

# **LEASE MANAGEMENT**

**College Name : SRI VASAVI COLLEGE (SFW), ERODE.**

**College code : bru17**

**TEAM ID : NM2025TMID23310**

## **TEAM MEMBERS :**

**Team Leader Name:** PAVITHRA M

**Email:** pavithramurugesanopm@gmail.com

**Team Member1:** BOOMIKA K

**Email:** boomika11@gmail.com

**Team Member2:** KAVIYADHARSINI V

**Email:** kdharshini325@gmail.com

**Team Member3:** ABISHEK N

**Email:** abisheknatraj09@gmail.com

**Team Member4:** SANTHOSH V

**Email:** vsanthosh2111@gmail.com

## **1.INTRODUCTION**

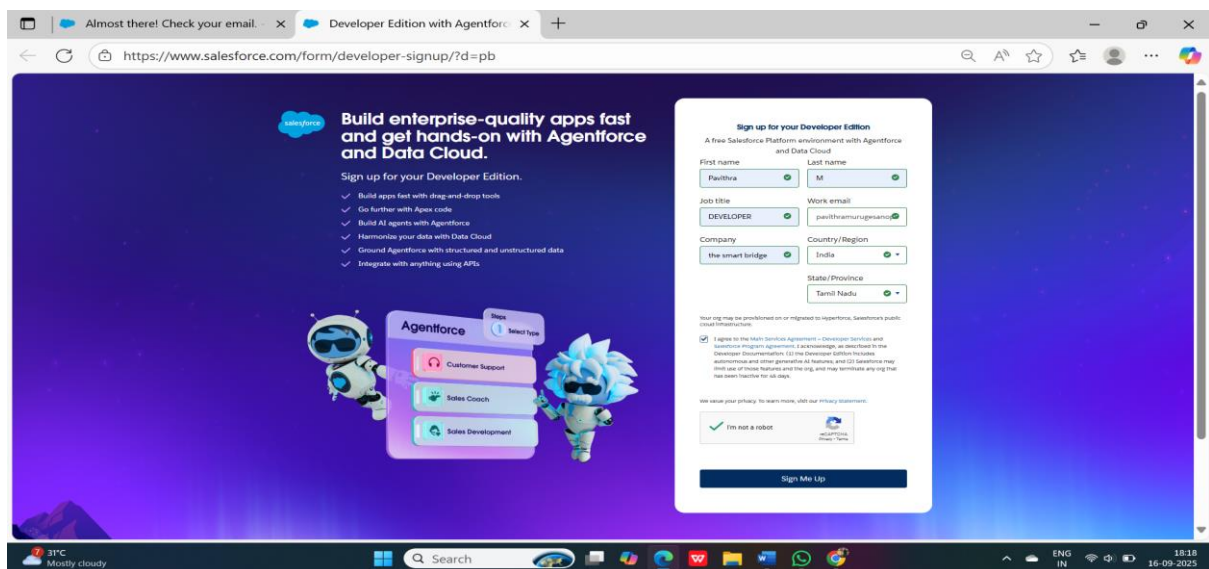
### **1.1 Project Overview**

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 alert

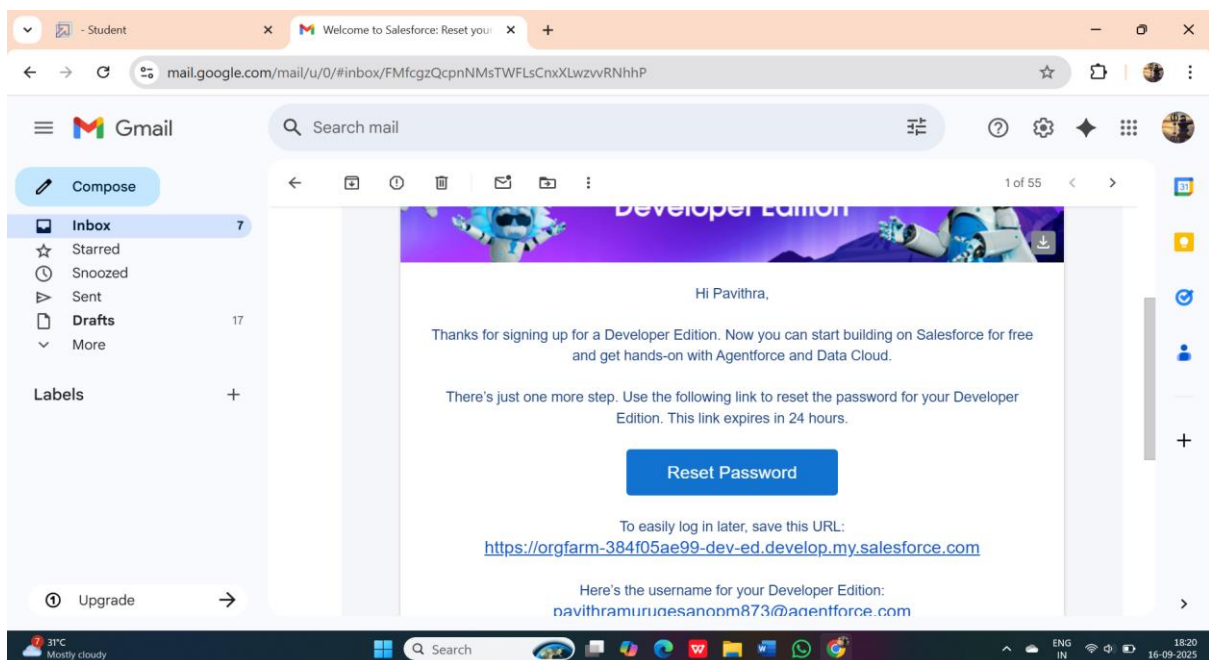
## 1.2 Purpose

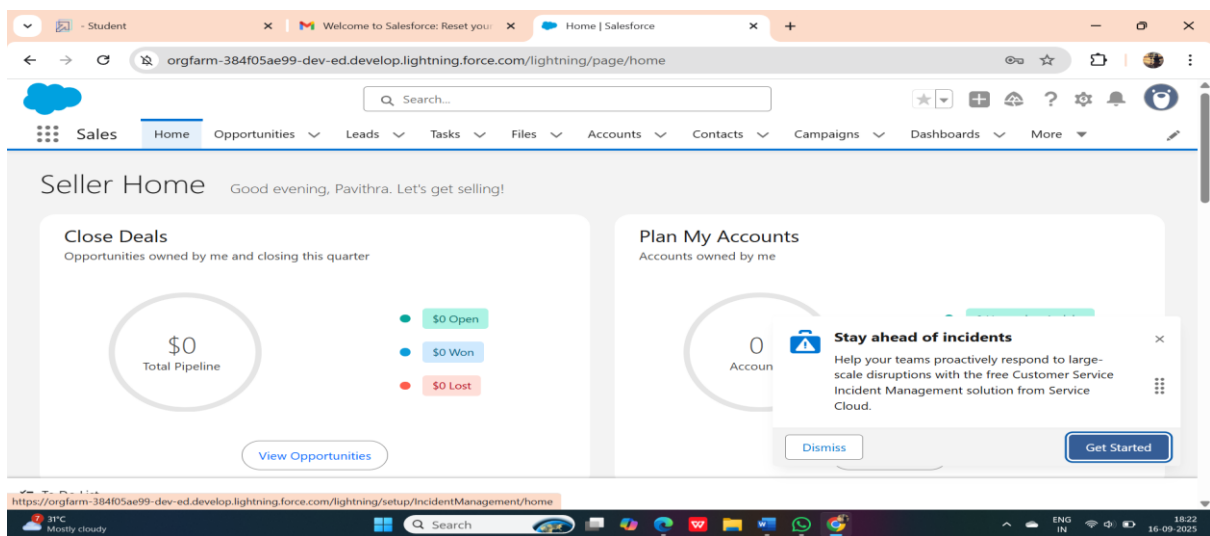
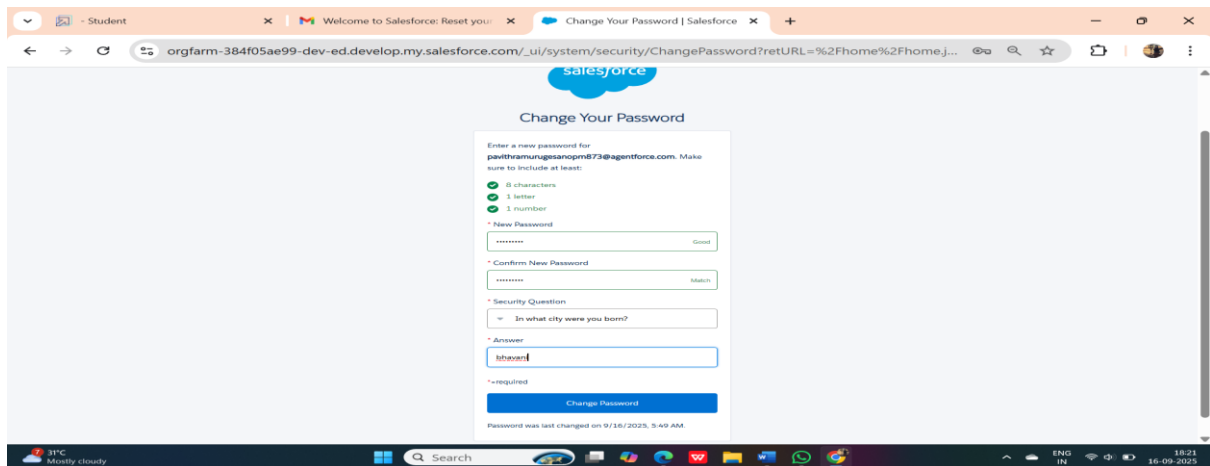
The main objective of the project is to enable organizations to efficiently manage properties, tenants, and lease-related activities. It reduces manual intervention, improves accuracy, and ensures better compliance and communication.

- **DEVELOPMENT PHASE** Creating Developer Account: By using this URL: <https://www.salesforce.com>

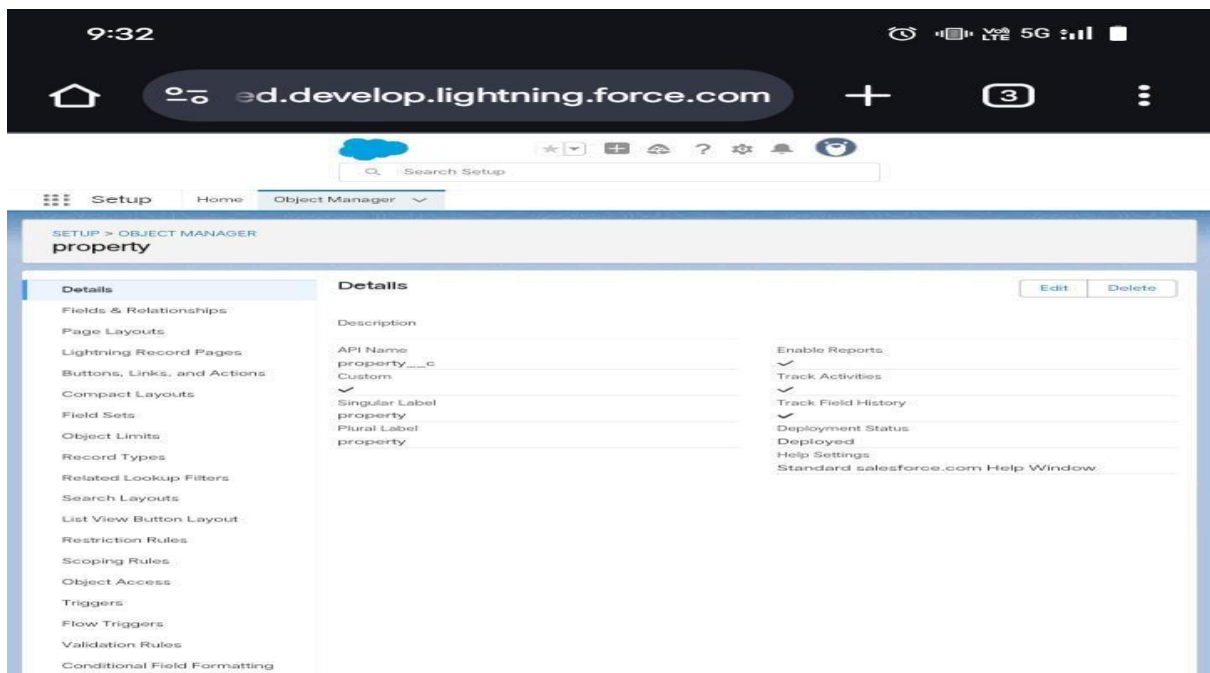


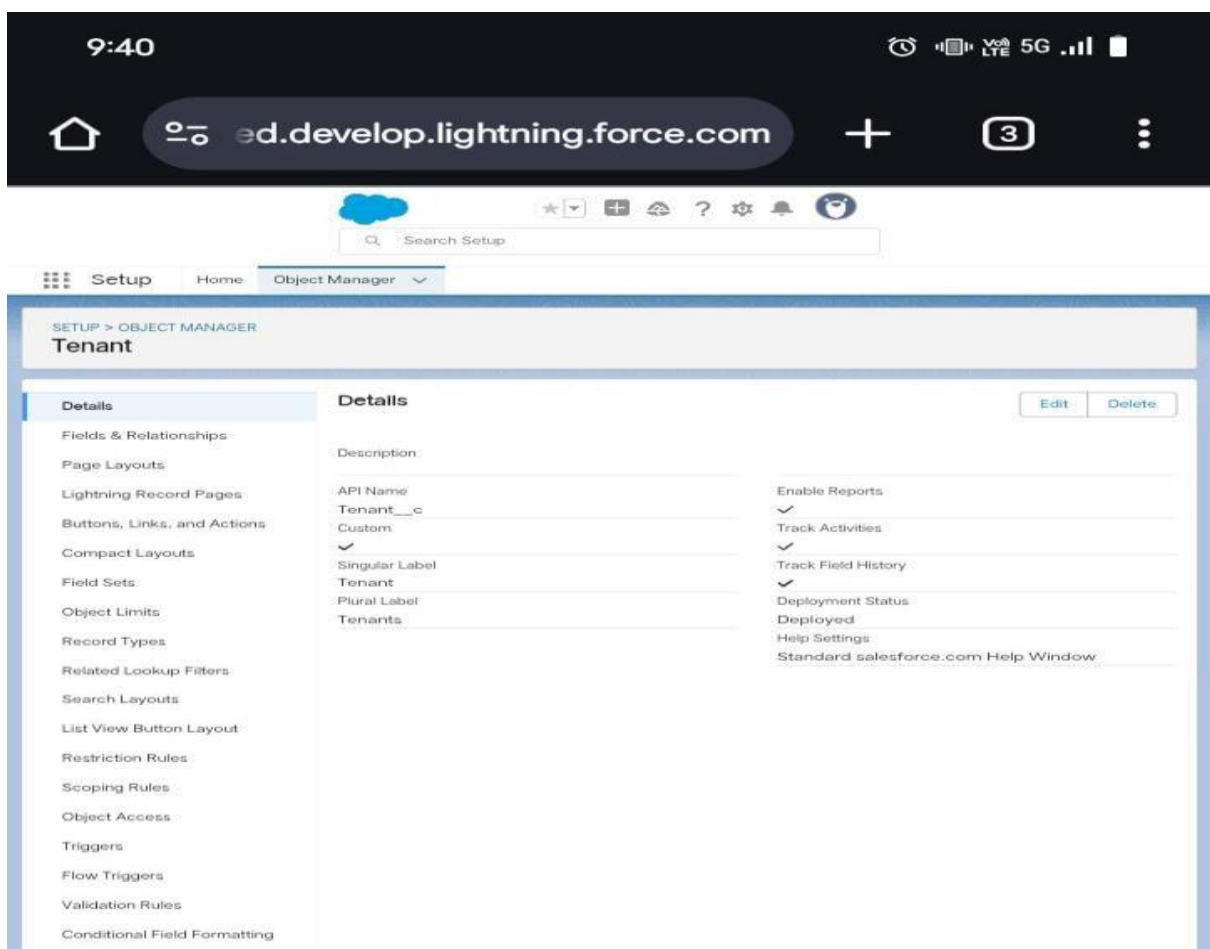
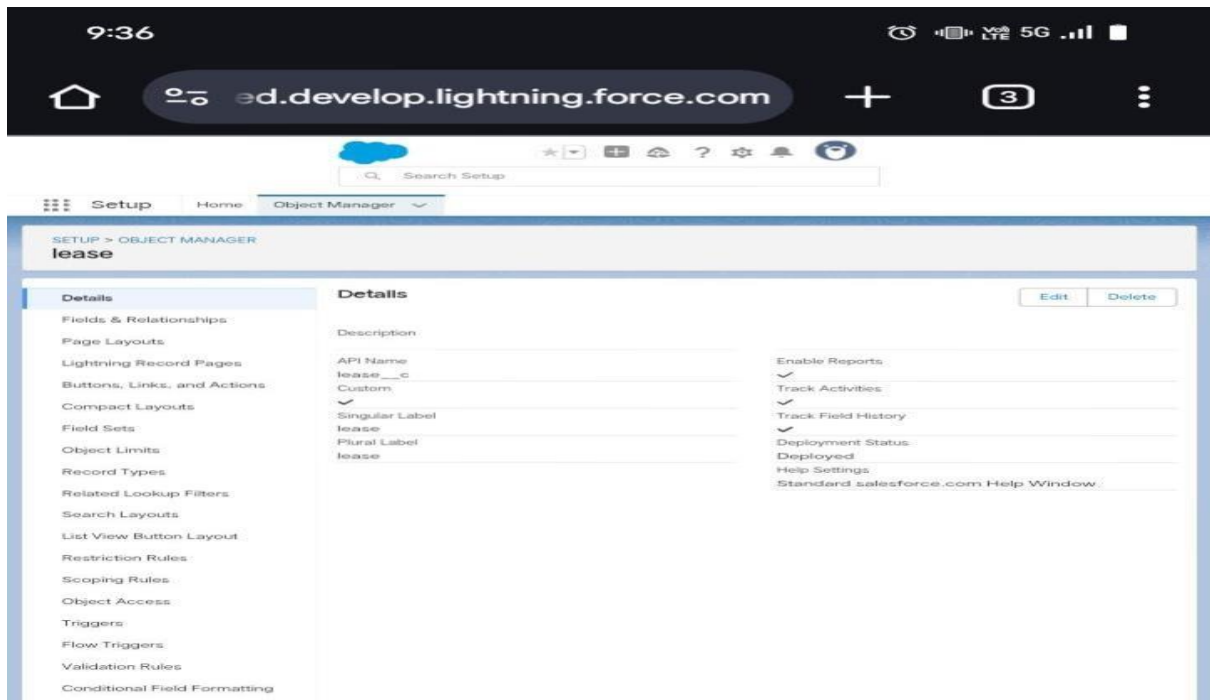
The screenshot shows the Salesforce Developer Edition sign-up page. The page has a dark blue header with the Salesforce logo and the text "Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud." Below this, there is a section titled "Sign up for your Developer Edition." with a list of benefits: "Build apps fast with drag-and-drop tools", "Go further with Apex code", "Build AI agents with Agentforce", "Harmonize your data with Data Cloud", "Ground Agentforce with structured and unstructured data", and "Integrate with anything using APIs". To the right of this list is a form titled "Sign up for your Developer Edition" with fields for "First name" (Pavithra), "Last name" (M), "Job title" (DEVELOPER), "Work email" (pavithramurugesan@gmail.com), "Company" (the smart bridge), "Country/Region" (India), and "State/Province" (Tamil Nadu). Below the form is a checkbox for "I agree to the Salesforce Developer Edition Terms of Service" and a "Sign Me Up" button.

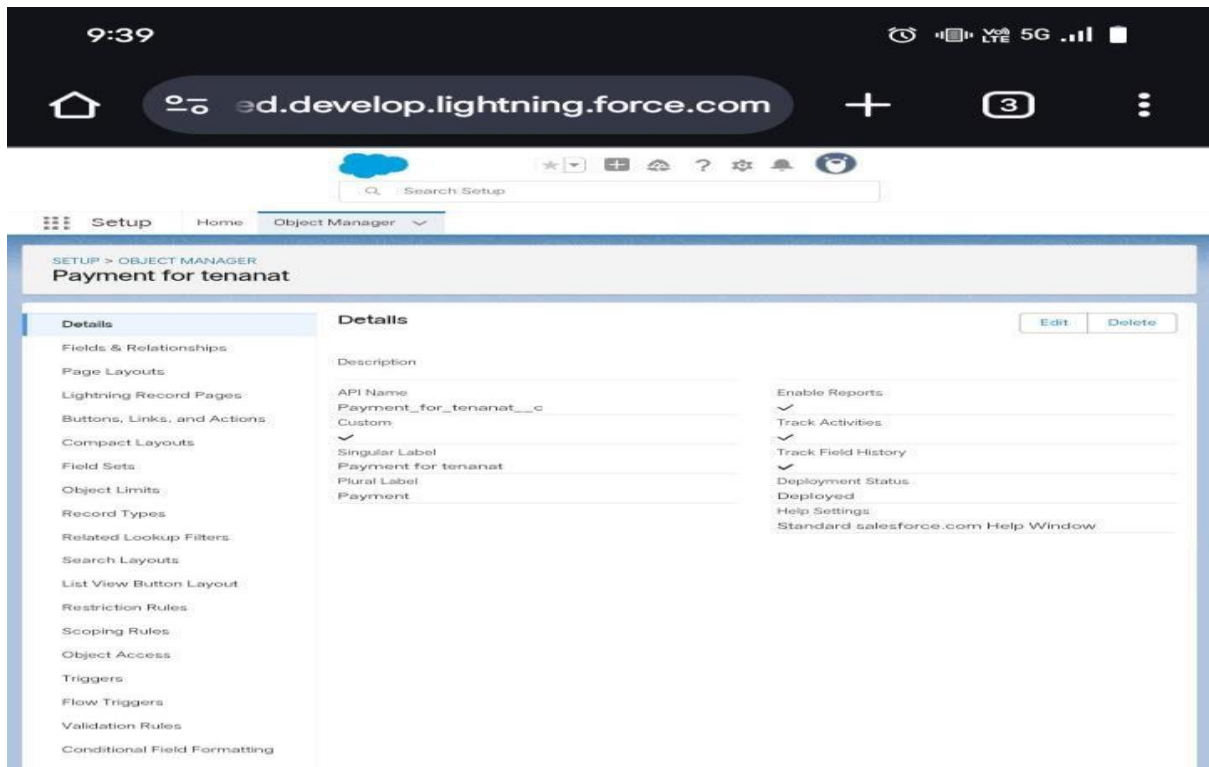




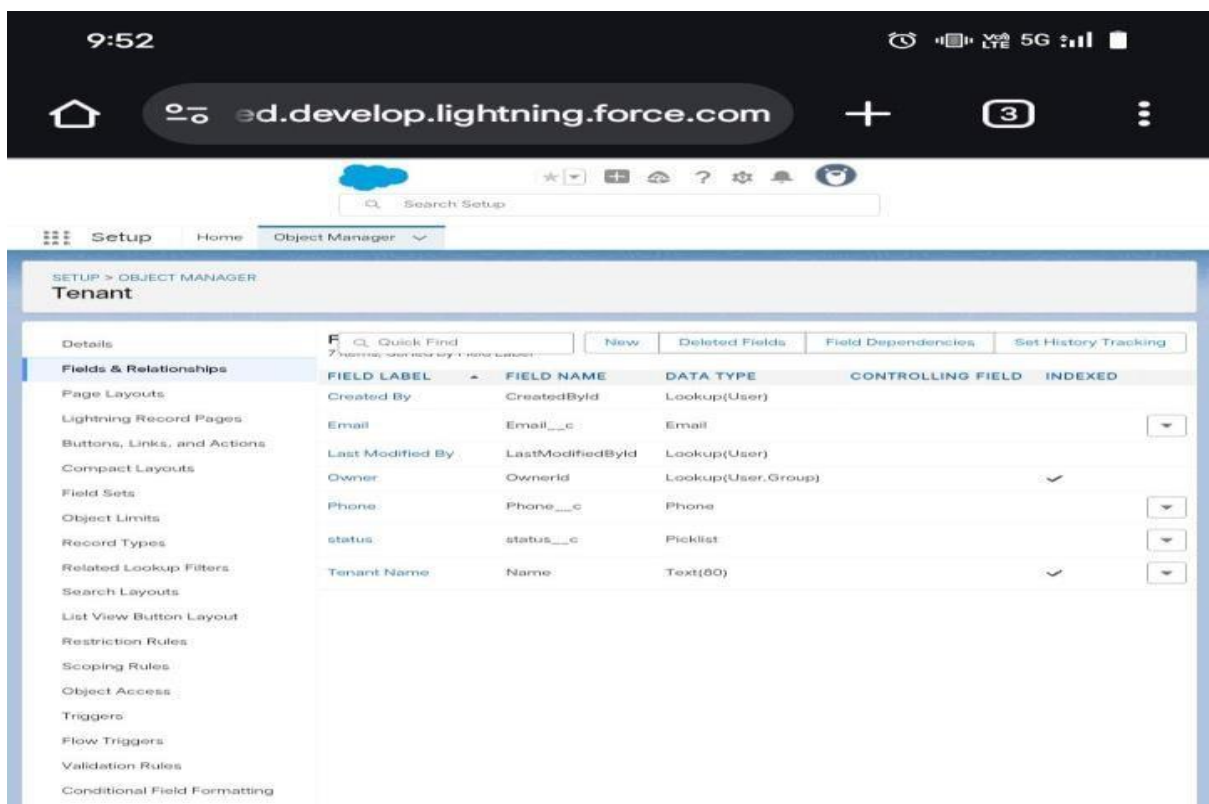
**Created objects: Property, Tenant, Lease, Payment**







- **Configured fields and relationships**



3:23

5G

ed.develop.lightning.force.com

4

Search Setup

Setup

Home

Object Manager

SETUP > OBJECT MANAGER

property

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

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Quick Find

New

Deleted Fields

Field Dependencies

Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIE...	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 Name	Name	Text(80)		
sqft	sqft__c	Text(10)		
Type	Type__c	Picklist		

3:28

5G

ed.develop.lightning.force.com

4

Search Setup

Setup

Home

Object Manager

SETUP > OBJECT MANAGER

Payment for tenanat

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

Restriction Rules

Scoping Rules

Object Access

Triggers

Flow Triggers

Validation Rules

Conditional Field Formatting

Quick Find

New

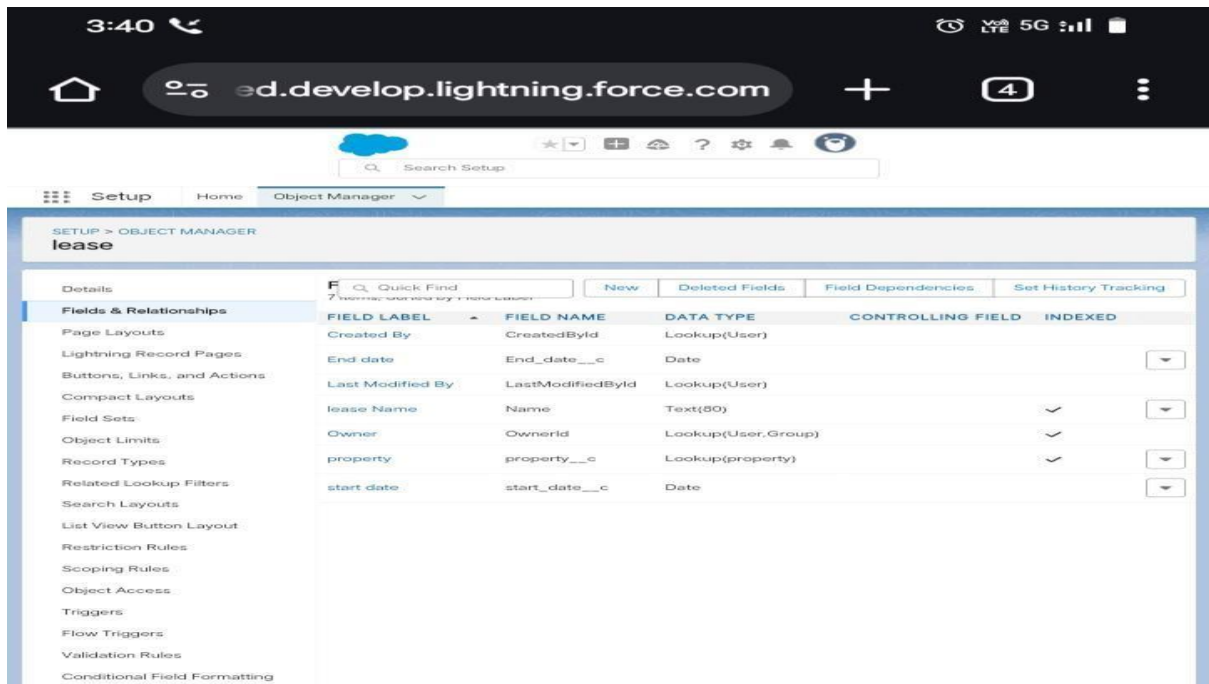
Deleted Fields

Field Dependencies

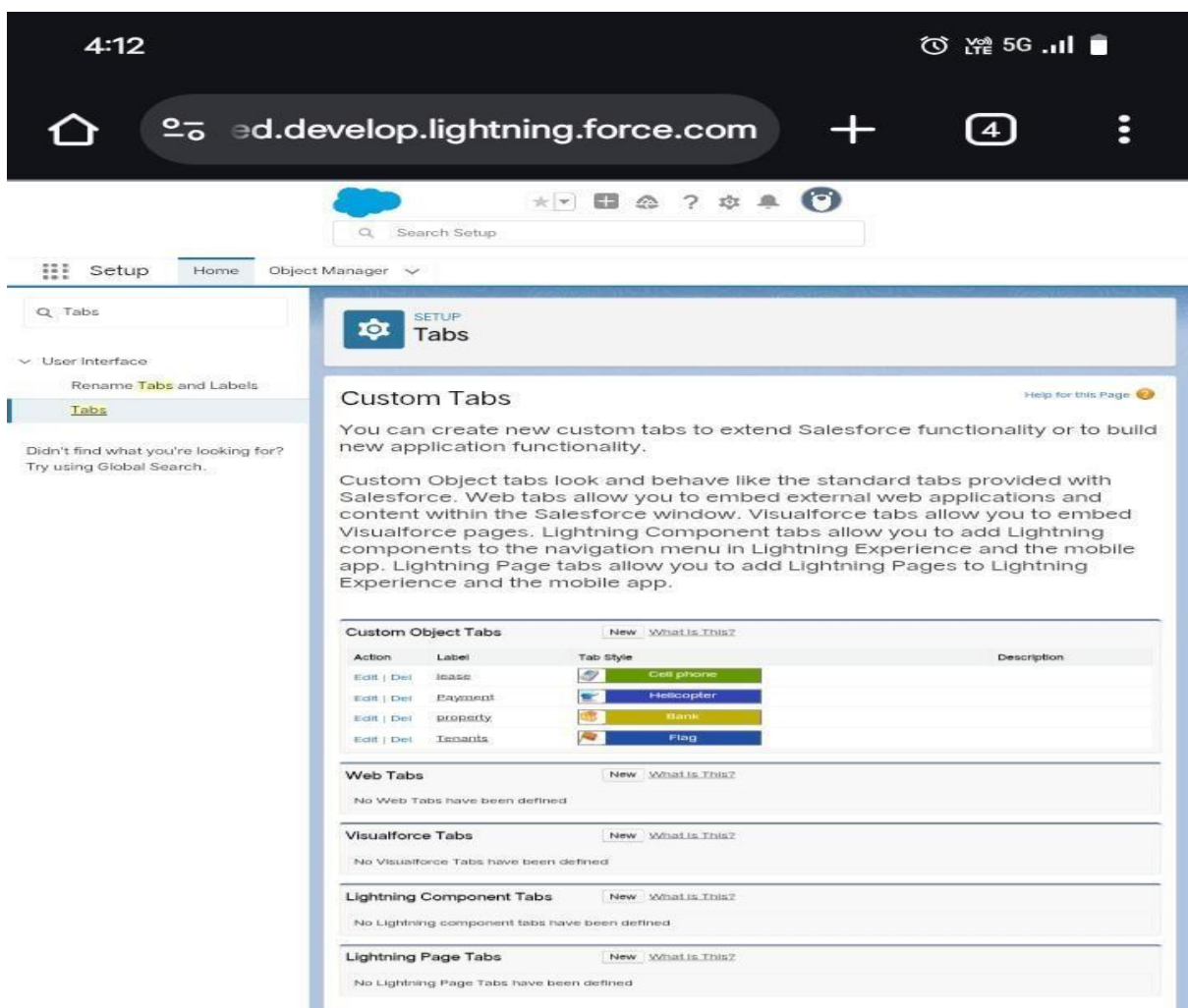
Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIE...	INDEXED
Amount	Amount__c	Number(10, 0)		
check for payment	check_for_payment__c	Picklist		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Payment date	Payment_date__c	Date		
Payment Name	Name	Text(80)		
property	property__c	Master-Detail(property)		





## Configuring the Tabs :



## Developed Lightning App with relevant tabs :

Lightning Experience App Manager

Clone Lightning App Lease Management

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\_clone

\* Developer Name ⓘ  
Lease\_Management\_clone


Description ⓘ  
Enter a description....

App Branding

Image ⓘ  
  
Clear

Primary Color Hex Value ⓘ  
 #0070D2

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

App Launcher Preview  
 Lease\_Management\_clone

Quick Save

Next

Clone Lightning App Lease Management

App Options

Navigation and Form Factor ⓘ

\* Navigation Style ⓘ  
☒ Standard navigation  
☐ Console navigation

\* Supported Form Factors ⓘ  
☒ Desktop and phone  
☐ Desktop  
☐ Phone

Setup and Personalization ⓘ

Setup Experience ⓘ  
☒ Setup (full set of Setup options)  
☐ Service Setup  
☐ Data Cloud Setup

App Personalization Settings ⓘ  
☐ Disable end user personalization of new items in this app  
☐ Disable temporary tabs for items outside of this app  
☐ Use Omni-Channel sidebar

Back

Quick Save

Next



## Clone Lightning App Lease Management

### 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...

Accounts

Activation Targets

Activations

All Sites

Alternative Payment...

Analytics

App Launcher

Appointment Categ...

Appointment Inuitati...

Approval Requests

Selected Items

property

lease

Tenants

Payment

[Back](#)[Quick Save](#)[Next](#)

## Clone Lightning App Lease Management

### User Profiles

Choose the user profiles that can access this app.

Available Profiles

Type to filter list...

Analytics Cloud Integration...

Analytics Cloud Security U...

Anypoint Integration

Authenticated Website

Authenticated Website

B2B Reordering Portal Buy...

Contract Manager

Custom: Marketing Profile

Custom: Sales Profile

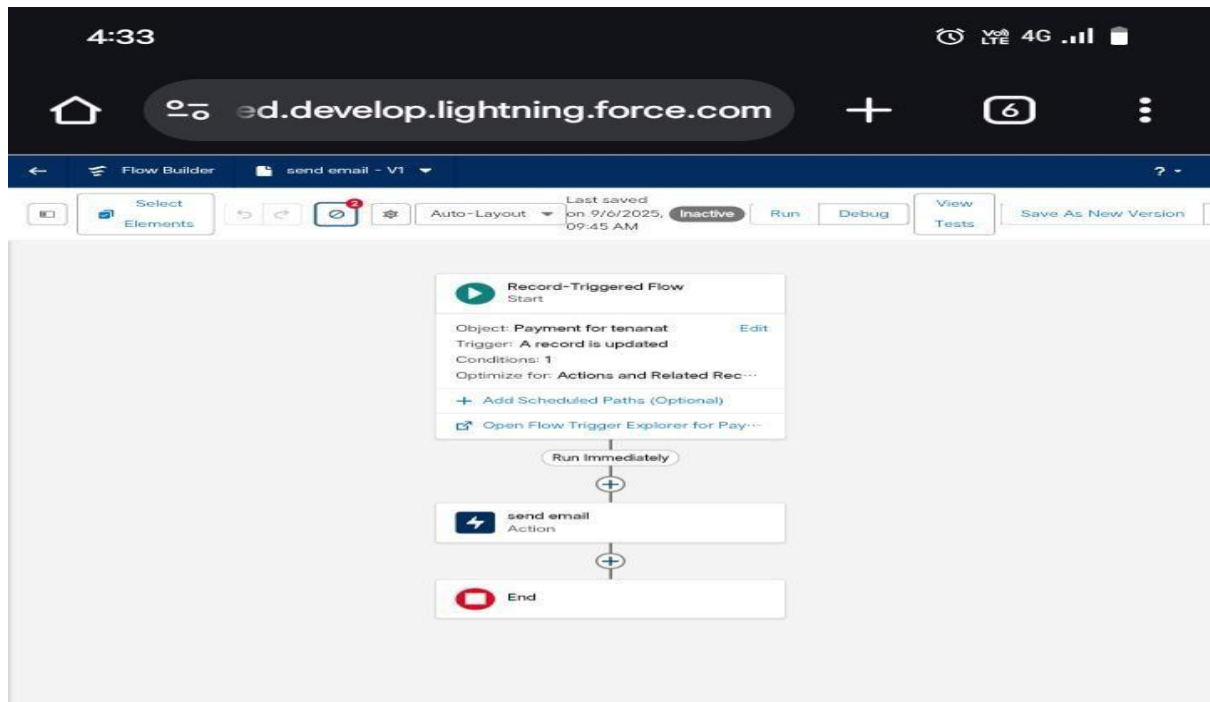
Custom: Support Profile

Selected Profiles

System Administrator

[Back](#)[Save & Finish](#)

## Implemented Flows for monthly rent and payment success :



## To create a validation rule to a Lease Object :

The screenshot shows the Salesforce Setup page for the "Lease" object. The left sidebar lists various setup options, with "Validation Rules" selected. The main content area displays the "Lease Validation Rule" configuration page.

Validation Rule Detail		
Rule Name	lease_end_date	Active
Error Condition Formula	End_date__c > start_date__c	
Error Message	Your End date must be greater than start date	Error Location: start date
Description		
Created By	BR0380LS, 9/2/2025, 12:00 AM	Modified By: BR0380LS, 9/2/2025, 12:00 AM

**4:37** **ed.develop.lightning.force.com**

Setup > OBJECT MANAGER > lease

**lease Validation Rule**

Define a validation rule by specifying an error condition and a corresponding error message. The error condition is written as a Boolean formula expression that returns true or false. When the formula expression returns true, the save will be aborted and the error message will be displayed. The user can correct the error and try again.

**Validation Rule Edit**

Rule Name: lease\_end\_date Save Save & New Cancel

Active: ☒ Description:

**Error Condition Formula**

Example: `Discount_Percent < 0.30` 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

`End_date__c > start_date__c`

Check Syntax

**Error Message**

Example: `Discount percent cannot exceed 30%`

This message will appear when Error Condition formula is true

Error Message: `Your End date must be greater than start date`

This error message can either appear at the top of the page or below a specific field on the page

Error Location: ☐ Top of Page ☒ Field (start\_date) [L]

Save Save & New Cancel

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

**4:43** **ed.develop.lightning.force.com**

Setup > Email > Classic Email Templates

**Classic Email Templates**

Text Email Template: **Leave rejected**

Preview your email template below.

**Email Template Detail**

Email Templates from Salesforce	Unified Pledge Classic Email Templates	Available For Use	<input checked="" type="checkbox"/>
Email Template Name	Leave_rejected	Last Used Date	
Template Unique Name	Leave_rejected	Times Used	
Encoding	Unicode (UTF-8)		
Author	Rajagals [C38096]		
Description			
Created By	Rajagals, 9/4/2025, 1:09 AM	Modified By	Rajagals, 9/4/2025, 1:08 AM

**Email Template**

Subject: Leave rejected

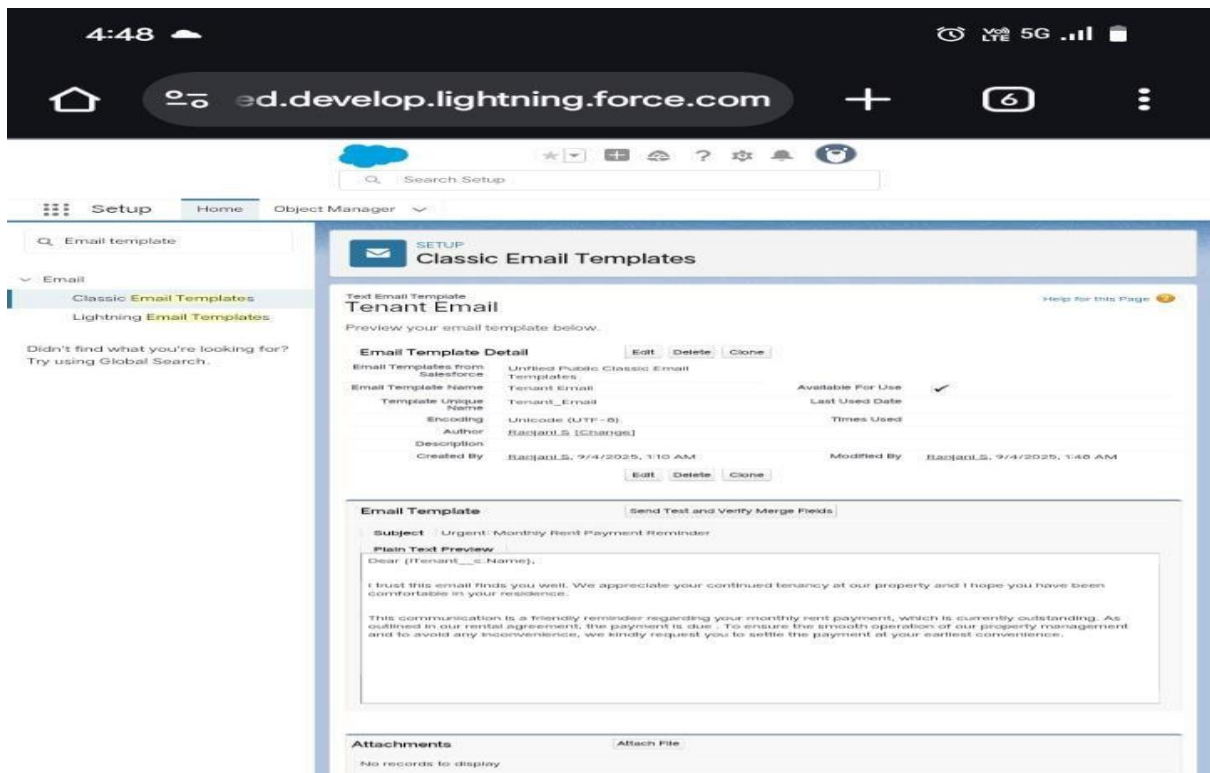
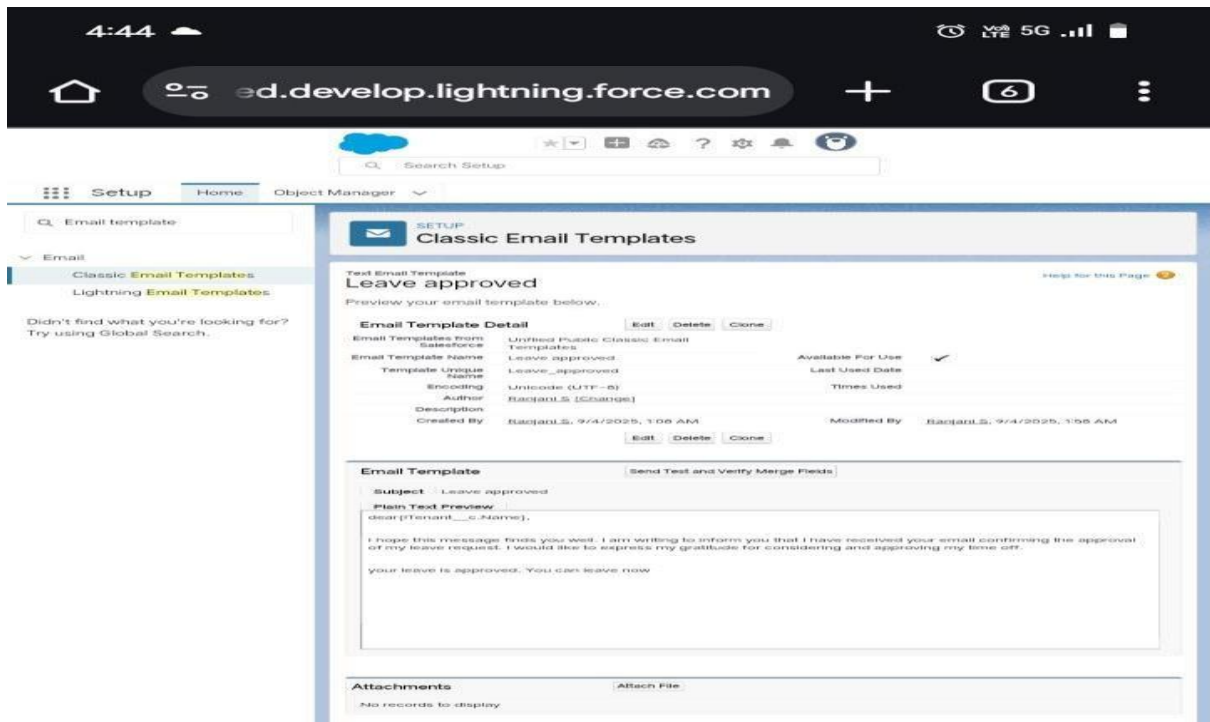
Plain Text Preview:

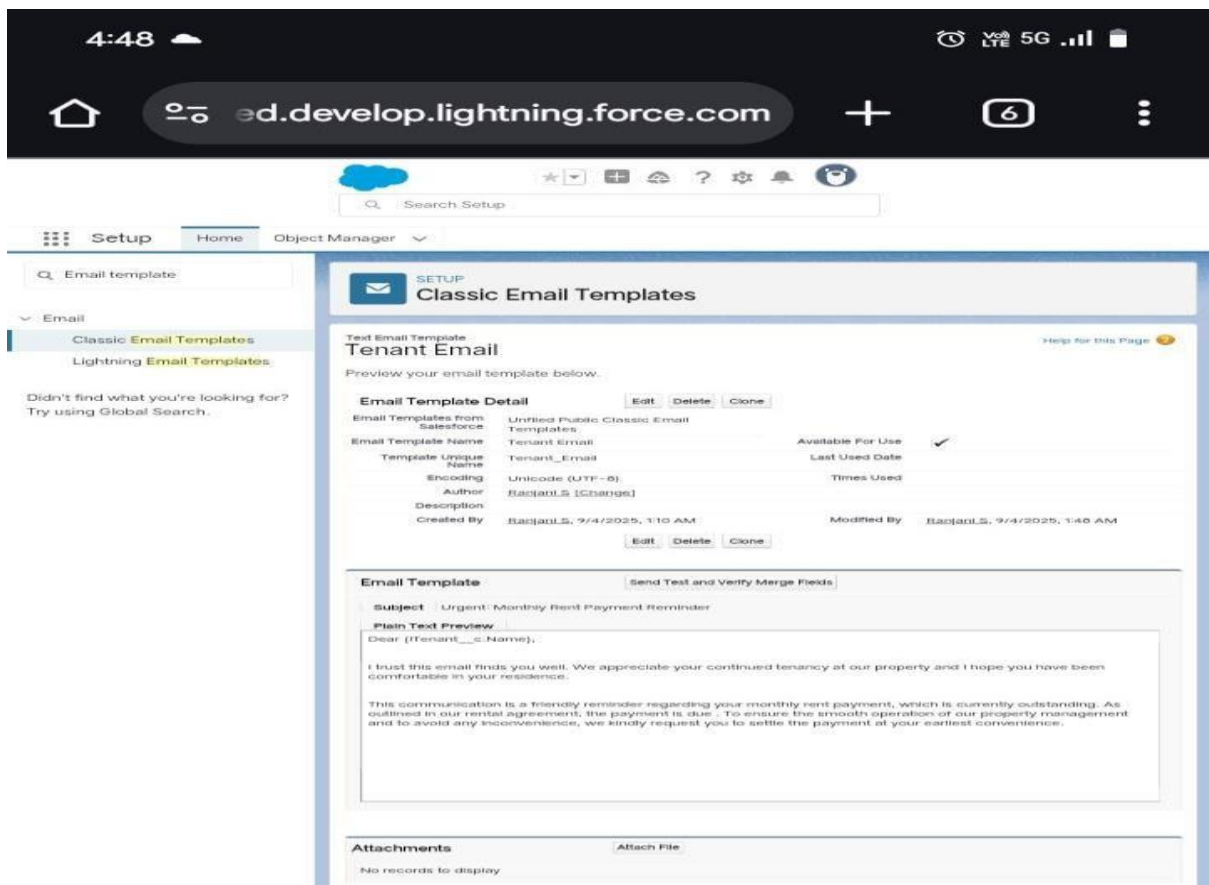
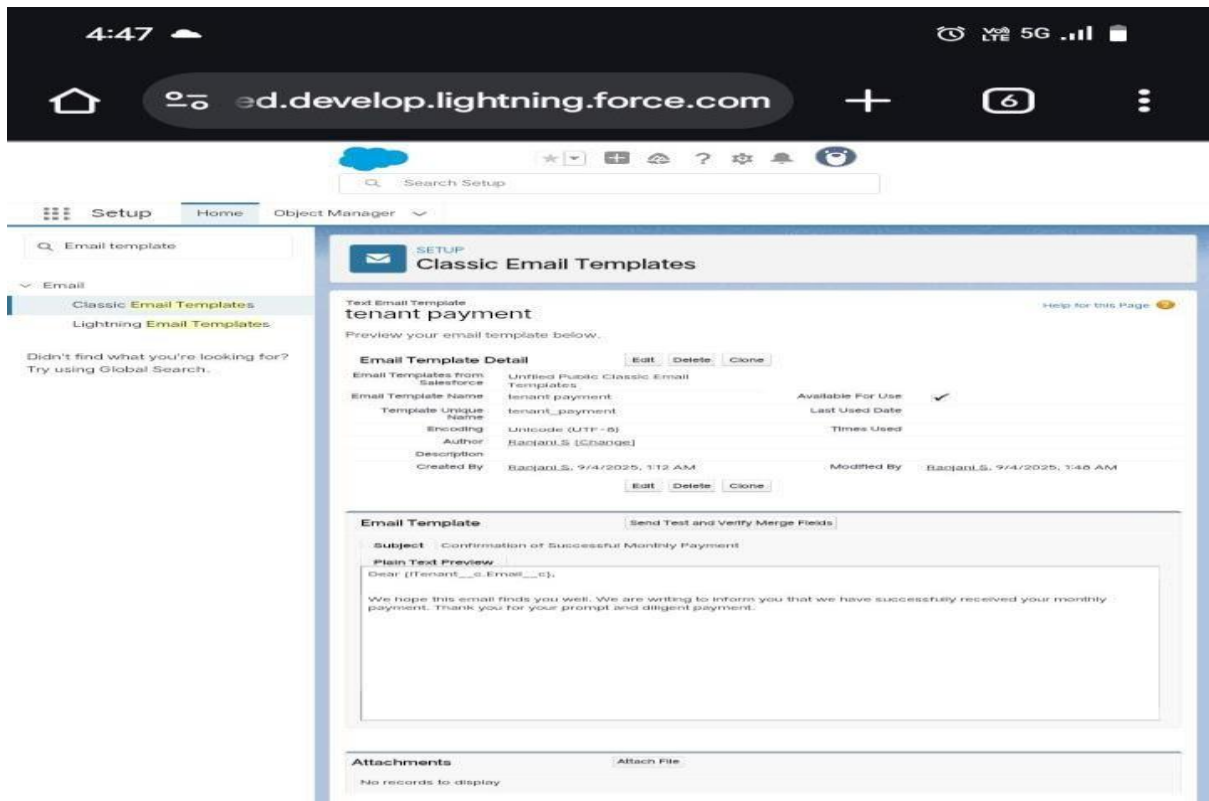
Dear {Tenant\_\_c.Name},

I hope this email finds you well. Your contract has not ended. So we can't approve your leave. Your leave has rejected

**Attachments**

No records to display





4:49

📶 VoLTE 5G

🏠

ed.develop.lightning.force.com

+

6

⋮

🔍 Search Setup

Setup

Home

Object Manager

🔍 Email template

Email

Classic Email Templates

Lightning Email Templates

Didn't find what you're looking for?

Try using Global Search...

SETUP

Classic Email Templates

Help for this Page

Text Email Template

tenant leaving

Preview your email template below.

Email Template Detail

Edit

Delete

Clone

Email Templates from Salesforce

Unused Public Classic Email Templates

Email Template Name

tenant leaving

Available For Use

✓

Template Unique Name

tenant\_leaving

Last Used Date

Encoding

Unicode (UTF-8)

Times Used

Author

Ranjan S. (Chadaga)

Description

Created By

Ranjan S. 9/4/2025, 1:05 AM

Modified By

Ranjan S. 9/4/2025, 1:47 AM

Edit

Delete

Clone

Email Template

Send Test and Verify Merge Fields

Subject

request for approve the leave

Plain Text Preview

Dear {Tenant\_\_c.CreatedBy},

Please approve my leave

Attachments

Attach File

No records to display

# Approval Process creation:

The screenshot shows the Salesforce Setup page for Approval Processes. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled 'Approval Processes' and includes a 'Try Flow Approval Processes!' section. This section introduces Flow Approval Processes as an enhanced alternative to classic Approval Processes, highlighting features like a flexible user-friendly interface, record change triggers, apex extensibility, detailed logging, and dynamic routing. Below this, a list of steps for getting started is provided: 1. Read the help topics, 2. View the checklist, 3. Create a custom user hierarchical relationship field, 4. Create email templates, 5. Create an approval process using either the Jump Start or Standard Wizard, 6. Add Approval History Related List to all page layouts, and 7. Activate the process to deploy to your users. At the bottom, there is a table showing active and inactive approval processes for the 'Tenant' org.

**Approval Processes**

**Try Flow Approval Processes!**

Introducing **Flow Approval Processes**, an enhanced alternative to classic Approval Processes.

- Flexible, User-Friendly Interface:** Create everything your approval process needs in Flow Builder.
- Record-Change Triggers:** Automate approvals based on specific record updates.
- Apex Extensibility:** Customize with Apex for advanced functionality.
- Detailed Logging:** Ensure compliance with comprehensive audit trails.
- Dynamic Routing:** Route approvals based on data and business rules.

Get started with Flow Approval Processes in the Approval app where you can manage approval submissions, approval work items, and flow approval processes in one location.

[Open Approval App](#)

Approvals are complex business processes that require information gathering and planning before implementing. It is recommended that you follow the instructions below before getting started.

1. Read the help topics
2. View the checklist
3. Create a custom user hierarchical relationship field
4. Create email templates
5. Create an approval process using either the Jump Start or Standard Wizard
6. Add Approval History Related List to all page layouts
7. Activate the process to deploy to your users

**Manage Approval Processes For: Tenant**

A listing of both active and inactive approval processes for **Tenants** is displayed below. To create a new approval process, click **Create New Approval Process** then select **Use Jump Start Wizard** to set up your approval process in a few short steps. Or, select **Use Standard Wizard** to configure all approval options.

[Create New Approval Process](#)

Active Approval Processes	
No approval processes available	

Inactive Approval Processes	
Action	Approval Process Name
<a href="#">Edit</a>	<a href="#">check_for_vacant</a>

The screenshot shows the Salesforce Setup page for the 'check\_for\_vacant' approval process. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled 'Approval Processes' and includes a 'Process Definition Detail' section. This section displays the process name, unique name, description, entry criteria, record editability, approval assignment, email template, initial submission actions, and approval steps. Below this, there are sections for 'Initial Submission Actions', 'Approval Steps', 'Final Approval Actions', 'Final Rejection Actions', and 'Recall Actions'. The 'Initial Submission Actions' section shows a table with columns for Action, Type, and Description. The 'Approval Steps' section shows a table with columns for Action, Type, and Description. The 'Final Approval Actions', 'Final Rejection Actions', and 'Recall Actions' sections each show a table with columns for Action, Type, and Description.

**Approval Processes**

**Tenant: check\_for\_vacant**

[Back to Approval Process List](#)

**Process Definition Detail** [Edit](#) [Clone](#) [Delete](#)

Process Name	Unique Name	Active	Next Automated Approver Determined By
check_for_vacant	check_for_vacant	<input type="checkbox"/>	<a href="#">Next Automated Approver Determined By</a>

**Description** [View](#) [Edit](#) [Add](#)

**Entry Criteria** [View](#) [Edit](#) [Add](#)

**Record Editability** [View](#) [Edit](#) [Add](#)

**Approval Assignment** [View](#) [Edit](#) [Add](#)

**Email Template** [View](#) [Edit](#) [Add](#)

**Initial Submitters** [View](#) [Edit](#) [Add](#)

**Created By** [View](#) [Edit](#) [Add](#)

**Modified By** [View](#) [Edit](#) [Add](#)

**Initial Submission Actions** [Add Existing](#) [Add New](#)

Action	Type	Description
<a href="#">Edit</a>	Record Lock	Lock the record from being edited

**Approval Steps** [Add Existing](#) [Add New](#)

[New Approval Step](#)

[You have not yet defined any approval steps](#)

**Final Approval Actions** [Add Existing](#) [Add New](#)

Action	Type	Description
<a href="#">Edit</a>	Record Lock	Lock the record from being edited

**Final Rejection Actions** [Add Existing](#) [Add New](#)

Action	Type	Description
<a href="#">Edit</a>	Record Lock	Unlock the record for editing

**Recall Actions** [Add Existing](#) [Add New](#)

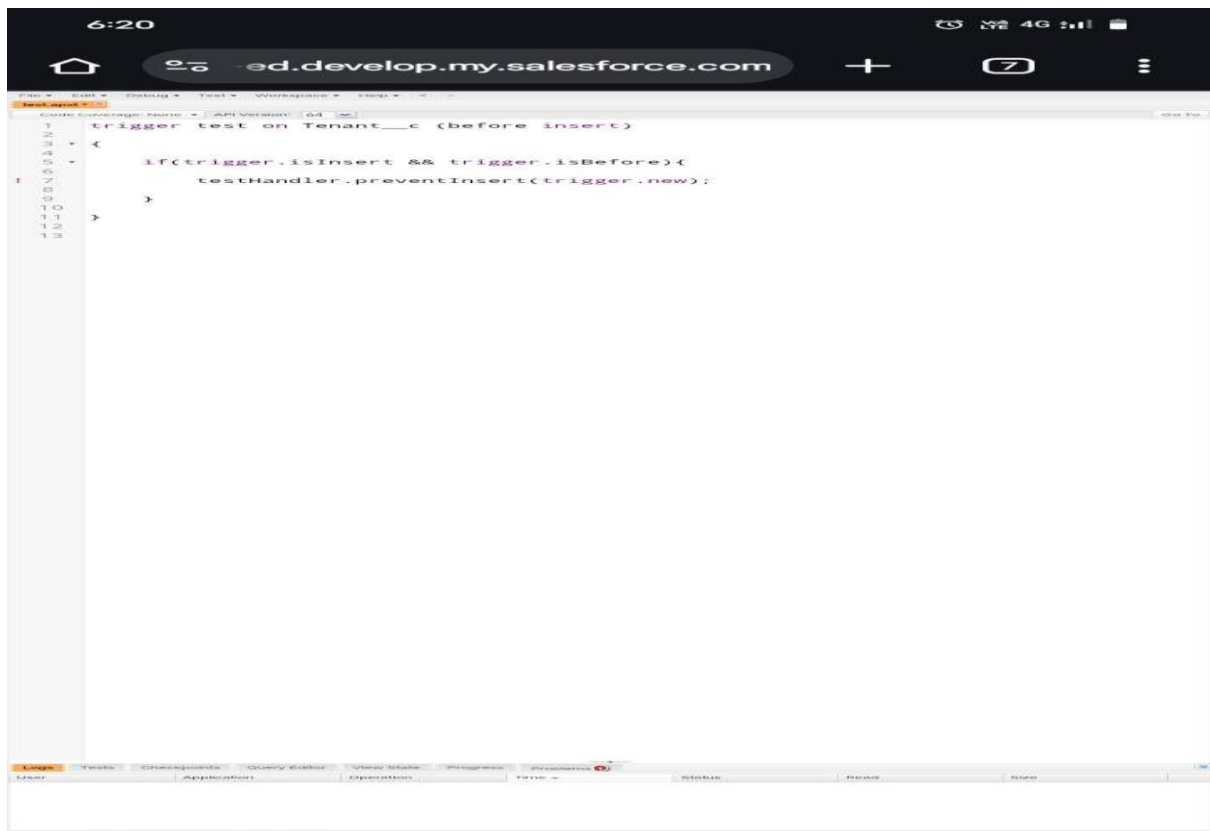
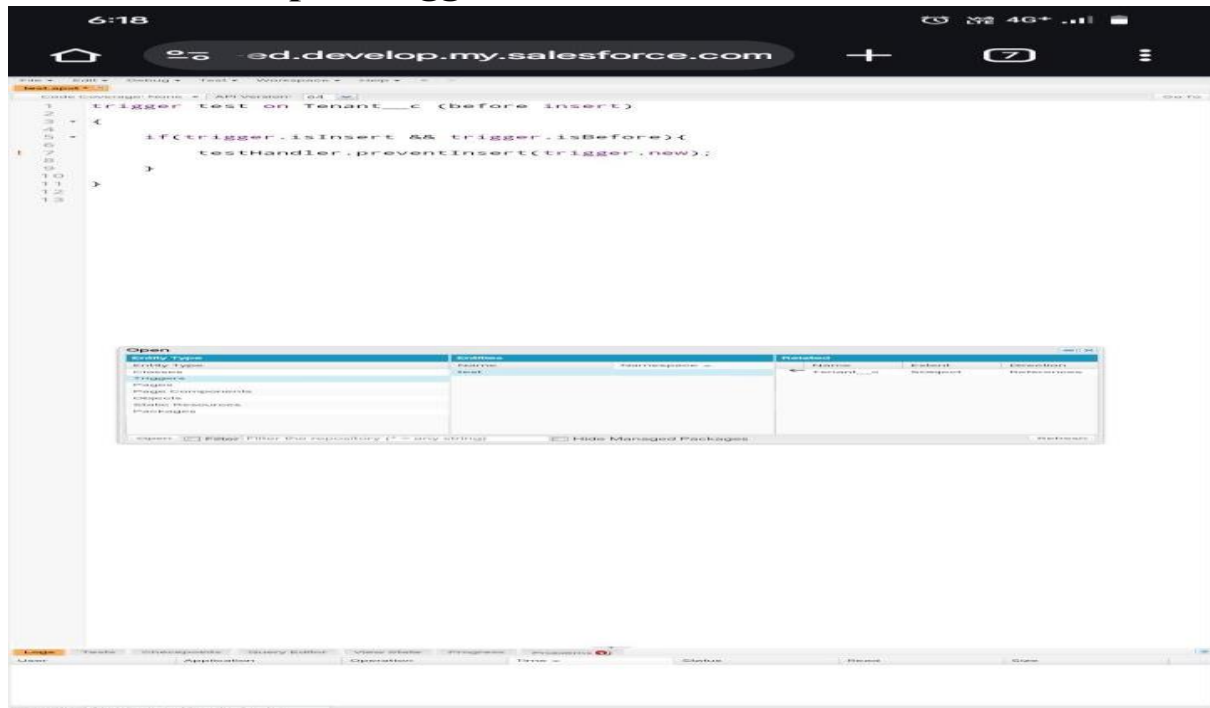
Action	Type	Description
<a href="#">Edit</a>	Record Lock	Unlock the record for editing

[Back To Top](#) [Always show me more records per related list](#)

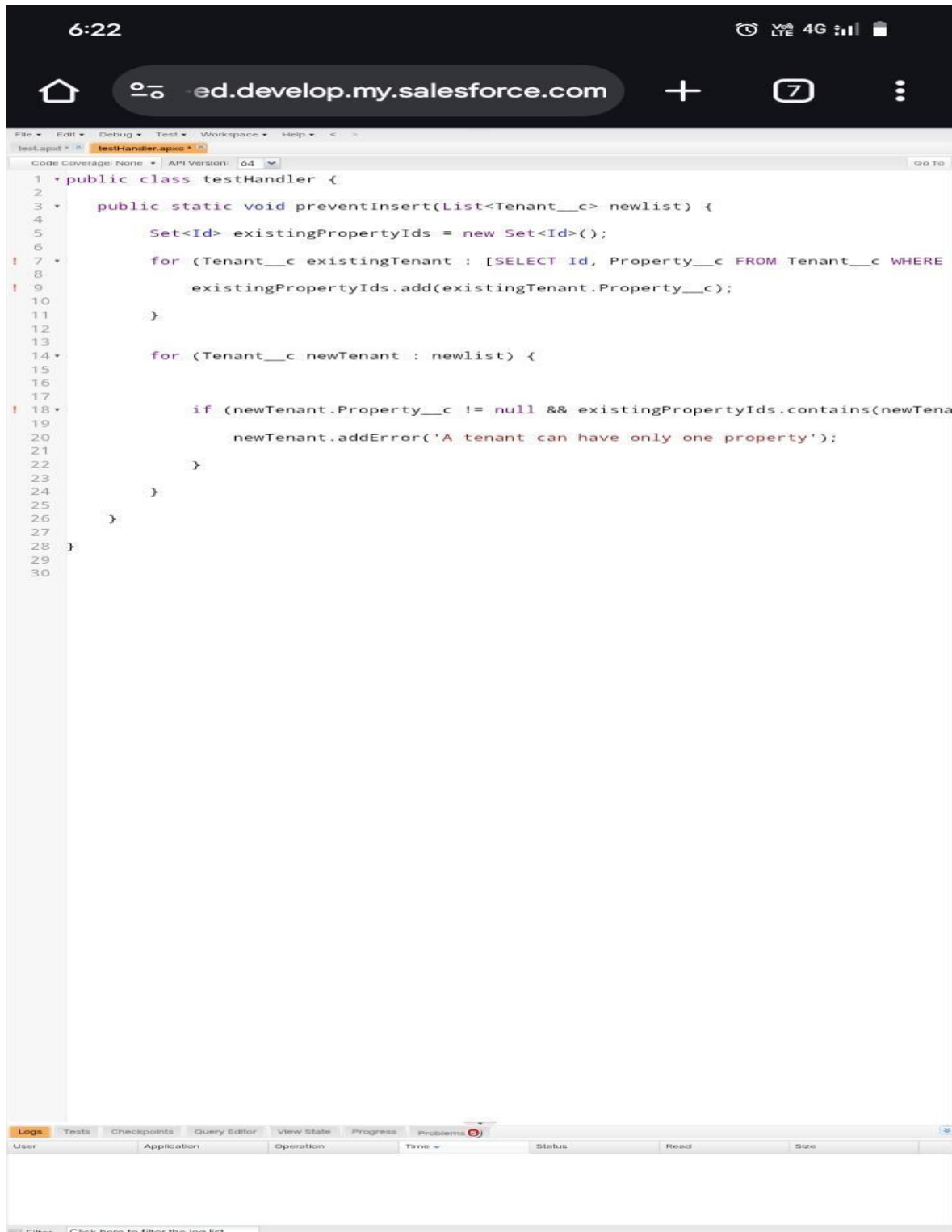


## Apex Trigger :

### Create an Apex Trigger:



## Create an Apex Handler class :



The screenshot shows the Salesforce IDE interface. At the top, the browser address bar displays 'ed.develop.my.salesforce.com'. The IDE window shows a file named 'testHandler.apex' with the following Apex code:

```
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  
8  
9             existingPropertyIds.add(existingTenant.Property__c);  
10  
11         }  
12  
13  
14         for (Tenant__c newTenant : newList) {  
15  
16  
17  
18             if (newTenant.Property__c != null && existingPropertyIds.contains(newTena  
19  
20                 newTenant.addError('A tenant can have only one property');  
21  
22             }  
23  
24         }  
25  
26     }  
27  
28 }  
29  
30
```

The IDE interface includes a menu bar (File, Edit, Debug, Test, Workspace, Help) and a toolbar with options like Code Coverage, API Version, and Go To. The bottom of the screen shows a 'Logs' tab with a table header: User, Application, Operation, Time, Status, Read, Size. A 'Filter' button and a link 'Click here to filter the log list' are also visible.

## FLOWS :

The screenshot displays the Salesforce Flow Builder interface for a "Record-Triggered Flow". The flow is named "monthly payment" and is currently in the "Active" state. The flow diagram on the left shows a sequence of steps: "Start" (Record-Triggered Flow), "Run Immediately", "Send Email" (Action), and "End".

The right-hand pane shows the "Configure Start" configuration for the flow. The "Field" is set to "check for payment", the "Operator" is "Equal", and the "Value" is "1". The "When to Run the Flow for Updated Records" section is configured to "Every time a record is updated and meets the condition requirements". The "Optimize Flow" section includes options for "Fast Field Updates" and "Actions and Related Records". A toggle for "Is this flow making an external callout or connecting to an external system?" is set to "No".

The screenshot displays the Salesforce Flow Builder interface for a "Record-Triggered Flow". The flow is named "monthly payment" and is currently in the "Active" state. The flow diagram on the left shows a sequence of steps: "Start" (Record-Triggered Flow), "Run Immediately", "Send Email" (Action), and "End".

The right-hand pane shows the "Configure Start" configuration for the flow. The "Select Object" section is set to "Payment for tenant". The "Configure Trigger" section is configured to "Trigger the Flow When" a record is "updated". The "Set Entry Conditions" section is configured to "Set Entry Conditions" for the flow. A dropdown menu for "Condition Requirements" is set to "All Conditions Are True (AND)".

## Schedule class:

### Create an Apex Class

The screenshot displays the Salesforce IDE interface. At the top, the browser address bar shows 'ed.develop.my.salesforce.com'. The main editor window is titled 'MonthlyEmailScheduler.apex' and contains the following Apex code:

```
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
24
25             String emailSubject = 'Reminder: Monthly Rent Payment Due';
26
27             // Message()
28         }
29     }
30 }
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49 }
```

An 'Open' dialog box is overlaid on the code editor. It has two main sections: 'Entities' and 'Related'.

Entity Type	Entity Name	Namespace	Related Name	Extent	Direction
Classes	test-handler		Email	CustomField	References
Triggers	MonthlyEmailScheduler		Tenant__c	SObject	References
Pages			Tenant__c	SObject	References
Page Components					
Objects					
Static Resources					
Packages					

The 'Open' dialog box also includes a 'Filter' field with the text 'Filter the repository (\* = any string)' and a 'Hide Managed Packages' checkbox. A 'Refresh' button is located at the bottom right of the dialog.

At the bottom of the IDE, there is a 'Logs' tab and a table with columns: User, Application, Operation, Time, Status, Read, and Size. Below the table is a 'Filter' button and a link 'Click here to filter the log list'.

6:34

ed.develop.my.salesforce.com

+

8

FileEditDebugTestWorkspaceHelp<>

MonthlyEmailScheduler.apxc

Code Coverage: NoneAPI Version: 64Go To

1global class MonthlyEmailScheduler implements Schedulable {  
2  
3global void execute(SchedulableContext sc) {  
4  
5Integer currentDay = Date.today().day();  
6  
7if (currentDay == 1) {  
8  
9sendMonthlyEmails();  
10  
11}  
12  
13}  
14  
15  
16public static void sendMonthlyEmails() {  
17  
18  
19List<Tenant\_\_c> tenants = [SELECT Id, Email\_\_c FROM Tenant\_\_c];  
20  
21  
22  
23  
24for (Tenant\_\_c tenant : tenants) {  
25  
26String recipientEmail = tenant.Email\_\_c;  
27  
28String emailContent = 'I trust this email finds you well. I am writing to  
29  
30String emailSubject = 'Reminder: Monthly Rent Payment Due';  
31  
32  
33Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();  
34  
35email.setToAddresses(new String[]{recipientEmail});  
36  
37email.setSubject(emailSubject);  
38  
39email.setPlainTextBody(emailContent);  
40  
41  
42  
43Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});  
44  
45  
46  
47  
48  
49



LogsTestsCheckpointsQuery EditorView StateProgressProblems






UserApplicationOperationTimeStatusReadSize









FilterClick here to filter the log list


# Schedule Apex class

6:40


Vol 4G+  

  ed.develop.lightning.force.com   

 Setup


Home

Object Manager 

Custom Code

Apex Classes

Didn't find what you're looking for?  
Try using Global Search.

 SETUP

Apex Classes

Apex Class

MonthlyEmailScheduler

Apex Class Detail

Edit Delete Download Security Show

Name MonthlyEmailScheduler

Namespace Prefix

Created By Banjan1.S. 9/5/2025, 9:21 PM

Class Body Class Summary Version Settings Trace Flags

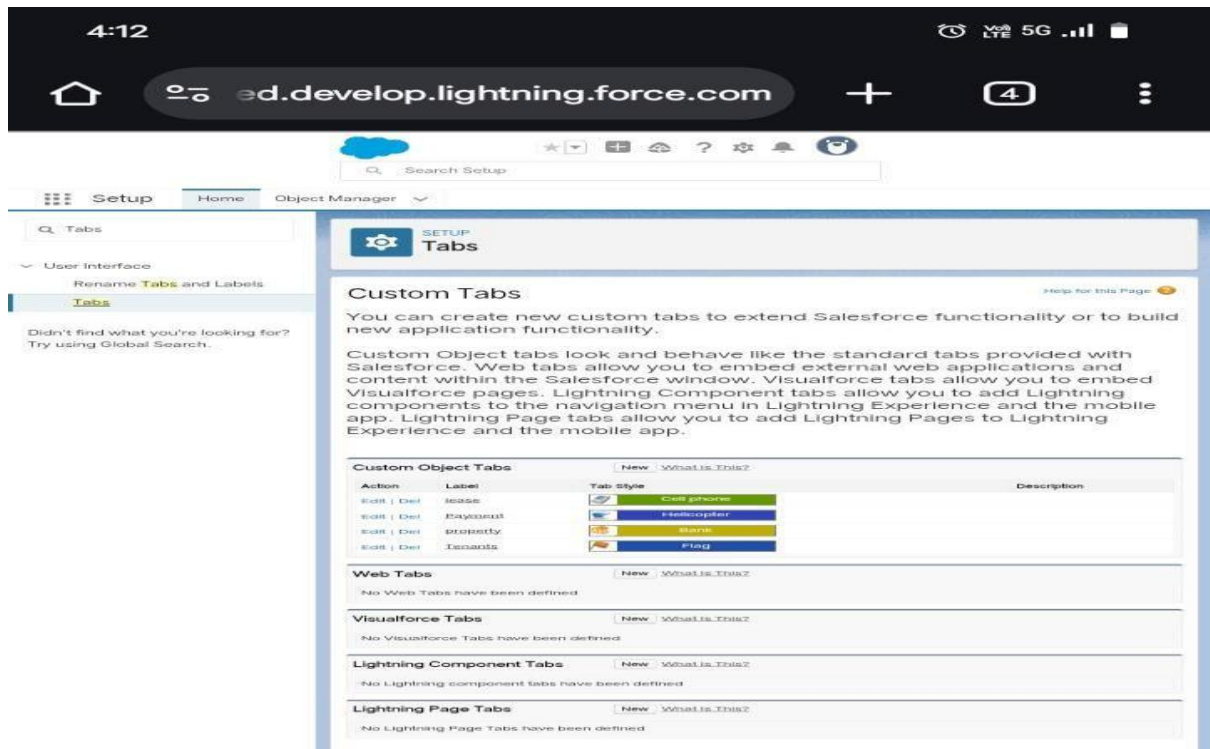
```
1 global class MonthlyEmailScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8
9             sendMonthlyEmails();
10
11         }
12
13     }
14
15     public static void sendMonthlyEmails() {
16
17         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18
19
20
21
22
23
24         for (Tenant__c tenant : tenants) {
25
26             String recipientEmail = tenant.Email__c;
27
28             String emailContent = 'I trust this email finds you well. I am writing to remind you that the monthly rent is c
29
30             String emailSubject = 'Reminder: Monthly Rent Payment Due';
31
32
33             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
34             email.setToAddresses(new String[] {recipientEmail});
35
36             email.setSubject(emailSubject);
37
38             email.setPlainTextBody(emailContent);
39
40
41
42             Messaging.sendEmail(new Messaging.SingleEmailMessage[] {email});
43
44         }
45
46     }
47
48 }
49 }
```

Edit Delete Download Security Show

# RESULTS

## Output Screenshots

### Tabs for Property, Tenant, Lease, Payment



### Flow runs:





## Approval process notifications



## ADVANTAGES & DISADVANTAGES:

**Advantages :** Financial Benefits

Operational Efficiency

Legal & Compliance

Flexibility

Transparency & Control

**Disadvantages:** Higher Long-Term Cost

No Ownership

Restrictions & Limitations

Complexity in Management

## **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

Trigger test on Tenant\_\_c (before insert)

```
{  
  
    If(trigger.isInsert && trigger.isBefore){  
  
        testHandler.preventInsert(trigger.new);  
    }  
  
}
```

**testHandler.apxc:**

```
public class testHandler {  
  
    public static void preventInsert(List<Tenant__c> newList) {  
  
        Set<Id> existingPropertyIds = new Set<Id>();  
  
        For (Tenant__c existingTenant : [SELECT Id, Property__c
```

```

FROM Tenant__c WHERE Property__c != null]) {

    existingPropertyIds.add(existingTenant.Property__c);

}

For (Tenant__c newTenant : newList) {

    If (newTenant.Property__c != null &&
existingPropertyIds.contains(newTenant.Property__c)) {
newTenant.addError('A tenant can have only one property');

    }

}

}

}

```

### **MonthlyEmailScheduler.apxc:**

```

Global class MonthlyEmailScheduler implements Schedulable {

    Global void execute(SchedulableContext sc) {

        Integer currentDay = Date.today().day();
    }
}

```

```
    If (currentDay == 1) {  
        sendMonthlyEmails();  
    }  
}
```

```
Public static void sendMonthlyEmails() {
```

```
    List<Tenant__c> tenants = [SELECT Id, Email__c FROM  
    Tenant__c];
```

```
    For (Tenant__c tenant : tenants) {
```

```
        String recipientEmail = tenant.Email__c;
```

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

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

```
        Messaging.SingleEmailMessage email = new  
        Messaging.SingleEmailMessage();
```

```
Email.setToAddresses(new String[] {recipientEmail});
```

```
Email.setSubject(emailSubject);
```

```
Email.setPlainTextBody(emailContent);
```

```
Messaging.sendEmail(new  
Messaging.SingleEmailMessage[] {email});
```

```
}
```

```
}
```

```
}
```