

Lease Management

College Name: Bishop Appasamy college of arts and science

College Code: bru3g

TEAM MEMBERS:

Team Leader Name: Meshach Fernandez

Email: meshachfernandez34@gmail.com

Team Member 1: Prathap

Email: prathapbacas@gmail.com

Team Member 2: Kewin kriflo

Email: kewincriflo@gmail.com

Team Member 3: Balaragan

Email: balareganb@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.



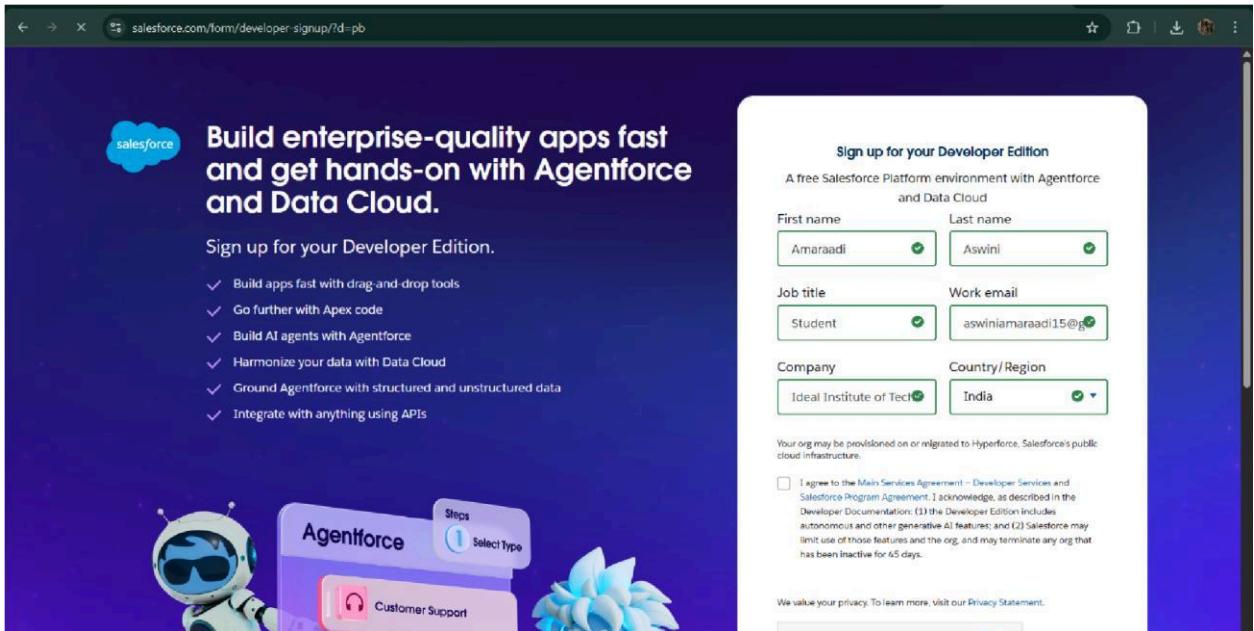
1.2 Purpose

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/sd=pb>



- Created objects: Property, Tenant, Lease, Payment

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

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

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main title is 'SETUP > OBJECT MANAGER Payment for tenantat'. On the left, a sidebar lists various configuration options under 'Details' and 'Fields & Relationships'. The main content area displays the 'Details' tab for the 'Payment for tenantat' object. It shows fields like 'API Name' (Payment_for_tenantat_c), 'Singular Label' (Payment for tenantat), and 'Plural Label' (Payment). On the right, there are sections for 'Enable Reports', 'Track Activities', 'Track Field History', 'Deployment Status', and 'Help Settings'. Buttons for 'Edit' and 'Delete' are located at the top right.

- Configured fields and relationships

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main title is 'SETUP > OBJECT MANAGER property'. On the left, a sidebar lists various configuration options under 'Details' and 'Fields & Relationships'. The main content area displays the 'Fields & Relationships' tab for the 'property' object. A table lists the fields with their labels, names, data types, controlling fields, and indexing status. Fields include Address, Created By, Last Modified By, Name, Owner, property, property Name, sqft, and Type.

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

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

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

App Settings

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
- Custom

Selected Items

- Payment
- Tenants
- property
- lease

Up ▾ Down ▾

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

App Settings

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

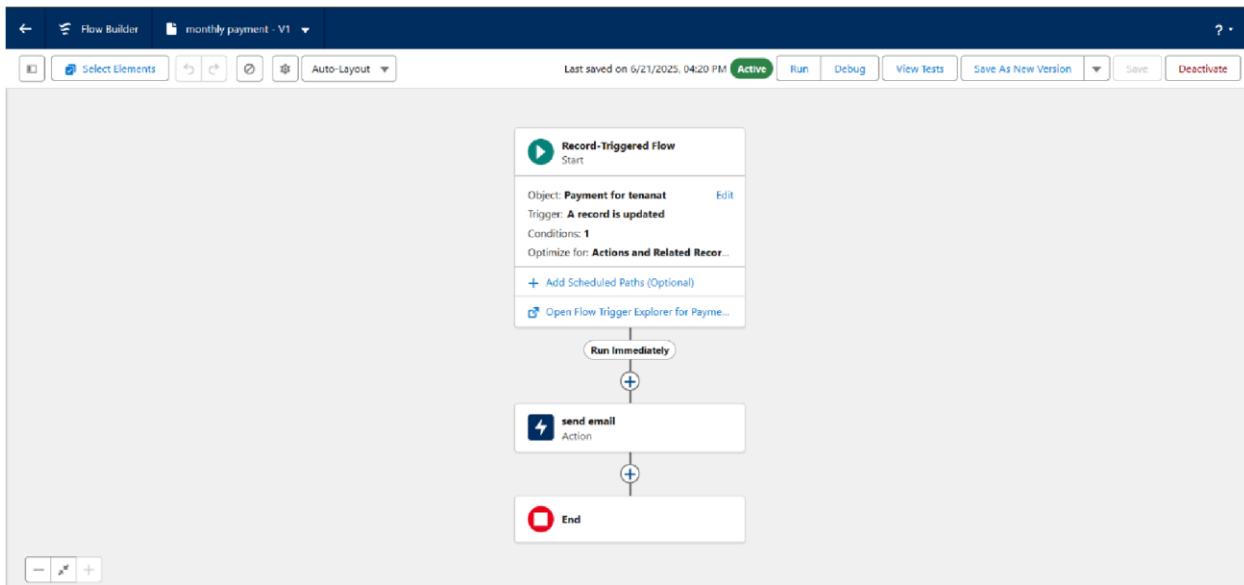
Selected Profiles

- System Administrator

Up ▾ Down ▾

The screenshot shows the Salesforce Lease Management interface. The top navigation bar includes 'Lease Management', 'Payment' (selected), 'Tenants', 'property', and 'lease'. Below the navigation is a search bar and a toolbar with icons for 'New', 'Import', 'Change Owner', and 'Assign Label'. The main area displays a list titled 'Recently Viewed' under the 'Payment' category. It shows 5 items updated a few seconds ago, with columns for 'Payment Name' and dropdown arrows for sorting. The items listed are: 1. Rahul, 2. Jack, 3. Raj, 4. Sam, and 5. Lahari.

- Implemented Flows for monthly rent and payment success

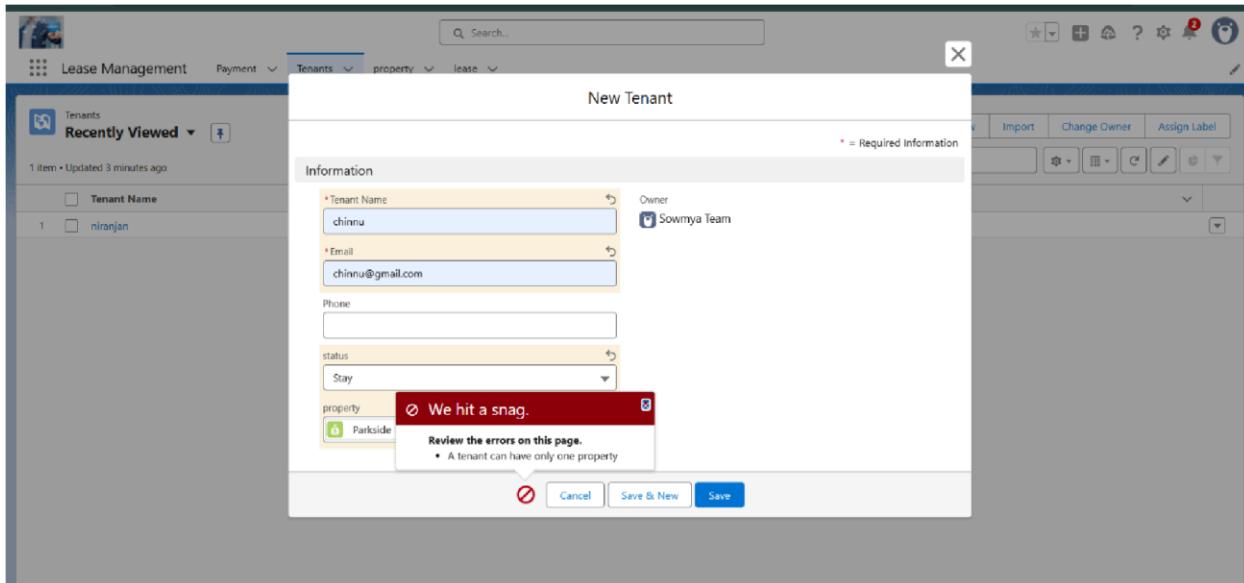


- To create a validation rule to a Lease Object

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

The screenshot shows the 'Validation Rule Detail' screen for the 'lease' object. The validation rule is named 'lease_end_date'. The error condition formula is `End_date_c <= start_date_c`. The error message is 'Your End date must be greater than start date'. The error location is 'start date'. The rule was created by 'Sowmya Team' on 6/19/2025, 5:37 AM, and modified by 'Sowmya Team' on 6/26/2025, 7:47 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 public static void sendMonthlyEmails() {
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';
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     }
36 }
```

- 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' category, 'Classic Email Templates' is selected. A specific template named 'Leave approved' is displayed in the center. The 'Email Template Detail' section shows the following information:

Category	Value
Email Templates from Salesforce	Leave approved
Template Unique Name	Leave_approved
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Change]
Description	
Created By	Sowmya Team, 6/20/2025, 1:06 AM
Modified By	Sowmya Team, 6/20/2025, 1:06 AM

The 'Email Template' preview section contains the following 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 screenshot shows the Salesforce Setup interface with the search bar set to 'email template'. Under the 'Email' category, 'Classic Email Templates' is selected. A specific template named 'tenant leaving' is displayed in the center. The 'Email Template Detail' section shows the following information:

Category	Value
Email Templates from Salesforce	tenant leaving
Template Unique Name	tenant_leaving
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Change]
Description	
Created By	Sowmya Team, 6/20/2025, 1:06 AM
Modified By	Sowmya Team, 6/20/2025, 1:06 AM

The 'Email Template' preview section contains the following content:

```

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

The screenshot shows the Salesforce Setup interface with the search bar set to "email template". The left sidebar under "Email" is expanded, showing "Classic Email Templates" selected. The main content area displays the "Classic Email Templates" page for a "Text Email Template" named "Leave rejected".

Email Template Detail:

Email Templates from Salesforce	Unified Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	Sowmya_Team [Changed]
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 search bar set to "email template". The left sidebar under "Email" is expanded, showing "Classic Email Templates" selected. The main content area displays the "Classic Email Templates" page for a "Text Email Template" named "Tenant Email".

Email Template Detail:

Email Templates from Salesforce	Unified Public Classic Email Templates
Email Template Name	Tenant Email
Template Unique Name	Tenant_Email
Encoding	Unicode (UTF-8)
Author	Sowmya_Team [Changed]
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 Template Name: tenant payment
- Template Unique Name: tenant_payment
- Encoding: Unicode (UTF-8)
- Author: Sowmya Team (Change)
- Created By: Sowmya Team
- Last Used Date: 6/20/2025, 1:13 AM
- Available For Use: checked
- Times Used: 1
- Modified By: Sowmya Team
- Modified Date: 6/20/2025, 1:13 AM

The 'Email Template' section below shows the subject 'Confirmation of Successful Monthly Payment' and a 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:
- Entry Criteria: Tenant__c.status equals Stay
- Record Editability: Administrator ONLY
- Active: checked
- Next Automated Approver Determined By:
- Allow Submitters to Recall Approval Requests: unchecked
- Created By: Sowmya Team
- Modified By: Sowmya Team
- Modified Date: 6/26/2025, 11:57 PM

The 'Initial Submission Actions' section shows a single action:

Action Type	Description
Record Lock	Lock the record from being edited

The 'Approval Steps' section shows one step:

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

For Check for Vacant:

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

- Process Definition Detail:**
 - Process Name: check for vacant
 - Unique Name: Check_for_vacant
 - Description: Tenant: check for vacant
 - Entry Criteria: Tenant__c.status EQUALS Leaving
 - Record Equality: Administrator ONLY
 - Approval Assignment Email Template: Leave_approved
 - Initial Submitters: Tenant Owner
 - Created By: Sowmya_Team
 - Modified By: Sowmya_Team
 - Status: Active
- Initial Submission Actions:**
 - Action: Record Lock
 - Type: Record Lock
 - Description: Lock the record from being edited
 - Details: PLEASE APPROVE MY LEAVE
- Approval Steps:**

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 an Apex trigger named 'test.apex' open. The trigger code is as follows:

```

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

```

A modal window titled 'Open' is displayed, showing a search results table for 'test'. The table has columns for Entity Type, Edition, and Related.

Entity Type	Edition	Related
Classes	Name	Namespace
Triggers	test	
Pages		
Page Components		
Objects		
Static Resources		
Packages		

Developer Console - Google Chrome
orgfarm-5d1e805f2-dev-ed-develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

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

test.apexp [testHandler.apexp | MonthlyEmailScheduler.apexp]

Code Coverage: None • API Version: 64 • Go To

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

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Create an Apex Handler class

Developer Console - Google Chrome
orgfarm-5d1e805f2-dev-ed-develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

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

test.apexp [testHandler.apexp | MonthlyEmailScheduler.apexp]

Code Coverage: None • API Version: 64 • Go To

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

Entity Type Entity Related

Entity Type	Entity	Related
Classes	testHandler	↳ test ApexTrigger Referenced
Triggers	MonthlyEmailScheduler	↳ property CustomerRef References
Pages		↳ Tenant__c Subject References
Objects		↳ Tenant__c Subject References
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/ApexCSIPage

File Edit Debug Test Workspace Help < >

testHandler.apxc MonthlyEmailScheduler.apxc

Code Coverage Name API Version: 64 Go To

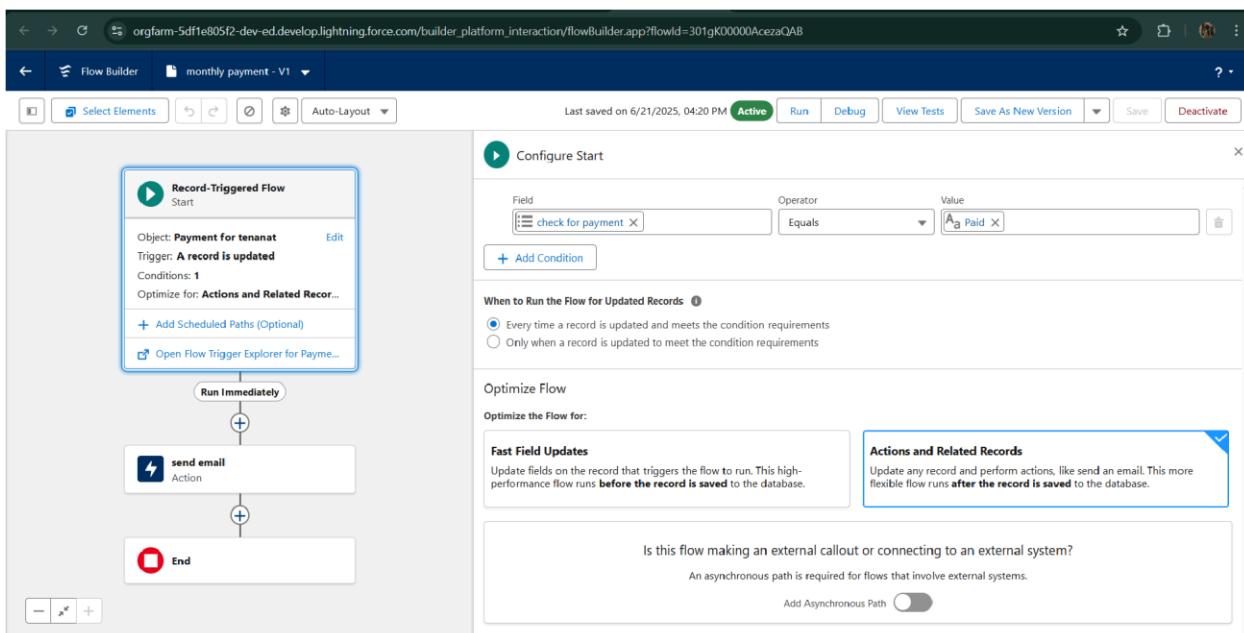
```

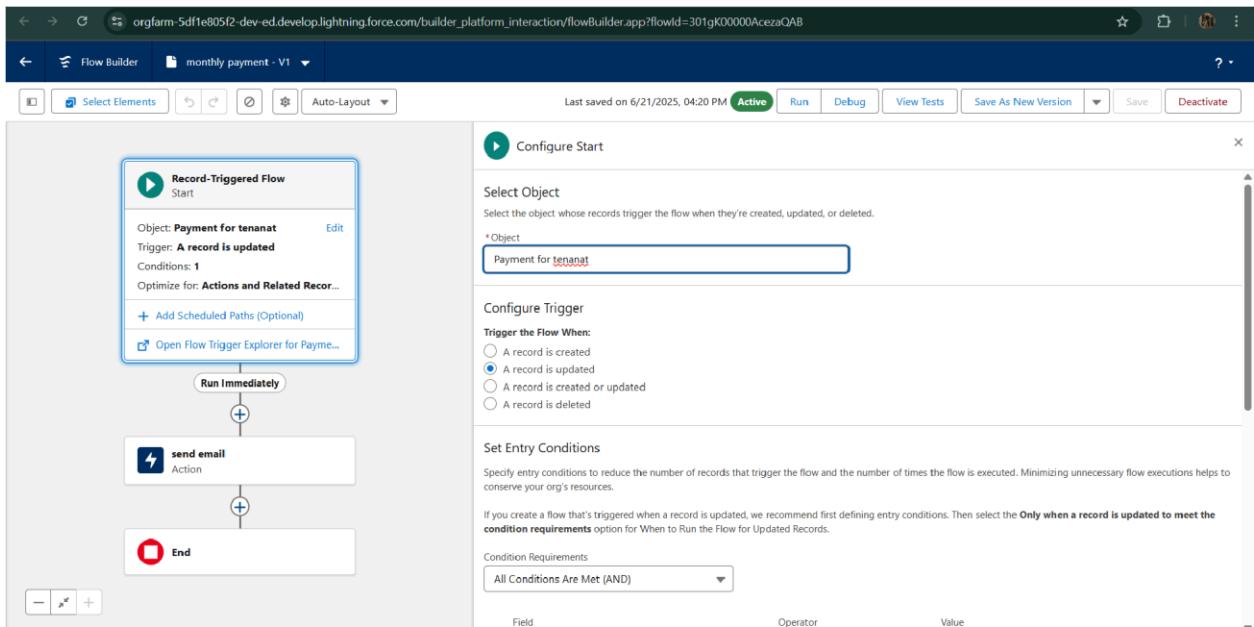
1 + public class testHandler {
2
3     public static void preventInsert(List<Tenant__c> newList) {
4
5         Set<Id> existingPropertyIds = new Set<Id>();
6
7         for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c WHERE Property__c != null]) {
8
9             existingPropertyIds.add(existingTenant.Property__c);
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





- Schedule class :
Create an Apex Class

```

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

```

Open

Entity Type	Entities	Related
Classes	testHandler	Name Extent CrossType References
Triggers	MonthlyEmailScheduler	Email CustomField Object References
Pages		Tenant__c Tenant__c Object References
Page Components		
Objects		
Static Resources		
Packages		

```

Developer Console - Google Chrome
orgafarm-5d1fe05f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage
Reset
50%
Reset

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 = 'Re: 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            messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
37
38        }
39
40    }
41
42

```

The screenshot shows the Apex code for the `MonthlyEmailScheduler` class. The class implements the `Schedulable` interface and contains a single method `execute`. This method checks if the current day is 1. If it is, it calls the static method `sendMonthlyEmails`. The `sendMonthlyEmails` method queries for all `Tenant__c` records and loops through them, sending a single email to each tenant's `Email__c` field. The email body is a fixed message about timely rent payment.

Schedule Apex class

The screenshot shows the Salesforce Setup page with the search bar set to "apex". The left sidebar is expanded, showing categories like Email, Custom Code, and Apex Classes. Under Apex Classes, the "Apex Classes" tab is selected, displaying the details for the `MonthlyEmailScheduler` class. The class is named `MonthlyEmailScheduler`, has a namespace prefix of `apex`, and was created by the "Somanya Team" on June 23, 2025, at 2:46 AM. It is currently active and has 0% code coverage. The "Class Body" tab is selected, showing the Apex code listed above.

```

Apex Class Detail
Name: MonthlyEmailScheduler
Namespace Prefix: apex
Created By: Somanya Team, 6/23/2025, 2:46 AM
Status: Active
Code Coverage: 0% (0/15)
Last Modified By: Somanya Team, 6/23/2025, 2:47 AM

Class Body
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 = 'Re: 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            messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
37
38        }
39
40    }
41
42

```

The screenshot shows a Salesforce Lightning page for a tenant named 'Aswini'. The 'Details' tab is selected. The form fields include:

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

Owner: Sowmya Team. Last Modified By: Sowmya Team. Last Modified: 6/26/2025, 11:06 PM.

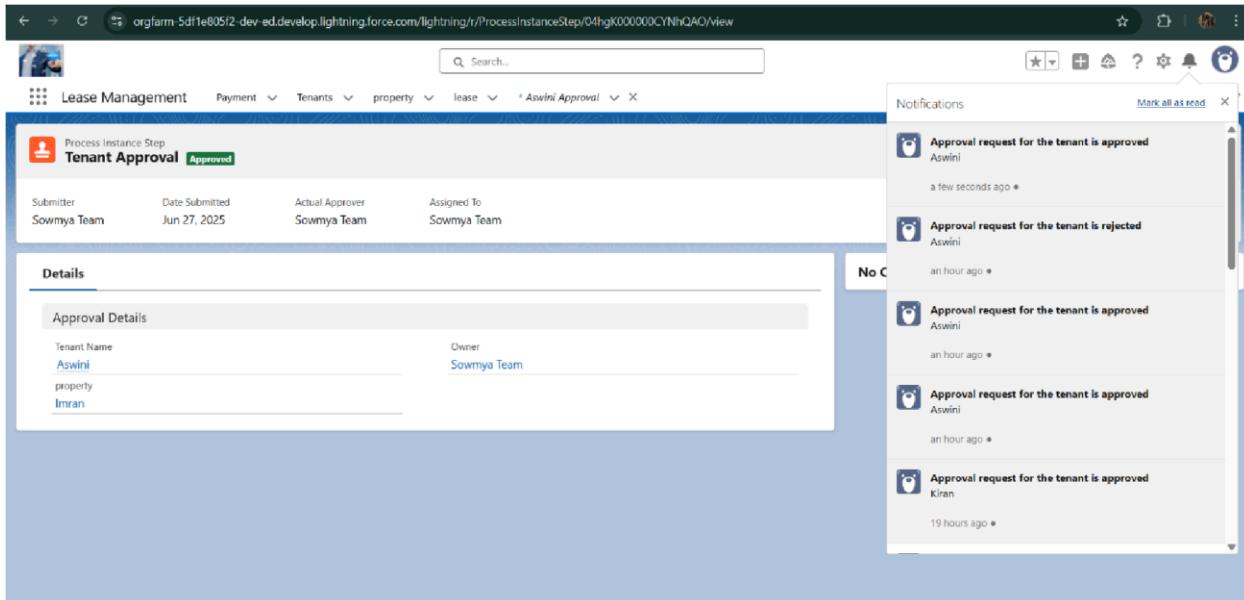
Activity sidebar:

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

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.

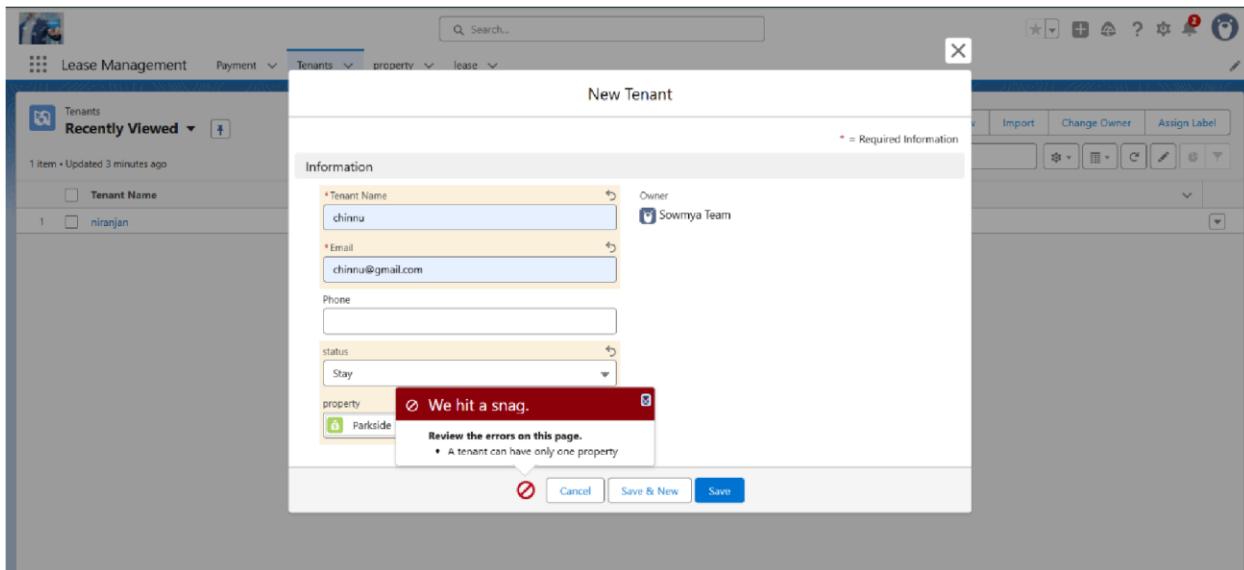
The screenshot shows the same Salesforce Lightning page for tenant 'Aswini'. A green success message at the top right says 'Tenant was submitted for approval.' The rest of the page is identical to the first screenshot, showing the tenant details and the activity sidebar.



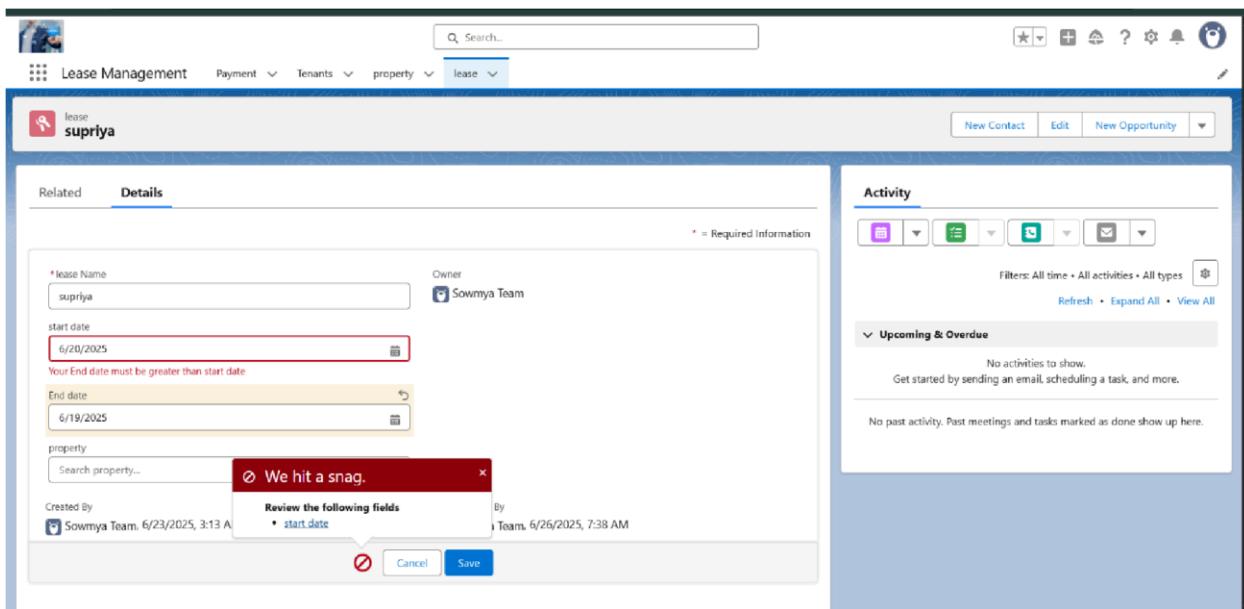
FUNCTIONAL AND PERFORMANCE TESTING

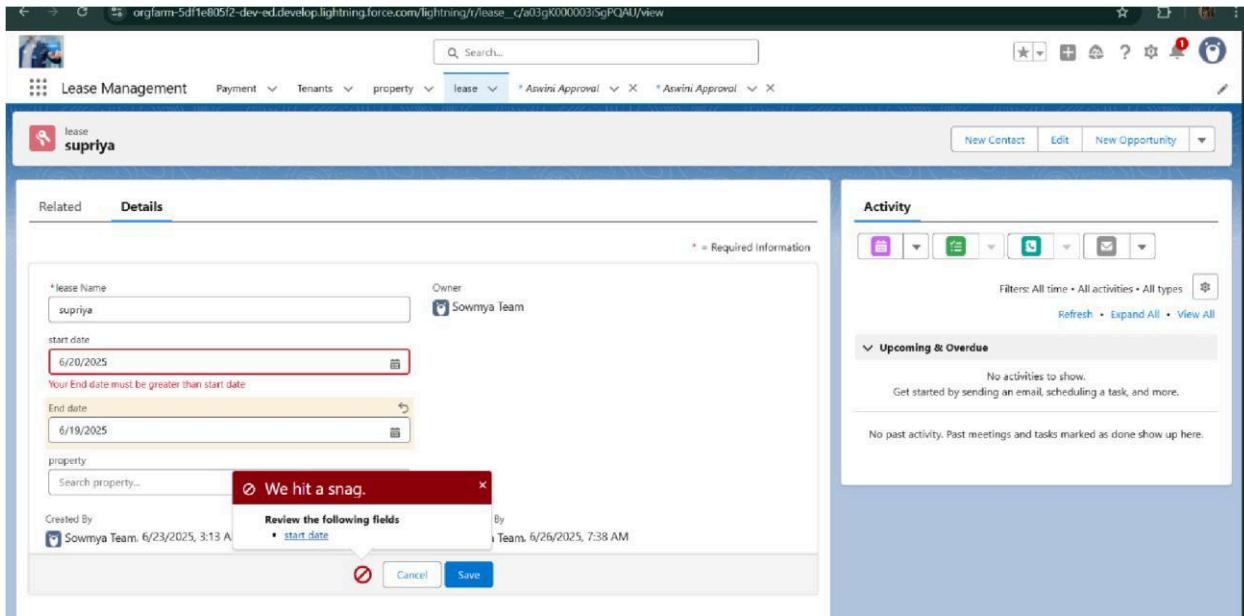
Performance Testing

- Trigger validation by entering duplicate tenant–property records

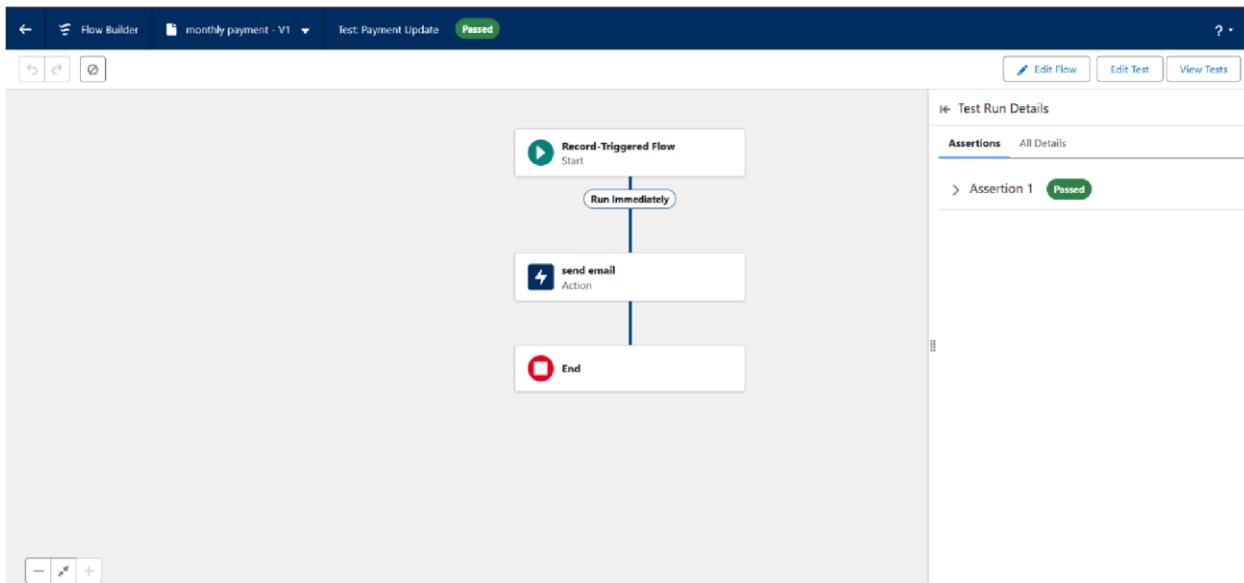


- Validation Rule checking





- Test flows on payment update



- Approval process validated through email alerts and status updates

The screenshot shows a CRM interface for 'Lease Management'. On the left, a 'Tenant' record for 'niranjan' is displayed under the 'Tenants' tab. The 'Details' tab is selected, showing fields for Tenant Name (niranjan), Email (niranjan1506@gmail.com), Phone, status (Stay), and property (Parkside Lofts). The record was created by 'Sowmya Team' on Jun 23, 2025, at 2:33 AM, and last modified by the same team at 3:58 AM. On the right, a 'Notifications' sidebar is open, listing several recent notifications:

- 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 table of steps taken for the tenant 'niranjan':

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

Below the approval history is a 'Payment' section showing two entries for 'Jack' and 'Rahul'. On the right, a sidebar provides options for 'New Contact', 'Edit', and 'New Opportunity'.

RESULTS

Output Screenshots

- Tabs for Property, Tenant, Lease, Payment

The screenshot shows the Salesforce Setup interface under the 'Tabs' section. It displays a table for 'Custom Object Tabs' with four entries:

Action	Label	Tab Style	Description
Edit Del	Lease	Keys	
Edit Del	Payment	Credit card	
Edit Del	property	Sack	
Edit Del	Tenants	Map	

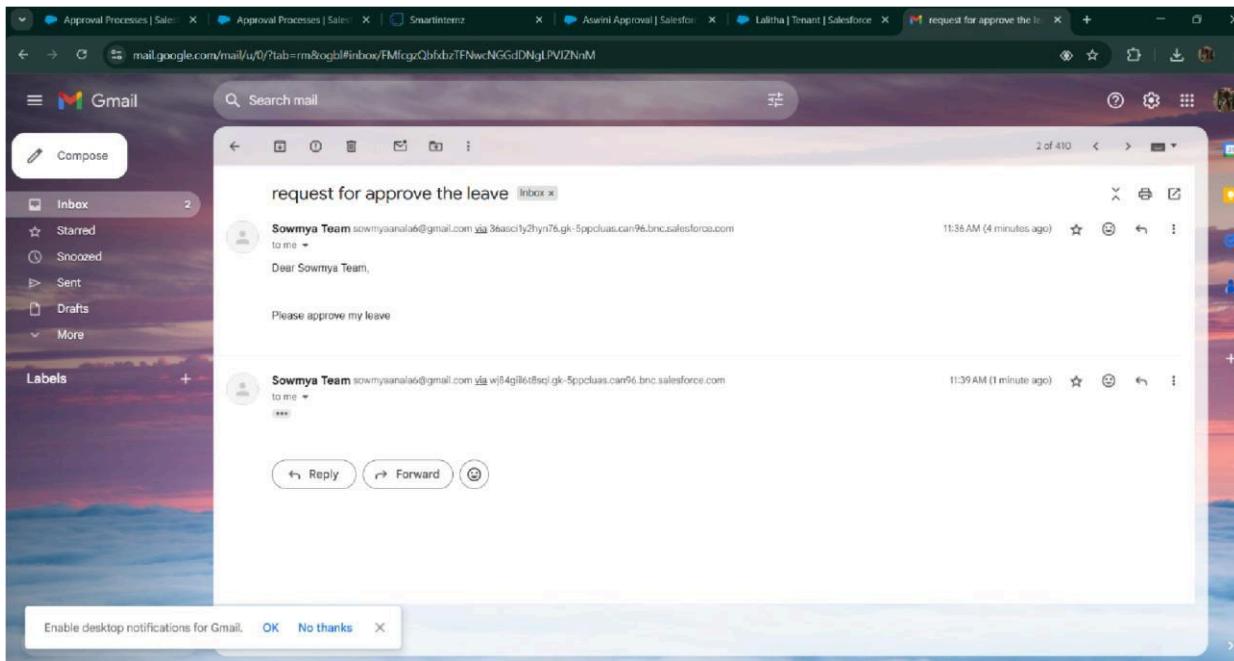
Below this are sections for 'Web Tabs' and 'Visualforce Tabs', both currently empty.

- Email alerts

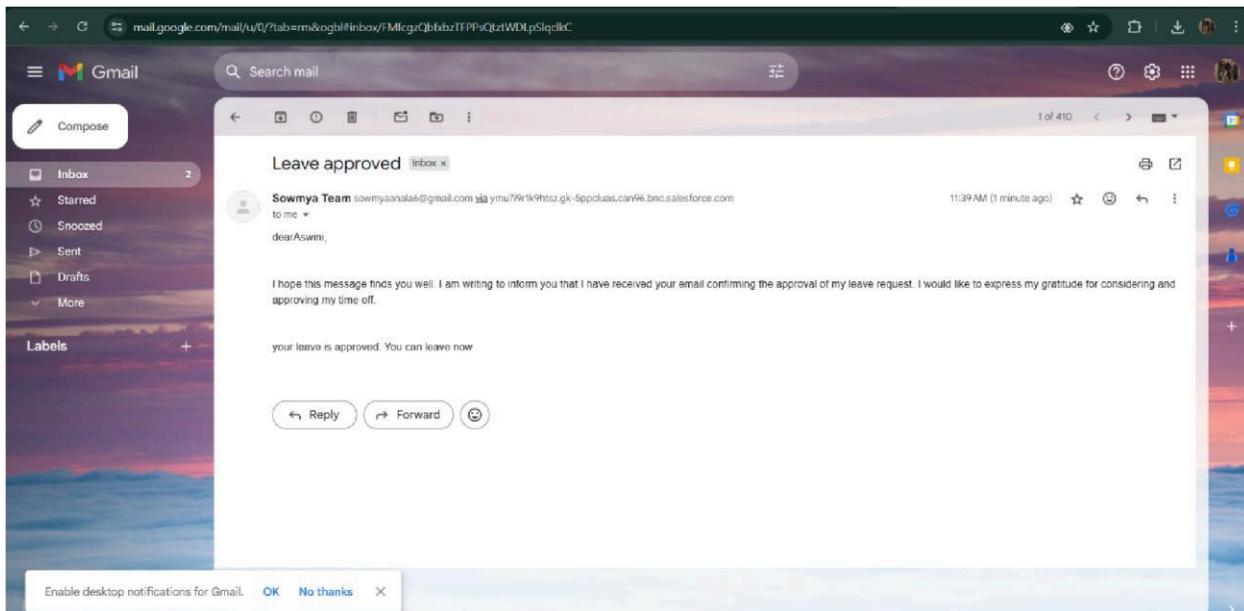
The screenshot shows the 'Lease Management' application interface. In the top navigation bar, 'Tenants > nitinjan' is selected. The main area displays the 'Approval History' for leave requests, showing the following data:

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

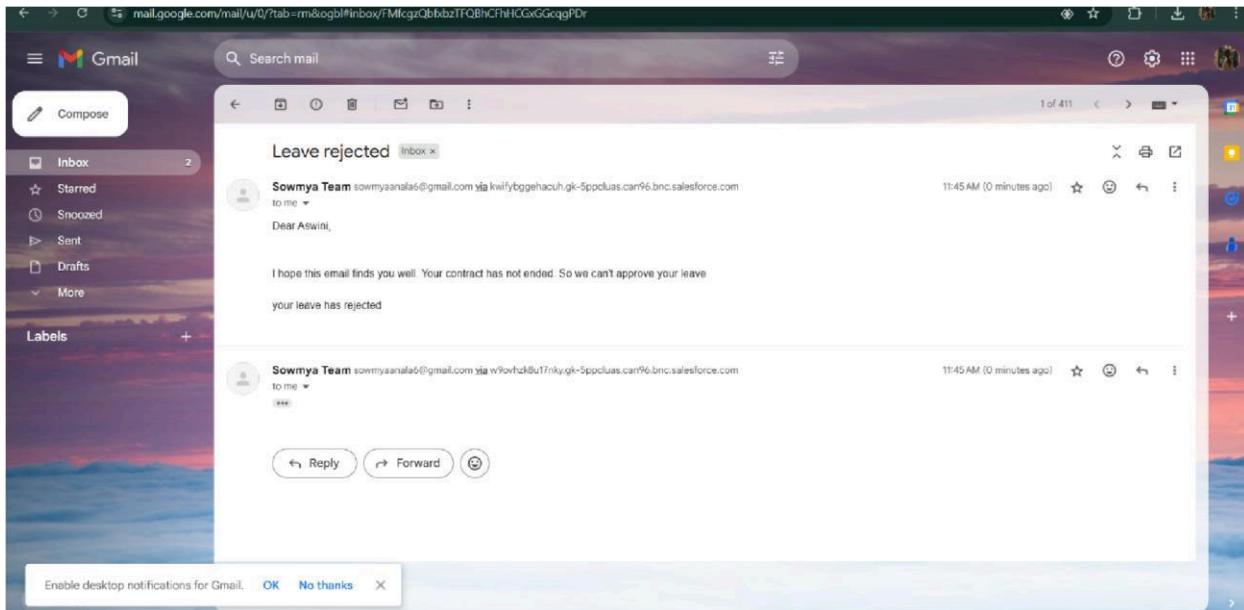
- Request for approve the leave



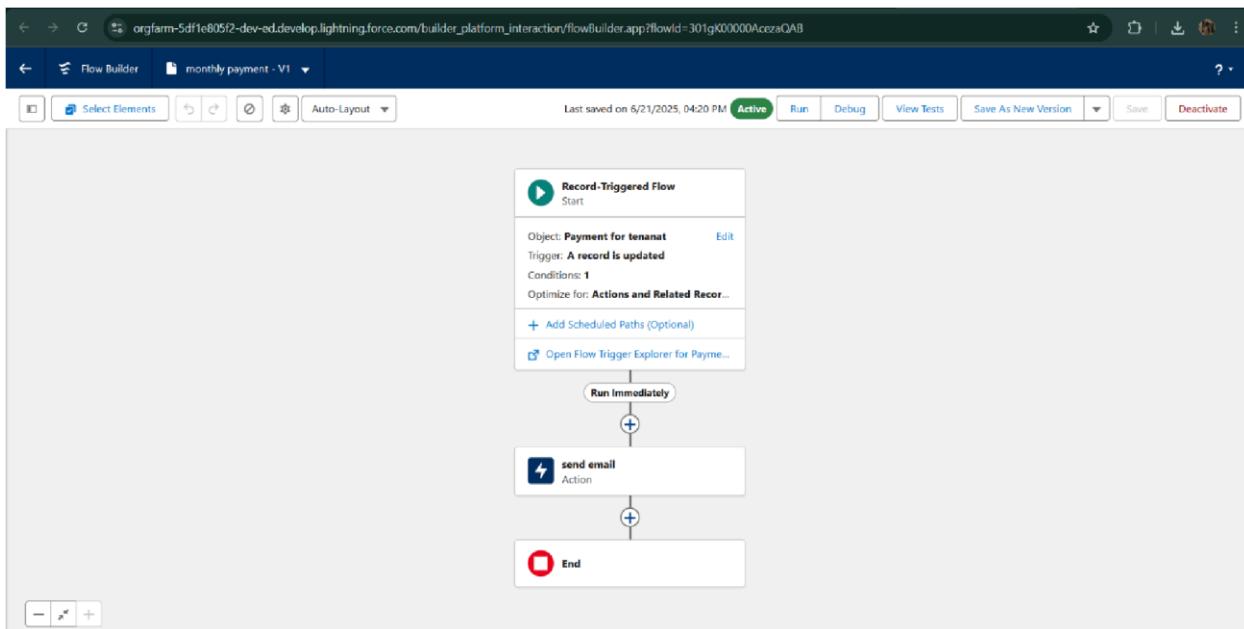
- Leave approved



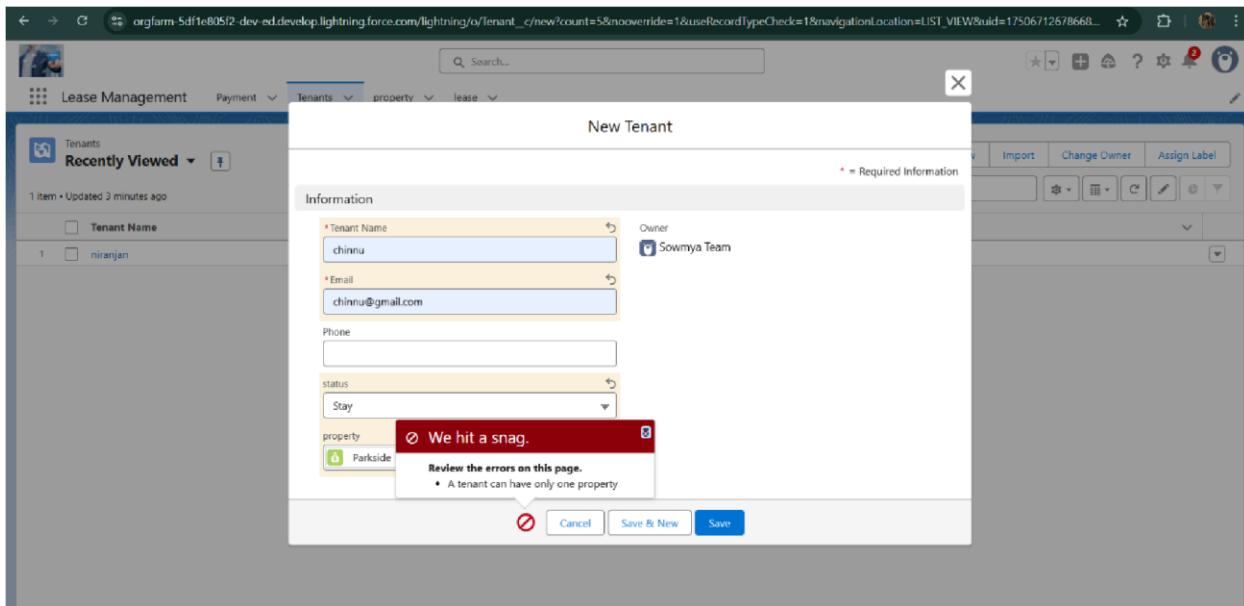
- Leave rejected



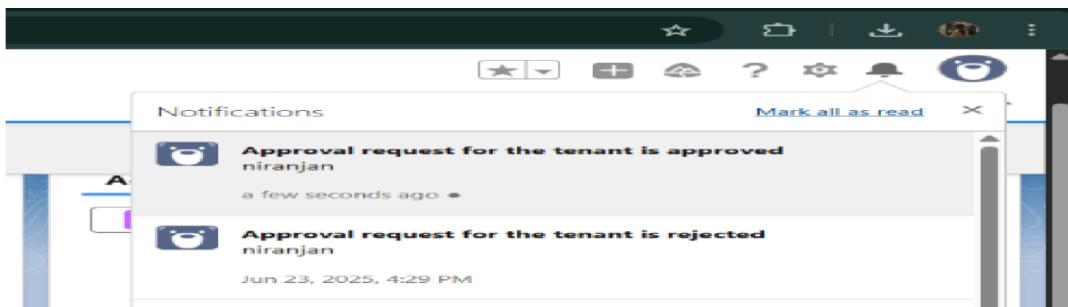
- Flow runs



- 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});
```

