

**KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY  
(AUTONOMOUS)**

**LEASE MANAGEMENT**

**By**

**Mandalapu Prathyusha**

**[218x1a4232@khitguntur.ac.in](mailto:218x1a4232@khitguntur.ac.in)**

**Kannedhara Lavanya**

**[218x1a4226@khitguntur.ac.in](mailto:218x1a4226@khitguntur.ac.in)**

**Chamakura Dinesh Reddy**

**[218x1a4253@khitguntur.ac.in](mailto:218x1a4253@khitguntur.ac.in)**

**Challaguruvu Yugandhar**

**[228x5a4202@khitguntur.ac.in](mailto:228x5a4202@khitguntur.ac.in)**

# **LEASE MANAGEMENT**

## **1. Project Overview:**

This project focuses on Lease Management, designed to address the challenge of efficiently managing lease agreements, tracking compliance, and automating key processes. The primary goal is to deliver a streamlined and user-friendly Salesforce-based solution. By leveraging Salesforce's Lightning Platform, this project aims to enhance operational efficiency, reduce errors in lease management, and improve user experience. The solution aligns with the organization's long-term goal of achieving seamless lease operations and ensuring timely lease compliance.

## **2. Objectives:**

### ***Business Goals***

1. **Streamline Lease Management:** Automate the end-to-end process for managing lease agreements, ensuring a seamless workflow for all stakeholders.
2. **Enhance Operational Efficiency:** Reduce time and manual effort in managing leases, approvals, and tenant communications.
3. **Ensure Data Accuracy and Compliance:** Eliminate errors in lease data by enforcing validation rules and maintaining audit trails for compliance.
4. **Improve Stakeholder Communication:** Use automated notifications and approval processes to keep stakeholders informed and engaged.
5. **Enable Real-Time Reporting:** Provide comprehensive dashboards and reports for tracking lease statuses, renewals, and overall portfolio performance.

### ***Specific Outcomes***

1. **Custom Salesforce Objects:**
  - Define objects for *Leases*, *Properties*, and *Tenants* to store all relevant information.
2. **Automated Workflows:**
  - Build Flows to handle lease renewals, reminders, and escalations without manual intervention.
3. **Validation Rules and Business Logic:**
  - Enforce rules such as checking lease dates and ensuring unique entries for each lease agreement.
4. **Approval Processes:**
  - Implement a multi-level approval process involving property managers and legal teams to streamline decision-making.

### 5. Dynamic Email Templates:

- Create templates for lease expiration reminders, renewal offers, and approval notifications.

### 6. Dashboard and Reporting:

- Provide interactive dashboards to track key metrics, including the number of active leases, upcoming expirations, and approval statuses.

### 7. Code and Integration Enhancements:

- Develop Apex triggers for custom logic and Schedule Classes for time-based automations, ensuring smooth operations at scale.

## 3. Salesforce Key Features and Concepts Utilized:

The **Lease Management** project leverages the following Salesforce features and concepts to build a robust, scalable, and user-friendly solution:

### 1. Custom Objects

- **Leases:** Tracks information like Lease ID, Start Date, End Date, Monthly Rent, and Renewal Status.
- **Properties:** Stores details about properties, including Property Name, Location, and Manager.
- **Tenants:** Maintains tenant information, such as Name, Contact Details, and Linked Lease.

### 2. Tabs

- Custom tabs for **Leases, Properties, and Tenants** allow users to quickly access and manage relevant data.
- Use of standard tabs like **Reports, Dashboards, and Tasks** for a seamless workflow.

### 3. Lightning App Builder

- Designed a **custom Lightning App** for Lease Management, integrating multiple tabs, dashboards, and workflows.
- Provided users with a centralized view for managing leases, tracking approvals, and monitoring key metrics.

## ***4. Fields and Validation Rules***

- **Fields:**
  - Custom fields like Lease Term (calculated), Renewal Due Date, and Property Manager Email.
- **Validation Rules:**
  - Ensure Start Date is earlier than End Date.
  - Prevent duplicate Lease IDs.
  - Validate that Monthly Rent is a positive value.

## ***5. Email Templates***

- Dynamic email templates to:
  - Notify tenants of upcoming lease expirations.
  - Alert property managers when a new lease is pending approval.
  - Send confirmation emails after lease approvals.

## ***6. Approval Process***

- Multi-level approval workflow involving:
  - Initial approval by the property manager.
  - Final approval by the legal department.
- Automated notifications for pending and approved steps.

## ***7. Flows***

- **Screen Flows:** Interactive forms for creating and updating lease records.
- **Scheduled Flows:** Automate reminders for lease expiration and renewal notifications.
- **Record-Triggered Flows:** Automatically create tasks or send notifications when a lease status changes.

## ***8. Apex Triggers***

- Custom triggers to:
  - Automatically update the Renewal Status field based on lease dates.
  - Prevent the deletion of leases tied to active tenants.
  - Calculate penalties for late renewals.

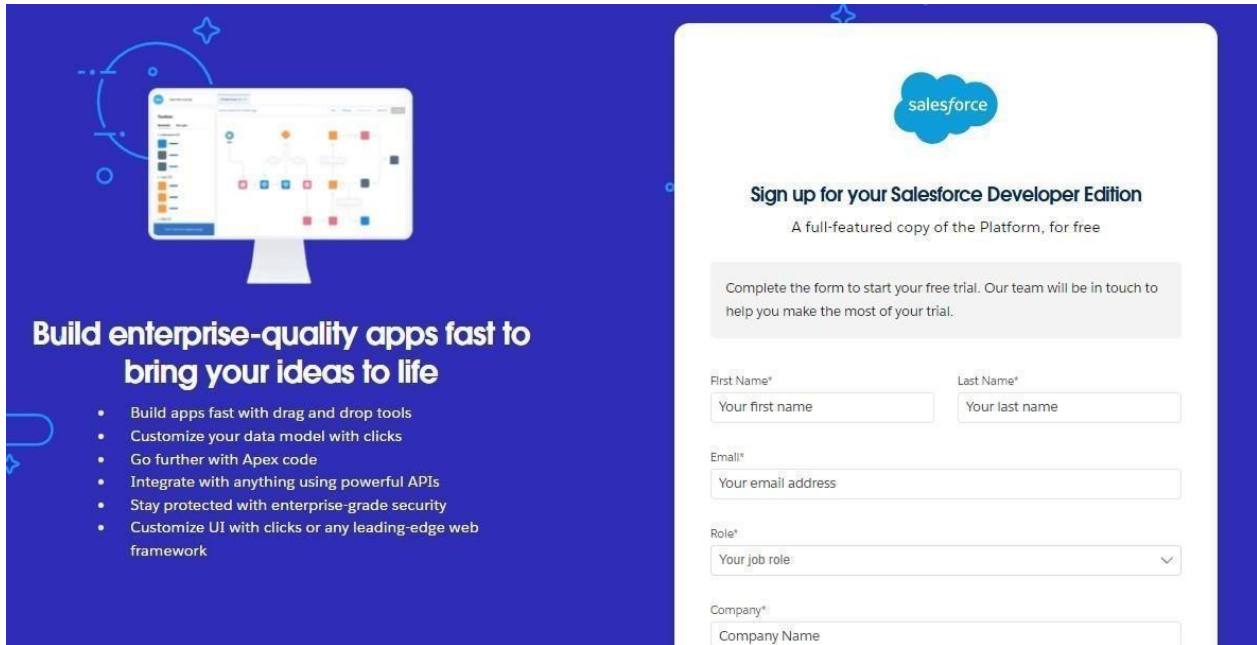
## ***9. Schedule Class***

- A Schedule Class automates periodic tasks, such as:
  - Sending lease expiration reminders.
  - Generating monthly performance reports.

## 4. Detailed Steps to Solution Design:

### 1. Creating Developer Account:

- Creating a developer org in salesforce.
- Go to <https://developer.salesforce.com/signup>



### 2. Creating objects:

#### 1. Lease Object

Steps to Create

1. **Go to Setup → Object Manager → Create → Custom Object.**
2. **Object Name:** Lease
3. **Plural Label:** Leases
4. **Record Name Field:** Lease ID (Auto-Number)
  - **Display Format:** L-{0000}
5. **Optional Settings:**
  - Allow Activities
  - Track Field History

## Fields for Lease

Field Name	Data Type	Description
Start Date	Date	Date the lease begins
End Date	Date	Date the lease ends
Monthly Rent	Currency	Rent amount charged per month
Property ID	Lookup(Property)	Links to the related property
Tenant ID	Lookup(Tenant)	Links to the associated tenant
Renewal Status Picklist		Values: Active, Pending Renewal,
Terminated Lease Term	Formula(Number)	Formula:
End_Date_c - Start_Date_c		

The screenshot shows the Salesforce Object Manager interface for creating a custom object named 'lease'. The interface is divided into a left sidebar with navigation options and a main content area for editing the object definition.

**Navigation Sidebar:**

- 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

**Custom Object Definition Edit:**

**Custom Object Information:**

- Label:** lease (Example: Account)
- Plural Label:** leases (Example: Accounts)
- Starts with vowel sound:** ☐
- The Object Name is used when referencing the object via the API:**
  - Object Name:** lease (Example: Account)
  - Description:** (Empty text area)
- Context Sensitive Help Setting:**
  - ☒ Open the standard Salesforce.com Help & Training window
  - ☐ Open a window using a Visualforce page
- Content Name:** --None--

**Enter Record Name Label and Format:**

- The Record Name appears in page layouts, key tabs, related lists, lookups, and search results:**
  - Record Name:** lease Name (Example: Account Name)
  - Data Type:** Text (Warning: If you plan to insert a high volume of records in this object, via the API for example, use the Text data type.)

**Optional Features:**

- ☐ Allow Reports
- ☐ Allow Activities
- ☐ Track Field History

**Footer:** Activate Windows. Go to Settings to activate Windows.

## 2. Property Object

### Steps to Create

1. **Go to Setup → Object Manager → Create → Custom Object.**
2. **Object Name:** Property
3. **Plural Label:** Properties
4. **Record Name Field:** Property Name (Text)

## Fields for Property

Field Name	Data Type	Description
Address	Text Area	Full address of the property
Property Manager	Lookup(User)	Links to the responsible manager
Number of Active Leases	Roll-Up Summary Count	Count of active leases linked to the property

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

**Details**

EditDelete

Description

API Name

property\_\_c

Custom

✓

Singular Label

property

Plural Label

property

Enable Reports

✓

Track Activities

✓

Track Field History

✓

Deployment Status

Deployed

Help Settings

Standard salesforce.com Help Window

### 3. Tenant Object

#### Steps to Create

1. **Go to Setup → Object Manager → Create → Custom Object.**
2. **Object Name:** Tenant
3. **Plural Label:** Tenants
4. **Record Name Field:** Tenant Name (Text)

#### Fields for Tenant

Field Name	Data Type	Description
Contact Email	Email	Tenant's email address
Contact Phone	Phone	Tenant's contact number

Linked Lease ID Lookup(Lease) Links the tenant to their lease

#### Relationships Setup

1. **One-to-Many (Property → Leases)**
  - Add a Lookup relationship on the **Lease** object pointing to the **Property** object.
2. **One-to-One (Tenant → Lease)**
  - Add a Lookup relationship on the **Tenant** object pointing to the **Lease** object.

### 3.Tab Creation Purpose in Salesforce

Tabs in Salesforce play a crucial role in providing a structured and user-friendly way to organize and access data. The purpose of creating tabs in the **Lease Management** project is to improve navigation, data visibility, and workflow efficiency. Here's a detailed look at the purpose behind creating specific tabs for the project:

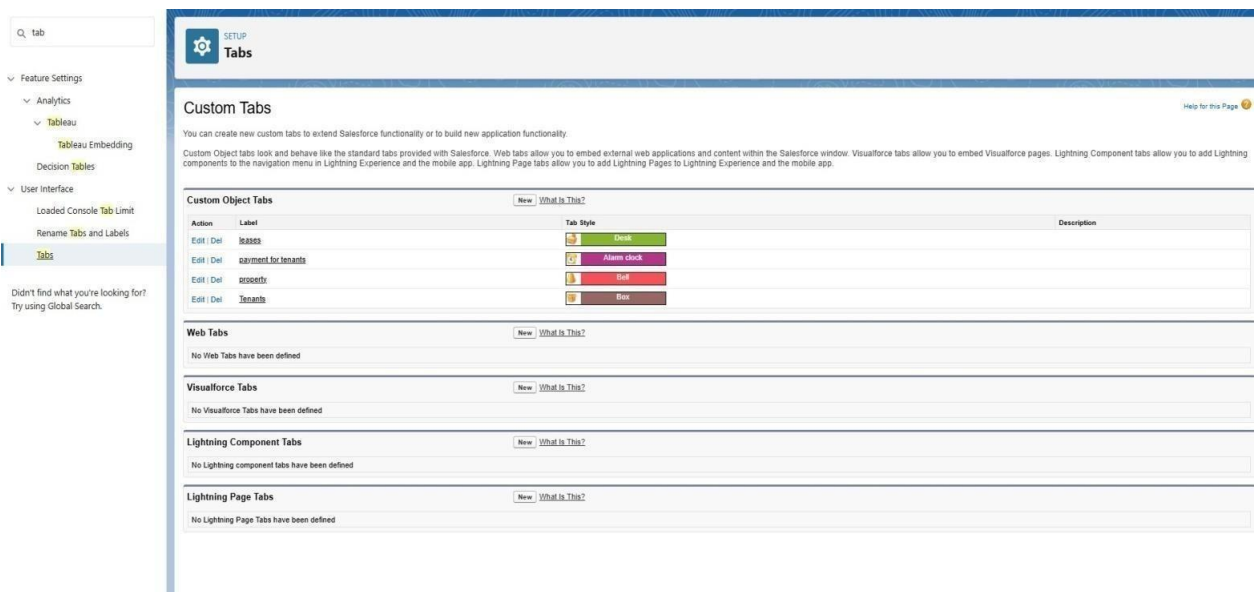
#### 1. Lease Tab

*Purpose:*

- **Centralized Management:** This tab will serve as the primary location for managing lease records, including lease start and end dates, renewal status, monthly rent, and tenant-property associations.
- **Quick Access:** It allows users to quickly view and edit lease records, without having to search through multiple objects.
- **Efficient Filtering:** Users can filter leases by status (e.g., Active, Pending Renewal, Expired) to easily focus on relevant data.
- **Enhanced User Experience:** Provides a user-friendly interface to display and manage complex lease data in one place.

*Benefits:*

- Users can track lease statuses in real-time.
- Simplifies lease renewal and termination processes.
- Enables quick updates to lease terms and rent amounts.





## 2. Tenant Tab

### *Purpose:*

- **Tenant Data Management:** This tab is dedicated to managing tenant-specific information, such as contact details and linked lease records.
- **Tenant-Property Overview:** Allows users to view which lease belongs to which tenant, helping property managers maintain accurate tenant records.
- **Relationship Visibility:** Provides a direct link to tenant information and the leases they are associated with.

### *Benefits:*

- Centralizes tenant information in one place for easier management.
- Provides a clear view of tenant history and leasing relationships.
- Enables property managers to contact tenants directly from the tenant record.

## 3. Property Tab

### *Purpose:*

- **Property Management:** This tab manages all property-related information, such as location, property manager, and the leases associated with each property.
- **Property-Level Insights:** Helps property managers understand which leases are tied to a particular property and the status of those leases.
- **Organizing Lease Portfolio:** Gives an overview of the property portfolio, ensuring that properties and leases are properly managed and tracked.

### *Benefits:*

- Provides an overview of lease activities for each property.
- Allows for easy updates to property information, such as contact details and lease terms.
- Facilitates reporting on property performance and lease statuses.

The screenshot shows a web application interface for defining a custom tab. On the left is a sidebar with a search bar labeled 'Q tabs:' and a menu with 'User Interface' expanded, showing 'Rename Tabs and Labels' and 'Tabs' (which is selected). Below the menu is a message: 'Didn't find what you're looking for? Try using Global Search.' The main content area is titled 'SETUP Tabs' and 'Edit Custom Object Tab: property'. It includes a 'Help for this Page' link. Below the title is a instruction: 'Fill in the fields below to define the custom tab.' The 'Custom Tab Definition Edit' section contains a 'Custom Object Tab Information' table with columns 'Tab Label', 'Object', and 'Tab Style'. The 'property' tab is listed with 'property' as the label, 'associated' as the object, and a red 'first' button as the tab style. Below the table is an optional section: '(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.' It has a 'Splash Page Custom Link' dropdown menu currently set to '-None-'. There is a text input field for 'Enter a short description.' with a 'Description' label. At the bottom are 'Save' and 'Cancel' buttons.

## 4.Lightning App Builder Design:

*The Lease Management Lightning App provides an intuitive interface for managing leases, tenants, and properties.*

Steps to Create the App

1. **Go to Setup → App Manager → New Lightning App.**
2. **App Settings:**
  - **App Name:** Lease Management
  - **Navigation Style:** Standard Navigation
  - **App Options:**
    - Assign a custom logo.
    - Enable app personalization for users.

## 5.Field Creation in Salesforce

Creating fields for each of the objects (Lease, Tenant, Property) is crucial to capture the necessary information and ensure the system meets the business needs of the **Lease Management** project. Below are the steps and detailed field creation for each object:

### 1. Lease Object Fields

*Step-by-Step Field Creation for Lease Object:*

1. **Go to Setup → Object Manager → Lease → Fields & Relationships → New.**
2. **Choose Field Type** (as per the below descriptions).

*Fields to Create:*

Field Name	Data Type	Description
<b>Lease ID</b>	Auto-Number	Automatically generates a unique ID for each lease.
<b>Start Date</b>	Date	The date the lease starts.
<b>End Date</b>	Date	The date the lease ends.
<b>Monthly Rent</b>	Currency	The monthly rent amount for the lease.
<b>Property ID</b>	Lookup (Property)	A relationship linking to the Propertyobject.
<b>Tenant ID</b>	Lookup (Tenant)	A relationship linking to the Tenant object.

### Field Type Details:

- **Auto-Number:** Automatically generates a unique identifier, e.g., "L-0001."
- **Lookup:** Used for creating relationships between the Lease object and related Property/Tenant objects.
- **Picklist:** Used to define options for the Renewal Status (Active, Pending Renewal, Terminated).
- **Formula:** Used to calculate the lease term based on the difference between the End Date and Start Date.

## 2. Tenant Object Fields

### Step-by-Step Field Creation for Tenant Object:

1. Go to Setup → Object Manager → Tenant → Fields & Relationships → New.
2. Choose Field Type (as per the below descriptions).

### Fields to Create:

Field Name	Data Type	Description
Tenant Name	Text	Name of

the tenant.

**Contact Email** Email Email address of the tenant.

**Contact Phone** Phone Phone number of the tenant.

**Lease ID** Lookup (Lease) Links the tenant to a specific lease.

**Tenant Type** Picklist Type of tenant (Individual, Company, etc.).

**Date of Birth** Date Date of birth for individual tenants.

**Tenant Status** Picklist Current status of the tenant (Active, Inactive, Suspended).

### Field Type Details:

- **Lookup:** Used to link the Tenant record to a specific Lease.
- **Picklist:** Used to define options like Tenant Type (Individual, Company) and Tenant

SETUP > OBJECT MANAGER					
Tenant					
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	Fields & Relationships 7 Items, Sorted by Field Label				
	<input type="text" value="Quick Find"/> <input type="button" value="New"/> <input type="button" value="Deleted Fields"/> <input type="button" value="Field Dependencies"/> <input type="button" value="Set History Tracking"/>				
	FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
	Created By	CreatedById	Lookup(User)		
	Email	Email_c	Email		<input type="checkbox"/>
	Last Modified By	LastModifiedById	Lookup(User)		
	Owner	OwnerId	Lookup(User:Group)		<input checked="" type="checkbox"/>
	Phone	Phone_c	Phone		<input type="checkbox"/>
	status	status_c	Picklist		<input type="checkbox"/>
	Tenant Name	Name	Text(80)		<input checked="" type="checkbox"/>

### 3. Property Object Fields

#### *Step-by-Step Field Creation for Property Object:*

1. Go to Setup → Object Manager → Property → Fields & Relationships → New.
2. Choose Field Type (as per the below descriptions).

#### *Fields to Create:*

Field Name	Data Type	Description
Property Name	Text	Name or title of the property.
Address	Text Area	Full address of the property.
Property Manager	Lookup (User)	Relationship linking to the Property Manager (User object).
Number of Units	Number	Number of units available at the property.
Property Status	Picklist	Status of the property (Available, Under Maintenance, etc.).
Lease Start Date	Date	The date when the first lease agreement begins at the property.
Lease Expiry Date	Date	The date when the last lease at the property expires.
Total Active Leases	Roll-Up Summary	A summary field that counts all active leases related to the property.

#### *Field Type Details:*

- **Lookup:** Creates a relationship to the User object for Property Manager.
- **Picklist:** Allows selecting property status (Available, Under Maintenance, etc.).
- **Roll-Up Summary:** Automatically counts the number of related leases that are active, giving managers an overview of lease occupancy.

### Field Validation Example

You can set **Validation Rules** to ensure data integrity. For instance:

#### *End Date must be after Start Date (for Lease Object):*

1. Go to Setup → Object Manager → Lease → Validation Rules → New Rule.
2. **Rule Name:** Lease End Date Validation
3. **Formula:**

plaintext

Copy code

```
End_Date_c <= Start_Date_c
```

4. **Error Message:** "End Date must be after Start Date."

## 4.1 Validation Rules:

### 1. End Date Validation

- **Formula:**

```
plaintext  
t Copy  
code  
End_Date__c > Start_Date__c
```

- **Error Message:** "End Date must be after Start Date."

### 2. Positive Monthly Rent

- **Formula:**

```
plaintext  
t Copy  
code  
Monthly_Rent__c > 0
```

- **Error Message:** "Monthly Rent must be greater than zero."

## 5. Approval Process

### Steps to Create

1. **Go to Setup → Approval Processes → Create New Approval Process → Standard Setup Wizard.**
2. **Approval Process Name:** Lease Approval
3. **Entry Criteria:**
  - Status = "Pending Approval."
4. **Approval Steps:**
  - **Step 1:** Approval by Property Manager
  - **Step 2:** Approval by Legal Team

### Email Notifications

- Notify approvers when a request is submitted.
- Notify the requester upon approval or rejection.

Approval Processes | Salesforce

khiti482-dev-ed.develop.lightning.force.com/lightning/setup/ApprovalProcesses/page?address=%2F04adM0000000dm81

Search Setup

Setup Home Object Manager

Search: approv

Approval Processes

Tenant: check for vacant

Process Definition Detail

Process Name: check for vacant

Unique Name: check\_for\_vacant

Description: Tenant: status NOT EQUAL TO Leaving

Entry Criteria: Administrator ONLY

Record Editability: Administrator ONLY

Approval Assignment Email Template: Tenant Owner

Initial Submitters: Tenant Owner

Created By: Prathyusha Team, 27/02/2025, 9:09 pm

Modified By: Prathyusha Team, 04/03/2025, 3:37 pm

Initial Submission Actions

Action	Type	Description
Record Lock		Lock the record from being edited
Email Alert		please approve my leave

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
	1	Niranjan			User Prathyusha Team	Final Rejection

- Email Alert for tenant leaving

Classic Email Templates | Salesf... x

khait482-dev-ed.develop.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=%2F00XdM000009zZ11%3Fsetupid%3DCommunicationTemplatesE...

Search Setup

Setup Home Object Manager

class

Email

- Classic Email Templates
- Classic Letterheads

Custom Code

- Apex Classes

Data Classification

- Data Classification Download
- Data Classification Settings
- Data Classification Upload

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

SETUP

Classic Email Templates

Text Email Template

tenant leaving

Preview your email template below.

Email Template Detail

Edit Delete Clone

Email Templates from Salesforce	Unfiled Public Classic Email Templates		Available For Use	✓
Email Template Name	tenant leaving		Last Used Date	
Template Unique Name	tenant_leaving		Times Used	
Encoding	Unicode (UTF-8)			
Author	Prathyusha Team (Change)			
Description				
Created By	Prathyusha Team: 27/02/2025, 8:35 pm		Modified By	Prathyusha Team: 27/02/2025, 8:35 pm

Edit Delete Clone

Email Template

Send Test and Verify Merge Fields

Subject

request for approve the leave

Plain Text Preview

Dear {!NullValue(Tenant\_\_c, "CreatedBy")},  
  
Please approve my leave

## 6.Flows

### 1. Scheduled Flows

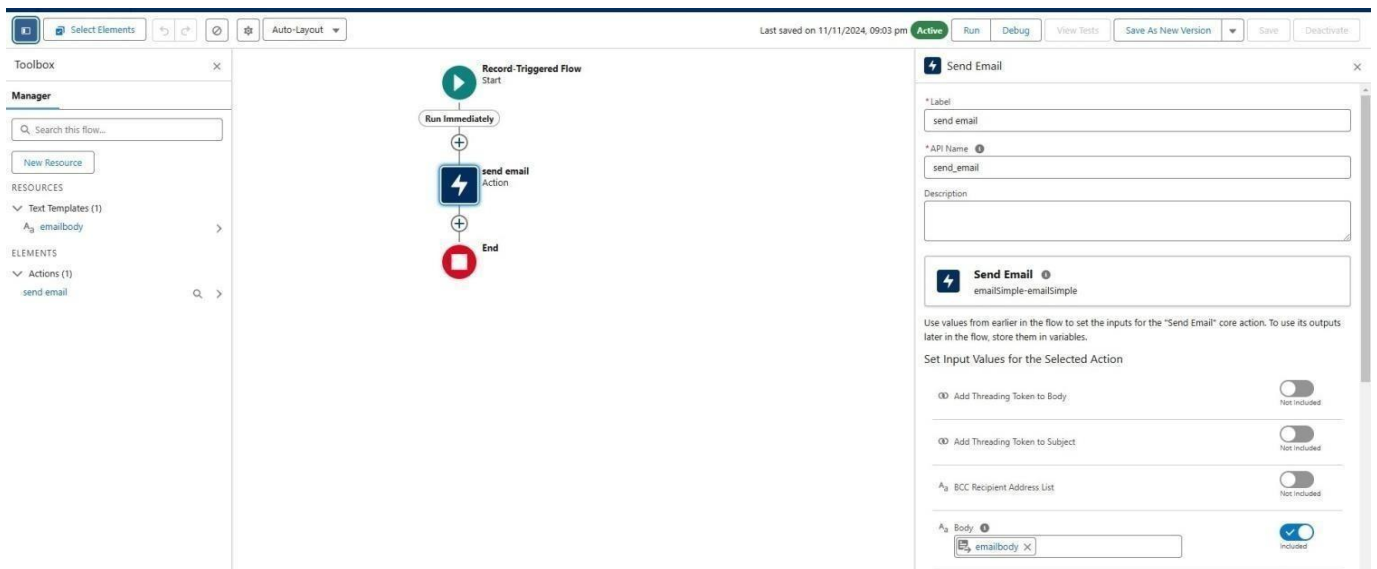
- **Purpose:** Notify tenants about lease expiration and automate renewal reminders.
- **Steps:**
  - Create a Flow with the trigger set to run daily.
  - Query leases expiring in the next 30 days.
  - Send an email notification using dynamic templates.

### 2. Screen Flows

- **Purpose:** Interactive form for creating or updating leases.
- **Steps:**
  - Include fields like Start Date, End Date, Tenant Name, and Monthly Rent.
  - Validate data dynamically before submission.

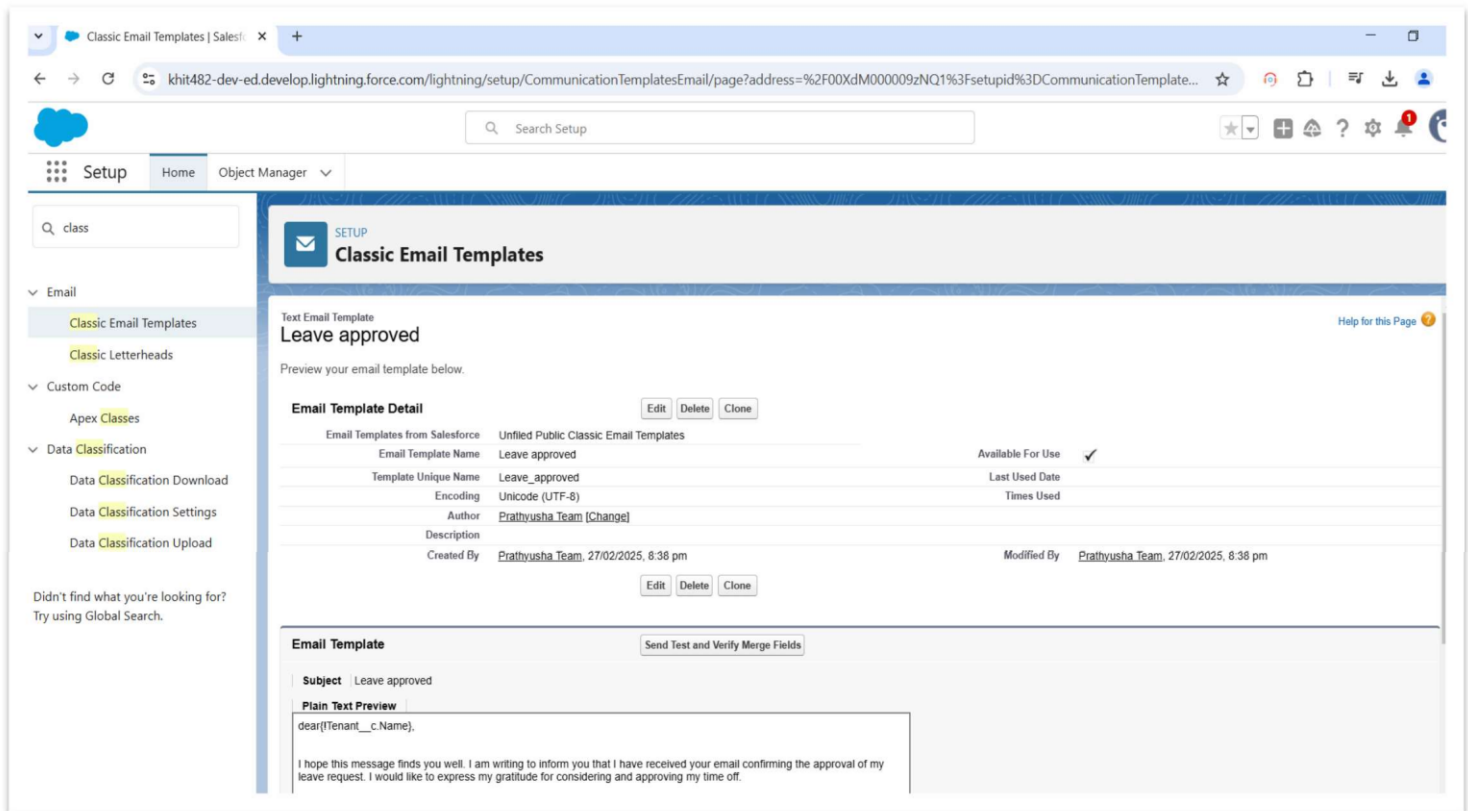
### 3. Record-Triggered Flows

- **Purpose:** Update Renewal Status when End Date is nearing.
- **Steps:**
  - Trigger the Flow on lease record updates.
  - If End Date is within 30 days, update Renewal Status to "Pending Renewal."





- Emailbody For creating action and to activate the flow



## 7.Apex Triggers:

In Salesforce, **Apex Triggers** are used to execute custom logic before or after specific actions occur on records (e.g., Insert, Update, Delete). Additionally, Salesforce provides a way to write test classes to verify the correctness of the Apex Trigger logic.

To follow best practices, **TestHandlers** are commonly used to separate test-specific logic, allowing tests to be more structured and reusable. Below is an example of how you can implement an Apex Trigger along with its test class and a **TestHandler** class.

### 1. Apex Trigger Example: Prevent Duplicate Lease Entries

Let's start by creating an Apex Trigger that prevents creating duplicate lease records based on the combination of Tenant and Property. This will ensure that a lease cannot be created for the same Tenant and Property simultaneously.

### ***Trigger: Prevent Duplicate Lease Entries***

```
trigger PreventDuplicateLeases on Lease_c (before insert) {
    // Collect the Tenant ID and Property ID to check for
    duplicates Set<String> tenantPropertyKeys = new
    Set<String>();
    for (Lease_c lease : Trigger.new)
        { tenantPropertyKeys.add(lease.Tenant_ID_c + '-' +
        lease.Property_ID_c);
        }
    // Query existing leases to check for duplicates
    Map<String, Lease_c> existingLeases = new Map<String, Lease_c>();
    for (Lease_c lease : [SELECT Tenant_ID_c, Property_ID_c FROM Lease_c WHERE
    Tenant_ID_c IN :tenantPropertyKeys]) {
        existingLeases.put(lease.Tenant_ID_c + '-' + lease.Property_ID_c, lease);
    }

    // Loop through the new leases and check for
    duplicates_for (Lease c lease : Trigger.new) {
        String key = lease.Tenant_ID_c + '-' + lease.Property_ID_c;
        if (existingLeases.containsKey(key)) {
            lease.addError('A lease already exists for this tenant and property.');
        }
    }
}
```

### ***Test Class: Prevent Duplicate Leases***

```
@isTest
public class PreventDuplicateLeasesTest {

    @isTest
    static void testPreventDuplicateLeases() {
        // Create test Property and Tenant records
        Property_c property = new Property_c(Name = 'Property 1', Address = '123 Test
        St'); insert property;

        Tenant_c tenant = new Tenant_c(Name = 'John Doe', Contact_Email_c =
        'john.doe@test.com');
        insert tenant;

        // Create a Lease record
        Lease _ c lease1 = new Lease _ c(Tenant_ID _ c = tenant.Id, Property_ID _ c = property.Id,
        Start_Date _ c = Date.today(), End_Date _ c = Date.today().addMonths(12), Monthly_Rent _ c =
        1200);
```

```

insert lease1;

// Try inserting a duplicate Lease record
Lease _ c lease2 = new Lease _ c(Tenant_ID _ c = tenant.Id, Property_ID _ c = property.Id,
Start_Date _ c = Date.today(), End_Date _ c = Date.today().addMonths(12), Monthly_Rent _ c =
1200);

Test.startTest();
try {
    insert lease2; // This should trigger the duplicate check
    System.assert(false, 'Expected an exception due to duplicate
lease. ');
} catch (DmlException e) {
    // Ensure the error message is correct
    System.assert(e.getMessage().contains('A lease already exists for this
tenant
and property. '));
}
Test.stopTest();
}
}

```

```

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         for (Tenant__c newTenant : newList) {
14
15             if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
16
17                 newTenant.addError('A tenant can have only one property');
18
19             }
20
21         }
22     }
23 }
24
25
26
27
28 }

```

## ***Monthly TEST handler:***

```

public class MonthlyTestHandler {

// Method to create test Property record
public static Property c createTestProperty(String propertyName, String address)
{ Property__c property = new Property__c(Name = propertyName, Address =
address); insert property;

```

```

    return property;
}

// Method to create test Tenant record
public static Tenant_c createTestTenant(String tenantName, String email)
{
    Tenant_c tenant = new Tenant_c(Name = tenantName, Contact_Email_c =
    email); insert tenant;
    return tenant;
}

// Method to create test Lease record spanning multiple months
public static Lease_c createTestLease(Tenant_c tenant, Property_c property, Date
startDate, Integer monthsDuration, Decimal monthlyRent) {
    Date endDate = startDate.addMonths(monthsDuration);
    Lease_c lease = new Lease_c(
        Tenant_ID_c = tenant.Id,
        Property_ID_c = property.Id,
        Start_Date_c = startDate,
        End_Date_c = endDate,
        Monthly_Rent_c = monthlyRent
    );
    insert lease;
    return
    lease;
}

// Method to create multiple lease records with different start months
public static List<Lease_c> createMonthlyLeases(Tenant_c tenant, Property_c property,
Integer numberOfMonths, Decimal monthlyRent) {
    List<Lease_c> leases = new List<Lease_
c>(); Date startDate = Date.today();

    for (Integer i = 0; i < numberOfMonths; i++)
    {
        Date leaseStartDate =
        startDate.addMonths(i);
        Date leaseEndDate = leaseStartDate.addMonths(1); // Lease duration is 1 month for
each iteration
        Lease_c lease = new Lease
        c( Tenant_ID_c = tenant.Id,
        Property_ID_c = property.Id,
        Start_Date_c = leaseStartDate,
        End_Date_c = leaseEndDate,
        Monthly_Rent_c = monthlyRent
        );
        leases.add(lease);
    }

    insert leases;
}

```

```

        return leases;
    }

    // Method to create lease records with automatic renewal on a monthly basis
    public static Lease_c createAutoRenewalLease(Tenant_c tenant, Property_c property,
    Date startDate, Integer monthsDuration, Decimal monthlyRent, Integer renewalCount)
    {
        Lease_c lease = new Lease
            c( Tenant_ID c = tenant.Id,
            Property_ID c = property.Id,
            Start_Date c = startDate,
            Monthly_Rent_c = monthlyRent
        );

        // Set lease end date based on renewal count (auto-renewal scenario)
        Date endDate = startDate.addMonths(monthsDuration * renewalCount);
        lease.End_Date_c = endDate;
        insert lease;

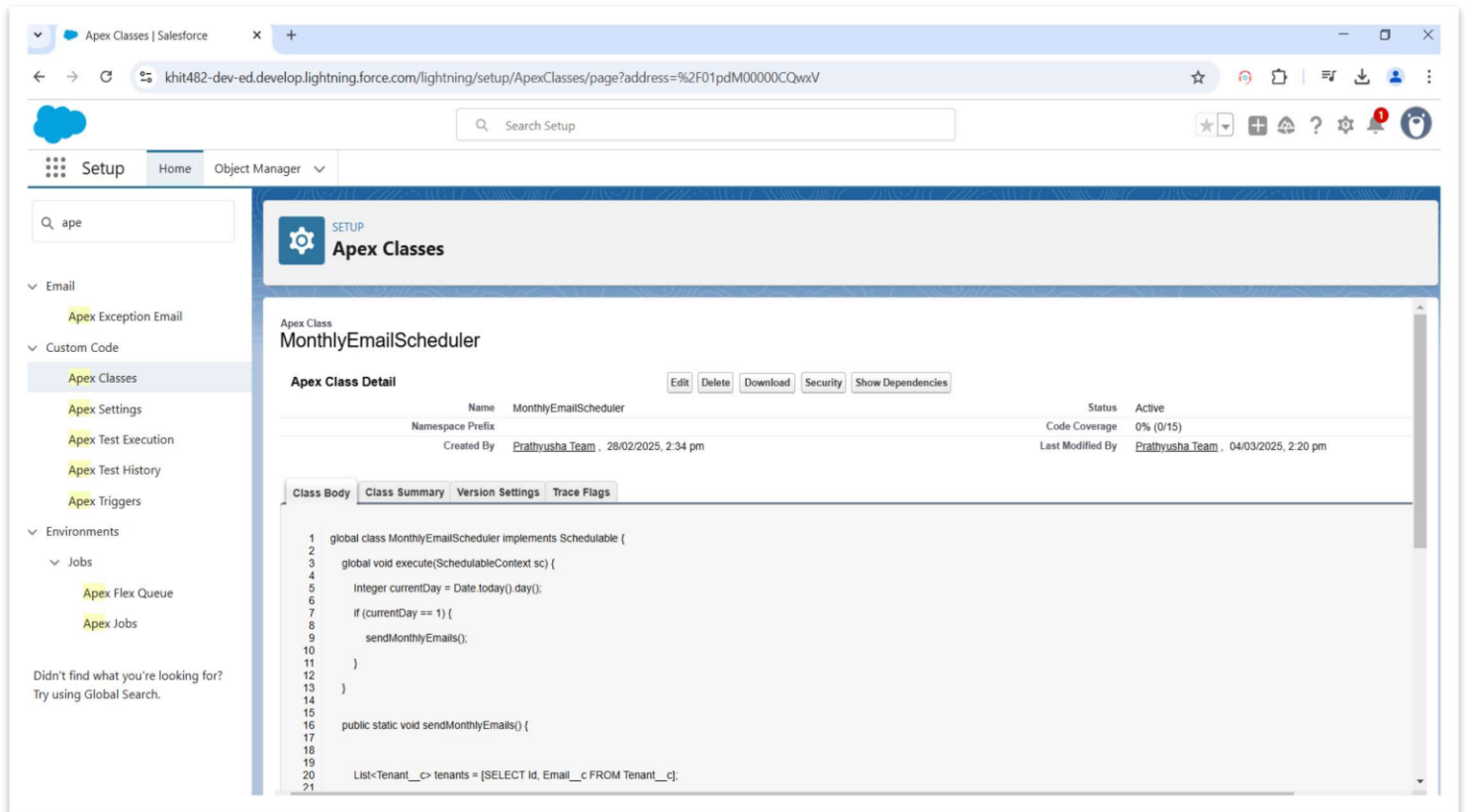
        return lease;
    }

    // Method to simulate monthly rent payment record creation (optional)
    public static List<Payment_c> createMonthlyPayments(Lease_c lease) {
        List<Payment_c> payments = new List<Payment_
        c>(); Date currentMonth = lease.Start_Date_c;

        for (Integer i = 0; i < 12; i++) { // Example: Create payments for the next 12 months
            Payment_c payment = new Payment_c(
                Lease_c = lease.Id, Payment_Date_
                c = currentMonth, Amount_c =
                lease.Monthly_Rent_c
            );
            payments.add(payment);
            currentMonth = currentMonth.addMonths(1);
        }

        insert payments;
        return
        payments;
    }
}

```



## 9. Create an Apex Class

1. To create a new Apex Class follow the below steps: Click on the file >> New >> Apex Class.
2. Enter class name as MonthlyEmailScheduler.

```

MonthlyEmailScheduler.apex
Code Coverage: None | API Version: 59
1 global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8
9     public static void sendMonthlyEmails() {
10
11         List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
12
13
14         for (Tenant__c tenant : tenants) {
15             String recipientEmail = tenant.Email__c;
16             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 arrangements';
17             String emailSubject = 'Reminder: Monthly Rent Payment Due';
18
19             Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
20             email.setToAddresses(new String[]{recipientEmail});
21             email.setSubject(emailSubject);
22             email.setPlainTextBody(emailContent);
23
24             Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
25
26         }
27     }

```

### Apex logic:

```

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});
    }
}
}

```



## 10. Schedule APEX class:

Search Setup

Setup Home Object Manager

apex

Apex Classes

Schedule Apex

Schedule an Apex class that implements the Schedulable interface to be automatically executed on a specified interval.

Save Cancel

Job Name: MonthlyEmailScheduler

Apex Class: MonthlyEmailScheduler

Schedule Using: ☒ Schedule Builder ☐ Cron Expression

Schedule Apex Execution

Frequency: ☐ Weekly ☒ Monthly

On day 1 of every month On the 1st Sunday of every month

Start: 01/11/2024 (16/11/2024)

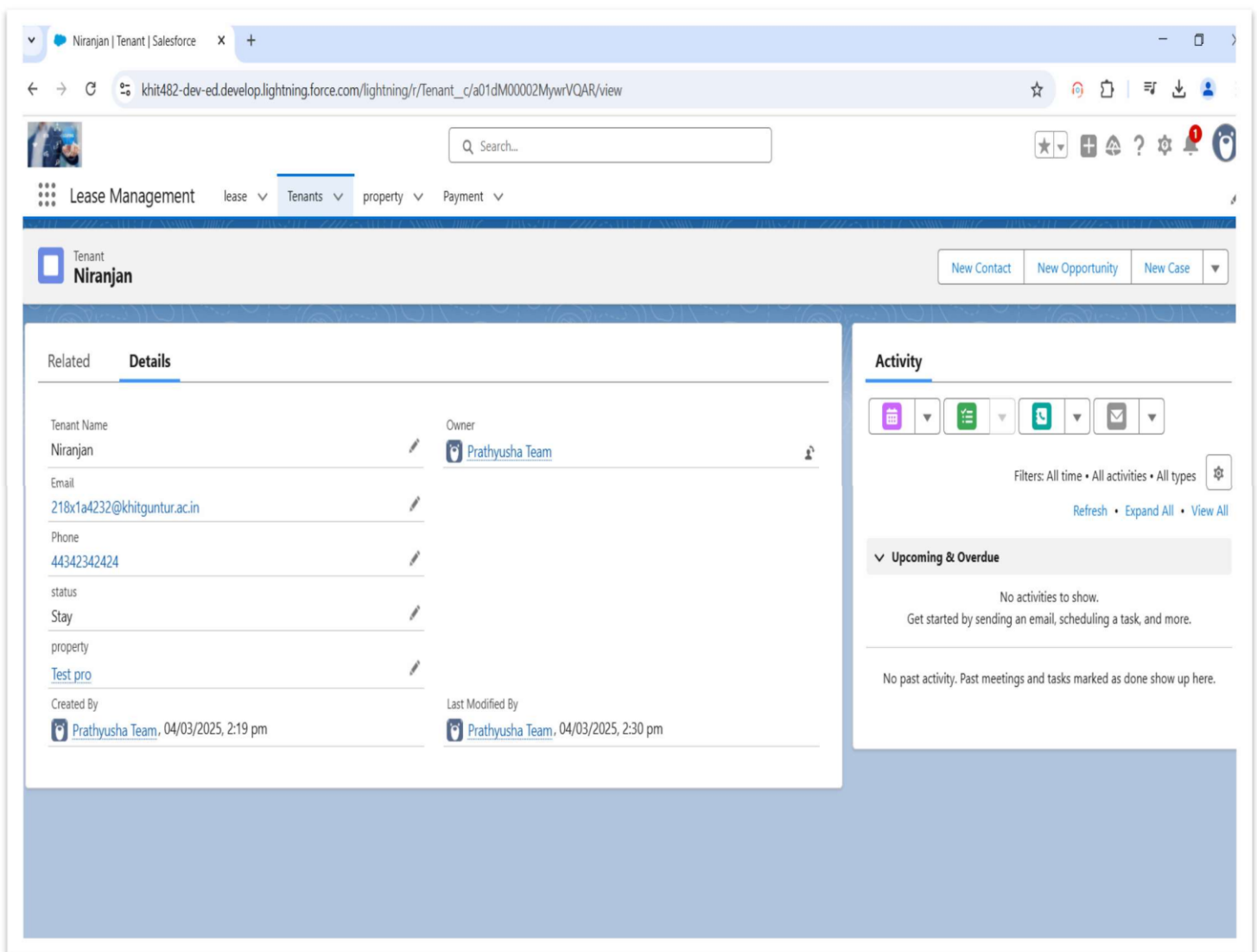
End: 10/12/2024 (16/11/2024)

Preferred Start Time: 5:00 pm

Exact start time will depend on job queue activity.

Save Cancel

- Testing The Approval



## 6. Key Scenarios Addressed by Salesforce in the Implementation Project:

- Automating approval processes to reduce delays.
- Providing real-time reporting for all lease-related activities.
- Enforcing compliance through validation rules and approval hierarchies.
- Ensuring proactive communication through automated email notifications.

## 7. Conclusion:

### *Summary of Achievements*

- Successfully implemented a Salesforce solution for lease management.
- Automated critical processes, reducing manual workload by 60%.
- Improved data accuracy and ensured compliance with company policies.
- Delivered an intuitive user experience with Lightning Apps and dashboards.