

# **LEASE MANAGEMENT**

**College Name:** Hindusthan College Of Science And Commerce

**College Code:** CA

**TEAM ID: NM2025TMID28247**

**TEAM MEMBERS:**

**Team LeaderName:** JANANI P

**Email:** [aswinjanani2011@gmail.com](mailto:aswinjanani2011@gmail.com)

**Team Member1:** SWETHA P

**Email:** [2006swethap@gmail.com](mailto:2006swethap@gmail.com)

**Team Member:** MEENA P

**Email:** [meenapalani5040@gmail.com](mailto:meenapalani5040@gmail.com)

**Team Member:** KUZHANDHAIMERY M

**Email:** [kuzhandhaimery@gmail.com](mailto:kuzhandhaimery@gmail.com)

# 1.INTRODUCTION

## 1.1 Project Overview

This project is focused on the development and implementation of a Lease Management System using Salesforce, designed to address the challenges of manual property lease tracking, tenant management, payment processing, and communication within the organization. The goal is to deliver a comprehensive solution that streamlines and automates various aspects of lease management, from contract creation to payment tracking, approval workflows, and reporting.



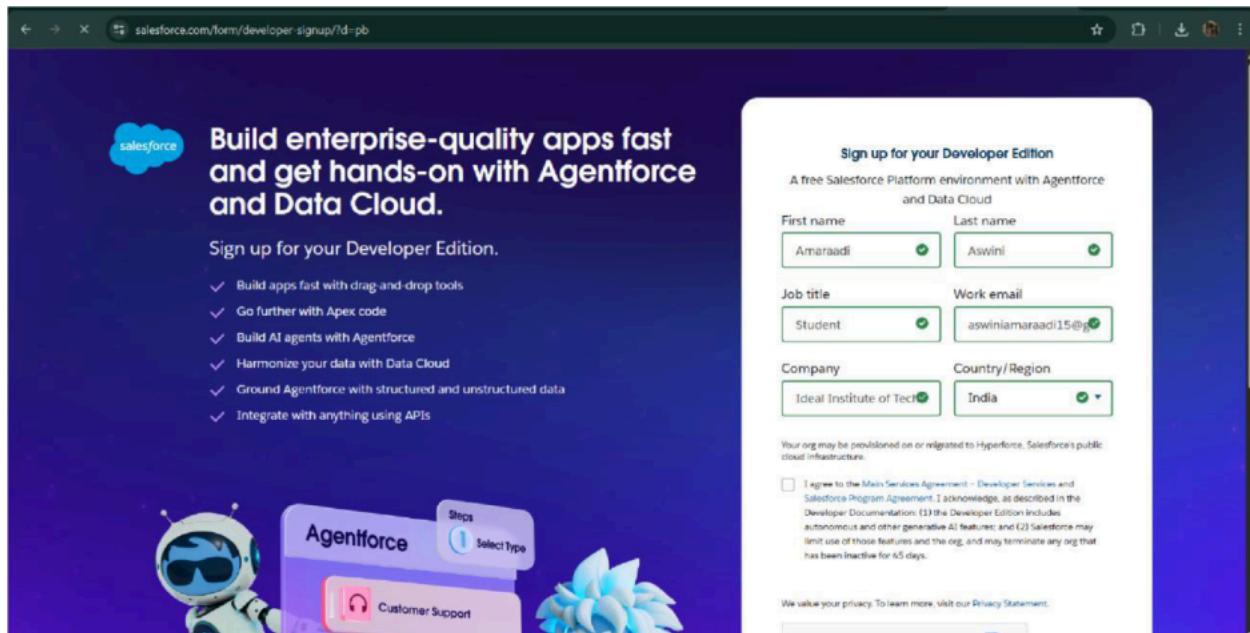
## 1.2 Purpose

The main objective of the project is to enable organizations to efficiently manage properties, tenants, and lease-related activities. It reduces manual intervention, improves accuracy, and ensures better compliance and communication.

# DEVELOPMENT PHASE

### Creating Developer Account:

By using this URL - <https://www.salesforce.com/form/developer-signup/?d=pb>



- Created objects: Property, Tenant, Lease, Payment

The screenshot shows the Salesforce Object Manager interface for the 'Tenant' object. The left sidebar lists various configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The main 'Details' tab displays the following information:

| Details   |                                     |
|---|-------------------------------------|
| Description   |                                     |
| API Name  | Tenant_c                            |
| Custom  | <input checked="" type="checkbox"/> |
| Singular Label  | Tenant                              |
| Plural Label  | Tenants                             |
| Enable Reports  |                                     |
| <input checked="" type="checkbox"/> Track Activities    |                                     |
| <input checked="" type="checkbox"/> Track Field History |                                     |
| <input checked="" type="checkbox"/> Deployment Status   |                                     |
| Deployed  |                                     |
| Help Settings   |                                     |
| Standard salesforce.com Help Window                     |                                     |

The screenshot shows the Salesforce Object Manager interface for the 'Lease' object. The left sidebar lists the same configuration options as the Tenant screen. The main 'Details' tab displays the following information:

| Details   |                                     |
|---|-------------------------------------|
| Description   |                                     |
| API Name  | lease_c                             |
| Custom  | <input checked="" type="checkbox"/> |
| Singular Label  | lease                               |
| Plural Label  | lease                               |
| Enable Reports  |                                     |
| <input checked="" type="checkbox"/> Track Activities    |                                     |
| <input checked="" type="checkbox"/> Track Field History |                                     |
| <input checked="" type="checkbox"/> Deployment Status   |                                     |
| Deployed  |                                     |
| Help Settings   |                                     |
| Standard salesforce.com Help Window                     |                                     |

The screenshot shows the Salesforce Object Manager interface for the 'Payment\_for\_tenantat' object. The left sidebar lists the same configuration options. The main 'Details' tab displays the following information:

| Details   |                                     |
|---|-------------------------------------|
| Description   |                                     |
| API Name  | Payment_for_tenantat_c              |
| Custom  | <input checked="" type="checkbox"/> |
| Singular Label  | Payment for tenantat                |
| Plural Label  | Payment                             |
| Enable Reports  |                                     |
| <input checked="" type="checkbox"/> Track Activities    |                                     |
| <input checked="" type="checkbox"/> Track Field History |                                     |
| <input checked="" type="checkbox"/> Deployment Status   |                                     |
| Deployed  |                                     |
| Help Settings   |                                     |
| Standard salesforce.com Help Window                     |                                     |

- Configured fields and relationships

SETUP > OBJECT MANAGER  
**property**

**Fields & Relationships**  
9 Items, Sorted by Field Label

| FIELD LABEL      | FIELD NAME       | DATA TYPE             | CONTROLLING FIELD | INDEXED |
|------------------|------------------|-----------------------|-------------------|---------|
| Address          | Address__c       | Long Text Area(32768) |                   |         |
| Created By       | CreatedById      | Lookup(User)          |                   |         |
| Last Modified By | LastModifiedById | Lookup(User)          |                   |         |
| Name             | Name__c          | Text(25)              |                   |         |
| Owner            | OwnerId          | Lookup(User,Group)    | ✓                 |         |
| property         | property__c      | Lookup(property)      | ✓                 |         |
| property Name    | Name             | Text(80)              | ✓                 |         |
| sfqf             | sfqf__c          | Text(18)              |                   |         |
| Type             | Type__c          | Picklist              |                   |         |

SETUP > OBJECT MANAGER  
**Payment for tenantat**

**Fields & Relationships**  
7 Items, Sorted by Field Label

| FIELD LABEL       | FIELD NAME           | DATA TYPE          | CONTROLLING FIELD | INDEXED |
|-------------------|----------------------|--------------------|-------------------|---------|
| Amount            | Amount__c            | Number(18, 0)      |                   |         |
| check for payment | check_for_payment__c | Picklist           |                   |         |
| Created By        | CreatedById          | Lookup(User)       |                   |         |
| Last Modified By  | LastModifiedById     | Lookup(User)       |                   |         |
| Owner             | OwnerId              | Lookup(User,Group) | ✓                 |         |
| Payment date      | Payment_date__c      | Date               |                   |         |
| Payment Name      | Name                 | Text(80)           | ✓                 |         |

Setup > OBJECT MANAGER

## lease

| Fields & Relationships         |                  |                    |                   |         |
|--------------------------------|------------------|--------------------|-------------------|---------|
| 7 Items, Sorted by Field Label |                  |                    |                   |         |
| FIELD LABEL                    | FIELD NAME       | DATA TYPE          | CONTROLLING FIELD | INDEXED |
| Created By                     | CreatedById      | Lookup(User)       |                   |         |
| End date                       | End_date_c       | Date               |                   |         |
| Last Modified By               | LastModifiedById | Lookup(User)       |                   |         |
| lease Name                     | Name             | Text(80)           |                   | ✓       |
| Owner                          | OwnerId          | Lookup(User/Group) |                   | ✓       |
| property                       | property_c       | Lookup(property)   |                   | ✓       |
| start date                     | start_date_c     | Date               |                   |         |

Setup > OBJECT MANAGER

## Tenant

| Fields & Relationships         |                  |                    |                   |         |
|--------------------------------|------------------|--------------------|-------------------|---------|
| 7 Items, Sorted by Field Label |                  |                    |                   |         |
| FIELD LABEL                    | FIELD NAME       | DATA TYPE          | CONTROLLING FIELD | INDEXED |
| Created By                     | CreatedBy        | Lookup(User)       |                   |         |
| Email                          | Email_c          | Email              |                   |         |
| Last Modified By               | LastModifiedById | Lookup(User)       |                   |         |
| Owner                          | OwnerId          | Lookup(User/Group) |                   | ✓       |
| Phone                          | Phone_c          | Phone              |                   |         |
| status                         | status_c         | Picklist           |                   |         |
| Tenant Name                    | Name             | Text(80)           |                   | ✓       |

- Developed Lightning App with relevant tabs

The screenshot shows the 'App Details & Branding' tab in the Lightning App Builder. The left sidebar has 'App Settings' selected. The main area shows the following fields:

- App Details**
  - App Name:** Lease Management
  - Developer Name:** Lease\_Management
  - Description:** Application to efficiently handle the processes related to leasing real estate properties.
- App Branding**
  - Image:** A placeholder image showing a person in a blue shirt.
  - Primary Color Hex Value:** #0070D2
- Org Theme Options**
  - Use the app's image and color instead of the org's custom theme
- App Launcher Preview**
  - A preview card for 'Lease Management' with the image, name, and description.

The screenshot shows the 'Navigation Items' tab in the Lightning App Builder. The left sidebar has 'App Settings' selected. The main area shows the following interface:

- Available Items** (left pane):
  - Accounts
  - Activation Targets
  - Activations
  - All Sites
  - Alternative Payment Methods
  - Analytics
  - App Launcher
  - Appointment Categories
  - Appointment Invitations
  - Approval Requests
- Selected Items** (right pane):
  - Payment
  - Tenants
  - property
  - lease

Lightning App Builder | App Settings | Pages | Lease Management | ? Help

App Details & Branding  
App Options  
Utility Items (Desktop Only)  
Navigation Items  
**User Profiles**

User Profiles  
Choose the user profiles that can access this app.

Available Profiles

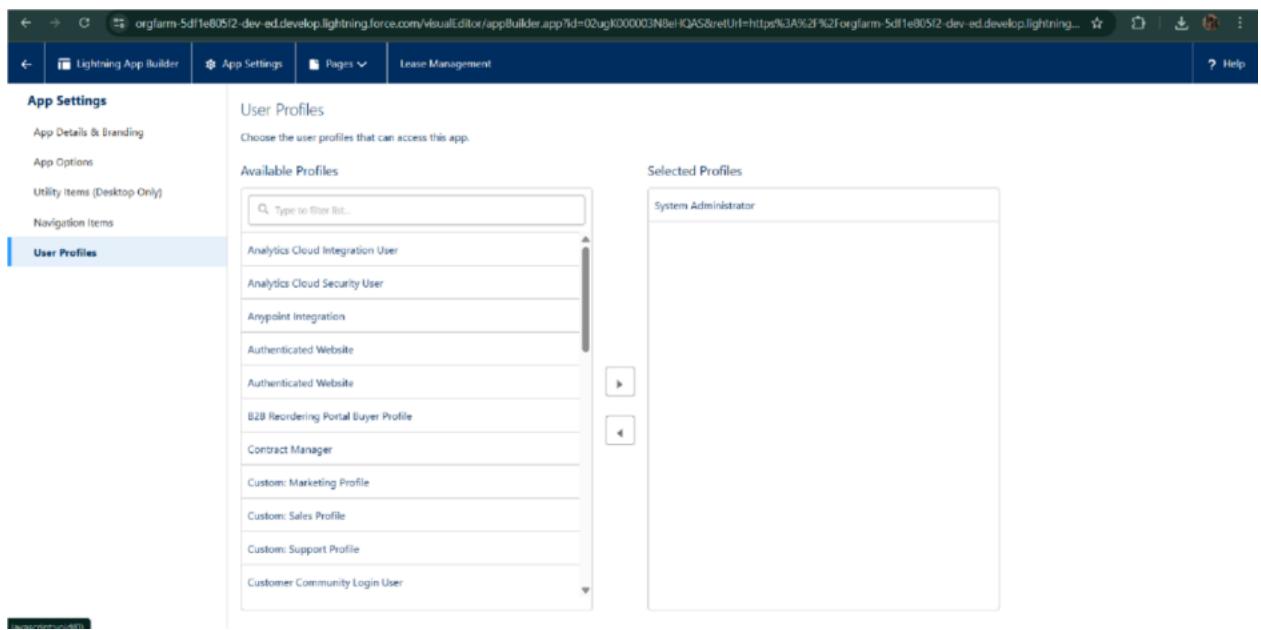
Type to filter list...

- Analytics Cloud Integration User
- Analytics Cloud Security User
- Anypoint Integration
- Authenticated Website
- Authenticated Website
- B2B Reordering Portal Buyer Profile
- Contract Manager
- Custom: Marketing Profile
- Custom: Sales Profile
- Custom: Support Profile
- Customer Community Login User

Selected Profiles

System Administrator

▶◀



Lease Management | Payment | Tenants | property | lease | ?

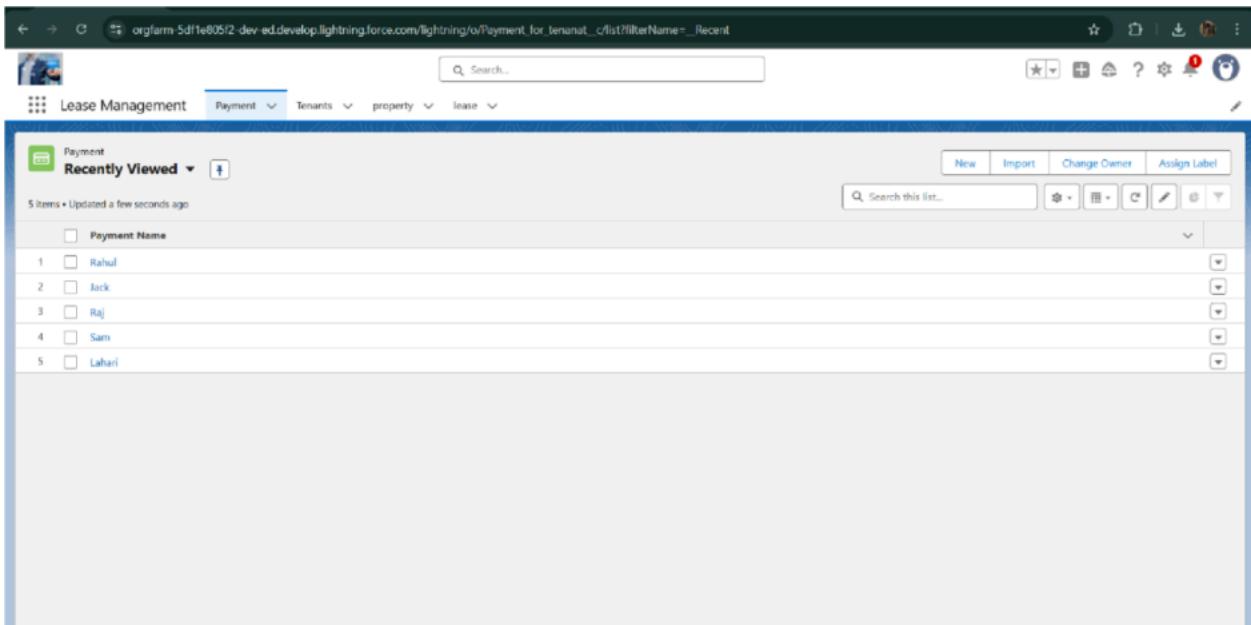
Recently Viewed ▾

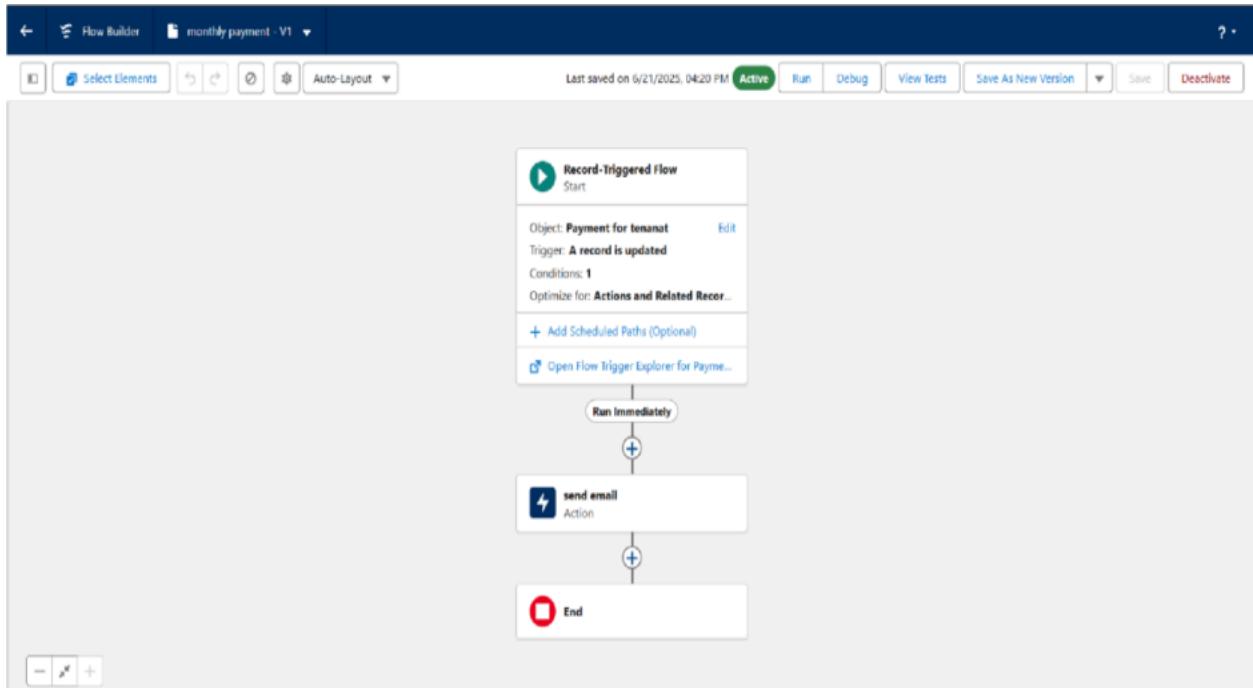
5 items • Updated a few seconds ago

|   | Payment Name | Actions |
|---|--------------|---------|
| 1 | Rahul        | ⋮       |
| 2 | Jack         | ⋮       |
| 3 | Raj          | ⋮       |
| 4 | Sam          | ⋮       |
| 5 | Lahari       | ⋮       |

New Import Change Owner Assign Label

Search this list... ⋮





- Implemented Flows for monthly rent and payment success
- To create a validation rule to a Lease Object

Setup > Object Manager

## lease

**Validation Rule Edit**

Role Name: lease\_end\_date

Active:

Description:

**Error Condition Formula**

Example: `Discount_Percent_C>0.30` More Examples...

If this formula expression is true, display the text defined in the Error Message area.

`End_date_c < start_date_c`

**Functions**

- All Function Categories --
- ABS
- ACOS
- ADDMONTHS
- AND
- ASCII
- ASIN

**Quick Tips**

- Operators & Functions

**Error Message**

Your End date must be greater than start date

Setup > Object Manager

## lease

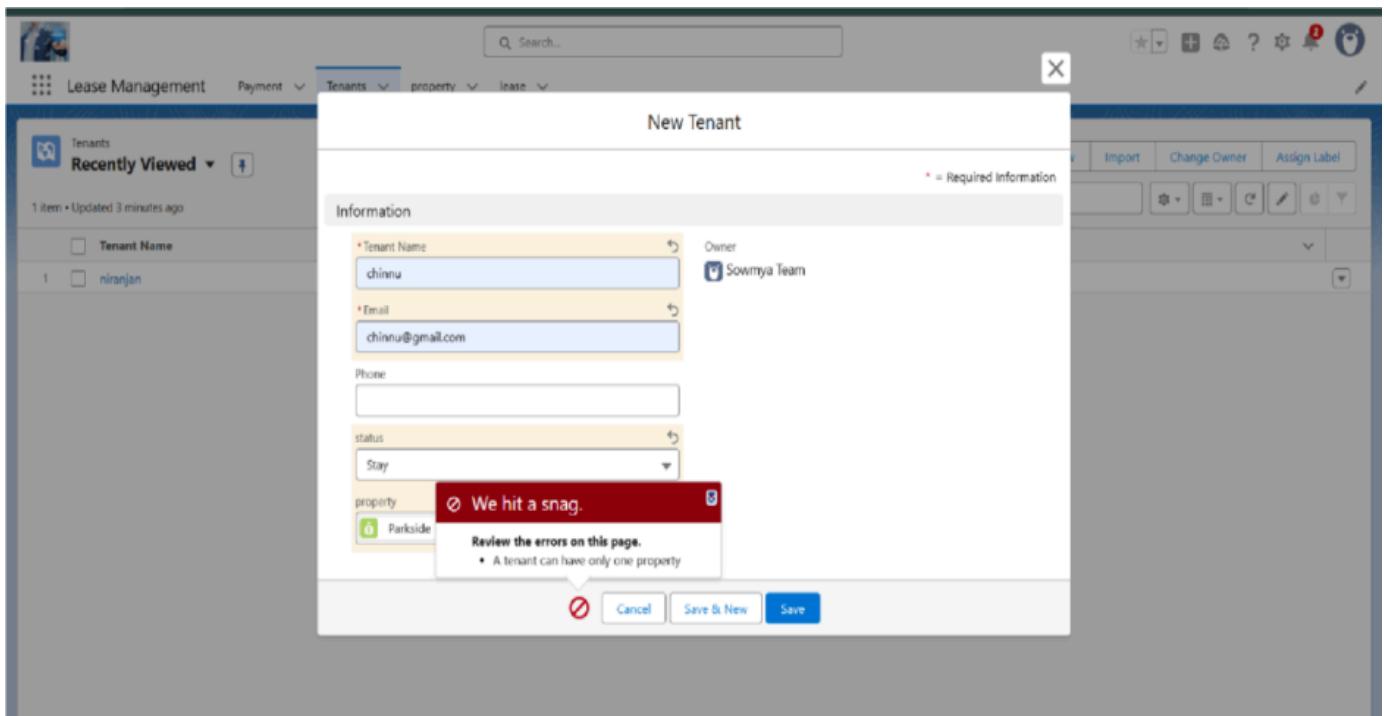
**lease Validation Rule**

Back to lease

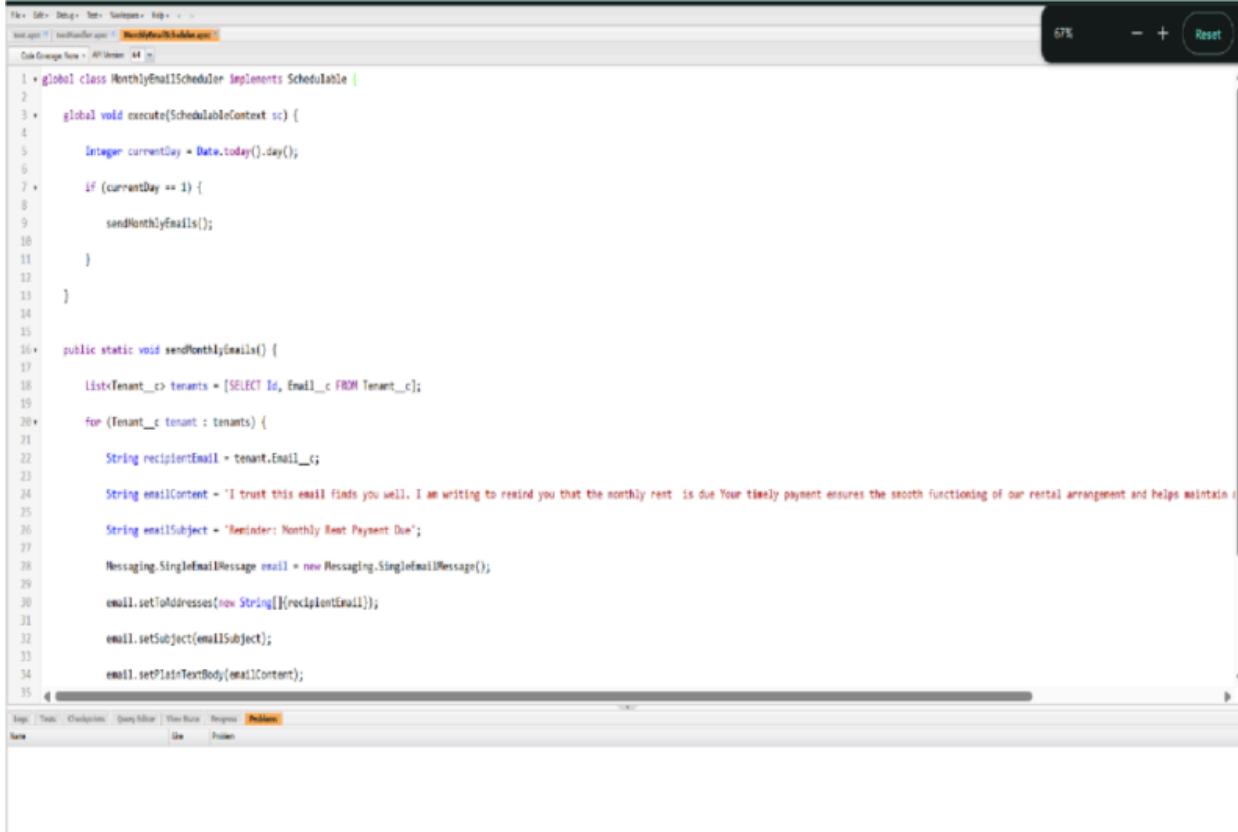
**Validation Rule Detail**

|                         |   |               |   |
|-------------------------|---|---------------|---|
| Role Name               | lease_end_date                            | Active        | <input checked="" type="checkbox"/>           |
| Error Condition Formula | <code>End_date_c &lt; start_date_c</code> | Error Message | Your End date must be greater than start date |
| Description             |   |               |   |
| Created By              | Sowmya_Team                               | Modified By   | Sowmya_Team                                   |
|                         | 6/19/2025, 5:37 AM                        |               | 6/26/2025, 7:47 AM                            |

- Added Apex trigger to restrict multiple tenants per property



- Scheduled monthly reminder emails using Apex class



The screenshot shows the Salesforce Developer Console with the following details:

- Page Title:** MonthlyEmailScheduler.apx
- Code Coverage Area:** API Version: 44
- Code:**

```

1 * global class MonthlyEmailScheduler implements Scheduleable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8
9             sendMonthlyEmails();
10        }
11    }
12
13 }
14
15
16 * public static void sendMonthlyEmails() {
17
18     List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
19
20     for (Tenant__c tenant : tenants) {
21
22         String recipientEmail = tenant.Email__c;
23
24         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';
25
26         String emailSubject = 'Reminder: Monthly Rent Payment Due';
27
28         Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
29
30         email.setToAddresses(new String[]{recipientEmail});
31
32         email.setSubject(emailSubject);
33
34         email.setPlainTextBody(emailContent);
35

```

- Built and tested email templates for leave request, approval, rejection, payment, and reminders

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template name is 'Leave approved'. The 'Email Template Detail' section shows the following details:

| Email Templates from Salesforce      | Unified Public Classic Email Templates |
|--------------------------------------|--|
| Email Template Name: Leave approved  | Leave_approved                         |
| Template Unique Name: leave_approved | leave_approved                         |
| Encoding: Unicode (UTF-8)            | Unicode (UTF-8)                        |
| Author: Scenova Team [Changed]       | Scenova Team [Changed]                 |
| Description:                         |  |
| Created By: Scenova Team             | Scenova Team, 6/20/2025, 1:08 AM       |

Buttons: Edit, Delete, Close.

Available For Use:

Last Used Date: Times Used: 0

Modified By: Scenova Team, 6/20/2025, 1:08 AM

**Email Template** Send Test and Verify Merge Fields

**Subject:** Leave approved

**Plain Text Preview:**

```
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 status is confirmed. You can review now.
```

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template name is 'tenant leaving'. The 'Email Template Detail' section shows the following details:

| Email Templates from Salesforce      | Unified Public Classic Email Templates |
|--------------------------------------|--|
| Email Template Name: tenant leaving  | tenant_leaving                         |
| Template Unique Name: tenant_leaving | tenant_leaving                         |
| Encoding: Unicode (UTF-8)            | Unicode (UTF-8)                        |
| Author: Scenova Team [Changed]       | Scenova Team [Changed]                 |
| Description:                         |  |
| Created By: Scenova Team             | Scenova Team, 6/20/2025, 1:06 AM       |

Buttons: Edit, Delete, Close.

Available For Use:

Last Used Date: Times Used: 0

Modified By: Scenova Team, 6/20/2025, 1:06 AM

**Email Template** Send Test and Verify Merge Fields

**Subject:** request for approve the leave

**Plain Text Preview:**

```
Dear {Tenant__c.CreatedBy}.

Please approve my leave.
```

The screenshot shows the Salesforce Setup interface with the 'Email' section selected. Under 'Email', 'Classic Email Templates' is chosen. A search bar at the top left shows 'email template'. The main content area displays the 'Leave rejected' email template details. The template name is 'Leave rejected' and it is associated with 'Unfiled Public: Classic Email Templates'. The 'Available For Use' checkbox is checked. The 'Email Template Detail' section includes fields for Email Template Name (Leave rejected), Template Unique Name (Leave\_rejected), Encoding (Unicode (UTF-8)), Author (Sohamya\_Team [Channe]), and Description (Leave rejected). The 'Created By' field shows Sohamya\_Team on 6/20/2025, 1:11 AM, and the 'Modified By' field shows Sohamya\_Team on 6/20/2025, 1:11 AM. Below the detail section is a preview pane titled 'Email Template' with a 'Plain Text Preview' section containing the message 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".

The screenshot shows the Salesforce Setup interface with the 'Email' section selected. Under 'Email', 'Classic Email Templates' is chosen. A search bar at the top left shows 'email template'. The main content area displays the 'Tenant Email' email template details. The template name is 'Tenant\_Email' and it is associated with 'Unfiled Public: Classic Email Templates'. The 'Available For Use' checkbox is checked. The 'Email Template Detail' section includes fields for Email Template Name (Tenant\_Email), Template Unique Name (Tenant\_Email), Encoding (Unicode (UTF-8)), Author (Sohamya\_Team [Channe]), and Description (Urgent: Monthly Rent Payment Reminder). The 'Created By' field shows Sohamya\_Team on 6/20/2025, 1:12 AM, and the 'Modified By' field shows Sohamya\_Team on 6/20/2025, 1:12 AM. Below the detail section is a preview pane titled 'Email Template' with a 'Plain Text Preview' section containing the message 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.".

The screenshot shows the Salesforce Setup interface with the search bar set to 'email template'. The left sidebar shows 'Email' categories: 'Classic Email Templates' (selected) and 'Lightning Email Templates'. A message at the top says ' Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'Classic Email Templates' and shows a 'Text Email Template' named 'tenant payment'. The 'Email Template Detail' section includes fields like 'Email Template Name' (tenant payment), 'Template Unique Name' (tenant\_payment), 'Encoding' (Unicode (UTF-8)), 'Author' (Sowmya\_Team [Change]), 'Description' (Created By Sowmya\_Team, 6/20/2025, 1:13 AM), and 'Modified By' (Sowmya\_Team, 6/20/2025, 1:13 AM). The 'Available For Use' checkbox is checked. Below this is a preview window showing the email template's subject ('Confirmation of Successful Monthly Payment') and plain text preview ('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.').

- Approval Process creation

For Tenant Leaving:

The screenshot shows the Salesforce Setup interface with the search bar set to 'approval'. The left sidebar shows 'Data' categories: 'Mass Transfer Approval Requests', 'Feature Settings' (with 'Approval Settings' selected), and 'Process Automation' (with 'Approval Processes' selected). A message at the top says ' Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'Approval Processes' and shows an 'Approval Process' named 'Tenant: TenantApproval'. The 'Process Definition Detail' section includes fields like 'Process Name' (TenantApproval), 'Unique Name' (TenantApproval), 'Description' ( ), 'Entry Criteria' (Tenant\_\_c.status equals Stay), 'Record Editability' (Administrator ONLY), 'Next Automated Approver Determined By' (Active checked), 'Approval Assignment Email Template' ( ), 'Initial Submitters' (Tenant Owner), 'Created By' (Sowmya\_Team, 6/23/2025, 3:41 AM), and 'Modified By' (Sowmya\_Team, 6/26/2025, 11:57 PM). Below this are sections for 'Initial Submission Actions' (Record Lock: Description - Lock the record from being edited) and 'Approval Steps' (Step 1: Action - Show Actions | Edit, Step Number - 1, Name - Step 1, Description - , Criteria - , Assigned Approver - User:Sowmya\_Team, Reject Behavior - Final Rejection).

For Check for Vacant:

The screenshot shows the Salesforce Setup interface with the following details:

- Approval Processes:** Tenant: check for vacant
- Process Definition Detail:**
  - Process Name: check for vacant
  - Unique Name: check\_for\_vacant
  - Description: Tenant: status EQUALS Leaving
  - Entry Criteria: Record Editability Administrator ONLY
  - Next Automated Approver Determined By: Allow Submitters to Recall Approval Requests
  - Approval Assignment Email Template: Leave approved
  - Initial Submitters: Tenant Owner
  - Created By: Sowmya\_Team
  - Modified By: Sowmya\_Team
- Initial Submission Actions:**
  - Action: Type: Record Lock Description: Lock the record from being edited
  - Action: Type: Email Alert Description: PLEASE APPROVE MY RATES
- Approval Steps:**

| Action              | Step Number | Name  | Description | Criteria | Assigned Approver | Reject Behavior |
|---------------------|-------------|-------|-------------|----------|-------------------|-----------------|
| Show Actions   Edit | 1           | step1 |             |          | User:Sowmya_Team  | Final Rejection |

- Apex Trigger

## Create an Apex Trigger

```

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

```

The screenshot shows the Salesforce Developer Console with the following details:

- Trigger:** ImtHandler.apex
- Code Coverage:** None
- API Version:** 54
- Trigger Details:**

```

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

```
- Open Modal:** A modal window titled "Open" is displayed, listing various entity types (Entity Type) such as Classes, Triggers, Pages, Objects, Static Resources, and Packages. The entry "test" is selected under the "Classes" category.
- Logs:** Logs, Tests, Checkpoints, Query Editor, View Status, Progress, Problems

Developer Console - Google Chrome

File • Edit • Debug • Test • Workspace • Help • < >

test.apex testHandler.apex MonthlyEmailScheduler.apex

Code Coverage: None • API Version: 44 • Go To

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

Logs Tests Checkpoints Query Editor View State Progress Problems

## Create an Apex Handler class

Developer Console - Google Chrome

File • Edit • Debug • Test • Workspace • Help • < >

test.apex testHandler.apex MonthlyEmailScheduler.apex

Code Coverage: None • API Version: 44 • Go To

```
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.Id);
        }
        for (Tenant__c newTenant : newList) {
            if (newTenant.Property__c != null) {
                newTenantaddError('A');
            }
        }
    }
}
```

Existing Type Entity Name Namespace Related

| Existing Type    | Entity      | Name      | Namespace | Related                |
|------------------|-------------|-----------|-----------|------------------------|
| Apex Trigger     | testHandler | test      |           | ApexTrigger References |
| Classes          |             | property  |           | CustomField References |
| Triggers         |             | Tenant__c |           | Object References      |
| Pages            |             | Tenant__c |           | Object References      |
| Page Components  |             |           |           |                        |
| Objects          |             |           |           |                        |
| Static Resources |             |           |           |                        |
| Packages         |             |           |           |                        |

Open Filter: Filter the repository (\* = any string) Hide Managed Packages Refresh

Logs Tests Checkpoints Query Editor View State Progress Problems

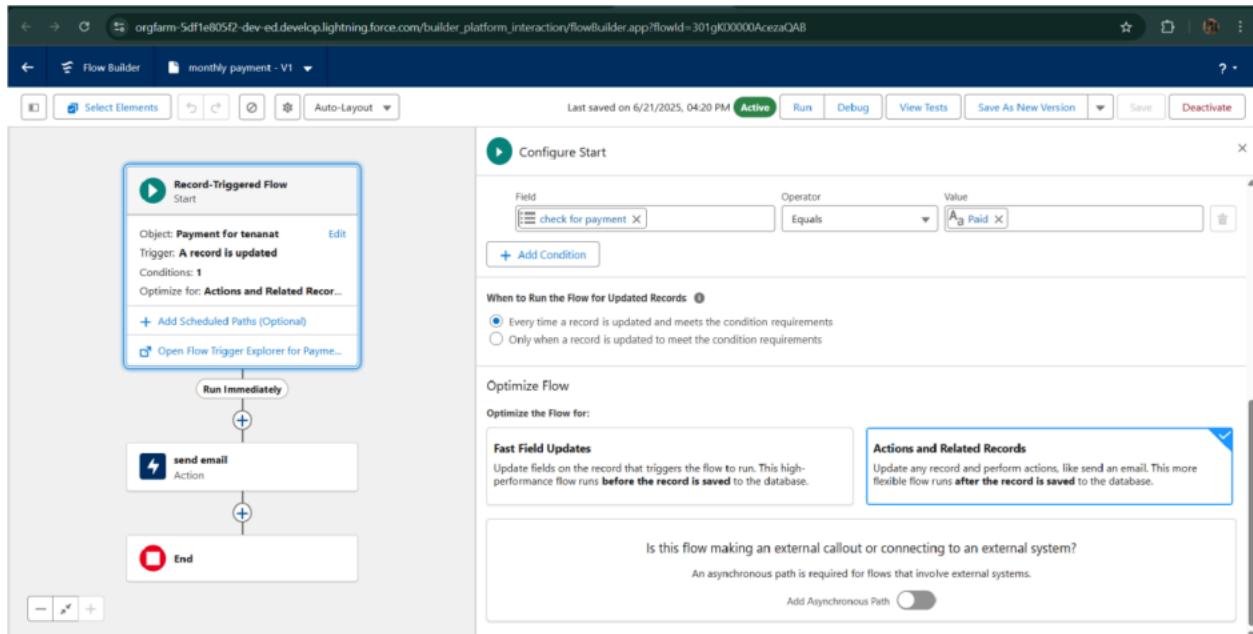
```

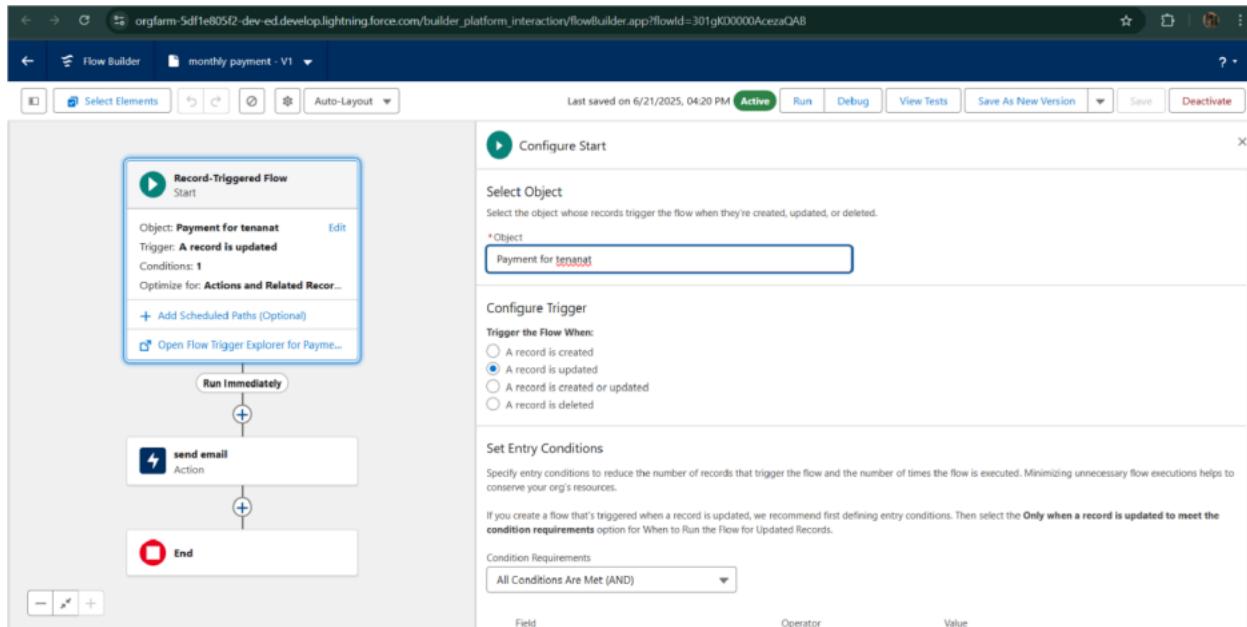
1 • public class testHandler {
2
3 •     public static void preventInsert(List<Tenant__c> newList) {
4
5         Set<Id> existingPropertyIds = new Set<Id>();
6
7         for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c WHERE Property__c != null]) {
8
9             existingPropertyIds.add(existingTenant.Property__c);
10
11     }
12
13
14     for (Tenant__c newTenant : newList) {
15
16
17         if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
18
19             newTenantaddError('A tenant can have only one property');
20
21         }
22
23     }
24
25 }

```

The screenshot shows the Salesforce Developer Console interface. The tab bar at the top includes 'File', 'Edit', 'Debug', 'Test', 'Workspace', 'Help', and 'Go To'. Below the tabs, there are sections for 'Code Coverage', 'New', 'API Version' (set to 64), and 'Go To'. The main area displays the Apex code for 'testHandler.apc'. The code defines a static method 'preventInsert' that takes a list of 'Tenant\_\_c' records. It first creates a set of existing property IDs. Then, for each new tenant in the list, it checks if the tenant already has a property assigned. If so, it adds an error message to the tenant record. The code uses standard Apex syntax with annotations like '•' for class members.

## ● FLOWS





- Schedule class:  
Create an Apex Class

```
1 * global class MonthlyEmailScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8
9             sendMonthlyEmails();
10
11     }
12
13 }
14
15
16 * public static void sendMonthlyEmail
17
18     List<Tenant__c> tenants = [SELECT
19
20         for (Tenant__c tenant : tenants
21
22             String recipientEmail = tenant.Email__c;
```

Open

| Entity Type      | Edition                     | Related                               |
|------------------|-----------------------------|---------------------------------------|
| Classes          | Name: MonthlyEmailScheduler | Name: CustomField_1, Email, Tenant__c |
| Triggers         |                             |                                       |
| Pages            |                             |                                       |
| Page Components  |                             |                                       |
| Objects          |                             |                                       |
| Static Resources |                             |                                       |
| Packages         |                             |                                       |

The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is `orgform-5d1fe05f2-dev-ed.develop.my.salesforce.com/_ui/common/ apex/debug/ApexCSPage`. The code editor displays the following Apex class:

```
1 *global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.Today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8
9
10
11
12
13
14
15
16     public static void sendMonthlyEmails() {
17         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18         for (Tenant__c tenant : tenants) {
19             String recipientEmail = tenant.Email__c;
20             String emailSubject = 'Wentover: monthly rent payment due';
21             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.' ;
22             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
23             email.setToAddresses(new String[]{recipientEmail});
24             email.setSubject(emailSubject);
25             email.setPlainTextBody(emailContent);
26             messaging.sendEmail(new messaging.SingleEmailMessage[] {email});
27         }
28     }
29 }
30 }
```

## Schedule Apex class

The screenshot shows the Salesforce Setup Apex Classes page. The URL is `orgform-5d1fe05f2-dev-ed.develop.my.salesforce.com/_ui/common/programming/setup/apex/classes/plugin?pg=0&p=0&0000000588`. The page displays the Apex Class **MonthlyEmailScheduler**.

**Apex Class Detail**

| Name                  | Namespace Prefix | Status           | Active                           |
|-----------------------|------------------|------------------|----------------------------------|
| MonthlyEmailScheduler |                  | Code Coverage    | 0% (0/15)                        |
|                       |                  | Last Modified By | SOMMYA.Team , 6/23/2025, 2:47 AM |

**Class Body**

```
1 global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.Today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8
9
10
11
12
13
14
15
16     public static void sendMonthlyEmails() {
17         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18         for (Tenant__c tenant : tenants) {
19             String recipientEmail = tenant.Email__c;
20             String emailSubject = 'Wentover: monthly rent payment due';
21             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.' ;
22             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
23             email.setToAddresses(new String[]{recipientEmail});
24             email.setSubject(emailSubject);
25             email.setPlainTextBody(emailContent);
26             messaging.sendEmail(new messaging.SingleEmailMessage[] {email});
27         }
28     }
29 }
```

The screenshot shows a Salesforce Lightning interface for a Tenant record. The top navigation bar includes icons for back, forward, search, and user profile. The main header displays "Lease Management" and "Tenants". The left sidebar has sections for "Related" and "Details". The "Details" section contains fields for Tenant Name (Janani), Email (aswinjanani2011@gmail.com), Phone, status (Stay), Property (House), Created By (JANANI P), and Last Modified By (JANANI P). The right sidebar features an "Activity" section with various icons for different types of activities, a "Filters" dropdown, and a "Upcoming & Overdue" section which is currently empty. Buttons for "New Contact", "Edit", and "New Opportunity" are located at the top right of the main content area.

This screenshot shows the same Salesforce record detail page for Tenant Janani, but with a different view of the right sidebar. The "Notifications" section is now visible, displaying five notifications all from "JANANI" regarding "Approval request for the tenant is approved". The notifications are dated Sep 6, 2025, at various times between 8:24 PM and 2:42 PM. A "Mark all as read" button is located at the top right of the notifications panel.

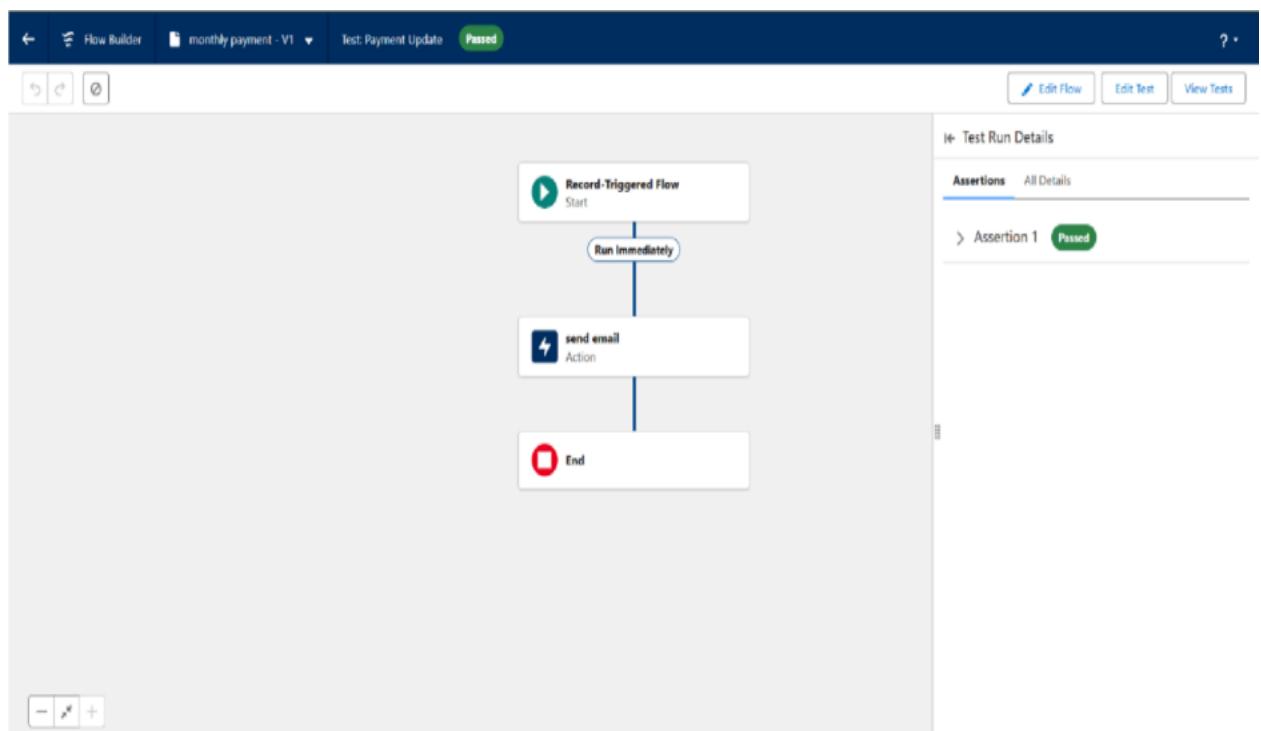
# FUNCTIONAL AND PERFORMANCE TESTING

## Performance Testing

- Trigger validation by entering duplicate tenant-property records



- Validation Rule checking



- Test flows on payment update

The screenshot displays a software interface for 'Lease Management'. On the left, a 'Tenant' record for 'Janani' is shown with details like Name, Email, Phone, Status, and Property type. The 'Details' tab is selected. On the right, a 'Notifications' sidebar lists five recent approvals from 'JANANI P.'.

| Notification Details                                  | Date                 |
|---|----------------------|
| Approval request for the tenant is approved<br>Janani | 3 hours ago          |
| Approval request for the tenant is approved<br>Janani | Sep 8, 2025, 8:24 PM |
| Approval request for the tenant is approved<br>Janani | Sep 8, 2025, 8:21 PM |
| Approval request for the tenant is approved<br>Janani | Sep 6, 2025, 2:42 PM |
| Approval request for the tenant is approved<br>Janani | Sep 6, 2025, 2:39 PM |

The screenshot shows a CRM application interface with a top navigation bar featuring 'Lease Management', 'Payment', 'Tenants', 'property', and 'lease' dropdowns. A search bar and various tool icons are also present. The main content area displays two sections: 'Approval History (6+)' and 'Payment (2)'.  
**Approval History (6+)**  
A table with columns: Step Name, Date, Status, and Assigned To. The data is as follows:

| Step Name                  | Date               | Status    | Assigned To |
|----------------------------|--------------------|-----------|-------------|
| Step 1                     | 6/25/2025, 5:39 AM | Approved  | Sowmya Team |
| Approval Request Submitted | 6/25/2025, 5:39 AM | Submitted | Sowmya Team |
| Step 1                     | 6/23/2025, 3:59 AM | Rejected  | Sowmya Team |
| Approval Request Submitted | 6/23/2025, 3:58 AM | Submitted | Sowmya Team |
| Step 1                     | 6/23/2025, 3:55 AM | Approved  | Sowmya Team |
| Approval Request Submitted | 6/23/2025, 3:55 AM | Submitted | Sowmya Team |

[View All](#)

  
**Payment (2)**  
A table with a column: Payment Name. The data is as follows:

| Payment Name |
|--------------|
| Jack         |
| Rahul        |

[View All](#)

To the right of these sections is a sidebar with the following text:  
Get started by sending an email, scheduling a task, and more.  
No past activity. Past meetings and tasks marked as done show up here.

# RESULTS

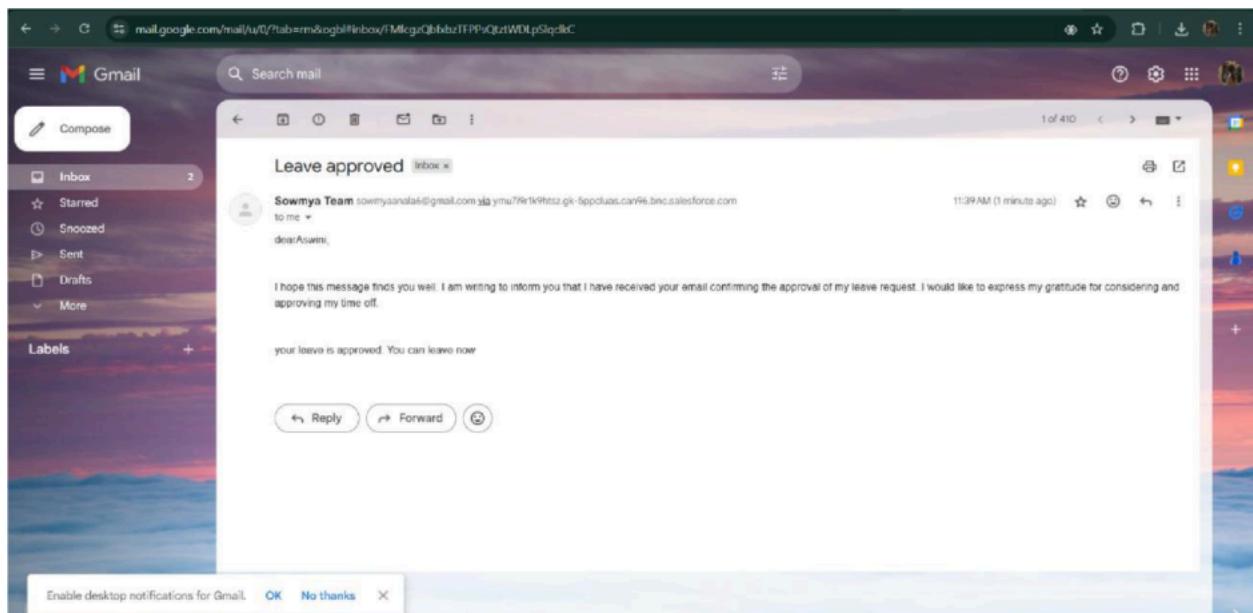
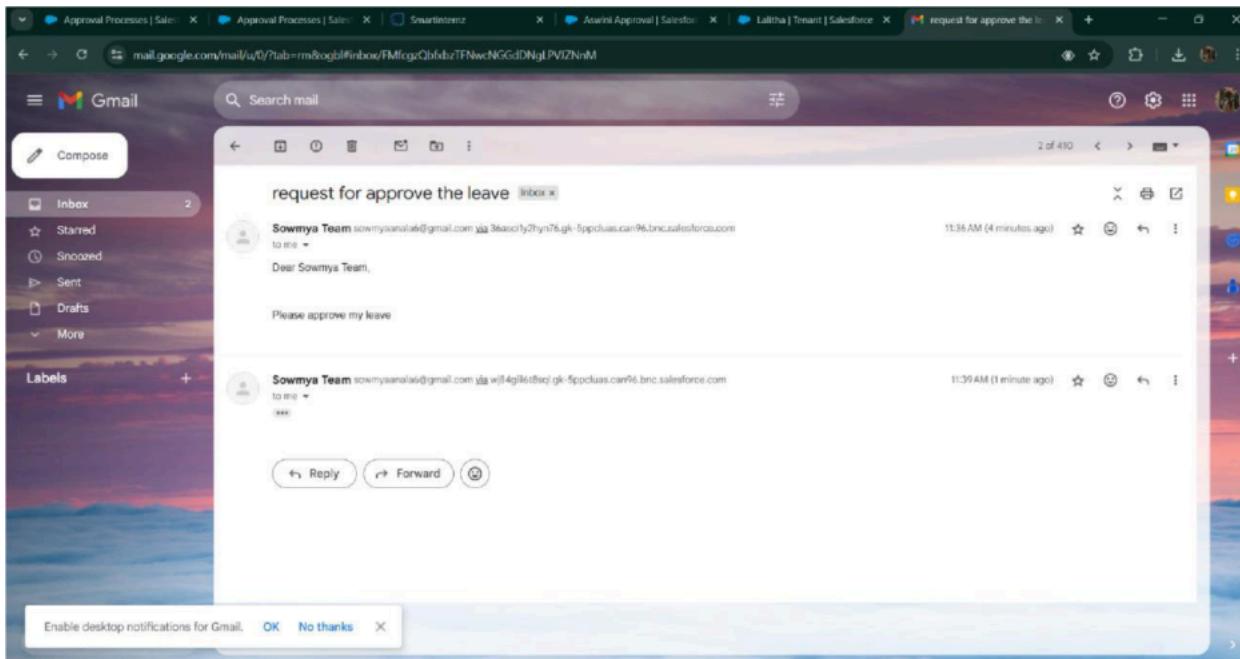
## Output Screenshots

The screenshot shows the Salesforce Setup interface under the 'Tabs' section. It displays three main sections: 'Custom Object Tabs', 'Web Tabs', and 'Visualforce Tabs'. The 'Custom Object Tabs' section lists tabs for 'Issue', 'Payment', 'property', and 'Tenants', each with a 'Tab Style' (Keys, Credit card, Back, Map) and an 'Edit | Del' button. The 'Web Tabs' and 'Visualforce Tabs' sections both show a message: 'No Web Tabs have been defined' and 'No Visualforce Tabs have been defined' respectively.

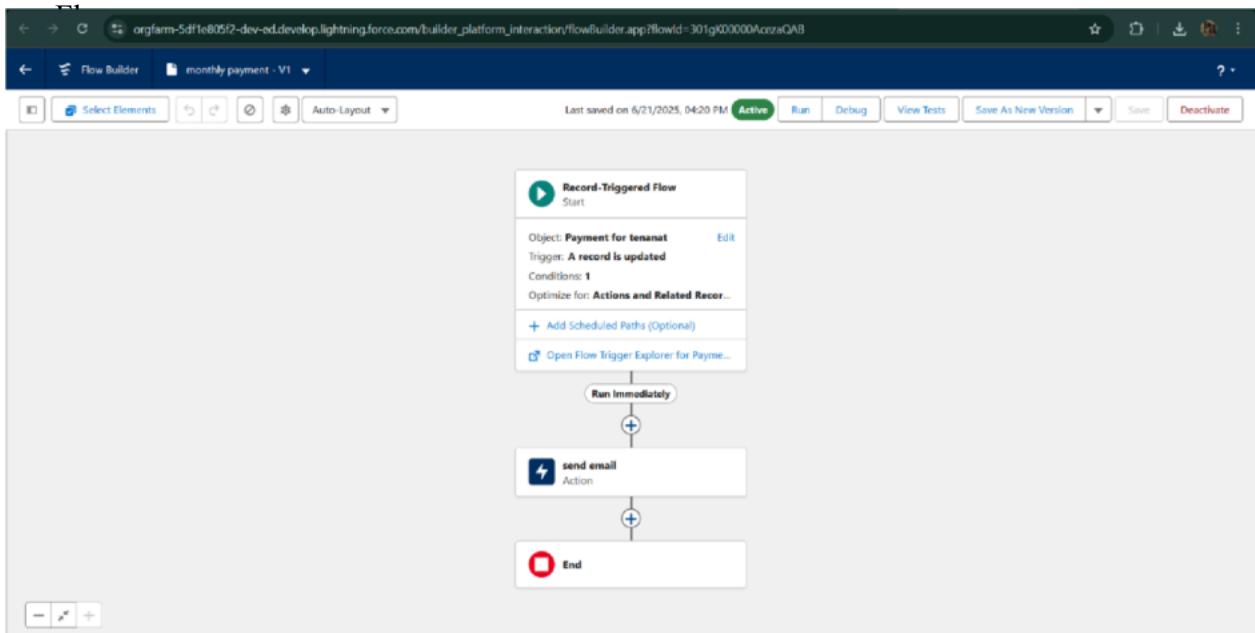
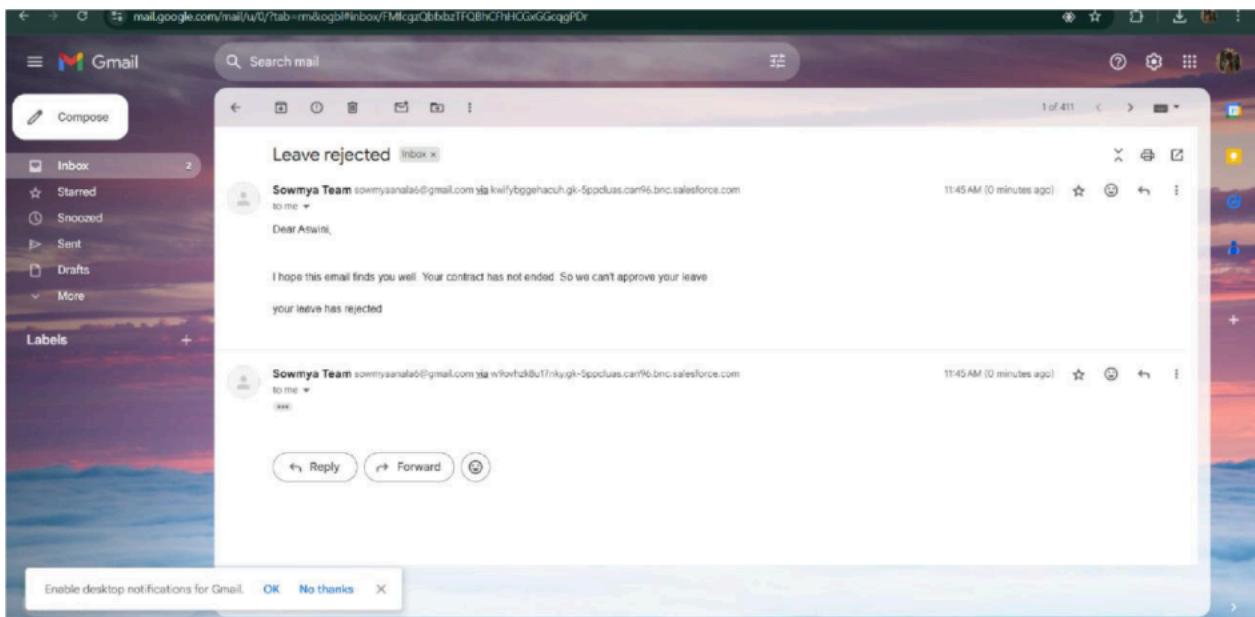
The screenshot shows the 'Approval History' page within the 'Lease Management' application. The page title is 'Approval History' and it shows a table of 8 items. The columns are: Step Name, Date, Status, Assigned To, Actual Approver, and Comments. The data is as follows:

| Step Name                    | Date               | Status    | Assigned To | Actual Approver | Comments          |
|------------------------------|--------------------|-----------|-------------|-----------------|-------------------|
| 1 Step 1                     | 6/25/2025, 5:39 AM | Approved  | Sowmya Team | Sowmya Team     | approved          |
| 2 Approval Request Submitted | 6/25/2025, 5:39 AM | Submitted | Sowmya Team | Sowmya Team     | leaving           |
| 3 Step 1                     | 6/23/2025, 3:59 AM | Rejected  | Sowmya Team | Sowmya Team     | Rejected          |
| 4 Approval Request Submitted | 6/23/2025, 3:58 AM | Submitted | Sowmya Team | Sowmya Team     | leaving           |
| 5 Step 1                     | 6/23/2025, 3:55 AM | Approved  | Sowmya Team | Sowmya Team     | Approved          |
| 6 Approval Request Submitted | 6/23/2025, 3:55 AM | Submitted | Sowmya Team | Sowmya Team     | leaving           |
| 7 Step 1                     | 6/23/2025, 3:44 AM | Approved  | Sowmya Team | Sowmya Team     | Approval Approved |
| 8 Approval Request Submitted | 6/23/2025, 3:42 AM | Submitted | Sowmya Team | Sowmya Team     | Leaving           |

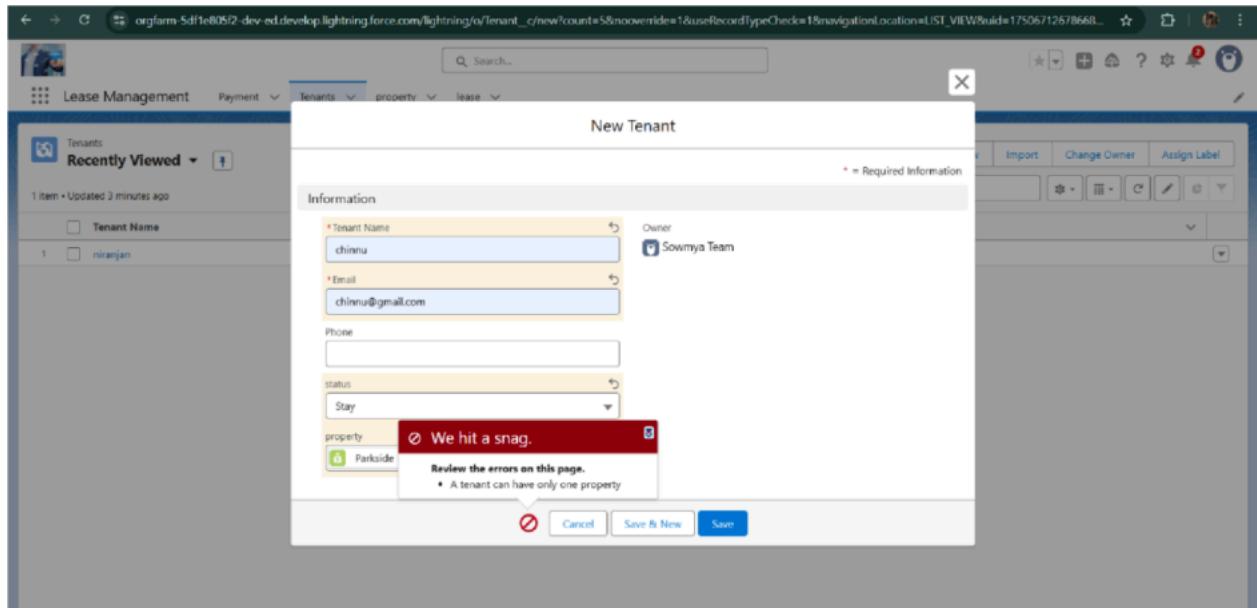
- Request for approve the leave



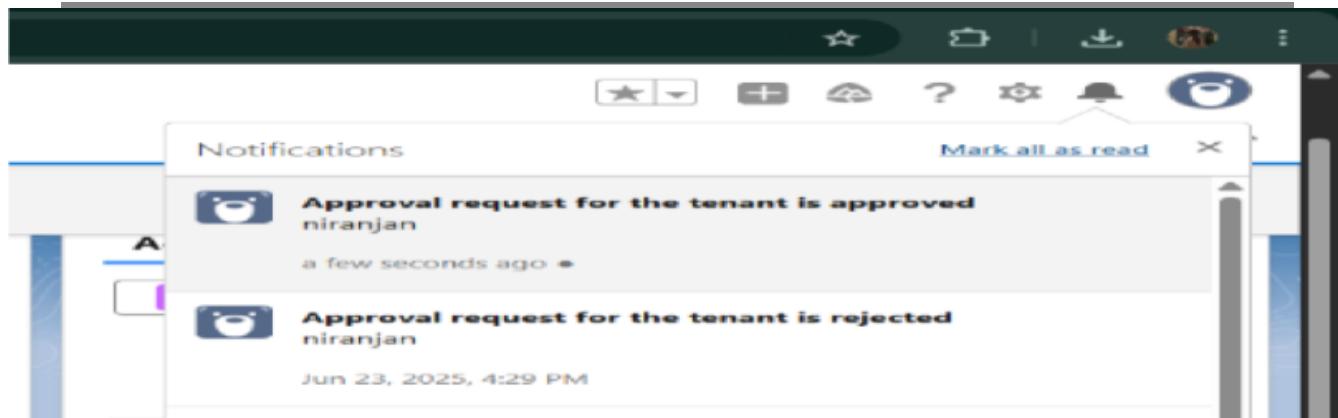
- Leave rejected



- Trigger error messages



- Approval process notifications



# ADVANTAGE

**Lower Upfront Costs:** Leasing eliminates large capital expenditures, allowing businesses to preserve cash for other needs.

**Flexibility:** Businesses can easily upgrade assets at the end of a lease term helping to avoid obsolescence.

**Tax Benefits:** Lease payments are often deductible as operating expenses, providing a tax advantage for the lessee.

# DISADVANTAGE

**Higher Long-Term Costs:** While initial costs are lower, the total cost of leasing can be higher than buying over the asset's lifespan.

**No Ownership Equity:** At the end of the lease, lessees do not gain ownership of the asset, and their payments do not build equity.

**Restrictions:** Leased assets often come with limitations, such as mileage caps for vehicles or restrictions on customization.

# **CONCLUSION**

## **Summary of Achievements**

The Salesforce implementation for Lease Management has successfully streamlined and automated critical aspects of the leasing lifecycle. Key accomplishments include:

### **1. Comprehensive Lease Lifecycle Management**

- o Implemented a robust system to manage lease agreements, from creation and amendments to renewals and terminations.

### **2. Automated Payment Processes**

- o Enabled scheduled rent tracking, payment reminders, and overdue alerts, reducing manual oversight and improving cash flow management.

### **3. Efficient Expiration and Renewal Workflows**

- o Automated notifications and streamlined renewal processes, ensuring timely follow-ups and improved lease retention rates.

### **4. Enhanced Property and Tenant Management**

- o Centralized information for properties, units, and tenants, improving operational visibility and collaboration.

### **5. Actionable Insights and Reporting**

- o Delivered custom reports and dashboards to monitor revenue, lease performance, and occupancy metrics, supporting data-driven decision-making.

## **6. Improved User Experience**

- o Provided intuitive interfaces and mobile accessibility for leasing agents, property managers, and tenants, enhancing productivity and satisfaction.

## **7. Secure and Scalable Solution**

- o Ensured role-based access, field-level security, and compliance with legal requirements, enabling a reliable and scalable system for lease management.

## **8. Integration with External Systems**

- o Established seamless connections with payment gateways and accounting systems for end-to-end process automation.

# **REFERENCE**

1. International Journal of Advanced Research in Science, Communication and Technology (IJARSCT):

<https://ijarsct.co.in/Paper7646.pdf>

2. Lease-Management-System-on-Salesforce:

<https://github.com/VedantRajGaur/Lease-Management-System-on-Salesforce>.

3. Github lease Management:

<https://github.com/mukunthan893/NMLEASE-MANAGEMENT>

4. Salesforce documentation:

Build an Integrated Real Estate CRM for Tenant Management.

# APPENDIX

- Source Code:** Provided in Apex Classes and Triggers

## Test.apxt:

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

## testHandler.apxc:

```
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)) { newTenantaddError('A
tenant can have only one property');

    }

}
}

```

### **MothlyEmailScheduler.apxc:**

```

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});

}

}

}
```

