# MOOKAMBIGAI COLLEGE OF ENGINEERING

Srinivasa Nagar , Kalamavur , Pudukkottai  District – 622502.

# SMARTINTERNZ

# PROJECT TITLE: LEASE MANAGEMENT

College Code : 8128

TEAM ID:  NM2025TMID04855

TEAM SIZE: 4

TEAM MEMBERS:

Team Leader        :       Manikandan R
                                   (812822104043)

Team  Member 1  :        Mohamed Yacoob N
                                    (812822104047)

Team  Member 2  :       Mohanraj S
                                   (812822104048)

Team  Member 3  :       Muthamizhselvan G
                                    (812822104049)

# MOOKAMBIGAI COLLEGE OF ENGINEERING
## Srinivasa Nagar, Kalamavur – 622 502.



## BONAFIDE CERTIFICATE

Certified that this is a bonafide record of the practical work done by
……………………………………………….Reg.No………………………….
in ……... semester in the .....…...............……………………………………………
laboratory during the academic year 20 ..… - 20......

**Course Coordinator**                                    **Head of the department**

Submitted for the University Practical Examination held on...............……………

**Internal Examiner**                                    **External Examiner**

# Title: Lease Management System

## Project Overview:

Real estate and asset leasing companies often struggle to manage lease agreements, tenant details, payment schedules, and property maintenance efficiently. Manual processes can lead to errors, missed renewals, and poor customer experience.
To address these challenges, this project leverages **Salesforce CRM** to build a **Lease Management System** that automates lease operations, manages property and tenant data, tracks rent payments, and provides insightful reports for decision-making.

The system uses **custom objects, workflows, automation, and dashboards** to ensure seamless lease tracking, automated renewal alerts, payment monitoring, and efficient communication between tenants, landlords, and administrators.
This project demonstrates how Salesforce technology can enhance operational efficiency, reduce manual workload, and improve transparency in lease operations.

## Objectives:

- **Centralized Lease Data:** Maintain detailed records of leased properties, tenants, contracts, and payment history.
- **Automate Lease Renewal Alerts:** Trigger notifications before lease expiry to avoid revenue loss.
- **Efficient Payment Tracking:** Monitor rent payments, pending dues, and generate payment receipts.
- **Enhance Communication:** Enable real-time updates between tenants and property managers.
- **Improve Operational Efficiency:** Reduce manual recordkeeping through automation and reports.
- **Role-Based Access Control:** Allow secure access to admins, agents, and tenants.
- **Data-Driven Insights:** Generate reports on lease performance, revenue trends, and occupancy rates.

## Student Outcomes:

- **Practical Salesforce Skills:** Students learn to configure custom objects, flows, validation rules, and dashboards specific to property leasing.
- **Real-World CRM Experience:** Understand the complete lifecycle of lease management in a CRM environment.
- **Problem-Solving Ability:** Develop skills to automate reminders, payment tracking, and lease renewals using Salesforce tools.
- **Team Collaboration:** Gain experience in group project development, testing, and deployment.
- **Industry Readiness:** Acquire exposure to Salesforce CRM applications in the real estate domain.

## System Requirements

### Hardware Requirements:

- Computer with minimum 4 GB RAM (8 GB recommended)
- Dual-core processor
- Stable internet connection

### Software Requirements:

- Salesforce Developer Edition Org
- Modern Web Browser (Google Chrome, Firefox, Edge)

## Phases Overview:

| Phase No | Phase Name | Phase Description |
|---|---|---|
| 1 | Requirement Analysis & Planning | Gathering requirements from landlords, tenants, and property managers; defining scope, data model, and workflows. |
| 2 | Salesforce Development – Backend | Creating custom objects(Property,Tenant,Lease Agreement,Payment),fields,relationship, and automation. |

| 3 | UI/UX Development | Building a Lightning App for lease management, designing page layouts, and customizing record pages. |
|---|---|---|
| 4 | Testing & Security | Creating users, profiles, and permission sets; configuring sharing rules and validating data integrity. |
| 5 | Deployment & Maintenance | Deploying to production, training users, and maintaining data consistency. |

## Phase 1: Requirement Analysis & Planning

### Project Goal:

To automate and manage property leasing processes using Salesforce CRM for efficient rent tracking, renewal reminders, and property management.

### Key Objectives:

- Manage property and tenant records.
- Automate payment and renewal alerts.
- Generate reports for revenue and occupancy.
- Enable secure communication between involved parties.

## Phase 2: Salesforce Development – Backend

### Milestone 1: Salesforce Developer Account Creation:

### Activity 1: Creating Developer Account
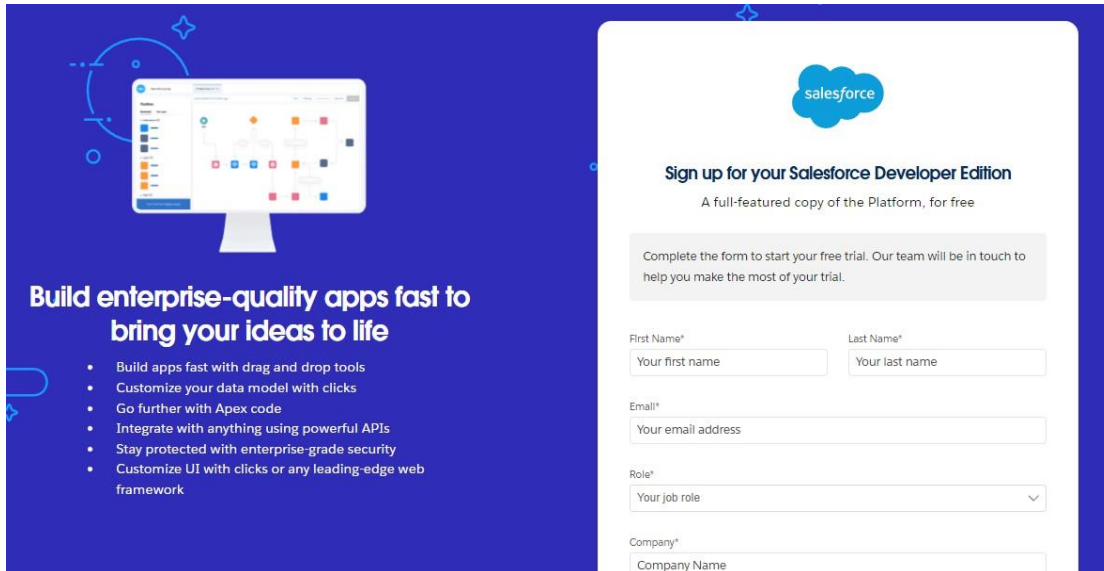
Creating a developer org in salesforce.

1. Go to https://developer.salesforce.com/signup
2. On the sign up form, enter the following details :
   1. First name & Last name
   2. Email

3. Role : Developer
4. Company : College Name
5. County : India
6. Postal Code : pin code
7. Username : should be a combination of your name and company

This need not be an actual email id, you can give anything in the format
: username@organization.com
Click on sign me up after filling these.



## Activity  2: Account Activation:

1. Go to the inbox of the email that you used while signing up. Click on the verify account to activate your account. The email may take 5-10mins.
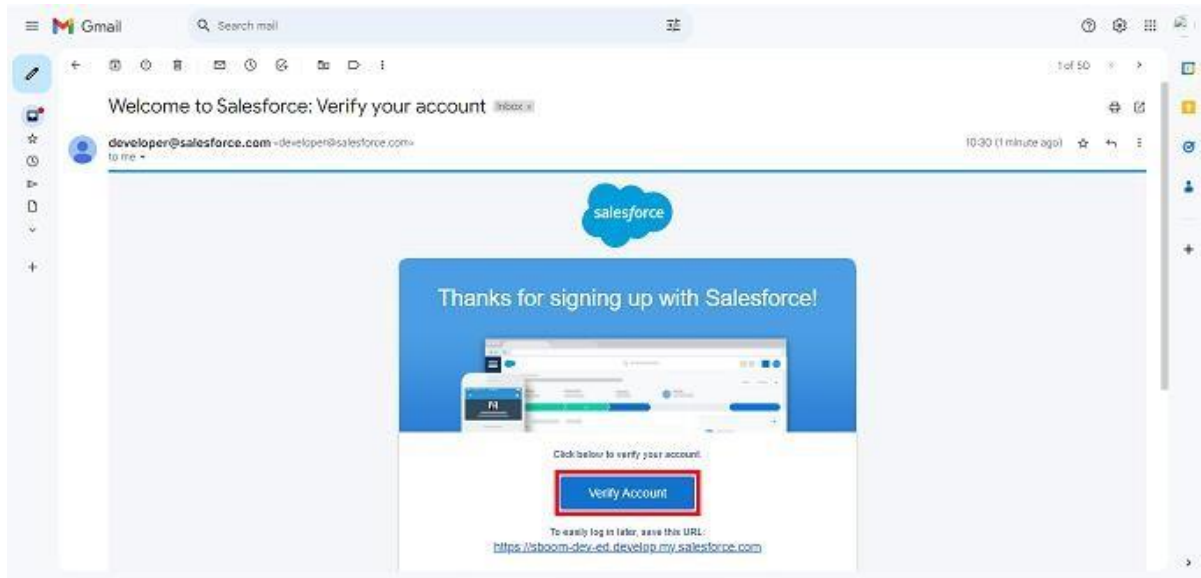
1. Click on Verify Account
2. Give a password and answer a security question and click on change password.
3. Give a password and answer a security question and click on change password.
4. Then you will redirect to your salesforce setup page.
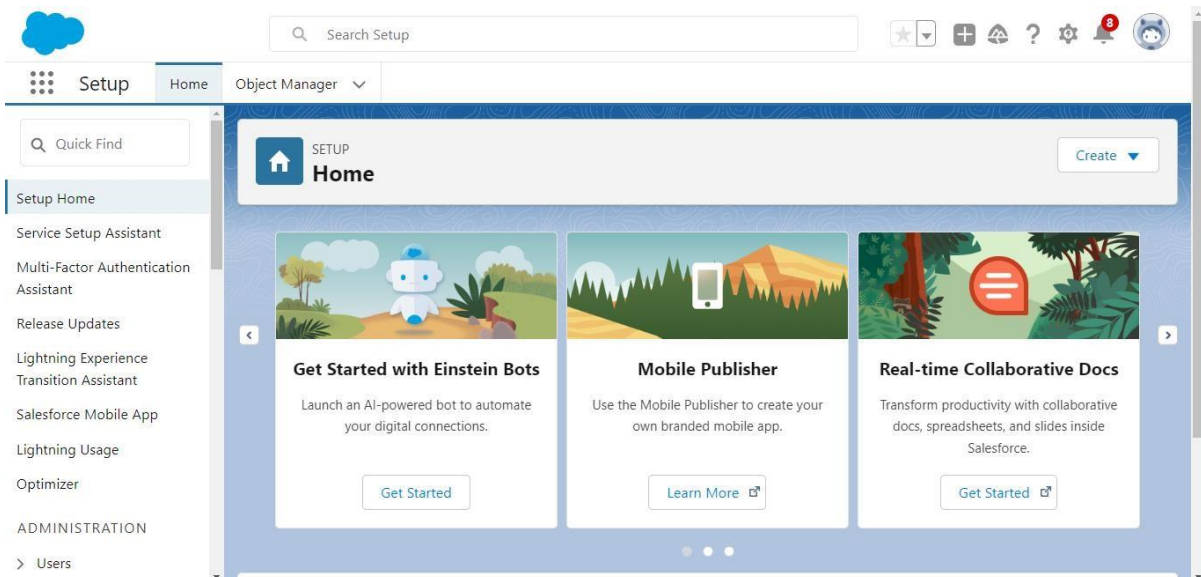


# Milestone 2: Objects

## Activity 1: Create Property Object

To create an object:

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
    1. Enter the label name>> property
    2. Plural label name>> property
    3. Enter Record Name Label and Format
        - Record Name >>property Name
        - Data Type >> Text
2. Click on Allow reports and Track Field History,Allow Activities
3. Allow search >> Save.

## Activity 2: Create Tenant Object

To create an object:

1. From the setup page >> Click on Object Manager >>Click on Create >> Click on Custom Object.
    1. Enter the label name>> Tenant
    2. Plural label name>> Tenants
    3. Enter Record Name Label and Format
        - Record Name >> Tenant Name
        - Data Type >> Text
2. Click on Allow reports and Track Field History,Allow Activities
3. Allow search >> Save.

## Activity 3: Create Payment Object

To create an object:

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
    1. Enter the label name>> Payment for tenanat
    2. Plural label name>> Payment
    3. Enter Record Name Label and Format
        - Record Name >> Payment Name
        - Data Type >> Text
2. Click on Allow reports and Track Field History,Allow Activities
3. Allow search >> Save.

## Activity 4: Create Lease Object

To create an object:

1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
    1. Enter the label name>> lease
    2. Plural label name>> lease
    3. Enter Record Name Label and Format
        - Record Name >> lease Name
        - Data Type >> Text
2. Click on Allow reports and Track Field History,Allow Activities
3. Allow search >> Save

# Milestone 3: Tabs

## Activity 1: Creating a Custom Tab

To create a Tab:( Property)
1. Go to setup page >>type Tabs in Quick Find bar >> click on tabs >> New (under custom object tab)



1. Select Object( property) >> Select the tab style >> Next (Add to profiles page) keep it as default >> Next (Add to Custom App) uncheck the include tab .
2. Make sure that the Append tab to users' existing personal customizations is checked.
3. Click save

## Activity 2: Creating Remaining Tabs

1. Now create the Tabs for the remaining Objects, they are "Payment for tenant,lease,tenant".
2. Follow the same steps as mentioned in Activity -1 .

# Phase 3: UI/UX Development

## Milestone 4: Lightning App

## Activity 1:  Create a Lightning App

To create a lightning app page:

1. Go to setup page >> search "app manager" in quick find >> select "app manager" >> click on New lightning App.
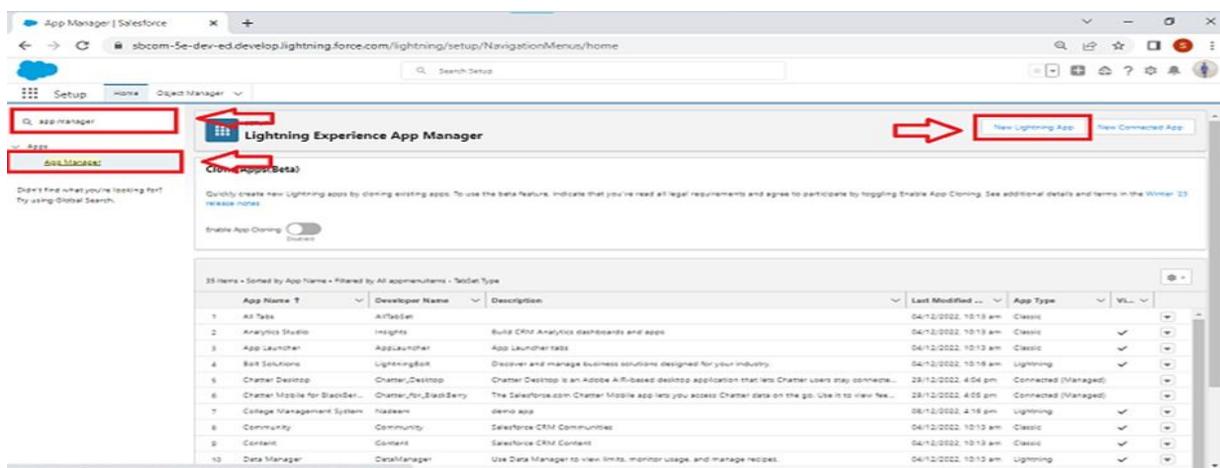


2. Fill the app name in app details and branding as follow

App Name : Lease ManagementDeveloper Name : This will auto populatedImage : optional (if you want to give any image you can otherwise not mandatory) Primary colour hex value : keep this default.

3.Then click Next  >> (App option page)Set Navigation Style as Standard Navigation >> Next. 4. Utility Items keep it as default >> Next.
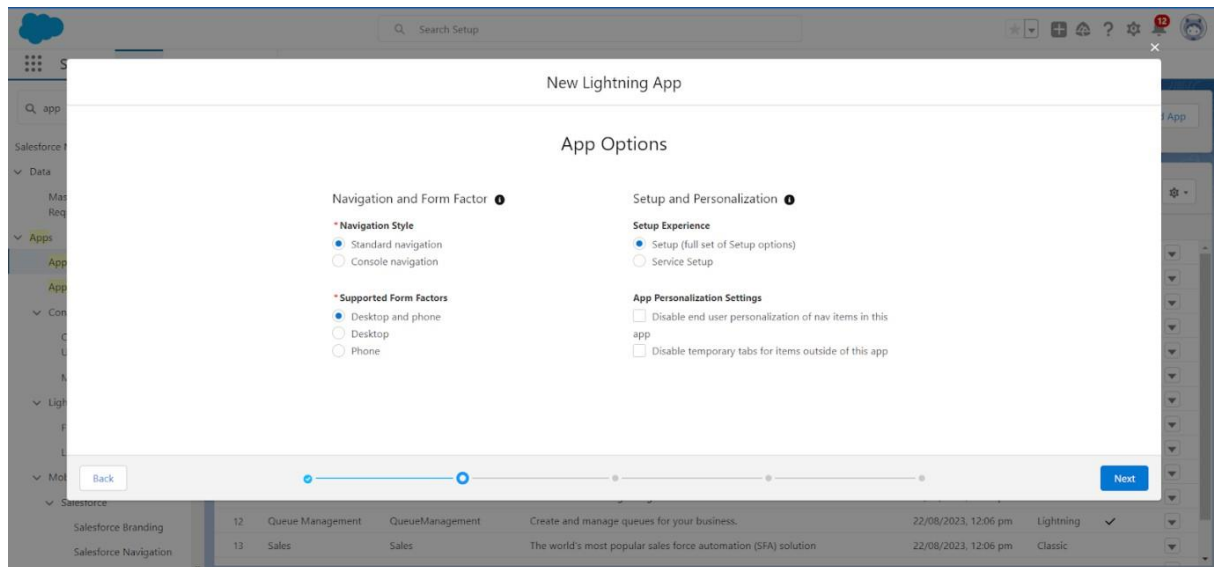
5. To Add Navigation Items:
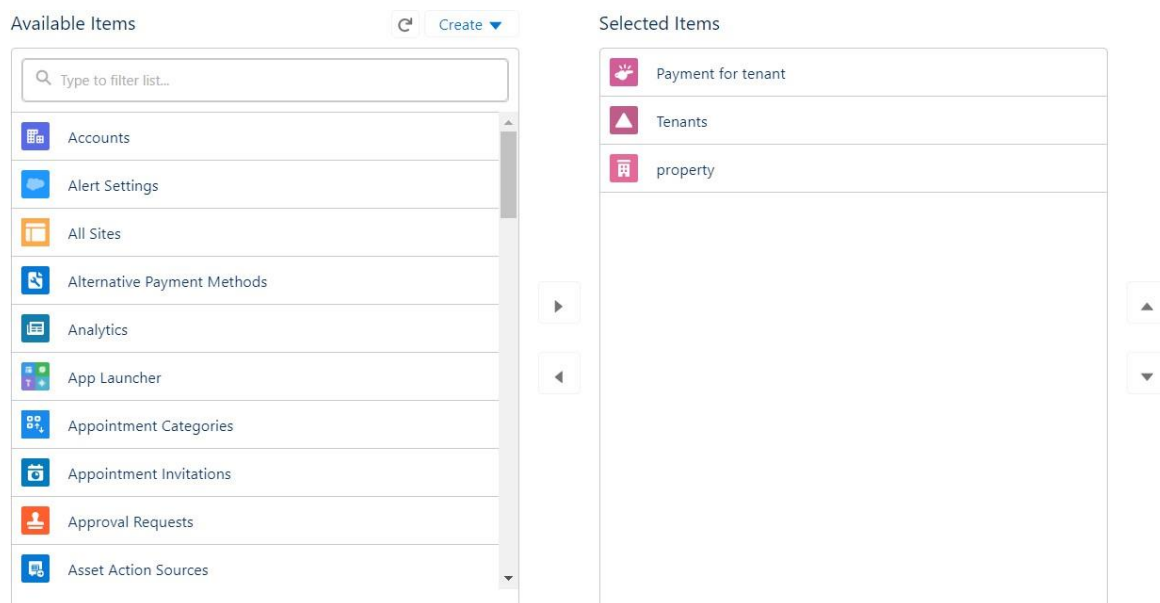


   Search for the item in the (Payment for tenant, Tenants,property,lease) from the search bar and move it using the arrow button ? Next? Next.

6. To Add User Profiles:
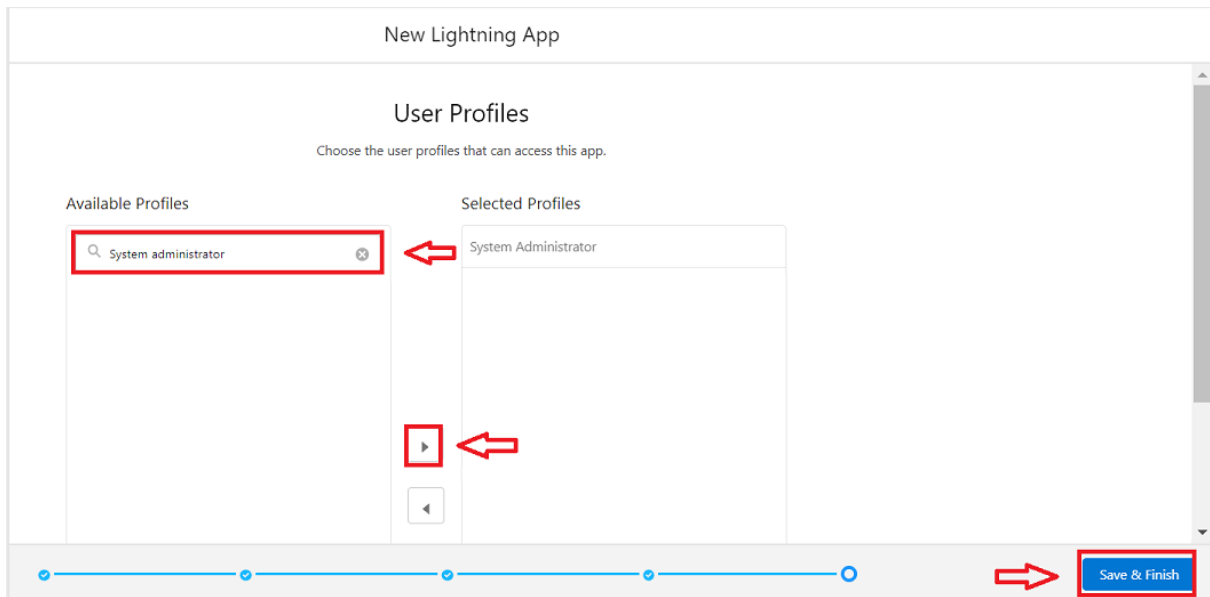
Search profiles (System administrator) in the search bar >>click on the arrow button >> save & finish.
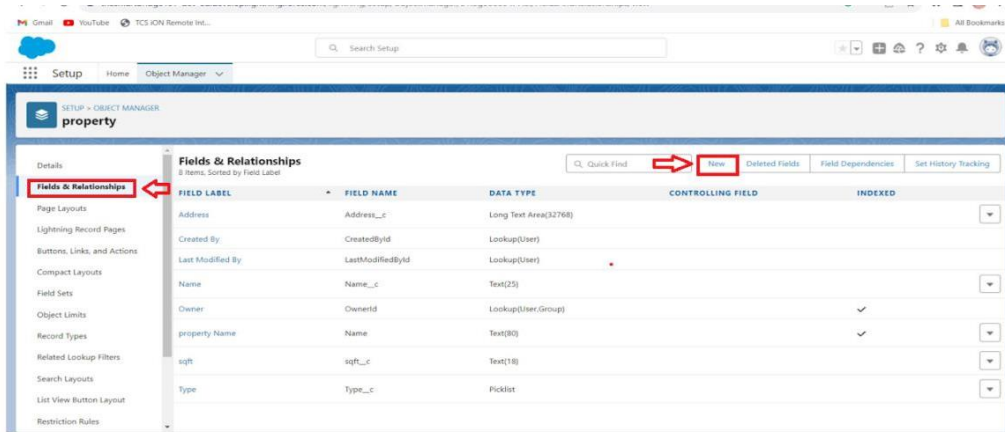
## Milestone 5: Fields

## Activity 1: Creation of fields for the property object
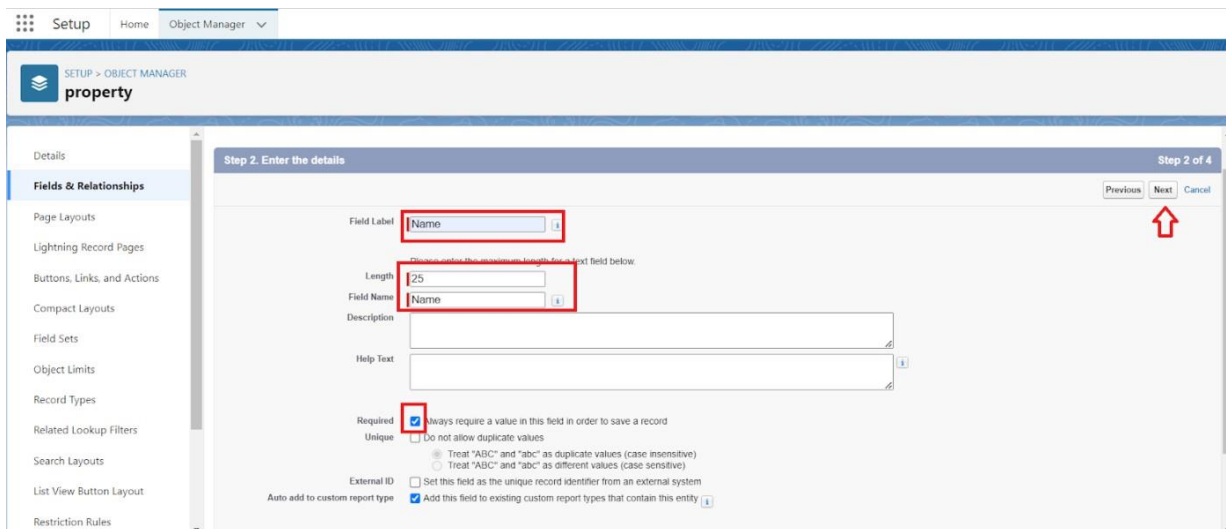
To create fields in an object:
1. Go to setup >> click on Object Manager >> type object name(property) in search bar >>click on the object.
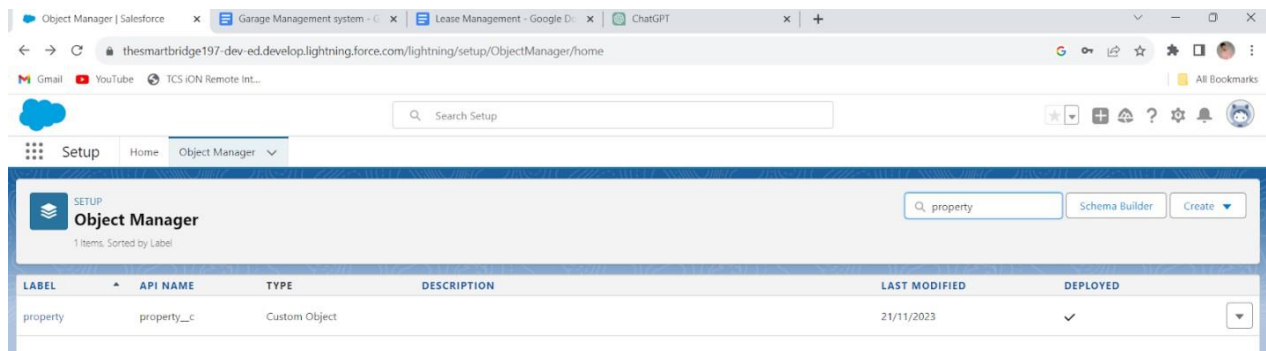


2.      Now click on "Fields & Relationships" >> New

3. Select Data Type as a "Text"
4. Click on next
5. Fill the Above as following:
   - Field Label: Name
   - Field Name : gets auto generated
   - Length : 25
   - Required :check box
   - Click on Next >> Next >> Save and new.



2. To create another fields in an object:

1. Go to setup >> click on Object Manager >>type object name(property) in search bar >>click on the object.
2. Now click on "Fields & Relationships" >>New

3. Select Data type as a "Long Text" and Click on Next
4. Fill the Above as following:
   - Field Label : Address
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

3. To create another fields in an object:
   5. Go to setup >> click on Object Manager >>type object name(property) in search bar >> click on the object.
6. Now click on "Fields & Relationships" >> New
7. Select Data type as a "picklist" and Click on Next
8. Fill the Above as following:
   - Field Label : Type
   - Field Name : gets auto generated
   - Enter values, with each value separated by a new line
   - Enter these values
     1BHK
     2BHK
     3BHK
   - Click on Next >> Next >> Save and new.

To create another fields in an object:
9. Go to setup >> click on Object Manager >> type object name(property) in search bar >> click on the object.
10. Now click on "Fields & Relationships" >> New
11. Select Data type as a " Text" and Click on Next
12. Fill the Above as following:
   - Field Label : sfqt
   - Field Name : gets auto generated
   - Length : 18
   - Click on Next >> Next >> Save.

## Activity 2: Creation of fields for the Tenant object

1. Go to setup >> click on Object Manager >> type object name(Tenant) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "Email" and Click on Next
4. Fill the Above as following:
   - Field Label : Email
   - Field Name : gets auto generated
   - Click on required check box

- Click on Next >> Next >> Save and new.

<u>To create another fields in an object:</u>

1. Go to setup >> click on Object Manager >> type object name(Tenant) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "phone" and Click on Next
4. Fill the Above as following:
   - Field Label : Phone
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

<u>To create another fields in an object:</u>

5. Go to setup >> click on Object Manager >> type object name(Tenant) in search bar >> click on the object.
6. Now click on "Fields & Relationships" >>New
7. Select Data type as a "picklist" and Click on Next
   8. Fill the Above as following:
   - Field Label : status
   - Field Name : gets auto generated
   - Enter values, with each value separated by a new line
   - Enter these values
        Stay
        Leaving
   - Click on Next >> Next >> Save
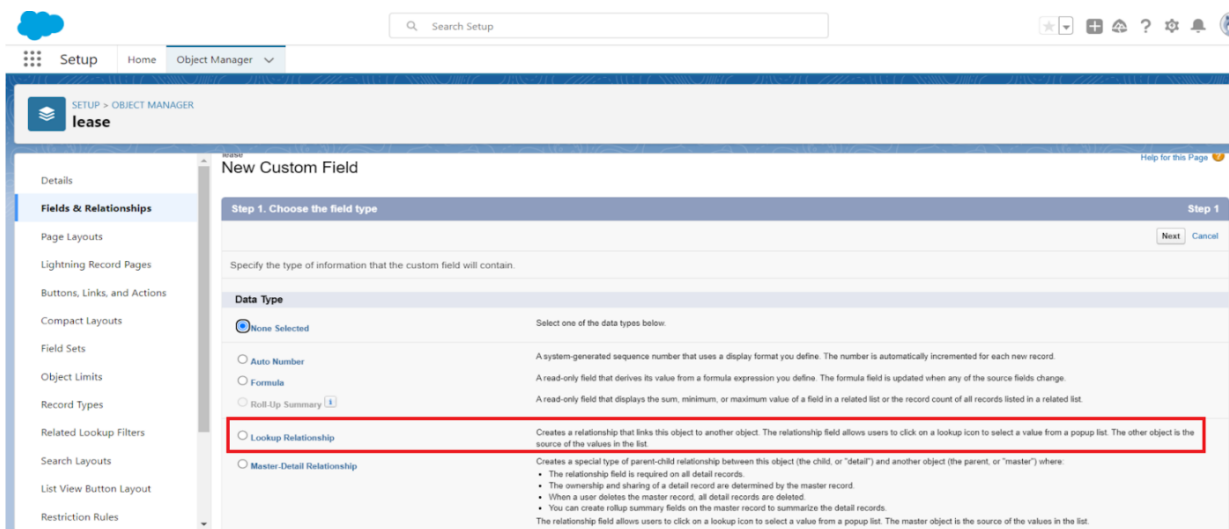

## Activity 3: Creation of fields for the Lease object

1. Go to setup >> click on Object Manager >> type object name(Lease) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "Date" and Click on Next
4. Fill the Above as following:
   - Field Label : start date
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

<u>To create another fields in an object:</u>

1. Go to setup >> click on Object Manager >> type object name(Lease) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "Date" and Click on Next
4. Fill the Above as following:
   - Field Label : End date
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

## Activity 4: Creation of fields for Payment for the Tenant object

1. Go to setup >> click on Object Manager >> type object name(Payment for tenant) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "Date" and Click on Next
4. Fill the Above as following:
   - Field Label : Payment date
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

To create another fields in an object:
1. Go to setup >> click on Object Manager >> type object name(Payment for tenant) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "Number" and Click on Next
4. Fill the Above as following:
   - Field Label : Amount
   - Length : 18
   - Field Name : gets auto generated
   - Click on Next >> Next >> Save and new.

To create another fields in an object:
1. Go to setup >> click on Object Manager >> type object name(Payment for tenant) in search bar >> click on the object.
2. Now click on "Fields & Relationships" >> New
3. Select Data type as a "picklist" and Click on Next
4. Fill the Above as following:
   - Field Label : check for payment
   - Field Name : gets auto generated
   - Enter values, with each value separated by a new line

- Enter these values
  Paid
  Not paid
- Click on Next >> Next >> Save and new.

## Activity 5: Creation of Lookup fields

Creation of Lookup Field on Lease Object :

1. Go to setup>> click on Object Manager >> type object name( Lease) in the search bar >> click on the object.



2.    Now click on "Fields & Relationships" >> New
3.    Select lookup relationship
4.    Select the related object " property" and click next.
5.    Field Name : property
6.    Field label : Auto generated
7.    Next >> Next >> Save.

Creation of Lookup Field on Payment Object :

8.    Go to setup >> click on Object Manager >> type object name( payment) in the search  bar >> click on the object.
9.    Now click on "Fields & Relationships" >> New
10.    Select lookup relationship
11.    Select the related object " Tenant" and click next.
12.    Field Name : Tenant

13. Field label : Auto generated
14. Next >> Next >> Save.


Creation of Lookup Field on Payment for tenant Object :

15. Go to setup>> click on Object Manager >> type object name( property) in the search bar >> click on the object.
16. Now click on "Fields & Relationships" >> New
17. Select masterdetail relationship
18. Select the related object " property" and click next.
19. Field Name : property
20. Field label : Auto generated
21. Next >> Next >> Save.

# Phase 4: Testing & Security

## Milestone 6: Validation Rules

### Activity 1: To create a validation rule to an Lease Object

1. Go to the setup page >> click on object manager >> From drop down click edit for Lease object.
2. Click on the validation rule >> click New.
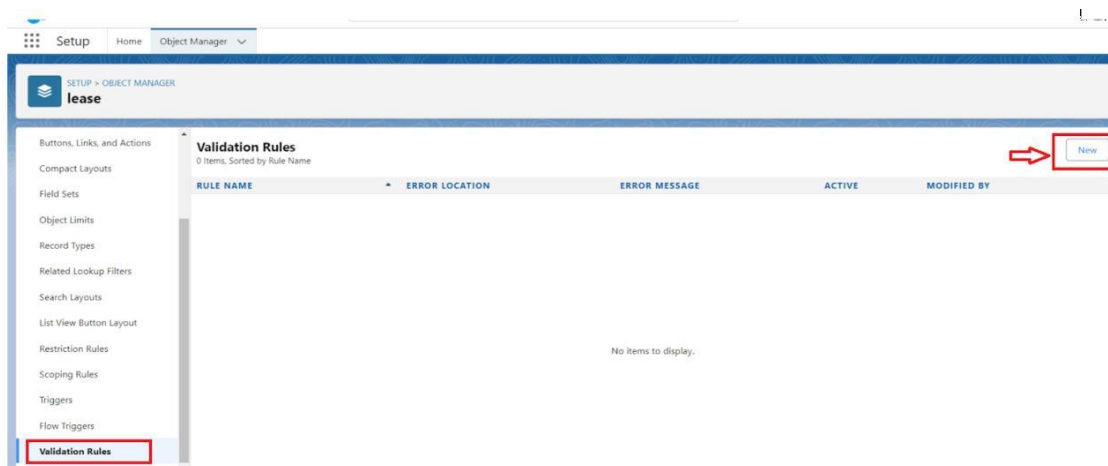


3. Enter the Rule name as " lease_end_date".
4. Insert the Error Condition Formula as :

End_date__c >
start_date__c
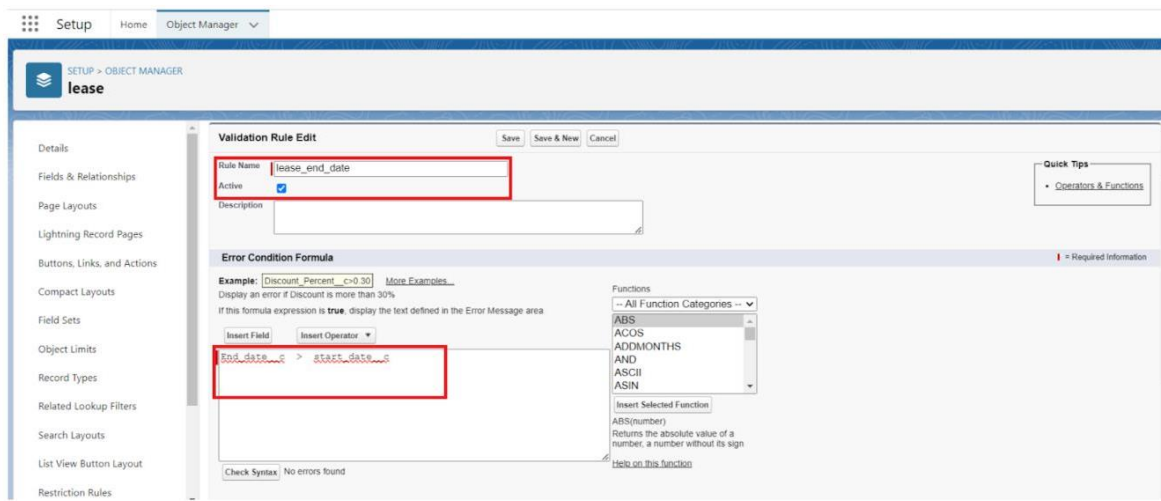
5. Enter the Error Message as "Your End date must be greater than start date", select the Error location as Field and select the field as "start date", and click Save.
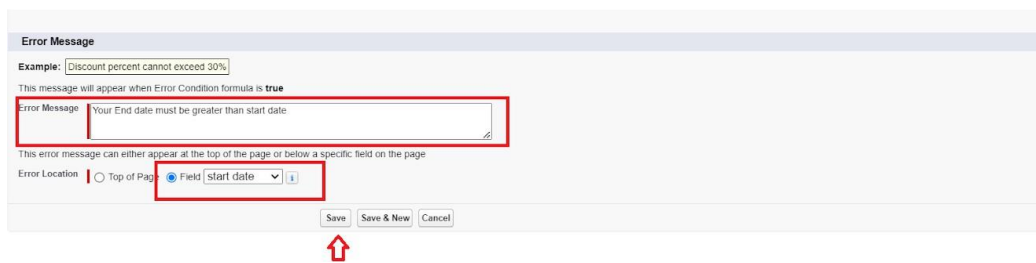


## **Milestone 7: Email Templates**

### **Activity 1: Create Email Template For Tenant Leaving**

To create Email Template:

1. Go to setup in quick find box enter email template >> click on classic Email Template.

2. Click on >> New Email Template===>Choose text

   **Folder : Unfiled public Classic Email templates**
   Click on available for use

3.  Email Template Name is "tenant  leaving"

4. Template Unique Name : Auto populated

5.  Subject : " request for approve the leave"
6. Email body :

**Dear {!Tenant_c.CreatedBy},**
**Please approve my leave.**

7. Save

## Activity 2: Create Email Template For Leave Approved

To create Email Template:

1.  Go to setup in quick find box enter email template >> click on classic Email
Template.

2.  Click on >> New Email Template===>Choose text

**Folder : Unfiled public Classic Email templates**

Click on available for use

3.  Email Template Name is "Leave approved"

4. Template Unique Name : Auto populated
5.  Subject : " Leave approved"
6. Email body :

**dear{!Tenant_c.Name},**

**I hope this message finds you well. I am writing to inform you that I have received
your email confirming the approval of my leave request. I would like to express my
gratitude for considering and approving my time off.**

**your leave is approved. You can leave now**

7. Save

## Activity 3: Create Email Template For Rejection for Leave

To create Email Template:
1.  Go to setup in quick find box enter email template >> click on classic Email Template.

2.  Click on >>New Email Template===>Choose text

   **Folder : Unfiled public Classic Email templates**
   Click on available for use

3.  Email Template Name is "Leave rejected"

4. Template Unique Name : Auto populated

5.  Subject : " Leave rejected"
6. Email body :
   **Dear {!Tenant__c.Name},**
   **I hope this email finds you well. Your contract has not ended. So we can't approve your leave.**
   **your leave has rejected**
7. Save

## Activity 4: Create Email Template For Monthly Payment

To create Email Template:

1.  Go to setup in quick find box enter email template >> click on classic Email Template.

2.  Click on >> New Email Template===>Choose text

   **Folder : Unfiled public Classic Email templates**
   Click on available for use

3.  Email Template Name is "Tenant Email"

4. Template Unique Name : Auto populated

5.  Subject : " Urgent: Monthly Rent Payment Reminder"
6. Email body :
**Dear {!Tenant__c.Name},**
  **I trust this email finds you well. We appreciate your continued tenancy at our property and I hope you have been comfortable in your residence.**

**This communication is a friendly reminder regarding your monthly rent payment, which is currently outstanding. As outlined in our rental agreement, the payment is due . To ensure the smooth operation of our property management and to avoid any inconvenience, we kindly request you to settle the payment at your earliest convenience**.

7. Save

## **Activity 5: Create Email Template For Successful Payment**

To create Email Template:

1.  Go to setup in quick find box enter email template >> click on classic Email Template.

2.  Click on >> New Email Template===>Choose text

   **Folder : Unfiled public Classic Email templates**
Click on available for use

3.  Email Template Name is "tenant payment"

4. Template Unique Name : Auto populated
5.  Subject : " Confirmation of Successful Monthly Payment"
6. Email body :
**Dear {!Tenant__c.Email__c},**

**We hope this email finds you well. We are writing to inform you that we have successfully received your monthly payment. Thank you for your prompt and diligent payment.**

7. Save

## **Milestone 8: Approval Processes**

### **Activity 1: Create Approval Process For check for vacant**

To create fields in an object:

1. Go to setup >> Approval Processes in quick find bar>>click on it.

2. Manage Approval Process For >> "Tenant" from the drop down.

3. Click on "Create New Approval  Process" >> Use standard setup wizard.

4.  Process Name "check for vacant" >> Click Next



.
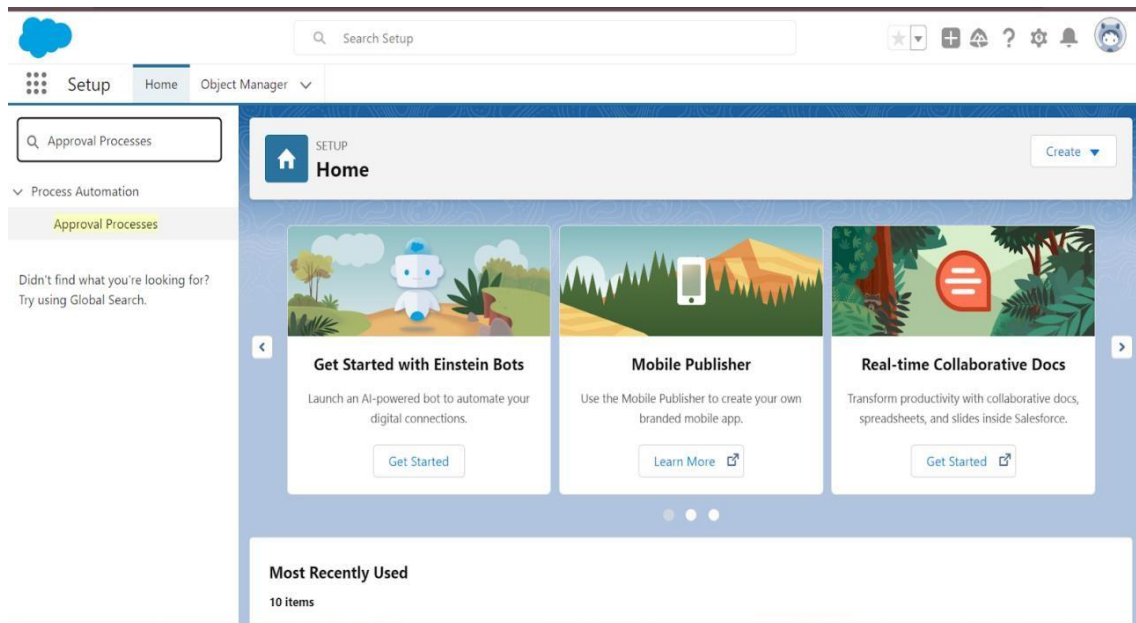
5.  Field "Tenant:status" >> Operator : Not equals , Value >> Click on the lookup filter icon and select "Leaving".
6. Click insert field,then click Next.
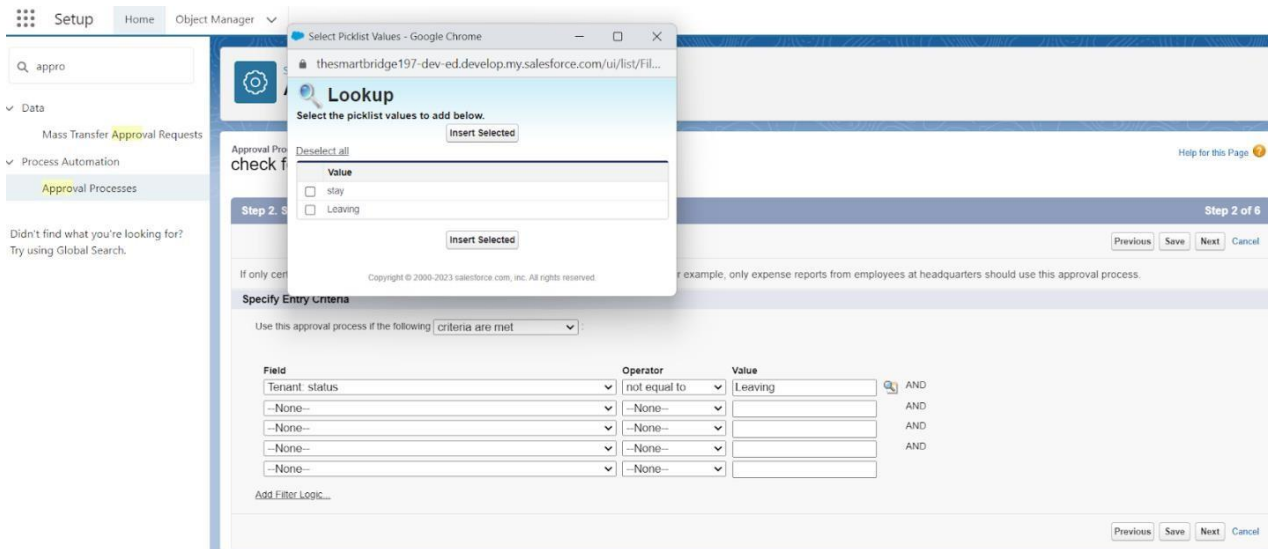
7. Next Automated Approver determined by "None" from the drop down.

8. Select the "Administrators ONLY can edit records during the approval process".Then Next.
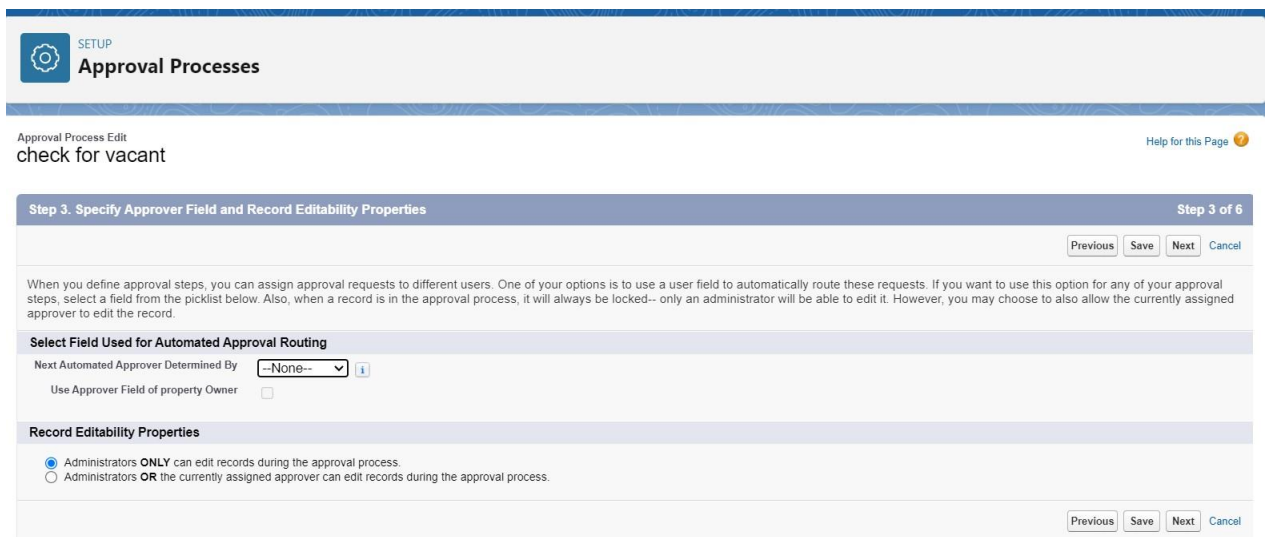


9. Click on next leave the email template click on next

10. From the available fields select >> Tenant Name, and then add >>Add it to the selected.Then Next.

- Make sure Display approver history is checked.
- And under security settings check the "Allow approvers to access the approval page only from within the Salesforce application. (Recommended)" option.

11. Submitter type Search>>Owner, Allowed Submitters>>Property Owner.Then Next.

- Then click save.



- Click on "i'll do this later. Take me back to the listing of all approval process for this object"
- Click go

## Activity 2: Initial Submission Action

1. Under initial submission action click on add new and then select email alert.



2. Description: "please approve my leave".
3. Unique name : auto populated
4. Email template : tenant leaving
5. Recipient type : Email field
6. Available Recipients : Email field : Email
7. From Email address : Current user's email

8. Click save

## Activity 3: Final Approved Action

1. Under Final approval action click o n new and then select email alert.

2. Description: "Tenant leaving".

3. Unique name : auto populated

4. Email template : Leave approved

5. Recipient type : Email field

6. Available Recipients : Email field : Email

7. From Email address : Current user's email

8. Click save

## Activity 4: Final Rejection Action

1. Under final rejection action click on add new and then select email alert.
2. Description: "your request for leave is rejected".
3. unique name : auto populated
4. Email template : leave rejected
5. Recipient type : Email field
6. Available Recipients : Email field : Email
7. From Email address : Current user's email
8. Click save

## Milestone 8: Apex triggers

## Activity 1: Create an Apex trigger

1. To create a new Apex Class follow the below steps:
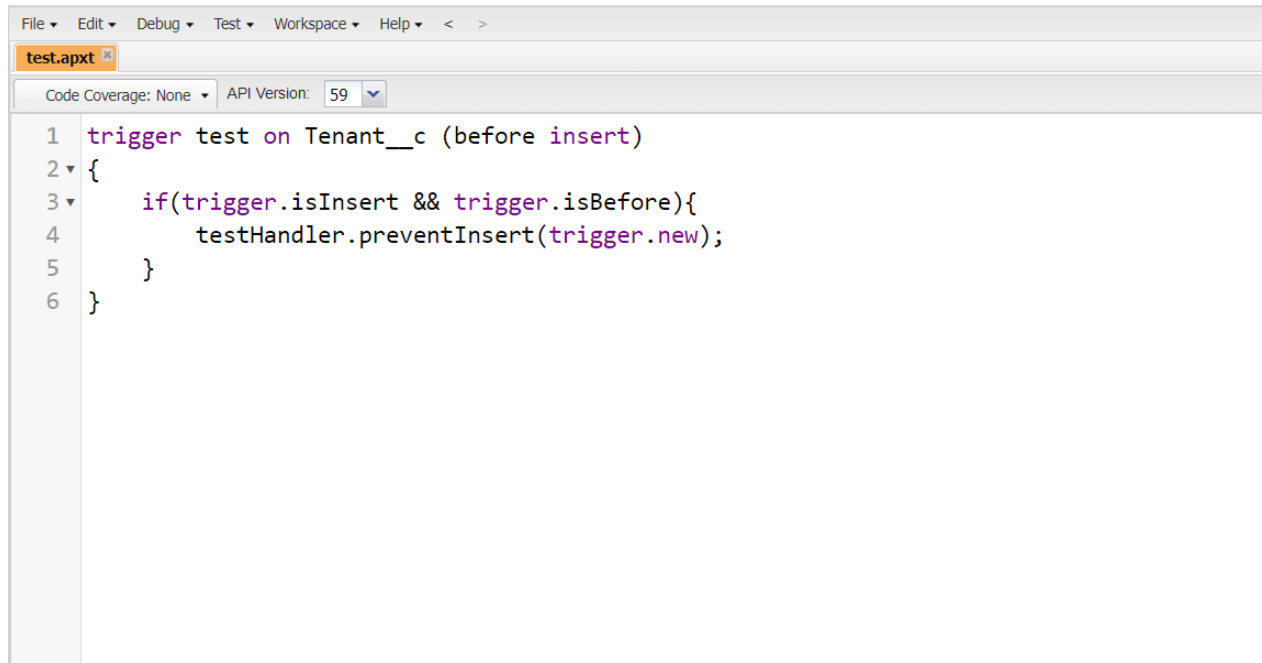
Click on the file >> New ? Apex Class.
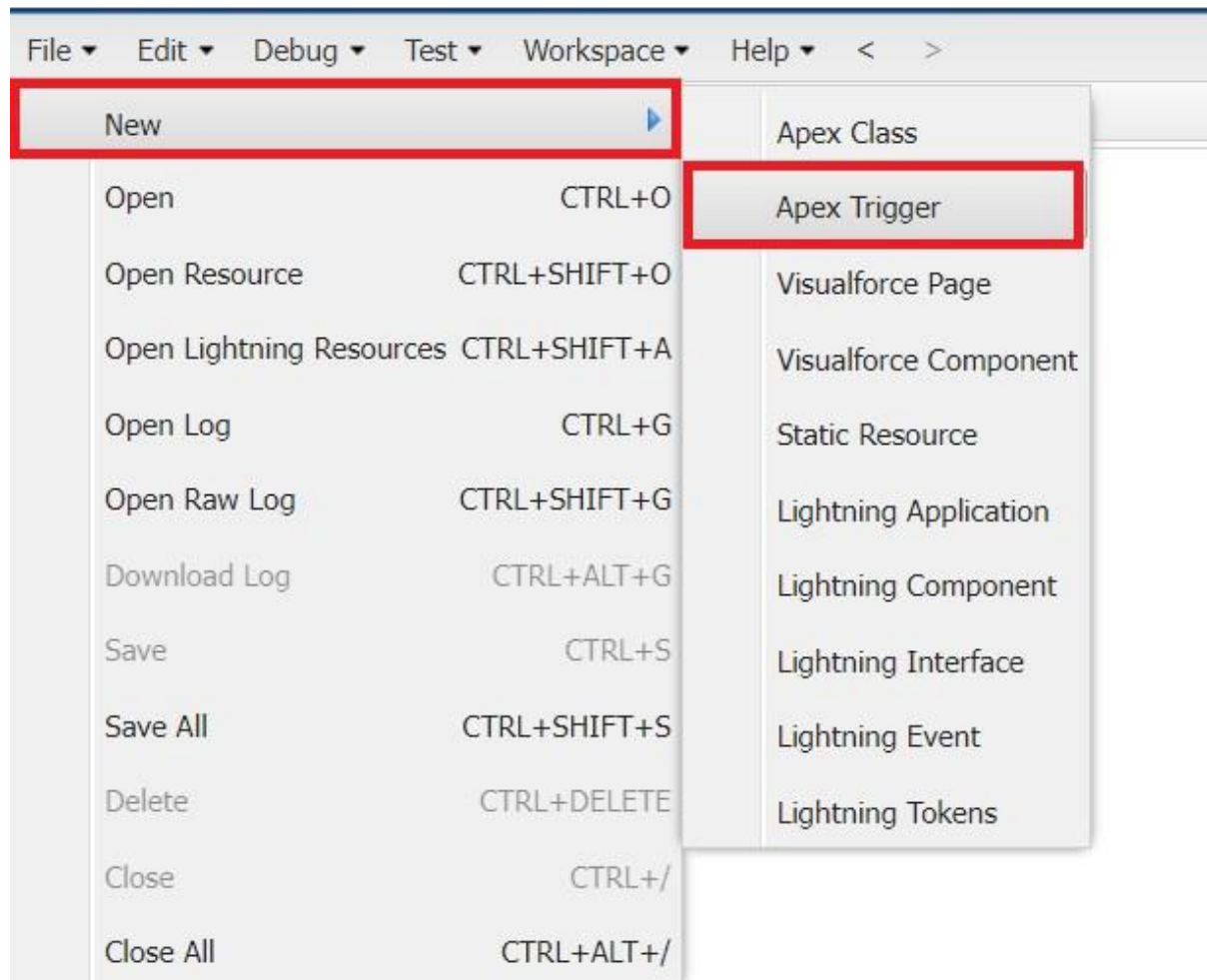
2. Give the Apex Trigger name as "test", and select "Tenant__c" from the dropdown for sObject.
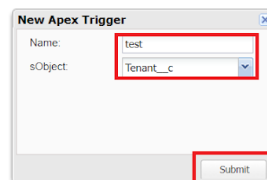


3. Click Submit.
4. Now write the code logic here
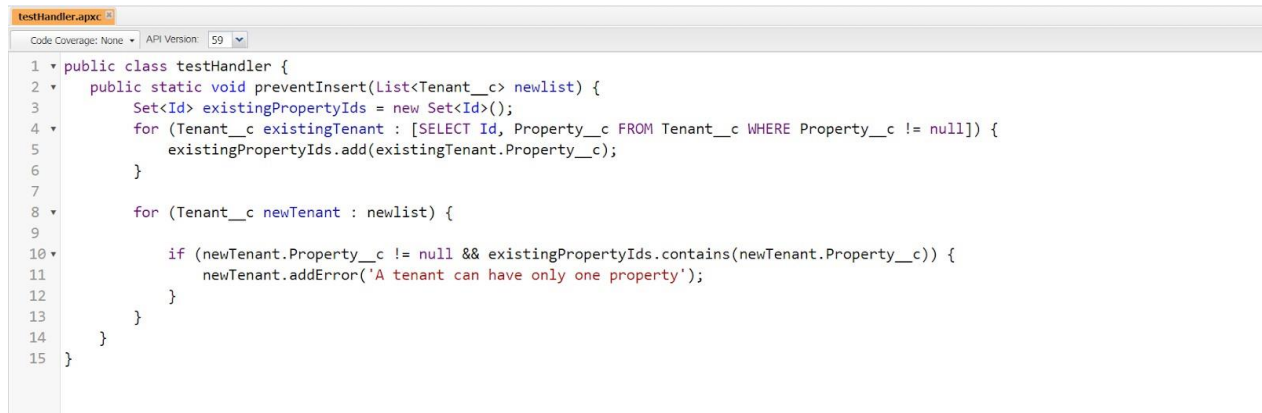
```
1   trigger test on Tenant__c (before insert)
2 ▾ {
3 ▾     if(trigger.isInsert && trigger.isBefore){
4           testHandler.preventInsert(trigger.new);
5       }
6   }
```

**Trigger Code:**

```
trigger test on Tenant__c (before insert)
{
  if(trigger.isInsert && trigger.isBefore){
     testHandler.preventInsert(trigger.new);
  }
}
```

## Activity 2: Create an Apex Handler Class

1. To create a new Apex Class follow the below steps:
   Click on the file >> New >>Apex Class.
2. Enter class name as testHandler.

```
1 ▾ public class testHandler {
2 ▾     public static void preventInsert(List<Tenant__c> newlist) {
3           Set<Id> existingPropertyIds = new Set<Id>();
4 ▾         for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c WHERE Property__c != null]) {
5               existingPropertyIds.add(existingTenant.Property__c);
6           }
7
8 ▾         for (Tenant__c newTenant : newlist) {
9
10 ▾            if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
11                  newTenant.addError('A tenant can have only one property');
12              }
13          }
14      }
15  }
```

## Apex logic:

```
public class testHandler {
  public static void preventInsert(List<Tenant__c> newlist) {
    Set<Id> existingPropertyIds = new Set<Id>();
    for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c
WHERE Property__c != null]) {
        existingPropertyIds.add(existingTenant.Property__c);
    }

    for (Tenant__c newTenant : newlist) {

      if (newTenant.Property__c != null &&
existingPropertyIds.contains(newTenant.Property__c)) {
          newTenant.addError('A tenant can have only one property');
        }
     }
  }
}
```

## Activity 3: Testing the Trigger'

Try to create new tenant with the existing property then it shows the error

## Milestone 10: Flows

## Activity 1: Create Flow for monthly payment

1. Go to setup >> type Flow in quick find box >> Click on the Flow and Select the New Flow.
2. Select the record Triggered flow.Click on create.

3. Under Object select "Payment for tenant". Click on A record is updated.

### Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

\* Object

Payment for tenant

### Configure Trigger

\* Trigger the Flow When:

○ A record is created
● A record is updated
○ A record is created or updated
○ A record is deleted

4. Set Entry Conditions
   Under Condition Requirements>>All Conditions are met

| Field: check_for_payment__c | Operator: Equals | Value : paid |
|---|---|---|

5. Click on : Every time a record is updated and meets the condition requirements
6. Click on : Actions and related records,done

### Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements

All Conditions Are Met (AND)

| Field | Operator | Value | |
|---|---|---|---|
| check_for_paymet__c | Equals | paid | 🗑 |

+ Add Condition

**When to Run the Flow for Updated Records** ⓘ

● Every time a record is updated and meets the condition requirements
○ Only when a record is updated to meet the condition requirements

\* Optimize the Flow for:

**Fast Field Updates**

Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

**Actions and Related Records**

Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

7. Under record trigger flow click on "+" icon and select action
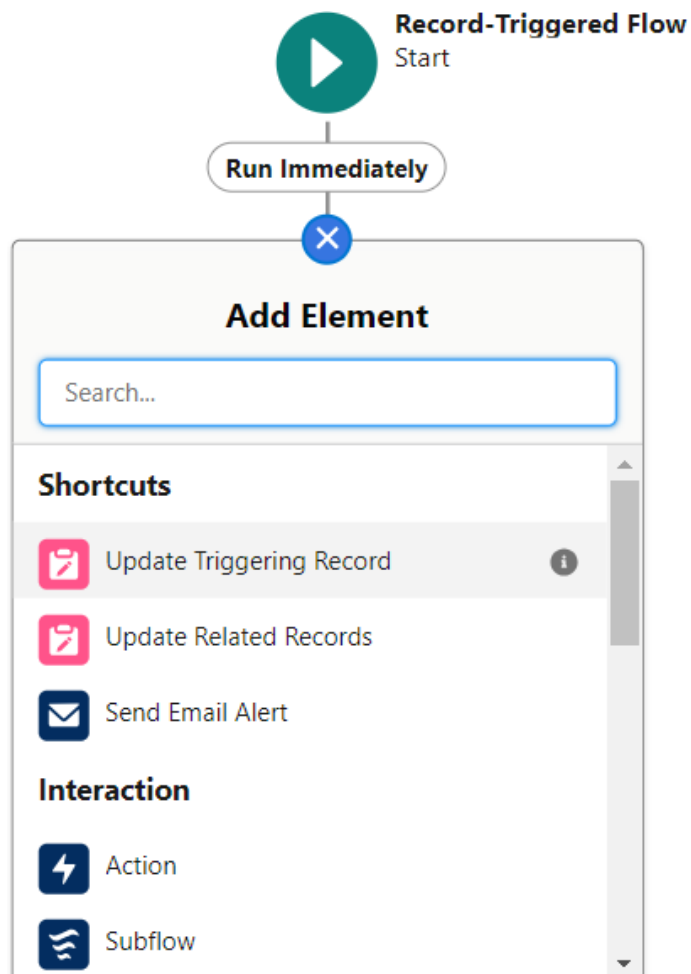


In action search for send email then click on send email (check below image)

8. Label : send email

API Name : send_email

9. Label : send email

10. API Name : send_email

11. Enable Body

12. Click on new resource
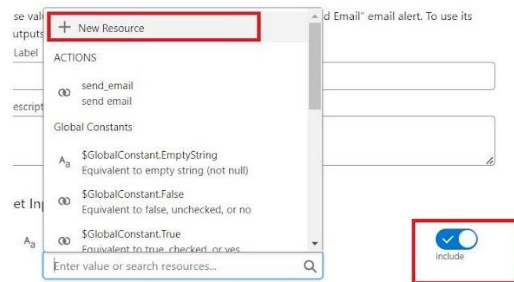


Under resource type select "Text Template"

API Name : emailbody

Under body:                 (paste the below text)

**Dear {!$Record.Tenant__r.Name},**

**We hope this email finds you well. We are writing to inform you that we have successfully received your monthly payment. Thank you for your prompt and diligent payment.**

14. Click Done.

15. Enable recipient Address List
Paste this ?{!$Record.Tenant__r.Email__c}

16. Click Done

17. Enable subject

Pate this >> Confirmation of Successful Monthly Payment

18. Click on save

Flow label : monthly payment
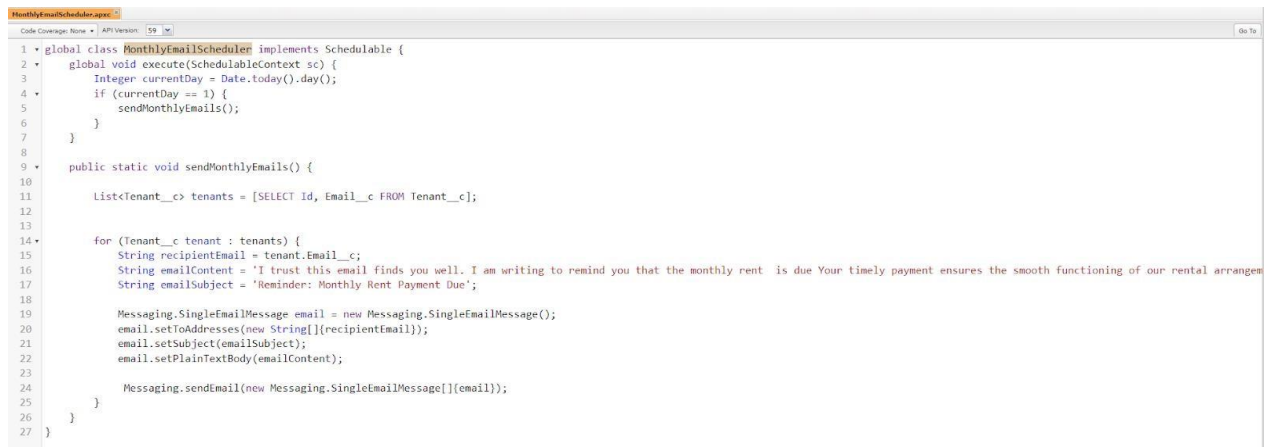
Flow API Name : monthly_payment

Click on activate

## **Milestone 11: Schedule Class**

### **Activity 1: Create an Apex class**

1. To create a new Apex Class follow the below steps:
Click on the file >> New >> Apex Class.
2. Enter class name as MonthlyEmailScheduler.



### **Apex logic:**

```
global class MonthlyEmailScheduler implements Schedulable {
    global void execute(SchedulableContext sc) {
        Integer currentDay = Date.today().day();
        if (currentDay == 1) {
            sendMonthlyEmails();
        }
    }

    public static void sendMonthlyEmails() {

        List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];


        for (Tenant__c tenant : tenants) {
            String recipientEmail = tenant.Email__c;
            String emailContent = 'I trust this email finds you well. I am writing to remind
you that the monthly rent is due Your timely payment ensures the smooth functioning of
our rental arrangement and helps maintain a positive living environment for all.';
```

```
        String emailSubject = 'Reminder: Monthly Rent Payment Due';

        Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
        email.setToAddresses(new String[]{recipientEmail});
        email.setSubject(emailSubject);
        email.setPlainTextBody(emailContent);

         Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
      }
   }
}
```

3. Save the code.


## Activity 2: Schedule Apex class

1. Enter Apex class in quick find box
2. Select schedule Apex



3. Enter job Name : MonthlyEmailScheduler
4. Apex class : MonthlyEmailScheduler
5. Frequency : Monthly===>select on day 1
6. Start date : 04/12/2023
7. End date : 04/01/2024
8. Preferred start time : 09:00 am
9. Save

## Schedule Apex

Schedule an Apex class that implements the 'Schedulable' interface to be automatically executed on a weekly or monthly interval.

| | |
|---|---|
| | Save  Cancel |
| **Job Name** | MonthlyEmailScheduler |
| **Apex Class** | MonthlyEmailScheduler 🔍 |
| **Schedule Apex Execution** | |

| | |
|---|---|
| **Frequency** | ○ Weekly  ● Monthly    ● On day [1 ▾] of every month  ○ On [the 1st ▾] [Sunday ▾] of every month |
| **Start** | 04/12/2023   [ 04/12/2023 ] |
| **End** | 04/01/2024   [ 04/12/2023 ] |
| **Preferred Start Time** | [9:00 am ▾] |
| | Exact start time will depend on job queue activity. |

Save  Cancel

## Testing the approval process:
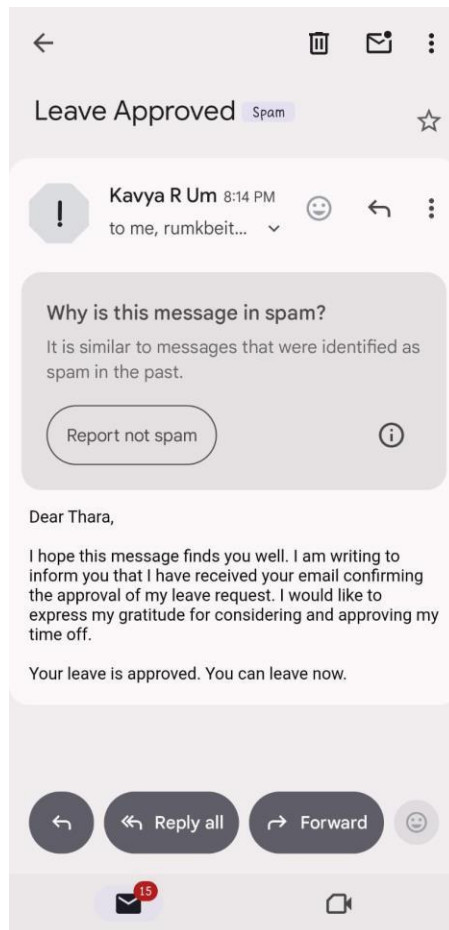


Enter any comment am\nd click on submit

Click on that notification



Click on approve
Give any comment and submit.
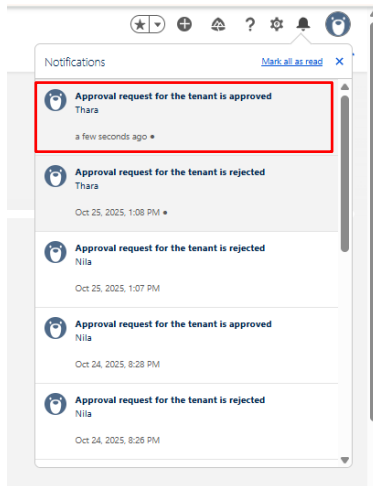
You will find notification like this and you will get an email check.

## Phase 5: Deployment & Maintenance:

Once the app was tested successfully, it was deployed to the **production environment**.

- The deployment process involved moving metadata and configurations using **Change Sets** to ensure a smooth transition from sandbox to production.
- End users were **trained** on how to use the application, manage lease records, and generate reports effectively.
- Continuous **maintenance and monitoring** were established to ensure data consistency, system performance, and to apply updates or enhancements as needed.
- Regular feedback collection and system audits were also planned to identify potential improvements and maintain high user satisfaction

## Testing Summary:

- All functional modules worked as designed.
- Automation flows and approval processes executed successfully.
- Emails were delivered correctly (occasionally redirected to spam folder).
- Validation rules effectively ensured data integrity.
- Overall application performance and response were stable and efficient.

## Limitations:

Despite successful implementation, the current version has certain limitations:

1. **Limited Access Control:** Approval and notification rights are restricted primarily to system administrators.
2. **Email Delivery Issues:** Some system-generated emails may be marked as spam depending on recipient settings.
3. **Lack of Integration:** No integration with external payment or lease accounting systems.
4. **No Mobile Optimization:** The system is functional in desktop browsers but not optimized for Salesforce mobile view.

## Future Enhancements:

1. **Add Payment Integration:**
   Connect the system with an online payment option so that tenants can pay their rent directly and automatically update their payment status.
2. **Improve Email Notifications:**
   Make the email templates more attractive and add more details such as payment amount, due date, and tenant name for better communication.
3. **Add Reports and Dashboards:**
   Create simple dashboards to show the number of active leases, total payments received, and tenants who haven't paid.
4. **Mobile Access:**
   Make the system easier to use on mobile devices using Salesforce mobile view for property owners and tenants.
5. **Lease Renewal Reminders:**
   Add an automatic reminder email or notification when a lease is about to expire, so tenants can renew it on time.

## Conclusion:

The **Lease Management System** project successfully demonstrates the end-to-end use of Salesforce CRM capabilities for real-world business automation.
It automates tenant record management, lease tracking, approval workflows, and rent reminders — reducing manual intervention and improving operational efficiency.

Through the combination of **custom objects, validation rules, approval processes, flows, and Apex scheduling**, the project provides a scalable and intelligent solution for property management.

This implementation showcases practical expertise in Salesforce configuration, automation, and declarative development, aligning perfectly with the goals of a Salesforce Developer project.
With further enhancements, the system can evolve into a full-scale, enterprise-level lease management platform.