

PROJECT TITLE

College Name: LRG COLLEGE FOR WOMEN

College Code :bru07

TEAM ID: NM2025TMID28009

TEAM MEMBERS:

Team LeaderName: Sandhiya S

Email: sandhiyaspsm@gmail.com

Team Member1: ShreeDharshini S

Email: shreeshreedharshini@gmail.com

Team Member: Saranya K

Email: saranyakumar.sarojini@gmail.com

Team Member: Harini P

Email: harini8041@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.



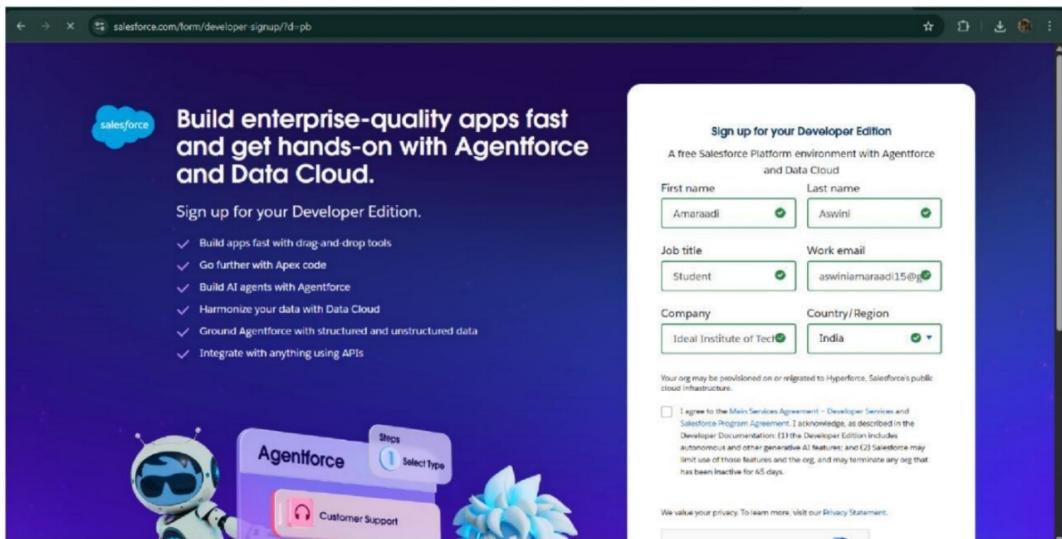
1.2 Purpose

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

The image displays two screenshots of the Salesforce Object Manager interface, showing the configuration details for the 'Tenant' and 'lease' objects.

Tenant Object Configuration:

- API Name:** Tenant_c
- Custom:** ✓
- Singular Label:** Tenant
- Plural Label:** Tenants
- Description:** (empty)
- Enable Reports:** ✓
- Track Activities:** ✓
- Track Field History:** ✓
- Deployment Status:** Deployed
- Help Settings:** Standard salesforce.com Help Window

lease Object Configuration:

- API Name:** lease_c
- Custom:** ✓
- Singular Label:** lease
- Plural Label:** lease
- Description:** (empty)
- Enable Reports:** ✓
- Track Activities:** ✓
- Track Field History:** ✓
- Deployment Status:** Deployed
- Help Settings:** Standard salesforce.com Help Window

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes 'Setup', 'Home', and 'Object Manager'. The main title is 'Payment for tenant'. On the left, a sidebar lists various configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The main content area is titled 'Details' and contains the following information:

Field	Value
API Name	Payment_for_tenant__c
Custom	✓
Singular Label	Payment for tenant
Plural Label	Payments
Description	
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

- 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 'property'. On the left, a sidebar lists various configuration options: Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Scoping Rules. The main content area is titled 'Fields & Relationships' and displays a table of fields:

Fields & Relationships 9 items, Sorted by Field Label				
FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Name	Name__c	Text(25)		
Owner	OwnerId	Lookup(User,Group)	✓	
property	property__c	Lookup(property)	✓	
property Name	Name	Text(30)	✓	
sfq	sfq__c	Text(18)		
Type	Type__c	Picklist		

SETUP > OBJECT MANAGER
Payment for tenant

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

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					
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 Settings

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:

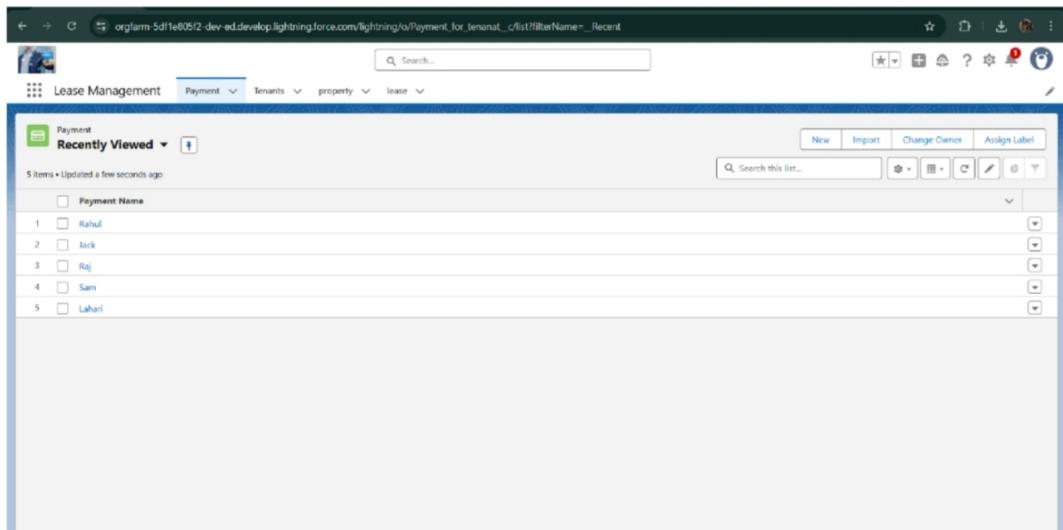
Primary Color Hex Value: #0070D2

Org Theme Options: Use the app's image and color instead of the org's custom theme

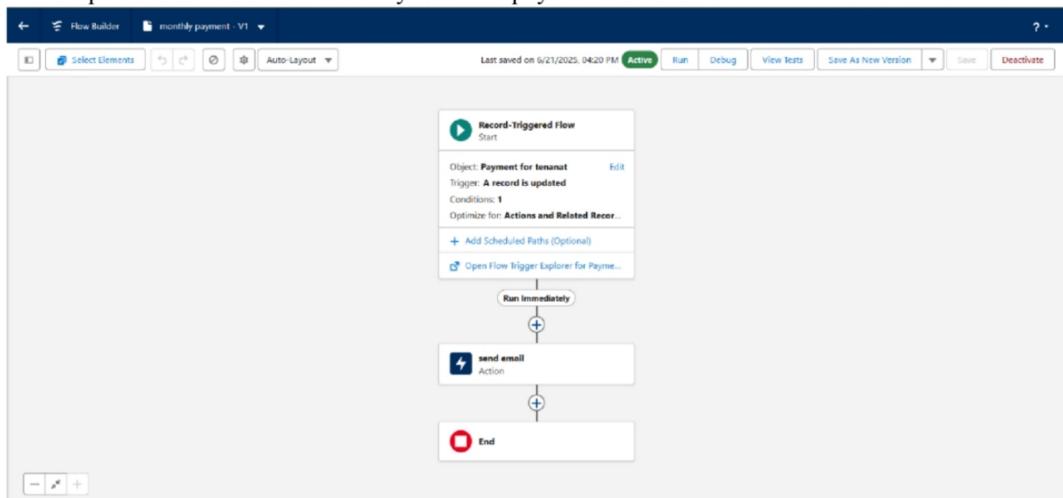
App Launcher Preview

The screenshot shows the 'Navigation Items' section of the Lightning App Builder. On the left, a sidebar lists 'App Settings' (App Details & Branding, App Options, Utility Items (Desktop Only)), 'Navigation Items' (selected), and 'User Profiles'. The main area is titled 'Navigation Items' with a sub-instruction: '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.' It features two lists: 'Available Items' (Accounts, Activation Targets, Activations, All Sites, Alternative Payment Methods, Analytics, App Launcher, Appointment Categories, Appointment Invitations, Approval Requests) and 'Selected Items' (Payment, Tenants, property, lease). Navigation arrows allow items to be moved between the lists.

The screenshot shows the 'User Profiles' section of the Lightning App Builder. The sidebar on the left shows 'App Settings' (App Details & Branding, App Options, Utility Items (Desktop Only)), 'Navigation Items' (selected), and 'User Profiles' (selected). The main area is titled 'User Profiles' with a sub-instruction: 'Choose the user profiles that can access this app.' It displays two lists: 'Available Profiles' (Analytics Cloud Integration User, Analytics Cloud Security User, Anypoint Integration, Authenticated Website, B2B Reordering Portal Buyer Profile, Contract Manager, Custom Marketing Profile, Custom Sales Profile, Custom Support Profile, Customer Community Login User) and 'Selected Profiles' (System Administrator). Navigation arrows allow profiles to be moved between the lists.



- Implemented Flows for monthly rent and payment success



- To create a validation rule to a Lease Object

Validation Rule Edit

Role Name: lease_end_date

Active:

Description:

Error Condition Formula:

```
Example: Discount_Percent_c > 30 More Examples...
Display an error if Discount is more than 30%
```

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

Insert Field Insert Operator

Check Syntax

Quick Tips: Operators & Functions

Functions: ABS, ACOS, ADDMONTHS, AND, ASCII, ASIN, Insert Selected Function, ABS(number)

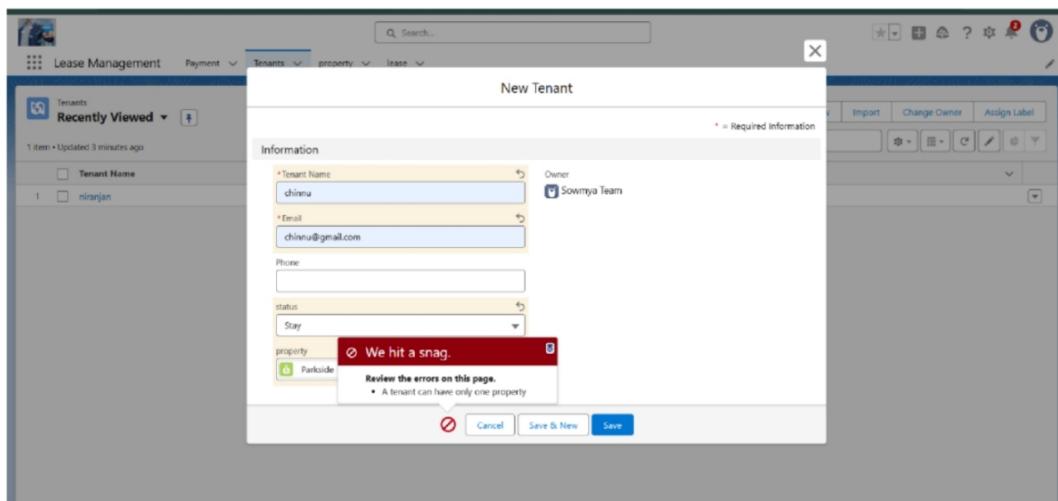
Help on this function: Returns the absolute value of a number, a number without its sign

lease Validation Rule

Validation Rule Detail

Role Name:	lease_end_date	Active:	<input checked="" type="checkbox"/>
Error Condition Formula:	End_date_c <= start_date_c	Error Location:	start_date
Error Message:	Your End date must be greater than start date	Description:	
Created By:	Severin Team, 6/19/2025, 5:37 AM	Modified By:	Severin Team, 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 Scheduleable {
2+
3+     global void execute(ScheduleDataContext 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+ }
```

Log Test Checkstyle Query Editor Plan Rule Response Problem

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

The screenshot shows the Salesforce Setup interface with the 'Email' tab selected. Under 'Classic Email Templates', the 'Leave approved' template is displayed. The template detail page includes fields like Email Template Name (Leave approved), Template Unique Name (Leave_approved), Encoding (Unicode (UTF-8)), Author (Screamer Team [Channe]), and Description (Leave request approved). The email body preview shows a message to the recipient confirming approval.

The screenshot shows the Salesforce Setup interface with the 'Email' tab selected. Under 'Classic Email Templates', the 'tenant leaving' template is displayed. The template detail page includes fields like Email Template Name (tenant leaving), Template Unique Name (tenant_leaving), Encoding (Unicode (UTF-8)), Author (Screamer Team [Channe]), and Description (Leave request for approval). The email body preview shows a message asking for leave approval.

The screenshot shows the Salesforce Setup interface with the 'Email' section selected. Under 'Email', 'Classic Email Templates' is chosen. A search bar at the top left shows 'email template'. The main content area displays the 'Leave rejected' template details.

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	Sonam's Team (Chennai)
Description	
Created By	Sonam's Team
Modified By	Sonam's Team
Last Used Date	6/20/2025, 1:11 AM
Times Used	

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 'Email' section selected. Under 'Email', 'Classic Email Templates' is chosen. A search bar at the top left shows 'email template'. The main content area displays the 'Tenant Email' template details.

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	Sonam's Team (Chennai)
Description	
Created By	Sonam's Team
Modified By	Sonam's Team
Last Used Date	6/20/2025, 1:12 AM
Times Used	

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 'Classic Email Templates' page open. The template 'tenant payment' is selected, showing its details and a preview of the email content.

Email Template Detail:

- Email Template Name: tenant payment
- Template Unique Name: insert_payment
- Encoding: Unicode (UTF-8)
- Author: Sowmya Team (Change)
- Description:
- Created By: Sowmya Team (6/20/2025, 1:13 AM)
- Modified By: Sowmya Team (6/20/2025, 1:13 AM)

Email Template 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 'Approval Processes' page open. An approval process named 'TenantApproval' is selected, showing its details and configuration steps.

Process Definition Detail:

- Process Name: TenantApproval
- Unique Name: TenantApproval
- Description:
- Entry Criteria: Tenant__c.status equals Stay
- Record Eligibility: Administrator ONLY
- Next Automated Approver Determined by:
- Allow Submitters to Recall Approval Requests:
- Initial Submitter: Tenant Owner
- Created By: Sowmya Team (6/23/2025, 3:41 AM)
- Modified By: Sowmya Team (6/26/2025, 11:57 PM)

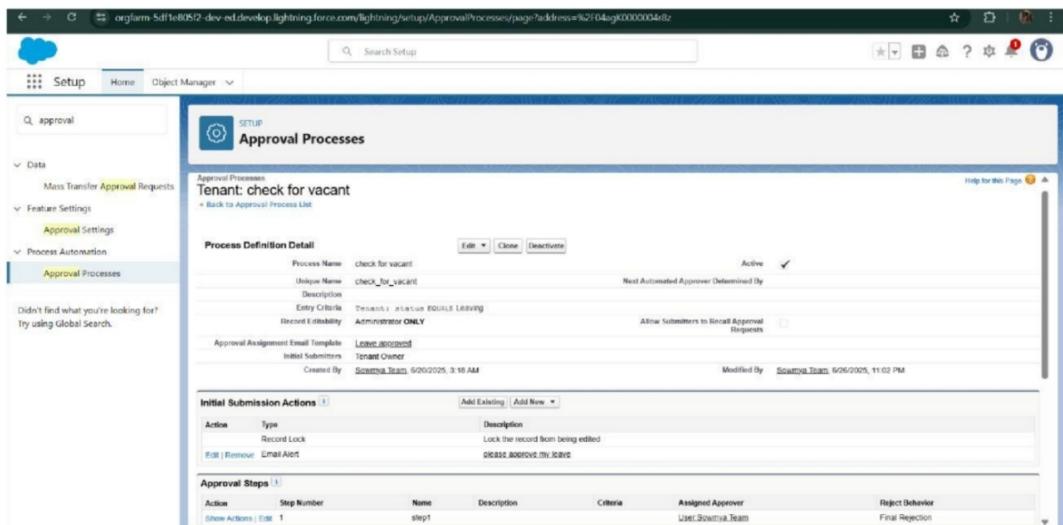
Initial Submission Actions:

- Action Type: Record Lock
- Description: Lock the record from being edited

Approval Steps:

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

For Check for Vacant:



- Apex Trigger

Create an Apex Trigger

```

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

```

```

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

```

Create an Apex Handler class

```

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

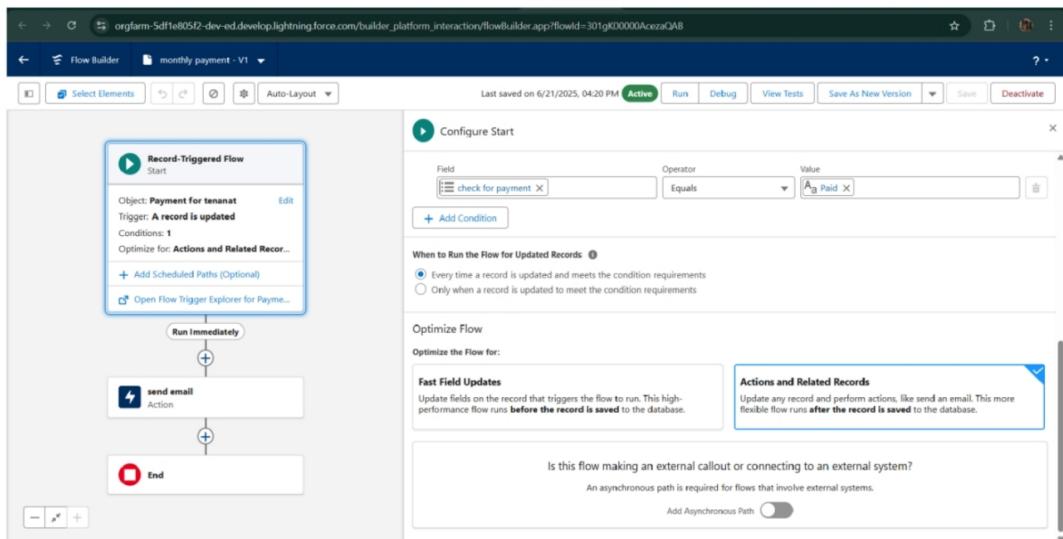
```

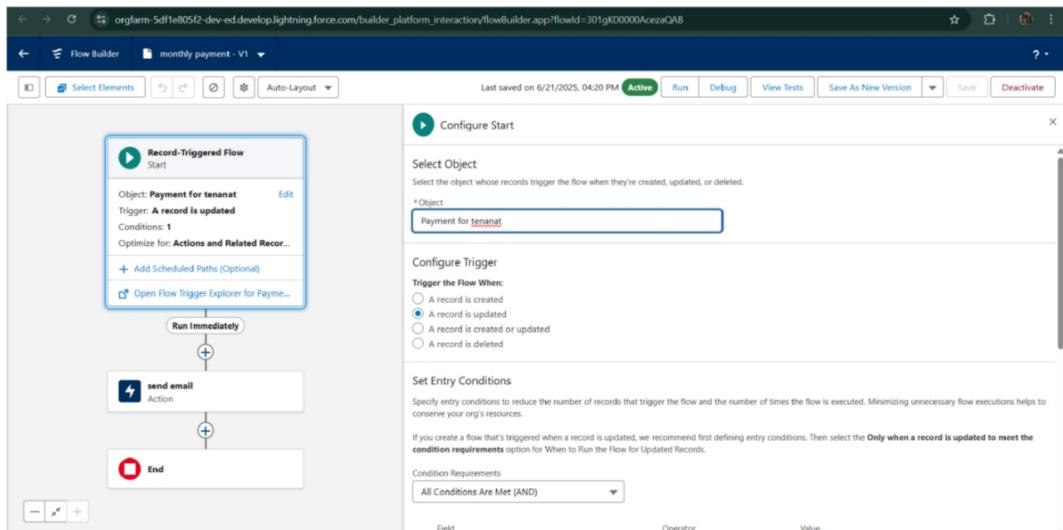
```

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             newTenant.addError('A tenant can have only one property');
20
21         }
22     }
23
}

```

● 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;
23

```

```

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              sendMonthlyEmail();
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 reminderEmail = 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 soon; payment ensures the smooth functioning of our rental arrangement and helps maintain a positive living environment for all.';
24
25          String emailSubject = 'Reminder: Monthly Rent Payment Due';
26
27          Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
28
29          mail.setToAddresses(tenant.Email__c);
30
31          mail.setSubject(emailSubject);
32
33          mail.setPlainText(emailContent);
34
35          Messaging.SingleEmailMessage[] emails = new Messaging.SingleEmailMessage[1];
36
37          emails[0] = mail;
38
39          Messaging.sendEmail(new Messaging.SingleEmailMessage[] {emails});
40      }
41  }
42 }

```

Schedule Apex class

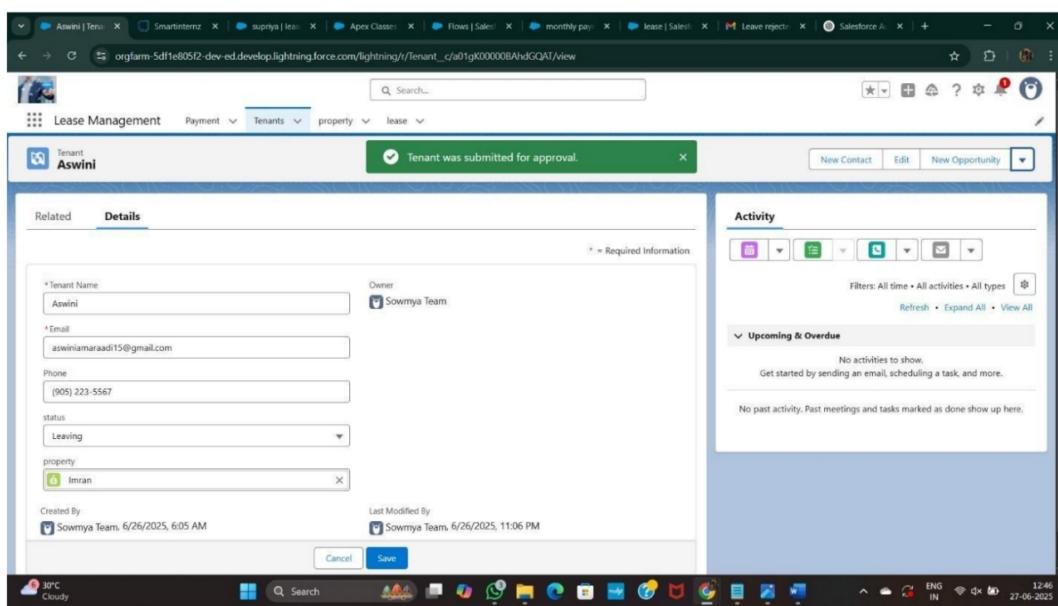
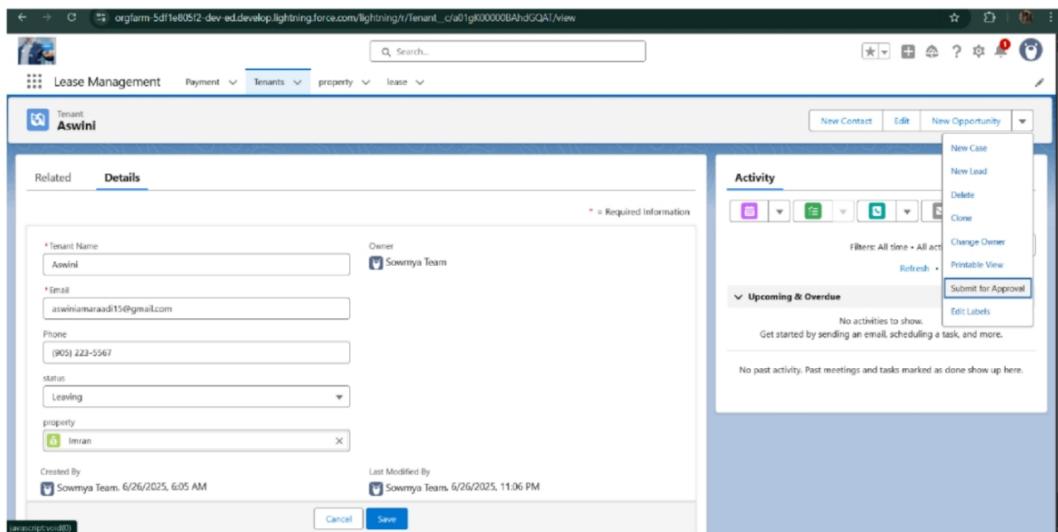
The screenshot shows the Salesforce Setup interface with the following details:

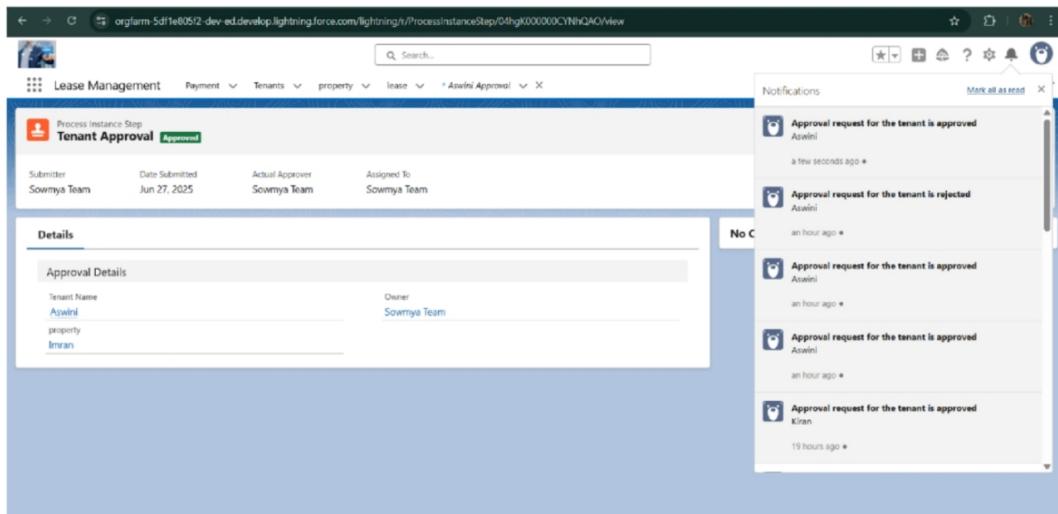
- Search Bar:** Search term: apex
- Left Navigation:**
 - Email
 - Apex Exception Email
 - Custom Code
 - Apex Classes (selected)
 - Apex Settings
 - Apex Test Execution
 - Apex Test History
 - Apex Triggers
 - Environments
 - Jobs
 - Apex Flex Queue
 - Apex Jobs
- Apex Classes Page:**
 - Apex Class Detail:** Name: MonthlyEmailScheduler, Namespace Prefix: None, Status: Active, Last Modified By: Soomya Team, Last Modified: 6/23/2025, 2:47 AM.
 - Code View:**

```

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              sendMonthlyEmail();
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 reminderEmail = 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 soon; payment ensures the smooth functioning of our rental arrangement and helps maintain a positive living environment for all.';
24
25          String emailSubject = 'Reminder: Monthly Rent Payment Due';
26
27          Messaging.SingleEmailMessage mail = new Messaging.SingleEmailMessage();
28
29          mail.setToAddresses(tenant.Email__c);
30
31          mail.setSubject(emailSubject);
32
33          mail.setPlainText(emailContent);
34
35          Messaging.SingleEmailMessage[] emails = new Messaging.SingleEmailMessage[1];
36
37          emails[0] = mail;
38
39          Messaging.sendEmail(new Messaging.SingleEmailMessage[] {emails});
40      }
41  }
42 }

```





FUNCTIONAL AND PERFORMANCE TESTING

Performance Testing

- Trigger validation by entering duplicate tenant-property records

New Tenant

Information

- *Tenant Name: chinu
- *Email: chinu@gmail.com
- Owner: Sowmya Team
- Property: Parkside

We hit a snag.

Review the errors on this page.

- A tenant can have only one property

Cancel Save & New Save

- Validation Rule checking

Lease Management

Lease: supriya

Related Details

Lease Name: supriya

Owner: Sowmya Team

Start date: 6/18/2025

Your End date must be greater than start date

End date: 6/19/2025

Property: Search property...

We hit a snag.

Review the following fields

- start_date

By j Teams, 6/26/2025, 7:38 AM

Cancel Save

Activity

New Contact Edit New Opportunity

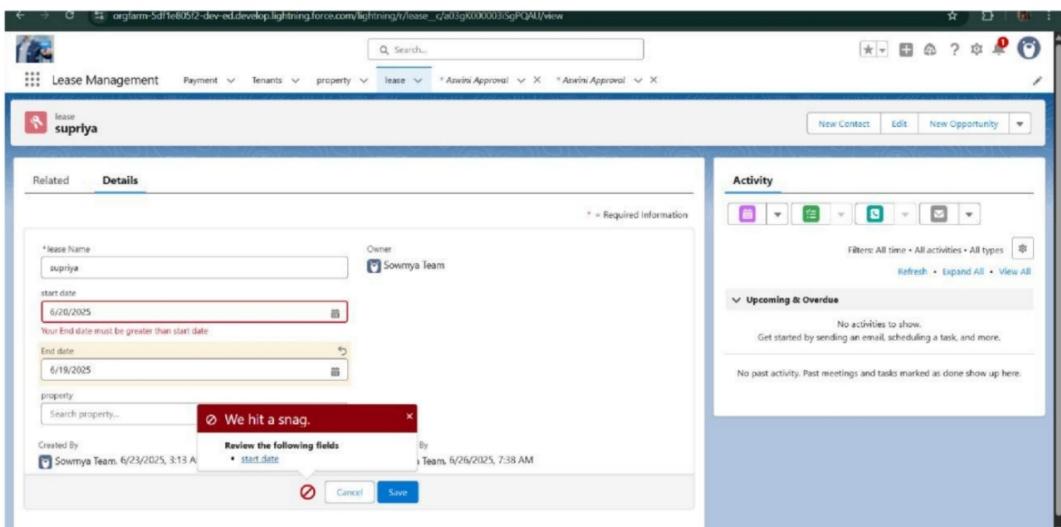
Filters: All time • All activities • All types

Upcoming & Overdue

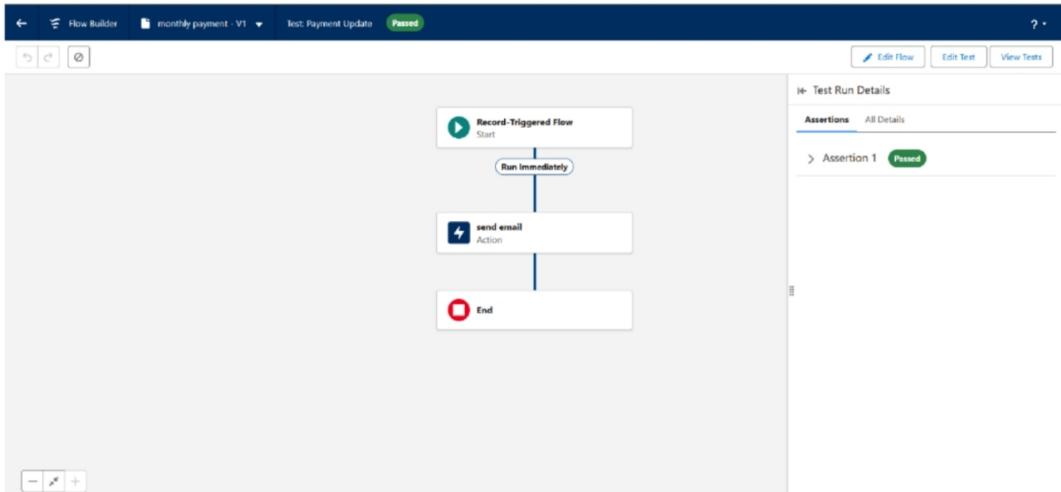
No activities to show.

Get started by sending an email, scheduling a task, and more.

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



- Test flows on payment update



- Approval process validated through email alerts and status updates

The screenshot shows the 'Lease Management' module in Microsoft Dynamics 365. On the left, the 'Details' tab of a tenant record for 'niranjan' is displayed. The record includes fields for Tenant Name (niranjan), Owner (Sowmya Team), Email (niranjan1506@gmail.com), Phone, Status (Stay), and Property (Parkade Lofts). It also shows the creation date (Jun 23, 2025, 2:33 AM) and last modified date (Jun 23, 2025, 3:58 AM). On the right, a 'Notifications' sidebar is open, showing a list of recent activity items:

- 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, 1:03 PM

The screenshot continues from the previous one, showing the same tenant record for 'niranjan'. Below the main details, there are two sections: 'Approval History (6+)' and 'Payment (2+)'. The Approval History table shows the following steps:

Step Name	Date	Status	Assigned To
Step 1	6/23/2025, 5:39 AM	Approved	Sowmya Team
Approval Request Submitted	6/23/2025, 5:39 AM	Submitted	Sowmya Team
Step 1	6/23/2025, 2:59 AM	Rejected	Sowmya Team
Approval Request Submitted	6/23/2025, 2:58 AM	Submitted	Sowmya Team
Step 1	6/23/2025, 2:55 AM	Approved	Sowmya Team
Approval Request Submitted	6/23/2025, 3:55 AM	Submitted	Sowmya Team

The Payment section shows two entries: 'Jack' and 'Rahul'.

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	Keys	Standard	
Edit Del	Credit Card	Standard	
Edit Del	Bank	Standard	
Edit Del	Tenant	Standard	

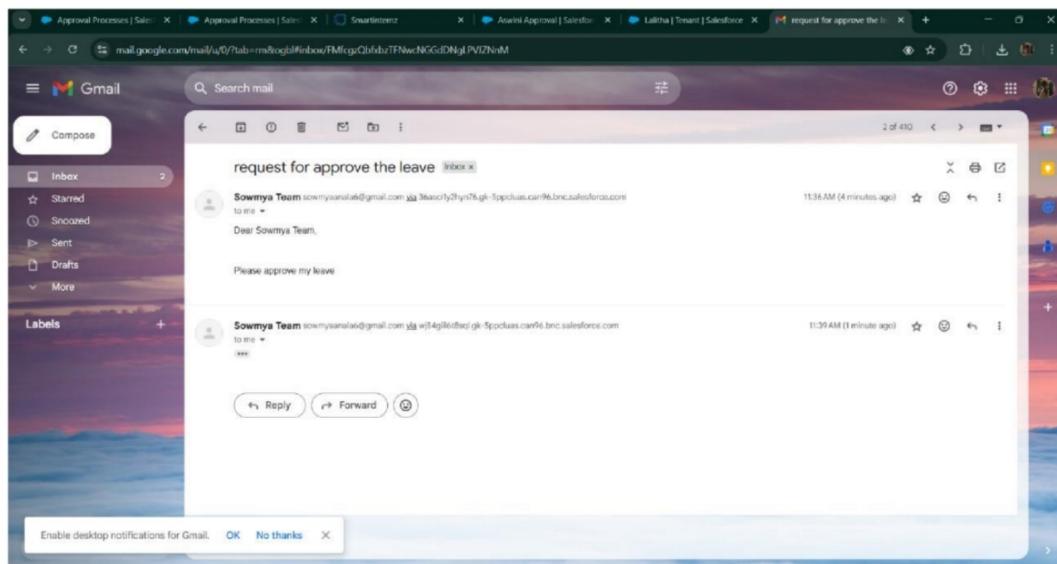
Below this, sections for 'Web Tabs' and 'Visualforce Tabs' show no items defined.

- Email alerts

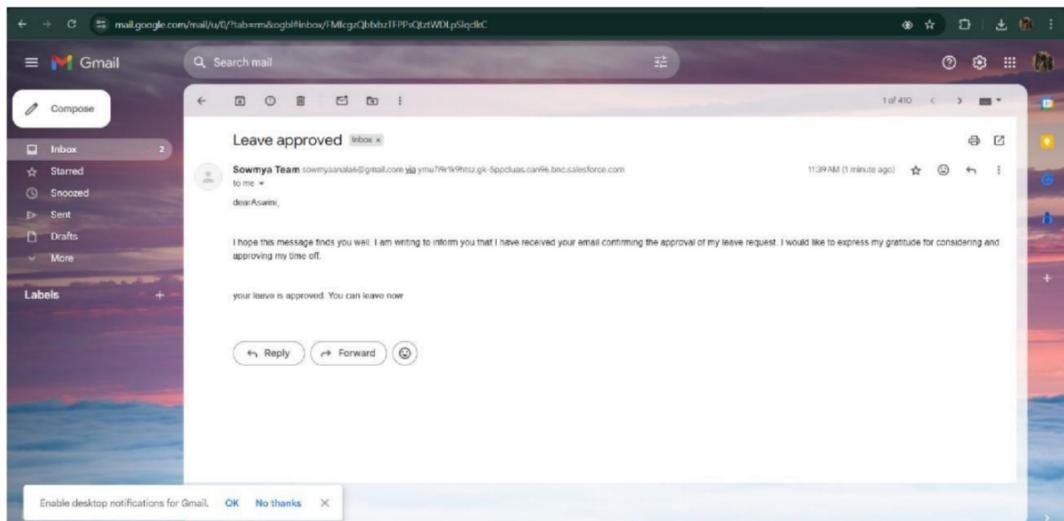
The screenshot shows the 'Lease Management' page with a 'Tenants' tab selected. Below it, the 'Approval History' section displays a table of leave requests:

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

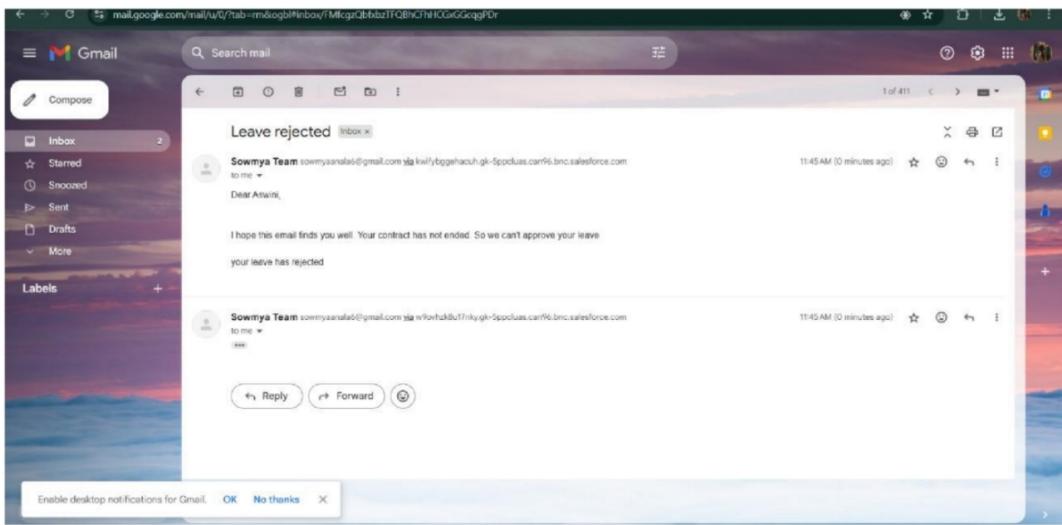
- Request for approve the leave



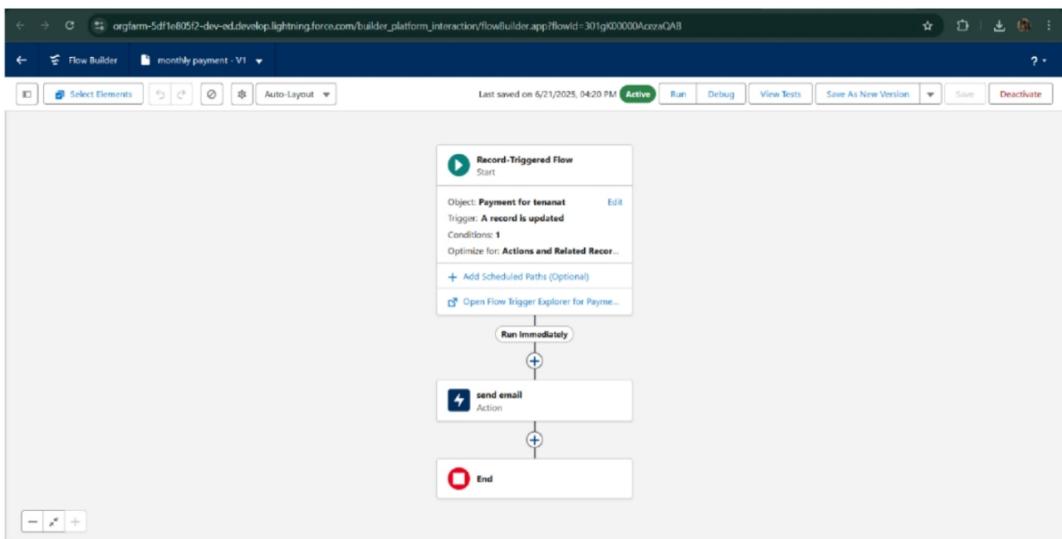
- Leave approved



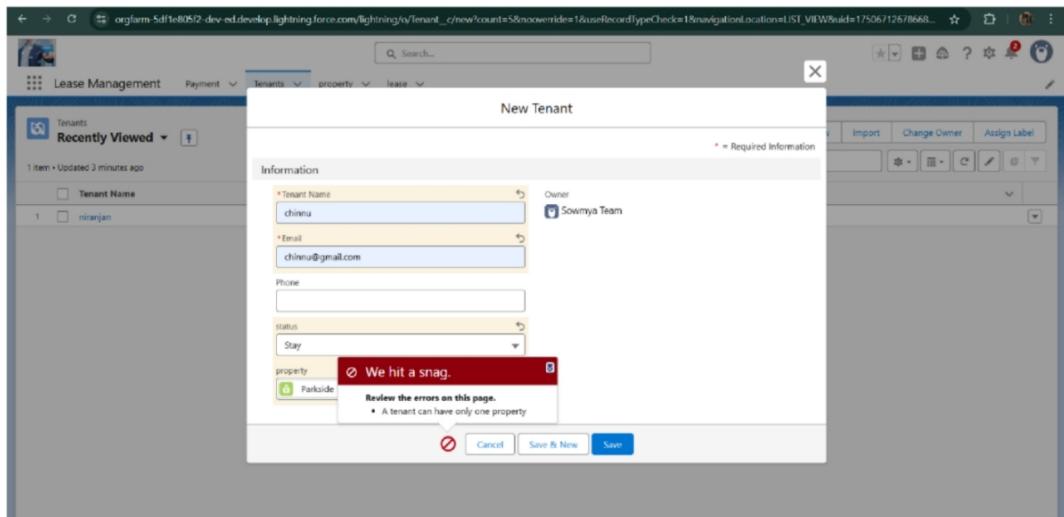
- Leave rejected



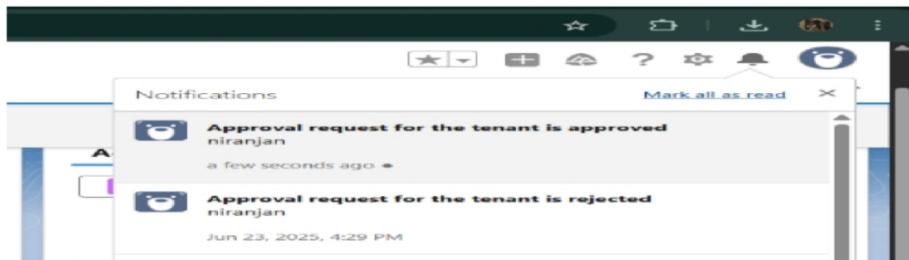
- Flow runs



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

    }

}
}

```

MothlyEmailScheduler.apxc:

```

global class MonthlyEmailScheduler implements Schedulable {

    global void execute(SchedulableContext sc) { Integer
currentDay = Date.today().day(); if (currentDay == 1) {
sendMonthlyEmails();

}

} public static void
sendMonthlyEmails() { List<Tenant__c>
tenants = [SELECT Id, Email__c FROM
Tenant__c]; for (Tenant__c tenant :
tenants) {

    String recipientEmail = tenant.Email__c;
    String emailContent = 'I trust this email finds you well. I am writing to remind you
that the monthly rent is due Your timely payment ensures the smooth functioning of our
rental arrangement and helps maintain a positive living environment for all.';

    String emailSubject = 'Reminder: Monthly Rent Payment Due';

```

```
Messaging.SingleEmailMessage email = new  
Messaging.SingleEmailMessage(); email.setToAddresses(new  
String[]{recipientEmail}); email.setSubject(emailSubject);  
email.setPlainTextBody(emailContent);  
Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});  
}  
}  
}
```