NAAN MUDHALVAN- SALESFORCE REPORT

CRM APPLICATION TO HANDLE THE CLIENTS AND THEIR PROPERTY RELATED REQUIREMENTS .

PROJECT CREATED BY:

Priyabhuvani S - (422422104019)

Vinitha M - (422422104020)

Jagan S - (422422104021)

Sathya R - (422422104022)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

COLLEGE CODE-4224

ANNA UNIVERSITY,CHENNAI-600025

TABLE OF CONTENTS:

CRM APPLICATION TO HANDLE THECLIENTSAND THEIR PROPERTY RELATED REQUIREMENTS

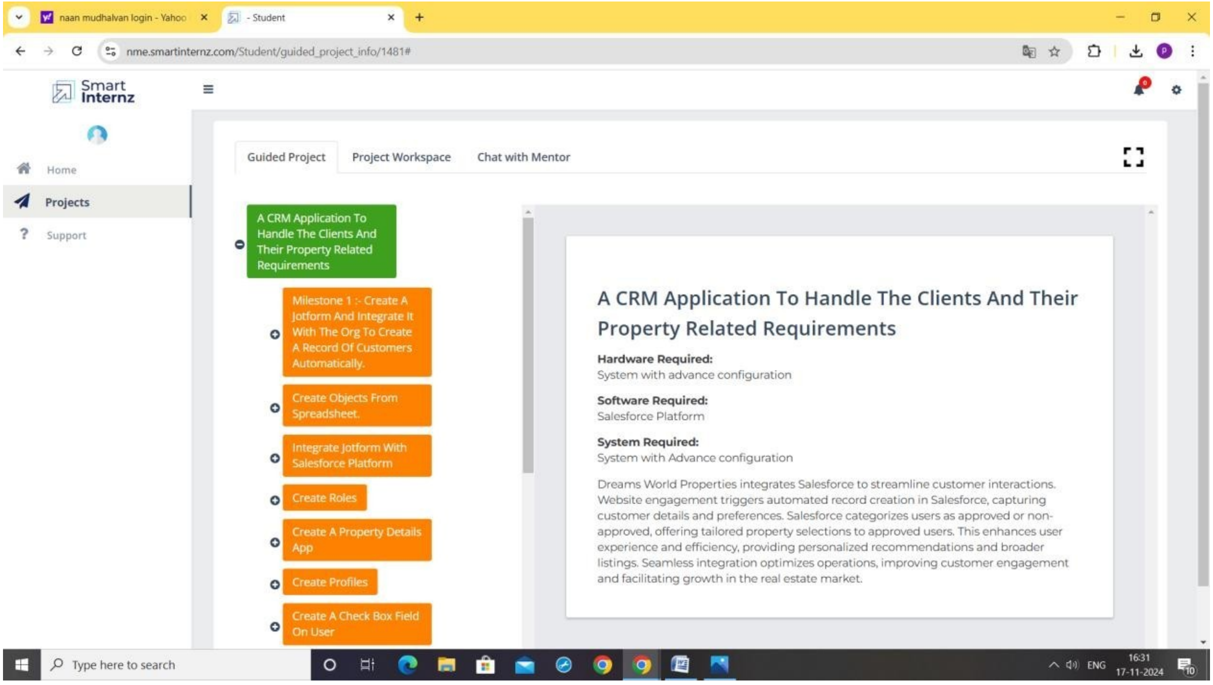
1. PROJECT OVERVIEW:

This projectfocuses on developing a CRM application usingSalesforce, enhanced with JotForm integration, to manageclients and their property-related requirements. Thesolution streamlines client interactions, centralizes propertydata, and automates workflows. JotForm is utilized forcollecting client data through customizable forms, whichintegrate seamlessly into Salesforce to ensure accurate andefficient dataprocessing.

2. PROJECT OBJECTIVES:

- CentralizedDataManagement.
- WorkflowAutomation.
- SeamlessDataCollectionvia JotForm.
- EnhancedClientInteraction.
- ScalabilityandSecurity.
- Enhancecustomersatisfactionthroughefficientdatacollectionandcommunication.

FIGURE1:



3.KEY FEATURES AND CONCEPTS:

- **Apex Triggers:** Automated logic to synchronize account data,esuring accuracy in client details and property preferences.
- **Custom Objects and Fields:** Designed for client and property data storage and management.
- **JotForm Integration:** Enabled efficient data collection through user-friendly forms,directly populating Salesforcerecords.
- **Process Builder and Workflows**: Automated property matching and email notifications based on data received from JotForm.
- **Process Builder**: Designed workflows to match properties to client requirements automatically.
- **Reports and Dashboards**: Provided actionable insights on property trends and client activities.
- **Role-BasedAccessControls:**Ensured data security and privacy.

4.DETAILED STEPS TO SOLUTION DESIGN:

- **Custom Object Creation:** Created objects for clients, properties, and transactions with appropriate fields.
- **Integration with JotForm:** Linked JotForm to Salesforce for automatic client data import.
- **Workflow Automation:** Developed workflows using Process Builder to match properties with client requirements and send alerts.
- **UI Customization:** Designed user-friendly layouts and Lightning components form an aging data efficiently.
- **Security Implementation:** Applied profiles, roles, and sharing rules to protect sensitive client and property data.
- **Testing:** Conducted unit, system, and user acceptance testing to valid a workflows and integrations.

5.TESTING AND VALIDATION:

- **Unit Testing:** Verified individual components like Apex triggers and workflows.
- **System Testing:** Ensured seamless integration of JotForm and Salesforce workflows.
- **User Acceptance Testing (UAT):** Gathered feedback from test users to refine the system.
- **Performance Testing:** Ensured scalability under increased client and property data loads.

6.KEY SCENARIOS ADDRESSED BY JOTFORM AND SALESFORCE INTEGRATION:

- **Efficient Client Management:** Centralized client profiles with detailed property preferences an dinteraction.

- **Automated Property Matching:** Leveraged automation to match client requirements with suitable properties instantly.

- **Real-Time Updates:**Enabled clients to receive updates on property availability and transaction status.

- **Streamlined Data Collection:** Used JotForm for error-free and efficient data input directly into Salesforce.

- **Insights and Analytics:** Dashboards offered key metrics like property availability trends and client activity.

7.STEP BY STEP PROCESS EXPLANATION:

STEP 1: CREATE A JOTFORM AND INTEGRATE IT WITH A ORG TO CREATE A RECORD OF CUSTOMERS AUTOMATICALLY

Client wants a form for the customers to get the details directly into the salesforce so that the admins can create a user in the org.

FIGURE2:

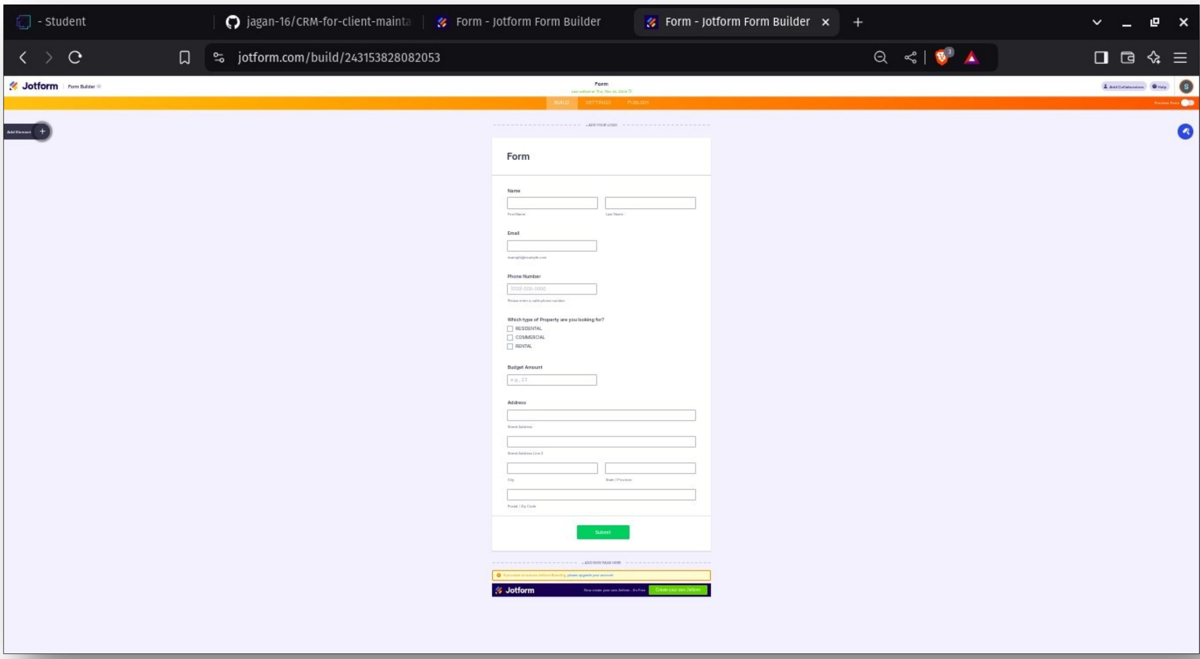


FIGURE3:

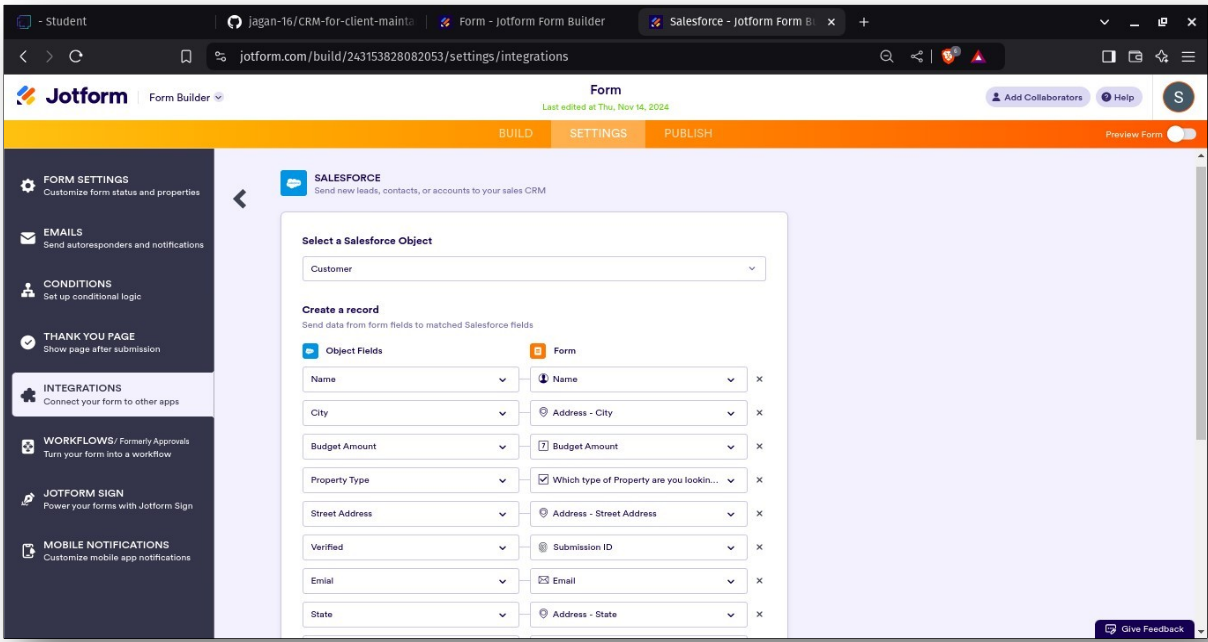
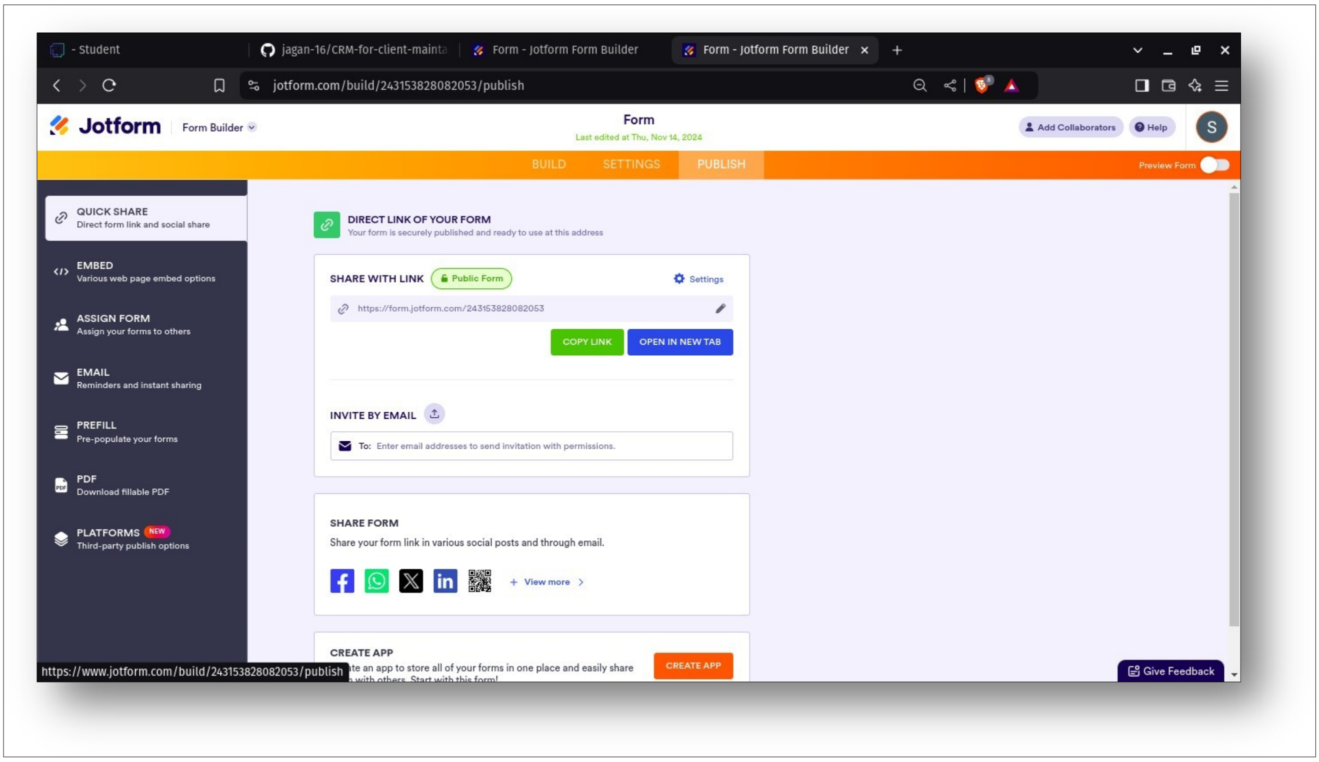


FIGURE4:



STEP2:CREATE OBJECTS FROM SPREADSHEET.

Directly Creating Objects from Spreadsheet in Salesforce.

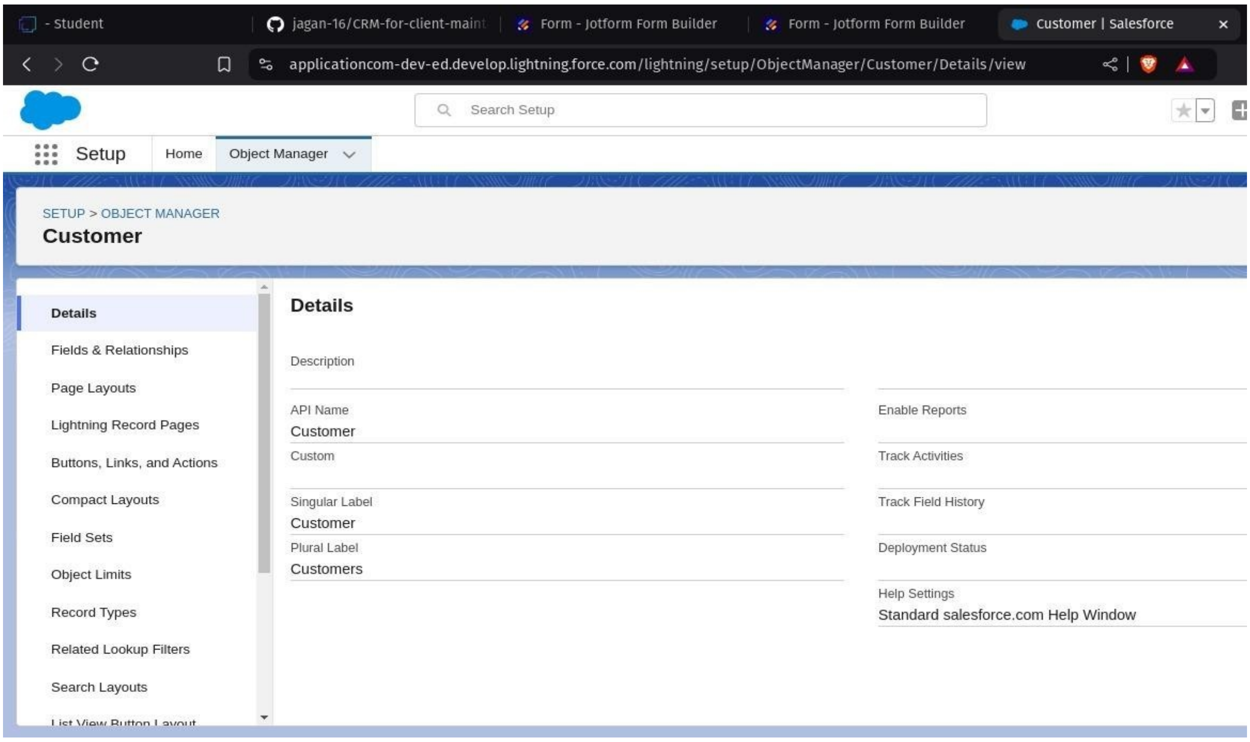
Create Customer object

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

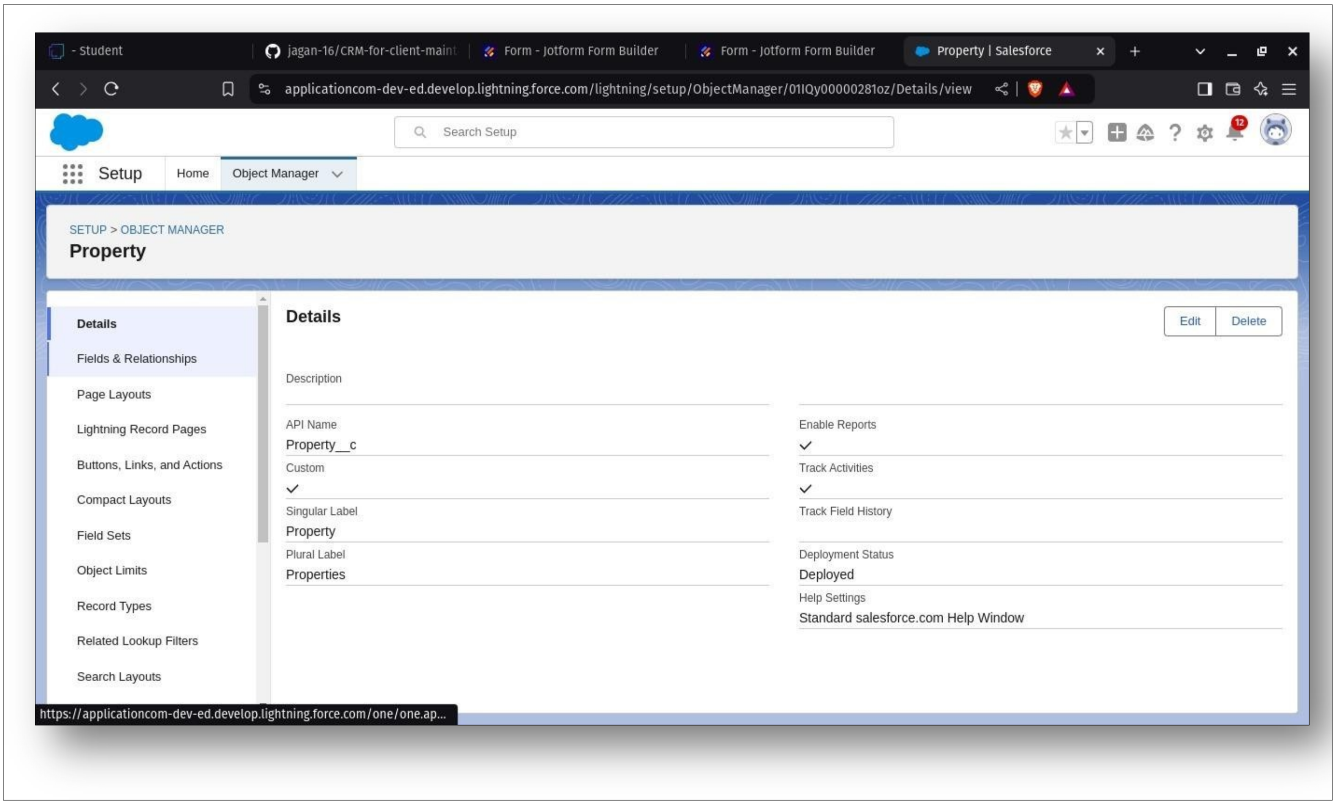
Create Property object

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

FIGURE5:



- Student
jagan-16/CRM-for-client-maint
Form - Jotform Form Builder
Form - Jotform Form Builder
Customer | Salesforce
applicationcom-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/Customer/Details/view
Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Customer
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
Details
Description
API Name
Customer
Custom
Singular Label
Customer
Plural Label
Customers
Enable Reports
Track Activities
Track Field History
Deployment Status
Help Settings
Standard salesforce.com Help Window

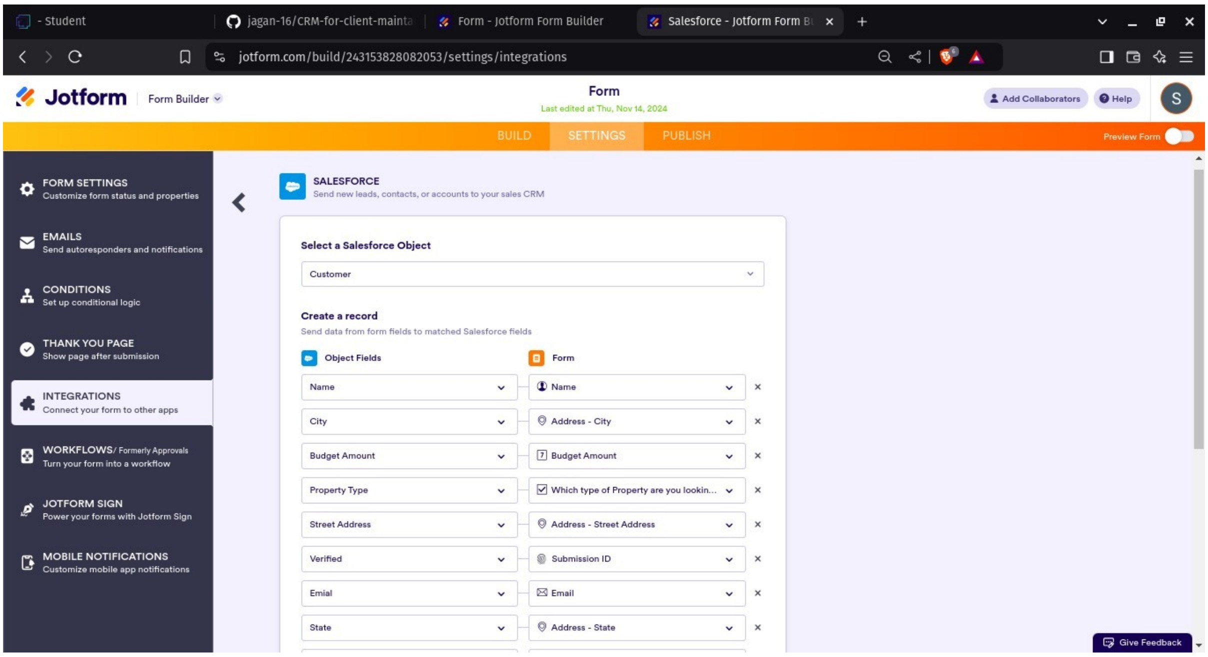


- Student
jagan-16/CRM-for-client-maint
Form - Jotform Form Builder
Form - Jotform Form Builder
Property | Salesforce
applicationcom-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01IQy00000281oz/Details/view
Search Setup
Setup
Home
Object Manager
SETUP > OBJECT MANAGER
Property
Details
Fields & Relationships
Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters
Search Layouts
Details
Edit
Delete
Description
API Name
Property__c
Custom
Singular Label
Property
Plural Label
Properties
Enable Reports
Track Activities
Track Field History
Deployment Status
Deployed
Help Settings
Standard salesforce.com Help Window
https://applicationcom-dev-ed.develop.lightning.force.com/one/one.ap...

STEP 3: INTEGRATE JOTFORM WITH SALESFORCE PLATFORM

In this Milestone we are going to integrate jotform with Salesforce

FIGURE6:

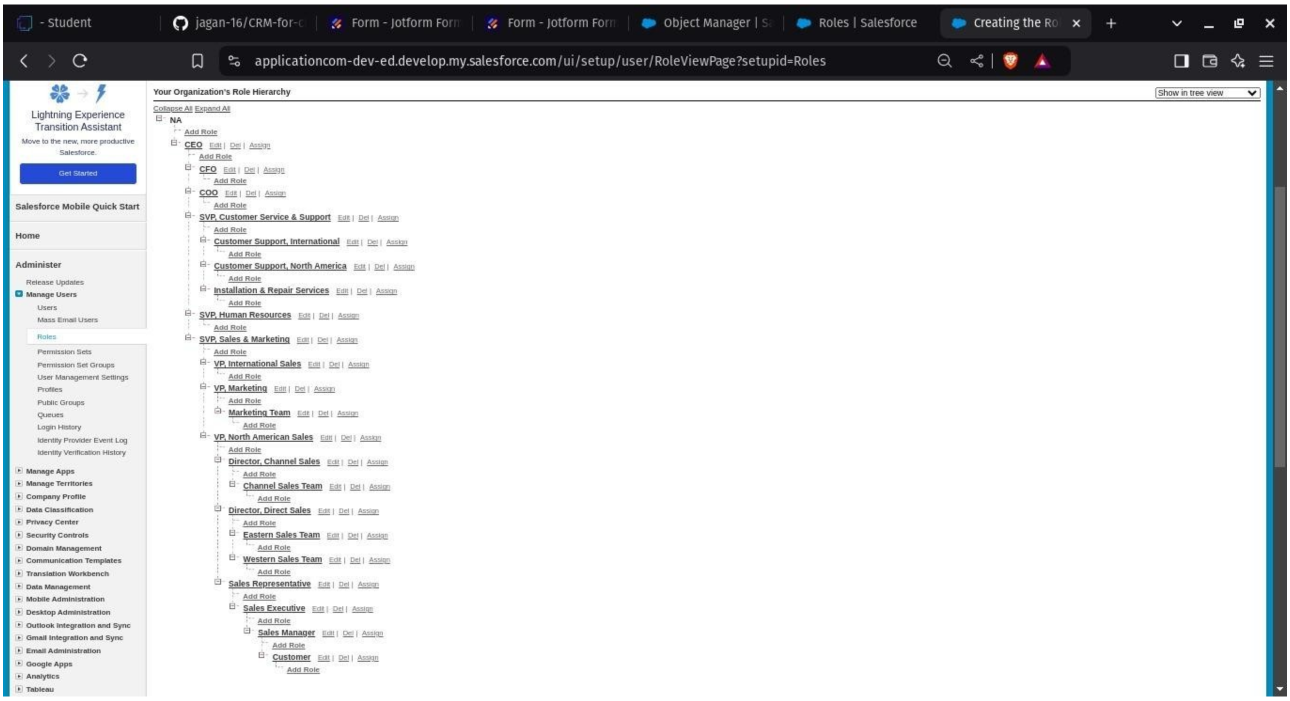


STEP4:CREATE ROLES

Create Roles as per business requirenment Steps:

- Go to Setup and Click on Roles, then click on Expand all and Add a Role just below the Sales Representative. * It will use the “System Administrator Profile”.
- Label-Sales Executive
- Reports to -Sales Representative
- Similarly Create a Role Name “Sales Manager” below Sales Executive which reports to Sales Executive, Also Add a Role below SalesManager labeled as “Customer” which reports to SalesManager.

FIGURE7:

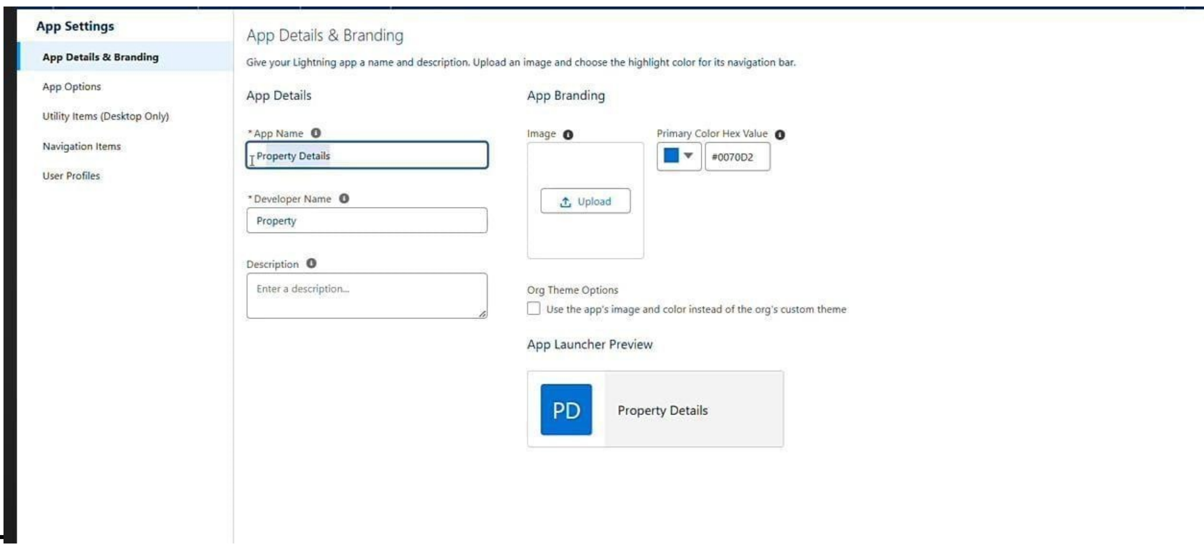


STEP5:CREATE A PROPERTY DETAILS APP

An App where the objects will be displayed Steps:

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

FIGURE8:



STEP6:CREATE PROFILES

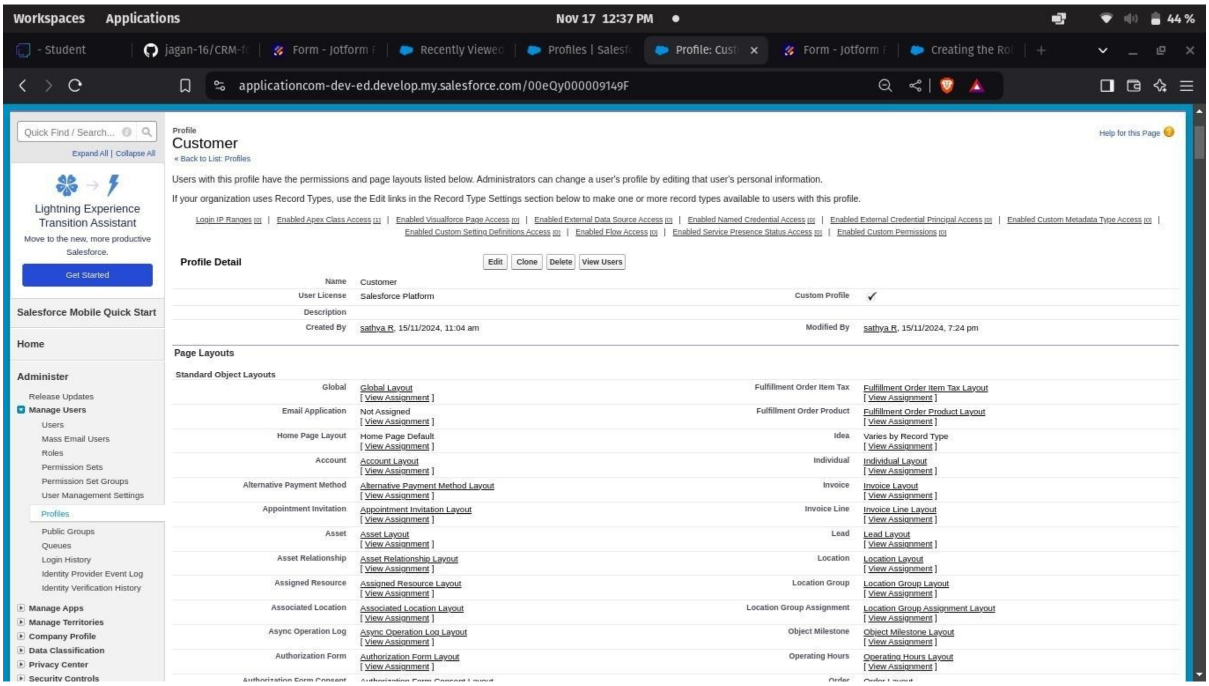
Create profiles as per business requirement

Customer:

Steps:

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

FIGURE9:



Manager:

Steps:

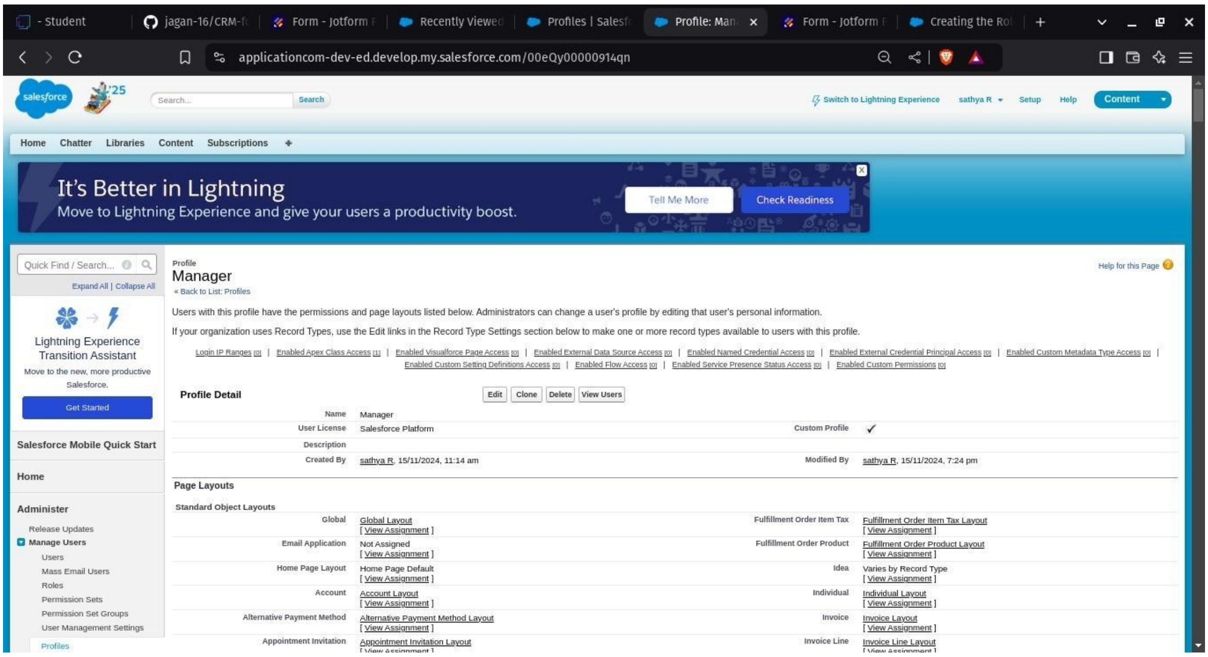
1. From Setup>> Go to Profiles and Clone Salesforce Platform User and Name it “Manager”..
2. Uncheck all the Custom Objects and Check only Property Details

From Custom App Settings.

3. Also Remove all the Standard Object Permissions.

4. Uncheck all the Custom Object Permissions and check only“modify all” from “Property” and “Customer”

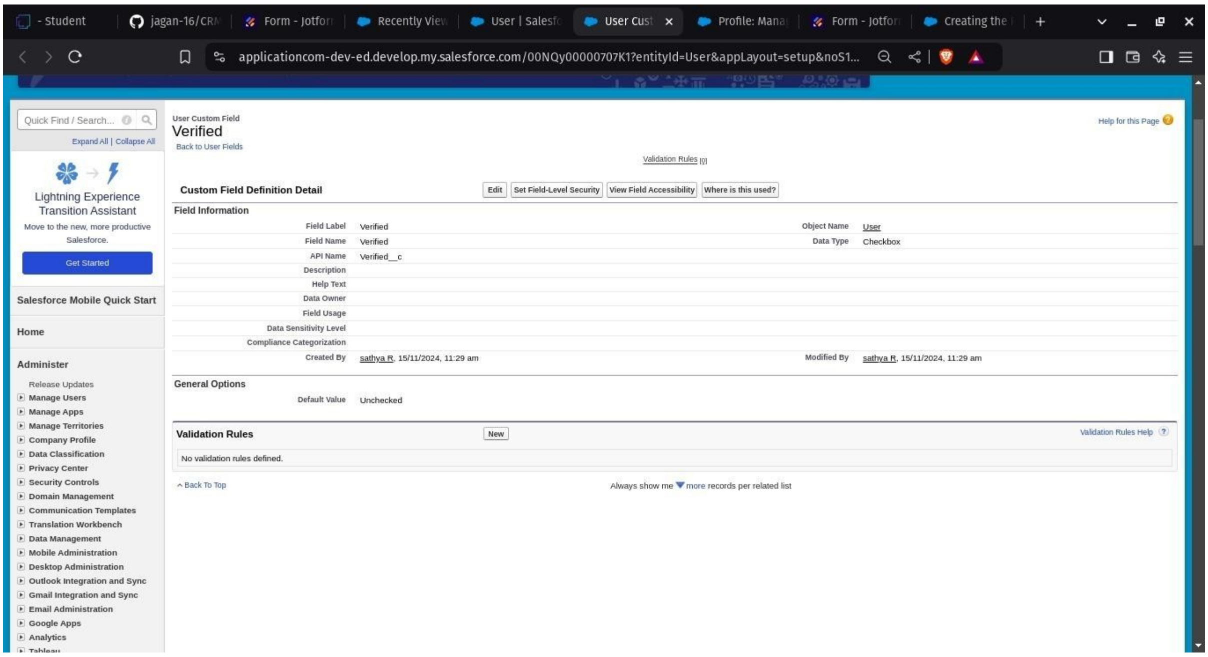
FIGURE10:



STEP7:CREATE A CHECKBOX FIELD ON USER

Create Field on the User as per the business requirement.

FIGURE11:

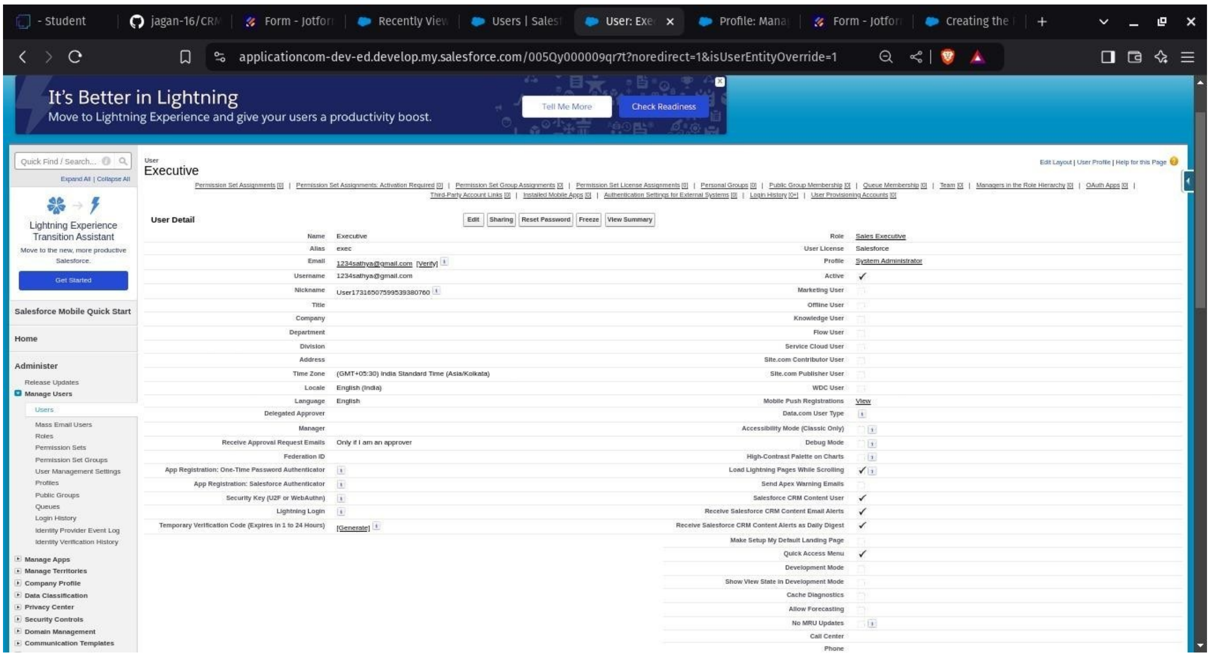


STEP8:CREATE USERS

USER1:

- GotoSetup-->Administration--> Users-->NewUser
- LastName- Executive
- Role- SalesExecutive
- License-Salesforce
- Profile - System Administrator
- 6.Save

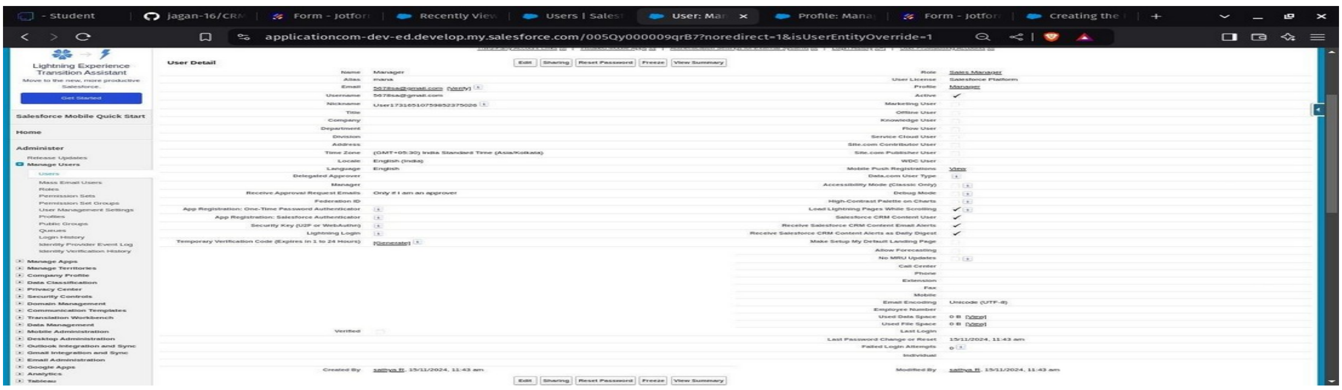
FIGURE12:



USER2:

- GotoSetup>>Administration>>Users>>NewUser
- LastName>>Manager
- Role>>SalesManager
- License>>Salesforce Platform
- Profile>>Manager
- Save

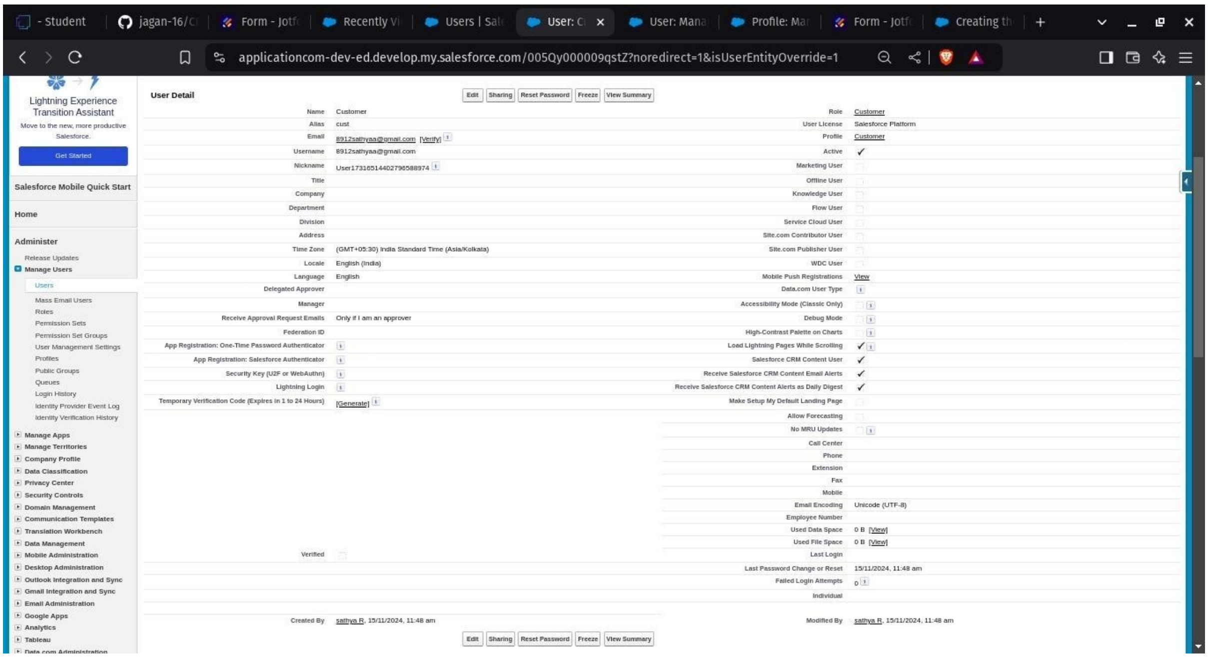
FIGURE13:



USER3:

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

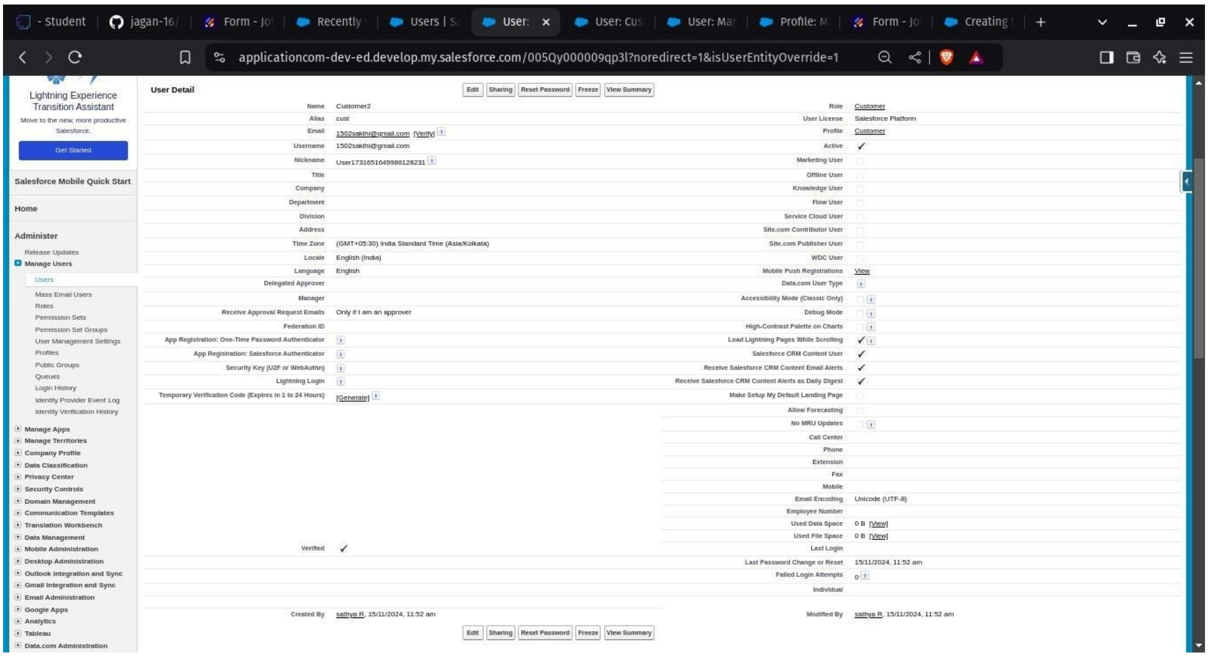
FIGURE:14



USER4:

- Goto Setup>>Administration>>Users>>New User
- Last Name>>Customer2
- Role>>Customer
- License>>Salesforce Platform
- Profile>>Customer
- Make Sure the verified checkbox is “checked”

FIGURE15:



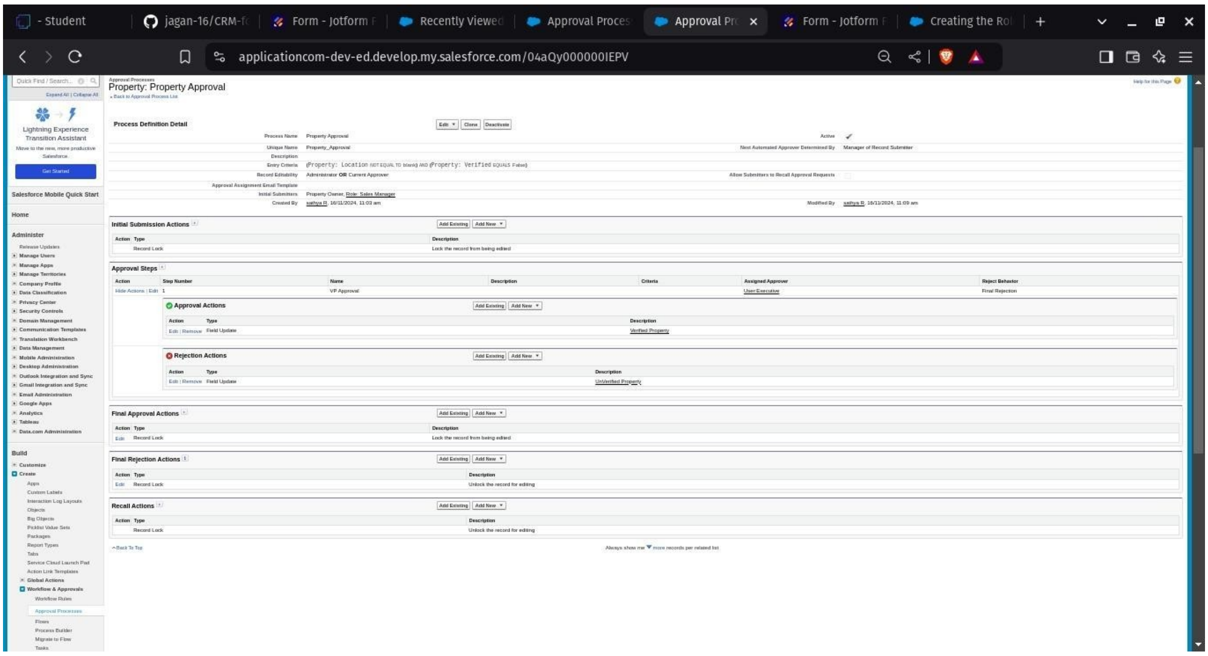
STEP 9: CREATE AN APPROVAL PROCESS FOR PROPERTY OBJECT

Steps:

- From Setup>>Process Automation>>Approval Process
- Process Name -PropertyA pproval
- Location is not equal to blank
- Verified Equals false.
- Click next and “Next Automated Approver Determined By Select Manager.
- From Record Edit ability Properties>>Click on
- Administrators OR the currently assigned approver can edit records during the approval process.
- From Step 5 Select Fields to Display on Approval Page Layout select Property, Owner, Location,Type.
- Click Next and Select the initial Submit
- Owner>>Property Owner
- Roles>>Sales Manager
- Save
- Add an approval step name “Executive Approval”.
- Specify the Criteria >>All record should enter.

- click next and select the Approver as “ Sales Executive “ and “Save”.
- Add One field Update as “UnVerified Property” Select Object>>Property
- Field to Update >> Verified Field DataType>>CheckBox
- Select CheckBox Option as “False” Save.
- Add One field Update as “UnVerified Property” Select Object>> Propert Field to Update>>Verified
- Field Data Type >> CheckBox
- Select CheckBox Option as “False”
- Activate the approval process

FIGURE16:



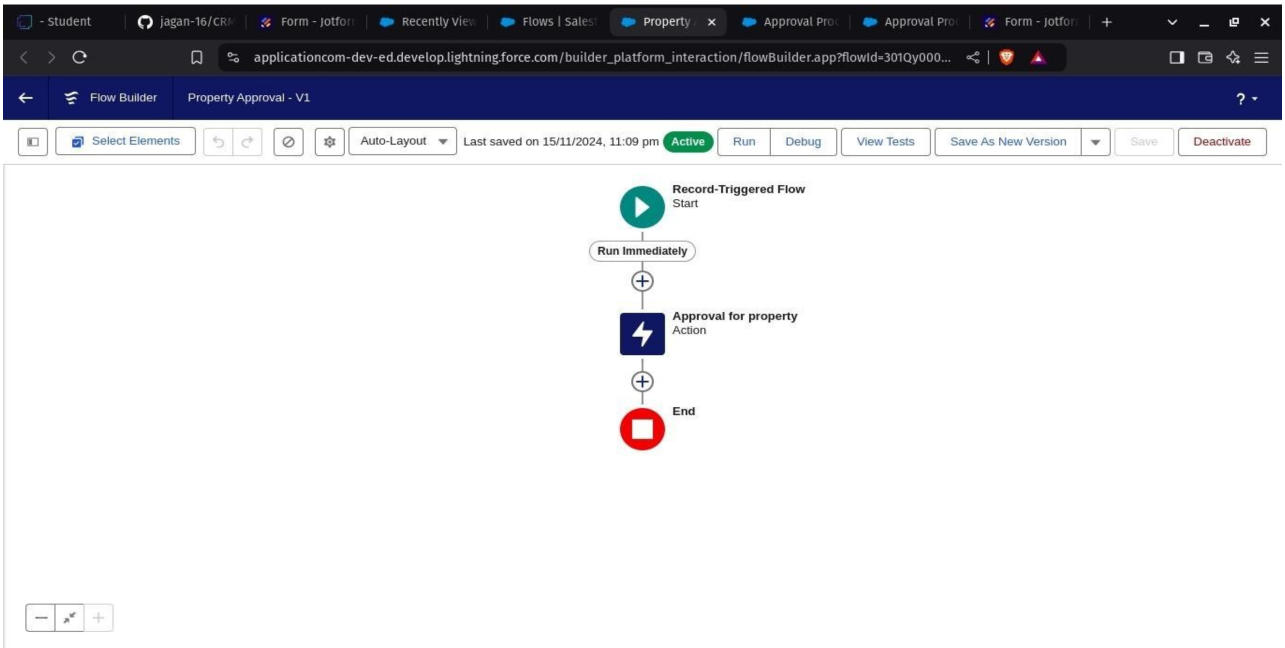
STEP10:CREATE A RECORD TRIGGER FLOW TO SUMBIT THE APPROVAL PROCESS AUTOMATICALLY

A flow that can submit the records directly for approval

Steps:

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

FIGURE17:



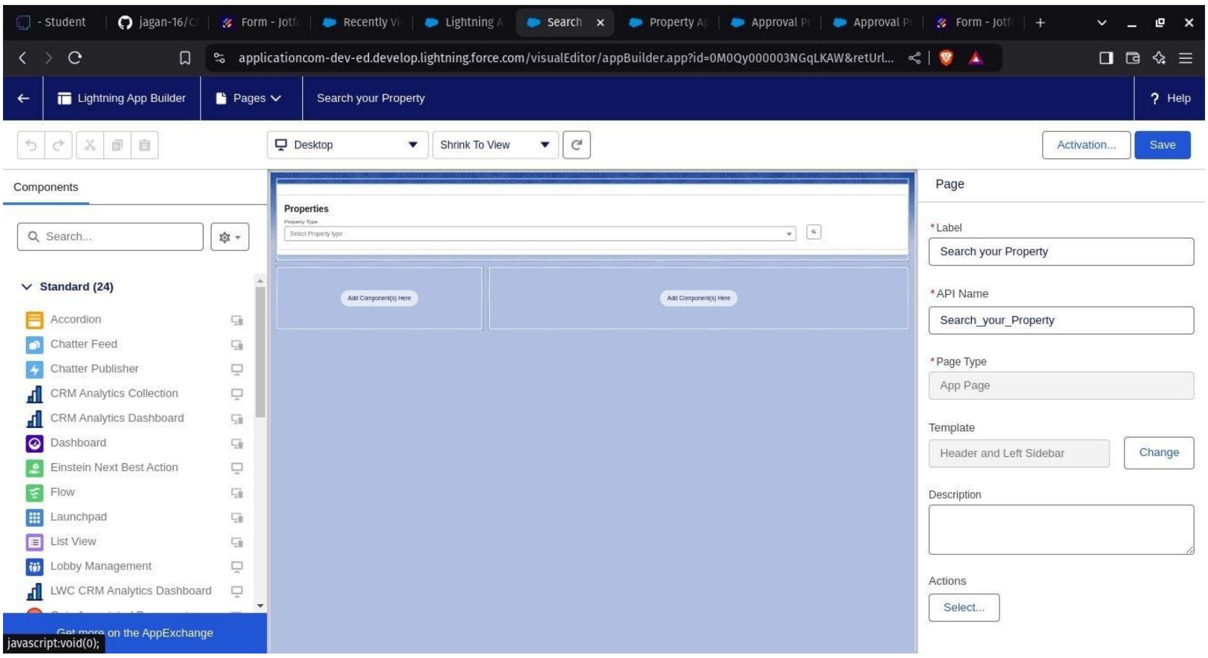
STEP11:CREATE AN APP PAGE

Create an App Page on the Property detailsObject named as “Search Your Property”

STEPS:

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

FIGURE18:



STEP12:CREATE A LWC COMPONENT

Create an LWC Component for the customers so that only verified customers can access the verified properties and non Verified customers can access non verified properties, and deploy it on"Search your Property Page"

STEPS:

1.Create an Apex Class and make it aura enabled and name it "PropertHandler_LWC"

CODE:

```
public class PropertHandler_LWC{

   @AuraEnabled(cacheable=true)

   public static list<Property__c> getProperty(string type , boolean verified){

       return [SELECT Id, Location__c, Property_Name__c, Type__c, Verified__c FROM Property__c Where
Type__c =: type AND Verified__c =: verified];


   }

}
```

CODE:HTML

```
<template>
        <lightning-card>
          <div class="slds-box">
            <div class="slds-text-align_left">
              <h1style="font-size:20px;"><b>Properties</b></h1>
            </div>
            <div>
              <div class="slds-gridslds-gutters">
                <div class="slds-colslds-size_5-of-6">
                  <lightning-combobox name="Type" label="Property Type" value={type
       var}placeholder="Select Property type"
                    options={propety options} on change={changehandler}></lightning-combobox>
                </div>
                <div class="slds-colslds-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={is true}>
         <div class="slds-box">
          <lightning-datatable key-field="id" data={property
      list}columns={columns}></lightning-datatable>
         </div>
        </template>
        <template if:false={is false}>
         <div class="slds-box">
          <div style="font-size:15px;"><b>No properties Are Found!!</b></div>
        </div>
        </template>
       </lightning-card>
      </template>
```

In Your Js File Write this code :

CODE:JS

```
import{LightningElement,api,track,wire}from'lwc';
import getProperty from
"@salesforce/apex/PropertHandler_LWC.getProperty"import{ getRecord}
from'lightning/uiRecordApi';
importUSER_IDfrom'@salesforce/user/Id';
export default class C_01_Property_Management extends LightningElement
  {@apirecordId
  userId =
  USER_ID;verified
  vartypevar
  isfalse =
  true;istrue=fal
  se;
  @track propertylist =
```

```
[];columns=[
   {label:'PropertyName',fieldName:'Property_Namec'},
   {label:'PropertyType',fieldName:'Typec'},
   {label:'PropertyLocation',fieldName:'Locationc'},
   {label:"Propertylink",fieldName:"Property_linkc"}
]
propetyoptions=[
   {label:"Commercial",value:"Commercial"},
   {label:"Residential",value:"Residential"},
   {label:"rental",value:"rental"}


]
@wire(getRecord, { recordId: "$userId", fields:
['User.Verifiedc'] })recordFunction({data,error}) {
   if (data)
      {console.log(data
      )
      console.log("This is the User Id --->
      "+this.userId);this.verifiedvar=data.fields.Verifiedc.
      value;
   } else {
      console.error(error)console.log('th
      isiserror')
   }
}


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(res
         ult)
         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.verified
            var);console.log(this.typevar)
         } else {
            this.is false =
            false;
            this.is true=false;
         }
```

```
            })
            .catch((error) =>
               {console.log(error)
            })
        }
    }
```

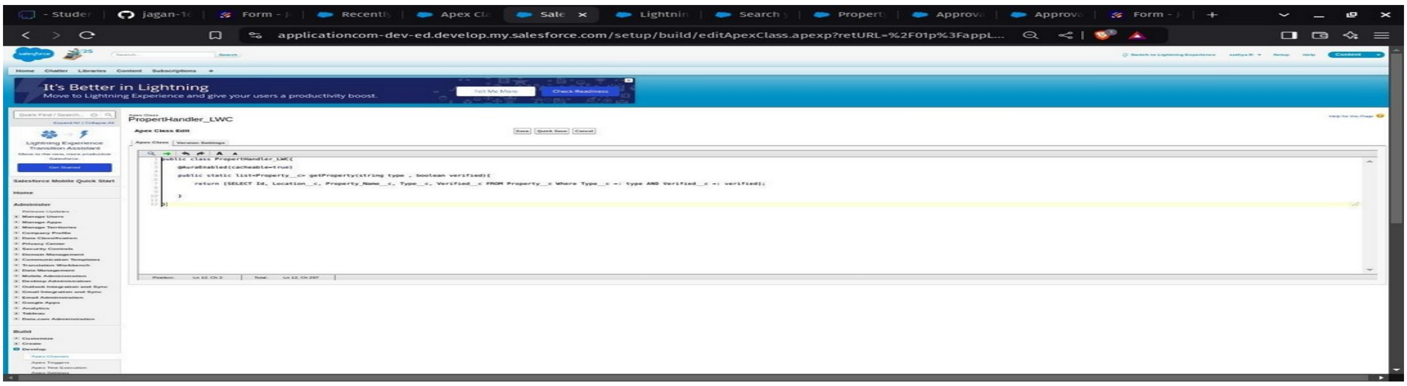
In Your meta file give your targets to deploy the component.

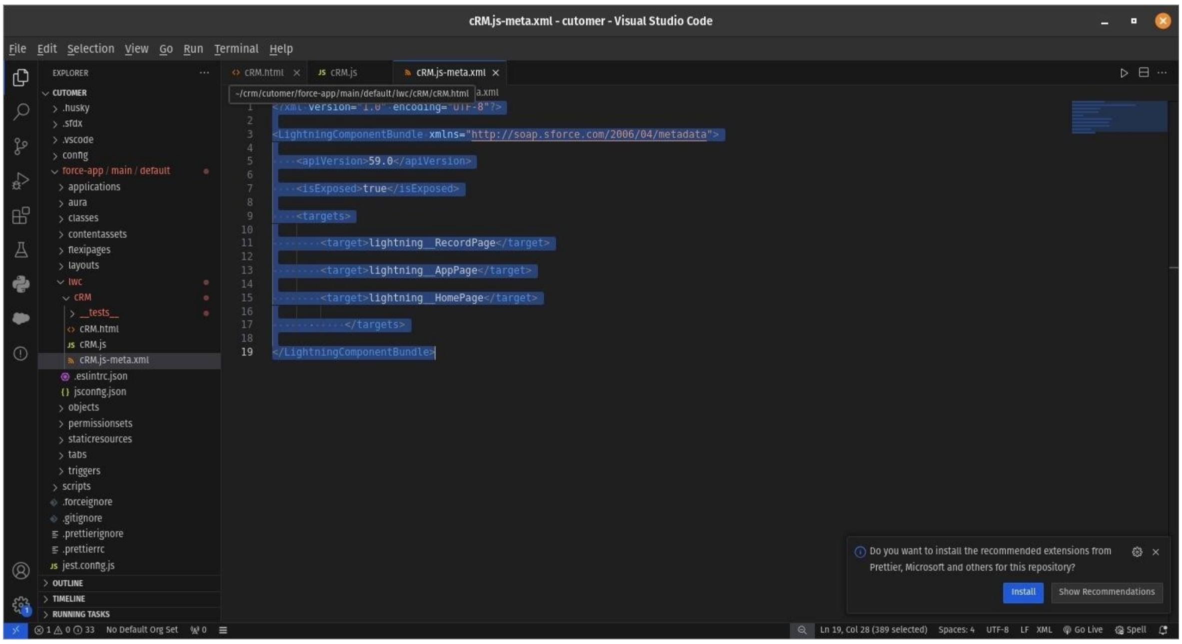
CODE:js-meta.xml

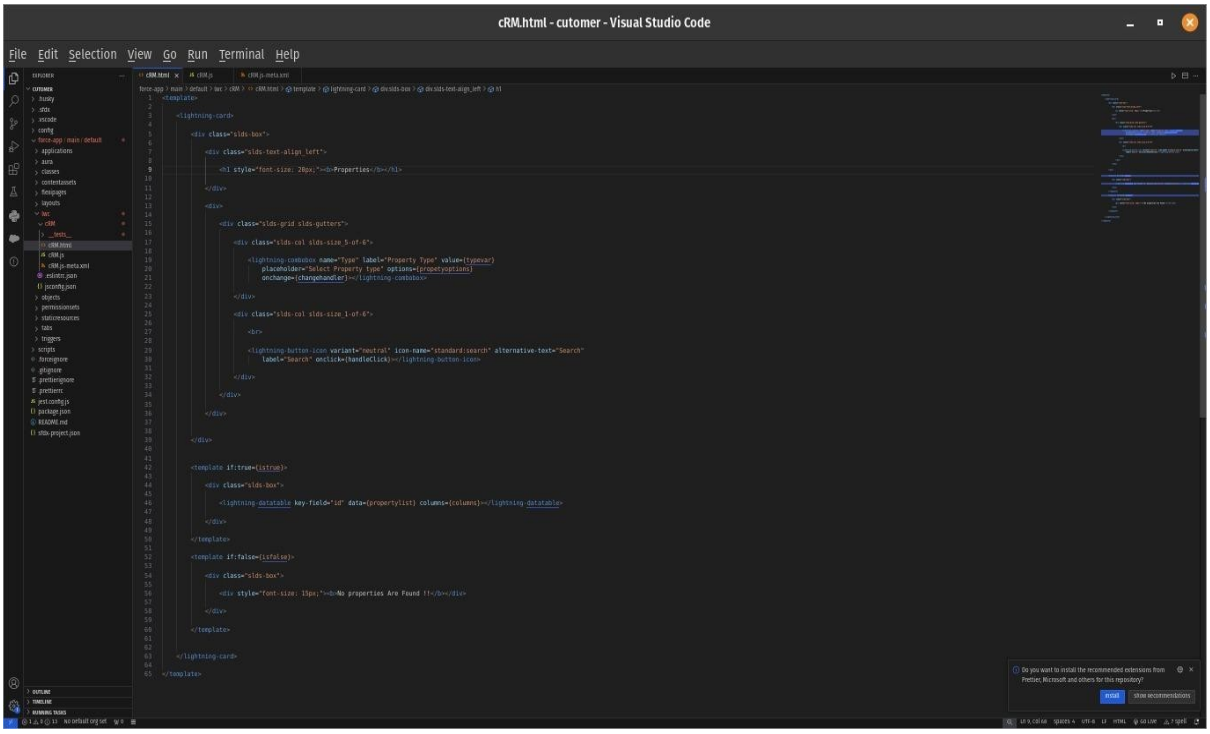
```
<?xmlversion="1.0"encoding="UTF-8"?>
<LightningComponentBundlexmlns="http://soap.sforce.com/2006/04/metadata">
   <apiVersion>59.0</apiVersion>
   <isExposed>true</isExposed>
   <targets>
      <target>lightningRecordPage</target>
      <target>lightningAppPage</target>
      <target>lightningHomePage</target>


         </targets>
</LightningComponentBundle>
```

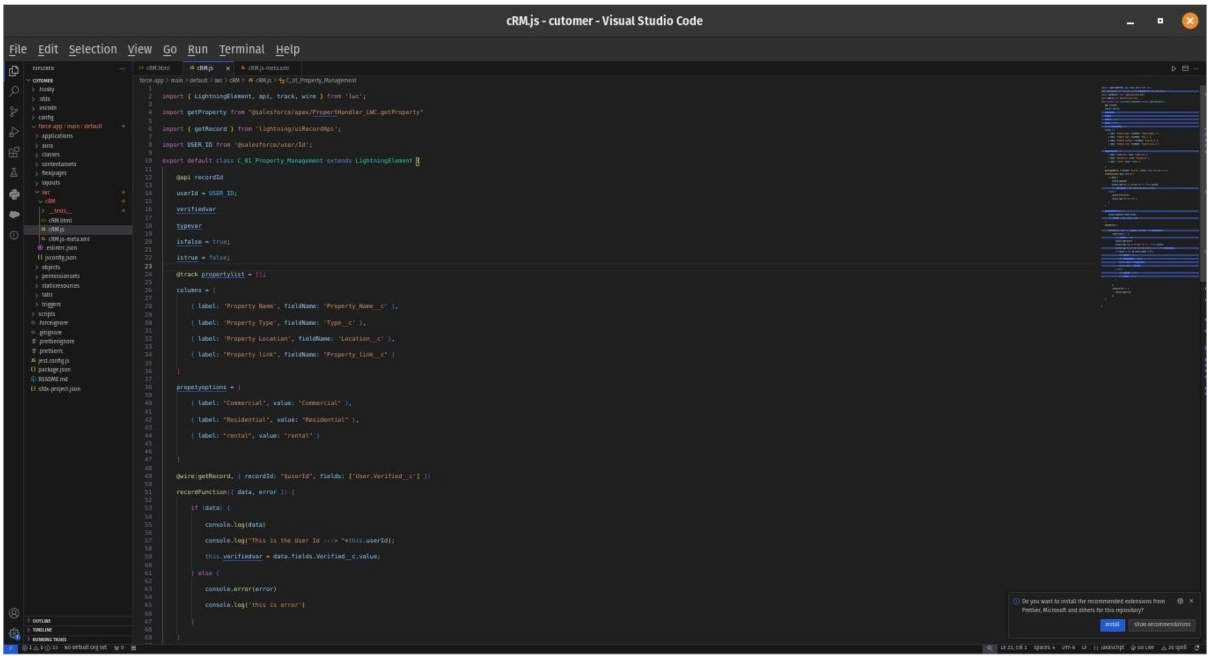
After Saving all the three Codes , Right Click and deploy this component to the org.

FIGURE19:




cRM.js-meta.xml - cutomer - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
CUTOMER
.husky
.sfdx
.vscode
config
force-app / main / default
applications
aura
classes
contentassets
flexipages
layouts
lwc
cRM
__tests__
cRM.html
cRM.js
cRM.js-meta.xml
.eslintrc.json
jsconfig.json
objects
permissionsets
staticresources
tabs
triggers
scripts
.forceignore
.gitignore
.prettierignore
.prettierrc
jest.config.js
OUTLINE
TIMELINE
RUNNING TASKS
cRM.html
cRM.js
cRM.js-meta.xml
~/crm/cutomer/force-app/main/default/lwc/cRM/cRM.html
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>59.0</apiVersion>
<isExposed>true</isExposed>
<targets>
<target>lightning__RecordPage</target>
<target>lightning__AppPage</target>
<target>lightning__HomePage</target>
</targets>
</LightningComponentBundle>
Do you want to install the recommended extensions from Prettier, Microsoft and others for this repository?
Install
Show Recommendations
No Default Org Set
Ln 19, Col 28 (389 selected)
Spaces: 4
UTF-8
LF
XML
Go Live
Spell


cRM.html - cutomer - Visual Studio Code
File Edit Selection View Go Run Terminal Help



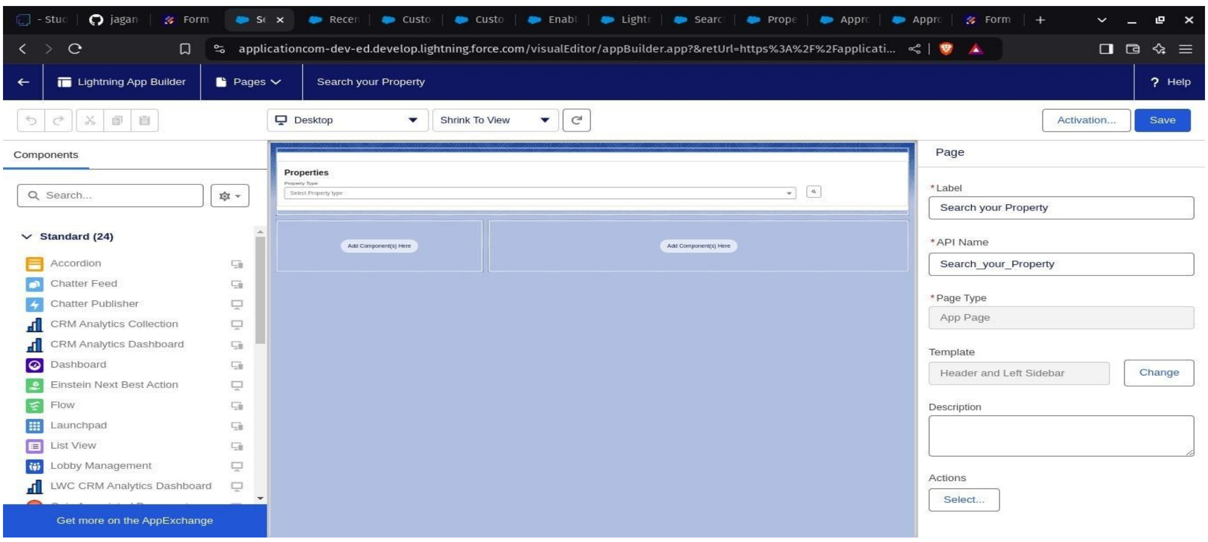
STEP13:DRAG THIS COMPONENT TO YOUR APP PAGE

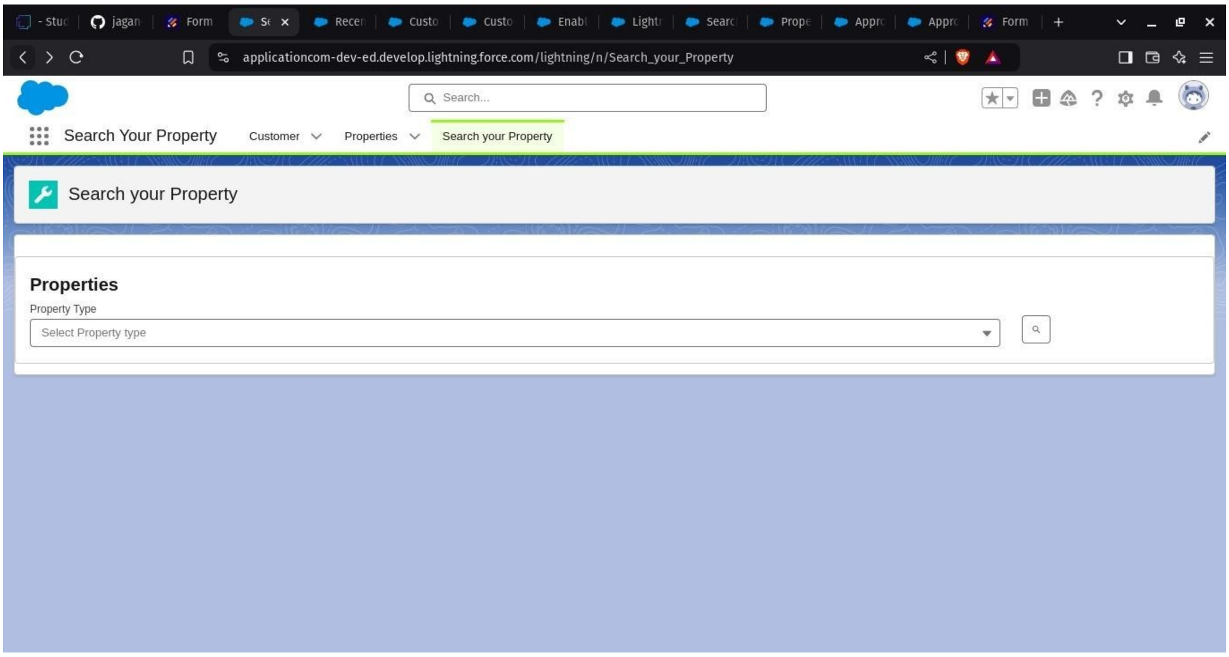
Adding the component to your page.

STEPS:

- From Setup>>Go to App Launcher>>Search for Property Details
- On this Page click on gear icon and clickon Edit Page
- Drag the Component to your App Page and Save the Page.

FIGURE20:





STEP14:GIVE ACCESS OF APEX CLASSES TO PROFILES

The Apex Class has a Security,Enable the security for the profiles that needs to access this class.

STEPS:

- From Setup >> Search For Apex Classes >> Click on “Security” behind “PropertyHandlerLWC”.
- From Profiles Add “Manager” and “Customer” and “Save”.

8. Conclusion

The CRM Property Management System successfully combines Salesforce and JotForm to address the complexities of managing client and property-related workflows. By centralizing data, automating processes, and enhancing communication, this solution has improved operational efficiency and client satisfaction. It is scalable, secure, and tailored for future business growth, making it an ideal tool for property management enterprises.