

LEASE MANAGEMENT

College Name: Shree Venkateshwara Arts and Science(Co-Education) College
College Code: brubd

TEAM ID: NM2025TMID20984

TEAM MEMBERS:

Team LeaderName: SURESH KRISHNA S
Email:sureshkrishnas2023aids@gmail.com

Team Member1: THIRUMOORTHY P
Email:thirumoorthyp2023aids@gmail.com

Team Member2: VIMAL R
Email:vimalr2023aids@gmail.com

Team Member3: YUVAN M
Email:yuvanm2023aids@gmail.com

Team Member4: THARUNRAJ T
Email:tharunrajt2023aids@gmail.com

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.

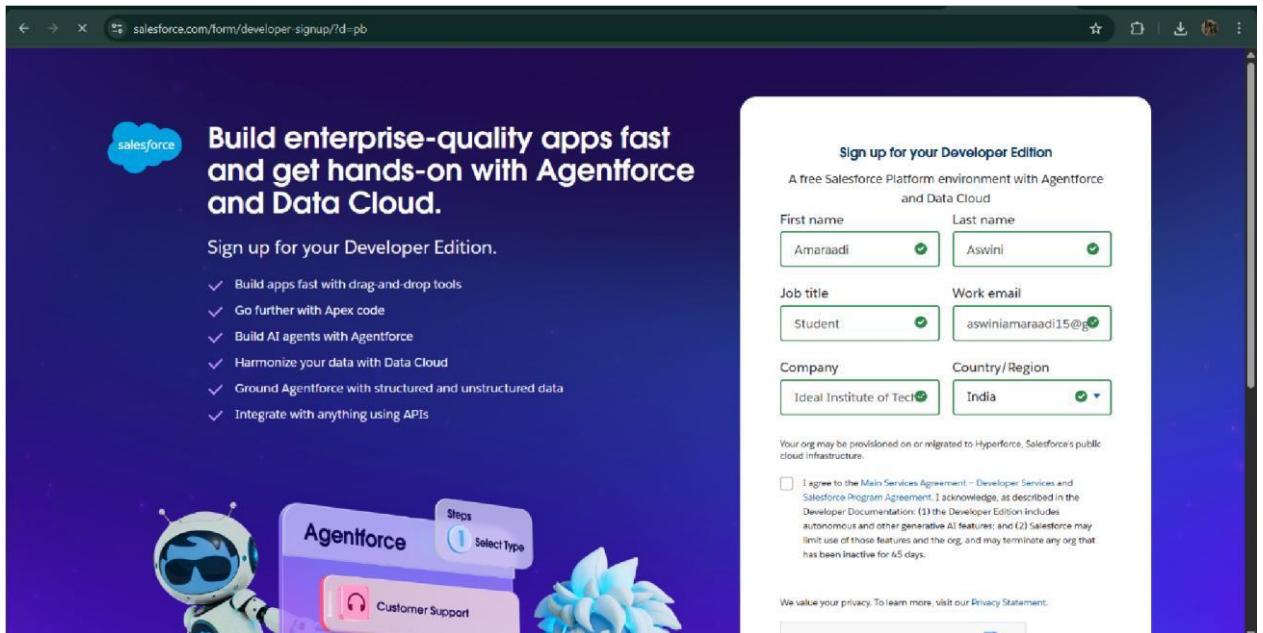
1.2 Purpose

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>



- Created objects: Property, Tenant, Lease, Payment

Details	
Description	API Name property__c Custom ✓ Singular Label property
	Plural Label property
	Enable Reports ✓ Track Activities ✓ Track Field History ✓ Deployment Status Deployed Help Settings Standard salesforce.com Help Window

SETUP > OBJECT MANAGER

Tenant

Details

Description

API Name: Tenant__c

Custom:

- Singular Label: Tenant
- Plural Label: Tenants

Enable Reports:

- ✓ Track Activities
- ✓ Track Field History

Deployment Status: Deployed

Help Settings: Standard salesforce.com Help Window

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

Edit Delete

This screenshot shows the Salesforce Object Manager interface for the 'Tenant' object. The left sidebar lists various configuration tabs like Fields & Relationships, Page Layouts, and Record Types. The main 'Details' tab is selected, showing the API name 'Tenant__c', a custom field, and labels for both singular and plural forms. On the right, there are sections for enabling reports, tracking activities and field history, and setting deployment status to 'Deployed'. A help link to the standard Salesforce help window is also present.

SETUP > OBJECT MANAGER

lease

Details

Description

API Name: lease__c

Custom:

- Singular Label: lease
- Plural Label: lease

Enable Reports:

- ✓ Track Activities
- ✓ Track Field History

Deployment Status: Deployed

Help Settings: Standard salesforce.com Help Window

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

Edit Delete

This screenshot shows the Salesforce Object Manager interface for the 'lease' object. Similar to the 'Tenant' object, it has an API name 'lease__c', a custom field, and labels for singular and plural forms. It includes sections for report enablement, tracking, deployment status set to 'Deployed', and help settings. The left sidebar contains the same configuration tabs as the 'Tenant' object.

The screenshot shows the Salesforce Setup interface under the Object Manager. A sidebar on the left lists various configuration options like Fields & Relationships, Page Layouts, and Record Types. The main pane displays the 'Details' section for the 'Payment for tenant' object. It includes fields for Description, API Name (Payment_for_tenant__c), Singular Label (Payment for tenant), Plural Label (Payment), and several checkboxes for Reports, Activities, Field History, Deployment Status, and Help Settings.

- Configured fields and relationships

The screenshot shows the Salesforce Setup interface under the Object Manager. The sidebar lists configuration options. The main pane shows the 'Fields & Relationships' section for the 'property' object. It lists nine fields: Address, Created By, Last Modified By, Name, Owner, property, property_Name, sfqt, and Type. The 'Owner' field is marked as indexed. The 'property' field is a lookup relationship to the 'property' object itself.

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)		<input checked="" type="checkbox"/>
property	property__c	Lookup(property)		<input checked="" type="checkbox"/>
property_Name	Name	Text(80)		<input checked="" type="checkbox"/>
sfqt	sfqt__c	Text(18)		<input checked="" type="checkbox"/>
Type	Type__c	Picklist		<input checked="" type="checkbox"/>

Setup > Object Manager

Payment for tenant

Fields & Relationships				
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				
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		

Fields & Relationships				
7 items. Sorted by Field Label				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	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

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

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: Placeholder image

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

Lease Management
Application to efficiently handle the processes relate...

[Lightning App Builder](#) [App Settings](#) [Pages](#) [Lease Management](#)

App Settings

[App Details & Branding](#)

[App Options](#)

[Utility Items \(Desktop Only\)](#)

Navigation Items

[User Profiles](#)

Navigation Items

Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

Available Items

Type to filter list... [Create](#)

- Accounts
- Activation Targets
- Activations
- All Sites
- Alternative Payment Methods
- Analytics
- App Launcher
- Appointment Categories
- Appointment Invitations
- Approval Requests

Selected Items

- Payment
- Tenants
- property
- lease

Up ▲ Down ▼

[Lightning App Builder](#) [App Settings](#) [Pages](#) [Lease Management](#)

App Settings

[App Details & Branding](#)

[App Options](#)

[Utility Items \(Desktop Only\)](#)

User Profiles

Choose the user profiles that can access this app.

Available Profiles

Type to filter list...

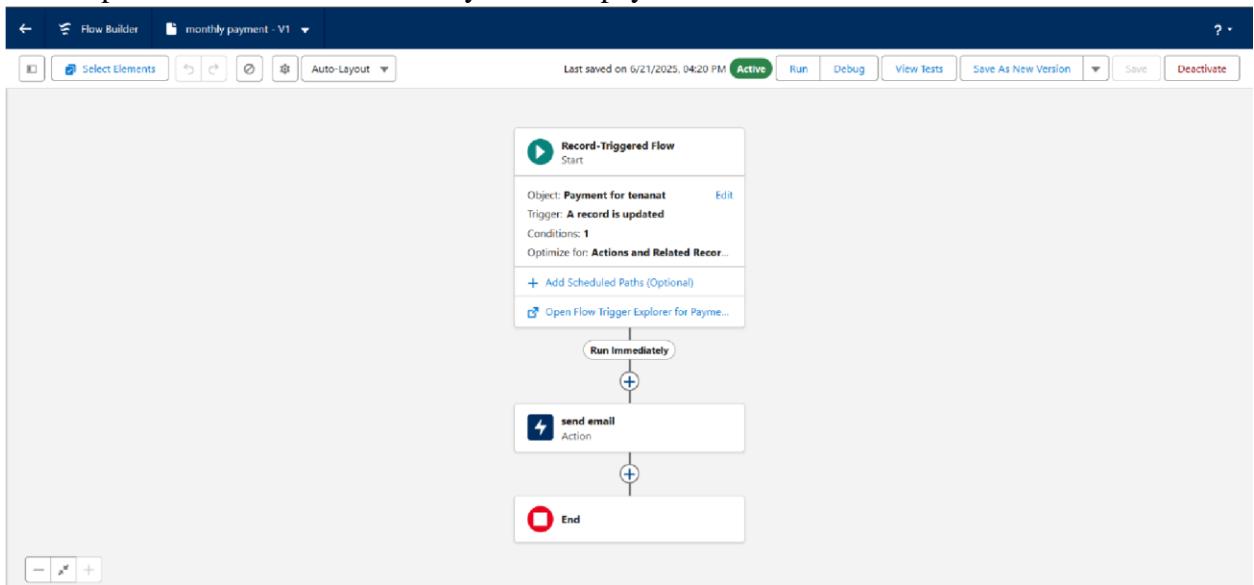
- Analytics Cloud Integration User
- Analytics Cloud Security User
- Anypoint Integration
- Authenticated Website
- Authenticated Website
- B2B Reordering Portal Buyer Profile
- Contract Manager
- Custom: Marketing Profile
- Custom: Sales Profile
- Custom: Support Profile
- Customer Community Login User

Selected Profiles

- System Administrator

The screenshot shows the Salesforce Lease Management interface. The top navigation bar includes tabs for 'Lease Management', 'Payment', 'Tenants', 'property', and 'lease'. A search bar is at the top right. Below the navigation is a section titled 'Recently Viewed' with a dropdown arrow. A table lists 5 items under 'Payment Name': 1. Rahul, 2. Jack, 3. Raj, 4. Sam, and 5. Lahari. To the right of the table are several icons for actions like New, Import, Change Owner, and Assign Label.

- Implemented Flows for monthly rent and payment success

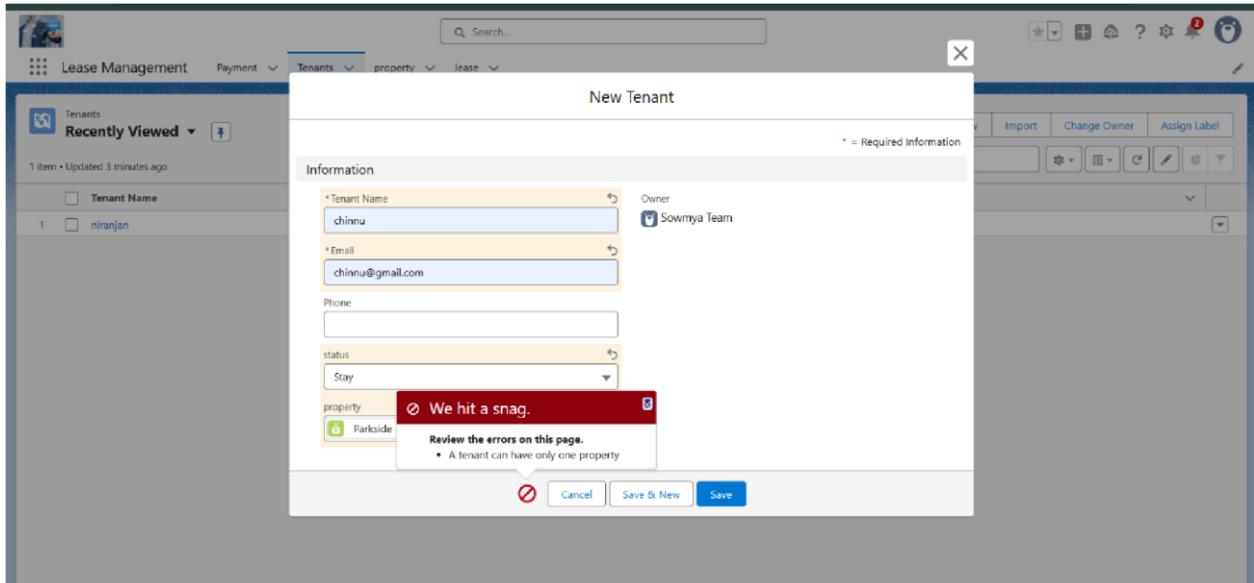


- To create a validation rule to a Lease Object

The screenshot shows the 'Validation Rule Edit' screen for the 'lease' object. The 'Rule Name' is 'lease_end_date'. The 'Error Condition Formula' is set to 'End_date_c <= start_date_c'. A tooltip for the 'ABS' function is displayed, stating: 'Returns the absolute value of a number, a number without its sign'. The 'Active' checkbox is checked.

The screenshot shows the 'Validation Rule Detail' screen for the 'lease_end_date' rule. The rule name is 'lease_end_date', the error condition formula is 'End_date_c <= start_date_c', and the error message is 'Your End date must be greater than start date'. The rule is active and was created by 'Sowmya Team' on 6/19/2025 at 5:37 AM.

- Added Apex trigger to restrict multiple tenants per property



- 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    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        }
36    }
37
38
39
40
41
42
43
44 }
```

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

The screenshot shows the Salesforce Setup interface with the search bar set to 'email template'. Under the 'Email' section, 'Classic Email Templates' is selected. A search result for 'Leave approved' is displayed in a modal window.

Email Template Detail:

- Email Template Name:** Leave approved
- Template Unique Name:** Leave_approved
- Encoding:** Unicode (UTF-8)
- Author:** Sowmya Team [Changes]
- Description:** Created By Sowmya Team, 6/20/2025, 1:06 AM
- Modified By:** Sowmya Team, 6/20/2025, 1:06 AM
- Available For Use:** checked
- Last Used Date:** Not specified
- Times Used:** Not specified

Email Template:

Subject: Leave approved

Main 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 screenshot shows the Salesforce Setup interface with the search bar set to 'email template'. Under the 'Email' section, 'Classic Email Templates' is selected. A search result for 'tenant leaving' is displayed in a modal window.

Email Template Detail:

- Email Template Name:** tenant leaving
- Template Unique Name:** tenant_leaving
- Encoding:** Unicode (UTF-8)
- Author:** Sowmya Team [Changes]
- Description:** Created By Sowmya Team, 6/20/2025, 1:06 AM
- Modified By:** Sowmya Team, 6/20/2025, 1:06 AM
- Available For Use:** checked
- Last Used Date:** Not specified
- Times Used:** Not specified

Email Template:

Subject: request for approve the leave

Main Text Preview:

```
Dear {Tenant__c.CreatedBy},  
  
Please approve my leave
```

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

Classic Email Templates

Leave rejected

Email Template Detail

Field	Value
Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Change]
Description	
Created By	Sowmya Team 6/20/2025, 1:11 AM
Modified By	Sowmya Team 6/20/2025, 1:11 AM

Email Template

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

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

Classic Email Templates

Tenant Email

Email Template Detail

Field	Value
Email Templates from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Tenant Email
Template Unique Name	Tenant_Email
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Change]
Description	
Created By	Sowmya Team 6/20/2025, 1:12 AM
Modified By	Sowmya Team 6/20/2025, 1:12 AM

Email Template

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.

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 a 'Text Email Template' named 'tenant payment'. The 'Email Template Detail' section shows the following information:

- Email Templates from Salesforce: Unified Public Classic Email Templates
- 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
- Available For Use: ✓
- Last Used Date: Times Used: 0
- Modified By: Sowmya Team 6/20/2025, 1:13 AM

The 'Email Template' preview shows the subject 'Confirmation of Successful Monthly Payment' and the plain text preview:

```

Subject: Confirmation of Successful Monthly Payment
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 'Data' section selected. Under 'Data', 'Approval Processes' is chosen. A search bar at the top left shows 'approval'. The main content area displays an approval process named 'Tenant: TenantApproval'. The 'Process Definition Detail' section shows the following information:

- Process Name: TenantApproval
- Unique Name: TenantApproval
- Description: Tenant: status equals Stay
- Entry Criteria: Record Editability: Administrator ONLY
- Active: ✓
- Next Automated Approver Determined By: [empty]
- Allow Submitters to Recall Approval Requests:
- Created By: Sowmya Team 6/20/2025, 3:41 AM
- Modified By: Sowmya Team 6/26/2025, 11:57 PM

The 'Initial Submission Actions' section shows:

Action Type	Description
Record Lock	Lock the record from being edited

The 'Approval Steps' section shows:

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

For Check for Vacant:

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

- Approval Processes** page.
- Tenant: check for vacant** process selected.
- Process Definition Detail** section:
 - Process Name: check for vacant
 - Unique Name: check_for_vacant
 - Description: Tenant status EQUALS Leaving
 - Entry Criteria: Tenant status EQUALS Leaving
 - Record Editability: Administrator ONLY
 - Next Automated Approver Determined By: Active (checked)
 - Approval Assignment Email Template: Leave approved
 - Initial Submitters: Tenant Owner
 - Created By: Sowmya Team
 - Modified By: Sowmya Team
- Initial Submission Actions** section:

Action	Type	Description
Record Lock		Lock the record from being edited
Email Alert		please approve my leave
- Approval Steps** section:

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

● Apex Trigger

Create an Apex Trigger

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

- File • Edit • Debug • Test • Workspace • Help • < >**
- test.apc** file is open.
- Code Coverage: None** and **API Version: 64**.
- trigger test on Tenant_c (before insert)** code is present.
- A modal window titled "Open" is displayed, listing "Entity Type" options:
 - Classes
 - Triggers
 - Pages
 - Page Components
 - Objects
 - Static Resources
 - Packages
- The "Triggers" option is selected in the modal.
- At the bottom of the screen, there are tabs for **Logs**, **Tests**, **Checkpoints**, **Query Editor**, **View Status**, **Progress**, and **Problems**. The **Problems** tab is active.

Developer Console - Google Chrome
 orgfarm-5d1fe805f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

```
File Edit Debug Test Workspace Help < >
test.apex testHandler.apc MonthlyEmailScheduler.apc
Code Coverage Name: API Version: 64 Go To
1 trigger test on Tenant__c (before insert)
2
3 +
4
5 if(trigger.isInsert && trigger.isBefore){
6
7   testHandler.preventInsert(trigger.new);
8
9 }
10
11 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Create an Apex Handler class

Developer Console - Google Chrome
 orgfarm-5d1fe805f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

```
File Edit Debug Test Workspace Help < >
test.apex testHandler.apc MonthlyEmailScheduler.apc
Code Coverage Name: API Version: 64 Go To
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(exi
10      }
11
12
13
14+     for (Tenant__c newTenant : newList) {
15
16
17       if (newTenant.Property__c != null) {
18
19         newTenantaddError('A
20
21
22
23 }
```

Entity Type Entity Name Namespace Related

Entity Type	Entity Name	Namespace	Related
Classes	testHandler	MonthlyEmailScheduler	← test ApexTrigger Referenced ← property CustomField References ← Tenant__c Object References ← Tenant__c Object References
Triggers			
Pages			
Page Components			
Objects			
Static Resources			
Packages			

Open Filter Hide Managed Packages Refresh

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Developer Console - Google Chrome

orgfarm-5df1e805f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSPage

File Edit Debug Test Workspace Help

test.apex testHandler.apxc MonthlyEmailScheduler.apac

Code Coverage Name API Version 64 Go To

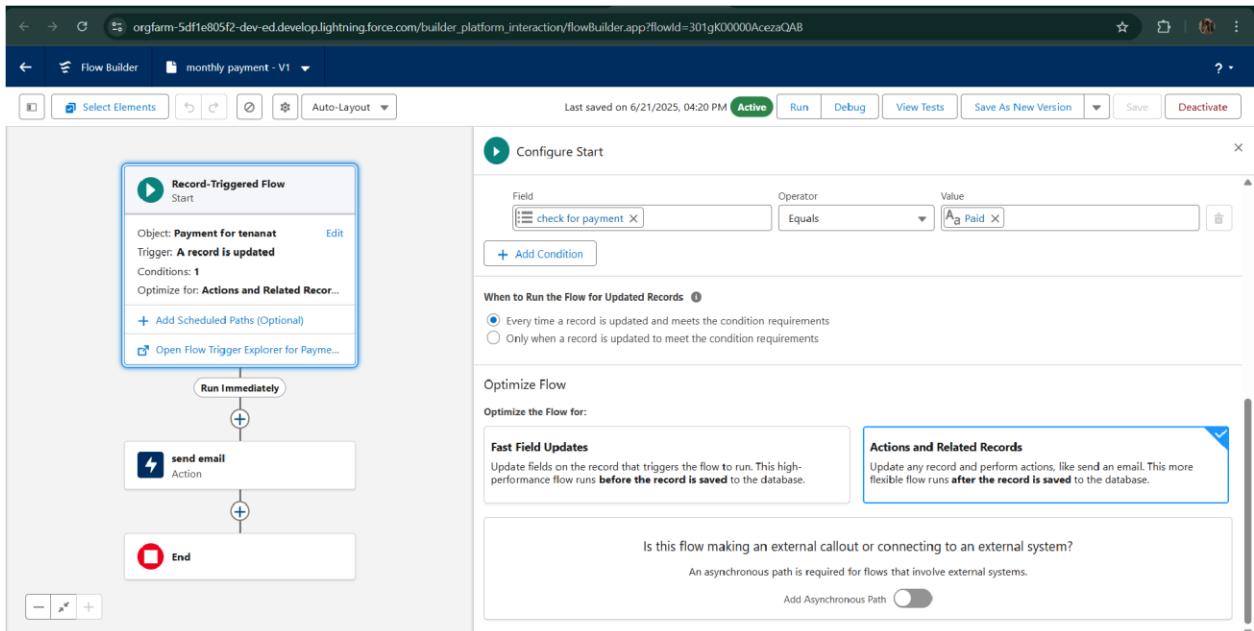
```

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     }
26
27 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

● FLOWS



Record-Triggered Flow Start

Object: **Payment for tenant** Edit
Trigger: **A record is updated**
Conditions: 1
Optimize for: **Actions and Related Records**

+ Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Payment for tenant...

Run Immediately

send email Action

End

Configure Start

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

* Object: **Payment for tenant**

Configure Trigger
Trigger the Flow When:
 A record is created
 A record is updated
 A record is created or updated
 A record is deleted

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

```

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;
23

```

Open

Entity Type	Entities	Related
Entity Type	Name Namespace	Name Extent Direction
Classes	testHandler	CrossTrigger References
Triggers	MonthlyEmailScheduler	Email CustomField References
Pages		← Tenant__c CObject References
Page Components		← Tenant__c SOBJECT References
Objects		
Static Resources		
Packages		

```

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        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 a positive living environment for all.';
25
26            String emailSubject = 'Rent Remind: Monthly Rent Payment Due';
27
28            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
29
30            email.setToNames(recipientEmail);
31            email.setSubject(emailSubject);
32            email.setPlainTextBody(emailContent);
33
34            messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
35
36        }
37
38    }
39
40 }
41
42

```

Schedule Apex class

The screenshot shows the Salesforce Setup Apex Classes page. The sidebar on the left has a search bar with 'apex' typed in. The main area displays the 'Apex Classes' section for the 'Apex Class' named 'MonthlyEmailScheduler'. The 'Apex Class Detail' table shows the following information:

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

The 'Class Body' tab is selected, showing the Apex code for the 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 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 a positive living environment for all.';
25
26            String emailSubject = 'Rent Remind: Monthly Rent Payment Due';
27
28            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
29
30            email.setToNames(recipientEmail);
31            email.setSubject(emailSubject);
32            email.setPlainTextBody(emailContent);
33
34            messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
35
36        }
37
38    }
39
40 }
41
42

```

Screenshot of the Salesforce Lightning interface showing the 'Lease Management' tab selected. A context menu is open over a tenant record named 'Aswini'. The menu options include:

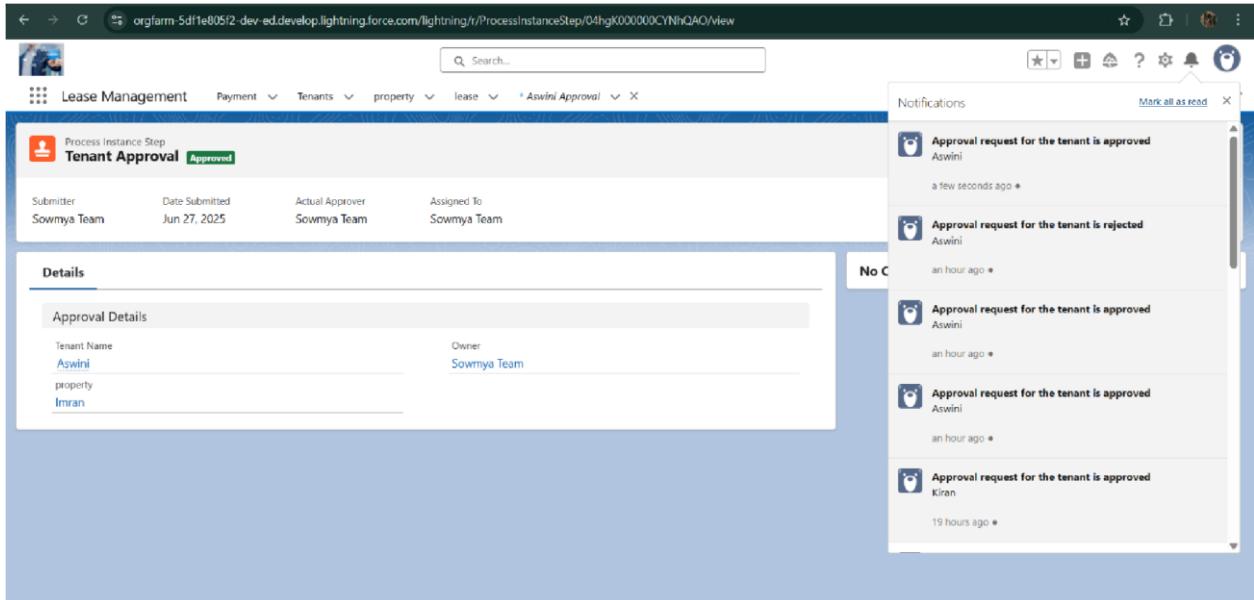
- New Case
- New Lead
- Delete
- Clone
- Change Owner
- Printable View
- Submit for Approval**
- Edit Labels

The 'Submit for Approval' option is highlighted.

Screenshot of the Salesforce Lightning interface showing the 'Lease Management' tab selected. A success message 'Tenant was submitted for approval.' is displayed in a green banner at the top right. The tenant record for 'Aswini' is shown with all fields filled in, including:

- * Tenant Name: Aswini
- * Email: aswiniamaraadi15@gmail.com
- Phone: (905) 223-5567
- Status: Leaving
- Property: Imran

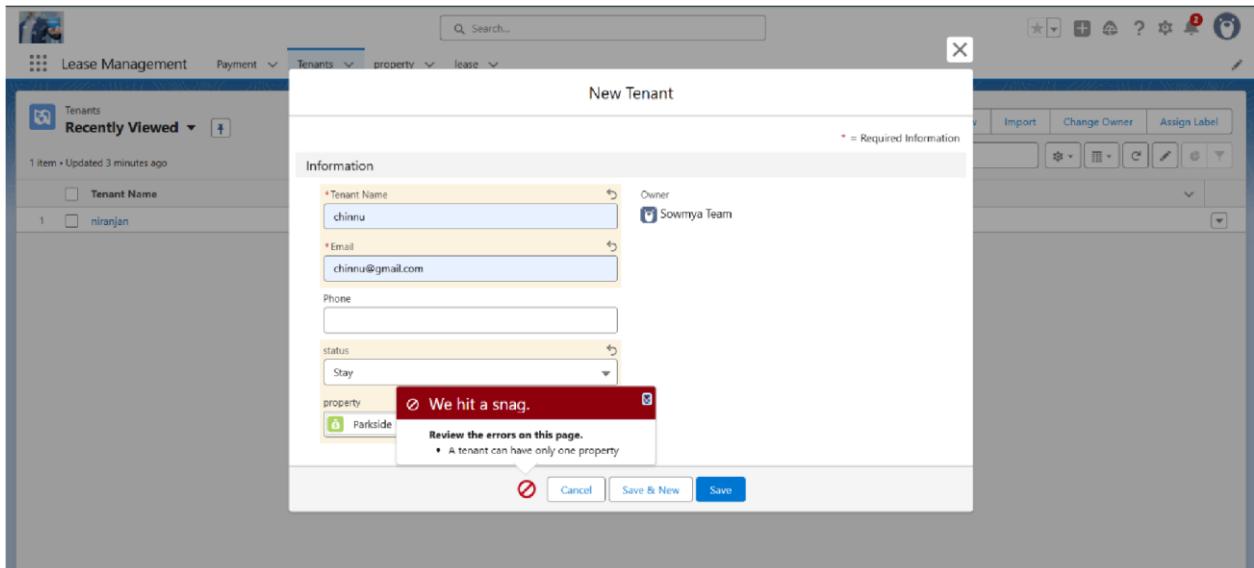
The 'Save' button is visible at the bottom right.



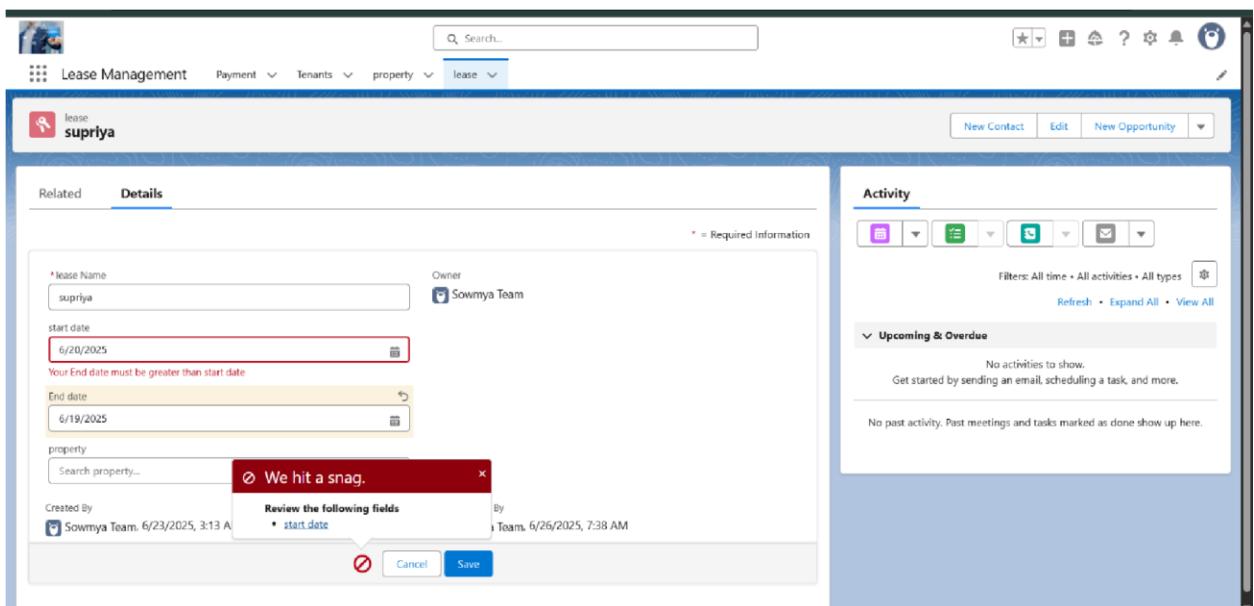
FUNCTIONAL AND PERFORMANCE TESTING

Performance Testing

- Trigger validation by entering duplicate tenant-property records

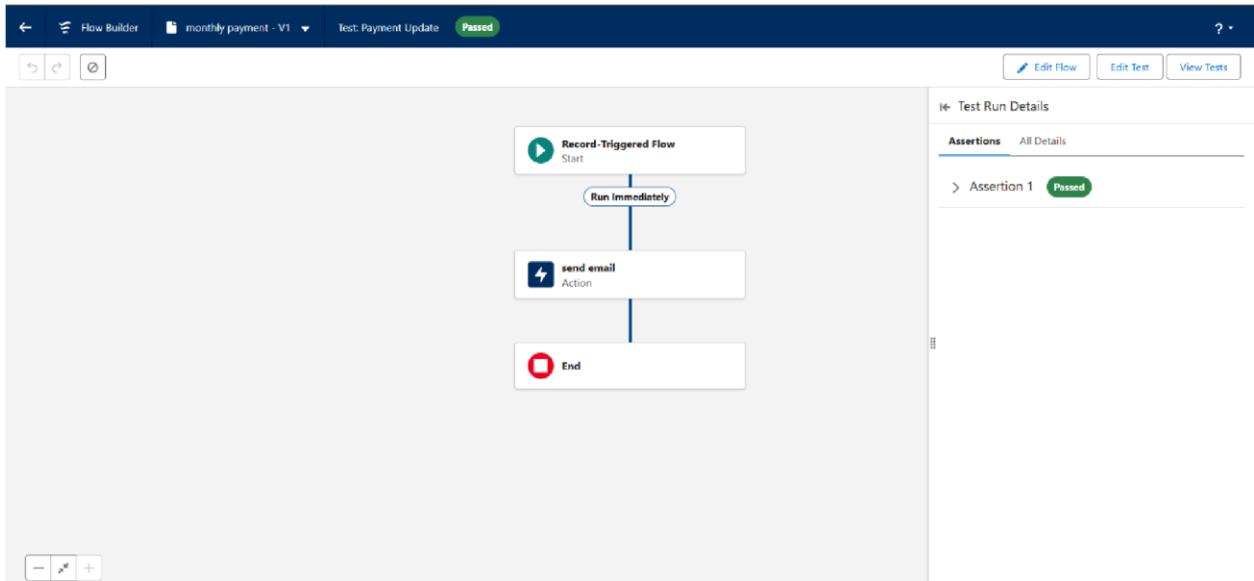


- Validation Rule checking



The screenshot shows a Salesforce Lease Management page for a lease named 'supriya'. The start date is set to 6/20/2025 and the end date to 6/19/2025. A validation error message 'We hit a snag.' appears, stating 'Review the following fields' and pointing to the start date field. The lease is owned by the 'Sowmya Team'.

- Test flows on payment update



- Approval process validated through email alerts and status updates

The screenshot shows a CRM interface for 'Lease Management'. On the left, a 'Tenant' record for 'niranjan' is displayed with fields for Name, Email, Phone, Status, and Property (Parkside Lofts). The 'Details' tab is selected. On the right, a 'Notifications' sidebar lists several recent messages:

- Approval request for the tenant is approved** (niranjan) - a few seconds ago
- Approval request for the tenant is rejected** (niranjan) - Jun 23, 2025, 4:29 PM
- Approval request for the tenant is approved** (niranjan) - Jun 23, 2025, 4:25 PM
- Approval request for the tenant is approved** (niranjan) - Jun 23, 2025, 4:14 PM
- New Guidance Center learning resource available** (Define Your Sales Process) - Jun 20, 2025, 1:28 PM

The screenshot shows a CRM interface for 'Lease Management'. On the left, the 'Approval History' section displays a list of steps with their dates, statuses, and assignees. The steps include 'Step 1' (Approved), 'Approval Request Submitted' (Submitted), 'Step 1' (Rejected), 'Approval Request Submitted' (Submitted), 'Step 1' (Approved), and 'Approval Request Submitted' (Submitted). Below this is a 'Payment' section showing two entries: 'Jack' and 'Rahul'. On the right, there is a sidebar with a message: 'Get started by sending an email, scheduling a task, and more.' and a note: 'No past activity. Past meetings and tasks marked as done show up here.'

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

RESULTS

Output Screenshots

- Tabs for Property, Tenant, Lease, Payment

The screenshot shows the Salesforce Setup interface under the 'Tabs' section. It displays four categories of tabs:

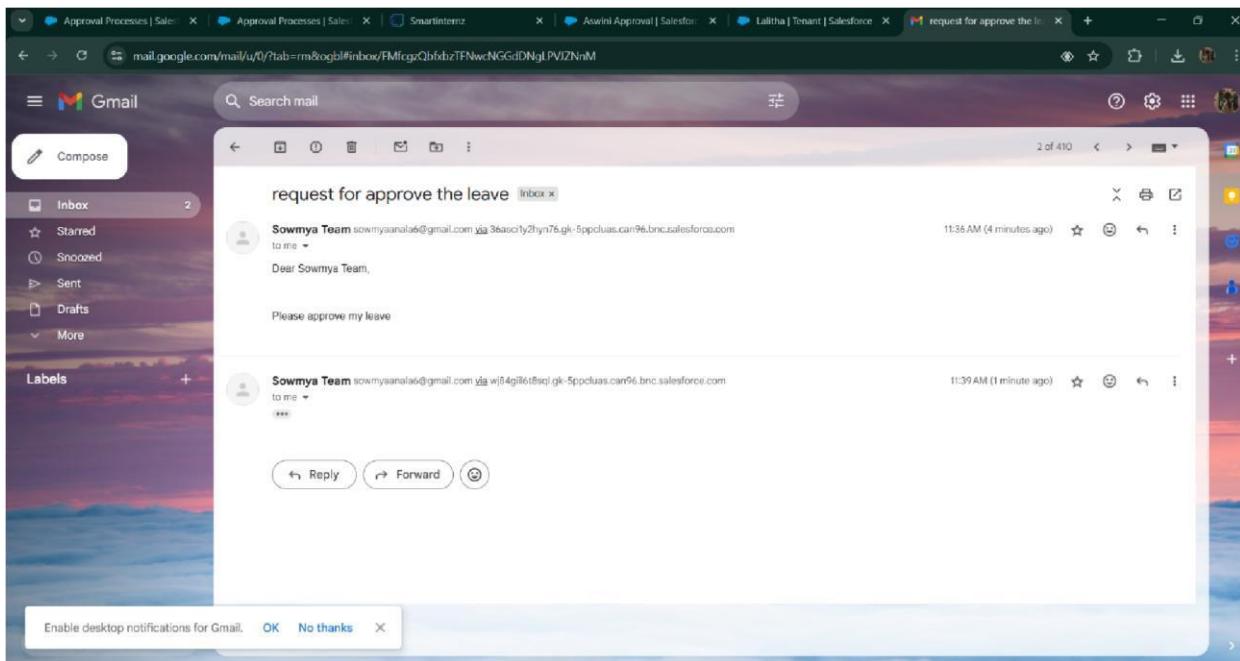
- Custom Object Tabs:** Contains tabs for Lease (Tab Style: Keys), Payment (Tab Style: Credit card), property (Tab Style: Sack), and Tenant (Tab Style: Map).
- Web Tabs:** Shows a message: "No Web Tabs have been defined".
- Visualforce Tabs:** Shows a message: "No Visualforce Tabs have been defined".

- Email alerts

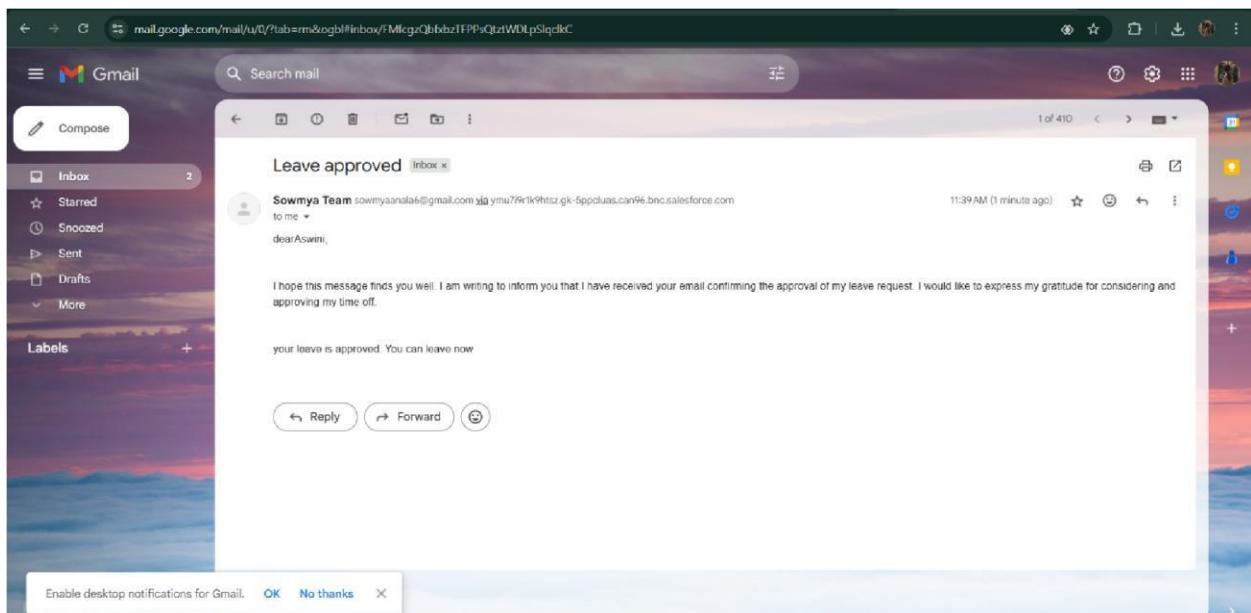
The screenshot shows the 'Lease Management' application interface. The current view is 'Approval History' for a tenant named 'niranjana'. The table displays the following data:

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

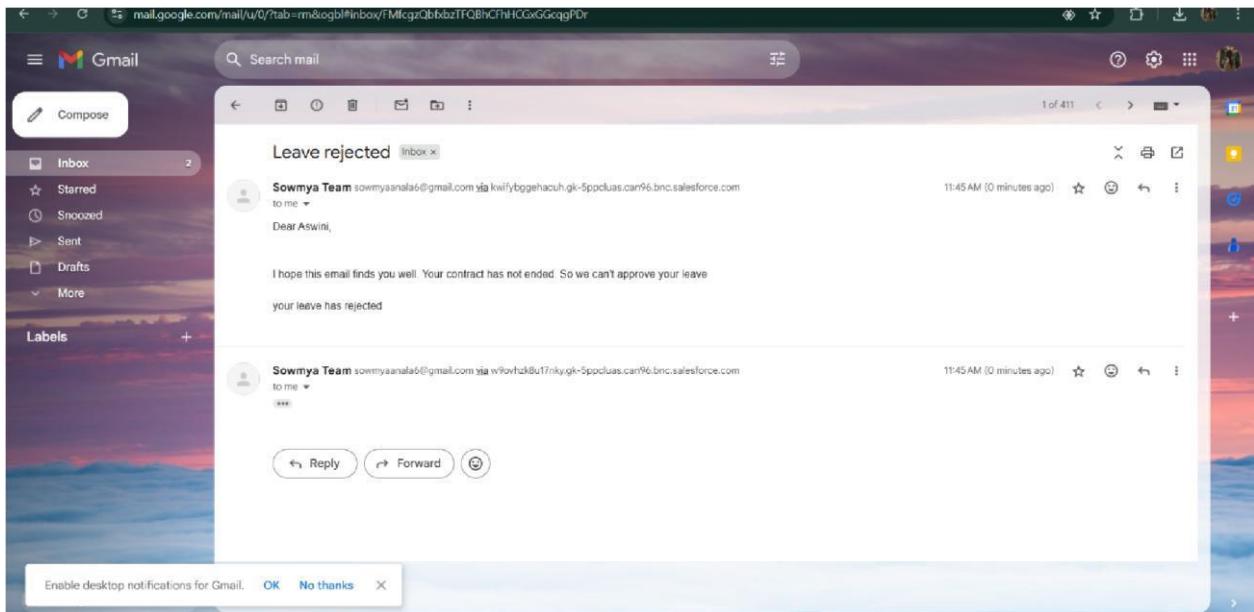
- Request for approve the leave



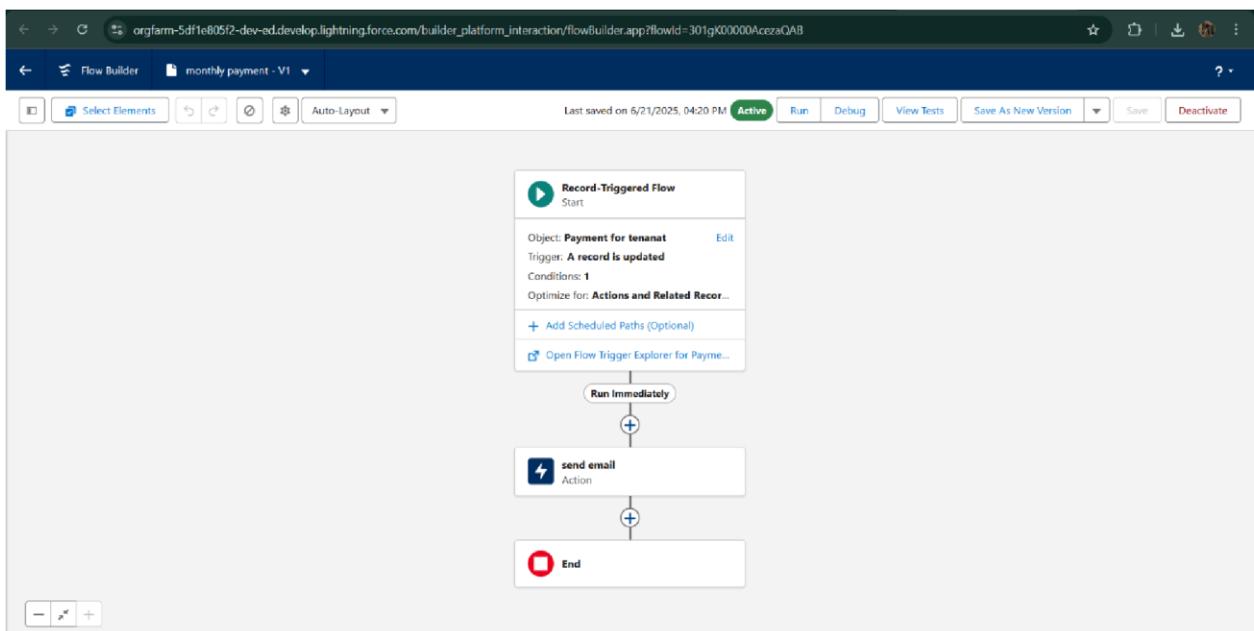
- Leave approved



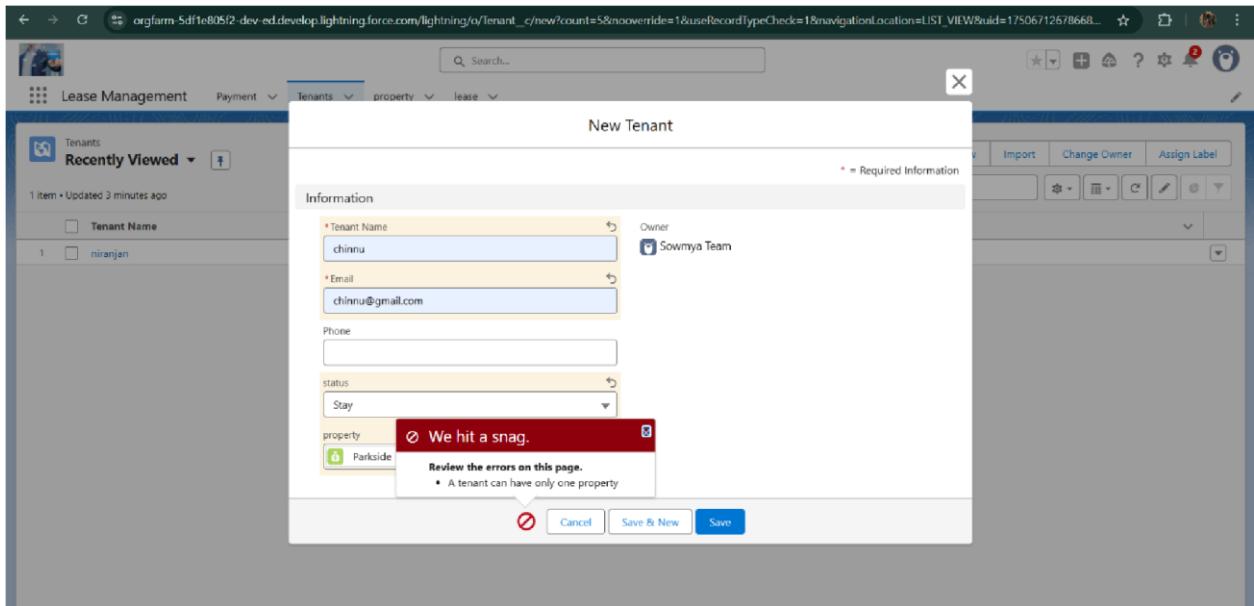
- Leave rejected



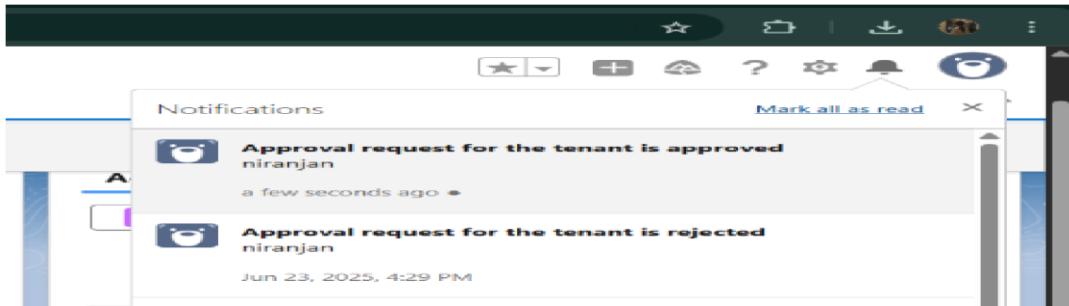
- Flow runs



- Trigger error messages



- Approval process notifications



ADVANTAGES & DISADVANTAGES

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');

        }

    }

}

```

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});  
    }  
}  
}
```