

# **LEASE MANAGEMENT**

**COLLEGE NAME :** SREE NARAYANA GURU COLLEGE

**COLLEGE CODE :** Bru36

**TEAM ID:** NM2025TMID26405

**TEAM MEMBERS:**

**Team LeaderName :** Krishna Kumar R

**Email :** krishnakumarrm3058@gmail.com

**Team Member 1 :** Antro Infant J

**Email :** antroinfant103@gmail.com

**Team Member 2 :** Anfal Dharvesh K

**Email :** anfalanfu2727@gmail.com

**Team Member 3 :** Anshil Rosario S

**Email :** Anshilrosario2005@gmail.com

**Team Member 4 :** Anas Muhammad M

**Email :** anasanasmuhammed01@gmail.com

# 1. INTRODUCTION

## 1.1 Project Overview

The Lease Management System is a Salesforce-based application that automates and streamlines real estate leasing processes. It manages tenant records, lease contracts, rent payments, and communication within a centralized platform, reducing manual effort and errors. Using Salesforce features like Flows, Approval Processes, and Email Alerts, the system ensures timely reminders, accurate payment tracking, and proper contract approvals. With dashboards and reports, stakeholders gain insights into occupancy, payments, and renewals, enabling better decision-making. Overall, it enhances efficiency, improves tenant satisfaction, and secures data with role-based access.



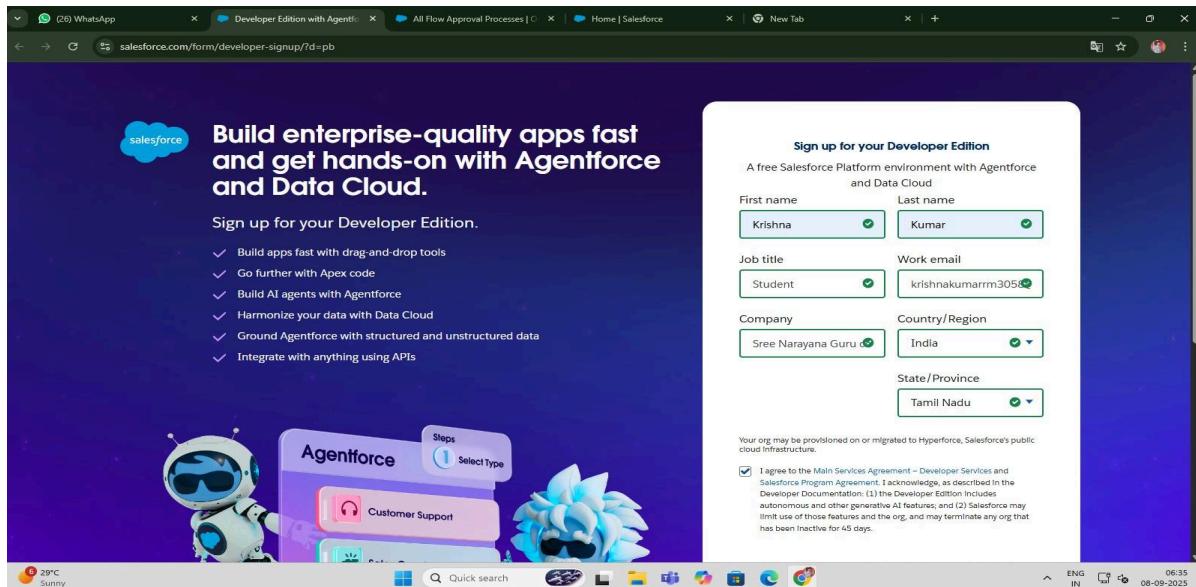
## 1.2 Purpose

The purpose of the Lease Management System is to provide a centralized, automated solution for managing properties, tenants, and lease activities. By using Salesforce features such as Flows, Approval Processes, and Email Alerts, the system reduces manual effort, improves accuracy, and ensures timely communication. It enhances compliance through secure role-based access and strengthens tenant–landlord relationships with transparent, efficient processes, ultimately increasing productivity and supporting better decision-making.

## DEVELOPMENT PHASE

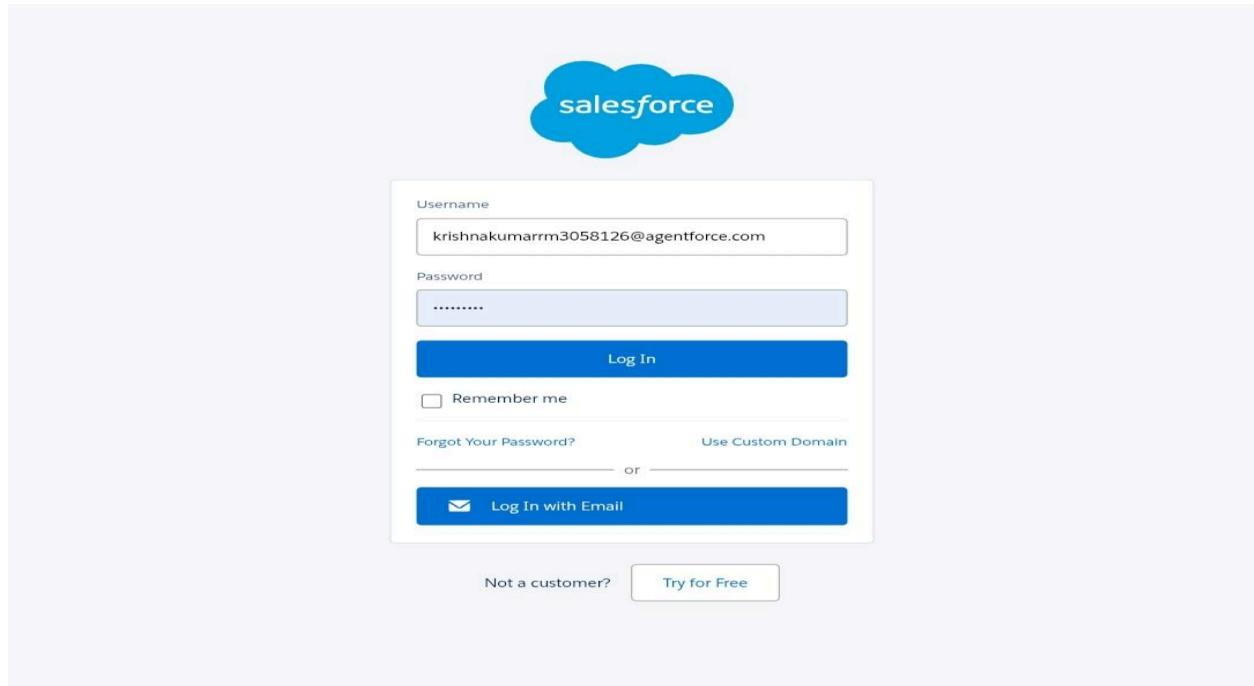
- **Creating Developer Account:**

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



Account activation then next login this link

<https://login.salesforce.com/>



- Created objects: Property, Tenant, Lease, Payment

A screenshot of the Salesforce Setup &gt; Object Manager interface. The page title is "Tenant". On the left, a sidebar lists various object configuration options like Fields &amp; Relationships, Page Layouts, and Record Types. The main content area shows the "Details" tab for the Tenant object. It includes sections for "Description" (API Name: Tenant\_c, Singular Label: Tenant, Plural Label: Tenants) and "Enable Reports" (checkboxes for Track Activities, Track Field History, and Deployment Status, all checked). Buttons for "Edit" and "Delete" are at the top right. The status bar at the bottom shows system information including weather (29°C sunny), quick search, and system status.

The screenshot shows the Salesforce Object Manager interface for the 'lease' object. The left sidebar lists various configuration tabs: 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 is selected. The right pane displays the 'Details' section for the 'lease' object. The 'Description' field is empty. The 'API Name' field is set to 'lease\_c'. The 'Custom' checkbox is checked. The 'Singular Label' field is set to 'lease'. The 'Plural Label' field is set to 'leases'. On the right, there are sections for 'Enable Reports' (checked), 'Track Activities' (checked), 'Track Field History' (checked), 'Deployment Status' (set to 'Deployed'), and 'Help Settings' (link to 'Standard salesforce.com Help Window'). At the top right of the main pane are 'Edit' and 'Delete' buttons. The bottom status bar shows the date and time as 08-09-2025.

The screenshot shows the Salesforce Object Manager interface for the 'property' object. The left sidebar lists the same configuration tabs as the previous screenshot. The main 'Details' tab is selected. The right pane displays the 'Details' section for the 'property' object. The 'Description' field is empty. The 'API Name' field is set to 'property\_c'. The 'Custom' checkbox is checked. The 'Singular Label' field is set to 'property'. The 'Plural Label' field is set to 'properties'. On the right, there are sections for 'Enable Reports' (checked), 'Track Activities' (checked), 'Track Field History' (checked), 'Deployment Status' (set to 'Deployed'), and 'Help Settings' (link to 'Standard salesforce.com Help Window'). At the top right of the main pane are 'Edit' and 'Delete' buttons. The bottom status bar shows the date and time as 08-09-2025.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for WhatsApp, Developer Edition with Agent!, Home | Salesforce, Payment for tenantat | Salesforce, and New Tab. The main header displays the URL: orgfarm-6fb9b6f541-dev-ed.lightning.force.com/lightning/setup/ObjectManager/01lgL000001veg5/Details/view. The page title is "Payment for tenantat". The left sidebar lists various configuration options under "Details": Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, Scoping Rules, and others. The main content area is titled "Details" and shows the following fields:

- Description
- API Name: Payment\_for\_tenantat\_c
- Custom:
- Singular Label: Payment for tenantat
- Plural Label: Payment
- Enable Reports:
- Track Activities:
- Track Field History:
- Deployment Status: Deployed
- Help Settings: [Standard salesforce.com Help Window](#)

At the bottom of the screen, the Windows taskbar shows the date and time (08-09-2025, 06:47), system tray icons, and a weather widget indicating 29°C and sunny.

- Configured fields and relationships

**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	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Name	Name__c	Text(25)		
Owner	OwnerId	Lookup(User,Group)	✓	
property	property__c	Lookup(property)	✓	
property/Name	Name	Text(80)	✓	
sfqt	sfqt__c	Text(18)		
Type	Type__c	Picklist		

**Fields & Relationships**  
8 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	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)	✓	
Payment date	Payment_date__c	Date		
Payment Name	Name	Text(80)	✓	
Tenant	Tenant__c	Lookup(Tenant)	✓	

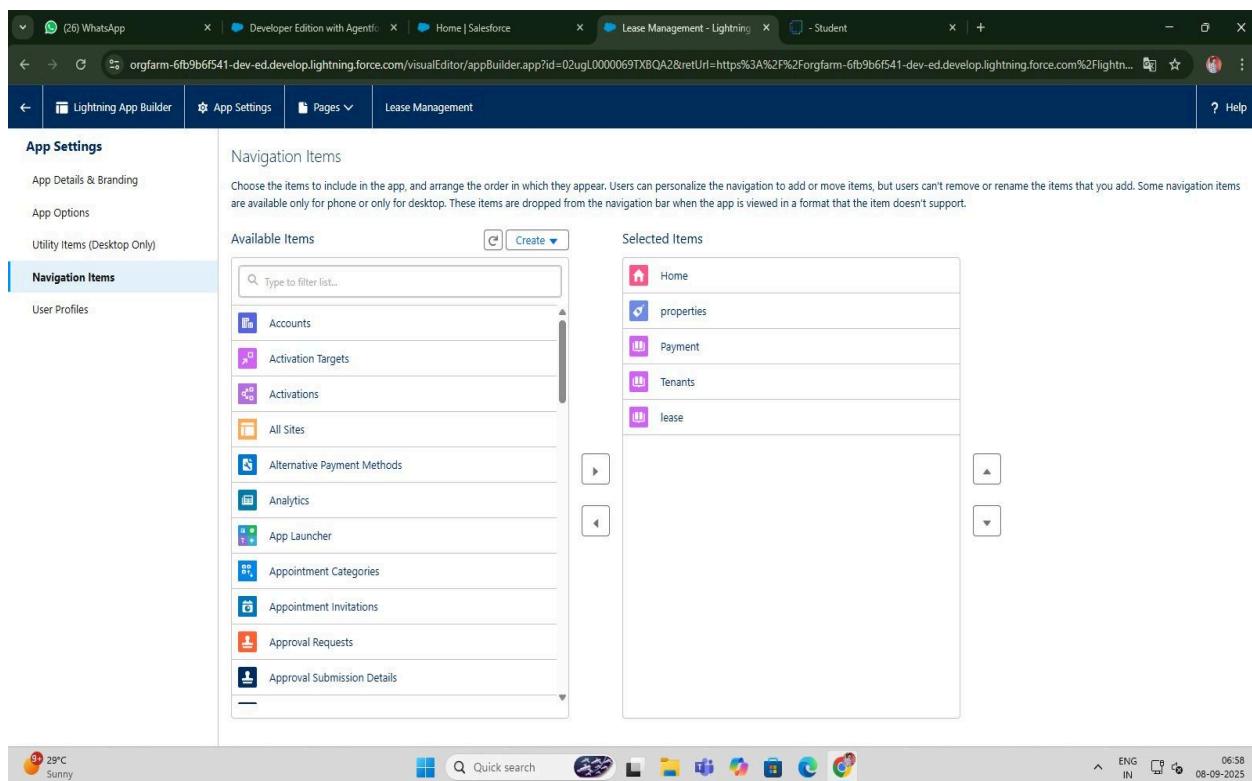
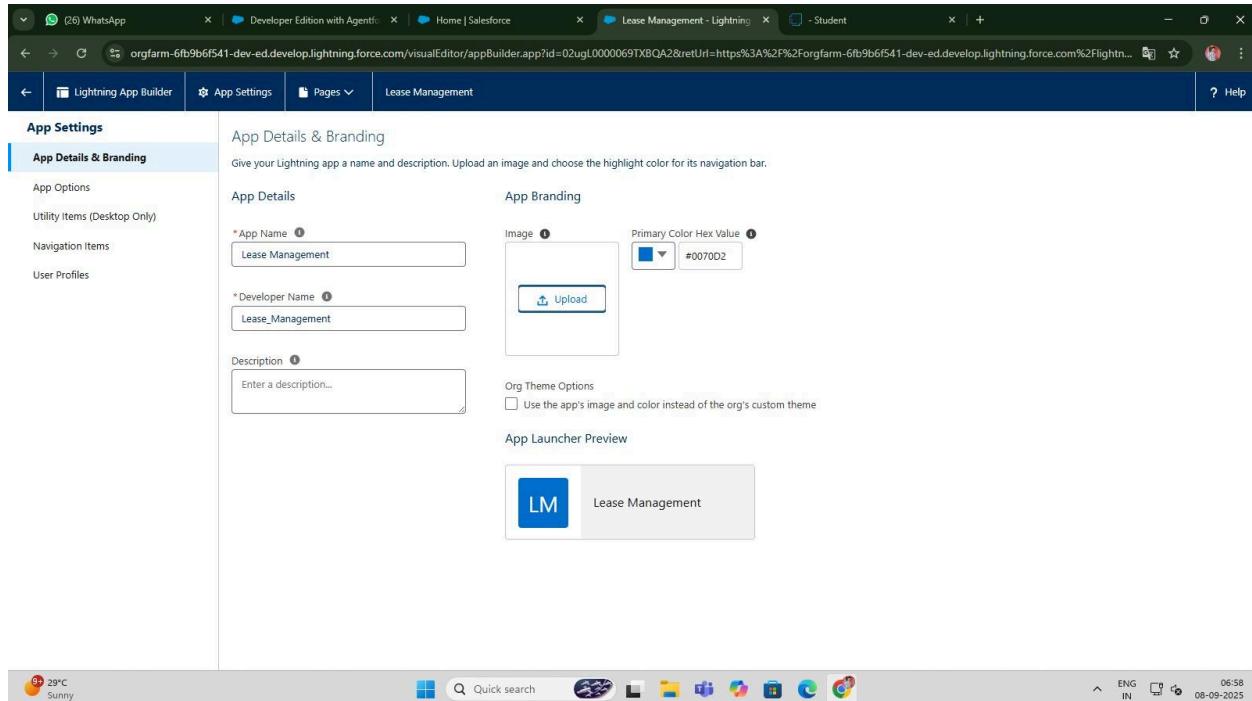
The screenshot shows the Salesforce Setup interface for the 'lease' object. The left sidebar lists various setup categories under 'Fields & Relationships'. The main content area displays the 'Fields & Relationships' table with the following data:

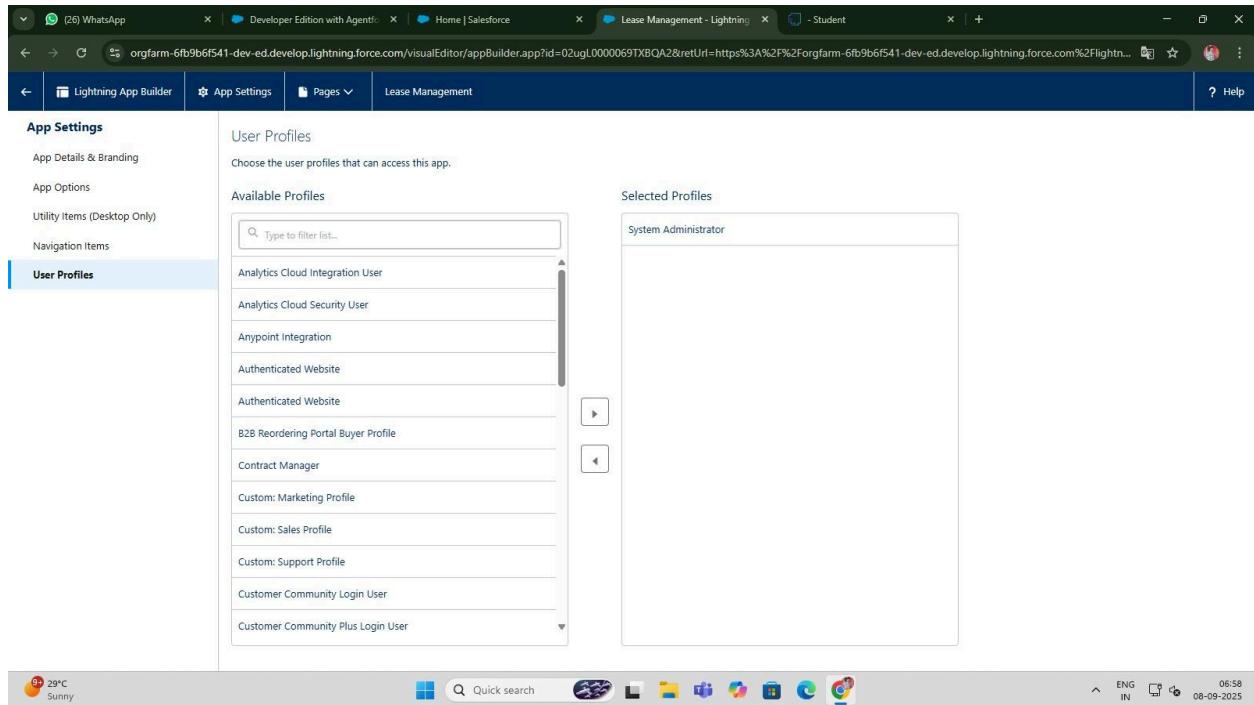
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
End date	End_date__c	Date		
Last Modified By	LastModifiedById	Lookup(User)		
lease Name	Name	Text(80)		✓
Owner	OwnerId	Lookup(User,Group)		✓
property	property__c	Lookup(property)		✓
start date	start_date__c	Date		

The screenshot shows the Salesforce Setup interface for the 'Tenant' object. The left sidebar lists various setup categories under 'Fields & Relationships'. The main content area displays the 'Fields & Relationships' table with the following data:

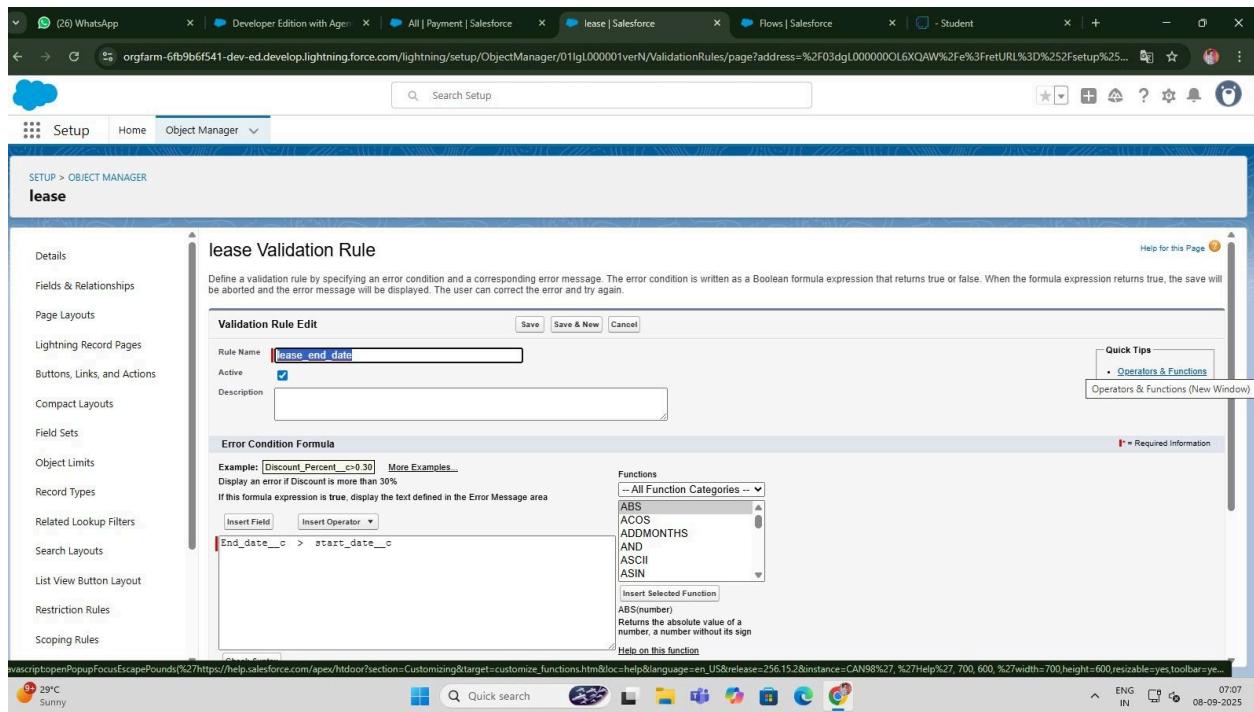
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Email	Email__c	Email		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		
property	property__c	Lookup(property)		✓
status	status__c	Picklist		
Tenant Name	Name	Text(80)		✓

- Developed Lightning App with relevant tabs





- create a validation rule to an Lease Object



- 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 left sidebar is collapsed, and the main area displays the 'Email Template Detail' for a template named 'tenant leaving'. The template is associated with 'Unified Public Classic Email Templates' and has the following details:

Email Template Name	tenant_leaving	Available For Use	✓
Template Unique Name	tenant_leaving	Last Used Date	
Encoding	Unicode (UTF-8)	Times Used	
Author	Krishna Kumar R [Change]		
Description			
Created By	Krishna Kumar R 9/2/2025, 3:13 AM	Modified By	Krishna Kumar R 9/2/2025, 3:13 AM

The 'Email Template' section below shows the template content:

```

Subject : request for approve the leave
Plain Text Preview
Dear {Tenant__c.CreatedBy},

Please approve my leave

```

The system status bar at the bottom indicates it's 07:15 on 08-09-2025.

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The left sidebar is collapsed, and the main area displays the 'Email Template Detail' for a template named 'Leave approved'. The template is associated with 'Unified Public Classic Email Templates' and has the following details:

Email Template Name	Leave_approved	Available For Use	✓
Template Unique Name	Leave_approved	Last Used Date	
Encoding	Unicode (UTF-8)	Times Used	
Author	Krishna Kumar R [Change]		
Description			
Created By	Krishna Kumar R 9/2/2025, 3:17 AM	Modified By	Krishna Kumar R 9/2/2025, 3:17 AM

The 'Email Template' section below shows the template content:

```

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 leave is approved. You can leave now

```

The system status bar at the bottom indicates it's 07:21 on 08-09-2025.

The screenshot shows the Salesforce Setup interface under the 'Email' category, specifically the 'Classic Email Templates' section. A search bar at the top left contains 'email'. The main content area displays the 'Leave rejected' template. The 'Email Template Detail' section shows the following details:

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	Krishna Kumar R (Change)
Description	
Created By	Krishna Kumar R 9/2/2025, 3:24 AM
Modified By	Krishna Kumar R 9/2/2025, 3:24 AM

Below this, the 'Email Template' section shows the preview content:

Subject: Leave rejected  
Plain Text Preview:  
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

At the bottom right of the template preview, there are buttons for 'Send Test and Verify Merge Fields', 'Edit', 'Delete', and 'Clone'.

The system status bar at the bottom indicates: 29°C Mostly sunny, ENG IN 07:21 08-09-2025.

The screenshot shows the Salesforce Setup interface under the 'Email' category, specifically the 'Classic Email Templates' section. A search bar at the top left contains 'email'. The main content area displays the 'Tenant Email' template. The 'Email Template Detail' section shows the following details:

Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Tenant_Email
Template Unique Name	Tenant_Email
Encoding	Unicode (UTF-8)
Author	Krishna Kumar R (Change)
Description	
Created By	Krishna Kumar R 9/2/2025, 3:28 AM
Modified By	Krishna Kumar R 9/2/2025, 3:28 AM

Below this, the 'Email Template' section shows the preview content:

Subject: Urgent: Monthly Rent Payment Reminder  
Plain Text Preview:  
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.

At the bottom right of the template preview, there are buttons for 'Send Test and Verify Merge Fields', 'Edit', 'Delete', and 'Clone'.

The system status bar at the bottom indicates: 29°C Mostly sunny, ENG IN 07:22 08-09-2025.

## ● Approval Process creation

For Check for Vacant:

The screenshot shows the Salesforce Setup interface with the 'Approval Processes' page open. The process name is 'check for vacant'. Key details include:

- Process Name:** check for vacant
- Unique Name:** check\_for\_vacant
- Description:** Tenant: check for vacant
- Entry Criteria:** Tenant: status NOTEQUAL To Leaving
- Record Editability:** Administrator ONLY
- Initial Submitters:** Tenant Owner
- Created By:** Krishna Kumar R, 9/2/2025, 3:49 AM
- Modified By:** Krishna Kumar R, 9/8/2025, 12:18 AM

**Initial Submission Actions:**

Action	Type	Description
Record Lock	Record Lock	Lock the record from being edited
Email Alert	Email Alert	please approve my leave

**Approval Steps:** You have not yet defined any approval steps.

## ● Initial Submission Action

The screenshot shows the classic Salesforce interface with the 'Approval Processes' page open. The process name is 'check for vacant'. Key details include:

- Process Name:** check for vacant
- Unique Name:** check\_for\_vacant
- Description:** Tenant: check for vacant
- Entry Criteria:** Tenant: status NOTEQUAL To Leaving
- Record Editability:** Administrator ONLY
- Initial Submitters:** Tenant Owner
- Created By:** Krishna Kumar R, 9/8/2025, 7:52 AM
- Modified By:** Krishna Kumar R, 9/8/2025, 8:35 PM

**Initial Submission Actions:**

Action	Type	Description
Record Lock	Record Lock	Lock the record from being edited
Email Alert	Email Alert	please approve my leave

**Approval Steps:** You have not yet defined any approval steps.

The screenshot shows the 'Email Alert Detail' page for an alert named 'please approve my leave'. The alert has a description of 'please approve my leave' and a unique name of 'please\_approve\_my\_leave'. It is associated with an email template 'tenant leaving' and an object 'Tenant'. The alert was created by Krishna Kumar R on 9/8/2025, 8:05 AM. The 'Approval Processes Using This Email Alert' section shows a single approval process named 'check for vacant' with type 'Tenant' and state 'Active'. The 'Entitlement Processes Using This Email Alert' section shows no entitlement processes. The 'Flows Using This Email Alert' section shows no flows.

- Final Approval Action

The screenshot shows the 'Email Alert Detail' page for an alert named 'Tenant leaving'. The alert has a description of 'Tenant leaving' and a unique name of 'Tenant\_leaving'. It is associated with an email template 'Leave approved' and an object 'Tenant'. The alert was created by Krishna Kumar R on 9/8/2025, 8:07 AM. The 'Approval Processes Using This Email Alert' section shows a single approval process named 'check for vacant' with type 'Tenant' and state 'Active'. The 'Entitlement Processes Using This Email Alert' section shows no entitlement processes. The 'Flows Using This Email Alert' section shows no flows.

- Final Rejection Action

Salesforce '25

Search... [Search](#) [Switch to Lightning Experience](#) Krishna Kumar R [Setup](#) [Help](#) [Sales](#)

[Home](#) [Chatter](#) [Campaigns](#) [Leads](#) [Accounts](#) [Contacts](#) [Opportunities](#) [Forecasts](#) [Contracts](#) [Orders](#) [Cases](#) [Solutions](#) [+](#) [-](#)

Quick Find / Search... [Edit](#) [Delete](#) [Clone](#)

[Expand All](#) | [Collapse All](#)

**Salesforce Mobile Quick Start**

**Email Alert** [Printable View](#) | [Help for this Page](#) [?](#)

[Rules Using This Email Alert](#) | [Approval Processes Using This Email Alert](#) | [Entitlement Processes Using This Email Alert](#)

**Email Alert Detail** [Edit](#) [Delete](#) [Clone](#)

Description	your request for leave is rejected	Email Template	<a href="#">Leave rejected</a>
Unique Name	<a href="#">your_request_for_leave_is_rejected</a>	Object	Tenant
From Email Address	Current User's email address		
Recipients	Email Field: Email		
Additional Emails			
Created By	<a href="#">Krishna Kumar R</a> 9/8/2025, 8:08 AM	Modified By	<a href="#">Krishna Kumar R</a> 9/8/2025, 8:08 AM
<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Clone</a>			

**Rules Using This Email Alert** [Rules Using This Email Alert Help](#) [?](#)

This alert is currently not used by any rules

**Approval Processes Using This Email Alert** [Approval Processes Using This Email Alert Help](#) [?](#)

Action	Approval Process Name	Description	Type	State
<a href="#">Edit</a>   <a href="#">Del</a>	<a href="#">check_for_vacant</a>		Tenant	Active

**Entitlement Processes Using This Email Alert**

This alert is currently not used by any entitlement processes

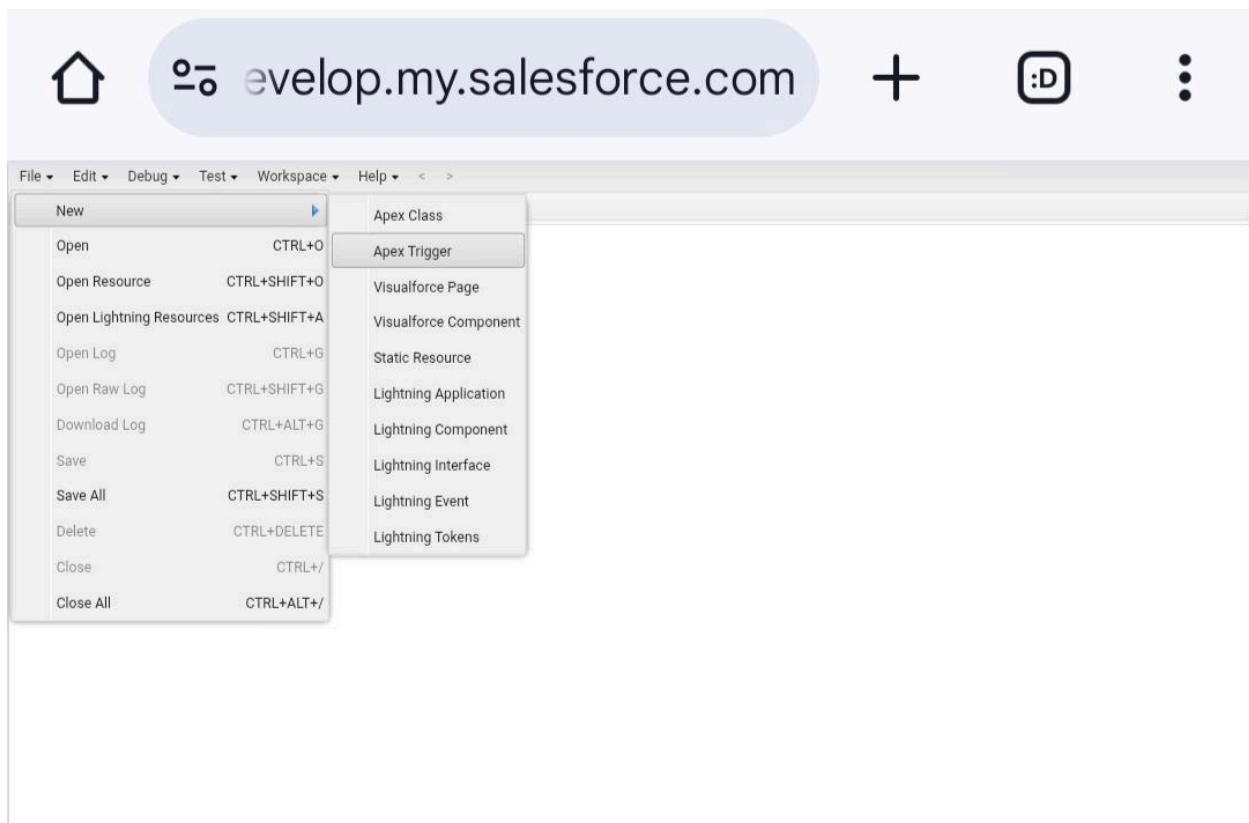
**Flows Using This Email Alert**

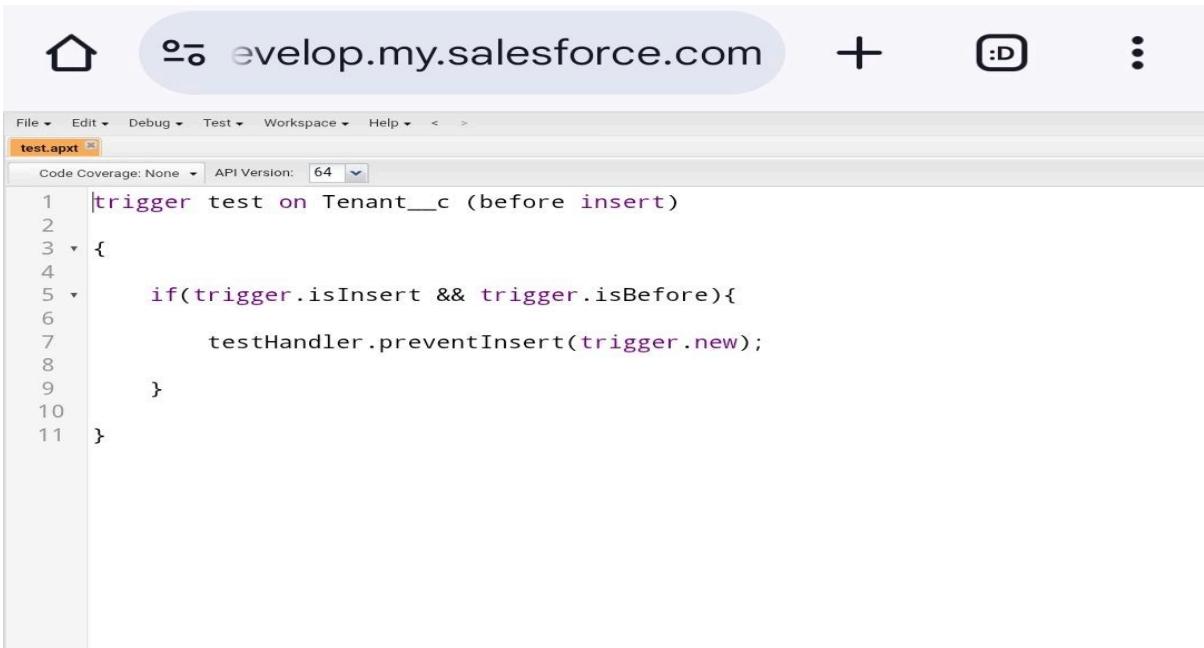
Flow Name	Version	Description	Object	Active
<a href="#">Always show me</a> <a href="#">▼ more records per related list</a>				

[^ Back To Top](#)

# Apex Trigger

- Create an Apex Trigger

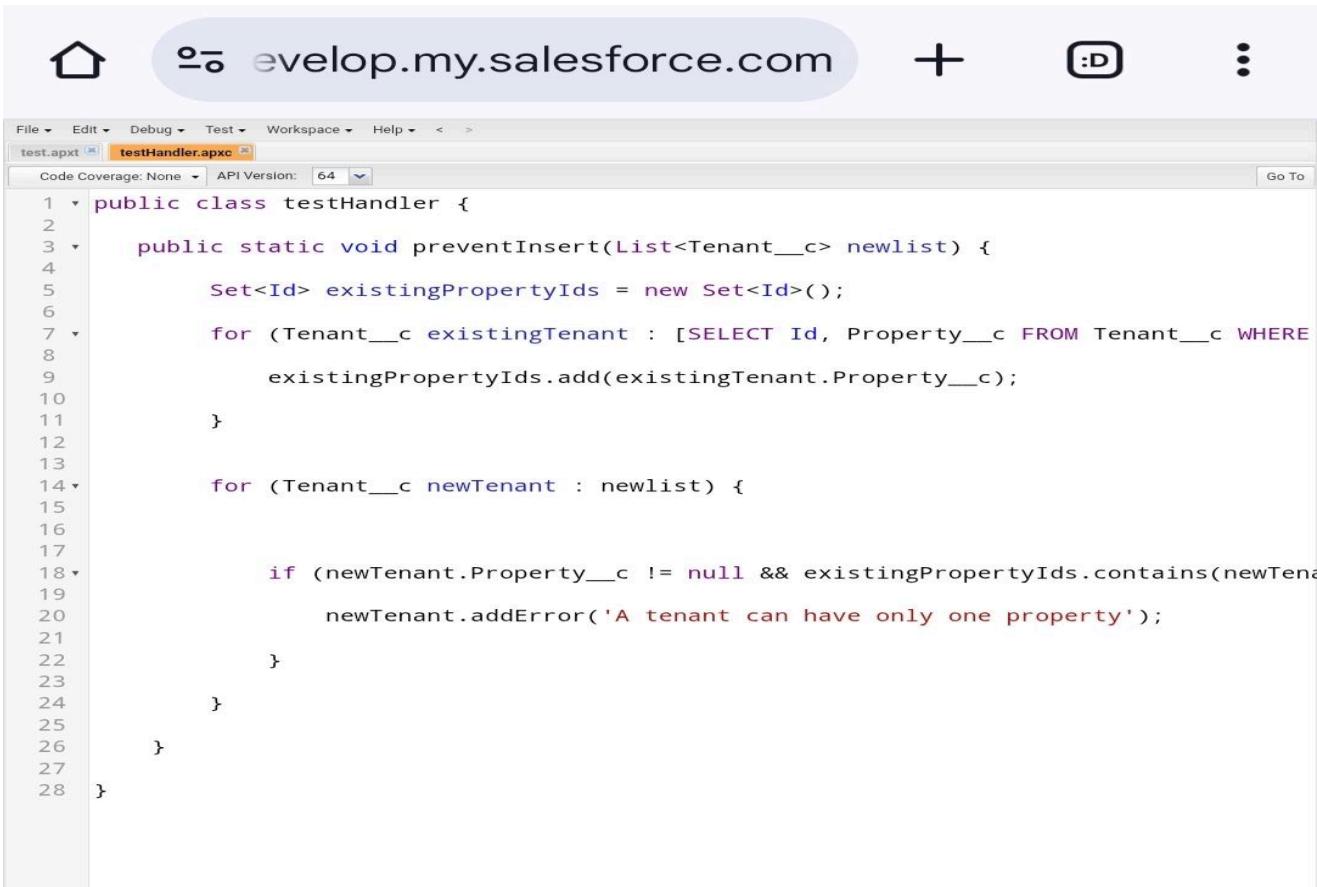




The screenshot shows the Salesforce developer console interface. At the top, there's a navigation bar with icons for Home, a search bar containing 'evelop.my.salesforce.com', and other navigation links like '+', ':D', and three dots. Below the navigation bar is a menu bar with options like File, Edit, Debug, Test, Workspace, Help, and a file browser. The main area is a code editor window titled 'test.apxt'. The code in the editor is:

```
1 trigger test on Tenant__c (before insert)
2 {
3     if(trigger.isInsert && trigger.isBefore){
4         testHandler.preventInsert(trigger.new);
5     }
6 }
```

- Create an Apex Handler class



The screenshot shows the Salesforce developer console interface. At the top, there's a navigation bar with icons for Home, a search bar containing 'evelop.my.salesforce.com', and other navigation links like '+', ':D', and three dots. Below the navigation bar is a menu bar with options like File, Edit, Debug, Test, Workspace, Help, and a file browser. The main area is a code editor window titled 'testHandler.apxc'. The code in the editor is:

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

- Testing the Trigger

Stay

property

Hyd

Created By Krishna Kumar R, 9/2/2025, 5:19 AM

Last Modified By Krishna Kumar R, 9/10/2025, 8:22 AM

No past activity. Past meetings and tasks marked as done show up here.

**New Tenant**

\* = Required Information

**Information**

\* Tenant Name Krishna Kumar R

\* Email krishnakumarrm3058@gmail.com

Phone

Status Stay

**We hit a snag.**

Review the errors on this page.

- A tenant can have only one property

The screenshot shows a software interface for managing tenants. On the left, there's a sidebar with navigation links like 'Stay', 'property', and 'Hyd'. Below that, it shows 'Created By' and 'Last Modified By' information for a specific record. A message at the top right says 'No past activity. Past meetings and tasks marked as done show up here.' In the center, a modal window titled 'New Tenant' is open. It has a section for 'Information' with fields for 'Tenant Name' (set to 'Krishna Kumar R'), 'Email' (set to 'krishnakumarrm3058@gmail.com'), 'Phone' (empty), and 'Status' (set to 'Stay'). A dropdown menu is also visible. A prominent red error message box is centered in the modal, containing the text 'We hit a snag.' and 'Review the errors on this page.' followed by a single bullet point: 'A tenant can have only one property'. At the bottom of the modal are three buttons: 'Cancel', 'Save & New', and 'Save'.

## Create Flow for monthly payment

The screenshot shows the Flow Builder interface with a flow titled "monthly payment - V1". The flow starts with a "Record-Triggered Flow Start" step, set to trigger on "Payment for tenantat" when "A record is updated". It has a condition "Optimize for: Actions and Related Recor..." and a scheduled path "Add Scheduled Paths (Optional)". The flow then branches into two parallel paths. The top path contains a "Run Immediately" step followed by a "send email" action (triggered by a lightning bolt icon). The bottom path contains an "End" step. A modal window titled "Configure Start" is open, showing the configuration for the start step.

**Flow Details:**

- Start Step:** Record-Triggered Flow Start
- Object:** Payment for tenantat
- Trigger:** A record is updated
- Conditions:** 1
- Optimize for:** Actions and Related Records
- Actions:**
  - send email
- End Step:** End

**Configure Start Modal:**

- Select Object:** Payment for tenantat
- Trigger the Flow When:**
  - A record is created
  - A record is updated
  - A record is created or updated
  - A record is deleted
- Set Entry Conditions:**
  - All Conditions Are Met (AND)
- Condition Requirements:**
  - Field: check for payment, Operator: Equals, Value: Paid

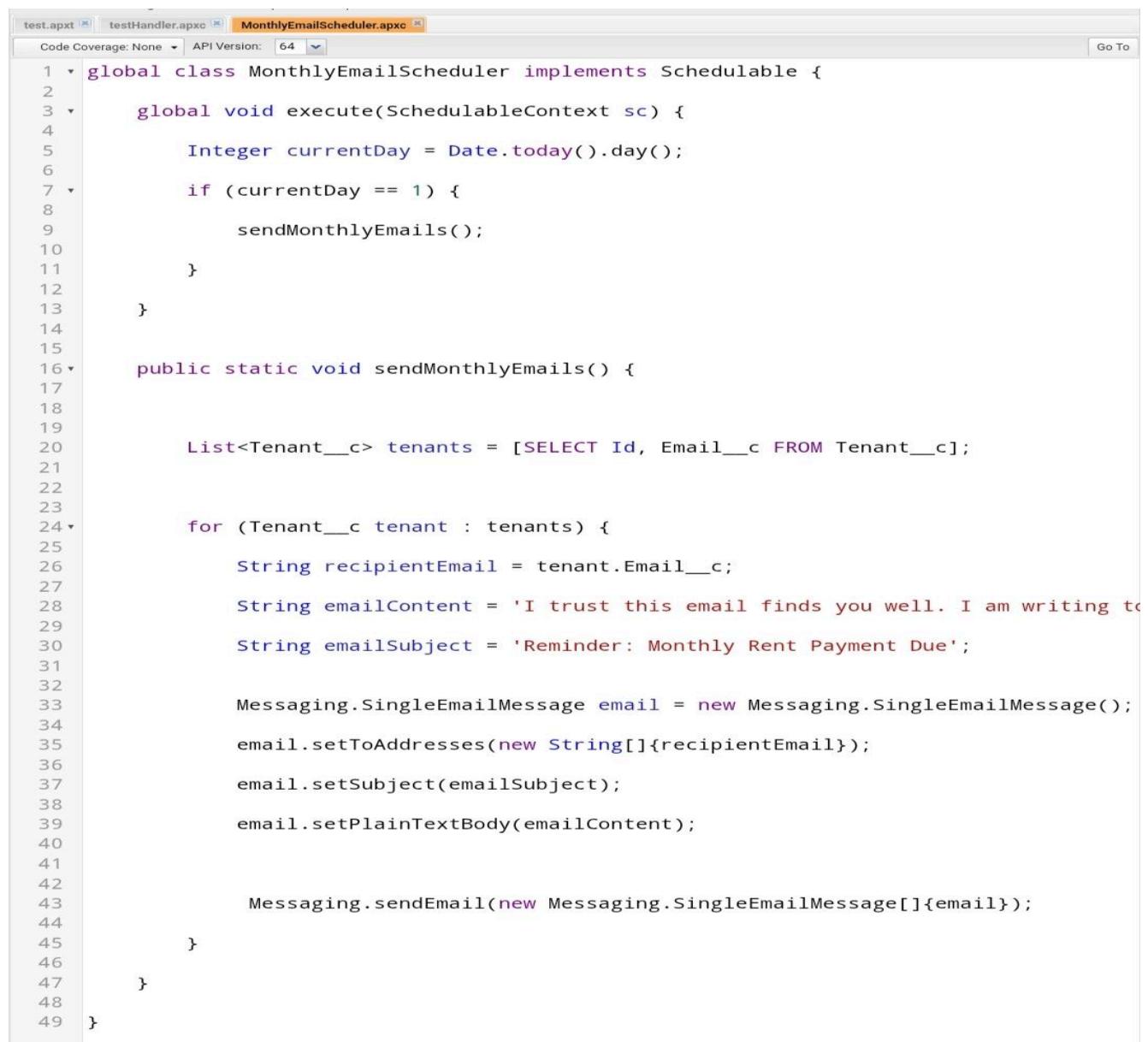
This screenshot shows the same flow "monthly payment - V1" with additional configuration options visible. The "Configure Start" modal is still open, showing the entry conditions and condition requirements. Below the modal, there are several optimization and related records sections.

**Optimization Options:**

- When to Run the Flow for Updated Records:**
  - Every time a record is updated and meets the condition requirements
  - Only when a record is updated to meet the condition requirements
- Optimize Flow:**
  - Fast Field Updates: Update fields on the record that triggers the flow to run. This high-performance flow runs before the record is saved to the database.
  - Actions and Related Records: Update any record and perform actions, like send an email. This more flexible flow runs after the record is saved to the database.
- External Callouts:**
  - Is this flow making an external callout or connecting to an external system?
  - An asynchronous path is required for flows that involve external systems.
  - Add Asynchronous Path: A toggle switch is shown as off.

# Schedule class

- Create an Apex Class



The screenshot shows the Salesforce IDE interface with the MonthlyEmailScheduler.apxc file open. The code implements a scheduled apex job that sends monthly emails to tenants. It uses the Messaging.SingleEmailMessage class to construct and send the email.

```
test.apxt testHandler.apxc MonthlyEmailScheduler.apxc
Code Coverage: None API Version: 64 Go To
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 sendMonthlyEmails() {
17
18
19
20         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
21
22
23         for (Tenant__c tenant : tenants) {
24
25             String recipientEmail = tenant.Email__c;
26
27             String emailContent = 'I trust this email finds you well. I am writing to';
28
29             String emailSubject = 'Reminder: Monthly Rent Payment Due';
30
31
32             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
33
34             email.setToAddresses(new String[]{recipientEmail});
35
36             email.setSubject(emailSubject);
37
38             email.setPlainTextBody(emailContent);
39
40
41
42             Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
43
44         }
45
46     }
47
48 }
49 }
```

# Schedule Apex class

The screenshot shows the Salesforce Apex Classes page. At the top, there's a message about the percentage of Apex used (0.03%) and links to estimate code coverage and compile all classes. Below this, a table lists two Apex classes: 'MonthlyEmailScheduler' and 'testHandler'. Both classes are active, created by Krishna Kumar R on 9/2/2025 at 10:13 AM, and have 64.0 API versions.

Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit   Del   Security	MonthlyEmailScheduler		64.0	Active	1,125	Krishna Kumar R	<input type="checkbox"/>
Edit   Del   Security	testHandler		64.0	Active	584	Krishna Kumar R	<input type="checkbox"/>

**Dynamic Apex Classes**

Dynamic Apex extends your programming reach by interacting with Lightning Platform components.

The screenshot shows the Apex Class Detail page for 'MonthlyEmailScheduler'. It displays the class body with the following Apex code:

```

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     public static void sendMonthlyEmails() {
9         List<Tenant__c> tenants = [SELECT id, Email__c FROM Tenant__c];
10        for (Tenant__c tenant : tenants) {
11            String recipientEmail = tenant.Email__c;
12            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.';
13            String emailSubject = 'Reminder: Monthly Rent Payment Due';
14            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
15            email.setToAddresses(new String[]{recipientEmail});
16            email.setSubject(emailSubject);
17            email.setPlainTextBody(emailContent);
18            Messaging.SingleEmailMessage[] emails = new Messaging.SingleEmailMessage[]{email};
19            Messaging.sendEmail(emails);
20        }
21    }
22}
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

```

- Testing the approval process

The screenshot shows a CRM application for 'Lease Management'. The main view displays a tenant record for 'Krishna'. The 'Details' tab is selected, showing fields like Tenant Name, Email, Phone, Status, and Property. The 'Owner' field shows 'Krishna Kumar R.' with a small photo icon. Below the details are 'Created By' and 'Last Modified By' sections. To the right is an 'Activity' section showing no upcoming or overdue activities. A context menu is open on the right side, with the 'New Case' option highlighted.

This screenshot shows the same CRM interface as above, but with a modal dialog box titled 'Submit for Approval' overlaid. The dialog contains a text area labeled 'Comments' with the text 'Leaving' entered. At the bottom are 'Cancel' and 'Submit' buttons. The background activity and tenant details are visible but dimmed.

**Tenant was submitted for approval.**

**Related** **Details**

Tenant Name: Krishna  
Owner: Krishna Kumar R.  
Email: krishnakumarrm3058@gmail.com  
Phone: (730) 632-6064  
status: Stay  
property: Hyd  
Created By: Krishna Kumar R., 9/2/2025, 5:19 AM

Last Modified By: Krishna Kumar R., 9/10/2025, 8:22 AM

**Activity**

No activities to show. Get started by sending an email, scheduling a task, and more.

No past activity. Past meetings and tasks marked as done show up here.

**Notifications** **Mark all as read**

Krishna Kumar R is requesting approval for tenant  
Tenant Name: Krishna • Owner: Krishna Kumar R.  
a few seconds ago •

Approval request for the tenant is approved  
Krishna  
Sep 9, 2025, 3:19 PM •

New Guidance Center learning resource available  
Define Your Sales Process  
Learn how to guide reps through the sales process.  
Sep 2, 2025, 5:41 PM •

New Guidance Center learning resource available  
Set Up Accounts & Contacts  
Start storing information about your customers with accounts and contacts.

Lease Management

Approval Request  
Tenant Approval Pending

Submitter Krishna Kumar R	Date Submitted Sep 10, 2025	Actual Approver Krishna Kumar R	Assigned To Krishna Kumar R
------------------------------	--------------------------------	------------------------------------	--------------------------------

**Details**

Approval Details

Tenant Name Krishna	Owner Krishna Kumar R
------------------------	--------------------------

**Submitter Comments**

Krishna Kumar R  
Leaving  
Sep 10, 2025, 9:03:29 AM

Approvals

Tenants

Tenant Krishna

**Related**

- Payments (0)
- Payment (0)
- Approval History (6)

Step Name	Date	Status	Assigned To
Approved	9/10/2025, ...	Approved	Krishna Ku...
Approval ...	9/10/2025, ...	Submitted	Krishna Ku...
Approved	9/9/2025, 2:...:	Approved	Krishna Ku...
Approval ...	9/9/2025, 2:...:	Submitted	Krishna Ku...
Approved	9/8/2025, 1...	Approved	Krishna Ku...
Approval ...	9/8/2025, 8:...:	Submitted	Krishna Ku...

**Activity**

No activities

No past activity. Past meetings and ta

# RESULTS

## Output Screenshots

### Tabs for Property, Tenant, Lease, Payment

The screenshot shows the Salesforce Setup interface under the 'Customize' tab. On the left, there's a sidebar with sections like Home, Chatter, Campaigns, Leads, Accounts, Contacts, Opportunities, Forecasts, Contracts, Orders, Cases, Solutions, and Administer. Under Administer, there are many sub-options such as Manage Users, Manage Apps, Manage Territories, Company Profile, Data Classification, Privacy Center, Security Controls, Domain Management, Communication Templates, Translation Workbench, Data Management, Mobile Administration, Desktop Administration, Outlook Integration and Sync, Gmail Integration and Sync, Email Administration, Google Apps, Analytics, Tableau, and Data.com Administration. The main content area is titled 'Custom Tabs' and contains sections for 'Custom Object Tabs', 'Web Tabs', 'Visualforce Tabs', 'Lightning Component Tabs', and 'Lightning Page Tabs'. Each section has a 'New' button and a 'What Is This?' link. Under 'Custom Object Tabs', there are four entries: 'Lease' (Tab Style: Books), 'Payment' (Tab Style: Books), 'Properties' (Tab Style: Bottle), and 'Tenants' (Tab Style: Books). The 'Description' column is empty for all.

The screenshot shows the 'Lease Management' application. At the top, there's a navigation bar with icons for Home, properties, Payment, Tenants, lease, \*Tenant\_c, and a search bar. Below the navigation bar, the title is 'Tenants > Krishna Approval History'. The main content area displays a table titled 'Approval History' with 6 items. The table has columns for Step Name, Date, Status, Assigned Approver, Actual Approver, and Comments. The data is as follows:

Step Name	Date	Status	Assigned Approver	Actual Approver	Comments
1	Approved 9/10/2025, 9:04 AM	Approved	Krishna Ku...	Krishna Kumar R	Approved
2	Approval Re... 9/10/2025, 9:03 AM	Submitted	Krishna Ku...	Krishna Kumar R	Leaving
3	Approved 9/9/2025, 2:49 AM	Approved	Krishna Ku...	Krishna Kumar R	Approve tenant
4	Approval Re... 9/9/2025, 2:44 AM	Submitted	Krishna Ku...	Krishna Kumar R	Leaving
5	Approved 9/8/2025, 11:21 PM	Approved	Krishna Ku...	Krishna Kumar R	Leaving
6	Approval Re... 9/8/2025, 8:39 PM	Submitted	Krishna Ku...	Krishna Kumar R	Leaving

# ADVANTAGES & DISADVANTAGES

## Advantages of Lease Management System

**Centralized Data Management** – All tenant, property, and lease records are stored in one place, reducing duplication and errors.

**Automation of Processes** – Workflows, approvals, and reminders minimize manual intervention, saving time and effort.

**Improved Accuracy** – Automated tracking of payments and contracts reduces chances of mistakes and missed deadlines.

**Better Communication** – Email alerts and notifications keep tenants and property managers informed in real time.

**Enhanced Compliance** – Role-based access ensures data security and adherence to legal requirements.

**Data Analytics & Reporting** – Dashboards provide insights into occupancy rates, payment status, and lease renewals.

**Scalability** – Can be extended to manage multiple properties and tenants as the business grows.

**Tenant Satisfaction** – Timely updates and smoother processes improve the overall tenant experience.

## Disadvantages of Lease Management System

**High Initial Setup Cost** – Implementing Salesforce and customizing it for lease management may require investment.

**Training Requirement** – Staff need to be trained to use Salesforce effectively, which can take time

**Dependency on Internet** – Being a cloud-based solution, it requires stable internet connectivity.

**Customization Complexity** – Advanced customizations may need skilled Salesforce developers, adding to costs.

**Data Migration Challenges** – Transferring existing lease and tenant records into the system can be time-consuming.

**Subscription Costs** – Ongoing Salesforce licensing fees may be expensive for smaller organizations

## CONCLUSION

The Lease Management System streamlines leasing operations by centralizing tenant data, contracts, payments, and communication within Salesforce. It enhances accuracy, reduces manual effort, and ensures timely rent collection and contract renewals through automation. With features like Flows, Approval Processes, and Email Alerts, the system improves efficiency and communication, while dashboards provide real-time insights for decision-making. Overall, it boosts productivity, tenant satisfaction, and compliance, offering a scalable foundation for future growth.

## 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)) {
```

```
newTenant.addError('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});  
}  
}  
}
```