

# **PROJECT TITLE:**

**Garage Management System**

**College Name: KAYPEEYES COLLEGE OF ARTS & SICENCE**

**College Code: bru5j**

**TEAM ID:** NM2025TMID27468

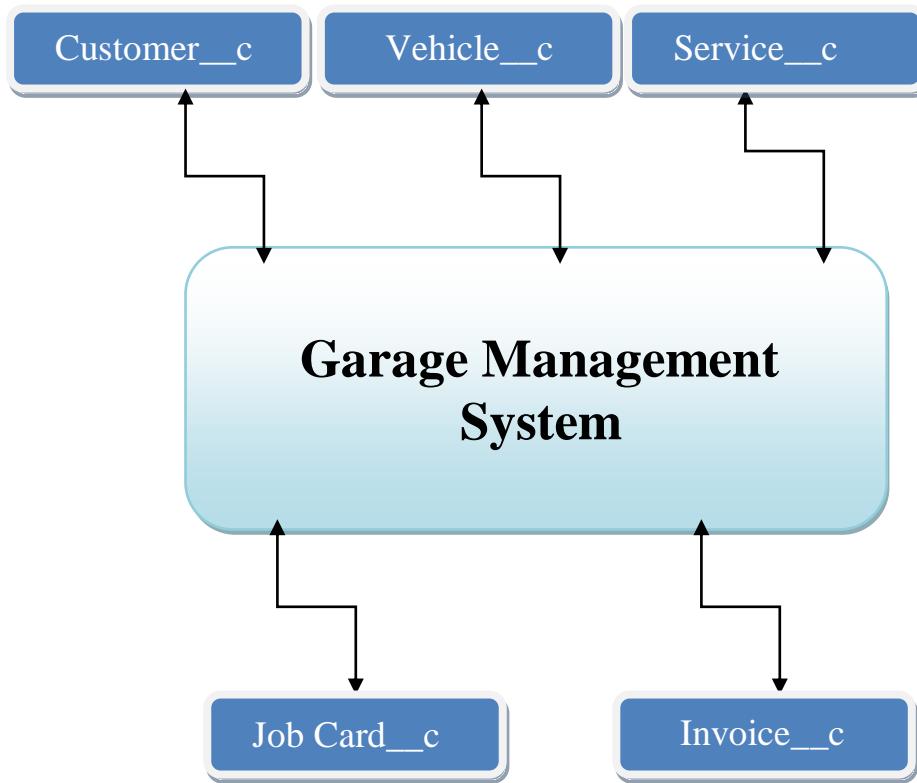
**TEAM MEMBERS:** 1

# 1. INTRODUCTION

## 1.1 Project Overview

The **Garage Management System (GM System)** is a software solution designed to streamline and automate the operations of a vehicle service garage or workshop. Traditional garage operations, such as vehicle check-ins, job card creation, service tracking, billing, and inventory management, are often managed manually, leading to inefficiencies, errors, and delays.

This system provides an integrated digital platform for managing all aspects of a garage, ensuring improved productivity, better customer service, and accurate record-keeping. It allows garage owners and employees to efficiently handle service requests, assign mechanics, track spare parts inventory, generate invoices, and maintain a detailed history of vehicles and customers.



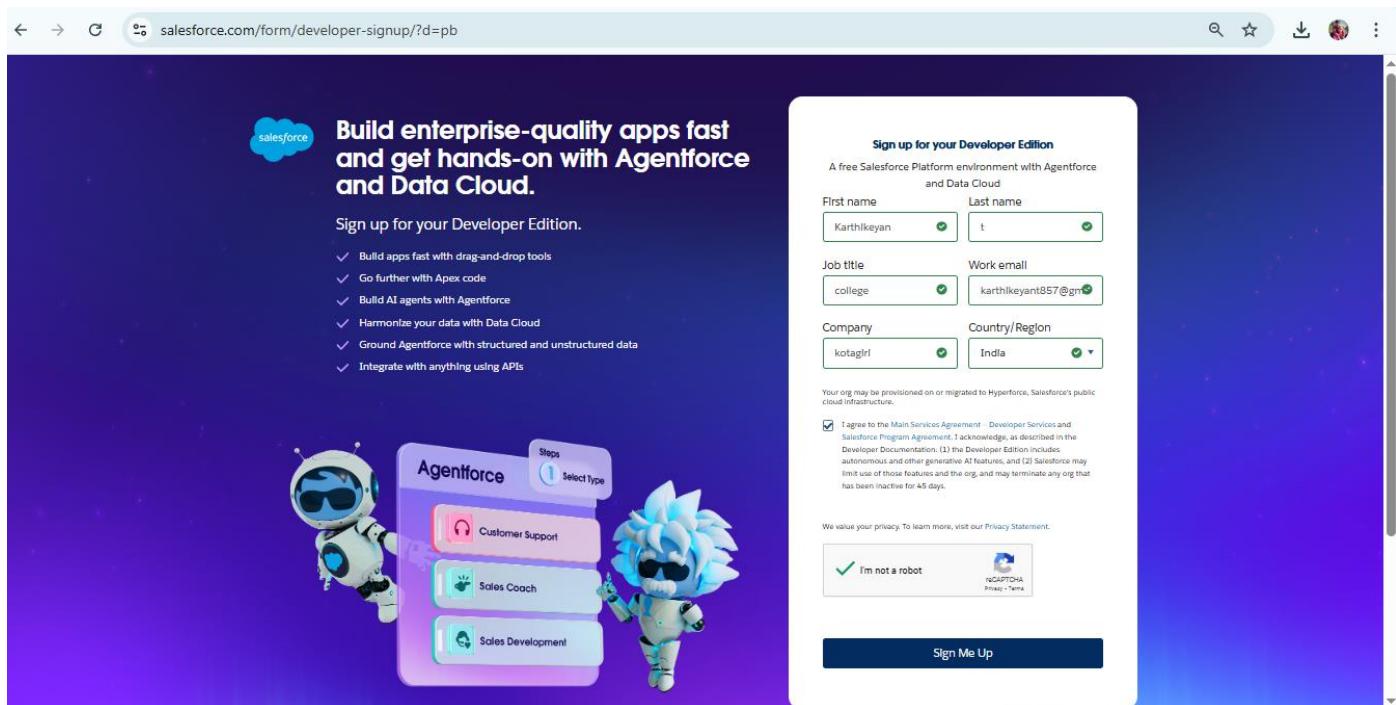
## 1.1 Purpose:

The purpose of the **Garage Management System (GMS)** is to provide a comprehensive and automated solution to manage all activities of a garage or automobile service center. The system is designed to replace manual record-keeping with a digital platform, ensuring accuracy, speed, and efficiency in daily operations. It aims to simplify the process of managing customer details, vehicle information, service requests, job cards, billing, and inventory, while reducing errors and saving time.

# DEVELOPMENT PHASE

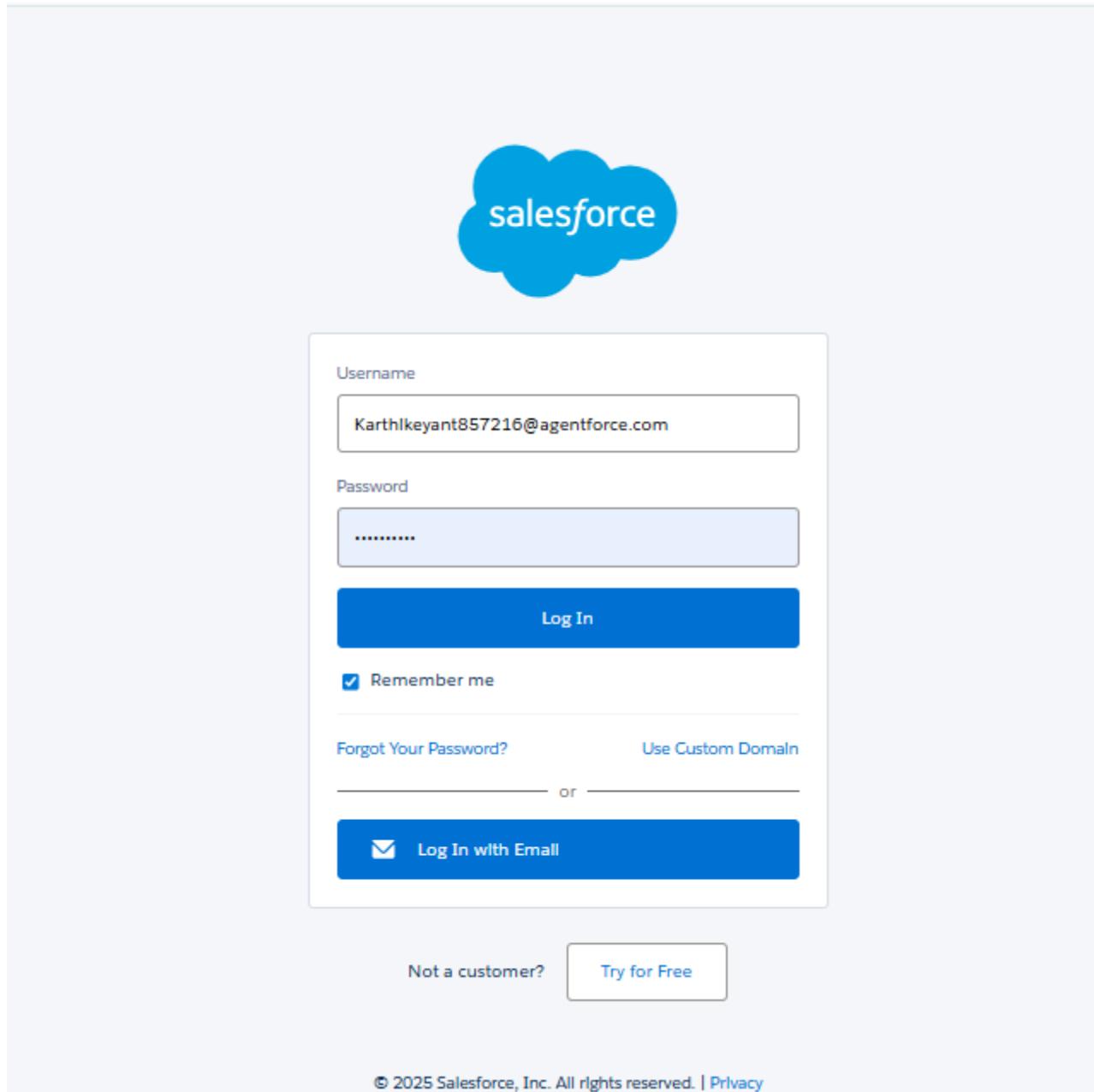
## Creating Developer Account:

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



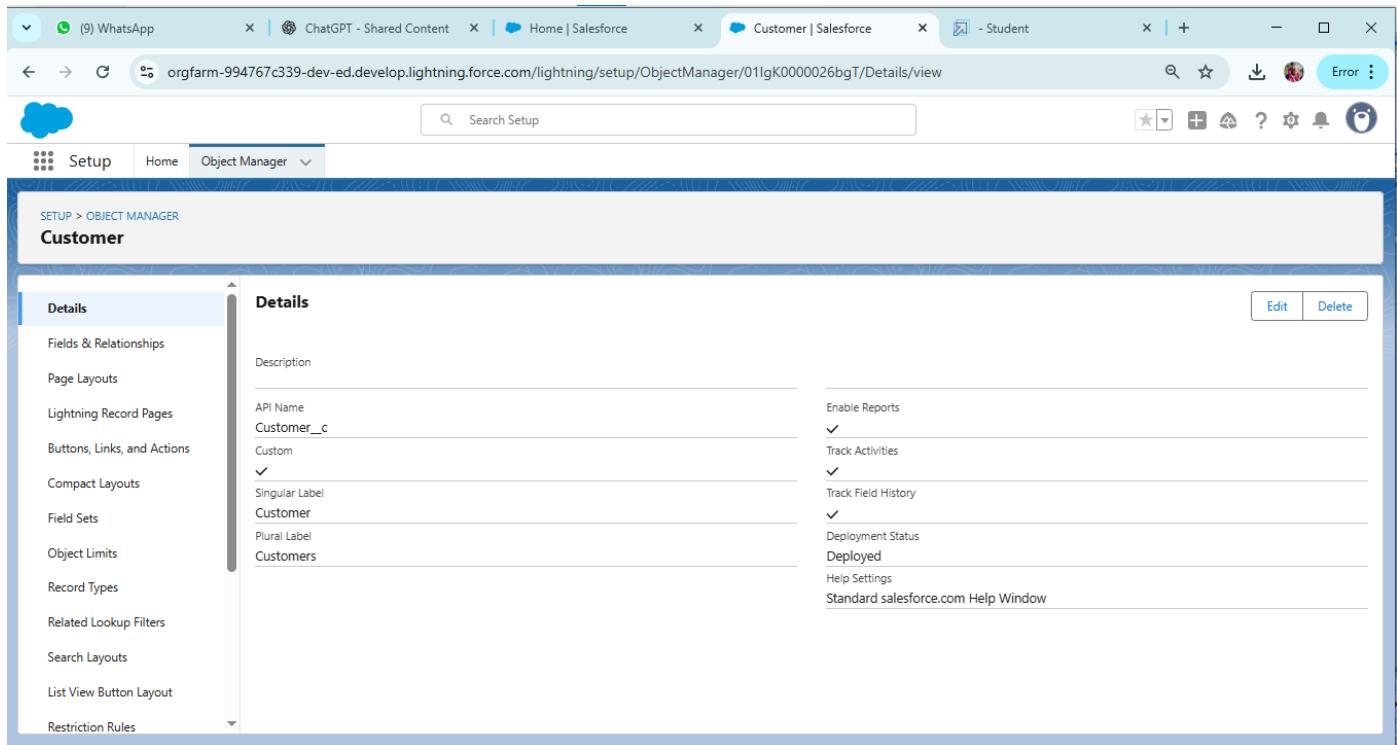
**Account activation then next login this link:**

<https://login.salesforce.com/?locale=in>



The image shows the Salesforce login page. At the top is the iconic blue cloud logo with the word "salesforce" in white. Below it is a light gray rectangular form. The first field is labeled "Username" and contains the email "Karthlkeyant857216@agentforce.com". The second field is labeled "Password" and contains several dots as a placeholder. Below these fields is a large blue "Log In" button with white text. To the left of the "Log In" button is a checkbox labeled "Remember me" with a checked mark. Below the "Log In" button are two links: "Forgot Your Password?" and "Use Custom Domain". A horizontal line with the word "or" in the center separates these links from another blue button below. This second button has a mail icon and the text "Log In with Email". At the bottom of the form, there are two links: "Not a customer?" on the left and "Try for Free" on the right, both enclosed in small rectangular boxes.

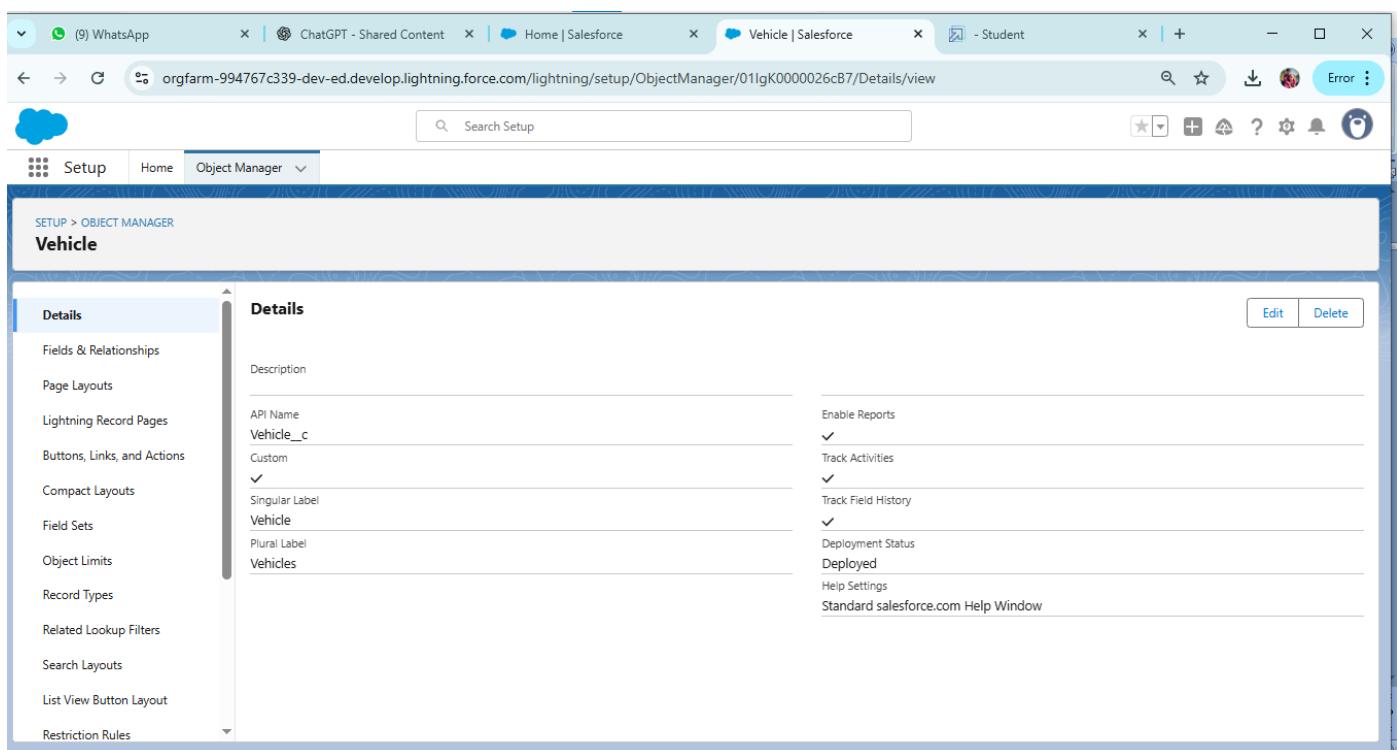
- **Created objects:** Customer, Vehicle, Service, Job Card and Invoice.



The screenshot shows the Salesforce Object Manager interface for the 'Customer' object. The left 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, and Restriction Rules. The main 'Details' tab displays the following information:

Setting	Value
Description	
API Name	Customer_c
Custom	✓
Singular Label	Customer
Plural Label	Customers
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

At the bottom right of the main area are 'Edit' and 'Delete' buttons.



The screenshot shows the Salesforce Object Manager interface for the 'Vehicle' object. The left sidebar lists the same configuration options as the Customer screen. The main 'Details' tab displays the following information:

Setting	Value
Description	
API Name	Vehicle_c
Custom	✓
Singular Label	Vehicle
Plural Label	Vehicles
Enable Reports	✓
Track Activities	✓
Track Field History	✓
Deployment Status	Deployed
Help Settings	Standard salesforce.com Help Window

At the bottom right of the main area are 'Edit' and 'Delete' buttons.

Screenshot of the Salesforce Object Manager for the Service object.

**Service**

**Details**

**Description**

**API Name**: Service\_\_c

**Custom**: ✓

**Singular Label**: Service

**Plural Label**: Services

**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**

Screenshot of the Salesforce Object Manager for the Job Card object.

**Job Card**

**Details**

**Description**

**API Name**: JobCard\_\_c

**Custom**: ✓

**Singular Label**: Job Card

**Plural Label**: Job Cards

**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 for the 'Invoice' object. The left sidebar lists various configuration tabs: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main pane displays the 'Details' tab for the 'Invoice' object. It includes sections for Description, API Name (Invoice\_\_c), Singular Label (Invoice), Plural Label (Invoices), Enable Reports (checked), Track Activities (checked), Track Field History (checked), Deployment Status (Deployed), Help Settings, and Standard salesforce.com Help Window.

- **Configured Fields and Relationships:**

The screenshot shows the Salesforce Object Manager interface for the 'Customer' object. The left sidebar lists configuration tabs: Details, Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main pane displays the 'Fields & Relationships' tab, which lists four items: Created By (CreatedBy), Customer Name (Name), Last Modified By (LastModifiedBy), and Owner (OwnerId). The 'INDEXED' column indicates that the 'Owner' field is indexed.

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Customer Name	Name	Text(80)		✓
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓

Screenshot of the Salesforce Setup interface showing the Object Manager for the 'Vehicle' object.

The left sidebar shows navigation links under '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 main content area displays the 'Fields & Relationships' section for the 'Vehicle' object, listing four items:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Vehicle Name	Name	Text(80)		✓

At the bottom, the Windows taskbar shows various pinned apps and the system clock indicates 10:37 PM on 9/7/2025.

Screenshot of the Salesforce Setup interface showing the Object Manager for the 'Service' object.

The left sidebar shows navigation links under '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 main content area displays the 'Fields & Relationships' section for the 'Service' object, listing four items:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Service Name	Name	Text(80)		✓

At the bottom, the Windows taskbar shows various pinned apps and the system clock indicates 10:38 PM on 9/7/2025.

Screenshot of the Salesforce Setup interface showing the Object Manager for the Job Card object.

The left sidebar shows the navigation path: SETUP > OBJECT MANAGER > Job Card.

The main content area displays the "Fields & Relationships" section. It lists four items, sorted by Field Label:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
JobCard Name	Name	Text(80)		✓
Last Modified By	LastModifiedBy	Lookup(User)		✓
Owner	OwnerId	Lookup(User,Group)		✓

The browser address bar shows the URL: <https://orgfarm-994767c339-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgK0000026cPd/FieldsAndRelationships/view>.

Screenshot of the Salesforce Setup interface showing the Object Manager for the Invoice object.

The left sidebar shows the navigation path: SETUP > OBJECT MANAGER > Invoice.

The main content area displays the "Fields & Relationships" section. It lists four items, sorted by Field Label:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Invoice Name	Name	Text(80)		✓
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓

The browser address bar shows the URL: <https://orgfarm-994767c339-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgK0000026cW5/FieldsAndRelationships/view>.

## Developed Lightning App with relevant tabs:

The screenshot shows the 'App Details & Branding' tab in the Lightning App Builder. The left sidebar lists 'App Settings' with 'App Details & Branding' selected. The main area contains fields for 'App Name' (Lease Management), 'Developer Name' (Lease\_Management), and a 'Description' box (Application to efficiently handle the processes related to leasing real estate properties.). It also includes sections for 'App Details', 'App Branding' (with a preview image and color #0070D2), and 'Org Theme Options'. A preview of the app launcher is shown at the bottom.

The screenshot shows the 'Navigation Items' tab in the Lightning App Builder. The left sidebar lists 'App Settings' with 'Navigation Items' selected. The main area displays 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 between the lists allow items to be moved.

Lightning App Builder | App Settings | Pages | Lease Management | Help

**User Profiles**

Available Profiles

Selected Profiles

System Administrator

Choose the user profiles that can access this app.

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

(9) WhatsApp - Student | Recently Viewed | NM\_studend\_c/list?filterName=\_Recent | Lightning Usage | Salesforce

Search...

Home NM\_studends NM\_studends \* Home X

NM\_studends Recently Viewed

9 items • Updated a few seconds ago

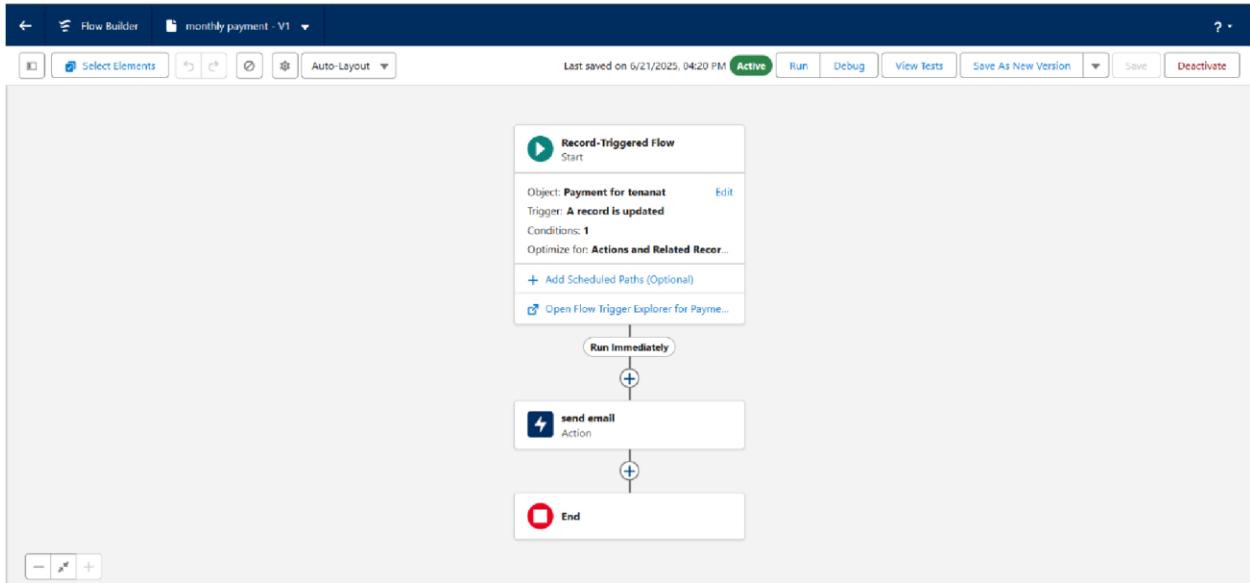
1	Pooja	2	Sathya	3	Pooja	4	Sathya	5	Preethi	6	Pawan	7	Mahalakshmi	8	Karthick	9	Monisha
<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>		<input type="checkbox"/>	

New Import Assign Label

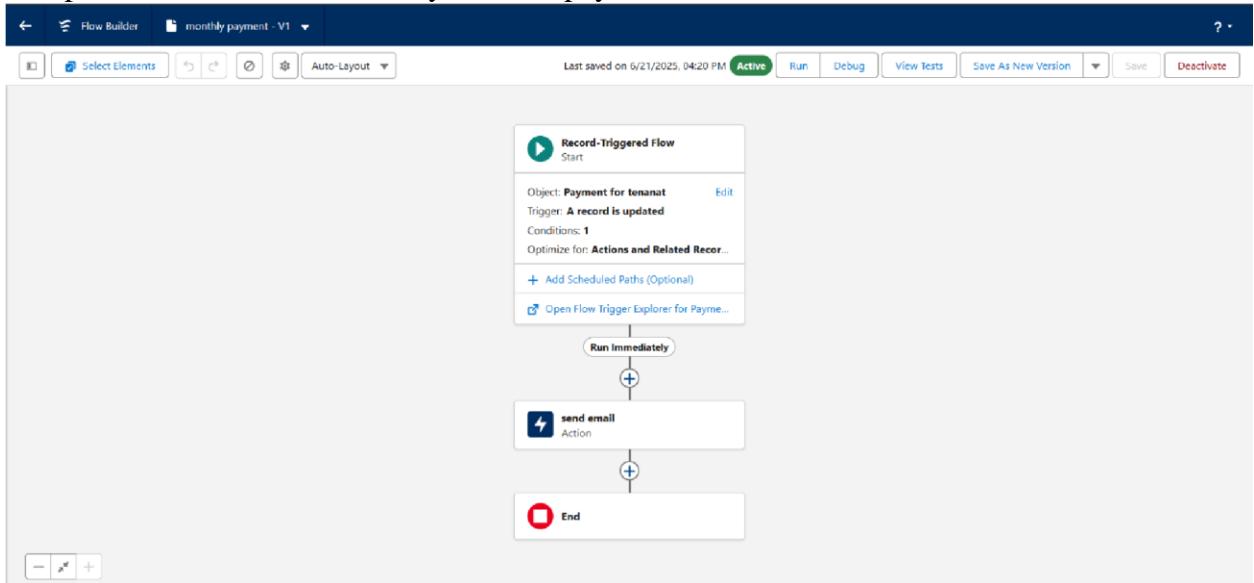
Search this list...

High UV 12:08 AM 9/8/2025

- Implemented Flows for monthly rent and payment success:



- Implemented Flows for monthly rent and payment success



- To create a validation rule to a Lease Object

The screenshot shows the Salesforce Sharing Settings page. The left sidebar has 'Sharing Settings' selected under 'Sharing'. The main area displays sharing rules for different objects:

- NM\_students Sharing Rules**: No sharing rules specified.
- sawthoi Sharing Rules**: No sharing rules specified.
- Service Sharing Rules**: A sharing rule operation is in progress. A warning message states: "A sharing rule operation is in progress. You can't create new owner-based sharing rules for Service targeting the following groups. The initiating user will receive an email when each operation finishes." The details show the rule was initiated by 'Karthikeyan T' on 9/11/2025, 12:05 AM, shared with 'Role: Manager', and has an access level of 'ReadWrite'.
- Vehicle Sharing Rules**: No sharing rules specified.

The screenshot shows the Salesforce Setup interface under the 'Sharing Settings' section. A specific group named 'sales team' is selected. The group details include:

- Label: sales team
- Group Name: sales\_team
- Grant Access Using Hierarchies: checked
- Description: (empty)
- Created By: Karthikeyan T, 9/8/2025, 9:42 PM
- Modified By: Karthikeyan T, 9/10/2025, 11:59 PM

The 'Members' section lists three users:

Name	Type
Sales person	Role
sales_person.artist	Role
artist	Role

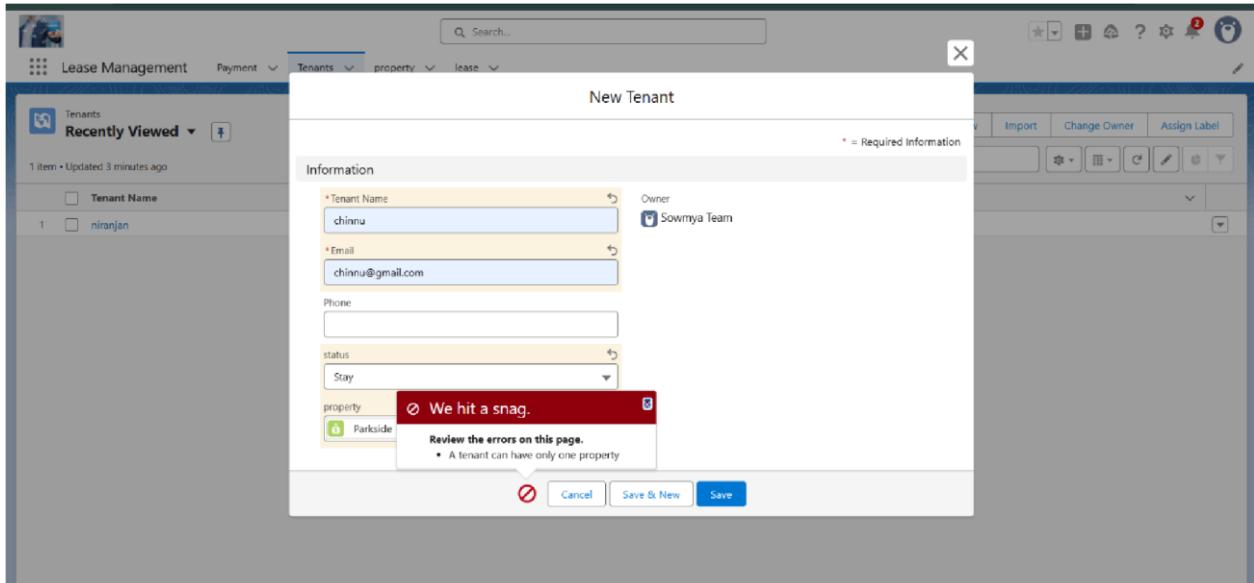
- Added Apex trigger to restrict multiple tenants per property

The screenshot shows the Salesforce Setup interface under the 'Users' section. A user named 'karthikeyan artist' is being edited. The 'General Information' tab displays the following fields:

Field	Value
First Name	karthikeyan
Last Name	artist
Alias	karti
Email	karthikeyan857@gmail.com
Username	karthikeyan857@gmail.com
Nickname	User175732315286321553
Title	(empty)
Company	(empty)
Department	(empty)
Division	(empty)

On the right side, there are several checkboxes for user roles and settings:

- Role: Manager
- User License: Salesforce
- Profile: Manager
- Active: checked
- Marketing User: unchecked
- Offline User: unchecked
- Knowledge User: unchecked
- Flow User: unchecked
- Service Cloud User: unchecked
- Sites.com Contributor User: unchecked
- Sites.com Publisher User: unchecked
- WDC User: unchecked
- Data.com User Type: None
- Data.com Monthly Addition Limit: 300
- Accessibility Mode (Classic Only): unchecked
- High-Contrast Palette on Charts: checked
- Load Lightning Pages While Scrolling: checked
- Debug Mode: unchecked
- Make Setup My Default Landing Page: unchecked
- Salesforce CRM Content User: checked



- 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' section, 'Classic Email Templates' is selected. A search result for 'Leave approved' is displayed. The 'Email Template Detail' section shows the following information:

Field	Value
Email Template Name	Leave approved
Template Unique Name	Leave_approved
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Changed]
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 area contains the following content:

```

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. The 'Email Template Detail' section shows the following information:

Field	Value
Email Template Name	tenant leaving
Template Unique Name	tenant_leaving
Encoding	Unicode (UTF-8)
Author	Sowmya Team [Changed]
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 area contains the following content:

```

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 '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 'Leave rejected'. The 'Email Template Detail' section includes fields like 'Email Template Name' (Leave rejected), 'Template Unique Name' (Leave\_rejected), 'Encoding' (UTF-8), 'Author' (Sowmya Team [Change]), 'Description' (Created By Sowmya Team 6/20/2025, 1:11 AM), and 'Modified By' (Sowmya Team 6/20/2025, 1:11 AM). The 'Available For Use' checkbox is checked. Below this, the 'Email Template' section shows the subject 'Leave rejected', a plain text preview with the message '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.', and a note indicating it's a 'Text Email Template'.

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 Email'. The 'Email Template Detail' section includes fields like 'Email Template Name' (Tenant Email), 'Template Unique Name' (Tenant\_Email), 'Encoding' (UTF-8), 'Author' (Sowmya Team [Change]), 'Description' (Created By Sowmya Team 6/20/2025, 1:12 AM), and 'Modified By' (Sowmya Team 6/20/2025, 1:12 AM). The 'Available For Use' checkbox is checked. Below this, the 'Email Template' section shows the subject 'Urgent: Monthly Rent Payment Reminder', a plain text preview with the message '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.', and a note indicating it's a 'Text Email Template'.

**Email Template Detail**

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

**Email Template**

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:

**Process Definition Detail**

- Process Name: TenantApproval
- Unique Name: TenantApproval
- Description: Tenant: status equals Stay
- Entry Criteria: Tenant: status equals Stay
- Record Editability: Administrator ONLY
- Active: checked
- Next Automated Approver Determined By: [empty]
- Allow Submitters to Recall Approval Requests: unchecked

**Initial Submission Actions**

Action Type	Description
Record Lock	Lock the record from being edited

**Approval Steps**

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 following details:

- Approval Processes** page.
- Tenant: check for vacant** process definition.
- 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: Soumya Team
  - Modified By: Soumya Team
  - Created Date: 6/20/2025, 3:18 AM
  - Modified Date: 6/26/2025, 11:02 PM
- Initial Submission Actions** section:
 

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

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

## ● Apex Trigger

### Create an Apex Trigger

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

- trigger test on Tenant\_c (before insert)**
- testHandler** class definition.
- Open** modal window showing the selected entity type as **Triggers**.
- Problems** tab is active.

```

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

```

Developer Console - Google Chrome

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

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 ← property CustomField ← Tenant__c Object ← Tenant__c Object
Triggers			
Pages			
Page Components			
Objects			
Static Resources			
Packages			

Open Filter: Filter the repository (\* = any string) Hide Managed Packages Refresh

Logs Tests Checkpoints Query Editor View State Progress Problems

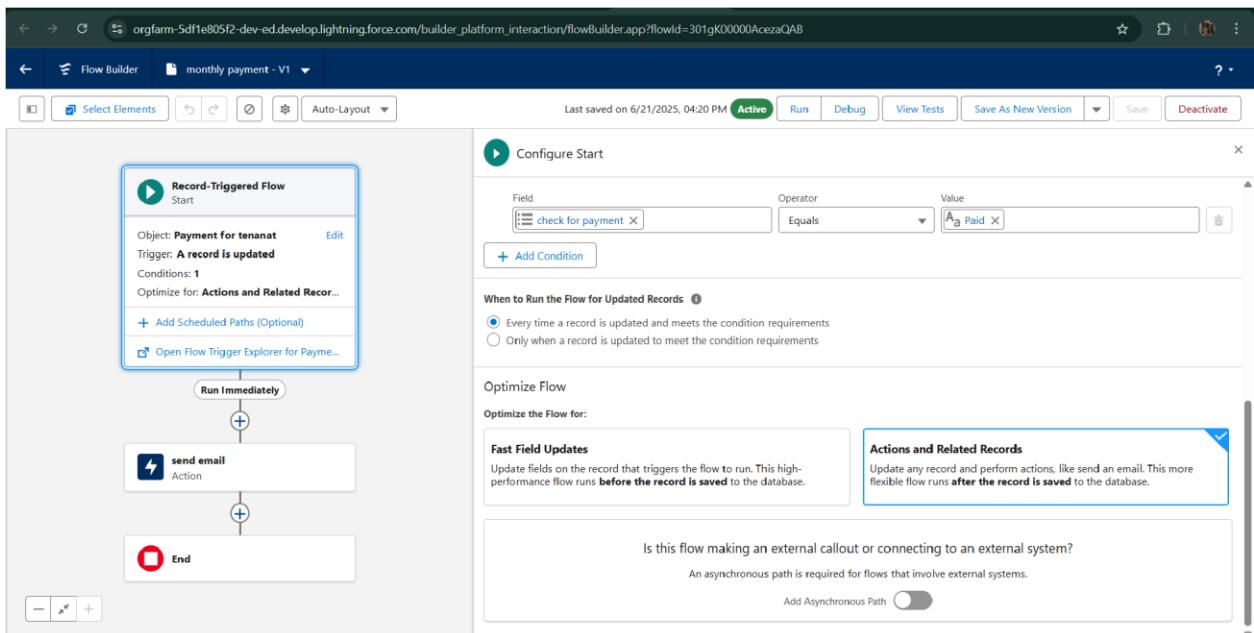
Name Line Problem

```

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 }

```

## ● 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;

```

**Open**

Entity Type	Entities	Related
Entity Type	Name Namespace	Name Extent Direction
Classes	testHandler	Email CrossTrigger References
Triggers	MonthlyEmailScheduler	Tenant__c CustomField References
Pages		Tenant__c SObject References
Page Components		
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

```

Tenant Aswini

Related Details

\* = Required Information

\* Tenant Name: Aswini

Owner: Sowmya Team

\* Email: aswiniamaraadi15@gmail.com

Phone: (905) 223-5567

Status: Leaving

Property: Imran

Created By: Sowmya Team, 6/26/2025, 6:05 AM

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

New Contact Edit New Opportunity

Activity

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.

Tenant Aswini

✓ Tenant was submitted for approval.

Related Details

\* = Required Information

\* Tenant Name: Aswini

Owner: Sowmya Team

\* Email: aswiniamaraadi15@gmail.com

Phone: (905) 223-5567

Status: Leaving

Property: Imran

Created By: Sowmya Team, 6/26/2025, 6:05 AM

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

New Contact Edit New Opportunity

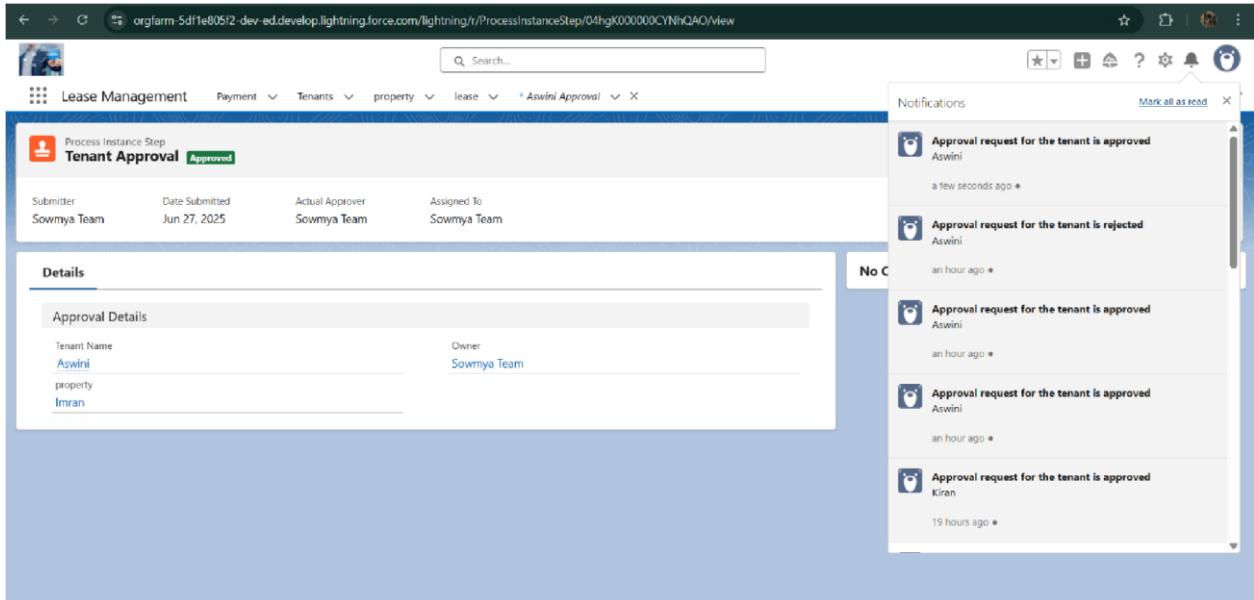
Activity

Upcoming & Overdue

No activities to show.

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

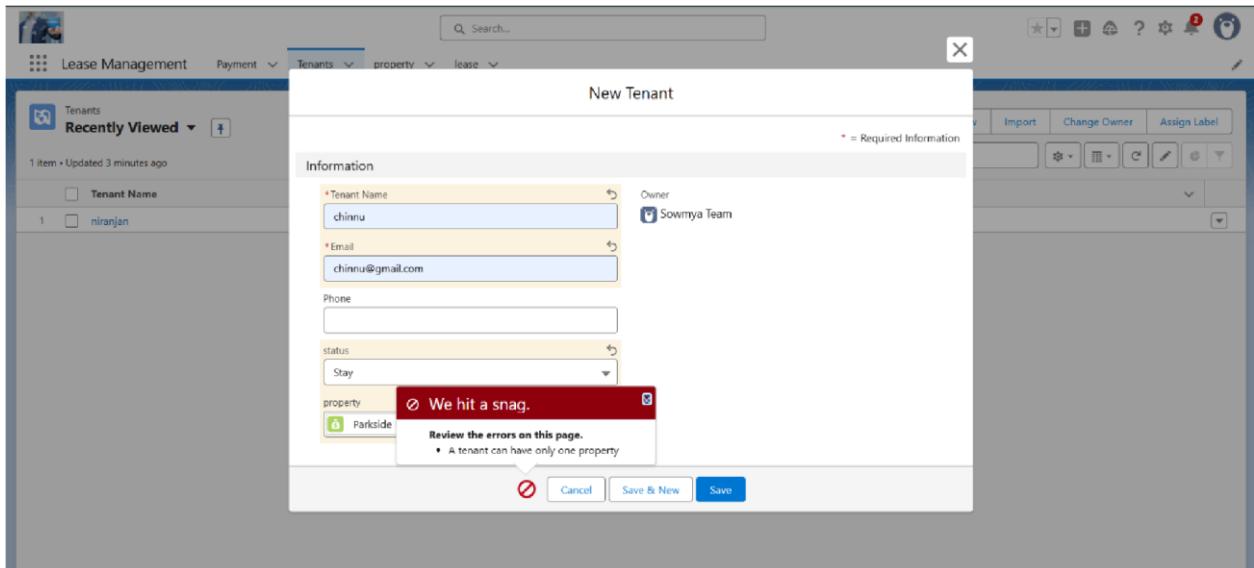
No past activity. Past meetings and tasks marked as done show up here.



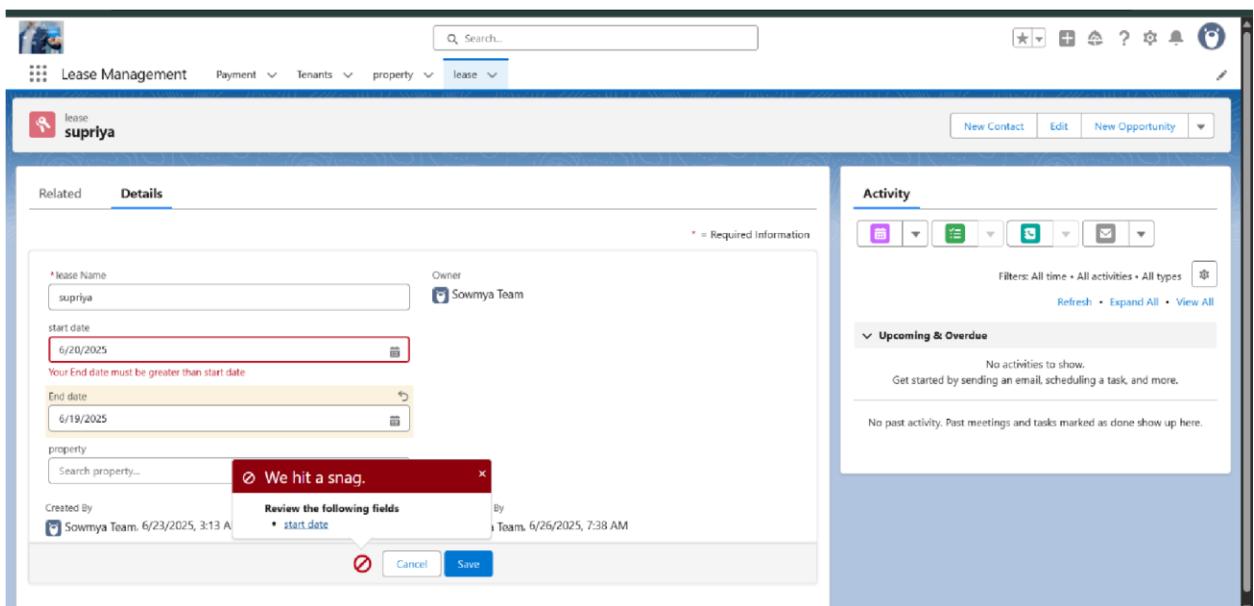
# FUNCTIONAL AND PERFORMANCE TESTING

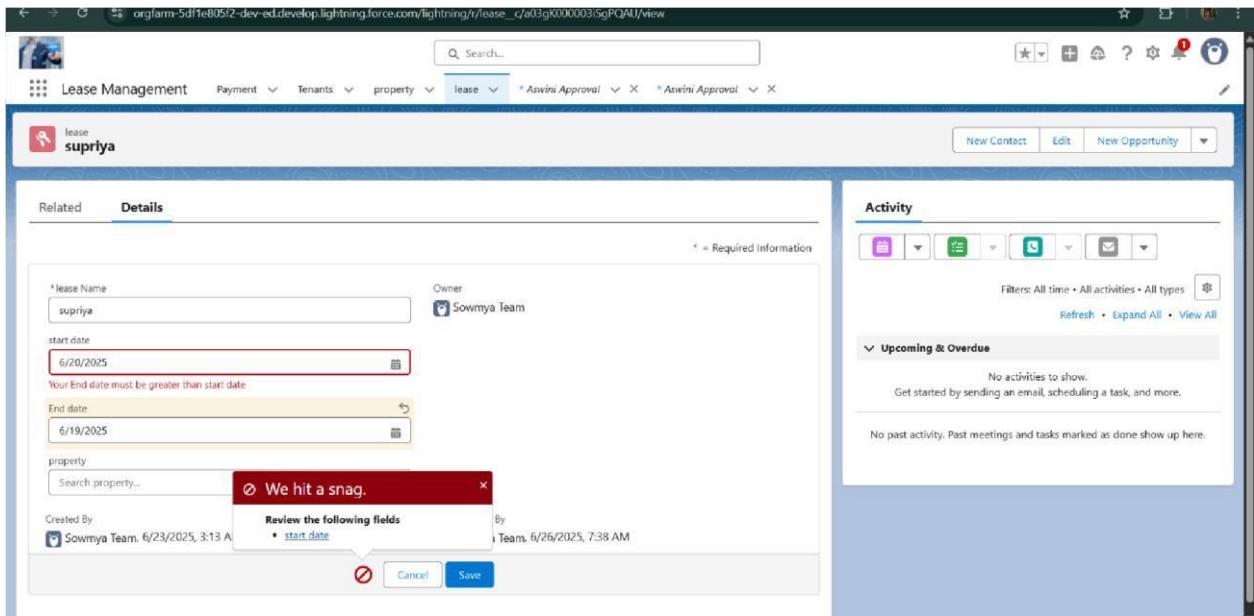
## Performance Testing

- Trigger validation by entering duplicate tenant-property records

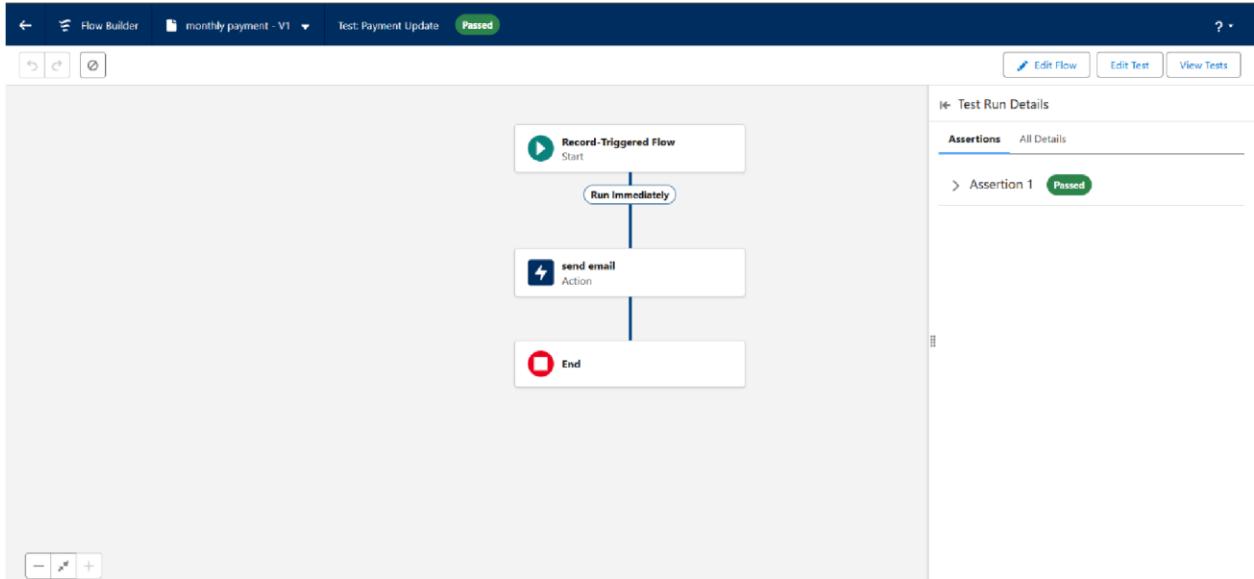


- Validation Rule checking





- 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 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:

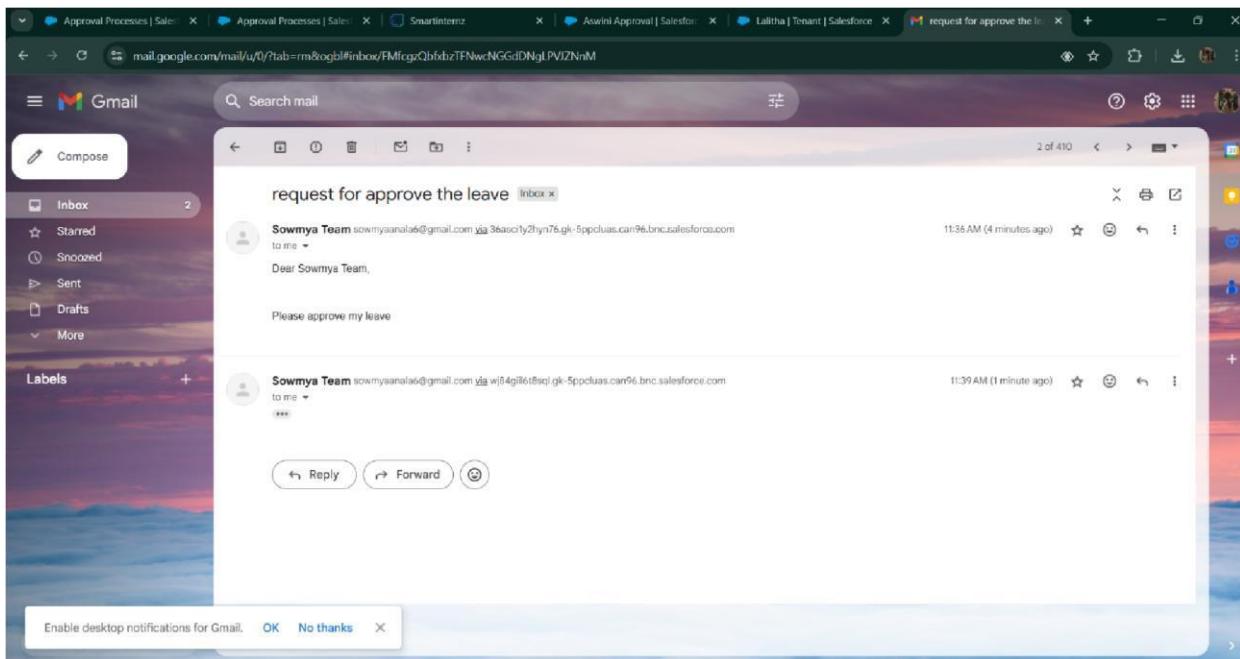
- Custom Object Tabs:** Contains tabs for Lease (Tab Style: Keys), Payment (Tab Style: Credit card), property (Tab Style: Sack), and Tenant (Tab Style: Map).
- Web Tabs:** Shows a message: "No Web Tabs have been defined".
- Visualforce Tabs:** Shows a message: "No Visualforce Tabs have been defined".

- Email alerts

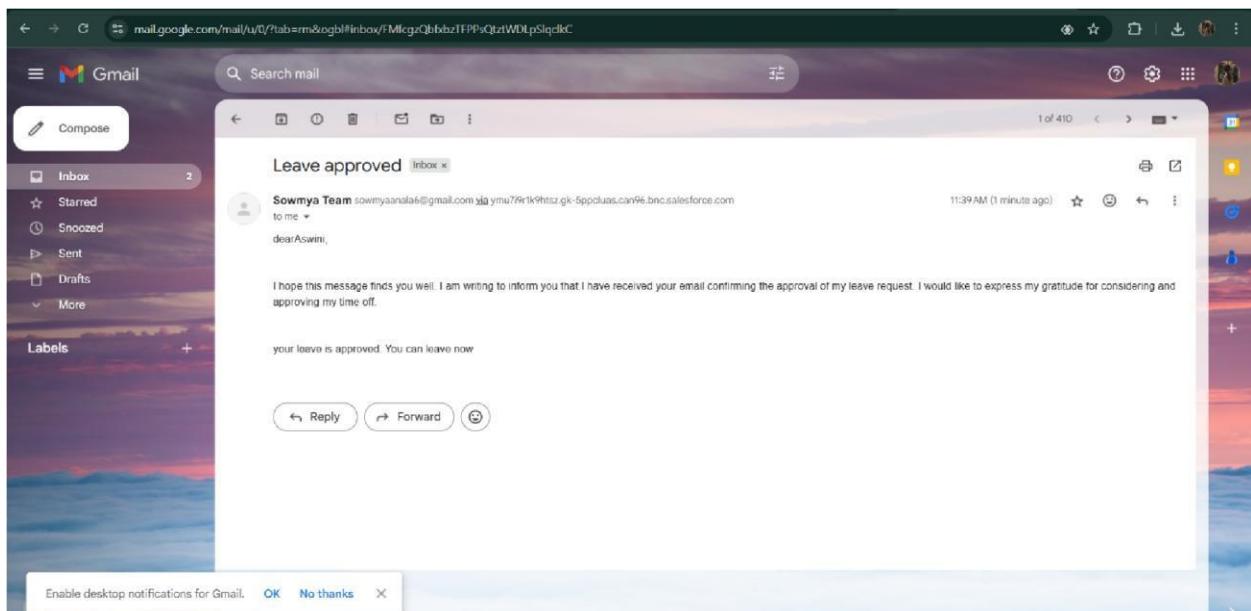
The screenshot shows the 'Lease Management' application interface. The current view is 'Approval History' for a tenant named 'niranjana'. The table lists the following approval steps:

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

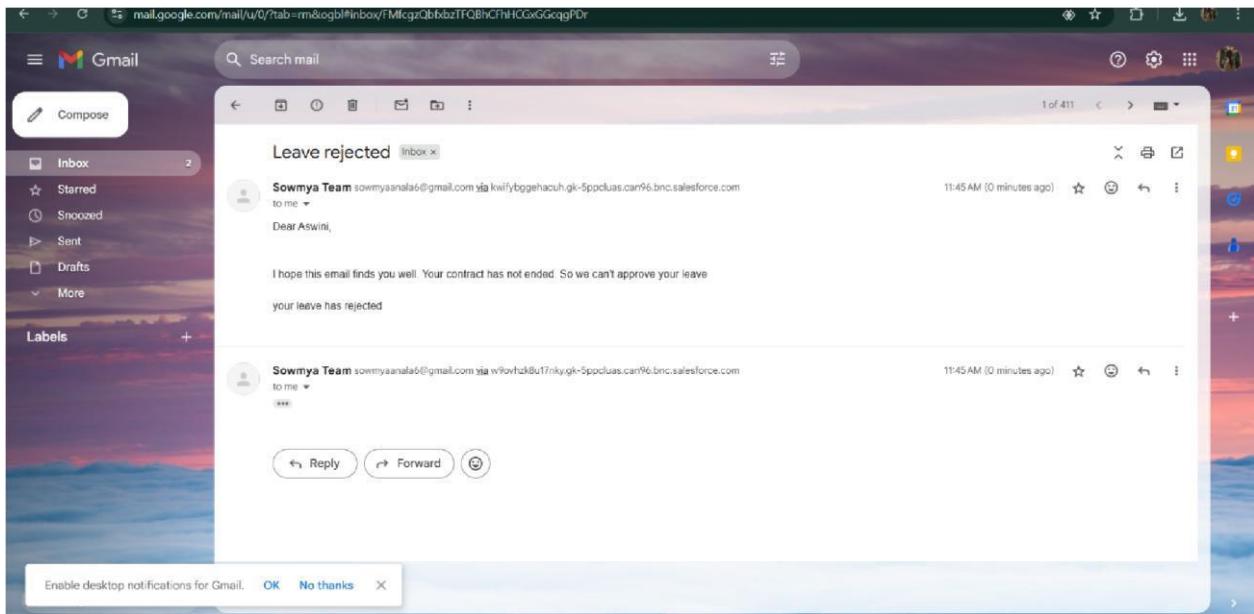
- Request for approve the leave



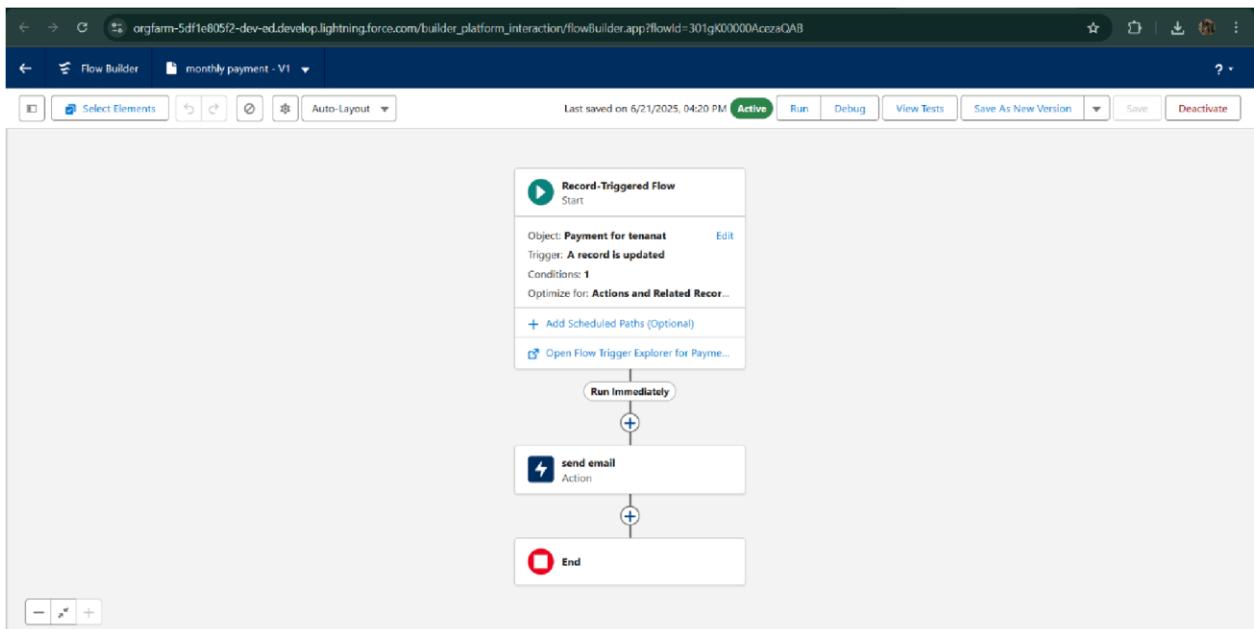
- Leave approved



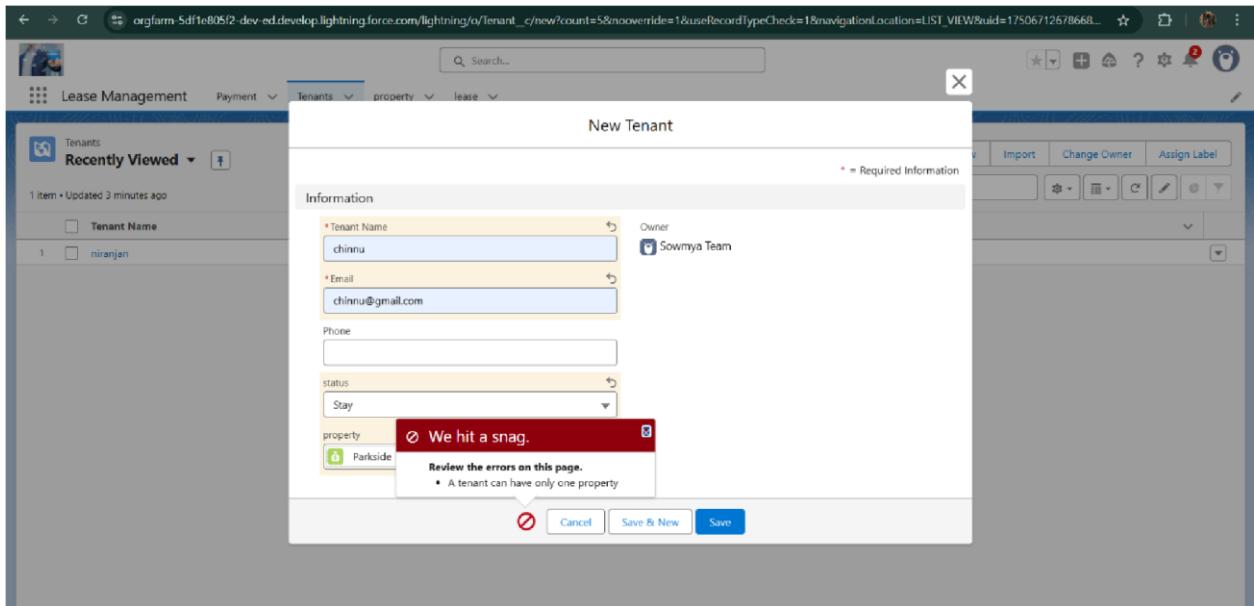
- Leave rejected



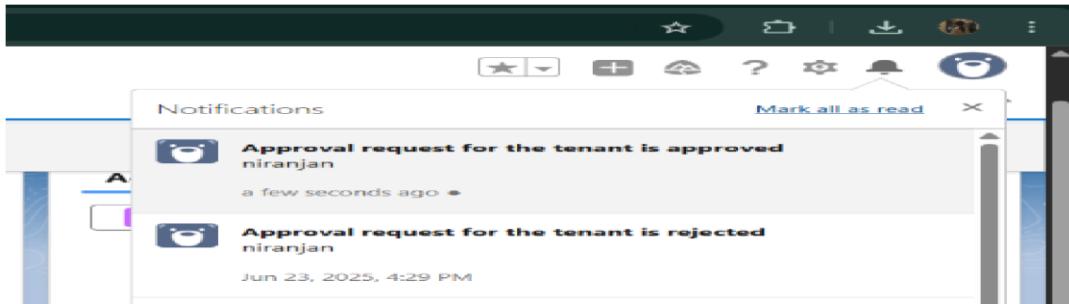
- Flow runs



- Trigger error messages



- Approval process notifications



## ADVANTAGES & DISADVANTAGES

# CONCLUSION

The **Garage Management System** developed using **Salesforce Apex Classes and Triggers** provides an efficient way to manage customer information, vehicles, service records, job cards, and invoices in a single platform. By automating key operations such as service cost calculation, job card tracking, and invoice generation, the system minimizes manual effort and reduces errors.

---

## 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);  
}
```

### GarageServiceHandler.cls:

```
public with sharing class GarageServiceHandler {  
  
    // Create a new job card  
  
    public static Id createJobCard(Id customerId, Id vehicleId, List<Id> serviceIds) {  
        JobCard__c job = new JobCard__c(  
            Customer__c = customerId,  
            Vehicle__c = vehicleId,  
            Status__c = 'Open',  
            TotalAmount__c = 0  
        );  
        insert job;  
  
        List<JobCard_Service__c> jobServices = new List<JobCard_Service__c>();
```

```
for (Id sid : serviceIds) {  
  
    jobServices.add(new JobCard_Service__c(  
  
        JobCard__c = job.Id,  
  
        Service__c = sid  
  
    ));  
  
}  
  
insert jobServices;  
  
  
return job.Id;  
}
```

```
// Close JobCard and generate Invoice  
  
public static Id generateInvoice(Id jobCardId) {  
  
    JobCard__c job = [SELECT Id, Customer__c, TotalAmount__c FROM JobCard__c WHERE Id =  
    :jobCardId LIMIT 1];
```

```
Invoice__c inv = new Invoice__c(  
  
    JobCard__c = job.Id,  
  
    Customer__c = job.Customer__c,  
  
    Amount__c = job.TotalAmount__c,  
  
    Status__c = 'Pending Payment'  
  
);  
  
insert inv;
```

```
job.Status__c = 'Closed';  
  
update job;
```

```

        return inv.Id;
    }
}

```

### **ServicePricingUtil.cls:**

```

public with sharing class ServicePricingUtil {

    // Calculate total service cost for a job card
    public static Decimal calculateTotal(Id jobCardId) {
        List<JobCard_Service__c> services = [
            SELECT Service__r.Price__c
            FROM JobCard_Service__c
            WHERE JobCard__c = :jobCardId
        ];

        Decimal total = 0;
        for (JobCard_Service__c js : services) {
            total += js.Service__r.Price__c;
        }
        return total;
    }
}

```

### **JobCardTrigger.trigger:**

```

trigger JobCardTrigger on JobCard_Service__c (after insert, after delete) {
    Set<Id> jobCardIds = new Set<Id>();
    if (Trigger.isInsert) {
        for (JobCard_Service__c js : Trigger.new) {
            jobCardIds.add(js.JobCard__c);
        }
    }
    if (Trigger.isDelete) {
        for (JobCard_Service__c js : Trigger.old) {
            jobCardIds.add(js.JobCard__c);
        }
    }
}

List<JobCard__c> jobsToUpdate = new List<JobCard__c>();
for (Id jobId : jobCardIds) {
    Decimal total = ServicePricingUtil.calculateTotal(jobId);
    jobsToUpdate.add(new JobCard__c(Id = jobId, TotalAmount__c = total));
}

if (!jobsToUpdate.isEmpty()) {
    update jobsToUpdate;
}

```

```
    }  
}
```

### **InvoiceTrigger.trigger:**

```
trigger InvoiceTrigger on Invoice__c (before insert, before update) {  
    for (Invoice__c inv : Trigger.new) {  
        if (inv.Status__c == 'Paid' && Trigger.isUpdate) {  
            inv.PaymentDate__c = System.today();  
        }  
    }  
}
```