

# **LEASE MANAGEMENT**

**College Name: KPR College of Arts Science and Research**

**College Code: bruaz**

**TEAM ID: NM2025TMID21504**

**TEAM MEMBERS:**

**Team Leader Name: PRASANTH L**

**Email: 23bai037@kprcas.ac.in**

**Team Member 1: YUVARAJ R**

**Email: 23bai060@kprcas.ac.in**

**Team Member 2: JOSHNA ASHIKA M**

**Email: 23bai018@kprcas.ac.in**

**Team Member 3: JOVITA J**

**Email: 23bai019@kprcas.ac.in**

# 1.INTRODUCTION

## 1.1 Project Overview

The Lease Management System is a Salesforce-based application designed to streamline the processes associated with leasing real estate properties. It handles tenant management, lease contracts, payments, and communication with automation features such as flows, approval processes, and email alerts.



## 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>

**Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud.**

Sign up for your Developer Edition.

- ✓ Build apps fast with drag and drop tools
- ✓ Go further with Apex code
- ✓ Build AI agents with Agentforce
- ✓ Harmonize your data with Data Cloud
- ✓ Ground Agentforce with structured and unstructured data
- ✓ Integrate with anything using APIs

**Sign up for your Developer Edition**

A free Salesforce Platform environment with Agentforce and Data Cloud.

First name: PRASANTH ✓ Last name: L ✓

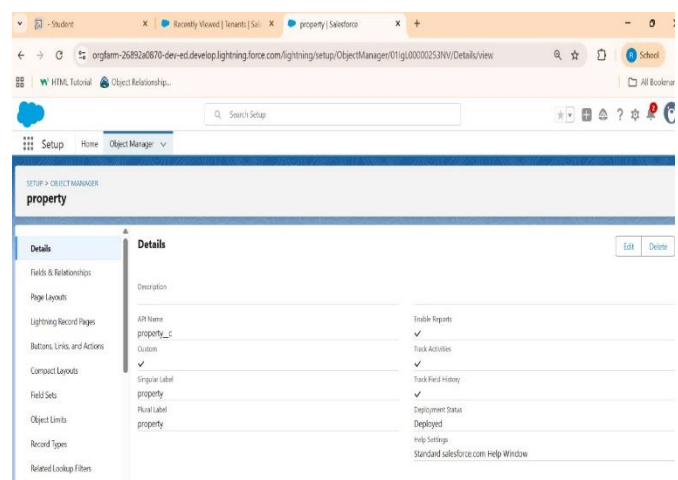
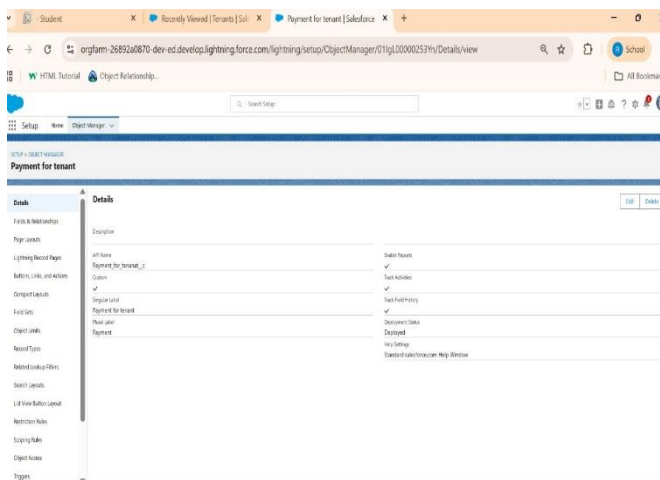
Job title: DEVELOPER ✓ Work email: 23ba1037@kprcas.ac ✓

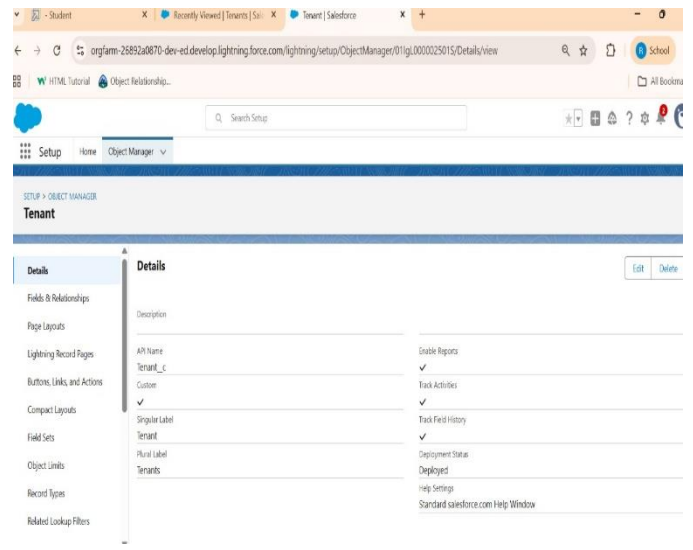
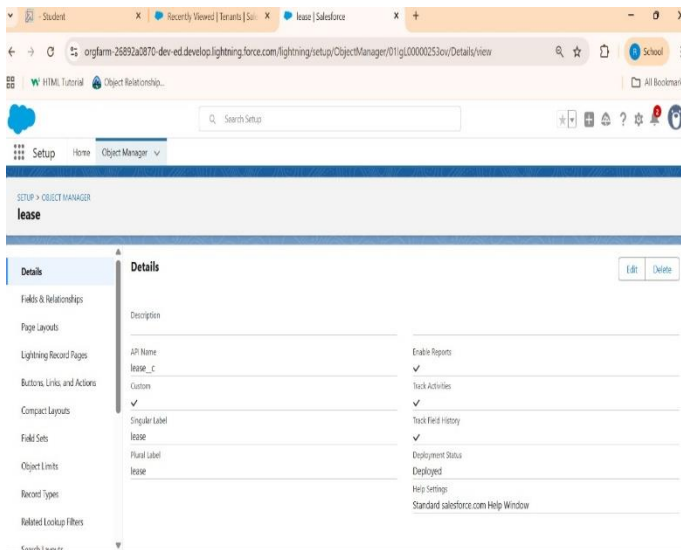
Company: KPRCAS ✓ Country/Region: India ✓

☒ I agree to the Main Services Agreement – Developer Services and Subscription Program Agreement. To acknowledge, see described in the Developer Documentation. (2) the Developer Edition includes autonomous and other generative AI features, and (3) Salesforce may limit use of these features and the org, and may terminate any org that has been inactive for 45 days.

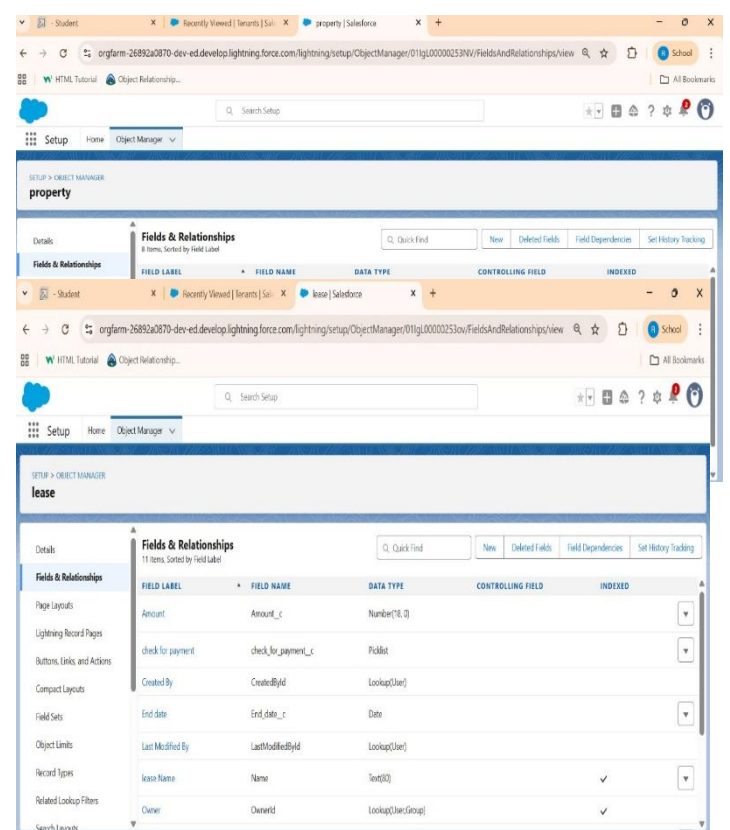
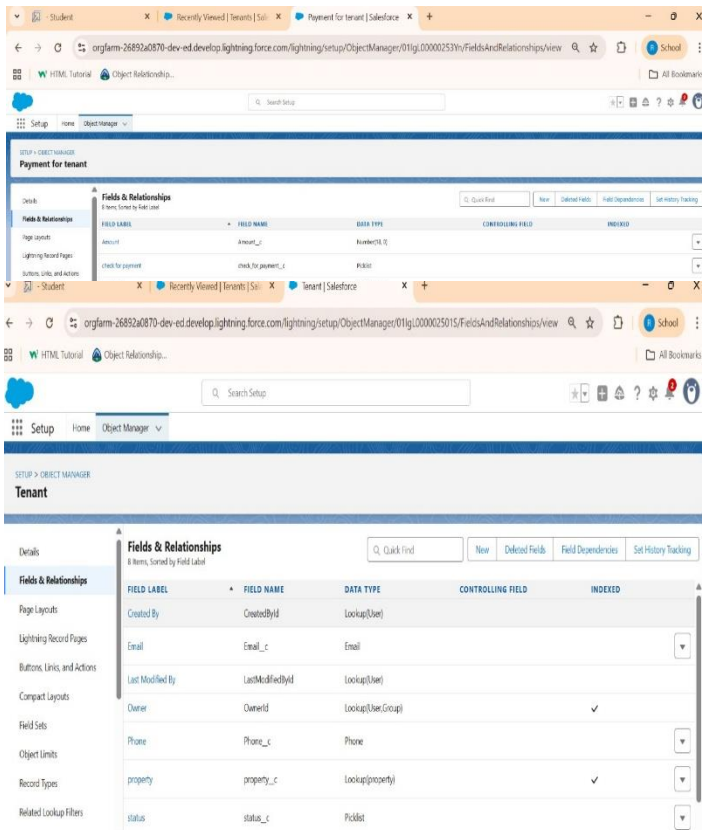
We value your privacy. To learn more, visit our Privacy Statement.

➤ Created objects: Property, Tenant, Lease, Payment





## ➤ Configured fields and relationships



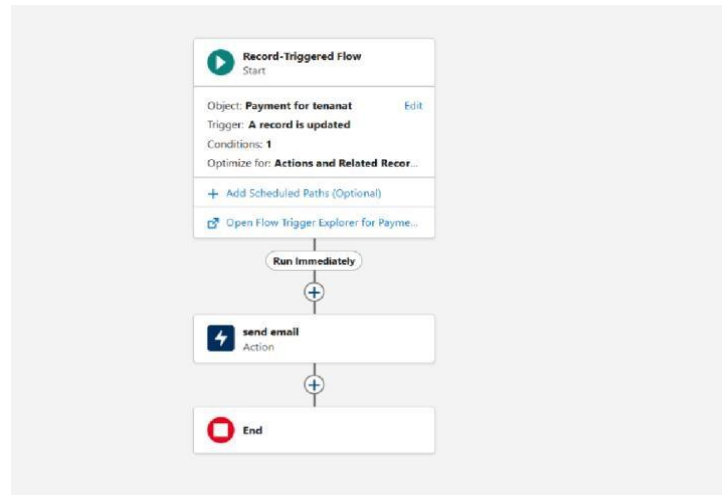
➤ Developed Lightning App with relevant tabs

The screenshot shows the 'App Details & Branding' configuration screen in the Lightning App Builder. The left sidebar has 'App Settings' selected, with 'App Details & Branding' as the active sub-tab. The main area is divided into two sections: 'App Details' and 'App Branding'. In 'App Details', the 'App Name' is 'Lease Management', the 'Developer Name' is 'Lease\_Management', and the 'Description' is 'Application to efficiently handle the processes related to leasing real-estate properties.'. In 'App Branding', an image is uploaded, and the 'Primary Color Hex Value' is '#0070D2'. Below this, there's an 'Org Theme Options' checkbox (unchecked) and an 'App Launcher Preview' showing the app's icon and name.

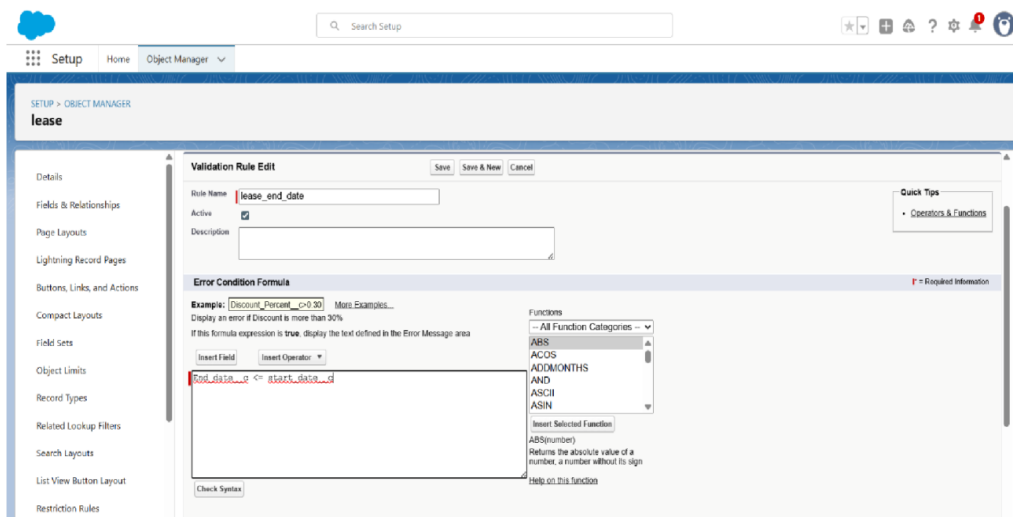
The screenshot shows the 'Navigation Items' configuration screen. The left sidebar has 'App Settings' selected, with 'Navigation Items' as the active sub-tab. The main area is divided into 'Available Items' and 'Selected Items'. The 'Available Items' list includes Accounts, Activation Targets, Activations, All Sites, Alternative Payment Methods, Analytics, App Launcher, Appointment Categories, Appointment Invitations, and Approval Requests. The 'Selected Items' list includes Payment, Tenants, property, and lease. Arrows are used to move items between the two lists.

The screenshot shows the Lightning App running in a web browser. The browser address bar shows the URL 'orgfarm-25692a0870-dev-ed.develop.lightning.force.com/lightning/o/property\_c/list?filterName=\_\_Recent'. The app's navigation bar shows 'Lease Management' as the active tab, with other tabs like 'Tenants', 'property', 'Payment', and 'lease' visible. The main content area shows a 'Recently Viewed' list with one item, 'property Name', which has a value of 'raksha'.

- Implemented Flows for monthly rent and payment success



- To create a validation rule to a Lease Object



- Added Apex trigger to restrict multiple tenants per property

Lease Management

Tenants  
All

0 items • Sorted by Tenant Name •

**New Tenant**

\* = Required Information

Information

\* Tenant Name  
BHARATH

\* Email  
23bai009@kprcas.ac.in

Phone  
6383846733

status  
Stay

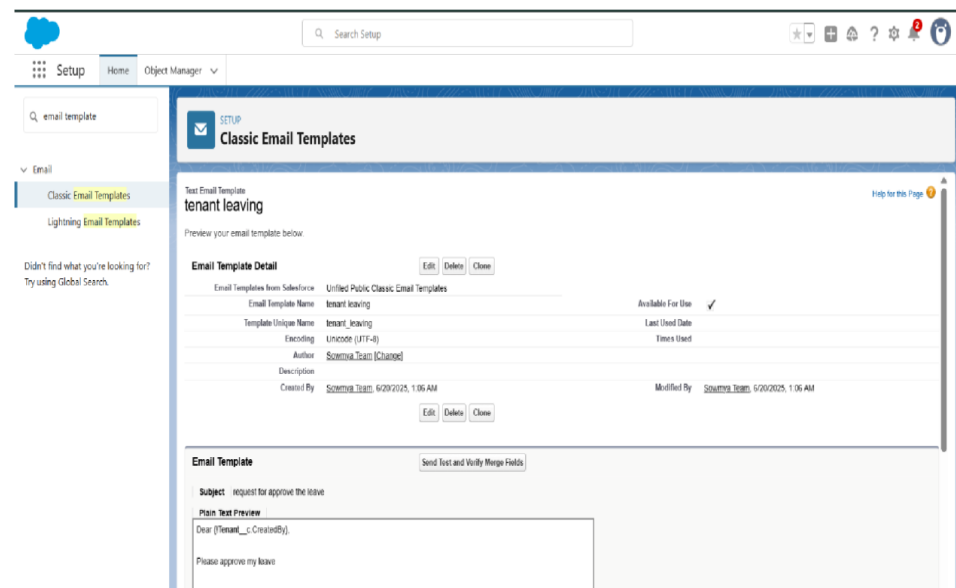
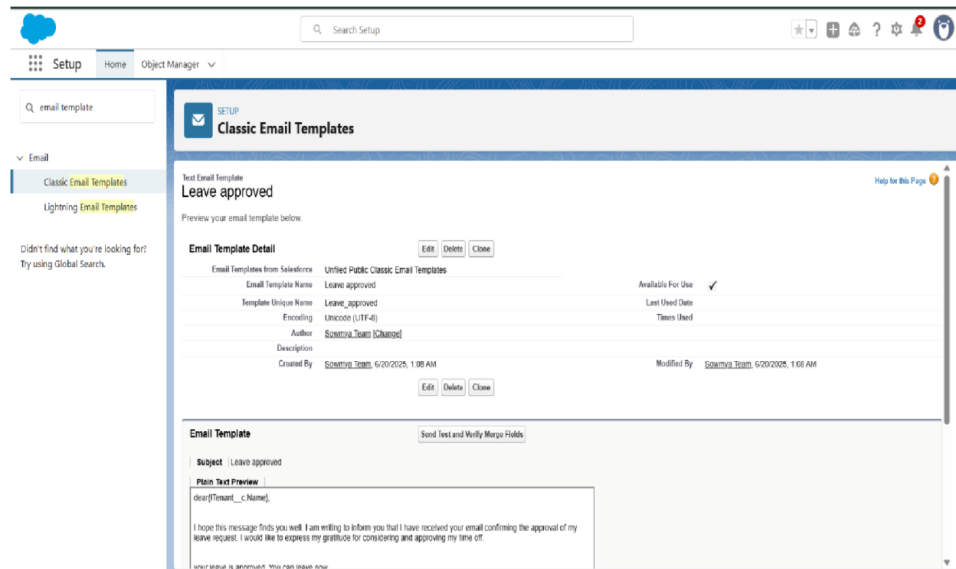
Owner  
BHARATH KUMAR

Cancel Save & New Save

- Scheduled monthly reminder emails using 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  
17     public static void sendMonthlyEmails() {  
18  
19         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];  
20  
21         for (Tenant__c tenant : tenants) {  
22  
23             String recipientEmail = tenant.Email__c;  
24  
25             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  
26  
27             String emailSubject = 'Reminder: Monthly Rent Payment Due';  
28  
29             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();  
30  
31             email.setToAddresses(new String[]{recipientEmail});  
32  
33             email.setSubject(emailSubject);  
34  
35             email.setPlainTextBody(emailContent);  
36  
37         }  
38     }  
39 }  
40
```

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





[illegible]

The screenshot displays the Salesforce Classic Email Templates interface. The left sidebar contains the navigation menu with 'Setup', 'Home', and 'Object Manager'. The main content area is titled 'Classic Email Templates' and shows a preview of an email template named 'tenant payment'. The preview includes fields for 'Email Template Detail', 'Email Template Name', 'Template Unique Name', 'Encoding', 'Author', 'Description', 'Created By', and 'Modified By'. The template content is visible in the bottom section, showing a subject line and a plain text preview.

**Approval Processes**

Tenant: TenantApproval

Process Definition Detail

Process Name: TenantApproval

Unique Name: TenantApproval

Description: TenantApproval

Entry Criteria: TENANT.L STATUS EQUALS Stay

Record Editability: Administrator ONLY

Approval Assignment Email Template: Tenant Owner

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

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Step 1	1	Step 1			User: Sowmya Team	Final Rejection

Approval Process creation

For Check for Vacant:

**Approval Processes**

Tenant: check for vacant

Process Definition Detail

Process Name: check for vacant

Unique Name: check\_for\_vacant

Description: check for vacant

Entry Criteria: TENANT.L STATUS EQUALS LEAVING

Record Editability: Administrator ONLY

Approval Assignment Email Template: Leave Approval

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

Email Alert: Please approve the leave

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
step1	1	step1			User: Sowmya Team	Final Rejection

**New Tenant**

Information

\* Tenant Name: BHARATH

\* Email: 23bai009@kprcas.ac.in

Phone: 6383846733

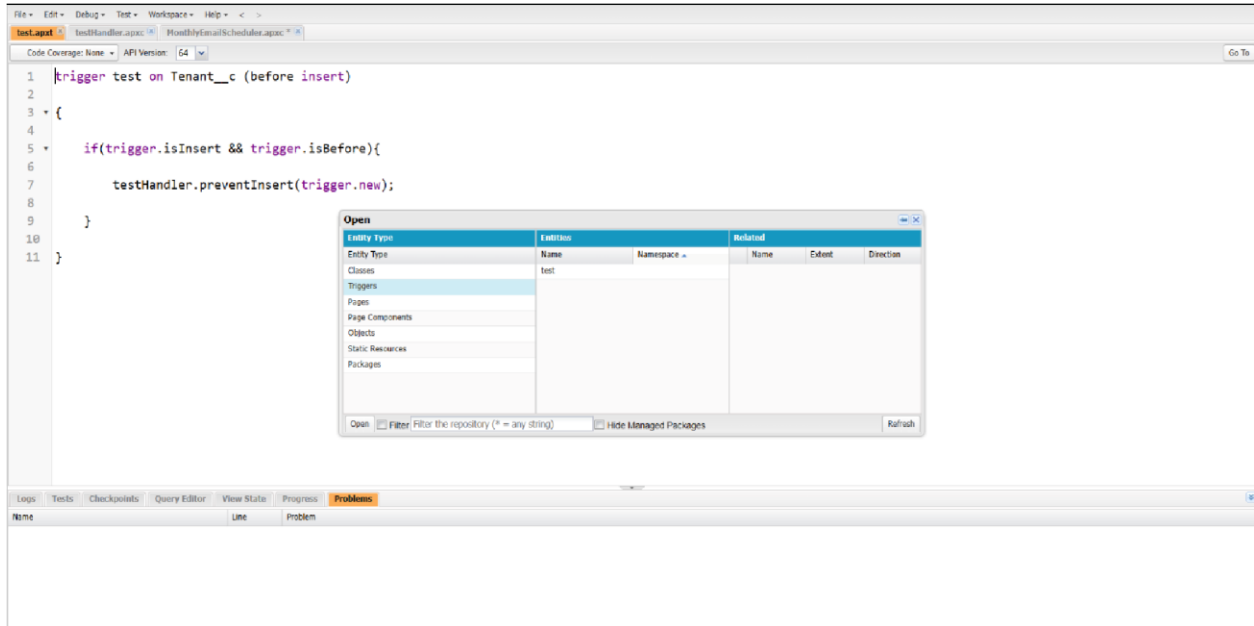
status: Stay

Owner: BHARATH KUMAR

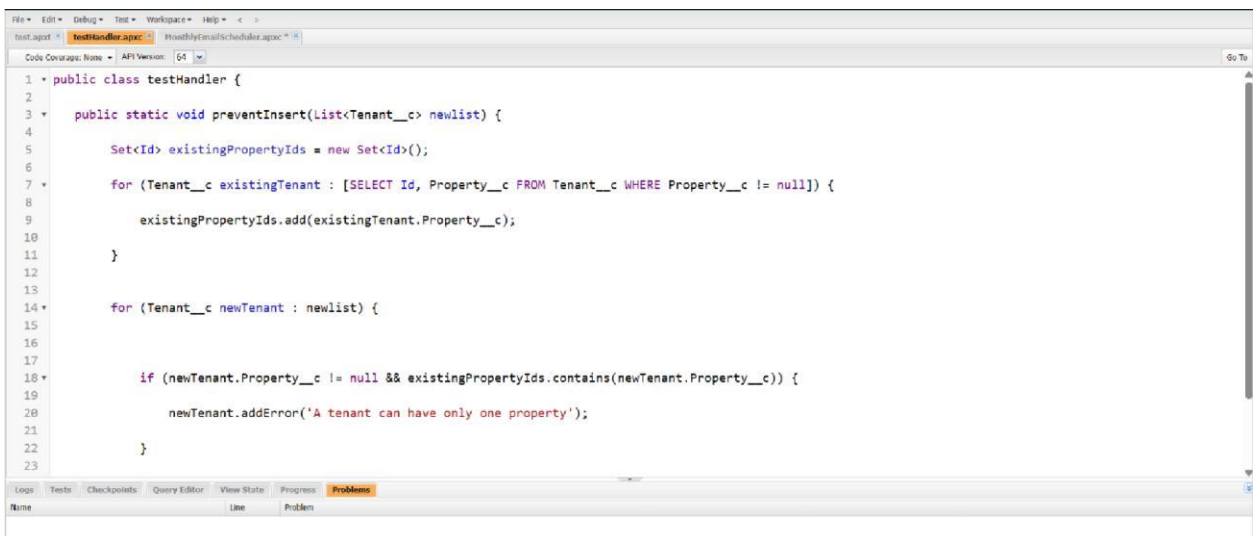
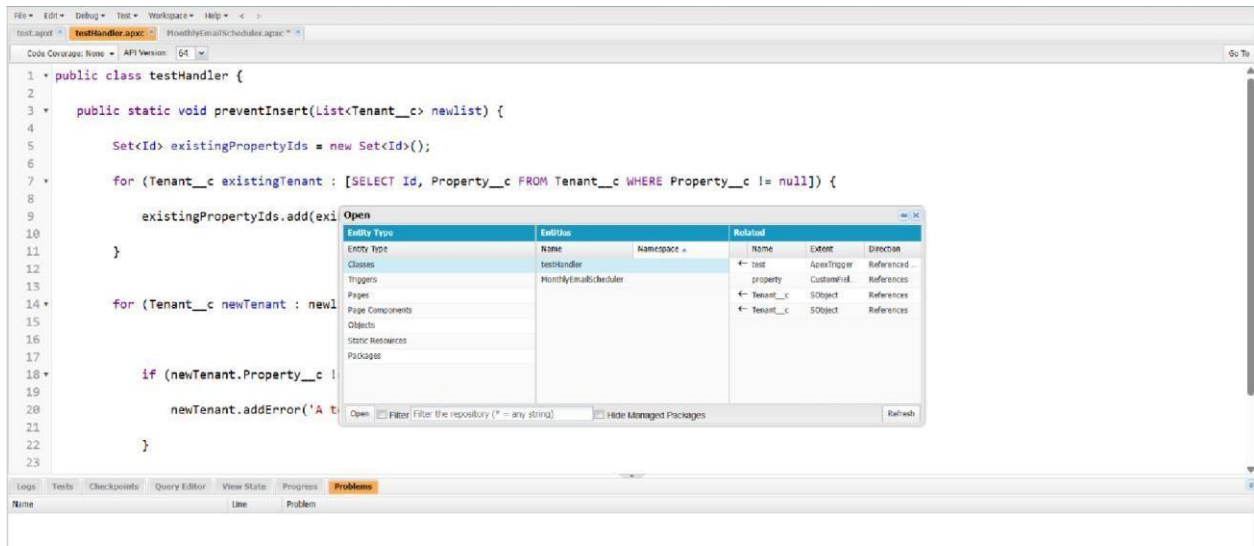
Cancel Save & New Save

# Apex Trigger

## Create an Apex Trigger



## Create an Apex Handler class



# FLOW

Flow Builder

monthly payment - V1

?

Select Elements

Auto-Layout

Last saved on 6/21/2025, 04:20 PM

Active

Run

Debug

View Tests

Save As New Version

Save

Deactivate

Record-Triggered Flow

Start

Object: **Payment for tenanat**

Trigger: **A record is updated**

Conditions: 1

Optimize for: **Actions and Related Recor...**

+ Add Scheduled Paths (Optional)

Open Flow Trigger Explorer for Payme...

Run Immediately

+ send email

End

Configure Start

Field: check for payment

Operator: Equals

Value: Paid

+ Add Condition

When to Run the Flow for Updated Records

☒ Every time a record is updated and meets the condition requirements

☐ Only when a record is updated to meet the condition requirements

Optimize Flow

Optimize the Flow for:

**Fast Field Updates**

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

**Actions and Related Records**

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

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

Flow Builder

monthly payment - V1

?

Select Elements

Auto-Layout

Last saved on 6/21/2025, 04:20 PM

Active

Run

Debug

View Tests

Save As New Version

Save

Deactivate

Record-Triggered Flow

Start

Object: **Payment for tenanat**

Trigger: **A record is updated**

Conditions: 1

Optimize for: **Actions and Related Recor...**

+ Add Scheduled Paths (Optional)

Open Flow Trigger Explorer for Payme...

Run Immediately

+ send email

End

Configure Start

Select Object

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

\* Object

Payment for **tenanat**

Configure Trigger

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

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

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

Condition Requirements

All Conditions Are Met (AND)

➤ Schedule class:

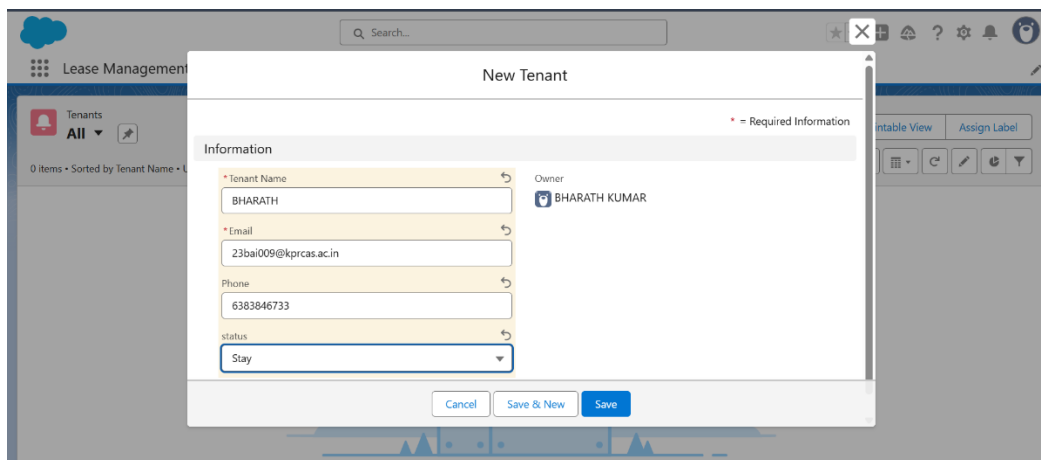
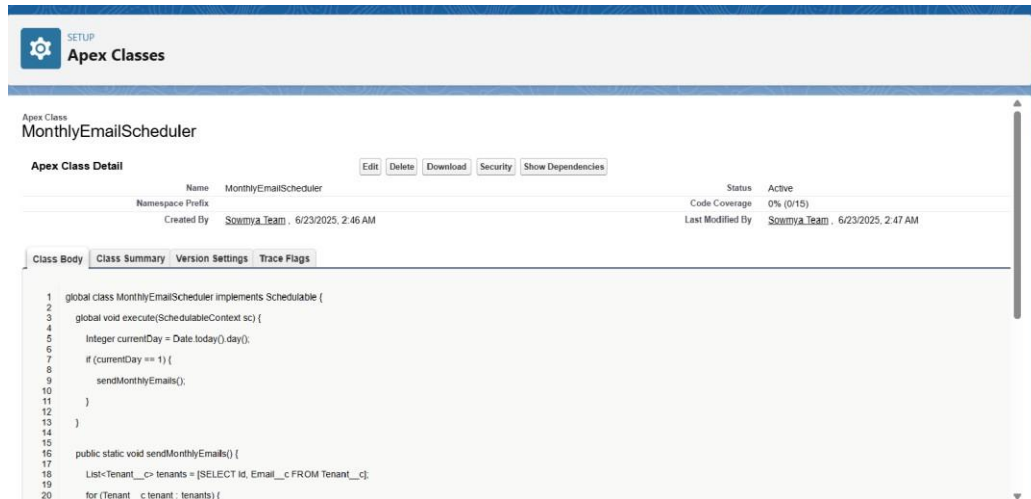
## Create an Apex Class

The screenshot displays the Salesforce IDE interface. The main editor window shows the code for the `MonthlyEmailScheduler` class, which implements the `Schedulable` interface. The code includes a `execute` method that checks the current day and sends emails to tenants. An `Open` dialog box is open, showing the class hierarchy and related entities. The `Related` tab is selected, showing the relationship between the `MonthlyEmailScheduler` class and the `Tenant__c` object.

```
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    public static void sendMonthlyEmail() {
16
17        List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18
19        for (Tenant__c tenant : tenants) {
20
21            String recipientEmail = tenant.Email__c;
22
23            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
24
25            String emailSubject = 'Reminder: Monthly Rent Payment Due';
26
27            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
28
29            email.setToAddresses(new String[] {recipientEmail});
30
31            email.setSubject(emailSubject);
32
33            email.setPlainTextBody(emailContent);
34
35            Messaging.sendEmail(new Messaging.SingleEmailMessage([email]));
36
37        }
38    }
39
40 }
41
42 }
```

Entity Type	Entity Name	Related Name	Extent	Direction
Entity Type	testrunner	testrunner	CronTrigger	References
Classes	MonthlyEmailScheduler	Email	CustomField	References
Triggers		Tenant__c	Object	References
Pages		Tenant__c	Object	References
Page Components				
Objects				
Static Resources				
Packages				

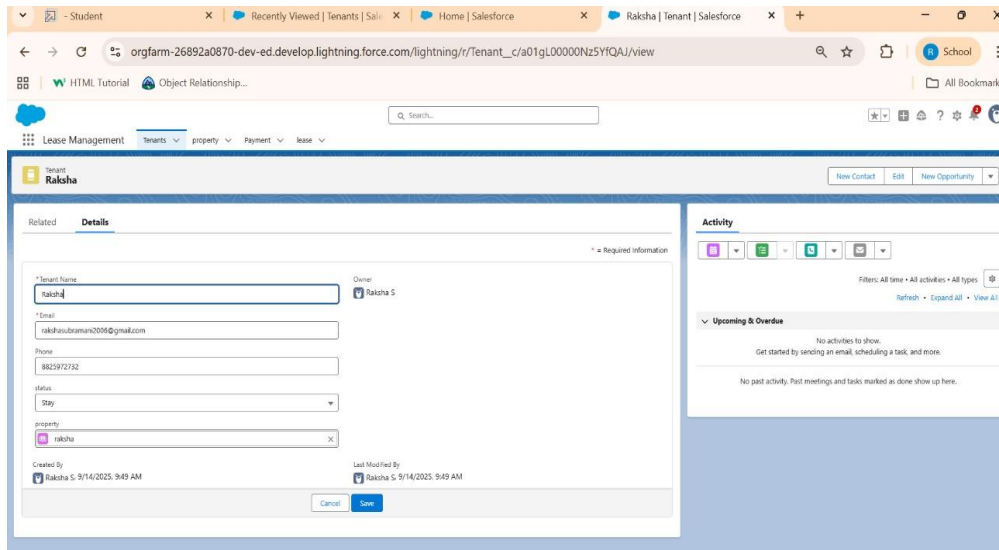
## Schedule Apex class



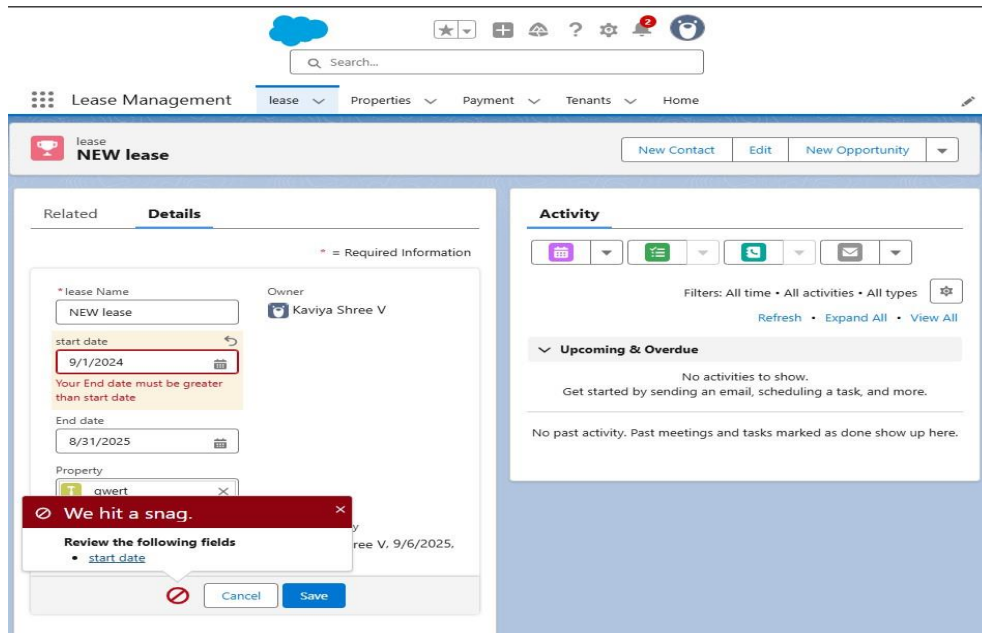
## FUNCTIONAL AND PERFORMANCE TESTING

### Performance Testing

- Trigger validation by entering duplicate tenant-property records

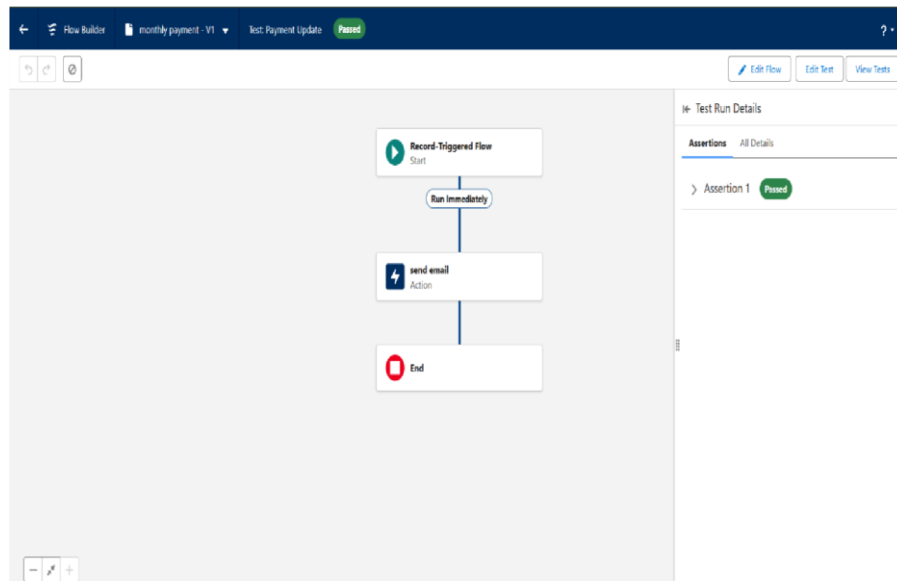
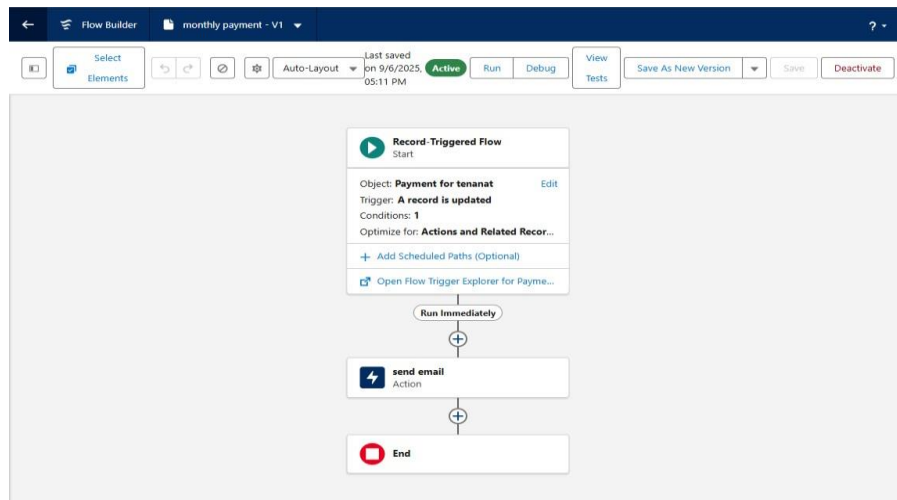


## ➤ Validation Rule checking





➤ Test flows on payment update



- Approval process validated through email alerts and status updates

Lease Management

Tenants  
All

0 items • Sorted by Tenant Name •

Search...

New Tenant

\* = Required Information

Information

\* Tenant Name  
BHARATH

\* Email  
23bai009@kprcas.ac.in

Phone  
6383846733

Status  
Stay

Owner  
BHARATH KUMAR

Cancel Save & New Save

## RESULTS

### Output Screenshots

- Tabs for Property, Tenant, Lease, Payment

Setup

Home Object Manager

Search Setup

Setup Tabs

Custom Tabs

You can create new custom tabs to extend Salesforce functionality or to build new application functionality.

Custom Object tabs look and behave like the standard tabs provided with Salesforce. Web tabs allow you to embed external web applications and content within the Salesforce window. Visualforce tabs allow you to embed Visualforce pages. Lightning Component tabs allow you to add Lightning components to the navigation menu in Lightning Experience and the mobile app. Lightning Page tabs allow you to add Lightning Pages to Lightning Experience and the mobile app.

Action	Label	Tab Style	Description
Edit   Del	Issue	Trophy	
Edit   Del	Payment	Calendar	
Edit   Del	Properties	House	
Edit   Del	Tenants	Scenes	

Web Tabs

No Web Tabs have been defined

Visualforce Tabs

No Visualforce Tabs have been defined

Lightning Component Tabs

No Lightning component tabs have been defined

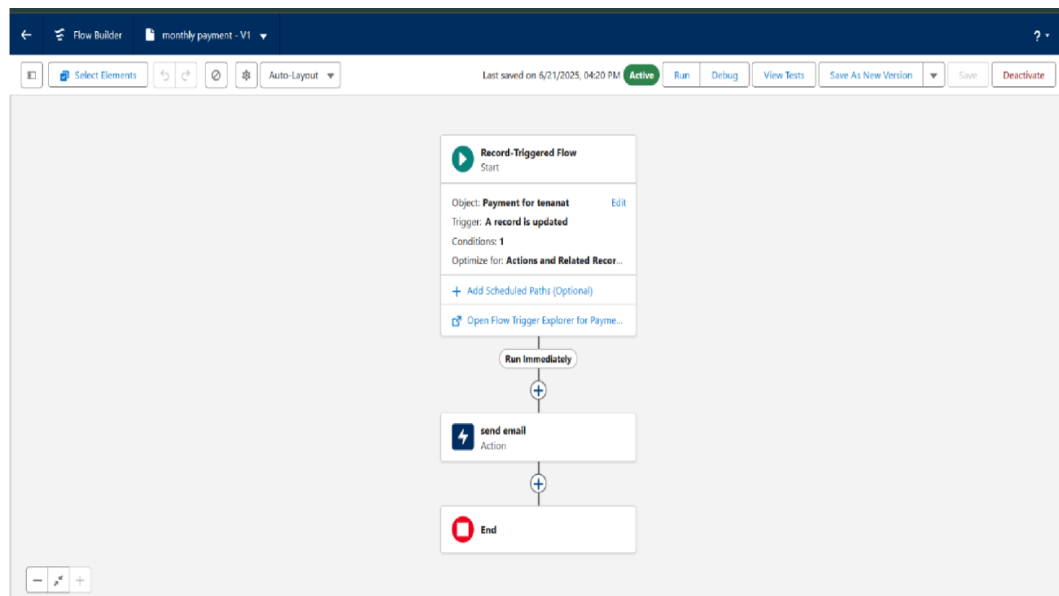
Lightning Page Tabs

➤ Email alerts

The screenshot shows a CRM interface with a top navigation bar containing 'Lease Management', 'lease', 'Properties', 'Payment', 'Tenants', and 'Home'. A search bar is located below the navigation bar. The main content area is divided into two panels. The left panel, titled 'Related', shows a 'Payments (0)' section with a 'New' button, and an 'Approval ... (4)' section with 'Approve' and 'Reject' buttons. Below this is a table with columns 'Step N...', 'Date', 'Status', and 'Assigne...'. The table contains four rows of approval records. The right panel, titled 'Activity', shows a list of activities with filters for 'All time', 'All activities', and 'All types'. It also includes a 'Refresh' button and a 'View All' link. The 'Upcoming & Overdue' section indicates 'No activities to show' and provides instructions on how to get started by sending an email or scheduling a task.

Step N...	Date	Status	Assigne...
Approval	9/6/2025...	Pending	Kaviya Sh...
Approval	9/6/2025...	Submitted	Kaviya Sh...
Approval	9/6/2025...	Rejected	Kaviya Sh...
Approval	9/6/2025...	Submitted	Kaviya Sh...

➤ Flow runs



## ➤ Trigger error messages

The screenshot displays the 'Lease Management' interface in Salesforce. The main section is titled 'NEW lease' and includes tabs for 'Related' and 'Details'. The 'Details' tab is active, showing a form with the following fields:

- \*lease Name: NEW lease
- Owner: Kaviya Shree V
- start date: 9/1/2024 (highlighted with a red border and a red error message: 'Your End date must be greater than start date')
- End date: 8/31/2025
- Property: qwert

A red error message box is displayed over the form, stating: 'We hit a snag. Review the following fields: start date'. The 'start date' field is highlighted in red. The 'End date' field is also highlighted in red. The 'Property' field is highlighted in red. The 'Owner' field is highlighted in red. The 'lease Name' field is highlighted in red. The 'Details' tab is highlighted in blue. The 'Related' tab is highlighted in blue. The 'NEW lease' title is highlighted in blue. The 'Lease Management' title is highlighted in blue. The 'Search...' field is highlighted in blue. The 'Home' link is highlighted in blue. The 'Tenants' link is highlighted in blue. The 'Payment' link is highlighted in blue. The 'Properties' link is highlighted in blue. The 'lease' link is highlighted in blue. The 'New Contact' button is highlighted in blue. The 'Edit' button is highlighted in blue. The 'New Opportunity' button is highlighted in blue. The 'Filters: All time • All activities • All types' text is highlighted in blue. The 'Refresh' button is highlighted in blue. The 'Expand All' button is highlighted in blue. The 'View All' button is highlighted in blue. The 'Upcoming & Overdue' section is highlighted in blue. The 'No activities to show.' text is highlighted in blue. The 'Get started by sending an email, scheduling a task, and more.' text is highlighted in blue. The 'No past activity. Past meetings and tasks marked as done show up here.' text is highlighted in blue. The 'Cancel' button is highlighted in blue. The 'Save' button is highlighted in blue.

## CONCLUSION

The Lease Management System successfully streamlines the operations of leasing through a structured, automated Salesforce application. It improves efficiency, communication, and data accuracy for both admins and tenants.

## 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');
    }
}
}
}
}
}

```

**MonthlyEmailScheduler.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});

}

}

}