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

**By**

**Harika Karampuri**

**karampuriharika@salesforce.com**

**KAKATIYA INSTITUTE OF TECHNOLOGY AND SCIENCE FOR WOMEN**

**Project Abstract**

Dreams World Properties' Salesforce Integration Project aimed to streamline customer interactions, enhance user experience, and optimize operations by integrating Salesforce with the company website. The project successfully automated record creation, capturing customer details and preferences, and categorized users as approved or non-approved, offering tailored property selections to approved users. This resulted in improved customer engagement, increased efficiency, and facilitated growth in the real estate market, ultimately enhancing user experience through personalized property recommendations and optimized operations.

Furthermore, the integration enabled Dreams World Properties to gain valuable insights into customer behavior, preferences, and needs, allowing them to refine their marketing strategies and improve customer satisfaction. The project's success has positioned Dreams World Properties as a leader in the real estate industry, known for its innovative approach to customer relationship management.

**INDEX**

**S.NO TASK Pg.No**

1 Project Overview 4

2 Objectives 5-36

2.1 Task-1 5

2.2 Task-2 6-7

2.3 Task-3 8-10

2.4 Task-4 11

2.5 Task-5 12

2.6 Task-6 13-15

2.7 Task-7 16

2.8 Task-8 17-18

2.9 Task-9 19-25

2.10 Task-10 26-27

2.11 Task-11 28

2.12 Task-12 29-34

2.13 Task-13 35

2.14 Task-14 36

3 Salesforce Key Features and Concepts Utilized 37

4 Testing and Validation 38-39

5 Conclusion 40

# PROJECT OVERVIEW

A CRM Application to Handle Clients and their Property-Related Requirements is a cutting-edge technology solution designed to revolutionize the way property management companies interact with their clients and manage their properties. In today's fast-paced and highly competitive real estate market, property management companies need to stay ahead of the curve by providing exceptional customer service, efficiently managing client relationships, and streamlining property-related operations. However, many property management companies still rely on manual processes, spreadsheets, and outdated software systems, which can lead to inefficiencies, errors, and missed opportunities. To address these challenges, this project aims to design and develop a comprehensive CRM application that integrates all aspects of client and property management, including client onboarding, property listings, service requests, scheduling, and reporting. By leveraging the latest technologies and best practices in CRM development, this application will enable property management companies to deliver personalized services, improve client satisfaction, and increase operational efficiency, ultimately driving business growth and success in the competitive real estate market.

**TASK-1**

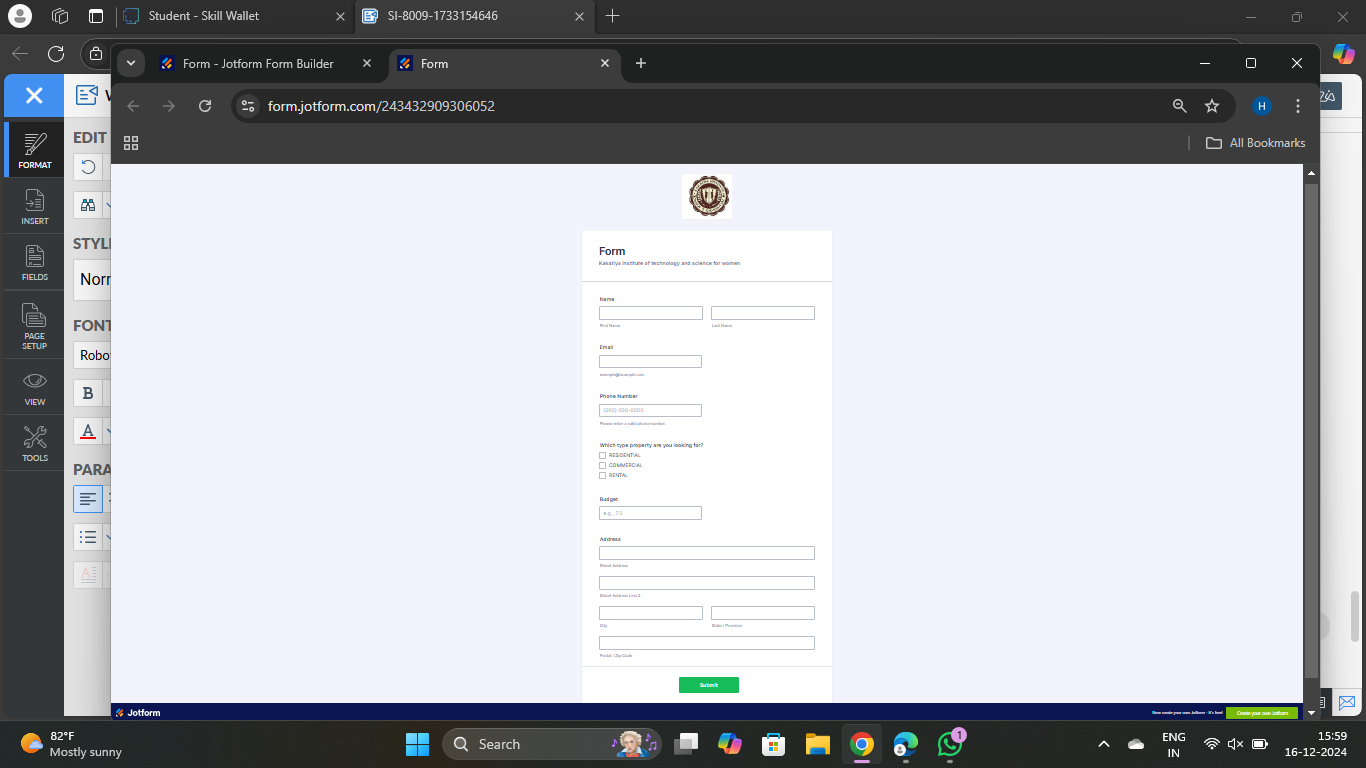
### **Create a Jotform and integrate it with the org to create a record of customers automatically.**

### 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.

**Activity-1**

Creating a Form to get the customer details like Name, Phone, Email, Address and type of property the customer is interested in.

https://form.jotform.com/243432909306052



**TASK-2**

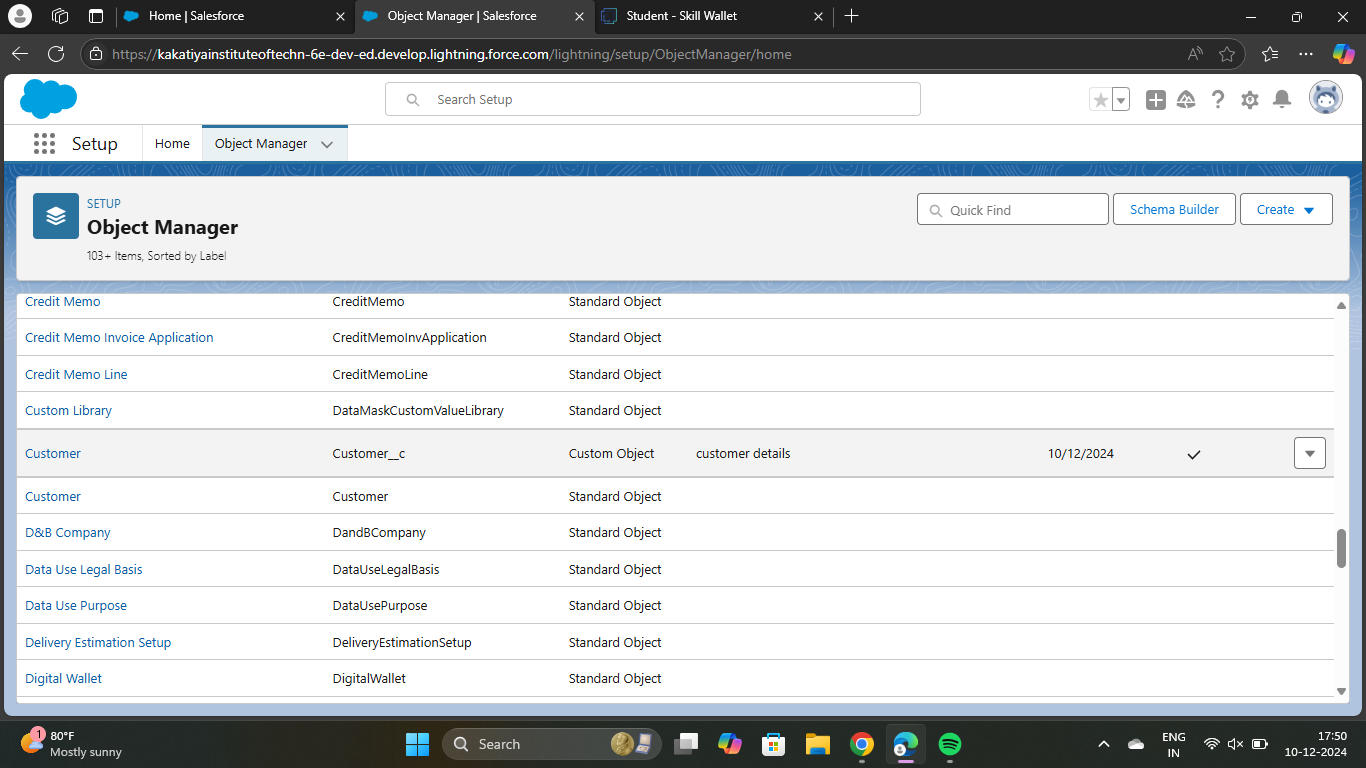
### **Create Objects from Spreadsheet.**

Directly Creating Objects from Spreadsheet in Salesforce.

### Create Customer object

Using the following Steps,

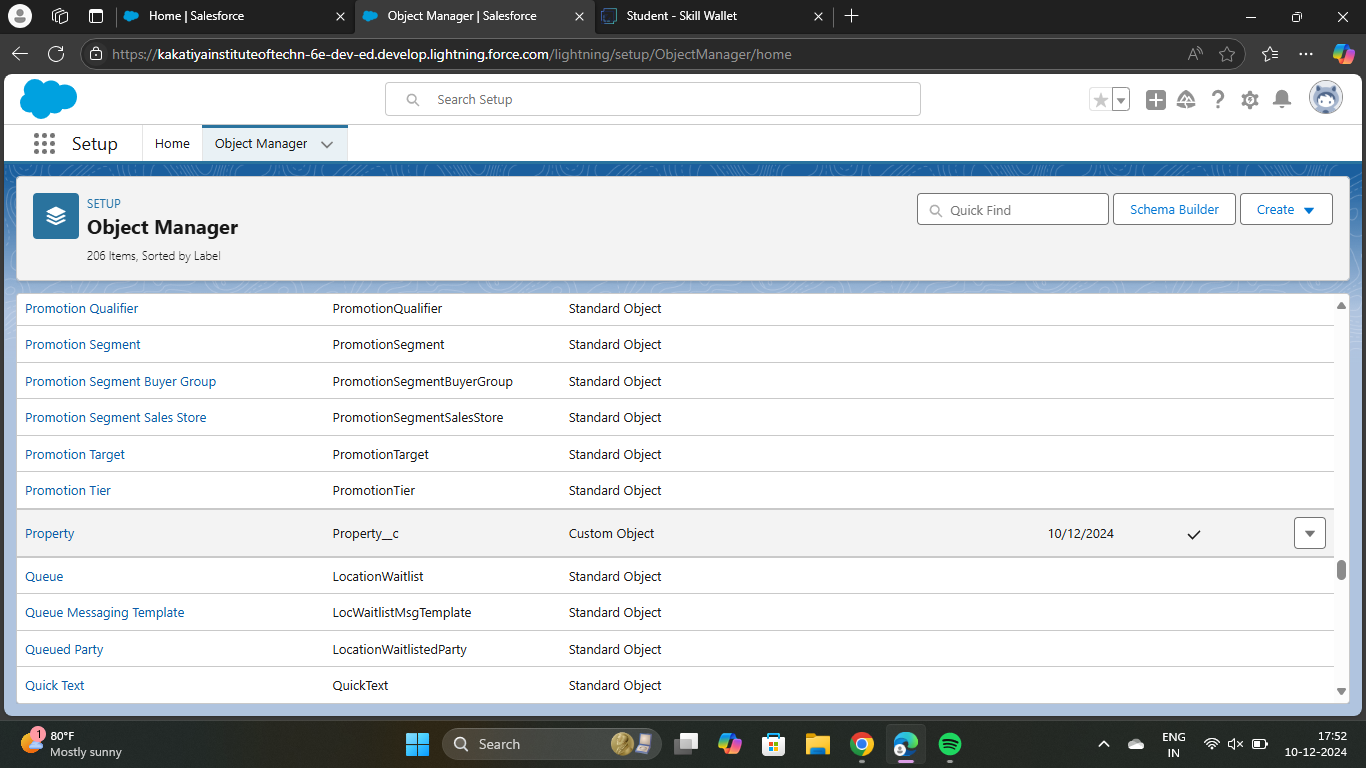
1. Go to your object manager and and click on create object from spreadsheet.
2. Click on the link to get the spreadsheet
3. [customer](https://docs.google.com/spreadsheets/d/1OnrKr4zVsbXrGvU6r0edgB6XOUlp4eCpfzHMSj8MN78/edit?usp=sharing)
4. After downloading, upload the file, map the fields and upload to create an object.



### Create Property object

Using the folowing steps,

1. Follow the same from the customer object to create the Property Object
2. [Property](https://docs.google.com/spreadsheets/d/1biFjB6xNxRItKb6CT6NHNNujHVVY9DTdXOkN2RL_hbY/edit?usp=sharing)



**TASK-3**

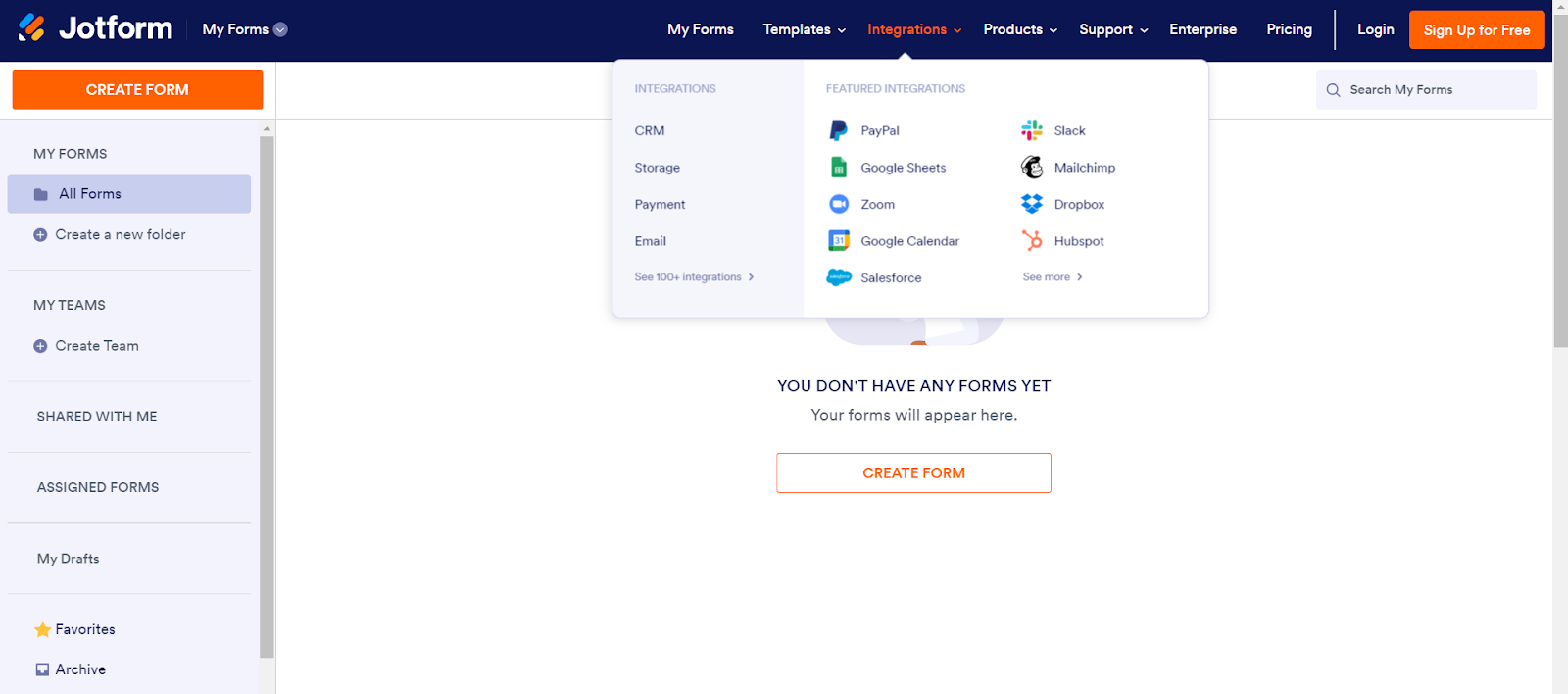
### **Integrate Jotform with Salesforce Platform**

In this Milestone we are going to integrate jotform with Salesforce.

### Activity 1

Using the following steps,

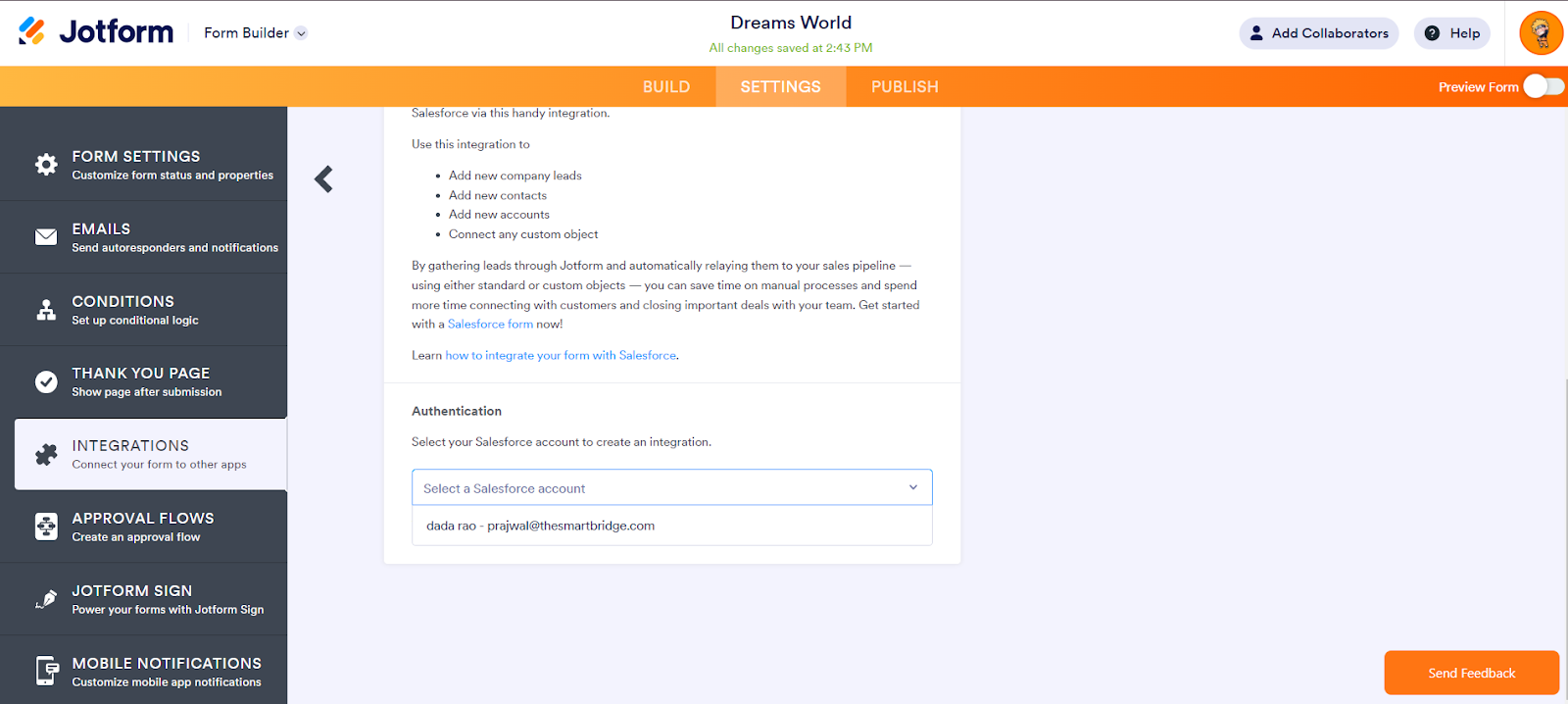
1. On the Jotform Platform, Click on Integration and choose Salesforce.



2. Click on User Integration and choose “Add to From”.



3. Select the Org with which you want to Integrate your jotform with.



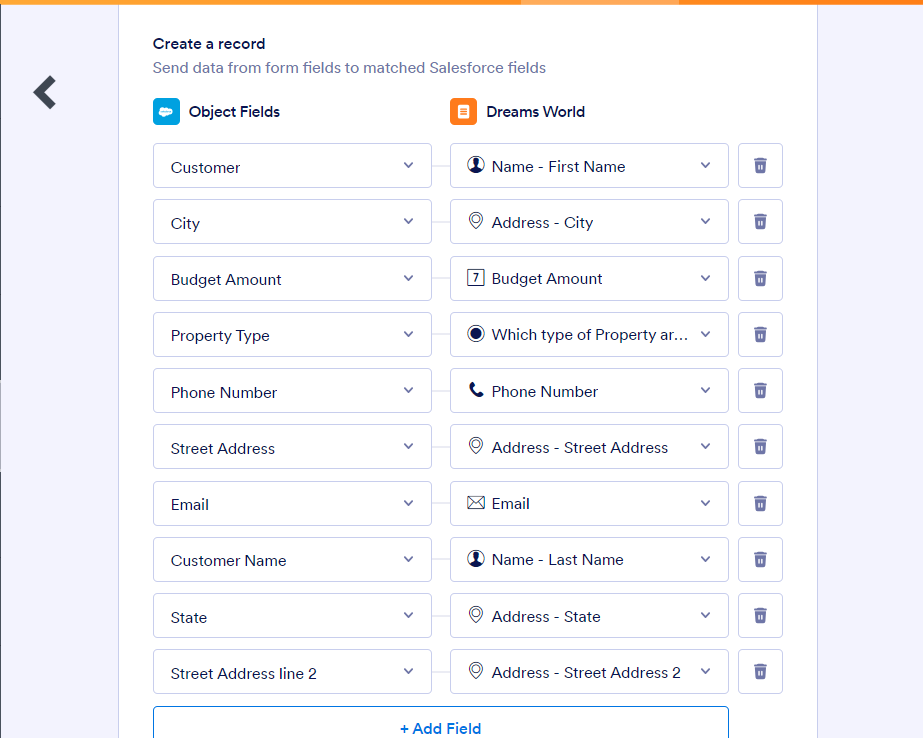
1. Select an Action - Create a record.

Select a Salesforce Object : - Customer

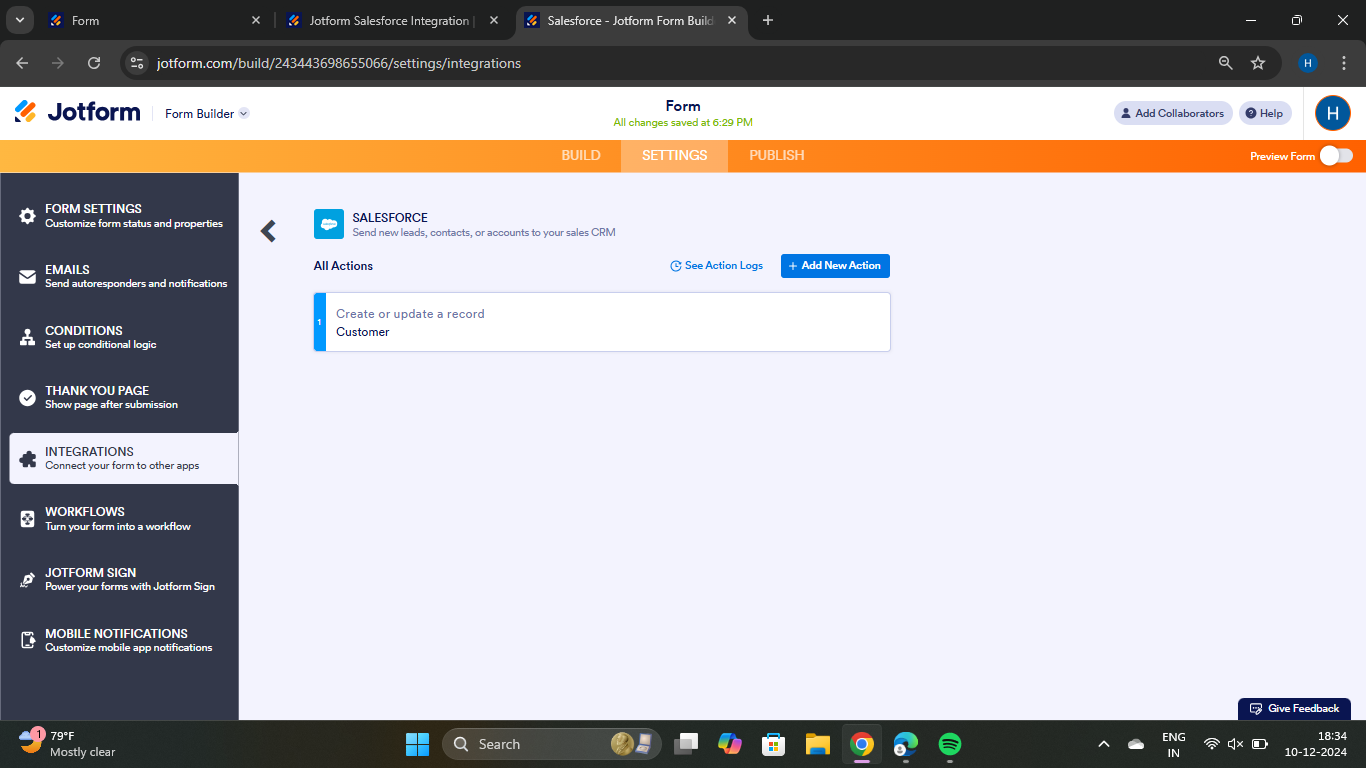


,

1. Map Each and every field on the Object with the fields on the form and “Save Action”.



1. Then “Save the Integration” and “Finish”.



**TASK-4**

### **Create Roles**

Create Roles as per business requirenment.

### Sales Executive Role

Using the following steps,

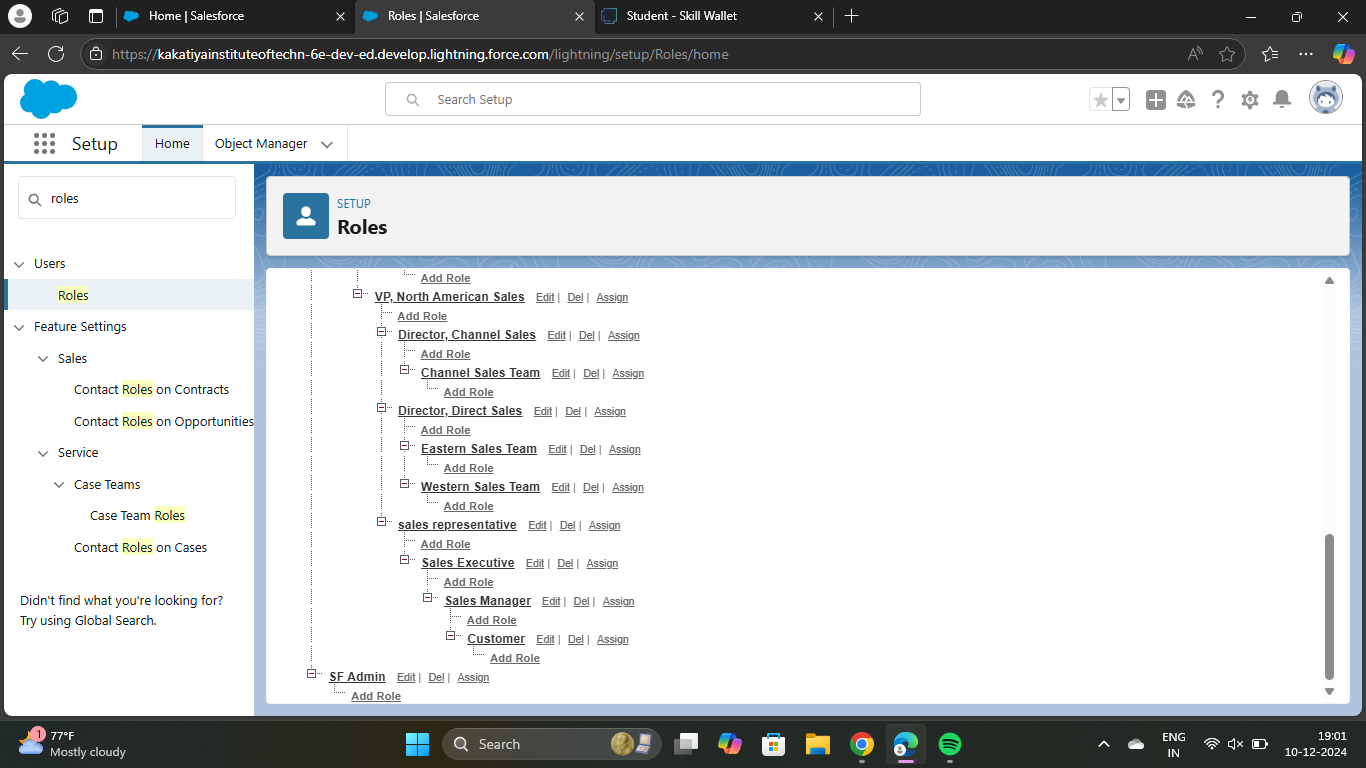
1) Go to Setup and Click on Roles, then click on Expand all and Add a Role just below the Sales Representative

\* It will use the “System Administrator Profile”.

2) 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 Sales Manager labeled as “Customer” which reports to Sales Manager.*



**,**

**TASK-5**

### **Create a Property Details App**

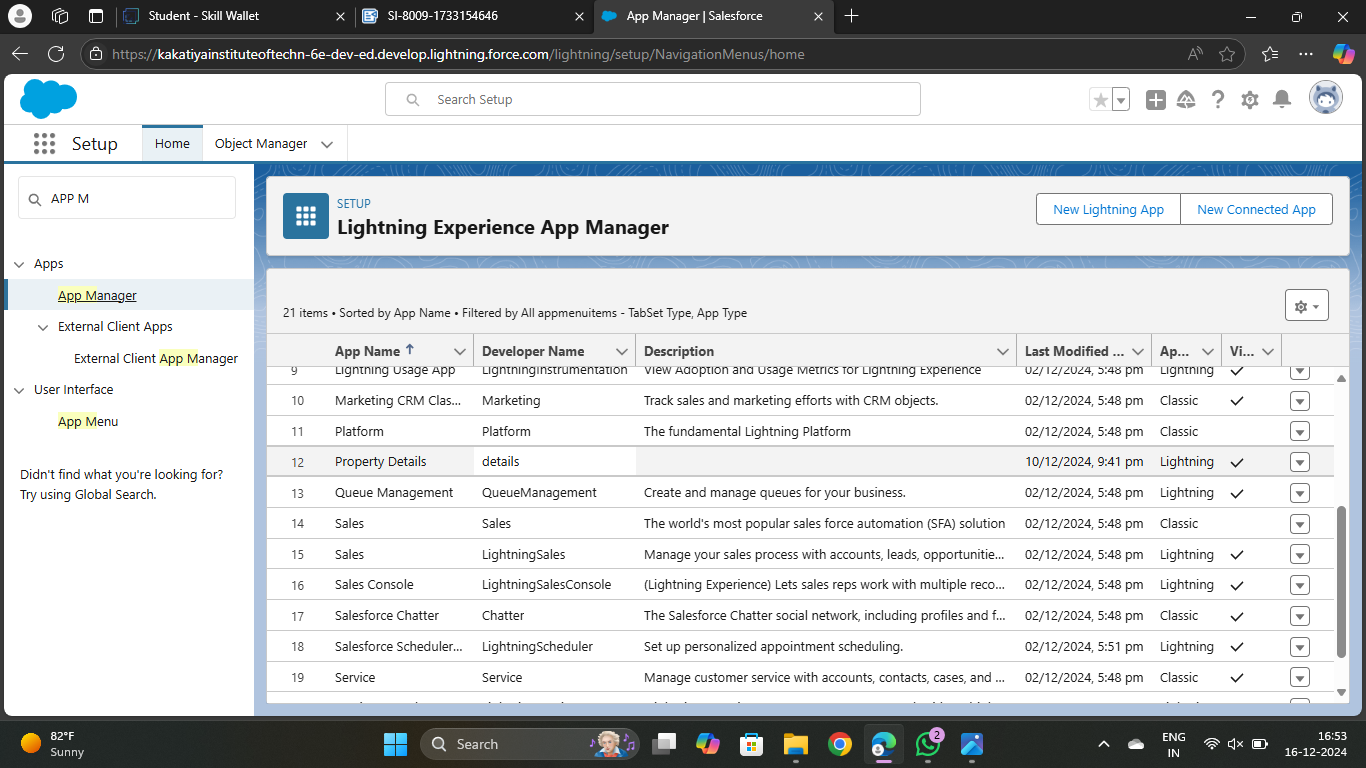
An App where the objects will be displayed.

### Activity 1

Using the following steps,

1. From Setup>> Go to App Manager and click on New Lightning App and Name it as “Property Details” and add “Customer” and “Property” Object.

2. Click Next >> Next >> Save and Add “System Admin ”Profile.



**TASK-6**

### **Create Profiles**

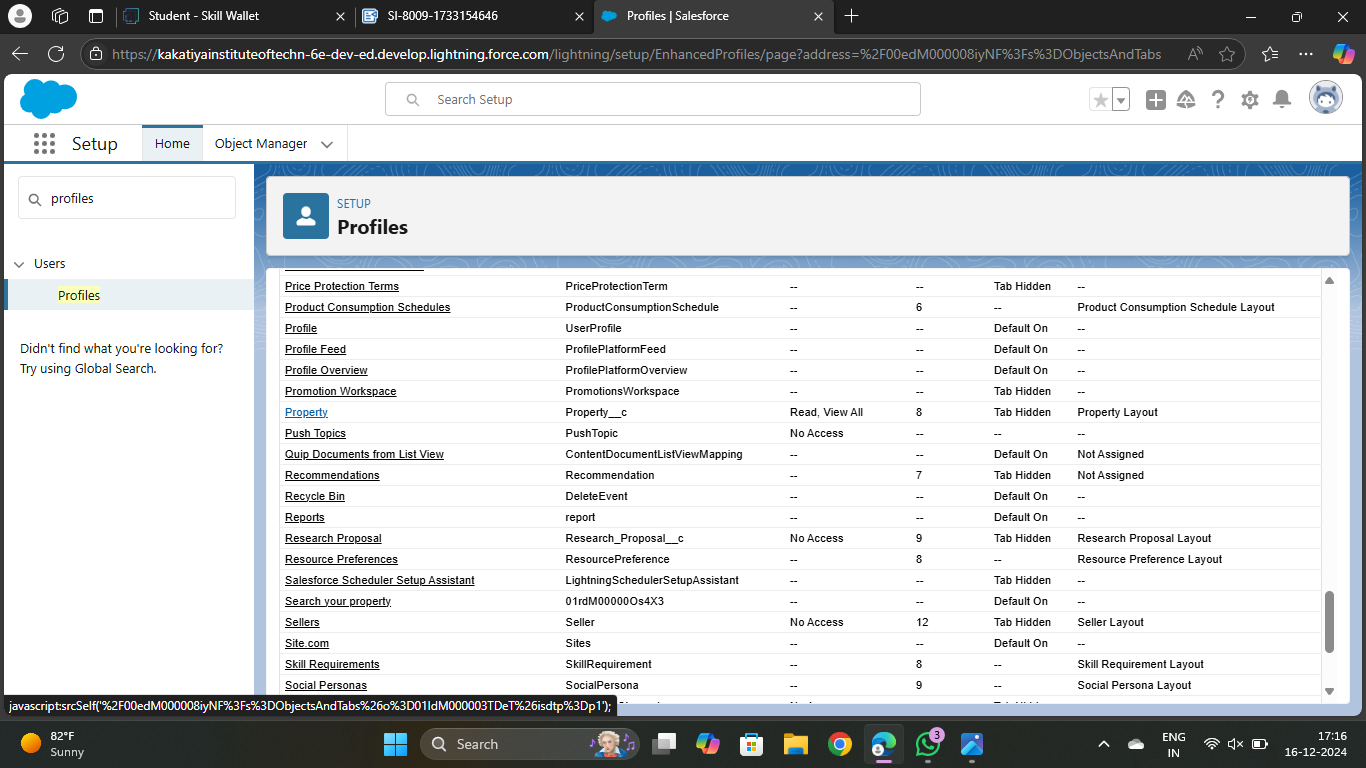
Create profiles as per business requirement.

### Customer

Using the following steps,

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

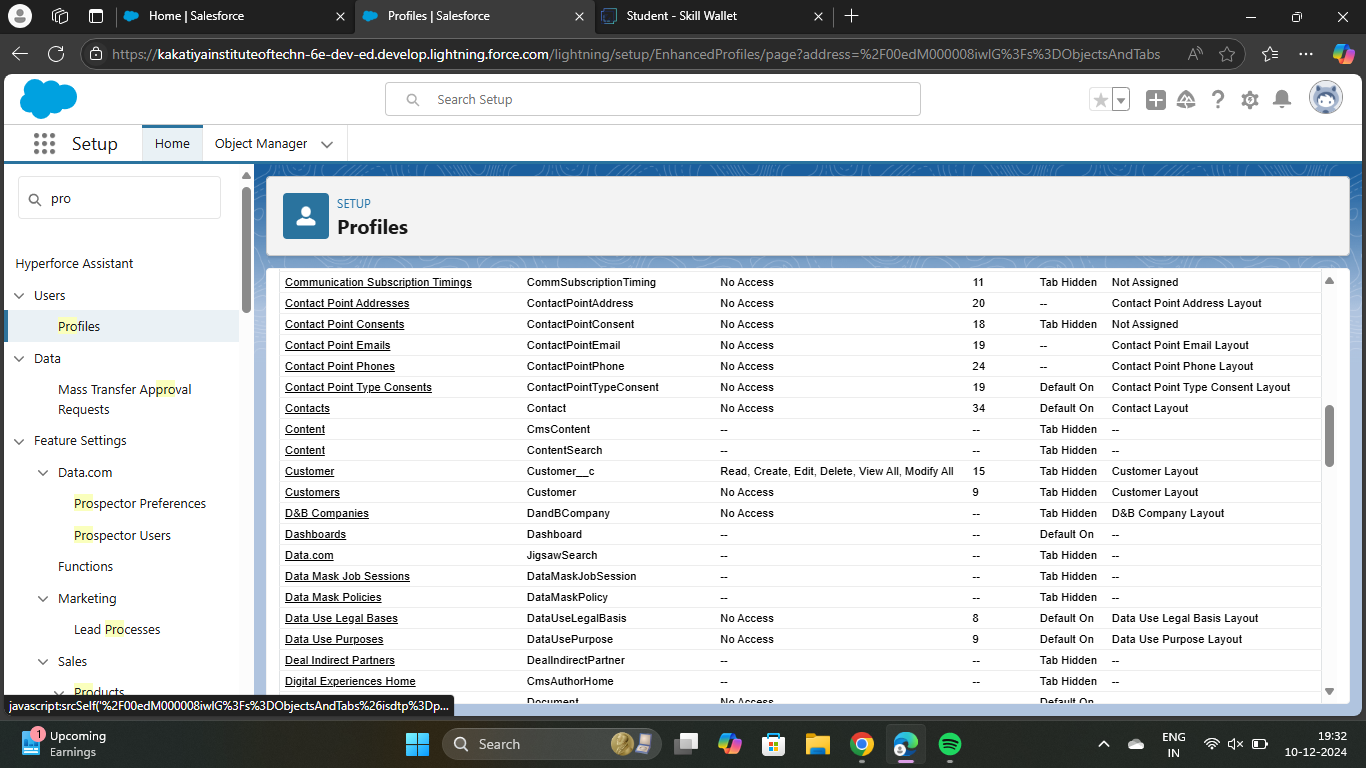
4. Uncheck all the Custom Object Permissions and check read and view all in “Property”

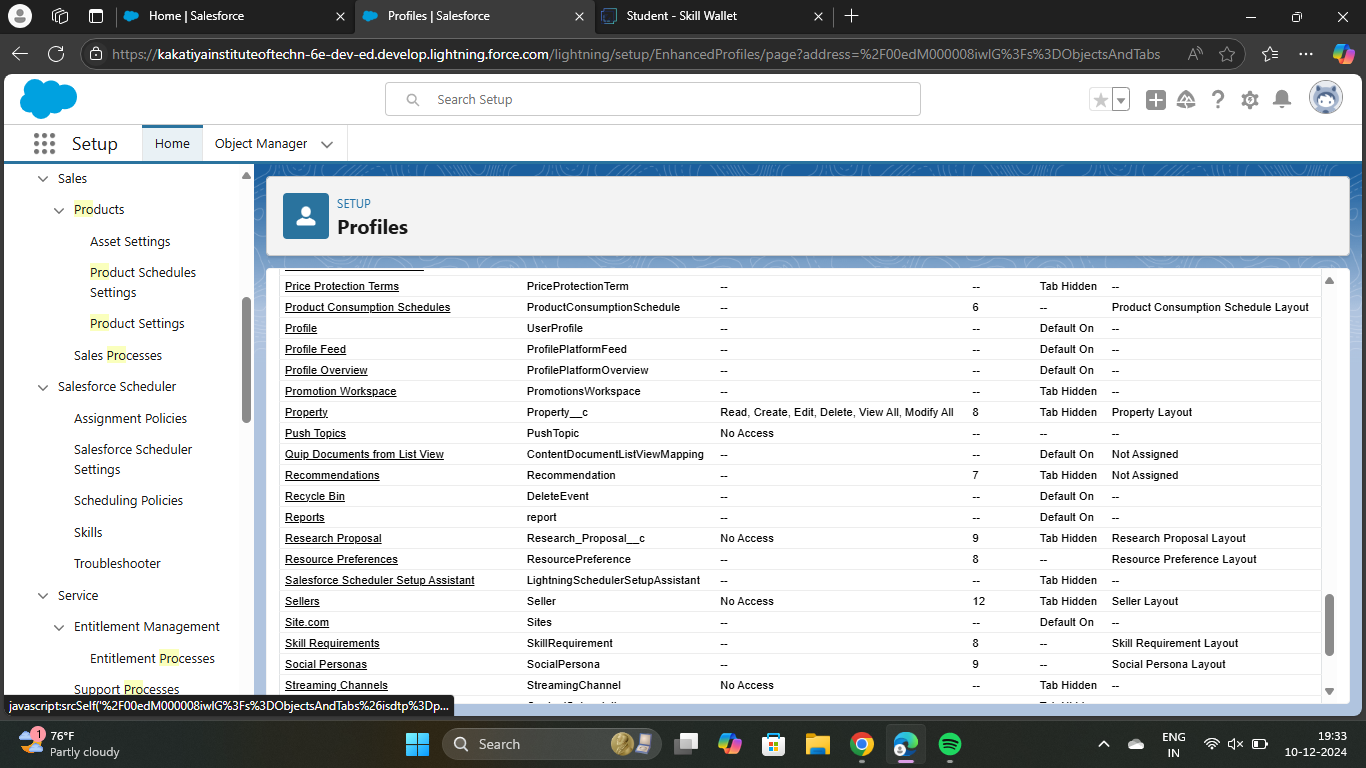


### Manager

Using the following 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”





**TASK-7**

### **Create a Check Box field on user**

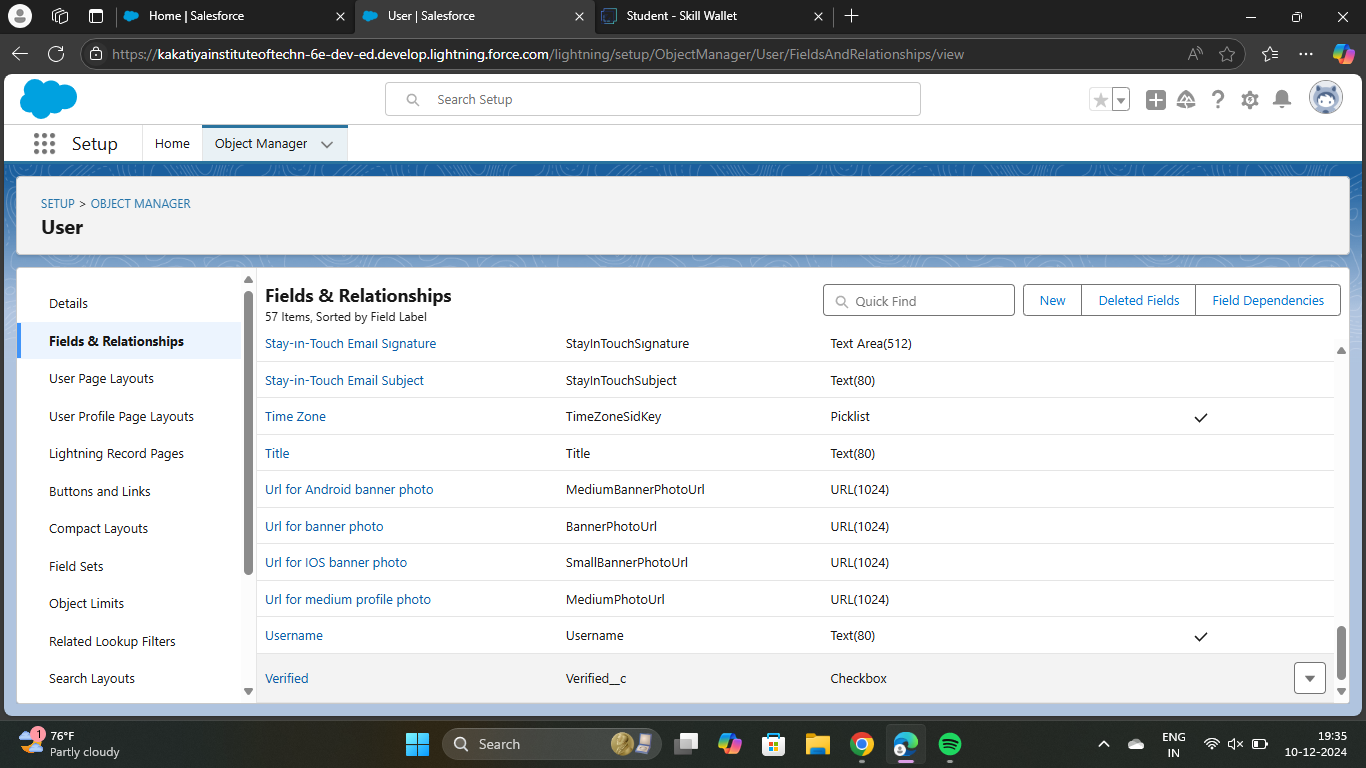
Create Field on the User as per the business requirement.

### Activity 1

Using the following steps,

1. Setup >> Object Manager >> Search for User >> Fields and Relationships

2. Create new Field Named as “Verified” as Data type “Check Box”



**TASK-8**

### **Create Users**

Create three different users with three different Roles and profiles as we have mentioned above.

### **User 1**

Using the following steps,

1. Go to Setup --> Administration --> Users --> New User
2. Last Name - Executive
3. Role - Sales Executive
4. License - Salesforce
5. Profile - System Administrator
6. Save

### **User 2**

Using the following steps,

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Manager
3. Role >> Sales Manager
4. License >> Salesforce Platform
5. Profile >> Manager
6. Save

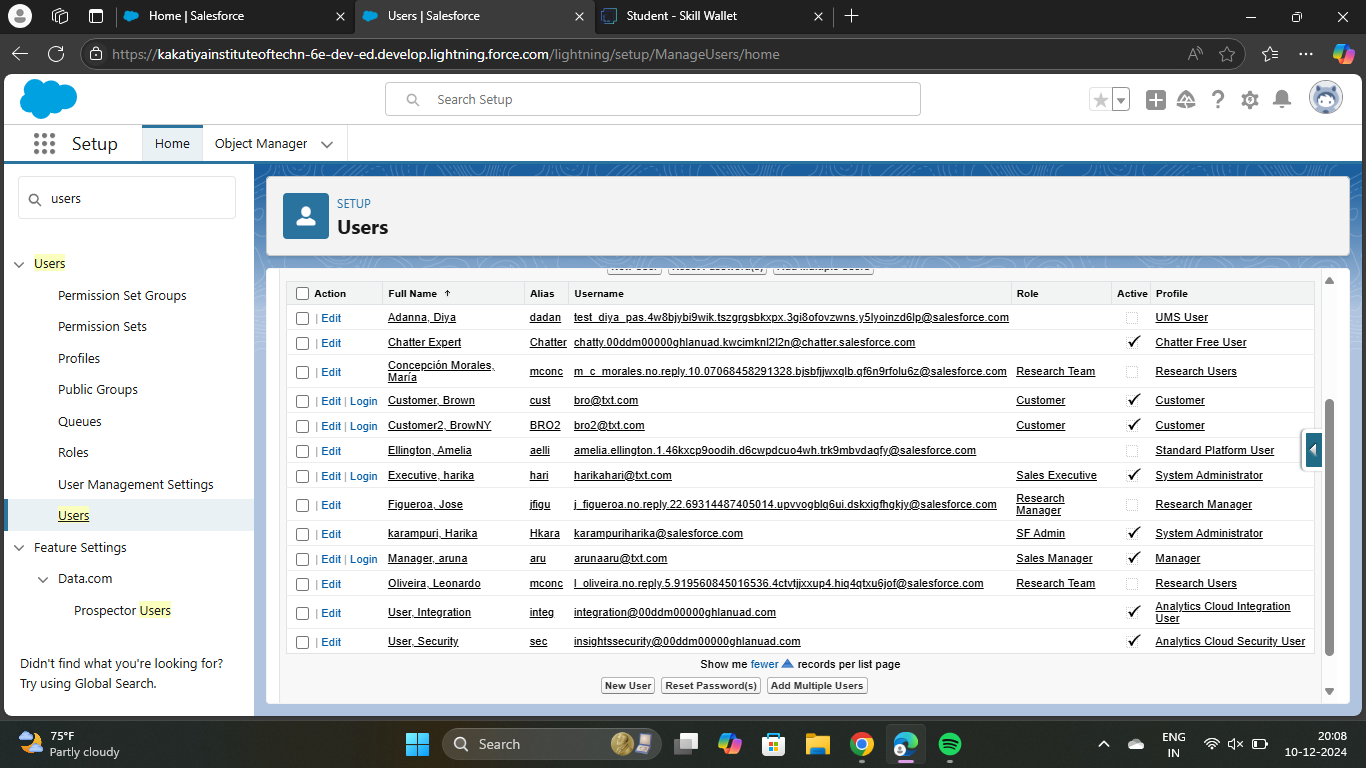
### **User 3**

Using the following steps,

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Customer
3. Role >> Customer
4. License >> Salesforce Platform
5. Profile >> Customer
6. Make Sure the verified check box is “Unchecked”
7. Save

### **User 4**

1. Go to Setup >> Administration >> Users >> New User
2. Last Name >> Customer2
3. Role >> Customer
4. License >> Salesforce Platform
5. Profile >> Customer
6. Make Sure the verified check box is “checked”
7. Save



**TASK-9**

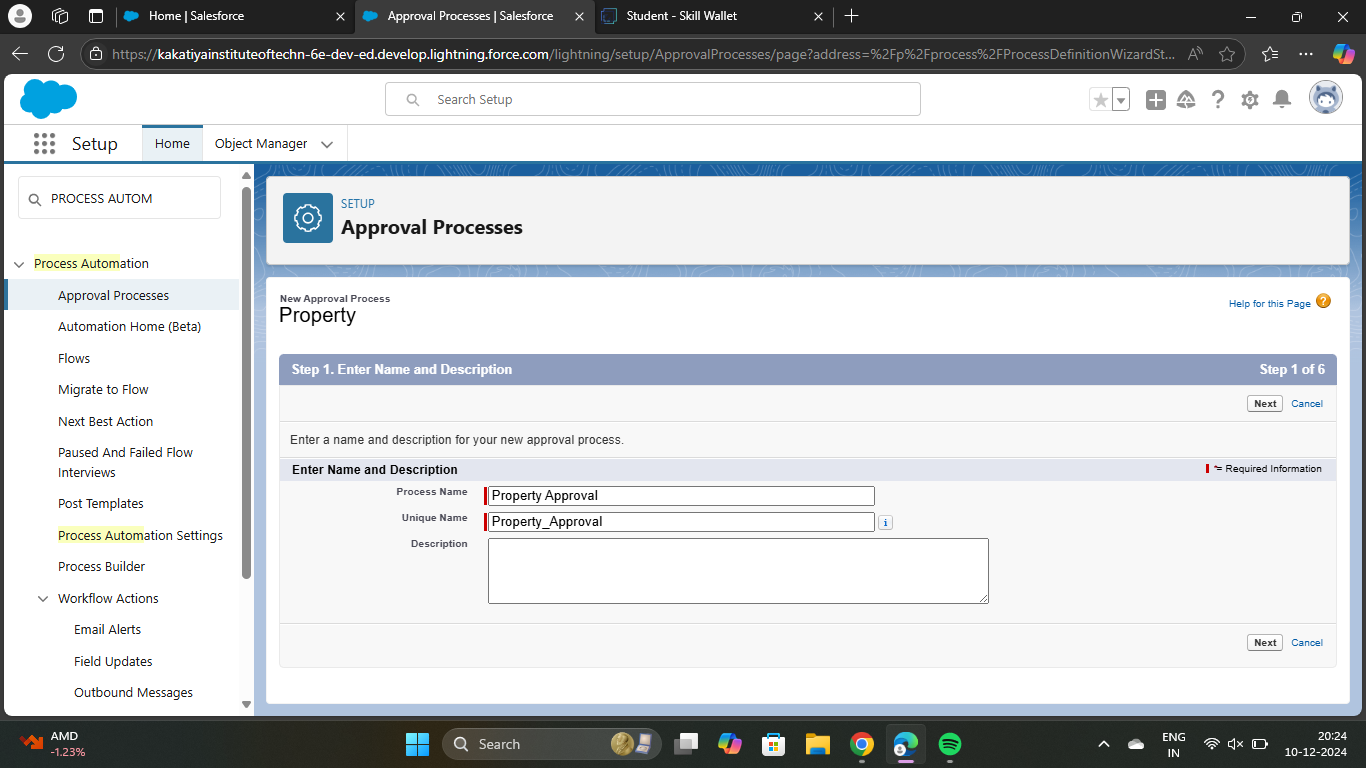
### **Create an Approval Process for Property Object**

An Approval process to approve or reject the records as according.

### **Activity 1**

Using the following steps,

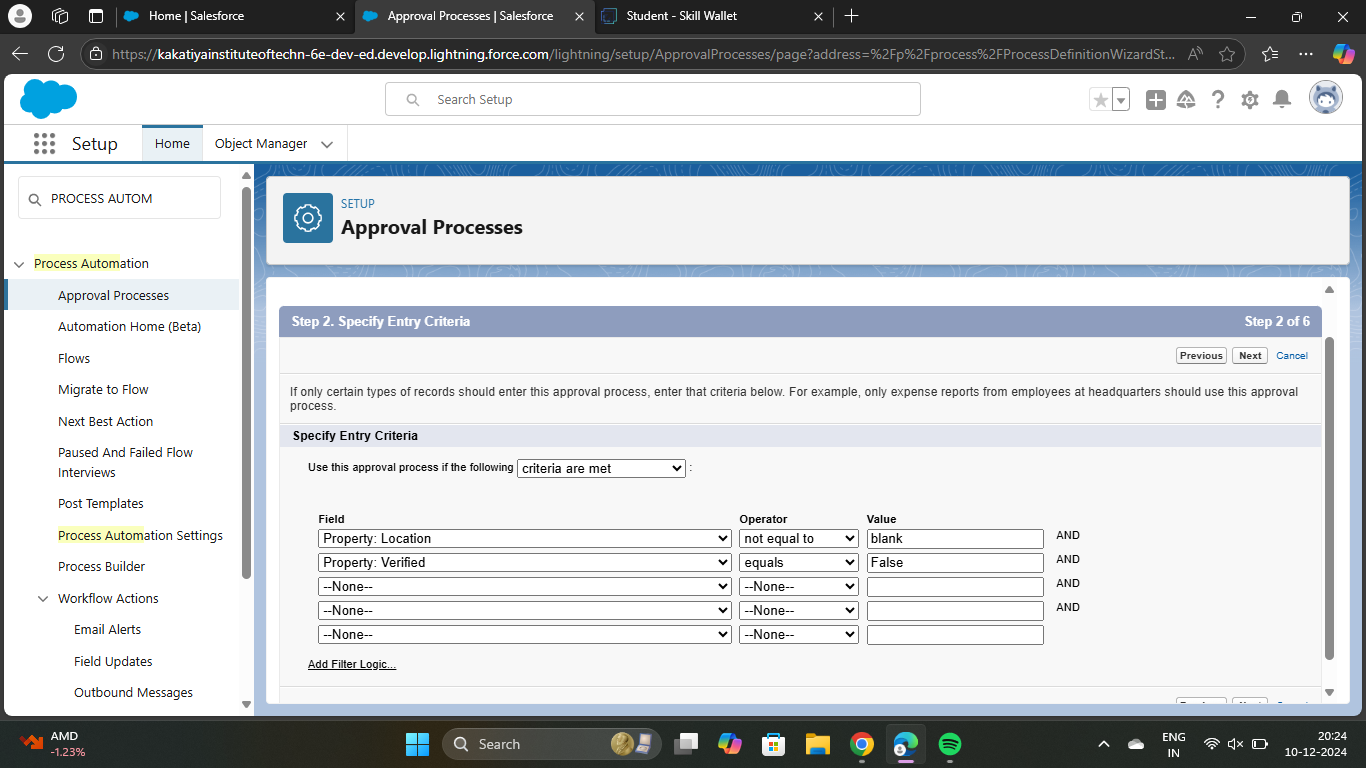
1. From Setup >> Process Automation >> Approval Process
2. Process Name - Property Approval



1. Give 2 criteria -

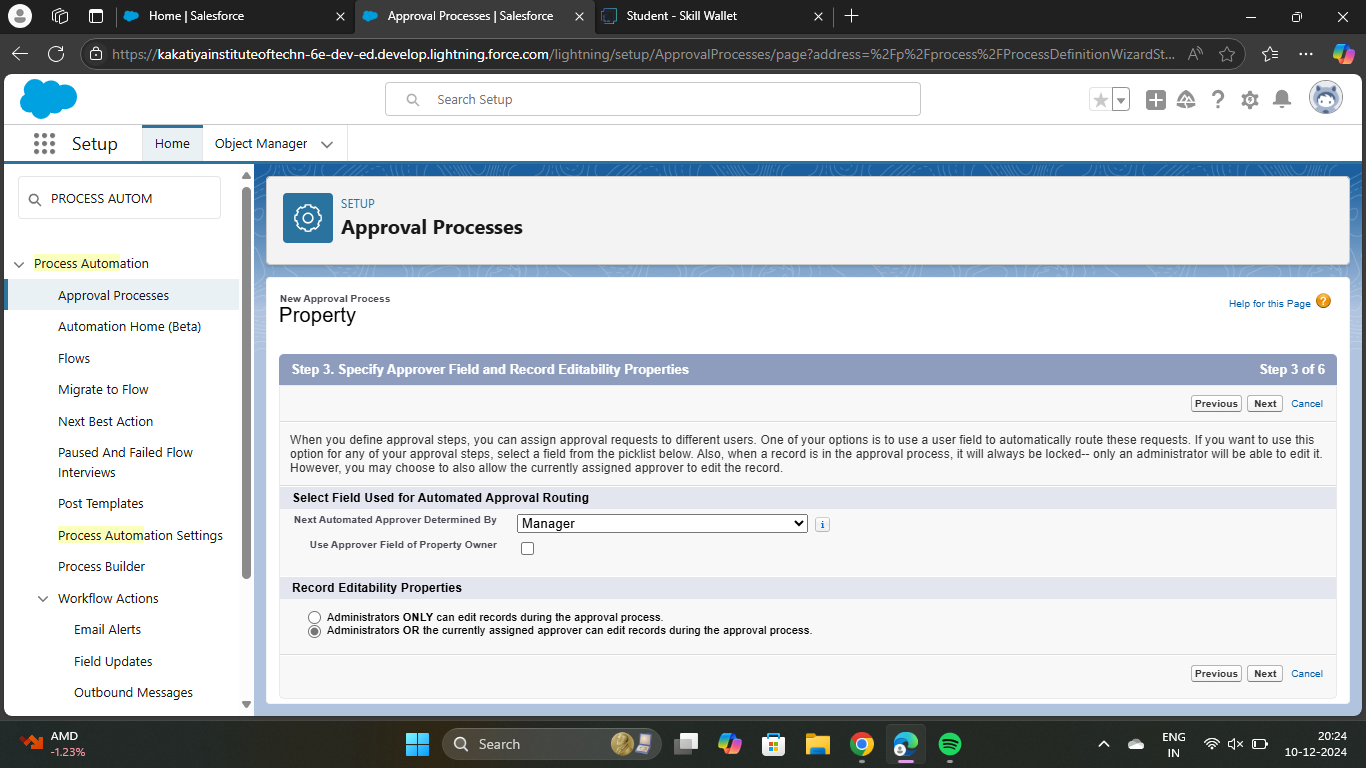
Location is not equal to blank,

Verified Equals false.

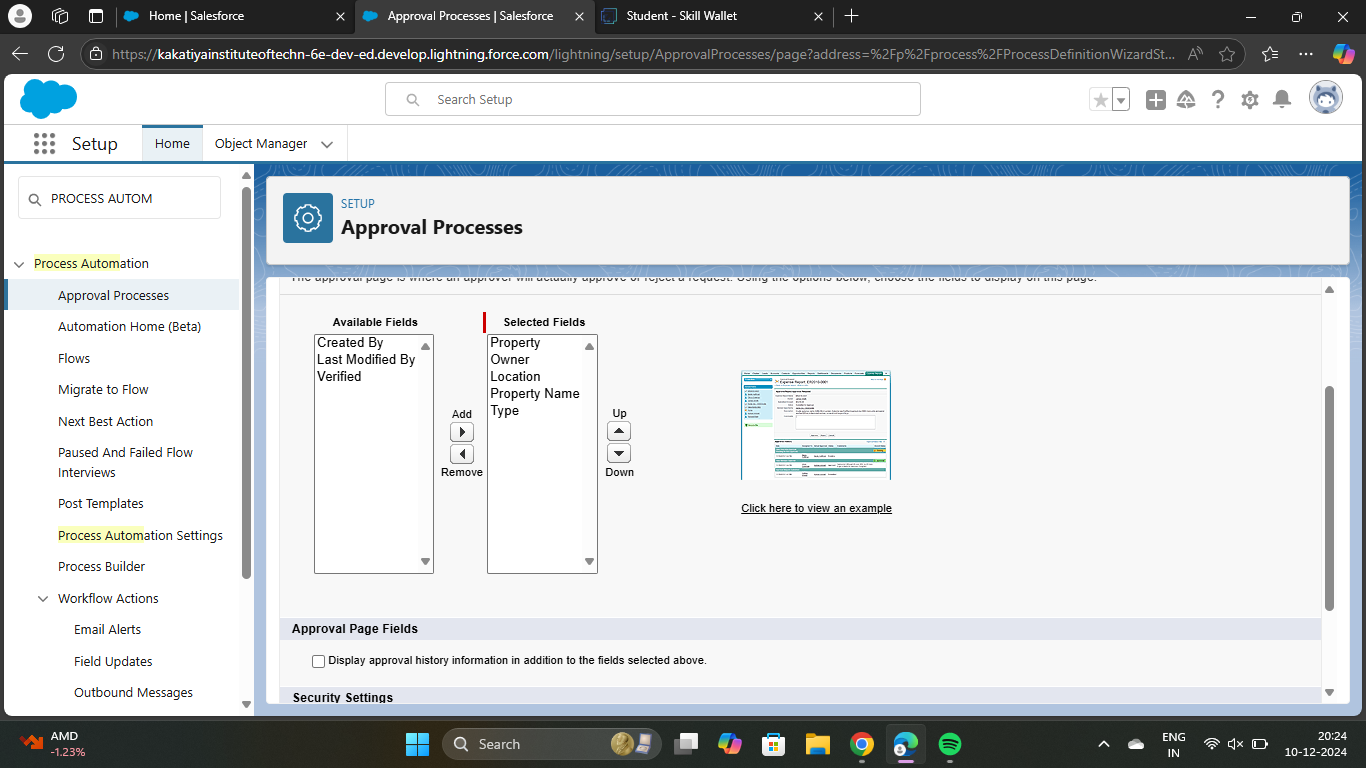


4. Click next and “Next Automated Approver Determined By” Select Manager

5. From Record Editability Properties >> Click on AdministratorsORthe currently assigned approver can edit records during the approval process.



6. From Step 5. Select Fields to Display on Approval Page Layout select Property, Owner, Location, Type.



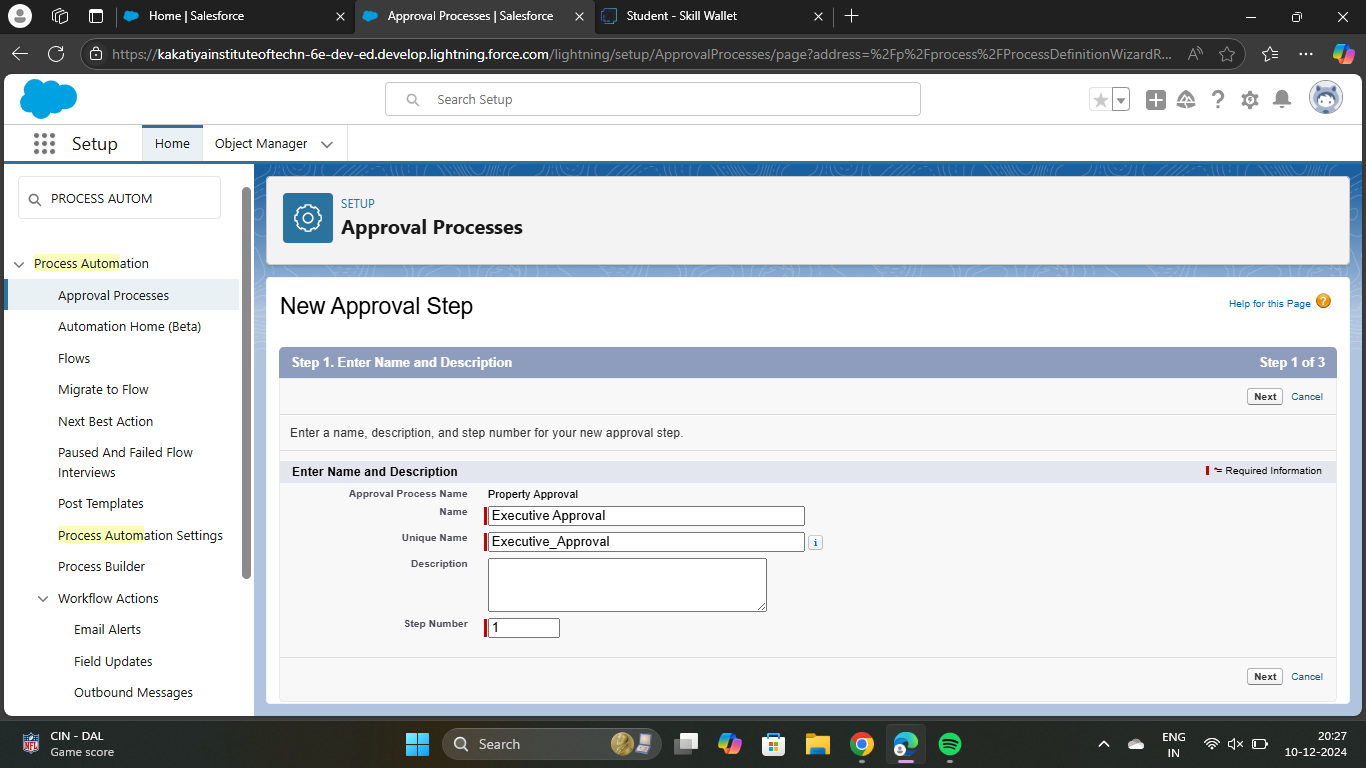
7. Click Next and Select the initial Submiters >>

Owner >> Property Owner

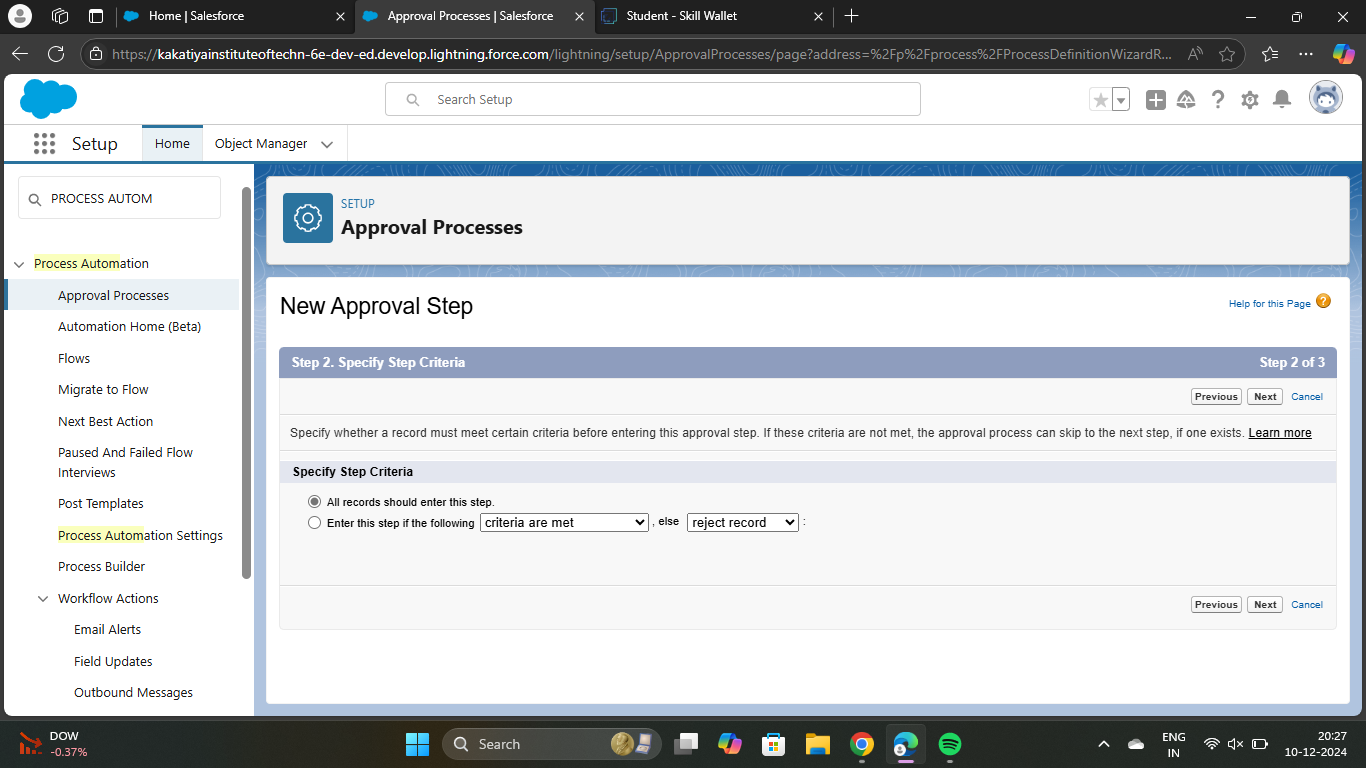
Roles >> Sales Manager

8. Save.

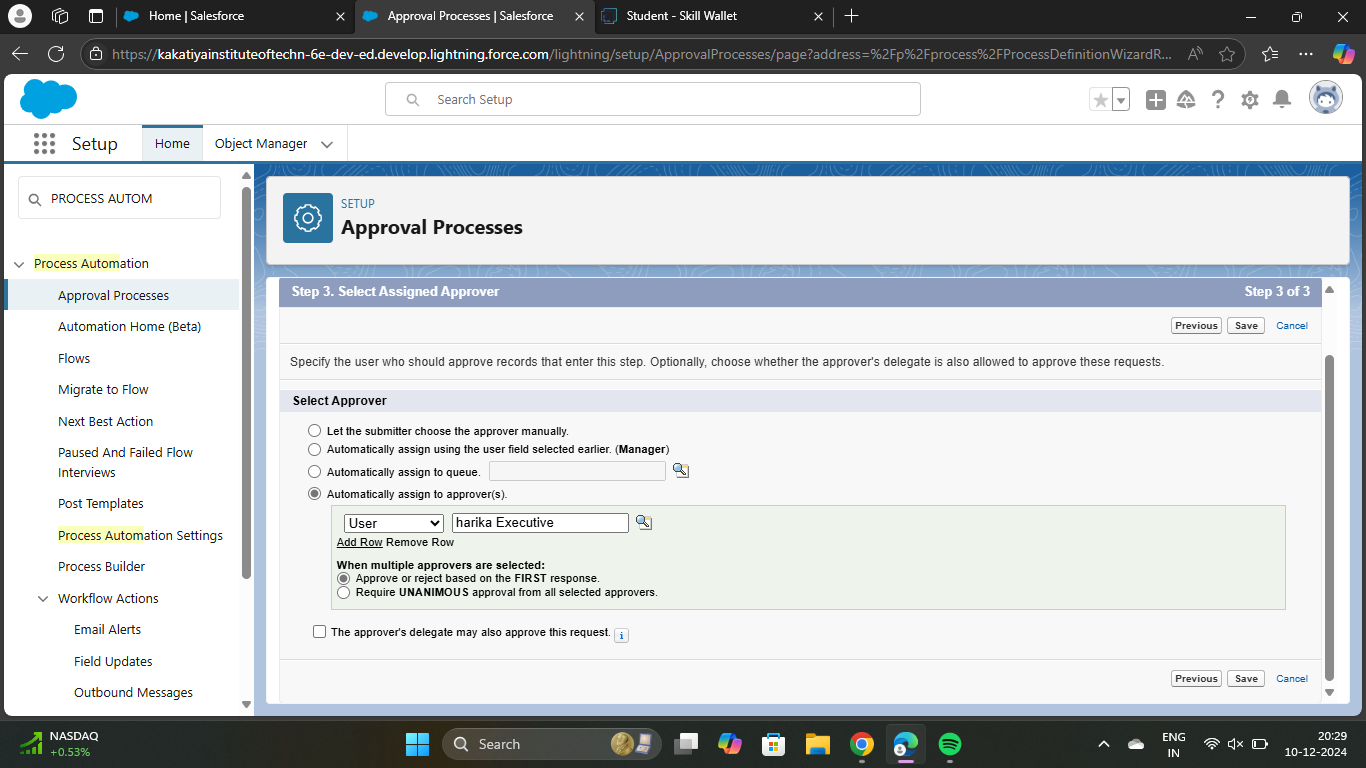
9. Add an approval step name “Executive Approval ”.



10. specify the Criteria >> All record should enter.



11. click next and select the Approver as “ Sales Executive “ and “Save”.



12. Add One field Update as “Verified Property”

Select Object >> Property

Field to Update >> Verified

Field Data Type >> CheckBox

Select CheckBox Option as “True”

Save.

13. Add One field Update as “UnVerified Property”

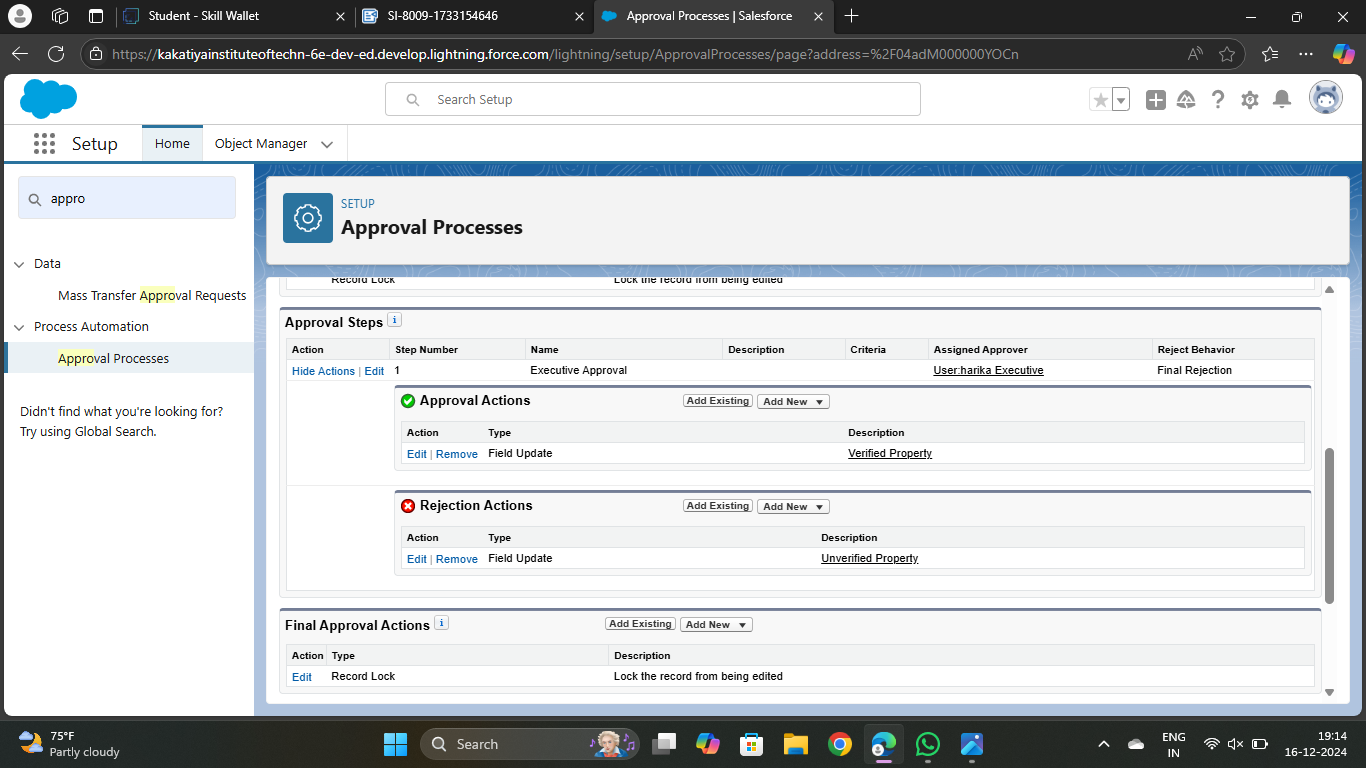
Select Object >> Property

Field to Update >> Verified

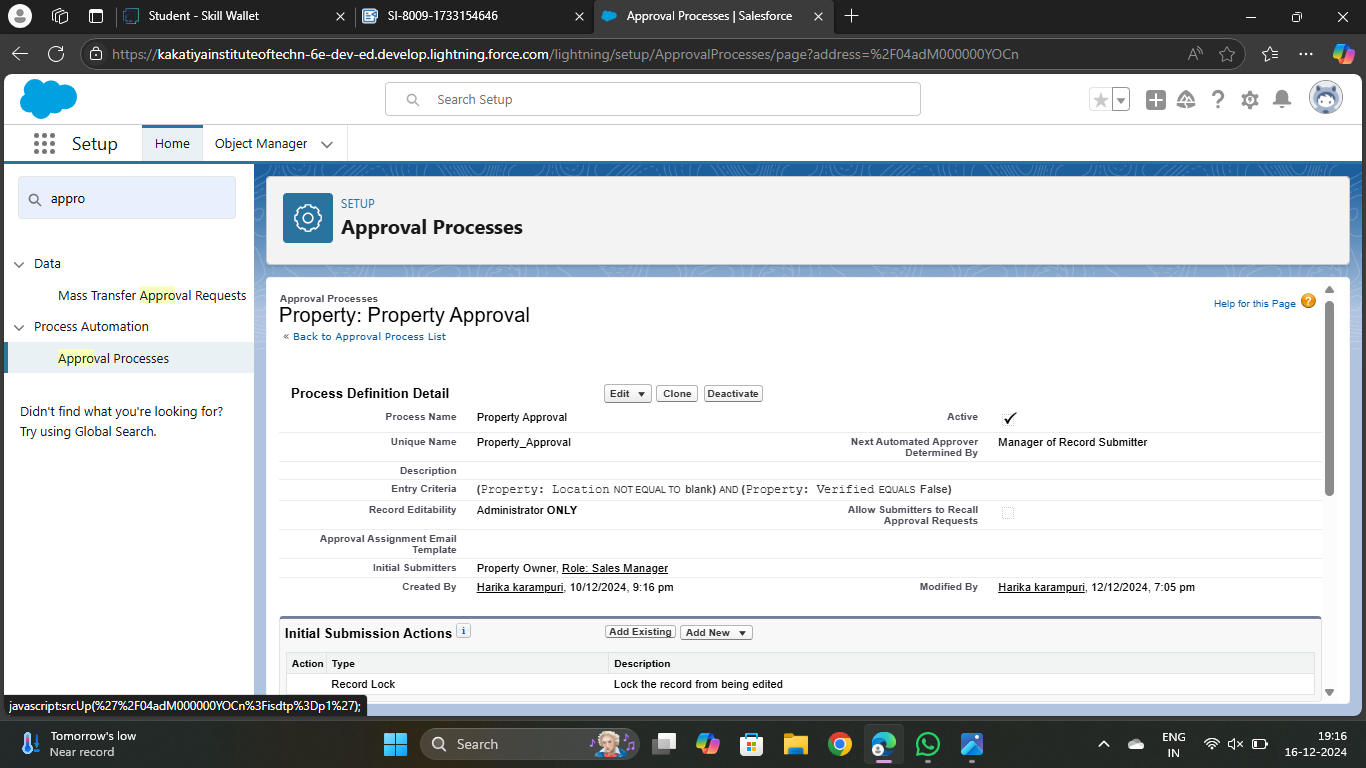
Field Data Type >> CheckBox

Select CheckBox Option as “False”

Save.



14. Activate the Approval Process.



**TASK-10**

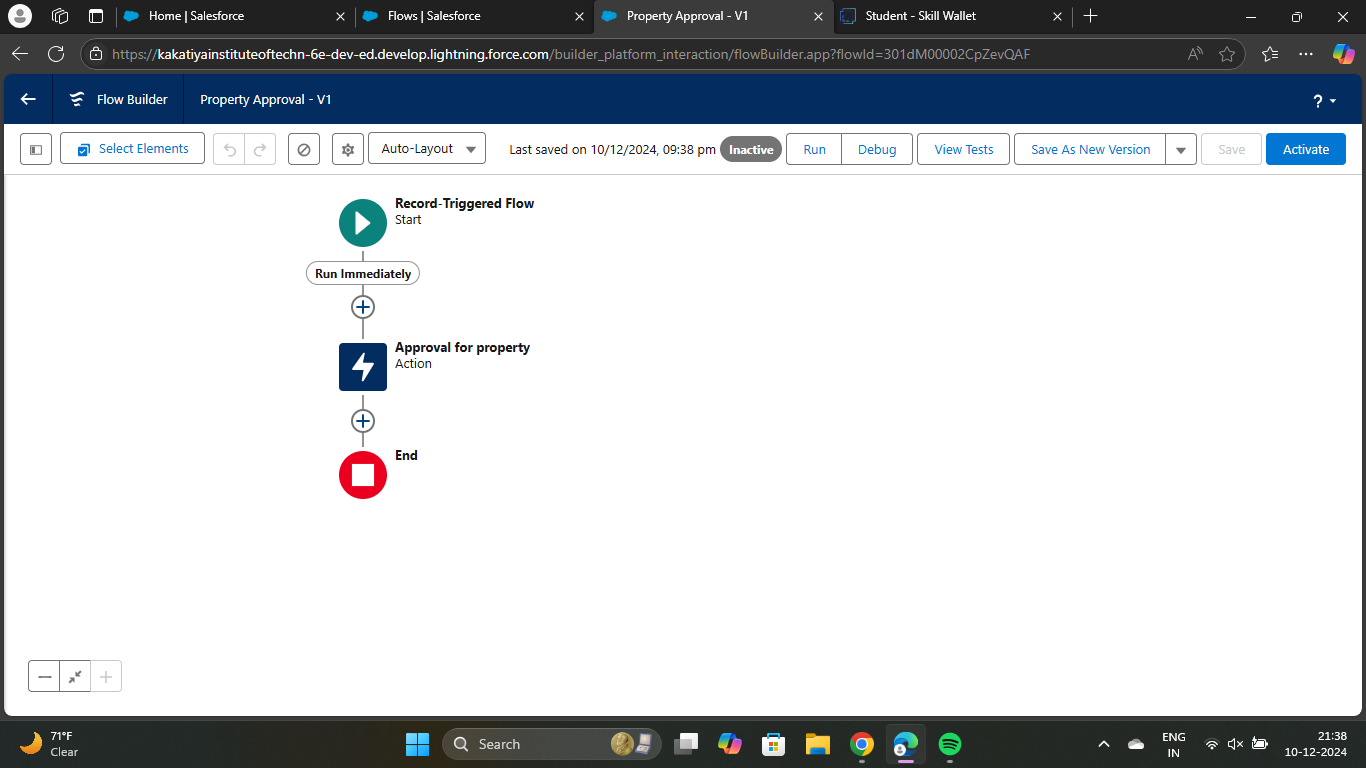
### **Create a Record trigger flow to submit the Approval Process Automatically.**

A flow that can submit the records directly for approval.

### **Activity 1**

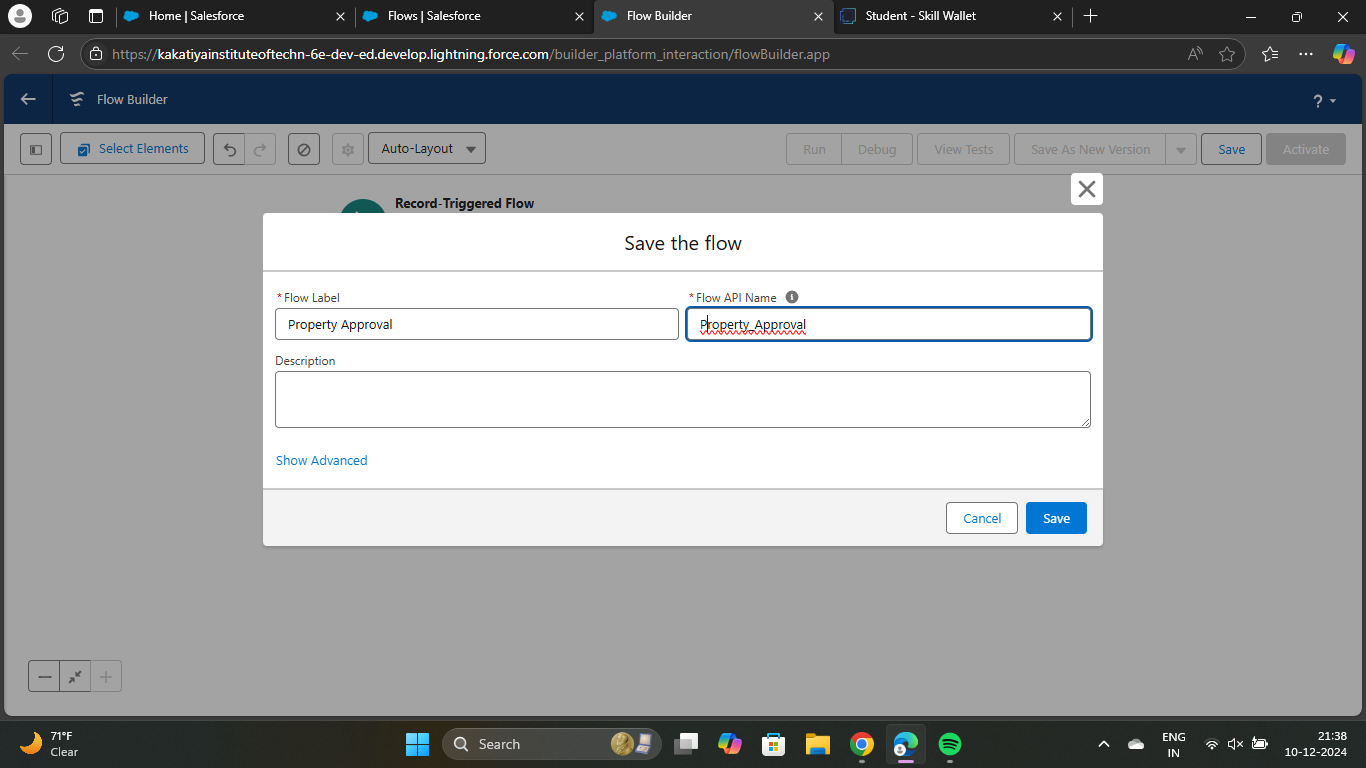
Using the following steps,

1. From Setup >> Search for Flows >> Click On New and Select “Record Trigger Flow”.
2. Select Object >> Property
3. Select “Trigger the flow when” >> “A record is created”
4. Set Entry Conditions >> “None”



1. Give Label >> Approval for property
2. Record Id >> {!$Record.Id}
3. Done

9. Save the Flow and Give label as “Property Approval” and “Activate”.



**TASK-11**

### **Create an App Page**

Create an App Page on the Property details Object named as “Search Your Property”.

### Activity 1

Using the following steps,

1. From Setup >> Go to Lightning App Builder >> Click on New >> Select App Page and

Click on Next.

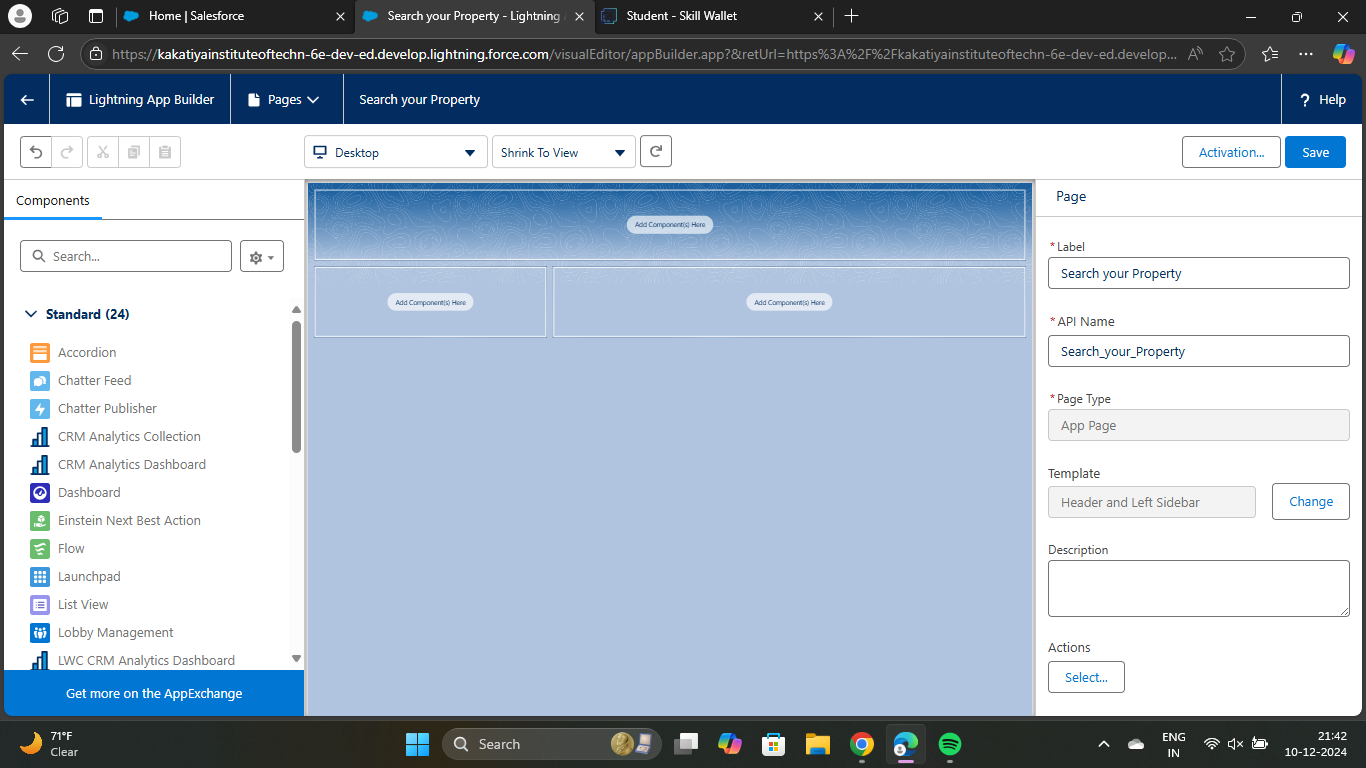
2. Give Label as “Search your Property” click “Next”.

3. Click “header and Left Sidebar” and Click on “Done.  
 4. Click on “Save ” and then click on “Activate”.

1. From Page Setting select page activation as “Activate for all Users”.

6. From Lightning Experience Click on “Property Details” and click on Add Page“.

7. Then Click on “Save”.



**TASK-12**

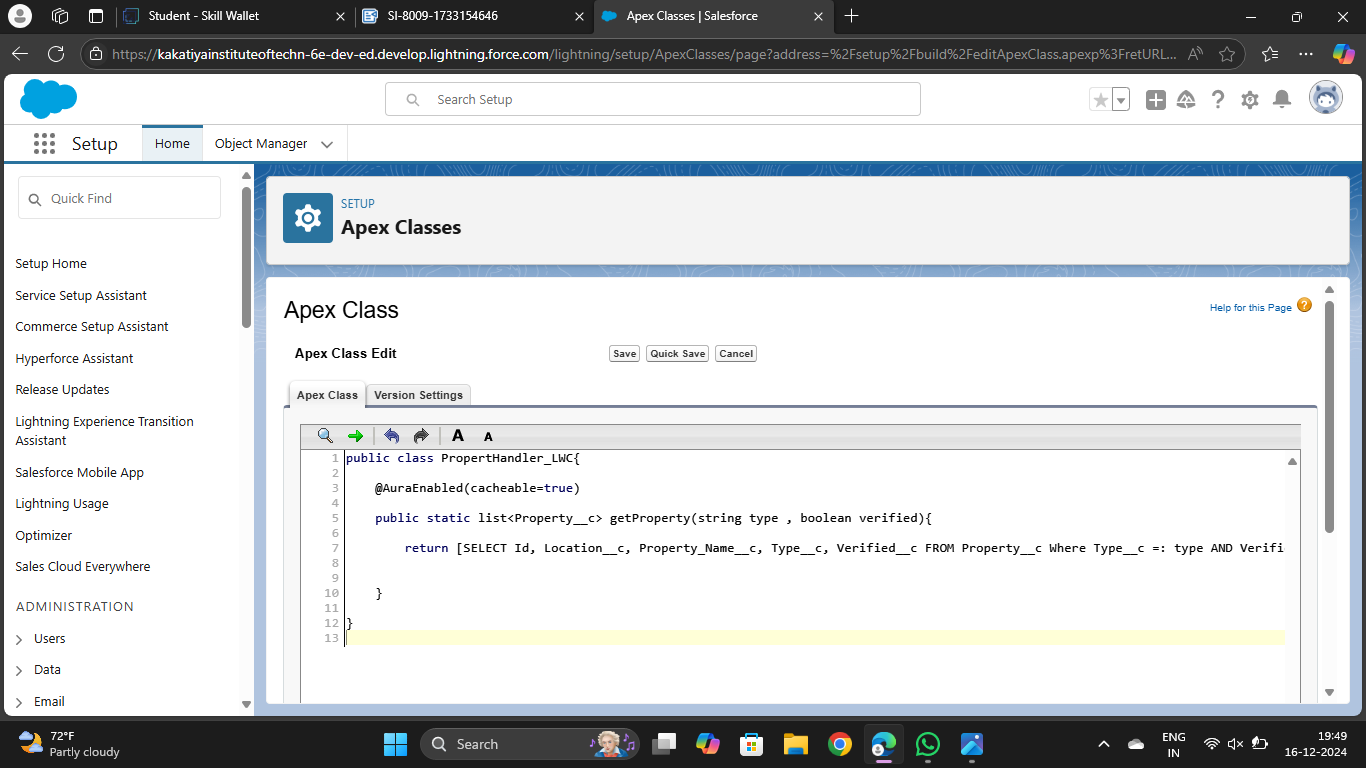
### **Create a LWC Component**

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”.

### Activity 1

Using the following steps,

1. Create an Apex Class and make it aura enabled and name it “PropertHandler\_LWC”.



1. Create a Lightning Web Component in your VsCode, and (ctrl+shift +P) and click on authorize an org.
2. Enter your login id and password to authorize your org.
3. Now (ctrl+shift +P) and Create a lightning Web Component and Name it Anything you want to. (Example - )
4. In your Html File Write this code : -

<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}>

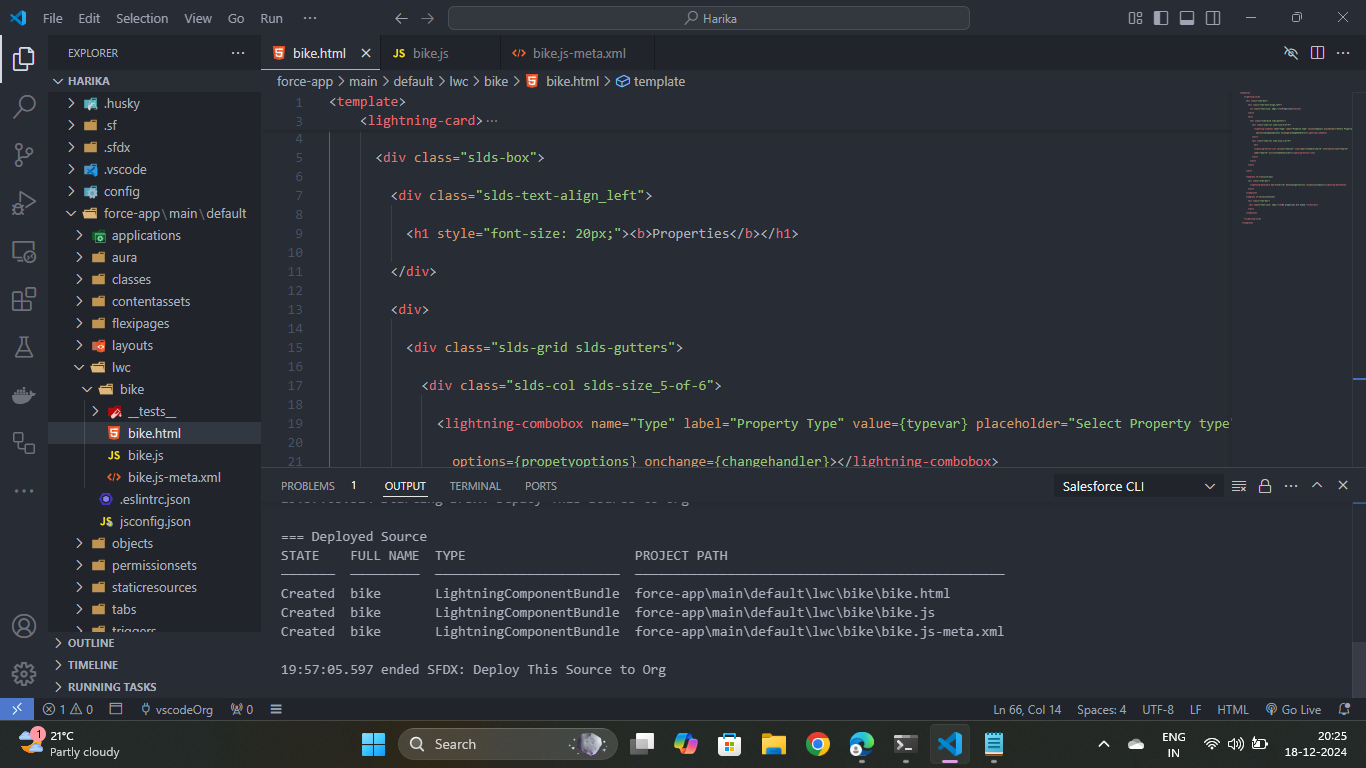
<div class="slds-box">

<div style="font-size: 15px;"><b>No properties Are Found !!</b></div>

</div>

</template>

</lightning-card></template>



1. In Your Js File Write this code : -

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;

}

})

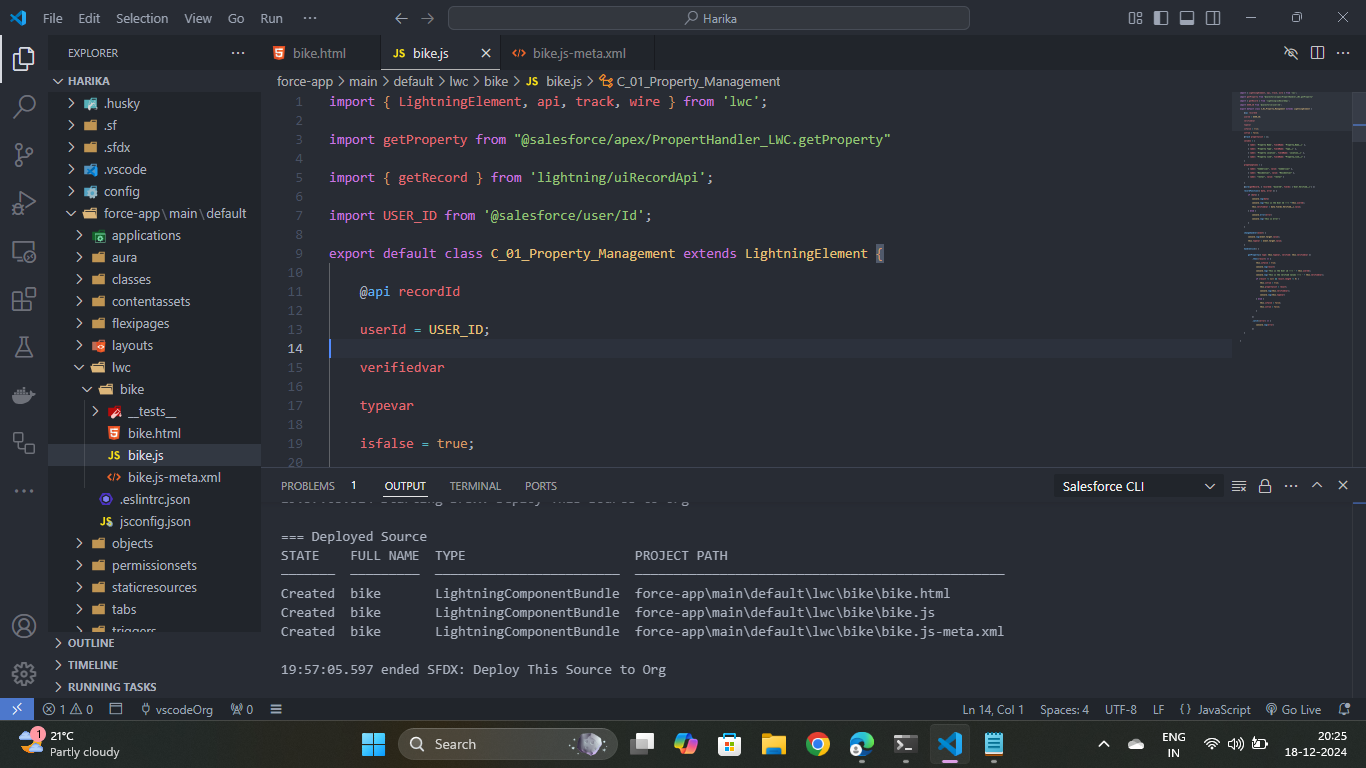
.catch((error) => {

console.log(error)

})

}

}



1. In Your metafile give your targets to deploy the component.

<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata">

<apiVersion>59.0</apiVersion>

<isExposed>true</isExposed>

<targets>

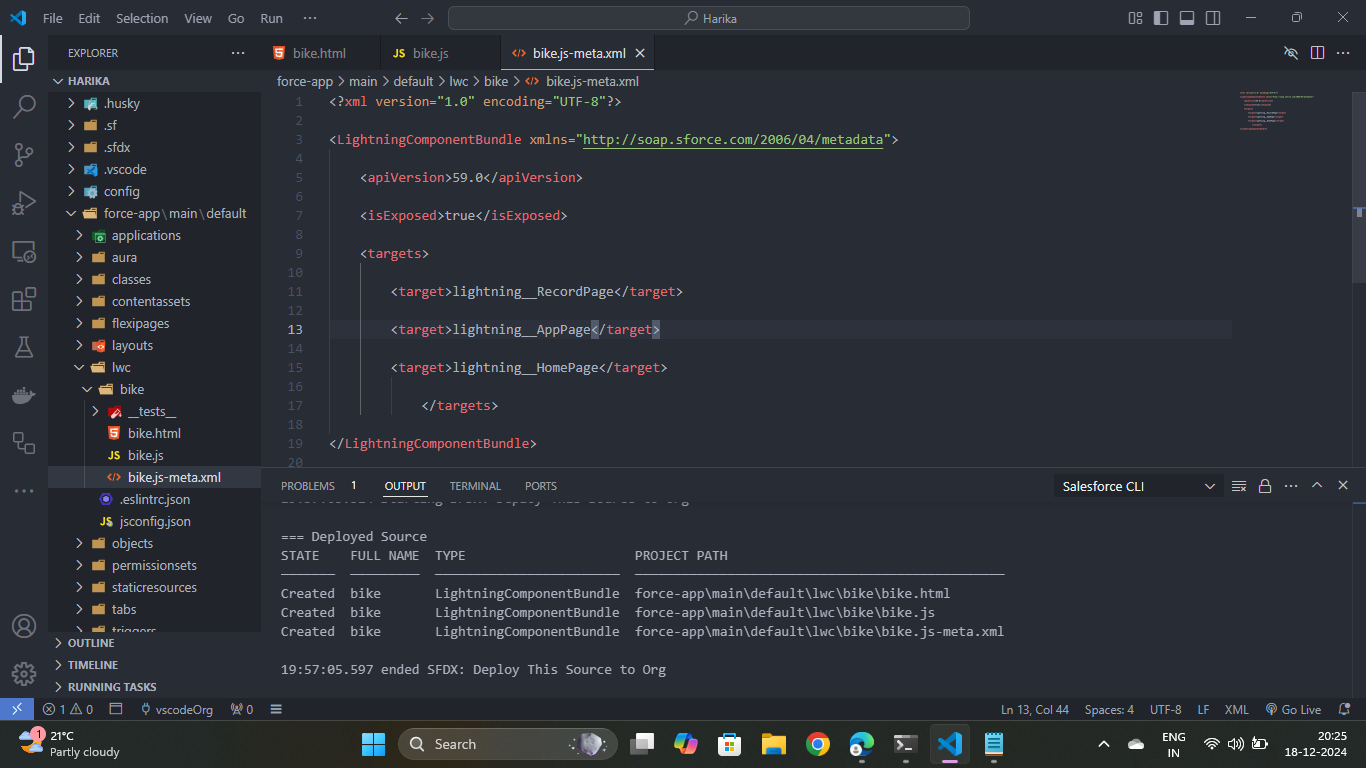
<target>lightning\_\_RecordPage</target>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_HomePage</target>

</targets>

</LightningComponentBundle>



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

**TASK-13**

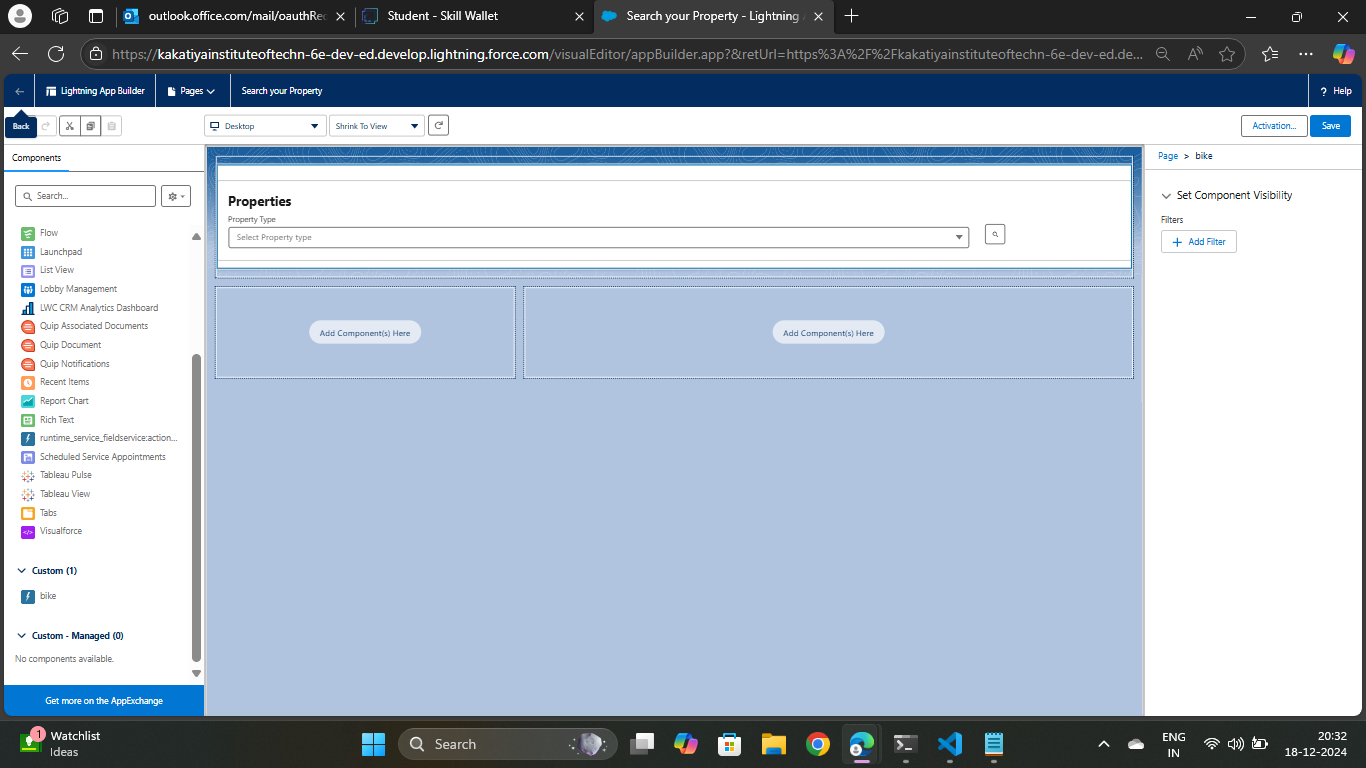
### **Drag this Component to your App Page**

Adding the Component to your Page.

### Activity 1

Using the following steps,

1. From Setup >> Go to App Launcher >> Search for Property Details.
2. On this Page click on gear icon and click on Edit Page.



**TASK-14**

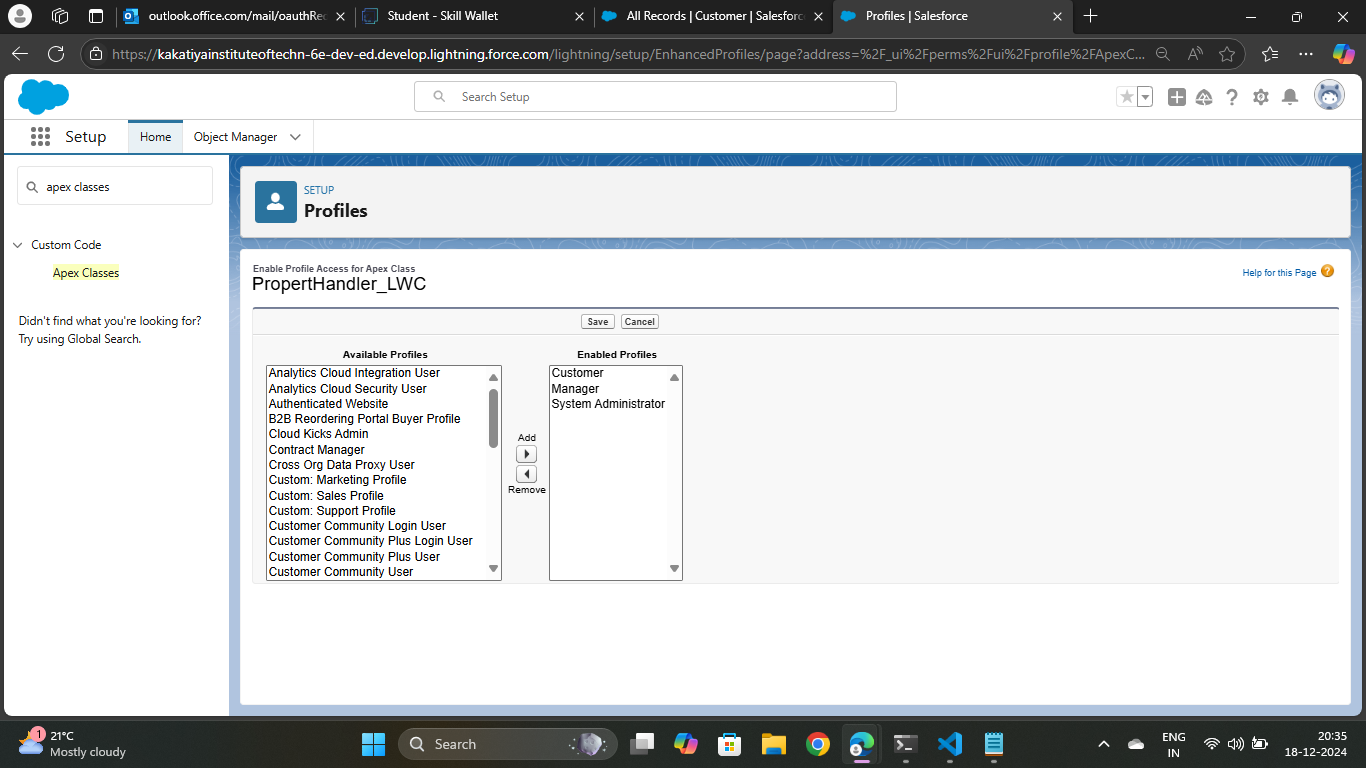
### **Give Access of Apex Classes to Profiles**

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

### Activity 1

Using the following steps,

1. From Setup >> Search For Apex Classes >> Click on “Security” behind “PropertyHandler\_\_LWC”.
2. From Profiles Add “Manager” and “Customer” and “Save”.



**Salesforce Key Features and Concepts Utilized**

**Key Features**

1. Accounts and Contacts: Store client information, including contact details, addresses, and relationships.

2. Leads and Opportunities: Manage new business opportunities, track leads, and convert them into clients.

3. Custom Objects: Create custom objects to manage properties, including details like address, type, and status.

4. Activity and Task Management: Schedule and track activities, tasks, and appointments related to clients and properties.

5. Reporting and Analytics: Generate reports and dashboards to analyze client interactions, property performance, and business trends.

6. Integration with Email and Calendar: Integrate with email and calendar systems to streamline communication and scheduling.

7. Mobile Access: Access the CRM application on-the-go using Salesforce's mobile app.

8. Security and Compliance: Ensure data security and compliance with industry regulations, such as GDPR and CCPA.

**Key Concepts**

1. Objects and Fields: Use custom objects and fields to store and manage client and property data.

2. Relationships and Lookups: Establish relationships between objects, such as clients and properties, using lookups and master-detail relationships.

3. Page Layouts and Views: Customize page layouts and views to display relevant information and streamline user workflows.

4. Workflows and Approvals: Automate business processes using workflows and approvals, such as assigning tasks or sending notifications.

5. Validation Rules and Formulas: Use validation rules and formulas to enforce data integrity and perform calculations.

7. Customization and Configuration: Customize and configure the CRM application using Salesforce's point-and-click interface.

**TESTING AND VALIDATION**

Testing Approach

The testing approach will include the following:

1. Unit Testing: Unit testing will be performed to ensure that individual components of the application work as expected.

2. Integration Testing: Integration testing will be performed to ensure that the different components of the application work together seamlessly.

3. System Testing: System testing will be performed to ensure that the entire application works as expected.

4. User Acceptance Testing (UAT): UAT will be performed to ensure that the application meets the requirements and works as expected from a user's perspective.

Test Cases

The following test cases will be executed:

1. Test case 1: JotForm integration with Salesforce

- Preconditions: JotForm account, Salesforce org

- Steps: Fill out JotForm, submit form, verify record creation in Salesforce

- Expected result: Record created in Salesforce

2. Test case 2: Object creation from spreadsheet

- Preconditions: Spreadsheet with property data, Salesforce org

- Steps: Import spreadsheet into Salesforce, verify object creation

- Expected result: Objects created in Salesforce

3. Test case 3: JotForm and Salesforce integration

- Preconditions: JotForm account, Salesforce org

- Steps: Fill out JotForm, submit form, verify record creation in Salesforce

- Expected result: Record created in Salesforce

4. Test case 4: Role and profile creation

- Preconditions: Salesforce org

- Steps: Create role, create profile, assign profile to user

- Expected result: Role and profile created, user assigned to profile

5. Test case 5: Property Details app

- Preconditions: Salesforce org, Property Details app

- Steps: Launch app, verify property details display

- Expected result: Property details displayed

6. Test case 6: Approval process and record trigger flow

- Preconditions: Salesforce org, approval process, record trigger flow

- Steps: Submit record for approval, verify approval process, verify record trigger flow

- Expected result: Record approved, record trigger flow executed

7. Test case 7: App page and LWC component

- Preconditions: Salesforce org, app page, LWC component

- Steps: Launch app page, verify LWC component display

- Expected result: LWC component displayed

8. Test case 8: Apex class access

- Preconditions: Salesforce org, Apex class

- Steps: Verify Apex class access, verify Apex class execution

- Expected result: Apex class accessed, Apex class executed

Validation

The validation process will include the following:

1. Requirements Validation: Validate that the application meets the requirements.

2. Functional Validation: Validate that the application functions as expected.

3. Performance Validation: Validate that the application performs as expected.

4. Security Validation: Validate that the application is secure.

**CONCLUSION**

The CRM application to handle clients and their properties project in Salesforce has been successfully completed. Throughout this project, several milestones were achieved, including:

1. JotForm Integration: A JotForm was created and integrated with the Salesforce org to automatically create customer records.

2. Object Creation from Spreadsheet: Custom objects were created from a spreadsheet to manage property details.

3. JotForm and Salesforce Integration: The JotForm was integrated with the Salesforce platform to streamline data capture and processing.

4. Role and Profile Creation: Roles and profiles were created to manage user access and permissions.

5. Property Details App: A Property Details app was created to provide a centralized platform for managing property information.

6. Approval Process and Record Trigger Flow: An approval process was created for the Property object, and a record trigger flow was implemented to automatically submit the approval process.

7. App Page and LWC Component: An app page was created, and an LWC component was developed and added to the app page.

8. Apex Class Access: Access to Apex classes was granted to profiles based on user roles and responsibilities.

This project has demonstrated the capabilities of Salesforce in managing client and property information. The custom-built application has streamlined data capture, processing, and analysis, enabling efficient management of clients and properties.