LEASE MANAGEMENT

COLLEGE NAME: University College of Engineering, BIT campus Tiruchirappalli

COLLEGE CODE: 8100

TEAM ID : NM2025TMID05996

TEAM SIZE : 4

TEAM MEMBERS

ROLES	NAME	EMAIL
Team Leader	Ramya S	Priyasathiam28@gmail.com
Team Member 1	Bharkavi R	bharkavibharkavi74@gmail.com
Team Member 2	Lakshmipriya M	priyam271004@gmail.com
Team Member 3	Sindhumathi M	sranjani155@gmail.com

1.INTRODUCTION

1.1 Project Overview:

This project involves the development of a CRM-based Lease Management System using Salesforce. It streamlines operations like property listing, lease tracking, rent collection, and tenant management for a property leasing business. The project leverages the Salesforce Lightning Platform to design custom objects, automate processes using Flows, and generate actionable insights through dashboards and reports.

1.2 Purpose:

The purpose is to replace manual lease management processes with an automated, cloud-based Salesforce CRM that enables real-time property updates, lease monitoring, rent tracking, and tenant communication.

The system enhances customer experience, operational efficiency, and business decision-making.

The purpose of this project is to design and implement a Salesforce-based CRM solution specifically for lease management businesses to:

- Centralize tenant and property information.
- Track and manage property leases and rent payments.
- Enable automated reminders for rent due dates and lease renewals.
- Facilitate efficient tenant onboarding and communication.
- Improve client retention through engagement automation.
- Generate detailed reports on occupancy, revenue trends, and staff performance.

PHASE 1 : IDEATION PHASE:

1.1 Problem Statement:

Traditional property leasing businesses often face challenges such as manual record-keeping, inefficient rent tracking, and poor visibility into lease agreements. There is often no unified system to manage tenants, properties, and payments, leading to delays, data errors, and reduced operational efficiency.

Many small to mid-sized property owners and leasing agencies struggle with maintaining tenant records, tracking rent payments, managing lease renewals, and handling maintenance requests. Existing manual systems or generic software solutions fail to meet the specific needs of the leasing industry, resulting in poor tenant experience and management inefficiencies.

1.2 Objective:

✓ **Tenant Data Management:** To maintain a centralized database of tenant profiles, including personal details, lease history, and payment records, enabling personalized communication and service.

✓ **Property and Lease Tracking:** To automate property listing, lease agreement management, and status tracking for better transparency and accuracy.

✓ **Rent Collection and Billing:** To simplify the rent collection process by automating invoice generation, payment reminders, and receipts through Salesforce tools.

✓ **Renewal and Notification Management:** To send automated alerts for upcoming lease expirations, rent dues, and maintenance requests.

✓ **Reporting and Analytics:** To provide dashboards and reports that help analyze occupancy rates, revenue trends, and property performance for data-driven decision-making.

1.3 Ideas collected during brainstorming:

- Use of custom objects for Properties, Tenants, Leases, Payments, and Maintenance Requests.
- Automate rent reminders and lease renewal notifications using Flows.
- Dashboard for monthly revenue and occupancy insights.
- Notifications for overdue rent or expiring leases.
- Implement record-triggered and scheduled flows for automatic updates.
- Introduce validation rules to prevent incomplete or inconsistent lease records.

PHASE 2 : REQUIREMENT ANALYSIS:

2.1. Functional and Non-Functional Requirements:

2.1.1 Functional Requirements

- **Tenant Data Management:**
  - Maintain a centralized database of tenant details (name, contact, lease history, payment records).
  - Segment tenants based on property type, lease duration, and payment behavior.
  - Enable personalized communication and follow-ups (e.g., rent reminders, renewal offers).
- **Property and Lease Management:**
  - Store detailed information about each property (location, size, type, rent amount, and status).
  - Track active, upcoming, and expired leases with start and end dates.
  - Automate lease renewal workflows and manage digital copies of agreements.
- **Rent Collection and Payment Tracking:**
  - Automate invoice generation for monthly rent and maintenance charges.
  - Track payments, generate receipts, and update tenant records in real time.
  - Provide alerts for overdue payments and generate rent collection reports.
- **Maintenance Request Management:**
  - Allow tenants to raise maintenance or service requests through a self-service portal.
  - Assign tasks automatically to property managers or technicians using Flows.
  - Track request status, resolution time, and staff performance.
- **User Management and Access Control:**

- Define roles for property managers, finance officers, maintenance staff, and administrators.
- Restrict data access and permissions based on user roles and responsibilities.
- Ensure secure handling of tenant and financial data through role-based acc

2.1.2 Non-Functional Requirements:

 Performance: The system should handle multiple concurrent users, such as property managers, tenants, and administrators, efficiently with minimal latency.

 Scalability: The system must support the addition of new properties, users, and lease records without degrading performance.

 Security: Tenant and payment information must be encrypted and protected in compliance with Salesforce's data security and privacy standards.

 Usability: The interface should be user-friendly and intuitive, enabling property staff and tenants to operate the system with minimal training.

 Reliability: The system should maintain 99.9% uptime, ensuring data consistency and integrity during transactions and updates.

 Maintainability: System configurations, workflows, and automation processes should be easily modifiable using Salesforce's declarative tools (Flows, Process Builder).

2.2. Object Relationships, Automations, and User Roles

Object Relationships

 Tenant → Lease → Payment → Renewal

 Property → Lease → Maintenance Request → Payment

 User (Property Manager / Finance Officer / Admin) → Assigned Properties, Tenants, and Leases

Automations

 Lease Assignment Rule: Automatically assign new lease records to property managers based on property location or type.

 Workflow Rules: Trigger automated rent reminders and renewal notifications for tenants.

 Payment Update: Automatically record payment status and update tenant ledger upon receipt.

 Approval Process: Approval workflow for high-value leases, property additions, or rent adjustments.

 Maintenance Request Automation: Automatically route maintenance requests to assigned staff and track resolution time.

User Roles

 Admin: Full control over all modules, settings, and user permissions.

 Property Manager: Manage lease agreements, tenant records, and maintenance requests.

 Finance Officer: Handle rent invoicing, payment tracking, and financial reports.

 Maintenance Staff: View and update assigned maintenance or service tasks.

2.3. Documentation and Traceability

 All requirements are documented in a Requirement Traceability Matrix (RTM), linking each functional requirement to corresponding design components, workflows, and test cases.

 Salesforce objects, fields, and flows are mapped directly to business requirements for traceability across all project phases.

 Any changes in scope or configuration are version-controlled and reviewed during sprint retrospectives to ensure alignment with business goals.

2.4. Dependencies and System Constraints

Dependencies:

 Integration with external payment gateways (e.g., Stripe, Razorpay, PayPal) for rent transactions.

 Access to property listing and tenant data from external sources, if applicable.

 Stable internet connectivity for Salesforce access and data synchronization.

 Availability of appropriate Salesforce editions and licenses (Sales Cloud, Service Cloud).

System Constraints:

 Limited customization options within certain Salesforce managed packages.

 Data storage limits based on Salesforce edition and allocated capacity.

 System performance depends on Salesforce API call limits and governor limits.

 Some automation features may be restricted by Salesforce's Flow execution limits or trigger constraints.

PHASE 3: PROJECT DESIGN

3.1 Problem Solution Fit :

Problem: Manual lease management processes and lack of centralized property, tenant, and payment tracking lead to inefficiencies, delayed operations, and poor visibility.

Solution: A Salesforce-based CRM system automating all major leasing workflows—covering property management, tenant tracking, rent collection, and maintenance handling—within a unified cloud platform.

3.2 Proposed Solution:

Property__c, Lease__c, Tenant__c, Payment__c, Maintenance_Request__c

● A Lightning App with custom navigation tabs for Properties, Leases, Tenants, Payments, and Maintenance.
● Automated Flows for rent reminders, lease renewals, and payment tracking.
● Dashboards and Reports for occupancy, revenue trends, and overdue payments.

3.3 Solution Architecture:

Objects and Relationships:

● Tenant__c ↔ Lease__c ↔ Payment__c ↔ Property__c ↔ Maintenance_Request__c

● Lookup and Master-Detail relationships used to connect records across modules.

● Formula fields to auto-calculate values such as total rent, due balance, and lease duration.

● Validation rules to ensure correct data entry (e.g., lease dates, rent amount, payment status).

● Record Types to distinguish between residential, commercial, and short-term leases.

3.4 Key Design Components:

• **ER Design:** Defines key entities—Tenant, Property, Lease, Payment, and Maintenance—with clear links for smooth data flow from onboarding to rent collection.

• **Modular Architecture:** Independent modules integrate seamlessly within Salesforce for scalability and flexibility.

• **Workflow & Automation:** Flows manage renewals, rent reminders, overdue alerts, and maintenance updates automatically.

• **Security & Access:** Role-based access using Profiles and Permission Sets ensures data protection and controlled visibility.

3.5 User Experience (UX) Considerations

● **Intuitive Navigation:** Clear Lightning App tabs for quick access to all leasing modules.
● **Responsive Design:** Optimized for desktop, tablet, and mobile devices.
● **Visual Dashboards:** Graphical reports displaying rent collection, occupancy rate, and lease status.
● **Consistency:** Unified layout and color scheme across all pages for a cohesive user experience.

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

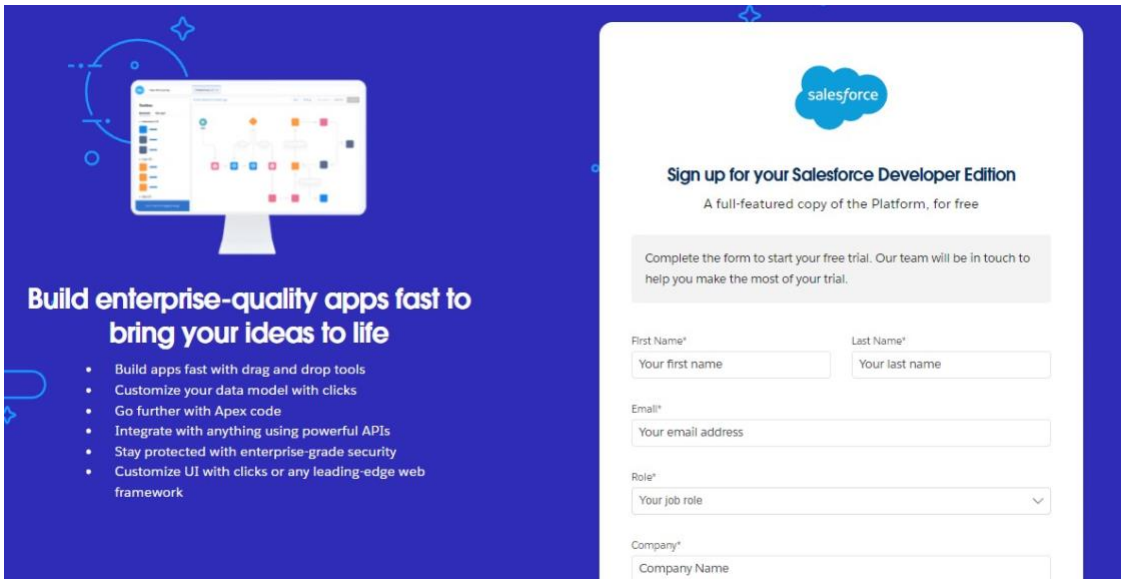
Milestone 1: Salesforce Developer Account Creation:

Activity 1: Creating Developer Account

Creating a developer org in salesforce.

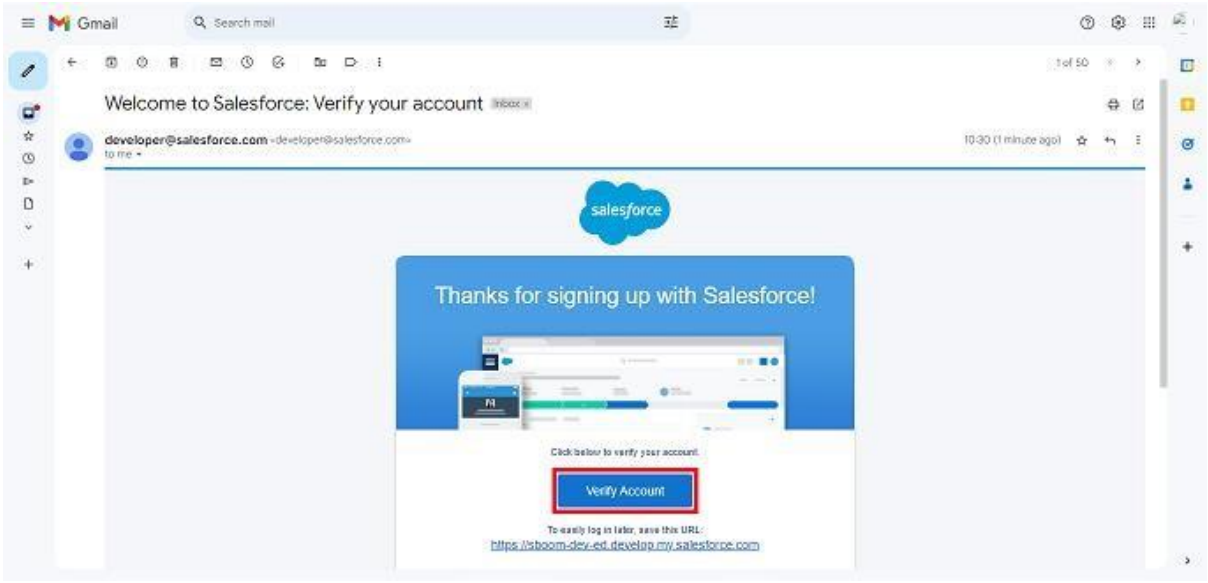
1. Go to https://developer.salesforce.com/signup 2.
On the sign up form, enter the following details :

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.



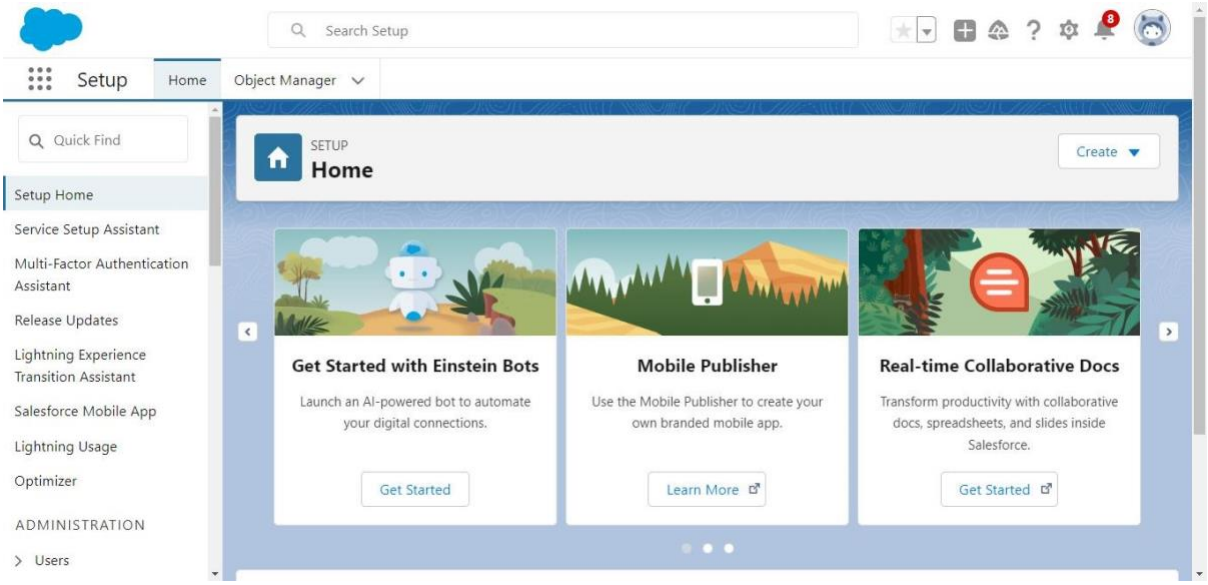
Activity 2: Account Activation:

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:

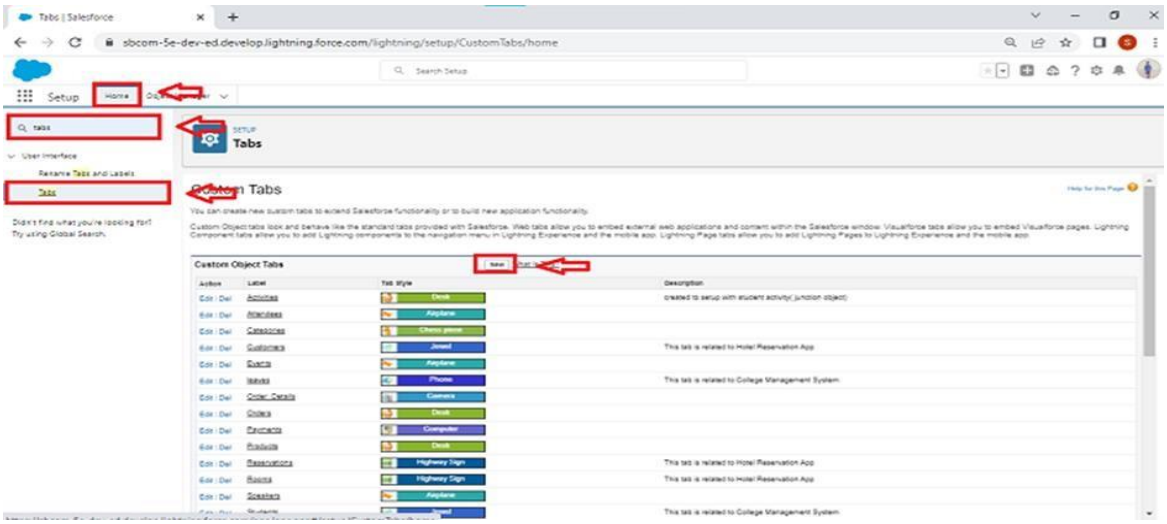
1. From the setup page >> Click on Object Manager >> Click on Create >> Click on Custom Object.
    1. Enter the label name>> lease
    2. Plural label name>> lease
    3. Enter Record Name Label and Format
        - Record Name >> lease Name
        - Data Type >> Text
2. Click on Allow reports and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

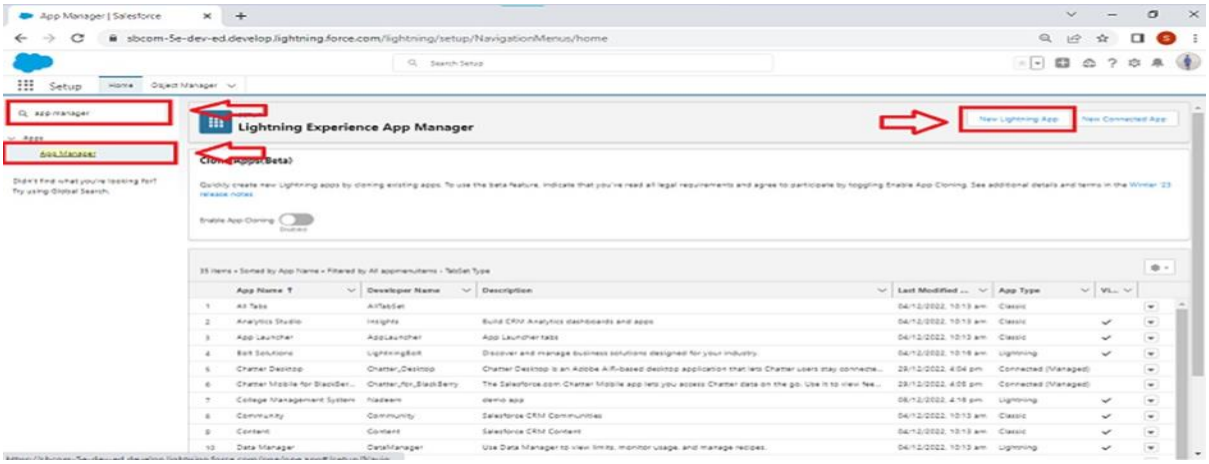
Phase 3: UI/UX Development

Milestone 4: Lightning App

Activity 1: Create a Lightning App

To create a lightning app page:

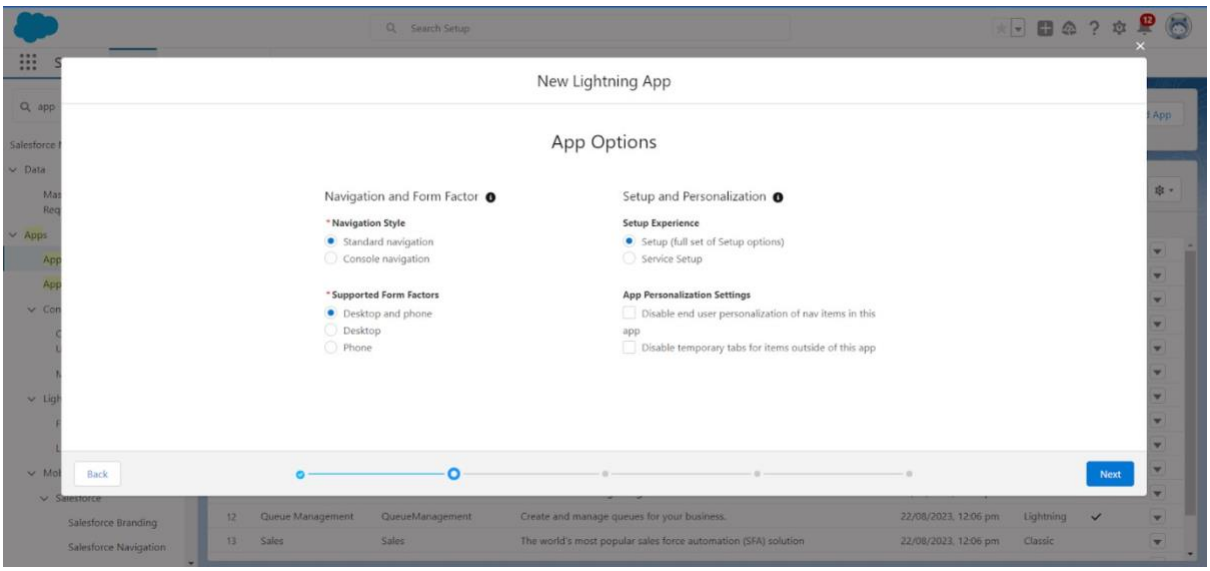
1. Go to setup page >> search “app manager” in quick find >> select “app manager” >> click on New lightning App.



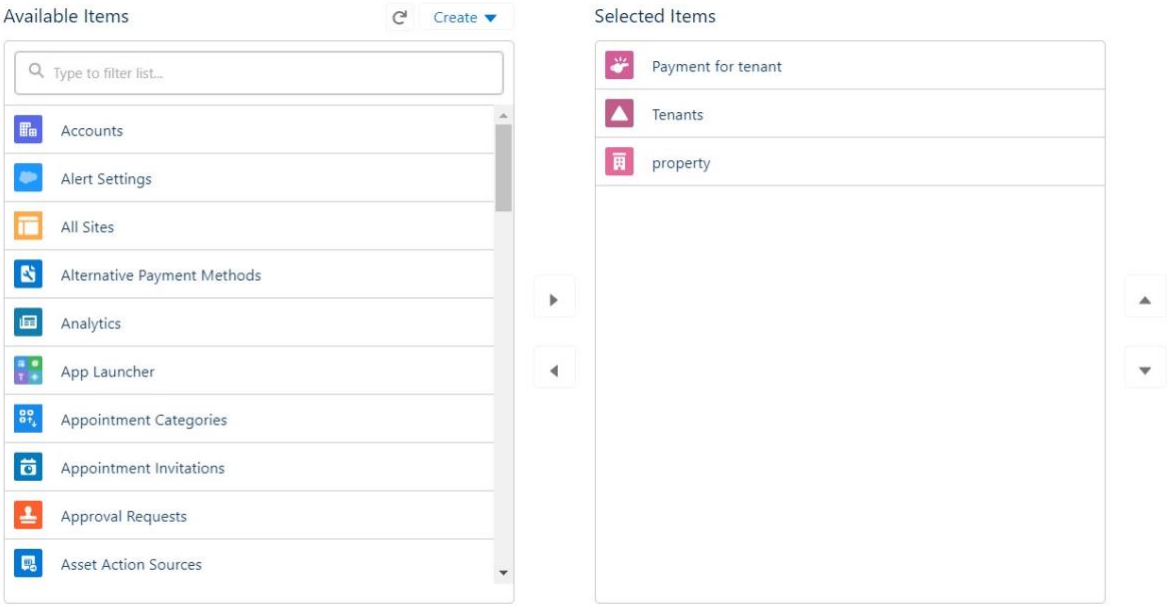
2. Fill the app name in app details and branding as follow

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.

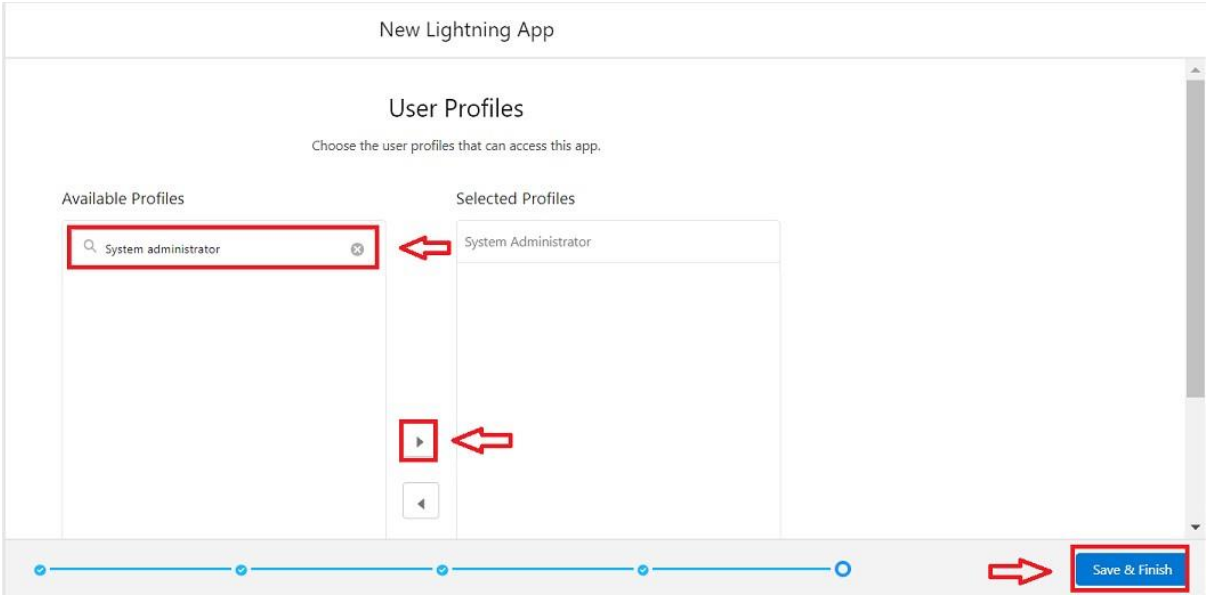


5. To Add Navigation Items:



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:



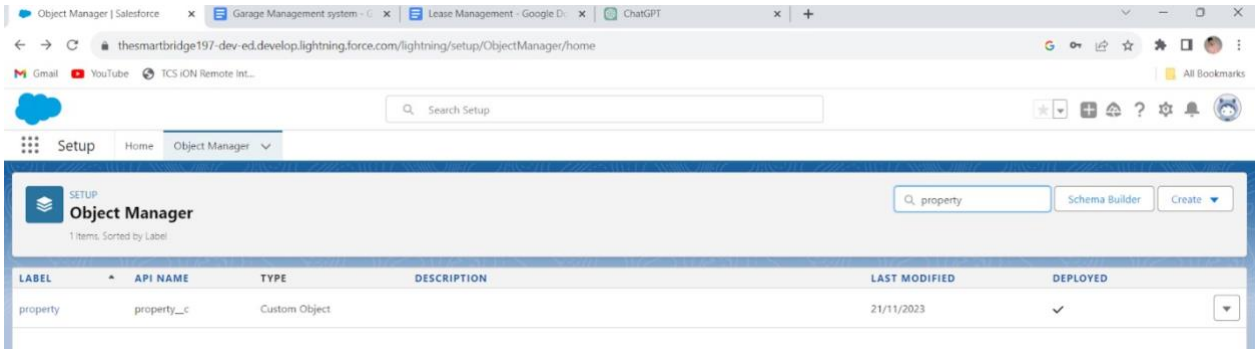
Search profiles (System administrator) in the search bar >>click on the arrow button >> save & finish.

Milestone 5: Fields

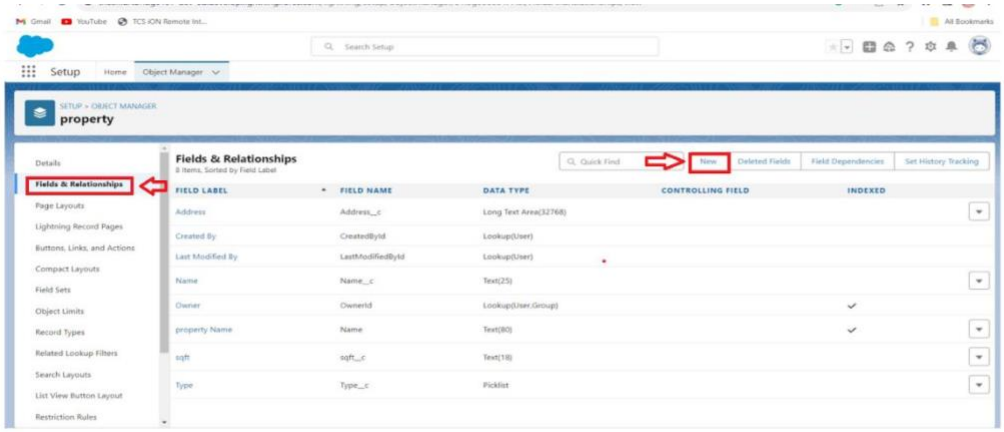
Activity 1: Creation of fields for the property object

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.



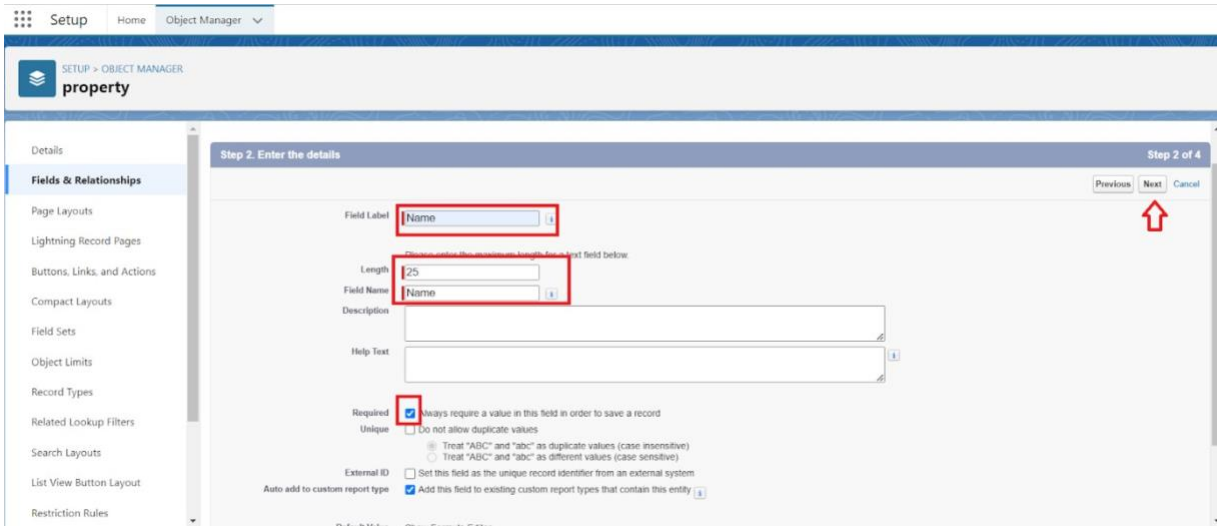
2. Now click on “Fields & Relationships” >> New



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.

To create another fields in an 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 “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.

To create another fields in an object:

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.

To create another fields in an 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 : 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.

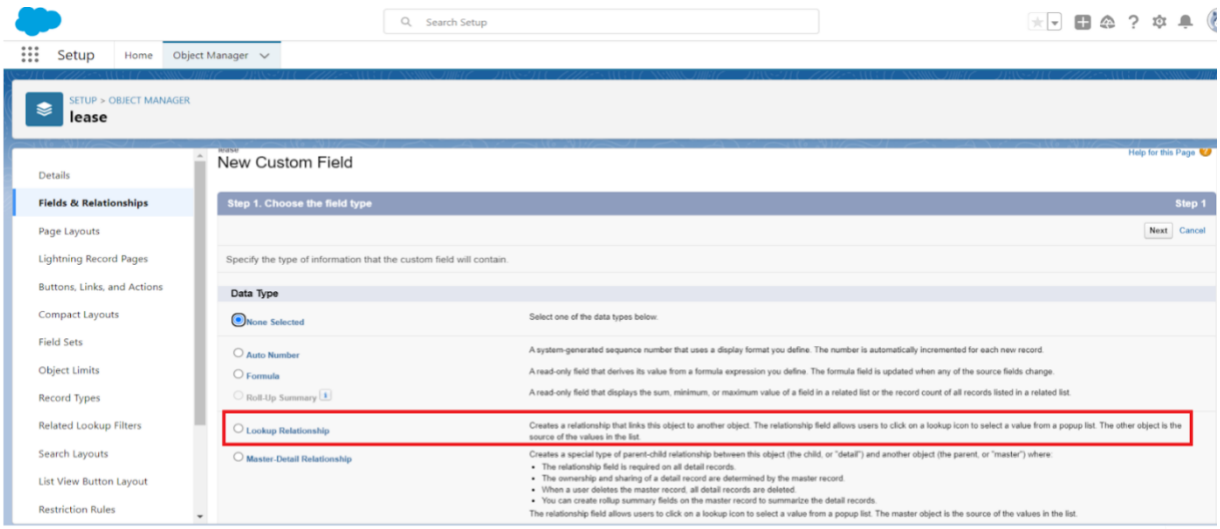
To create another fields in an object:
1.Go to setup >> click on Object Manager >> type object name(Payment for tenant) in search bar >> click on the object.
2. Now click on “Fields & Relationships” >> New
3. Select Data type as a “Number” and Click on Next
4. Fill the Above as following:
   - Field 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.

<u>Creation of Lookup Field on Payment Object :</u>

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.

<u>Creation of Lookup Field on Payment for tenant Object :</u>

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

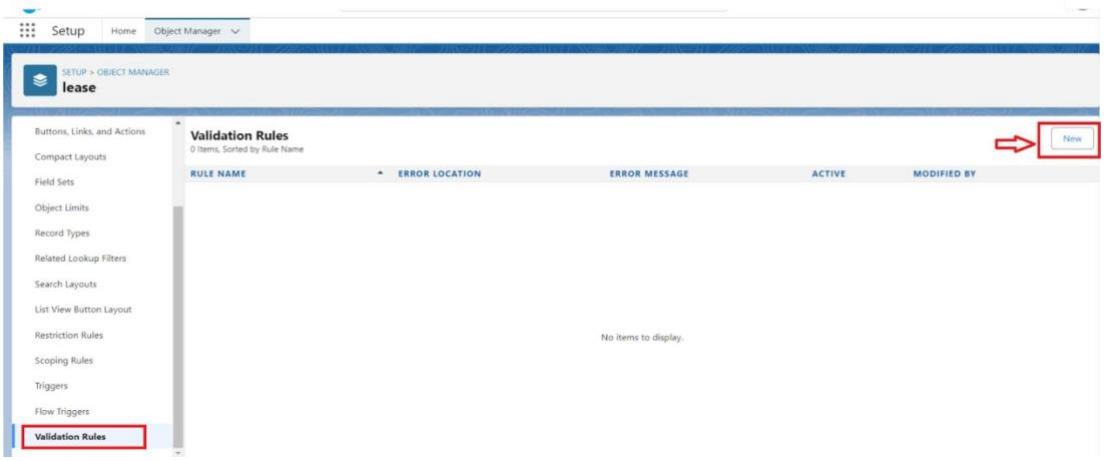
16. Now click on “Fields & Relationships” >> New
17. Select masterdetail relationship
18. Select the related object “ property” and click next.
19. Field Name : property
20. Field label : Auto generated
21. Next >> Next >> Save.

Phase 4: Testing & Security

Milestone 6: Validation Rules

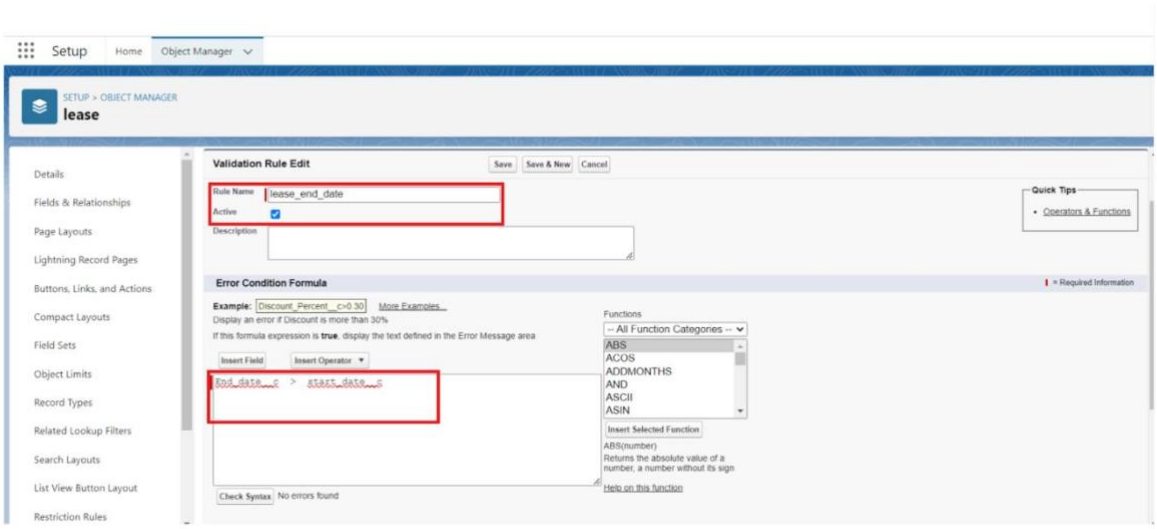
Activity 1: To create a validation rule to an Lease Object

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.

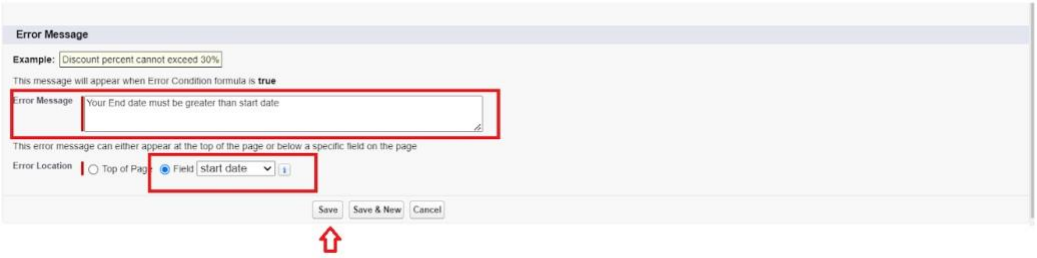


3. Enter the Rule name as “ lease_end_date”.
4. Insert the Error Condition Formula as :

```
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

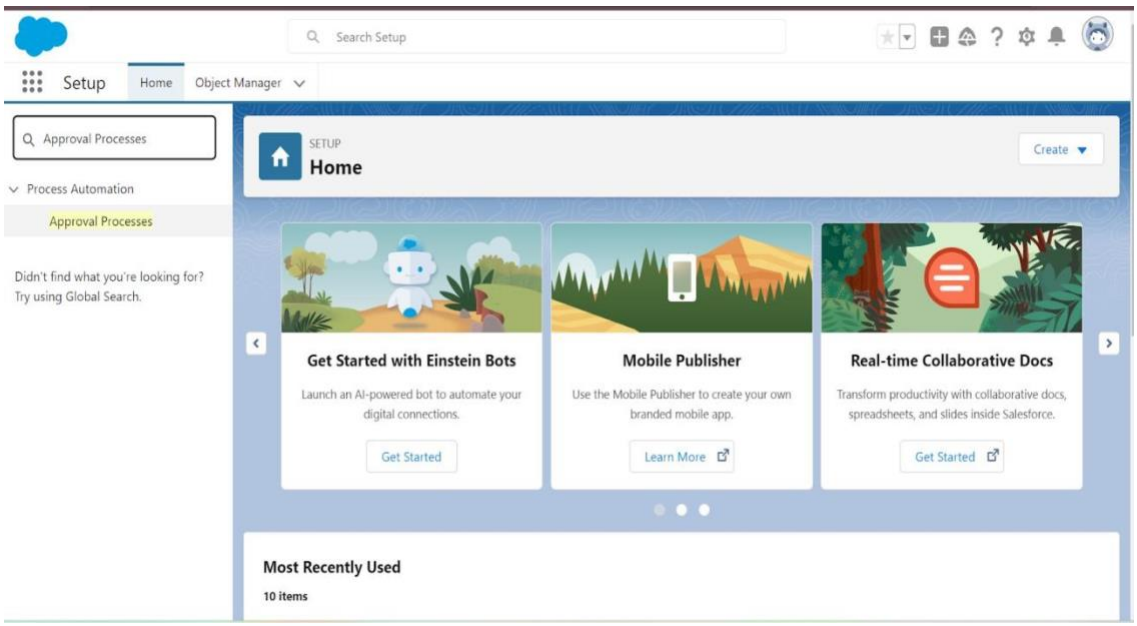
To create fields in an object:

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.

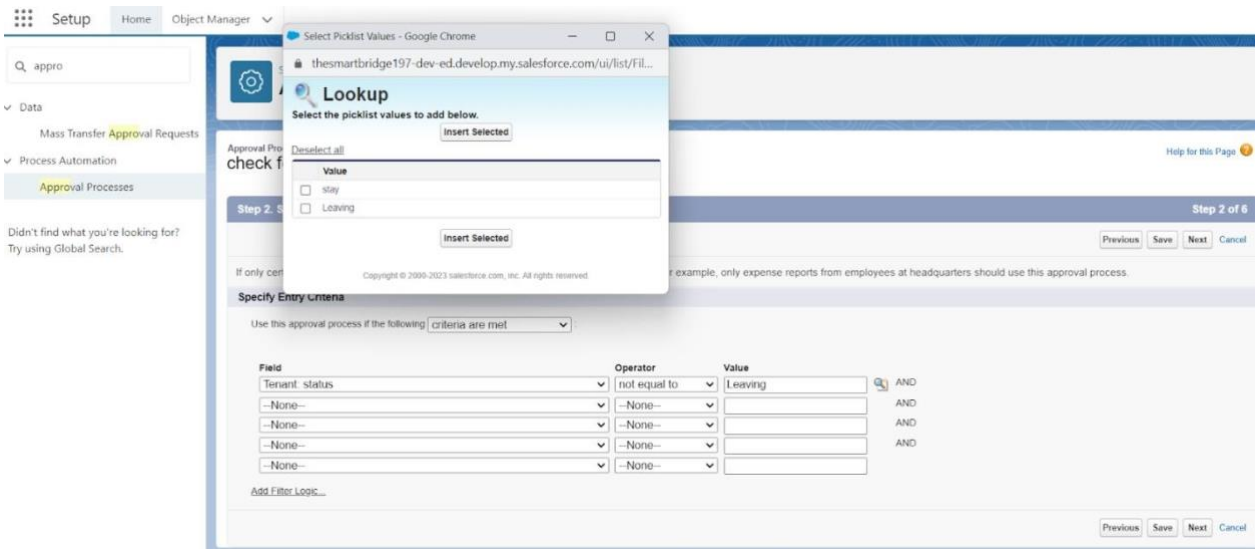
4. Process Name “check for vacant” >> Click Next



.

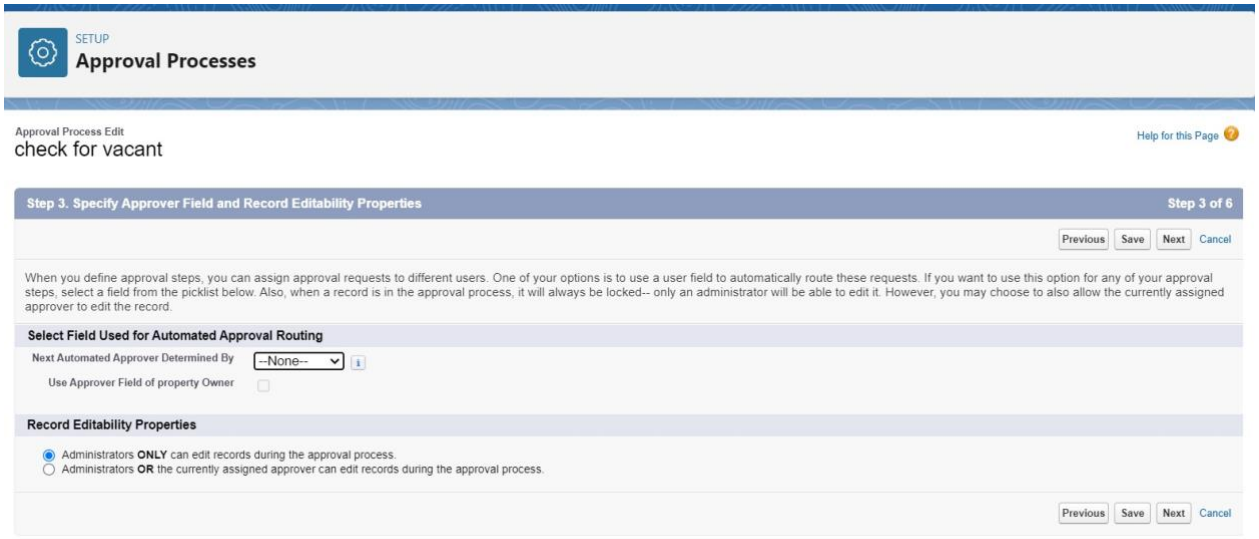
5. Field “Tenant:status” >> Operator : Not equals , Value >> Click on the lookup filter icon and select “Leaving”.

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.



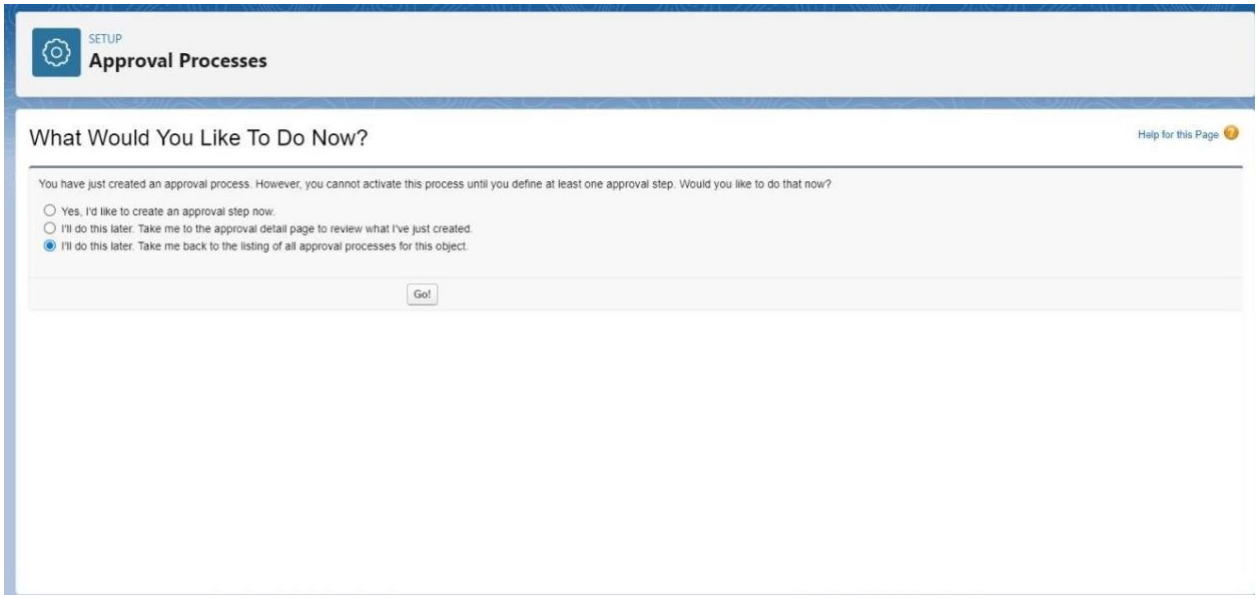
9. Click on next leave the email template click on next

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”.

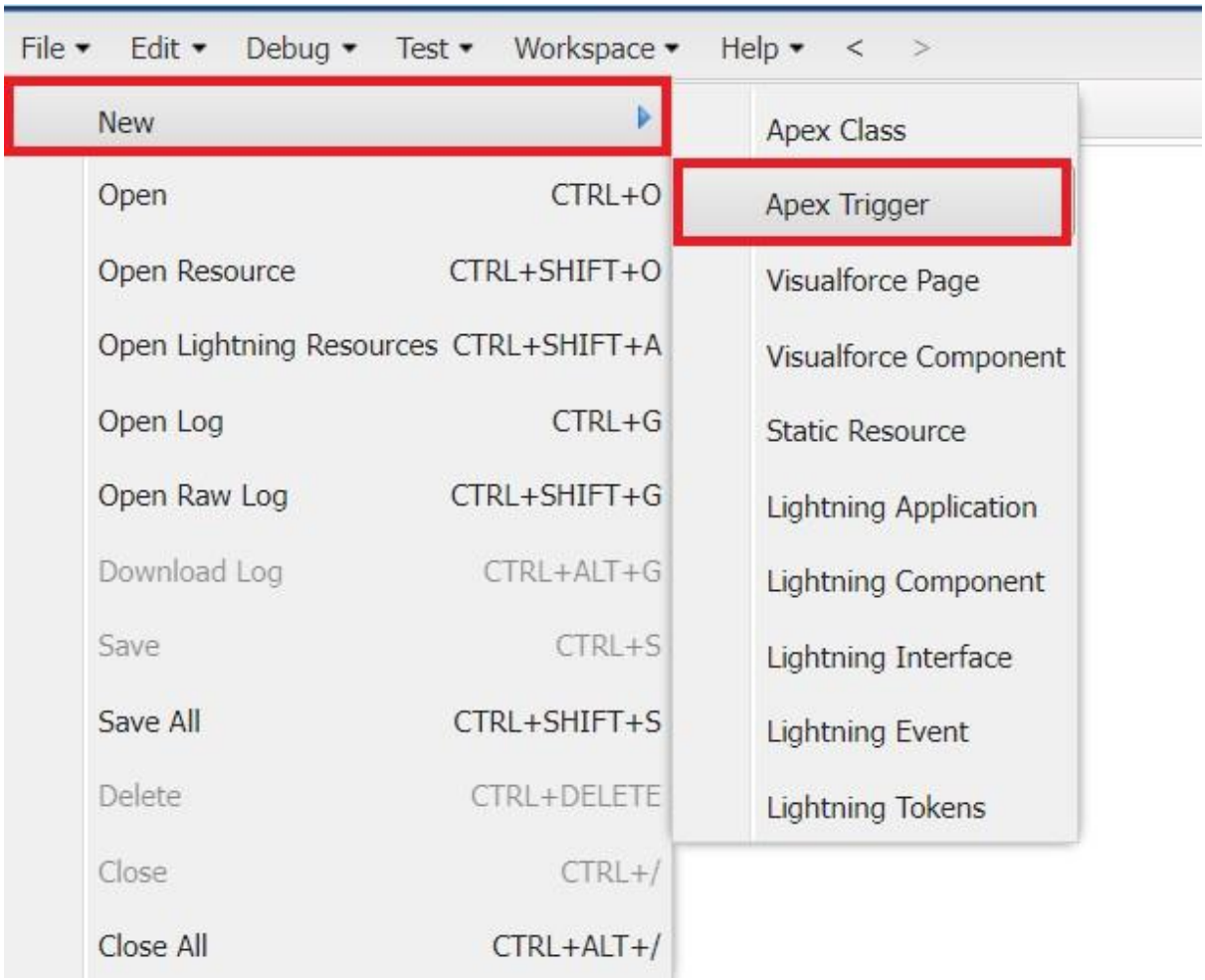
3.unique name : auto populated
4.Email template : leave rejected
5. Recipient type : Email field
6. Available Recipients : Email field : Email
7. From Email address : Current user’s email
8. Click save

Milestone 8:Apex triggers

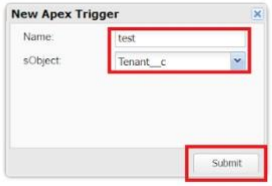
Activity 1: Create an Apex trigger

1. To create a new Apex Class follow the below steps:

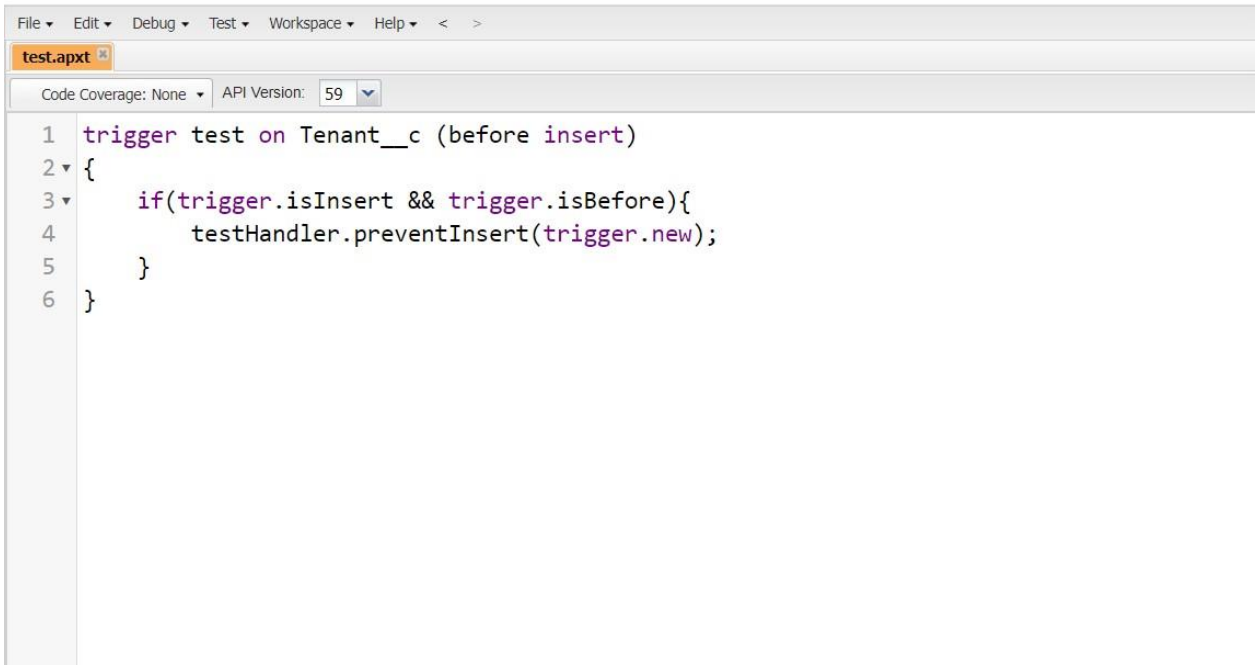
   Click on the file >> New ? Apex Class.



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



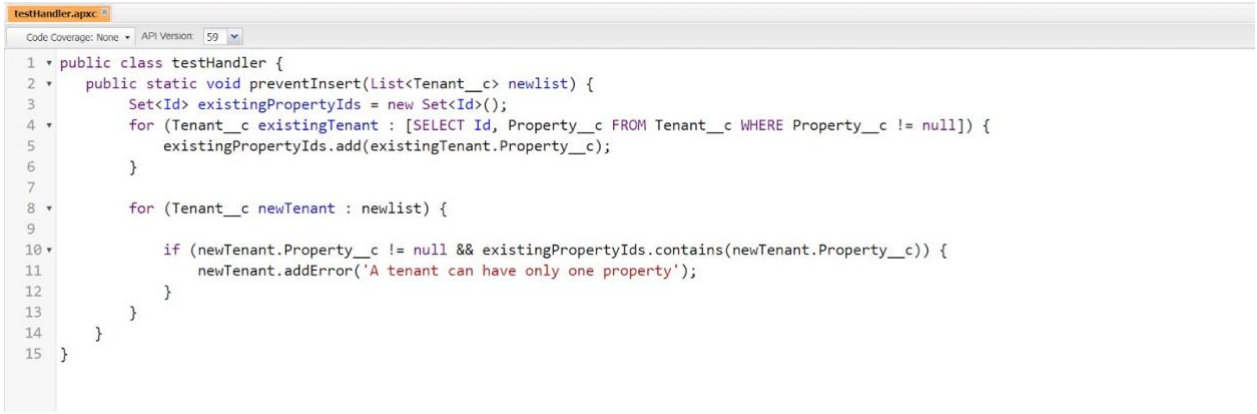
Trigger Code:

```
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.



Apex logic:

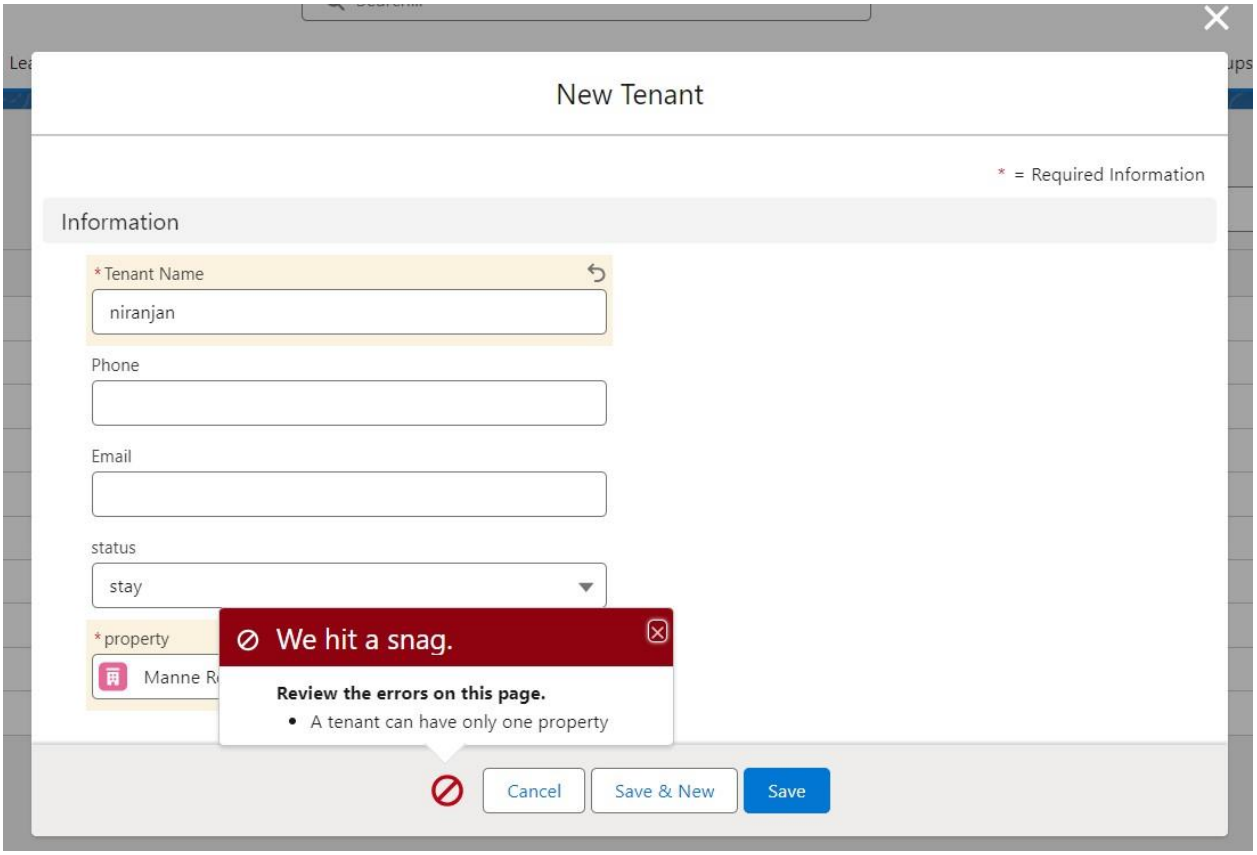
```
public class testHandler {
  public static void preventInsert(List<Tenant__c> newlist) {
Set<Id> existingPropertyIds = new Set<Id>();
      for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c
WHERE Property__c != null]) {
          existingPropertyIds.add(existingTenant.Property__c);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

      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'

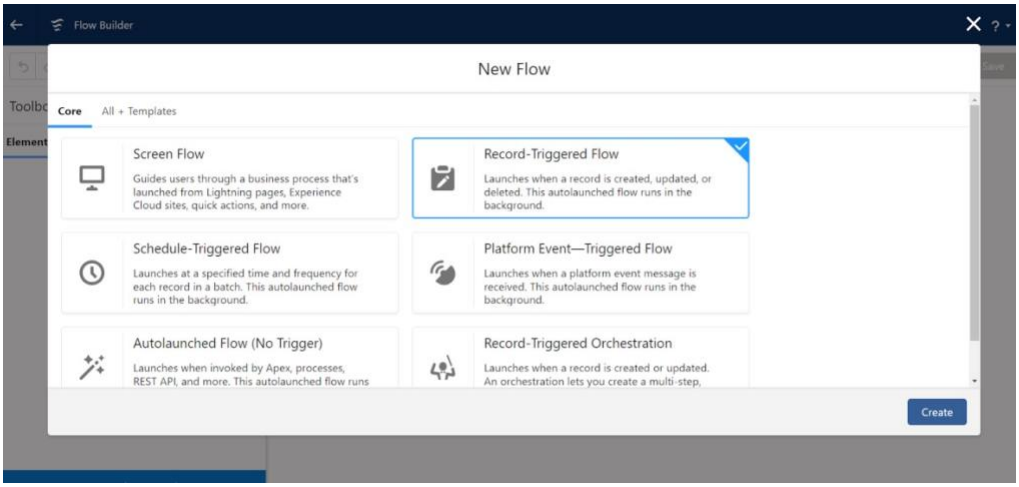
Try to create new tenant with the existing property then it shows the error



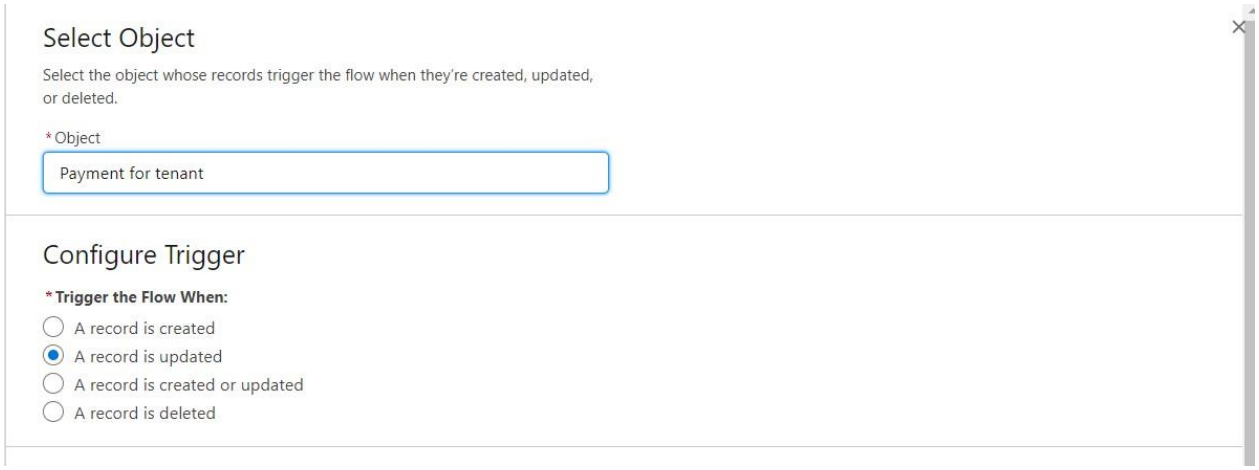
Milestone 10: Flows

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.



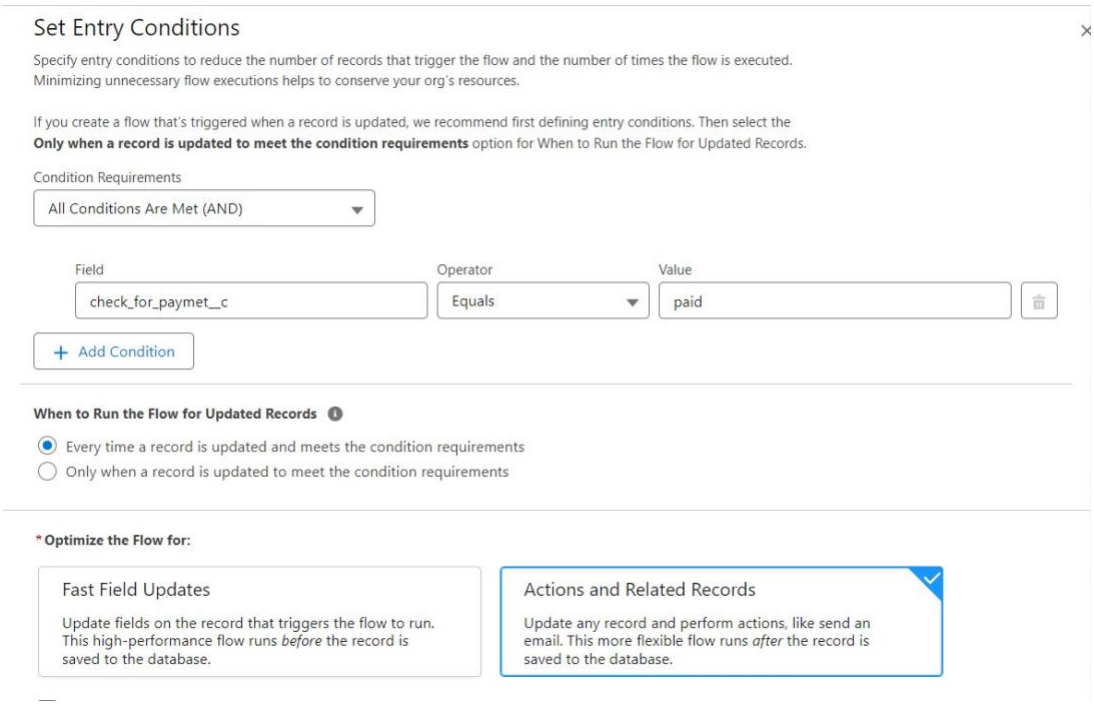
3. Under Object select ”Payment for tenant”. Click on A record is updated.



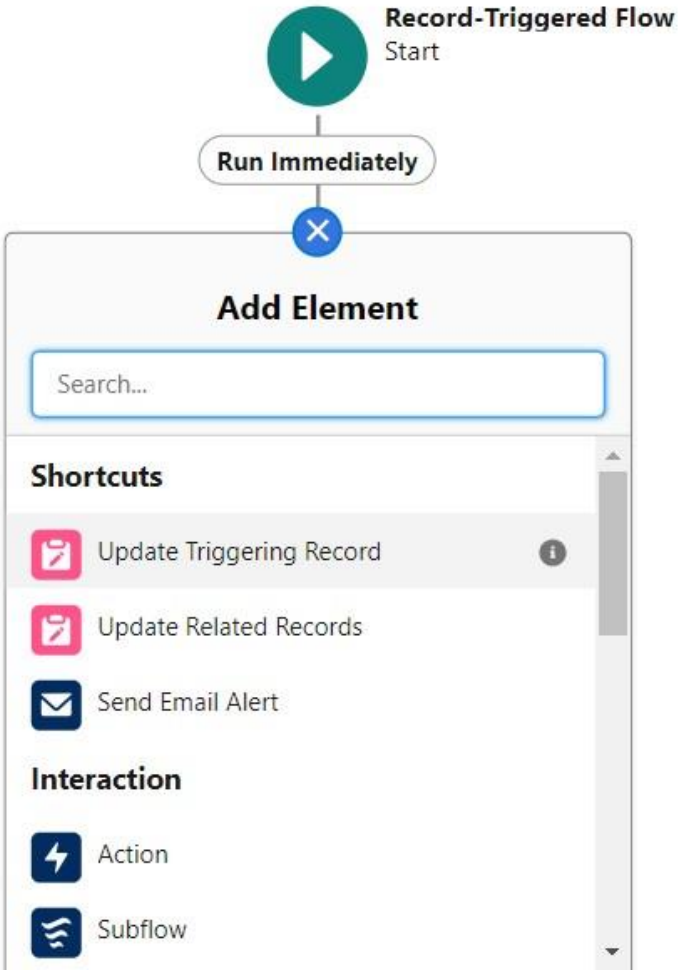
4. Set Entry Conditions
   Under Condition Requirements>>All Conditions are met

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



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)

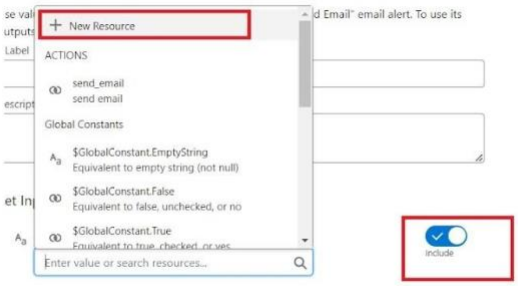
8. Label : send email

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.

```
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 arrangem
            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});
        }
    }
}
```

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';
```

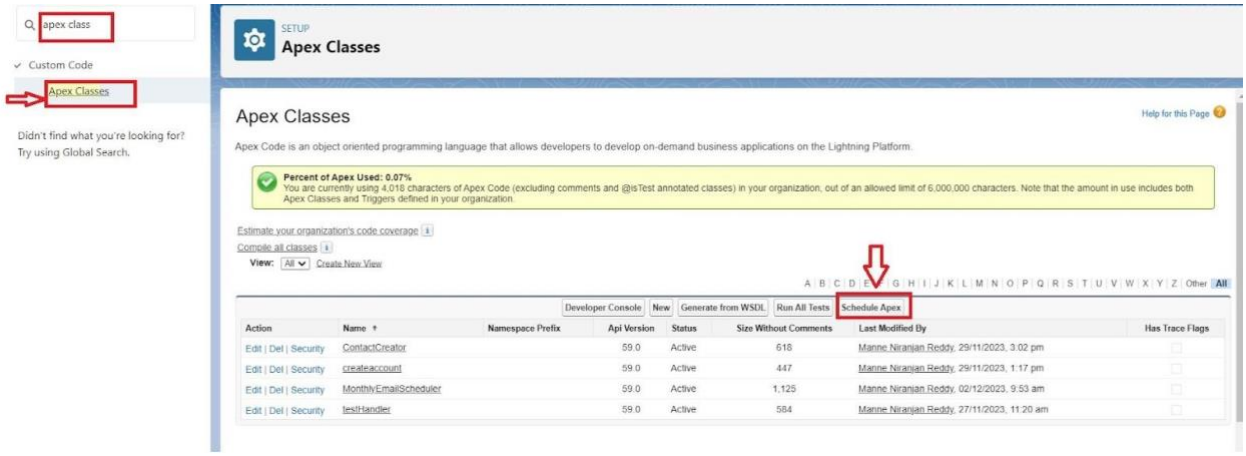
```
        Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
        email.setToAddresses(new String[]{recipientEmail});
        email.setSubject(emailSubject);
email.setPlainTextBody(emailContent);

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

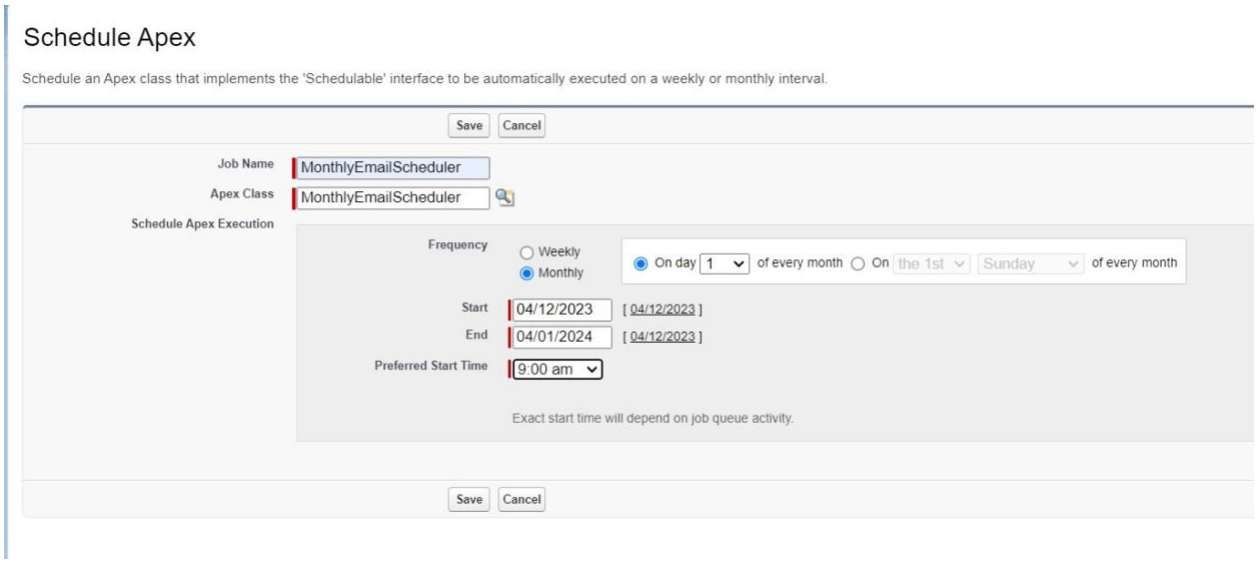
3.Save the code.

Activity 2: Schedule Apex class

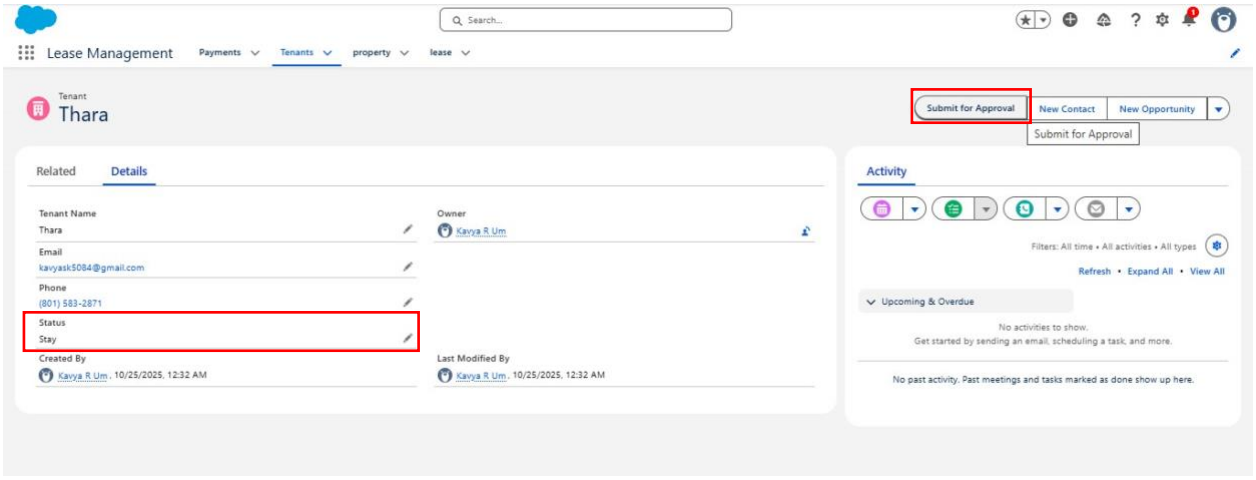
1. Enter Apex class in quick find box
2. Select schedule Apex



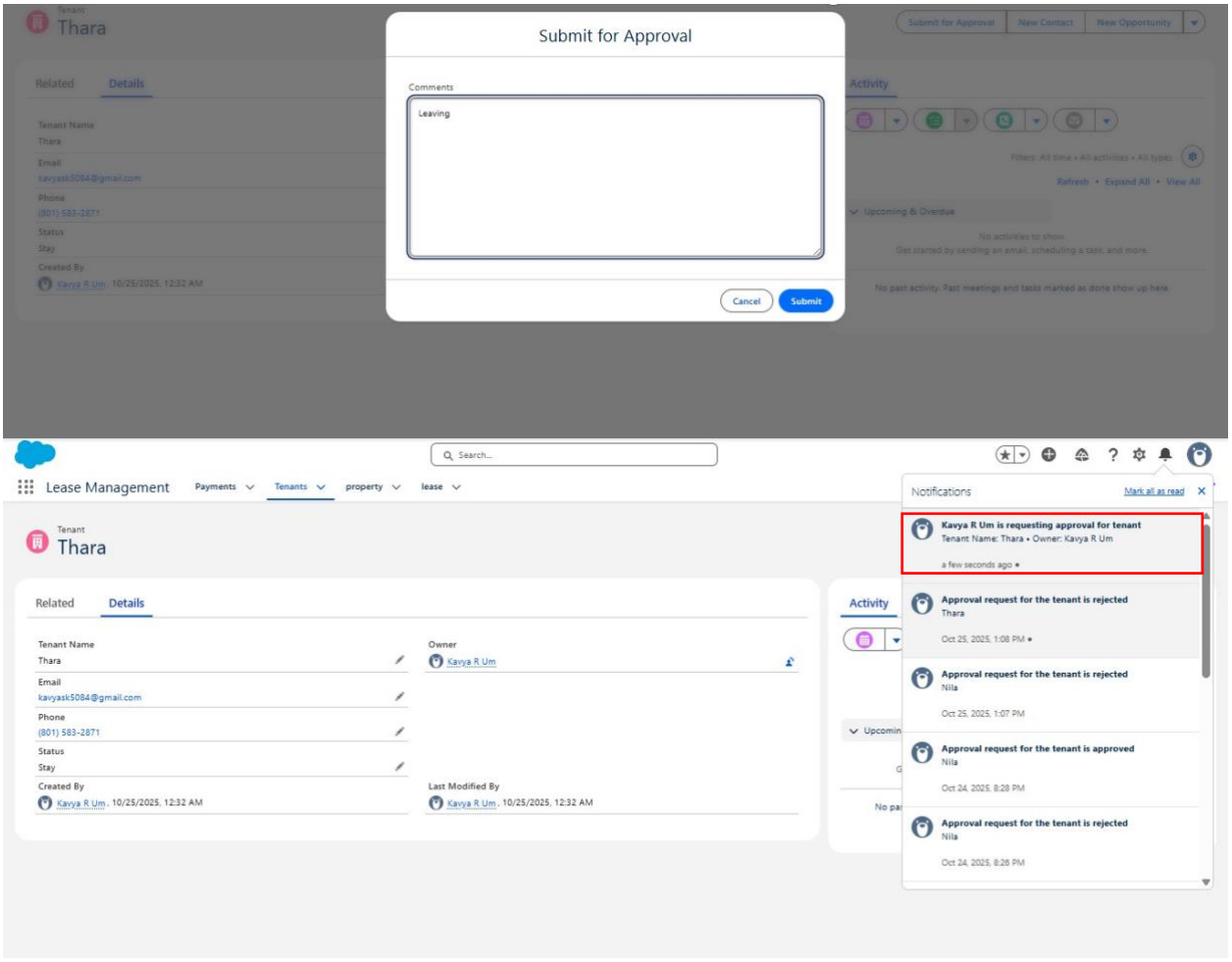
3. Enter job Name : MonthlyEmailScheduler
4. Apex class : MonthlyEmailScheduler
5. Frequency : Monthly===>select on day 1
6. Start date : 04/12/2023
7. End date : 04/01/2024
8. Preferred start time : 09:00 am
9. Save



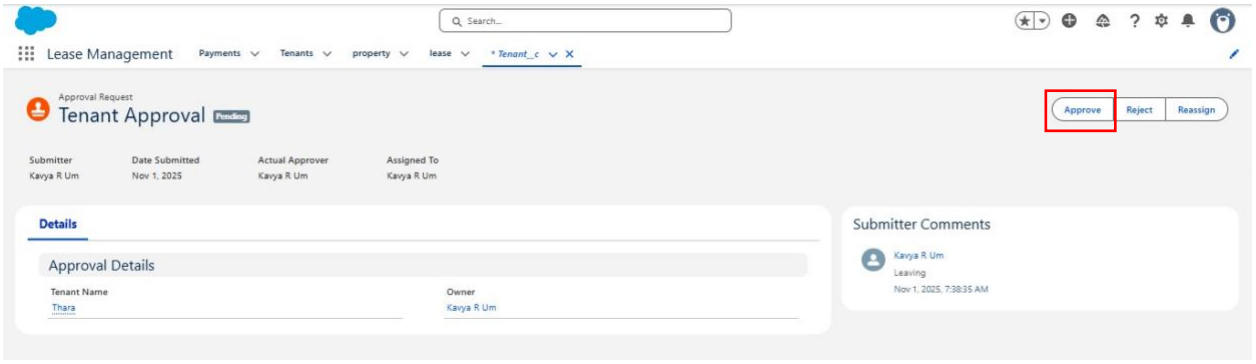
Testing the approval process :



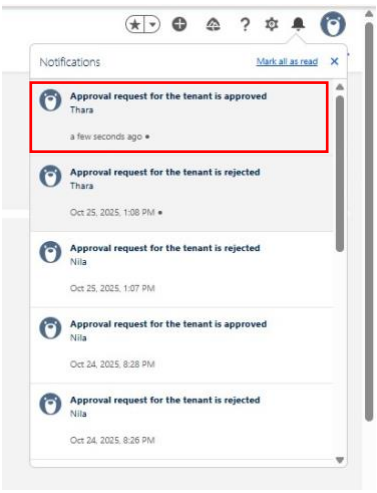
Enter any comment am\nd click on submit



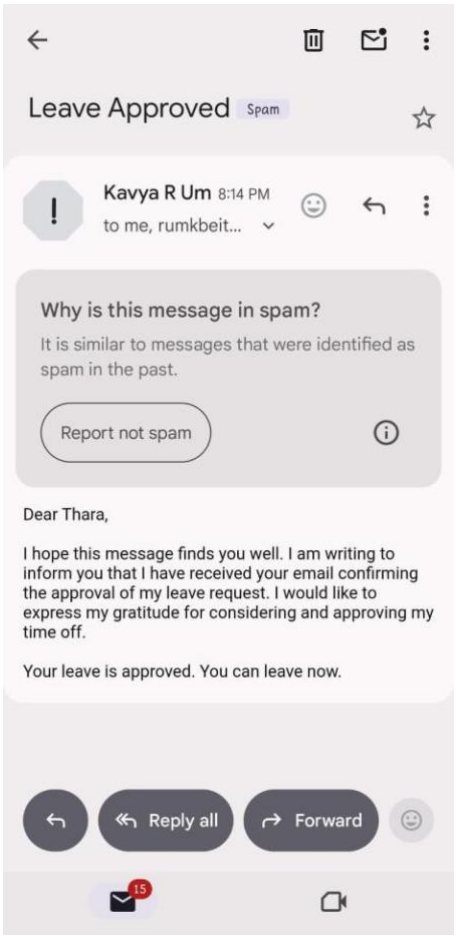
Click on that notification



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**.

✓ Once the CRM-based Lease Management application was fully tested and validated, it was deployed in the **Salesforce production environment** for real-time business use.

✓ All essential modules — including **Property, Tenant, Lease, Payment, and Maintenance Request** — were activated and integrated for seamless operation.

✓ **Dashboards and reports** were shared with management to monitor key metrics such as occupancy rates, rent collection, pending maintenance, and lease renewals.

✓ **Maintenance** involves performing regular system updates, monitoring automation performance, reviewing user feedback, and optimizing workflows for better efficiency.

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.

Testing Summary:

- Testing and debugging played a crucial role in ensuring the reliability, accuracy, and performance of the CRM Application for Lease Management. Multiple testing phases were conducted to verify that all modules — including property management, tenant tracking, lease processing, payment handling, and maintenance requests — functioned correctly and met the project requirements.
- Functional Testing was carried out to validate each feature against defined use cases. Modules such as lease creation, rent collection, and payment processing were tested using different input scenarios to ensure that workflows performed as expected.
- Integration Testing ensured seamless data flow between interconnected objects such as Tenant, Lease, Property, and Payment, confirming that information remained consistent across the system.

- User Acceptance Testing (UAT) was performed with sample users such as property managers, finance officers, and maintenance staff to evaluate the system's usability and performance under real operational conditions. Their feedback helped refine page layouts, validation rules, and automation flows.
- Performance Testing validated the system's ability to handle multiple concurrent users and ensured smooth performance with minimal latency.

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.