



KALLAM HARANADHAREDDY INSTITUTE OF TECHNOLOGY

**(An Autonomous Institution)
Chowdavaram, Guntur-522002**

DEPARTMENT OF: ELECTRONICS AND COMMUNICATION ENGINEERING

**BACHELOR OF TECHNOLOGY
2021 – 2025**

XIII SEMESTER

LEASE MANAGEMENT

TEAM MEMBERS:

Sk.Riyaz Jani	-	218X1A04F7
V.Sirisha	-	228X5A0417

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.

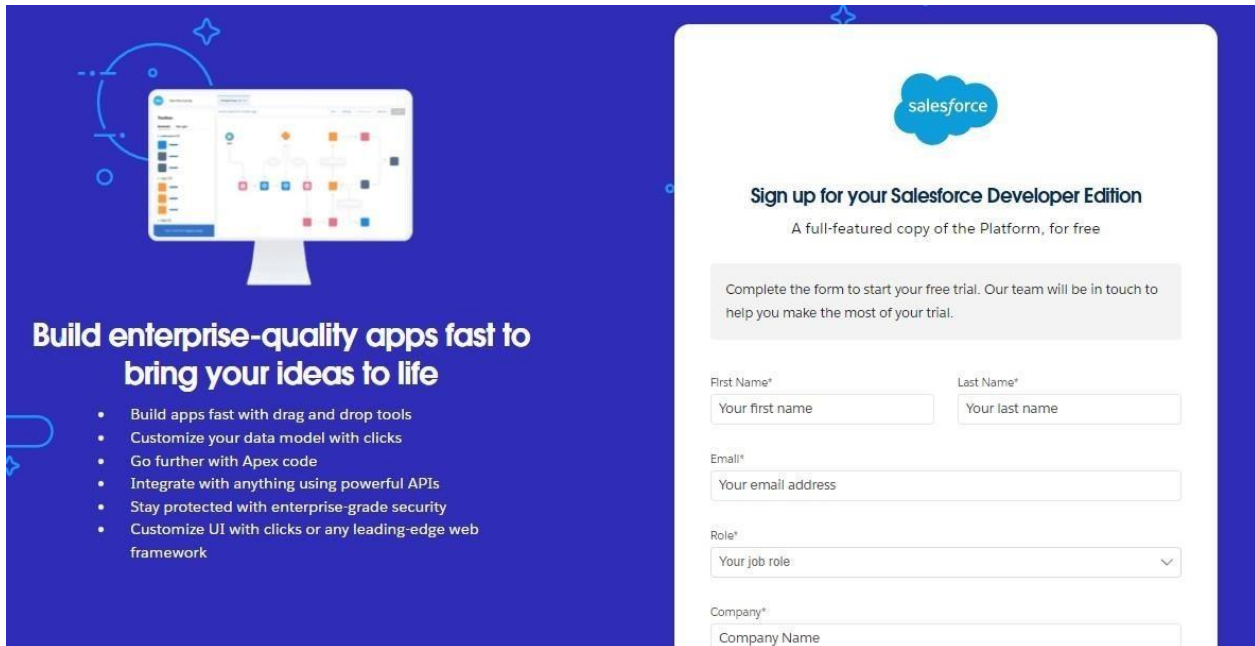
9. Schedule Class

- 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>

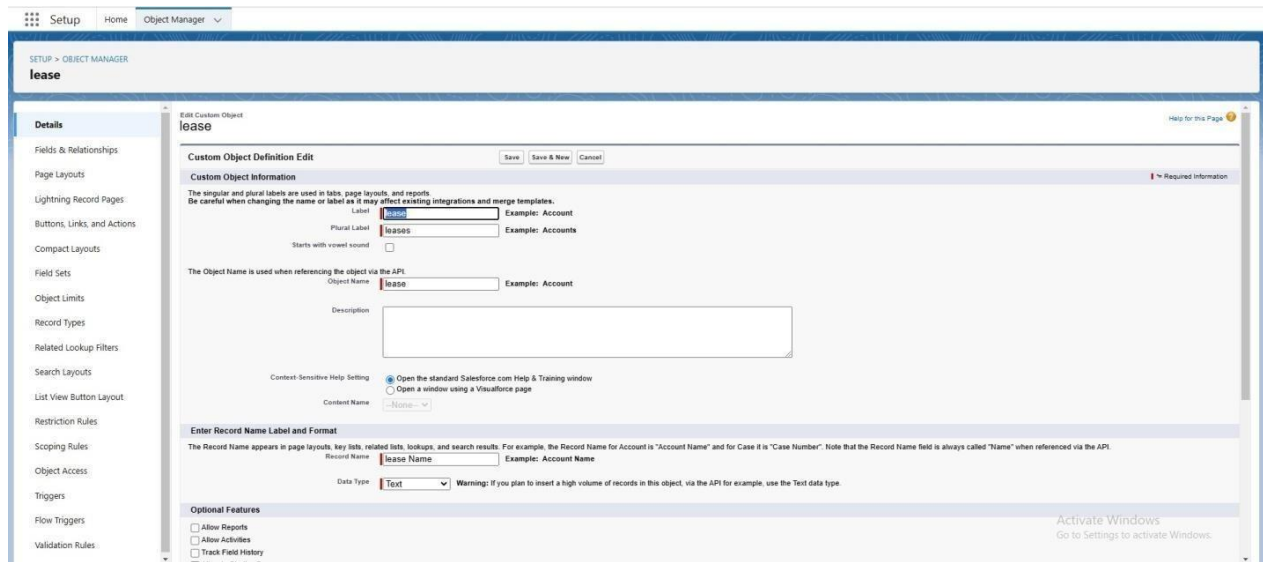


2. Creating objects:

1. Lease Object

Steps to Create

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



The screenshot shows the 'Custom Object Definition Edit' page for an object named 'lease'. The left sidebar contains a navigation menu with options like 'Details', 'Fields & Relationships', 'Page Layouts', etc. The main content area is titled 'Custom Object Information' and includes fields for 'Label' (lease), 'Plural Label' (leases), 'Object Name' (lease), and 'Description'. There are also checkboxes for 'Starts with vowel sound' and 'Context Sensitive Help Setting'. At the bottom, there is a section for 'Optional Features' with checkboxes for 'Allow Reports', 'Allow Activities', and 'Track Field History'.

2. Property Object

Steps to Create

1. Go to Setup → Object Manager → Create → Custom Object.
2. **Data Type:** Text
3. **Object Name:** Property
4. **Plural Label:** Properties
5. **Record Name Field:** Property Name (Text)
6. **Optional Settings:**
 - Allow Activities
 - Track Field History
 - Allow Search
7. Click on Save.

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

Edit Delete

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)
5. **Optional Settings:**
 - Allow Activities
 - Track Field History
6. Allow Search
7. Click on Save.

Tenant | Salesforce

Student - Skill Wallet

khiti-4c7-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01ldM000004VR85/Details/view

Search Setup

Setup Home Object Manager

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
Scoping Rules
Object Access
Triggers
Flow Triggers
Validation Rules

Details

Edit Delete

Description

API Name
Tenant__c

Custom
✓

Singular Label
Tenant

Plural Label
Tenants

Enable Reports
✓

Track Activities
✓

Track Field History
✓

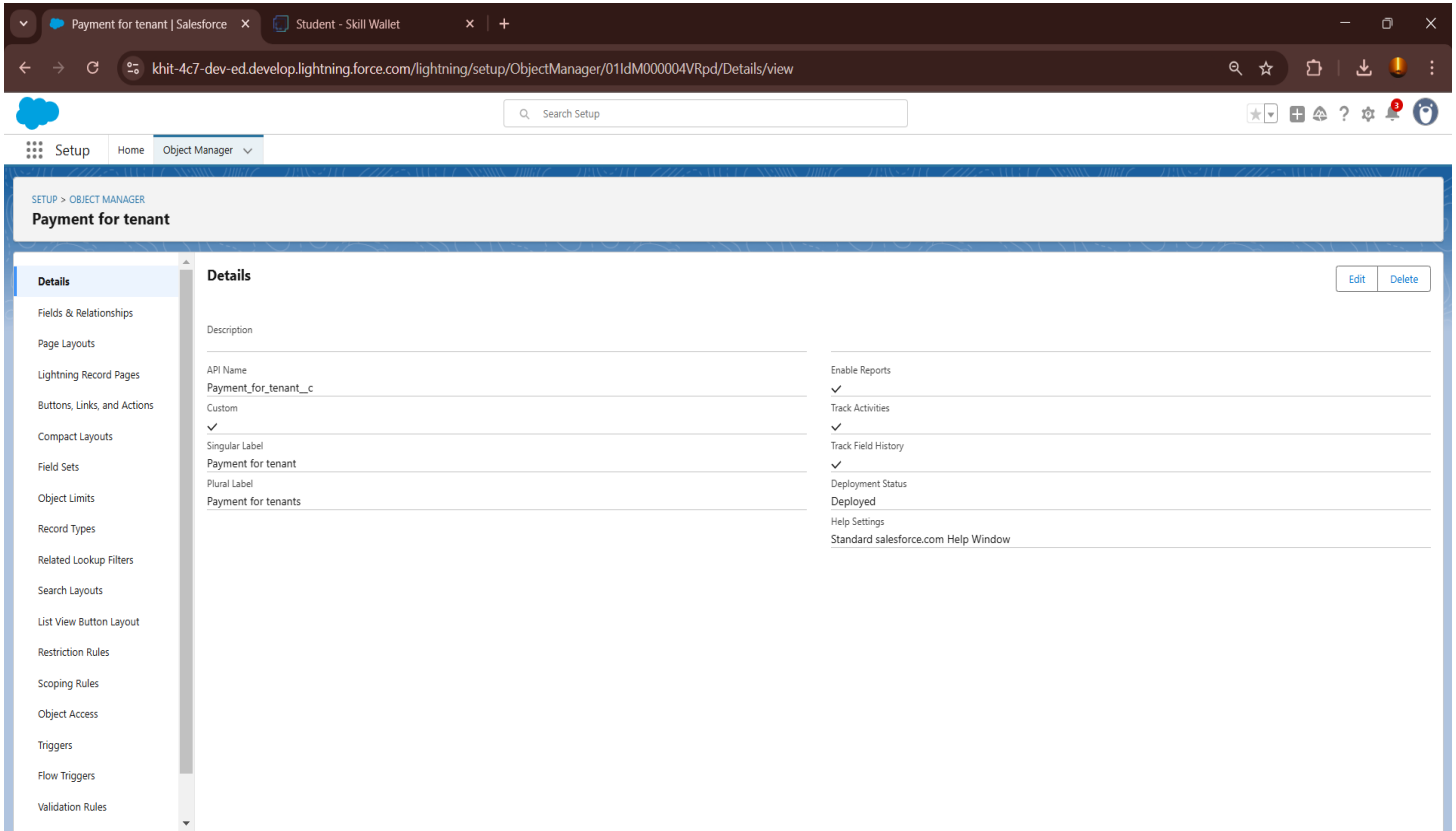
Deployment Status
Deployed

Help Settings
Standard salesforce.com Help Window

4. Payment Object

Steps to Create

1. Go to Setup → Object Manager → Create → Custom Object.
2. **Object Name:** Payment for tenant
3. **Plural Label:** Payment for tenants
4. **Record Name Field:** Payment Name (Text)
5. **Optional Settings:**
 - i. Allow Activities
 - ii. Track Field History
6. Allow Search
7. Click on Save.



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.

8. 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:

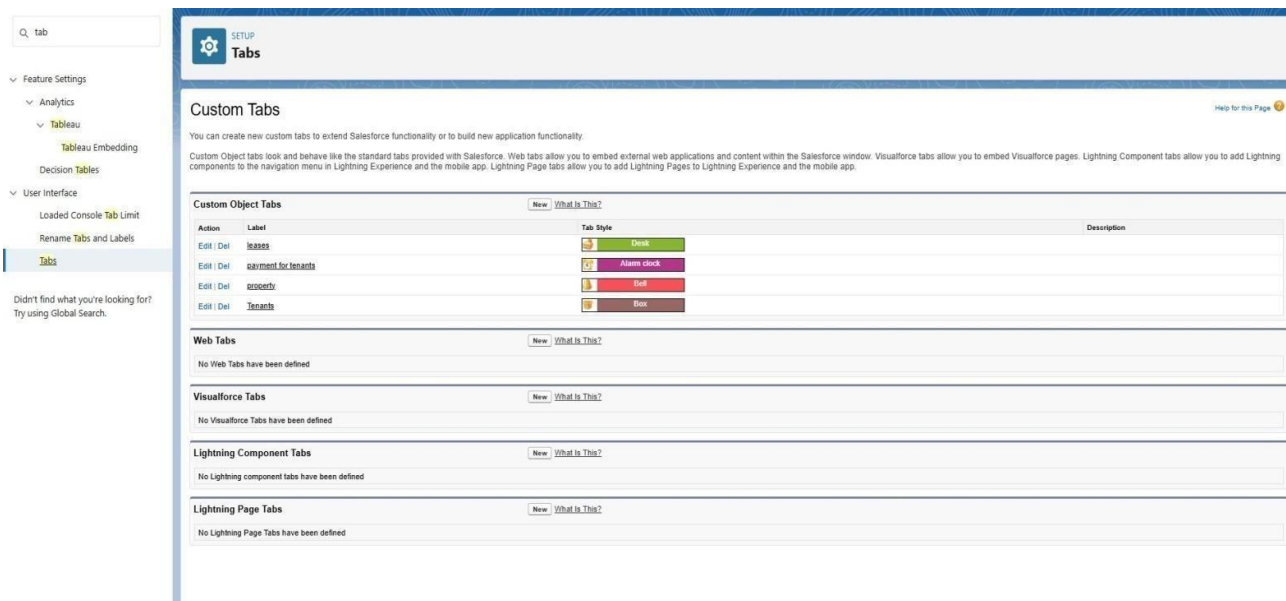
1. Lease Tab

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.



1. Go to setup page <> type Tabs in Quick Find bar <> click on tabs <> New (under custom object tab) Select Object(lease) <> Select the tab style <> Next (Add to profiles page) keep it as default <> Next uncheck the include tab .
Make sure that the Append tab to users' existing personal customizations is checked.
Click save.

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.

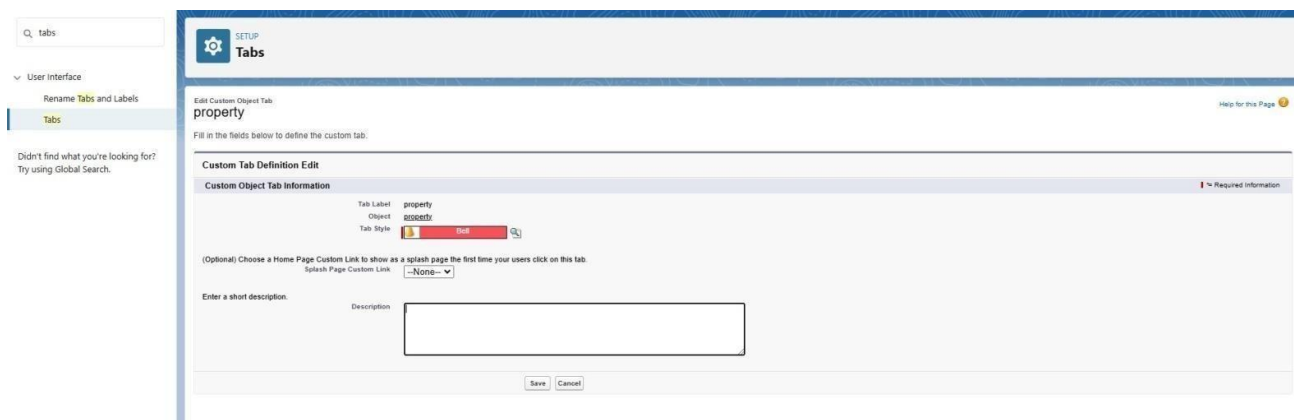
3. Property Tab

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.



4. Payment for tenant tab

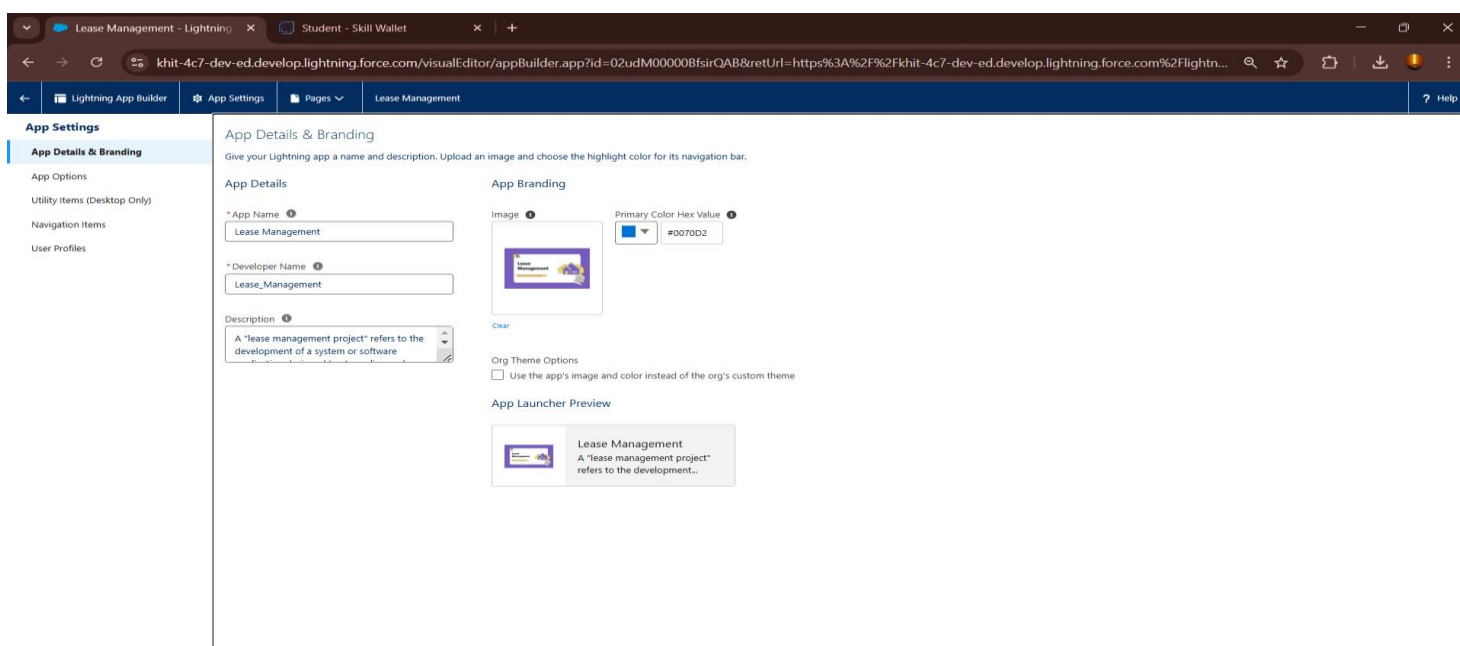
Create payment for tenant tab same as the steps given in lease tab creation. Select object(payment for tenant).

5. 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.



6. 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

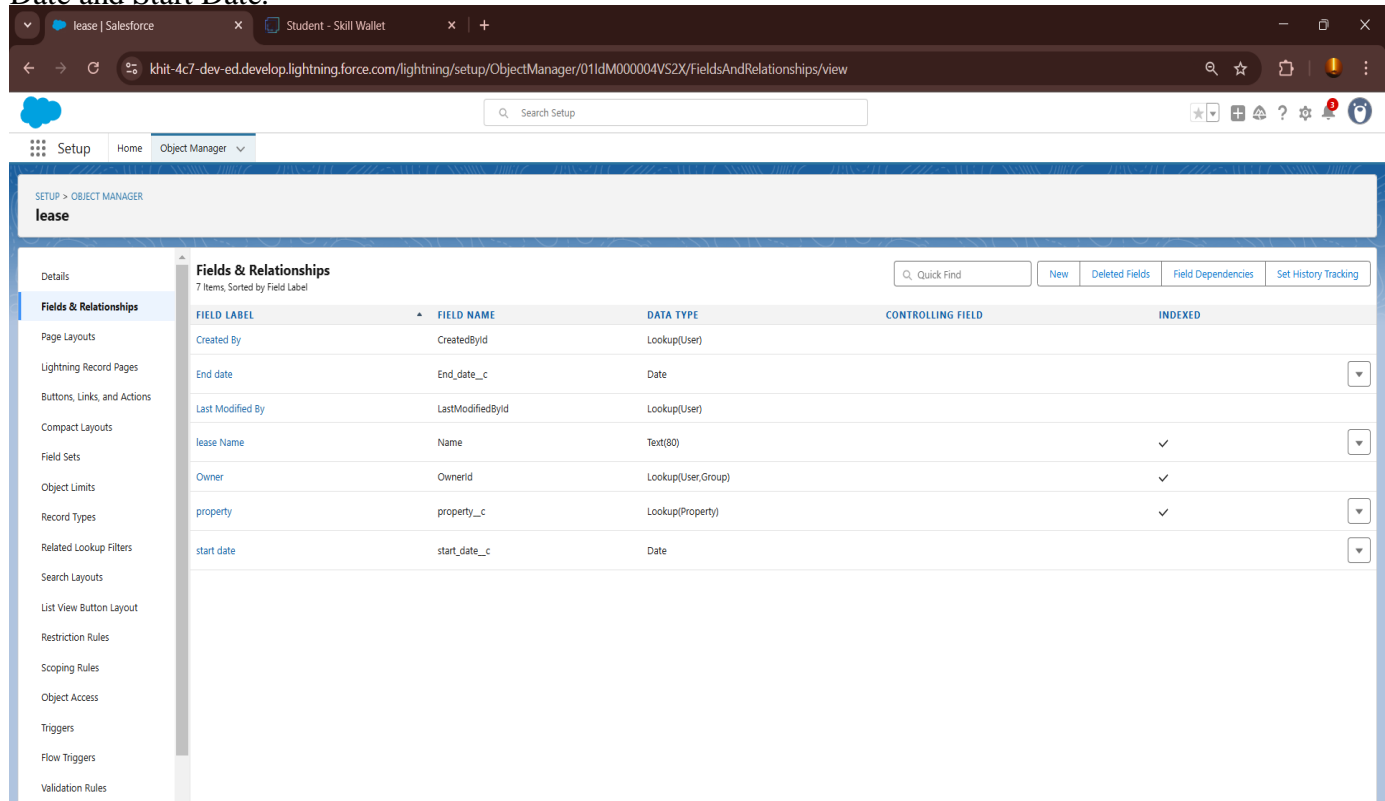
Step-by-Step Field Creation for Lease Object:

1. Go to Setup → Object Manager → Lease → Fields & Relationships → New.
2. Select Data type as a “Date” and Click on Next
3. Field Label : start date
4. Field Name : gets auto generated
5. Click on Next >> Next >> Save and new. Repeat the same steps and create another field label ‘end date’.

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.



The screenshot shows the Salesforce Setup interface. The breadcrumb trail is: SETUP > OBJECT MANAGER > lease. The left sidebar shows the navigation menu with 'Fields & Relationships' selected. The main content area displays the 'Fields & Relationships' section for the 'lease' object, showing 7 items sorted by Field Label. The table lists the following fields:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
End date	End_date_c	Date		
Last Modified By	LastModifiedById	Lookup(User)		
lease Name	Name	Text(80)		✓
Owner	OwnerId	Lookup(User,Group)		✓
property	property_c	Lookup(Property)		✓
start date	start_date_c	Date		

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
7 Items, Sorted by Field Label

Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedById	Lookup(User)		
Email	Email_c	Email		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User.Group)		✓
Phone	Phone_c	Phone		
status	status_c	Picklist		
Tenant Name	Name	Text(80)		✓

3. Payment for tenant object fields

1. Go to setup >> click on Object Manager >> type object name(Payment for tenant) in search bar >> click on the object.
2. Now click on “Fields & Relationships” >> New
3. Select Data type as a “Date” and Click on Next
4. Fill the Above as following:

- **Field Label :** Payment date, Data type : Date
- **Field Label :** Amount, Data type : Number, length : 18
- **Field Label :** check for payment, Enter values with each value separated by a new line
Enter these values: Paid, Not paid
- **Field Name :** gets auto generated
- **Click on Next >> Next >> Save.**

Payment for tenant | Salesforce x Student - Skill Wallet

khut-4c7-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01ldM000004VRpd/FieldsAndRelationships/view

Search Setup

Setup Home Object Manager

SETUP > OBJECT MANAGER

Payment for tenant

Details

Fields & Relationships
8 Items, Sorted by Field Label

Quick Find New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Amount	Amount_c	Number(18, 0)		
check for payment	check_for_payment_c	Picklist		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Payment date	Payment_date_c	Date		
Payment Name	Name	Text(80)		✓
property	property_c	Master-Detail(Property)		✓
Tenant	Tenant_c	Lookup(Tenant)		✓

4. Property Object Fields

Step-by-Step Field Creation for Property Object:

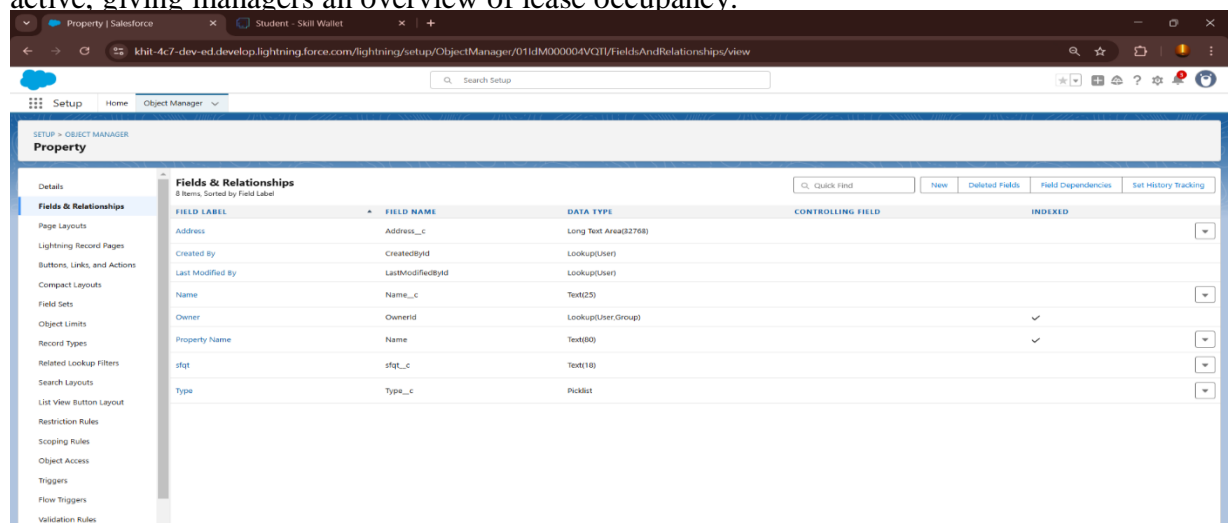
1. Go to Setup → Object Manager → Property → Fields & Relationships → New.
2. Choose Field Type (as per the below descriptions).

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.

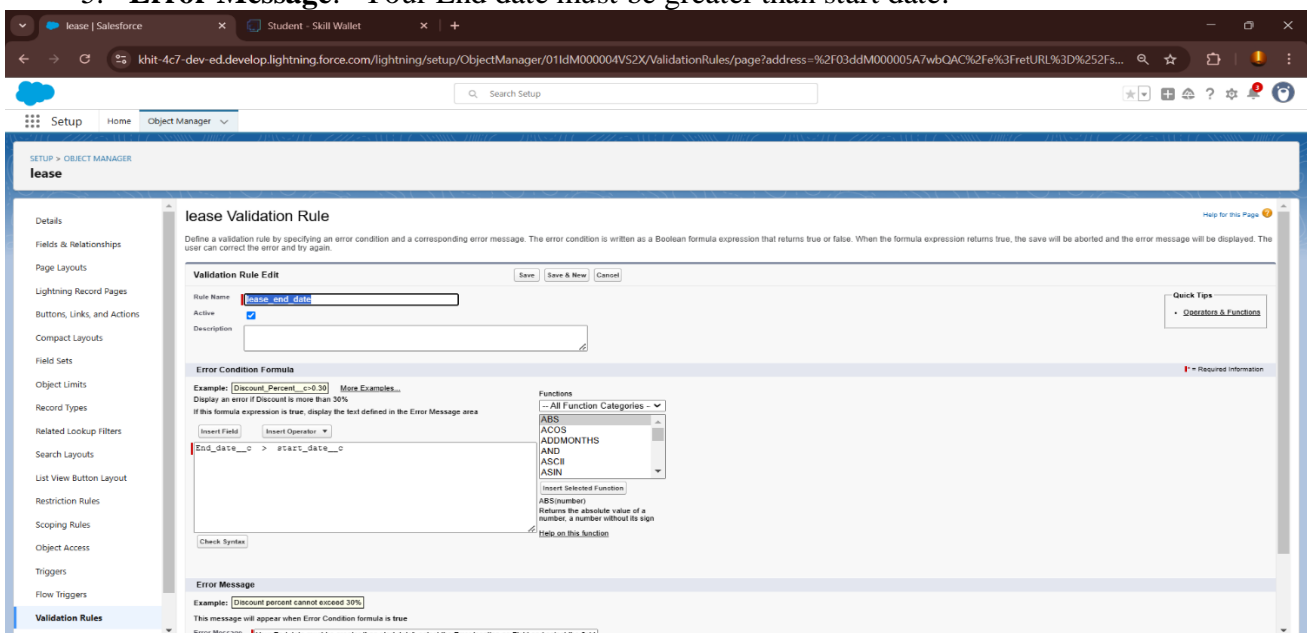


Field Validation Example

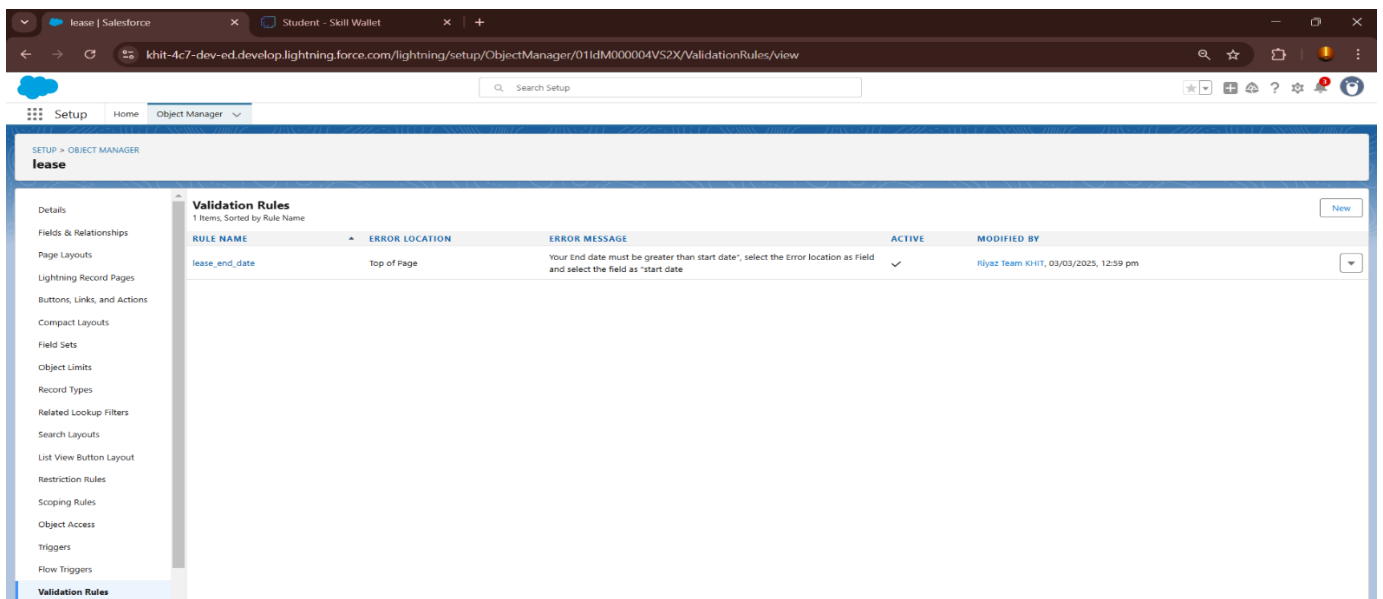
You can set **Validation Rules** to ensure data integrity. For instance:

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
3. **Formula:**
plaintext
Copy code
 $\text{End_date_c} > \text{start_date_c}$
4. **Error Message:** "Your End date must be greater than start date."



The screenshot shows the 'Lease Validation Rule' configuration page in Salesforce. The 'Rule Name' is 'lease_end_date', and it is set to 'Active'. The 'Error Condition Formula' is $\text{End_date_c} > \text{start_date_c}$. The 'Error Message' is 'Your End date must be greater than start date'. The 'Error Location' is set to 'Top of Page'.



The screenshot shows the 'Validation Rules' list for the 'Lease' object. It contains one rule named 'lease_end_date'.

RULE NAME	ERROR LOCATION	ERROR MESSAGE	ACTIVE	MODIFIED BY
lease_end_date	Top of Page	Your End date must be greater than start date", select the Error location as field and select the field as "start date"	✓	Riyaz Team KHIT, 03/03/2025, 12:59 pm

5.1 Validation Rules:

1. End Date Validation

- **Formula:**

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

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

2. Positive Monthly Rent

- **Formula:**

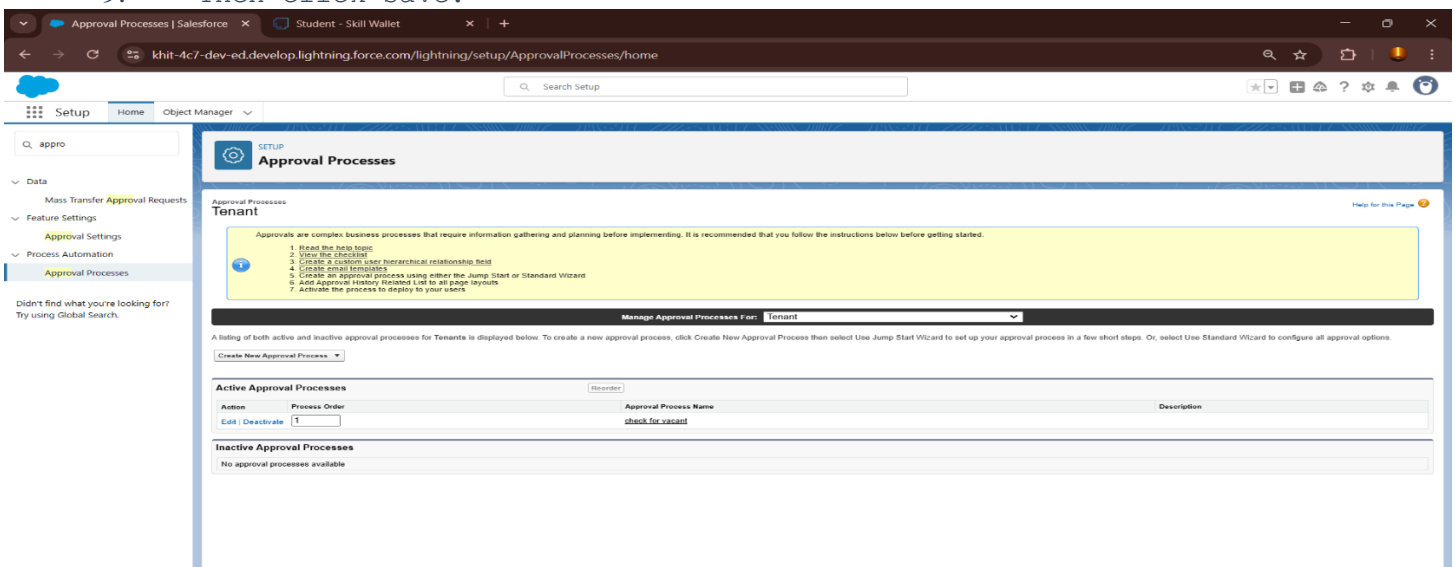
```
plaintext
Copy code
Monthly_Rent__c > 0
```

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

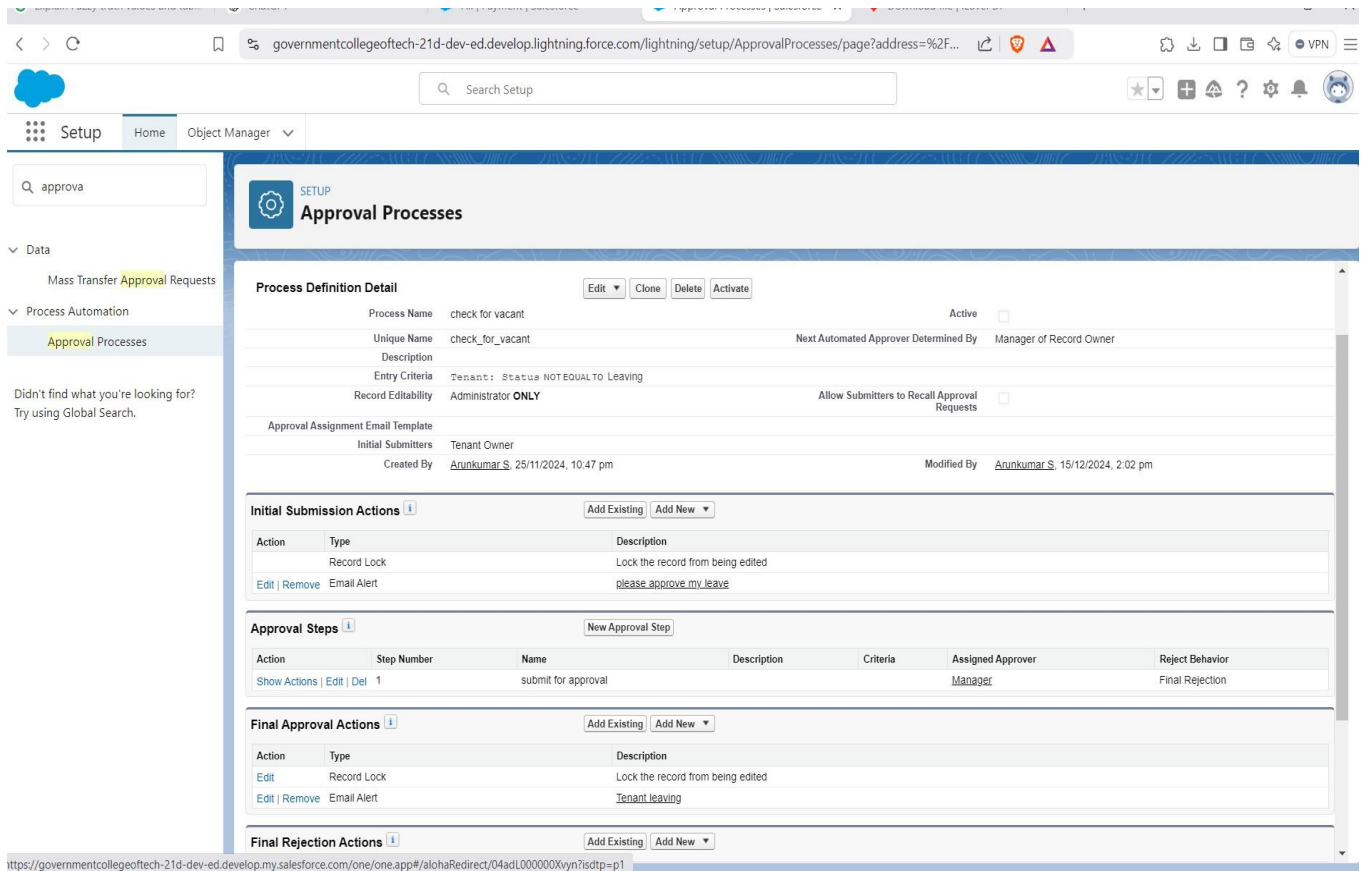
6.Approval Process

Steps to Create

1. **.Go to setup >> Approval Processes in quick find bar>>click on it.**
2. Manage Approval Process For >> "Tenant" from the drop down.
3. **Click on "Create New Approval Process" >> Use standard setup wizard.**
4. Process Name "check for vacant" >> Click Next.
5. Field "Tenant:status" >> Operator : Not equals , Value >> Click on the lookup filter icon and select "Leaving".
6. Select the "Administrators ONLY can edit records during the approval process".Then Next.
7. From the available fields select >> Tenant Name, and then add >>Add it to the selected.Then Next.
8. Submitter type Search>>Owner, Allowed Submitters>>Property Owner.Then Next.
9. Then click save.



The screenshot shows the Salesforce Setup interface for 'Approval Processes'. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled 'Approval Processes' and shows a list of active approval processes for the 'Tenant' object. The list has columns for Action, Process Order, Approval Process Name, and Description. One process is listed: 'check for vacant' with Process Order 1. There is also a section for inactive approval processes, which is currently empty.



governmentcollegeoftech-21d-dev-ed.develop.lightning.force.com/lightning/setup/ApprovalProcesses/page?address=%2F...

Search Setup

Setup Home Object Manager

Q approval

Data

Mass Transfer Approval Requests

Process Automation

Approval Processes

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

Approval Processes

Process Definition Detail Edit Clone Delete Activate

Process Name: check for vacant Active: ☐

Unique Name: check_for_vacant Next Automated Approver Determined By: Manager of Record Owner

Description:

Entry Criteria: Tenant: Status NOT EQUAL to Leaving

Record Editability: Administrator ONLY Allow Submitters to Recall Approval Requests: ☐

Approval Assignment Email Template:

Initial Submitters: Tenant Owner

Created By: Arunkumar S. 25/11/2024, 10:47 pm Modified By: Arunkumar S. 15/12/2024, 2:02 pm

Initial Submission Actions Add Existing Add New

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

Approval Steps New Approval Step

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions Edit Del	1	submit for approval			Manager	Final Rejection

Final Approval Actions Add Existing Add New

Action	Type	Description
Edit	Record Lock	Lock the record from being edited
Edit Remove	Email Alert	Tenant leaving

Final Rejection Actions Add Existing Add New

https://governmentcollegeoftech-21d-dev-ed.develop.my.salesforce.com/one/app#/alohaRedirect/04adL000000Xvyn?sdtp=p1

1. Under initial submission action click on add new and then select email alert.
2. Description: "please approve my leave".
3. unique name : auto populated
4. Email template : tenant leaving
5. Recipient type : Email field
6. Available Recipients : Email field : Email
7. From Email address : Current user's email
8. Click save.

Same as the above steps, follow the steps to create final approval action and final rejection action.

Case(i):- 1. Under Final approval action click on new and then select email alert.

2. Description: "Tenant leaving".
- Email template : Leave approved
- Recipient type : Email field
- Available Recipients : Email field : Email
- From Email address : Current user's email
- Click save

Case(ii):- Under final rejection action

Description: "your request for leave is rejected".


Email template : leave rejected and Recipient type : Email field

Available Recipients : Email field : Email and Click save.

Email Templates:-

1)Email Alert for tenant leaving

- Go to setup in quick find box enter email template >> click on classic Email Template.
- Click on >> New Email Template====>Choose text and Click on available for use.
- Email Template Name is “tenant leaving” and Subject : ” request for approve the leave”
- Email body :
Dear {!Tenant__c.CreatedBy},
Please approve my leave
- Save


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	<input checked="" type="checkbox"/>
Email Template Name	tenant leaving	Last Used Date	
Template Unique Name	tenant_leaving	Times Used	
Encoding	Unicode (UTF-8)		
Author	Arunkumar S [Change]		
Description			
Created By	Arunkumar S , 25/11/2024, 10:30 pm	Modified By	Arunkumar S , 25/11/2024, 10:30 pm

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

2) Create Email Template For Leave Approved

- Create and Follow the same first two steps in tenant leaving template
- Email Template Name is “Leave approved” and Subject : ” Leave approved”
- Email body :
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

- Click Save.



SETUP

Classic Email Templates

Text Email Template

Leave approved

Help for this Page

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	Leave approved	Last Used Date	
Template Unique Name	Leave_approved	Times Used	
Encoding	Unicode (UTF-8)		
Author	Arunkumar.S (Change)		
Description			
Created By	Arunkumar.S, 25/11/2024, 10:32 pm	Modified By	Arunkumar.S, 25/11/2024, 10:32 pm

Edit Delete Clone

Email Template

Send Test and Verify Merge Fields

Subject | Leave approved

Plain Text Preview

dear {!Tenant__c.Name},

I hope this message finds you well. I am writing to inform you that I have received your email confirming the approval of my leave request. I would like to express my gratitude for considering and approving my time off.

your leave is approved. You can leave now.

3)Create Email Template For rejection for leave

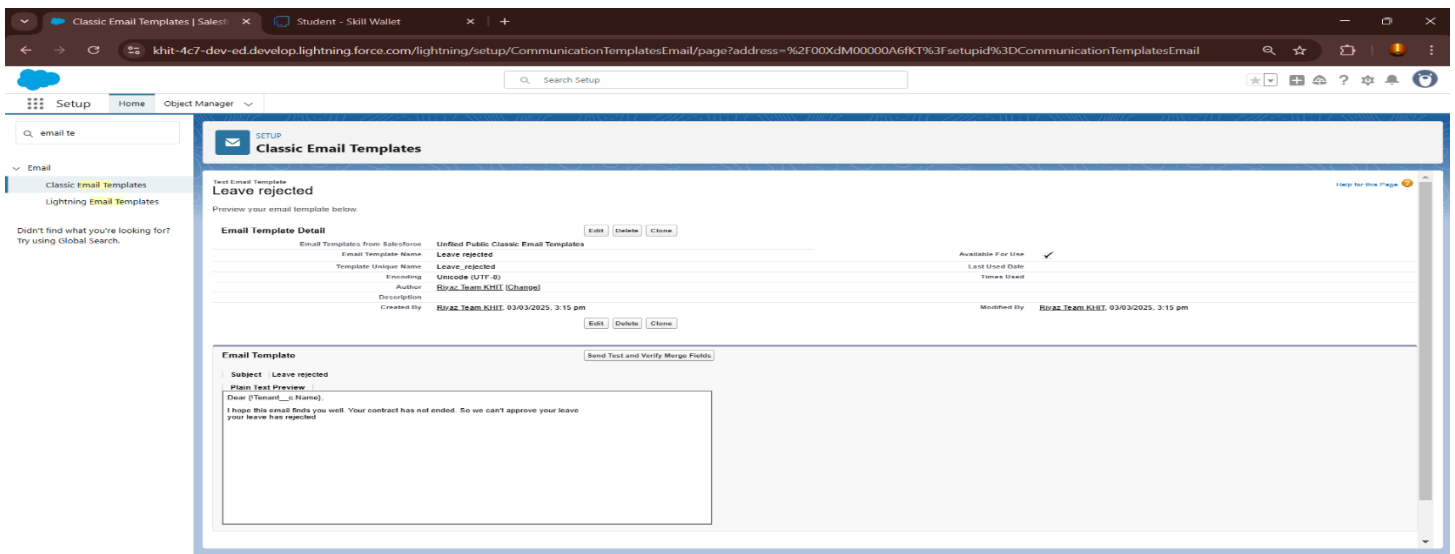
- Create and Follow the same first two steps in tenant leaving template
- Email Template Name is “Leave rejected” and Subject : ” Leave rejected”
- Email body :

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

- Click Save.



The screenshot shows the Salesforce Classic Email Templates setup page for a template named "Leave rejected". The page is viewed in a web browser with the URL `khiti-4c7-dev-ed.develop.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=%2F00XdM00000A6IKT%3Fsetupid%3DCommunicationTemplatesEmail`. The page layout includes a top navigation bar with "Setup", "Home", and "Object Manager" tabs. The main content area is titled "Classic Email Templates" and shows the details for the "Leave rejected" template. The details include the template name, unique name, encoding, author, and creation/modification dates. The "Plain Text Preview" section shows the email body content, which is a rejection of a leave request. The page also includes a "Send Test and Verify Merge Fields" button and a "Help for this Page" link.

4) Create Email Template For Monthly payment

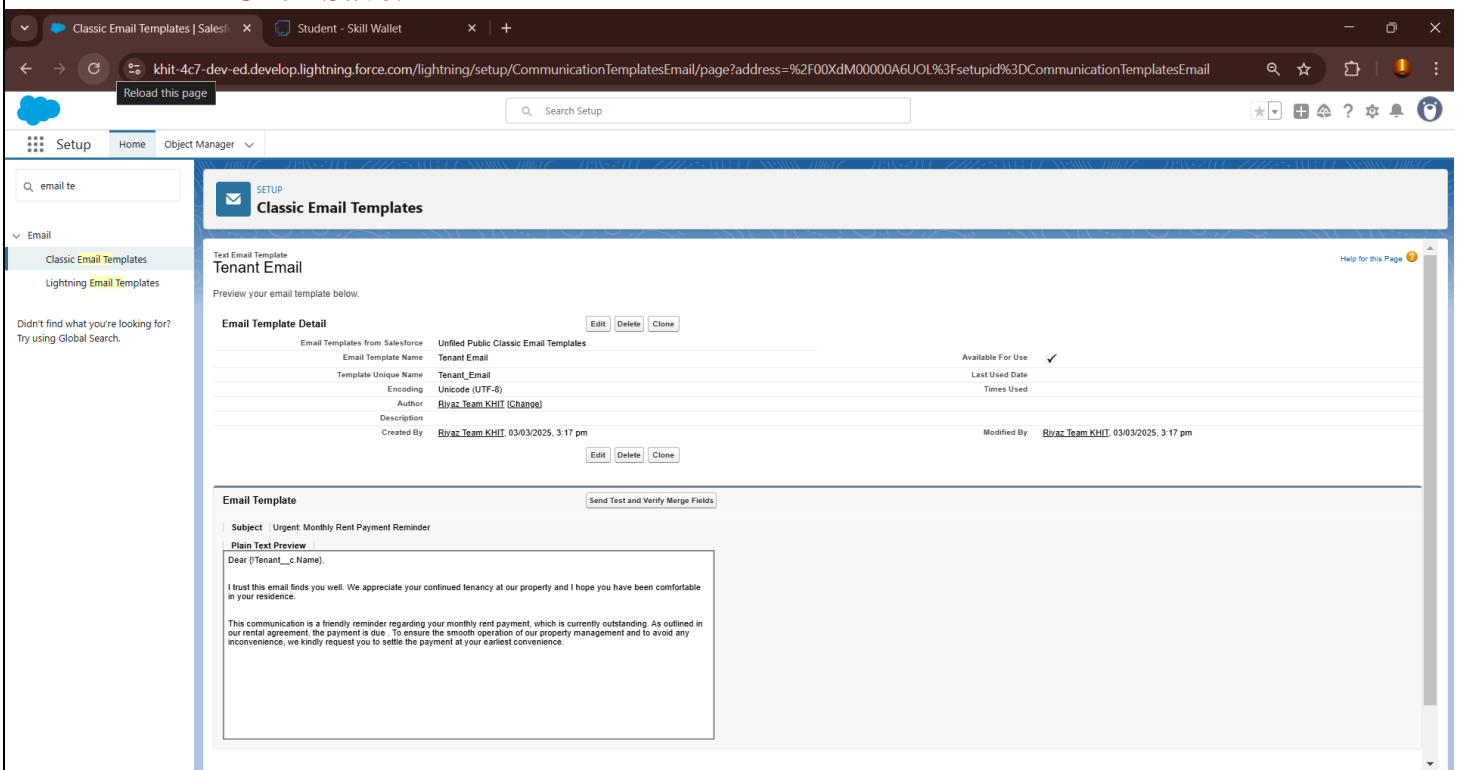
- Create and Follow the same first two steps in tenant leaving template
- Email Template Name is “Tenant Email” and
- Subject : ” Urgent: Monthly Rent Payment Reminder”
- Email body :

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.

This communication is a friendly reminder regarding your monthly rent payment, which is currently outstanding. As outlined in our rental agreement, the payment is due . To ensure the smooth operation of our property management and to avoid any inconvenience, we kindly request you to settle the payment at your earliest convenience .

- Click Save.



The screenshot shows the Salesforce Classic Email Templates setup page for a template named "Tenant Email". The page is divided into two main sections: "Email Template Detail" and "Email Template".

Email Template Detail:

- Email Template Name:** Tenant Email
- Template Unique Name:** Tenant_Email
- Encoding:** UTF-8
- Author:** Riyaz Team KHIT (Change)
- Description:**
- Created By:** Riyaz Team KHIT, 03/03/2025, 3:17 pm
- Modified By:** Riyaz Team KHIT, 03/03/2025, 3:17 pm
- Available For Use:** ☒
- Last Used Date:**
- Times Used:**

Email Template:

Subject: Urgent: Monthly Rent Payment Reminder

Plain Text Preview:

Dear {!Tenant__c.Name},

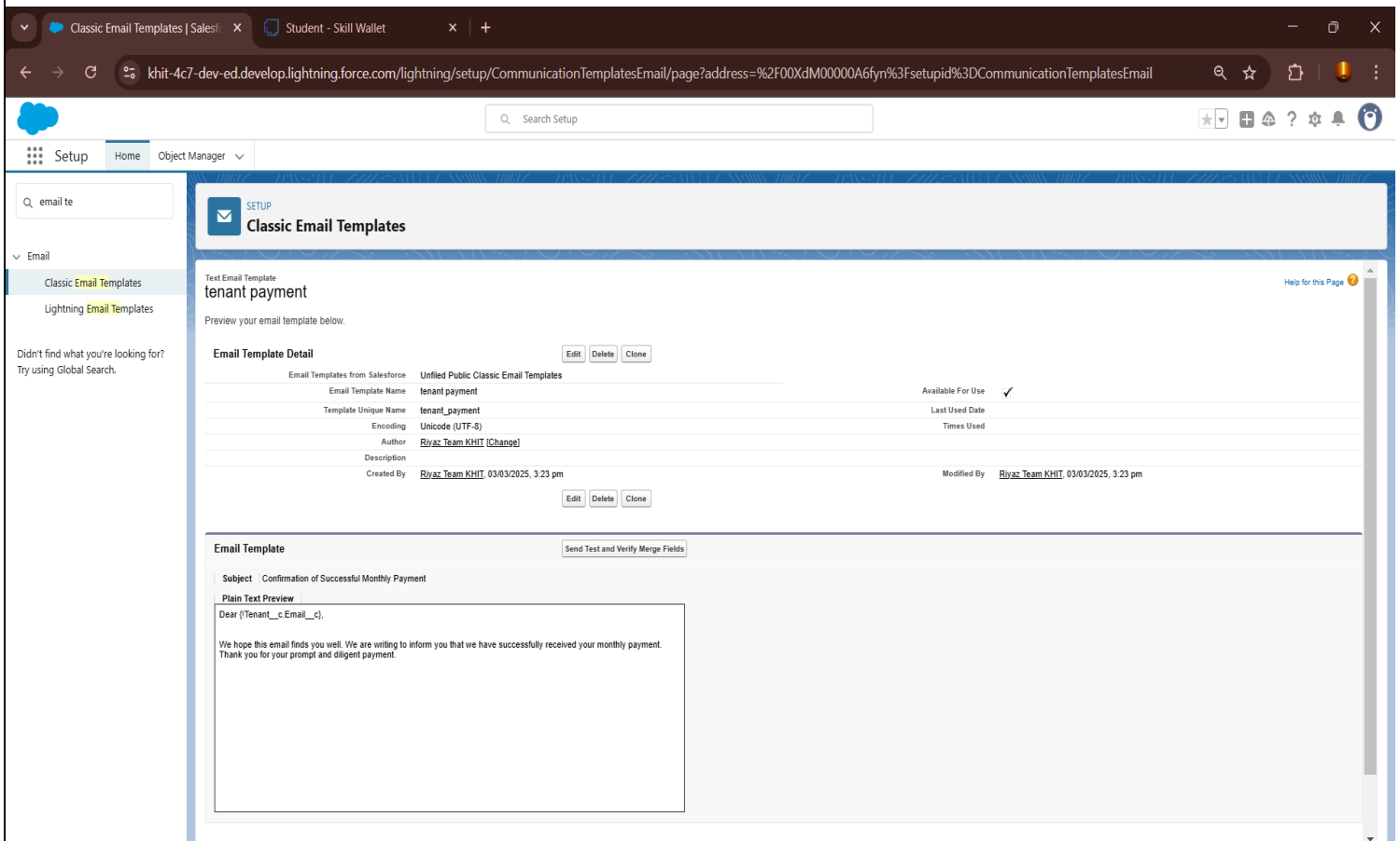
I trust this email finds you well. We appreciate your continued tenancy at our property and I hope you have been comfortable in your residence.

This communication is a friendly reminder regarding your monthly rent payment, which is currently outstanding. As outlined in our rental agreement, the payment is due . To ensure the smooth operation of our property management and to avoid any inconvenience, we kindly request you to settle the payment at your earliest convenience .

5) Create Email Template For successful payment

- Create and Follow the same first two steps in tenant leaving template
- Email Template Name is “tenant payment”
- Subject : ” Confirmation of Successful Monthly Payment”
- Email body :
- 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.



The screenshot shows the Salesforce Classic Email Templates interface. The browser address bar displays the URL: `khit-4c7-dev-ed.develop.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=%2F00XdM00000A6fyn%3Fsetupid%3DCommunicationTemplatesEmail`. The page title is "Classic Email Templates". The left sidebar shows the navigation menu with "Setup" selected. The main content area displays the details for a "tenant payment" email template. The "Email Template Detail" section shows the following information:

Email Template Detail	
Email Templates from Salesforce	Unified Public Classic Email Templates
Email Template Name	tenant_payment
Template Unique Name	tenant_payment
Encoding	Unicode (UTF-8)
Author	Riyaz Team KHIT (Change)
Description	
Created By	Riyaz Team KHIT, 03/03/2025, 3:23 pm
Modified By	Riyaz Team KHIT, 03/03/2025, 3:23 pm

The "Email Template" section shows the subject line: "Confirmation of Successful Monthly Payment". The "Plain Text Preview" shows the following text:

```
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.
```

We use email templates to increase productivity and ensure consistent messaging. Email templates with merge fields let you quickly send emails that include field data from Salesforce records like contacts, leads, or opportunities. You can use email templates when emailing groups of people—with list email or mass email—or just one person.

Salesforce email templates are the easiest way to get your emails done. They help you create and send quick emails that include merge fields from Salesforce records like Contacts, Leads, Opportunities, or Custom Objects.

When you have a large number of contacts or leads in Salesforce, it can be difficult to keep track of who needs to be notified about new information. Salesforce email templates allow you to combine all these contacts or leads into one email and then send it out simultaneously.

7. Flows

1. Scheduled Flows

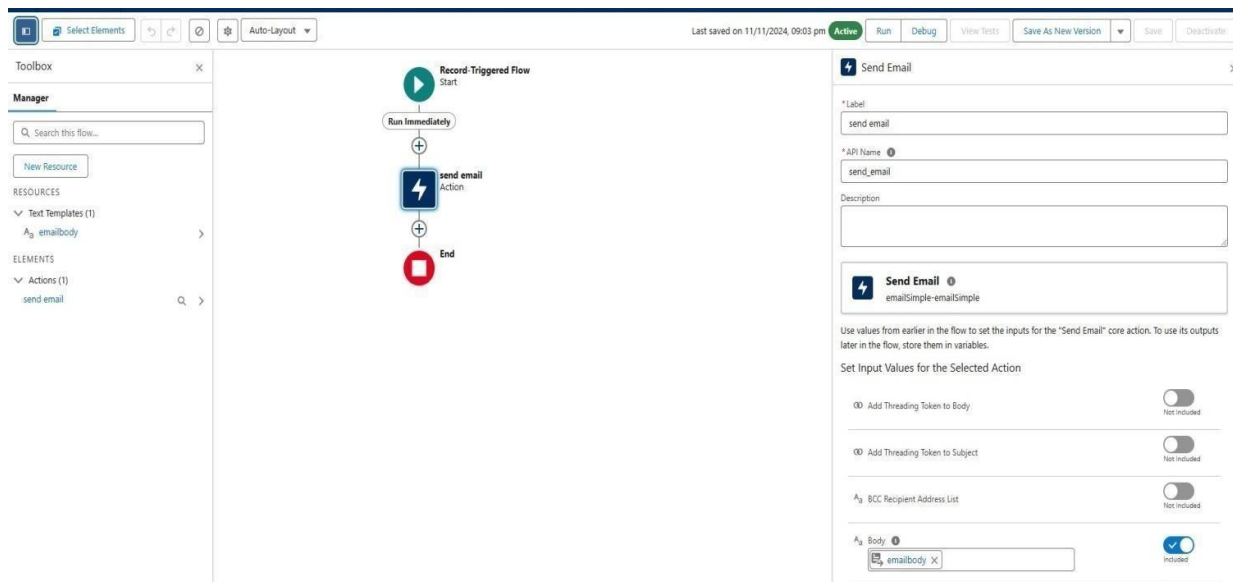
- **Purpose:** Notify tenants about lease expiration and automate renewal reminders.
- **Steps:**
 - Create a Flow with the trigger set to run daily.
 - Query leases expiring in the next 30 days.
 - Send an email notification using dynamic templates.

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."



8. 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.

Apex Trigger named as:-

1. Give the Apex Trigger name as “test”, and select “Tenant__c” from the dropdown for sObject.

Trigger Code:

```
trigger test on Tenant__c (before insert)

{

    if(trigger.isInsert && trigger.isBefore){

        TestHandler.preventInsert(trigger.new);

    }

}
```

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
```

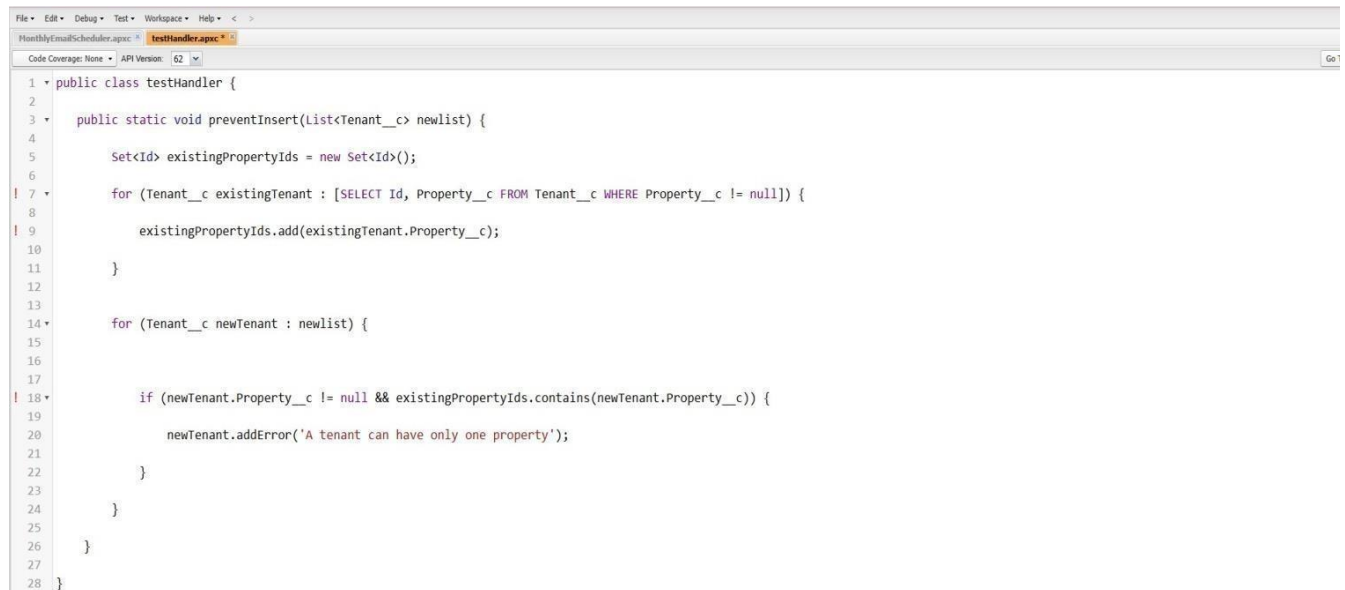
```
    System.assert(e.getMessage().contains('A lease already exists for this tenant
and property.');
```

```
    }
```

```
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;
    }
}
```



SETUP

Apex Classes

Apex Class

MonthlyEmailScheduler

Apex Class Detail

Edit

Delete

Download

Security

Show Dependencies

Name	MonthlyEmailScheduler	Status	Active
Namespace Prefix		Code Coverage	0% (0/15)
Created By	Arunkumar.S	Last Modified By	Arunkumar.S
	25/11/2024, 11:38 pm		25/11/2024, 11:38 pm

Class Body

Class Summary

Version Settings

Trace Flags

```

1 global class MonthlyEmailScheduler implements Schedulable {
2
3   global void execute(SchedulableContext sc) {
4     Integer currentDay = Date.today().day();
5     if (currentDay == 1) {
6       sendMonthlyEmails();
7     }
8   }
9 }
10
11
12
13
14
15 public static void sendMonthlyEmails() {
16
17
18   List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
19
20
21
22
23   for (Tenant__c tenant : tenants) {
24     String recipientEmail = tenant.Email__c;
25     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
26
27
28

```

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.

Developer Console - Google Chrome

khiti-4c7-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage

File

Edit

Debug

Test

Workspace

Help

TestHandler.apex

test.apex

Code Coverage: None

API Version: 63

Go To

```

1 public class TestHandler {
2
3   public static void preventInsert(List<Tenant__c> newlist) {
4
5     Set<Id> existingPropertyIds = new Set<Id>();
6
7     for (Tenant__c existingTenant : [SELECT Id, Property__c FROM Tenant__c WHERE Property__c != null]) {
8
9       existingPropertyIds.add(existingTenant.Property__c);
10
11     }
12
13     for (Tenant__c newTenant : newlist) {
14
15
16
17
18       if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
19
20         newTenant.addError('A tenant can have only one property');
21
22       }
23     }
24   }
25 }

```

Logs

Tests

Checkpoints

Query Editor

View State

Progress

Problems

User	Application	Operation	Time	Status	Read	Size
------	-------------	-----------	------	--------	------	------

Filter

Click here to filter the log list

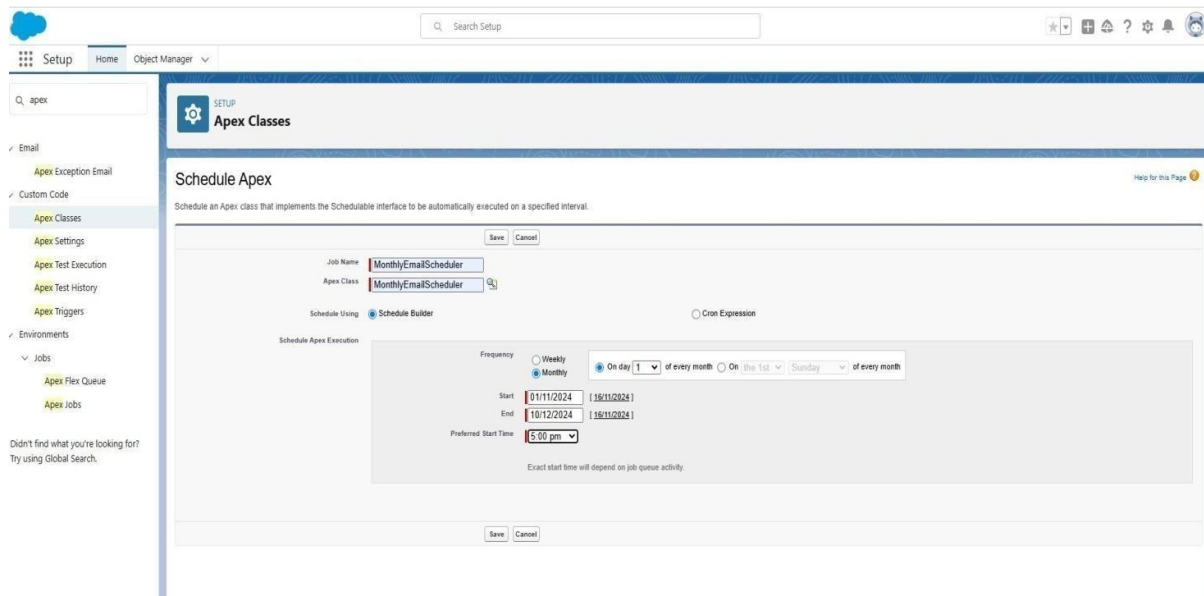
Apex logic:

```
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');
            }
        }
    }
}
```

10.Schedule APEX class:



The screenshot shows the Salesforce Setup interface for scheduling an Apex class. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled "Schedule Apex" and includes a "Schedule Apex Execution" section. The "Job Name" is set to "MonthlyEmailScheduler" and the "Apex Class" is also "MonthlyEmailScheduler". The "Schedule Using" option is set to "Schedule Builder". The "Frequency" is set to "Monthly" with a dropdown for "On day 1 of every month". The "Start" date is "01/11/2024" and the "End" date is "10/12/2024". The "Preferred Start Time" is "5:00 pm". The "Save" and "Cancel" buttons are visible at the bottom of the form.

- Testing the approval

Lease Management Payment ▾ Tenants ▾ lease ▾ property ▾

Tenant
Arunkumar S

New Contact Edit New Opportunity ▾

Related **Details**

Tenant Name Arunkumar S	Owner Arunkumar S
Email lucifer69203@gmail.com	
Phone 06379912451	
Status Leaving	
Created By Arunkumar S, 25/11/2024, 11:45 pm	Last Modified By Arunkumar S, 25/11/2024, 11:54 pm

Activity

Filters: All time • All activities • All types

Refresh • Expand All • View All

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.

7. 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.

8. 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.