves multiple approvers such as the Sales Executive and Sales Manager.

Field Updates: Based on the approval outcome, properties' Verified field is updated accordingly to True or False.

This approval process ensures that only authorized personnel can approve property records, which maintains data integrity.

➢ Record Trigger Flow to Automate Approval Submissions

A Record Trigger Flow was created to automatically submit new property records for approval based on specific criteria (e.g., Verified = false).

This automation eliminates manual intervention in the approval process, increasing the efficiency of the workflow.

➢ Property Details App Page Creation

A custom App Page was created using Lightning App Builder, named "Search Your Property".

This page displays Property details, allowing users to search properties based on different criteria (e.g., property type, verified status).

The app also enables Admins and Sales Managers to view all property records, while Customers only have access to properties based on their verified status.

➢ Lightning Web Component (LWC) for Property Search

A Lightning Web Component (LWC) was developed to display a property search functionality for users.

The component allows users to filter properties by type (e.g., Commercial, Residential, Rental) and by verification status.

User-specific access: Verified customers can view verified properties, while non-verified customers can only view non-verified properties.

The LWC utilizes an Apex class to fetch property data from Salesforce and display it in a dynamic and responsive table format.

Example:

```
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)
})
}
}
```

➢ Apex Class for Property Data Fetch

The Apex class (PropertyHandler_LWC) was created to fetch property data from the Property object based on filters (property type and verified status).

The Apex class is Aura-enabled, which allows it to be called directly from the LWC component for real-time data fetching.

Example:

```
public class PropertHandler_LWC {
 @AuraEnabled(cacheable=true)
 public static List<Property__c> getProperty(String type, Boolean verified) {
 return [SELECT Id, Location_c, Property_Namec, Typec, Verified_c
 FROM Property__c
 WHERE Type_c = :type AND Verified_c = :verified];
 }
}
```

➢ Security and Profile Access for Apex Classes

To ensure proper security, Apex Class Access was configured for the Manager and Customer profiles, granting them the necessary permissions to interact with the PropertyHandler_LWC Apex class.

Profiles were granted specific access to the Apex class, ensuring that only authorized users can trigger actions and access data using the class.

4.Detailed Steps to Solution Design

- **Requirements Gathering & Analysis:**

Engage with stakeholders to identify business needs, especially around property management, client information, and automation needs.

Define roles, permissions, and access levels for Admins, Sales Managers, Sales Executives, and Customers to ensure a tailored solution.

Document core objectives, including property approval workflows, search functionality, secure role-based access, and a centralized database for clients and properties.

- **Data Modeling: Custom Object Creation:**

Create Customer and Property objects in Salesforce:

Customer Object: Define fields such as Name, Contact Information, and Status to manage customer-specific details.

Property Object: Define fields for Property Name, Location, Type, Price, Verified Status, and a lookup relationship to the Customer object, linking properties with their owners.

Ensure these objects facilitate centralized and structured storage for easy access, reporting, and management**.**

- **Customer Data Collection Integration:**

Develop a Jotform for customer data entry, accessible from the website or other external channels.

Integrate Jotform with Salesforce API to automatically populate customer data in the Customer object, minimizing manual data entry.

Configure admin-level access to this data in Salesforce to improve efficiency and data accuracy.

- **Roles and Profiles Configuration for Access Control:**

Define Roles (e.g., Sales Executive, Sales Manager, Customer) and assign these to users based on their responsibilities and the data they need to access.

Create Profiles (e.g., System Administrator, Manager, Customer) to control permissions on objects, fields, and applications:

Grant access to certain profiles for specific fields and objects based on business rules.

Define permissions for viewing, editing, and approving property and customer records to meet security requirements.

- **Approval Process Design for Property Records:**

Set up an Approval Process for the Property object:

Establish criteria where only properties with Verified Status set to false are sent for approval.

Define approval stages involving multiple approvers, such as Sales Executives and Sales Managers.

Configure Field Updates to set Verified Status to true or false based on approval or rejection, maintaining data integrity.

- **Automation with Record Trigger Flow:**

Implement a Record Trigger Flow to automatically submit property records for approval when they meet certain criteria (e.g., Verified = false).

Design the flow to handle approval submissions without manual intervention, improving workflow efficiency and consistency.

- **Custom App Page Development using Lightning App Builder:**

Build a "Search Your Property" App Page to facilitate property searches and display key property details:

Use filters (e.g., property type, verified status) to allow users to easily find specific properties.

Configure different access views for Admins, Sales Managers, and Customers based on their roles and property verification status.

- **Develop Lightning Web Component (LWC) for Property Search:**

Create an LWC component for real-time property search functionality:

Allow users to filter properties by type (Commercial, Residential, Rental) and verified status.

Implement role-specific access, allowing verified customers to view verified properties and non-verified customers to view non-verified ones.

Design a dynamic and responsive table format using Apex integration to display search results.

5.Testing and Validation

Perform thorough testing across user roles to validate access controls, workflows, automation, and data accuracy.

Conduct user acceptance testing (UAT) with a sample of end-users to gather feedback on usability and adjust functionalities as needed.

1. Unit Testing

Unit testing focused on verifying the functionality and accuracy of individual components and code logic, especially custom objects, Apex classes, flows, and Lightning Web Components (LWC).

Key Areas of Unit Testing:

Apex Classes:

PropertyHandler_LWC Apex Class:

Verify that the getProperty method correctly filters properties based on input parameters (type and verified status).

Test handling of edge cases (e.g., when there are no matching properties or unexpected data input).

Ensure that the method only returns properties accessible to the user based on role-based access settings.

Approval Process:

Verify that properties with a "Verified" status of false are correctly submitted for approval.

Test each step of the approval process, ensuring accurate role-based approvals and correct field updates on approval or rejection.

Record Trigger Flow:

Validate that the flow correctly submits new property records for approval when the Verified status is false.

Test flow handling of various scenarios (e.g., missing data, invalid field values) to confirm robustness.

Unit Test Implementation:

Use Salesforce’s Apex test classes with test methods that cover different scenarios for each Apex class and flow.

Ensure that all tests meet a high coverage percentage (typically at least 75% per Software requirements) and pass successfully.

Example test class for PropertyHandler_LWC:

```
@isTest
private class PropertyHandler_LWC_Test {
    @isTest
    static void testGetProperty() {
        // Set up test data: create sample properties with different types and statuses
        Property_c testProperty1 = new Propertyc(Typec='Commercial', Verified_c=true);
        Property_c testProperty2 = new Propertyc(Typec='Residential', Verified_c=false);
        insert new List<Property__c>{testProperty1, testProperty2};

        // Test fetching verified Commercial properties
        List<Property__c> results = PropertyHandler_LWC.getProperty('Commercial', true);
        System.assertEquals(1, results.size());
        System.assertEquals('Commercial', results[0].Type__c);
        System.assertEquals(true, results[0].Verified__c);
    }
}
```

2. User Interface (UI) Testing

UI testing focused on validating that the end-to-end user experience, interaction flows, and visual elements worked as expected across different roles (Admin, Sales Manager, Sales Executive, Customer).

Key Areas of UI Testing:

Property Details App Page:

Verify that the "Search Your Property" page loads correctly for each user role.

Test the search functionality to ensure users can filter properties based on type and verified status.

Confirm that role-based access is respected: Admins and Sales Managers can view all properties, while Customers can only access properties they're permitted to view.

Lightning Web Component (LWC) for Property Search:

Test that users can select and filter property type (Commercial, Residential, Rental)and verified status without issues.

Confirm that verified customers see only verified properties, while non-verified customers see only non-verified properties.

Verify the data fetch and display functionality, ensuring the component displays search results dynamically and accurately in a responsive table format.

Jotform Integration:

Test that Jotform submissions are correctly mapped to the Customer Object, with fields populated accurately in Salesforce.

Confirm that new customer entries appear for Admins without any data inconsistencies or delays.

UI Test Scenarios:

Login and Role-Specific Access:

Test logins for each user role (Admin, Sales Manager, Sales Executive, Customer) and verify they can access only permitted features and records.

Approval Process UI:

For Sales Executives and Sales Managers, test the UI for property approval and rejection flows, verifying that the Verified status updates correctly based on the outcome.

Error Handling and Notifications:

Verify that any errors during search or data entry are handled gracefully with clear notifications.

Confirm that users receive appropriate feedback (e.g., “No properties found” message when applicable) during searches with no matching results.

Tools for UI Testing:

Salesforce Lightning Testing Service (LTS) for automated testing of Lightning \components.

Manual Testing by test users simulating different roles to evaluate UI functionality, navigation, and data accuracy.

3. User Acceptance Testing (UAT)

Conducted User Acceptance Testing with stakeholders and end-users from each role (Admins, Sales Managers, Sales Executives, and Customers).

Gathered feedback on usability, navigation, and overall satisfaction with the UI, making adjustments as necessary.

4. Post-Deployment Validation

After deployment, additional validation tests were conducted in the live environment to ensure:

- ✓ All configurations, security settings, and workflows performed as expected.
- ✓ Real-time data accuracy and system stability in handling actual property management operations.
- ✓ This testing and validation plan ensures that the Property Management System meets business requirements, is user-friendly, and maintains data

accuracy and security across all roles and workflows.

6.Key Scenarios Addressed by Salesforce in the Implementation Project

- **Centralized Data Management**: Custom objects were created for Customer and Property records, allowing for organized storage of customer details and property information. This centralization streamlined data management and made reporting easier.

- **Automated Data Collection**: Integration with Jotform automated the intake of customer information, reducing manual data entry and errors while speeding up the process for admins to view and manage customer records.

- **Role-Based Access Control**: Salesforce roles and profiles were configured for Admins, Sales Managers, Sales Executives, and Customers, ensuring users have appropriate access based on their roles, enhancing security and data integrity.

- **Approval Workflow Automation**: An approval process was established for property records, allowing only verified properties to proceed and involving relevant roles in approvals. This ensured compliance and maintained data accuracy.

- **Automated Approval Submissions**: A Record Trigger Flow automated the submission of property records for approval, saving time and ensuring a consistent workflow.

- **Enhanced Search Functionality**: The creation of a Property Details App Page and a Lightning Web Component allowed users to search and filter properties by type and verification status, improving user experience and helping users find relevant information quickly.

- **Real-Time Data Fetching with Apex:** An Apex class enabled real-time property data fetching based on user inputs, ensuring that users received accurate and up-to-date information on property listings.

- **User-Specific Property Access**: Verified and non-verified customers were granted specific access to view properties based on verification status, ensuring a personalized experience while maintaining data privacy.

7.Conclusion

Summary of Achievements

Successfully developed a Salesforce-based Property Management System that effectively manages property data, automates approval workflows, and controls user access based on roles and verification.

Created custom objects, implemented automation, and leveraged Lightning Web Components (LWC) to provide a seamless user experience and improve operational efficiency.

Achieved significant improvements in data integrity, faster processing times for approvals, and streamlined management of properties and customer interactions.

Lessons Learned

Understanding the complexities of role-based access control is crucial to ensure data security and appropriate system access.

Incorporating client feedback during design and development phases helped in creating a solution tailored closely to business needs, leading to a more refined final product.

Effective project management, from requirement gathering to testing, is essential in handling complex automation and custom development in Salesforce

Future Enhancements

Implement advanced analytics and reporting to provide insights on property performance and trends.

Introduce AI-driven recommendations for property management to optimize rental pricing and suggest property improvements.

Expand mobile functionality, enabling easier access and management of properties for users on the go.