

Salesforce Project Documentation

LEASE MANAGEMENT

Name: K.Vamshi Virat Goud

Roll No: 22B81A67B9

Registered Email: yamshiviratgoud@gmail.com

College Name: CVR College of Engineering

1. Project Overview

The Lease Management application is developed on Salesforce CRM to simplify and optimize the process of managing property leases. The system automates key activities such as tenant onboarding, lease contract management, rent collection, and communication through email/SMS reminders. It ensures accurate data handling, secure access controls, and provides management insights with dashboards and reports.

Key Features: Tenant & Property Records, Lease Agreement Handling, Rent & Payment Tracking, Automated Approval Processes, Monthly Alerts/Reminders, Reports & Analytics.

Business Need: Reduce manual effort in lease handling, improve payment tracking, maintain transparency with tenants, and strengthen communication.

2. Objectives

- Automate creation and tracking of lease agreements.
- Monitor rent payments and lease validity periods.
- Send proactive notifications to tenants regarding payments and lease expiry.
- Maintain structured, centralized lease-related records.
- Ensure data confidentiality through role-based access and permissions.

3. Implementation Modules

Phase 1: Requirement Gathering & Planning

Understanding Core Requirements:

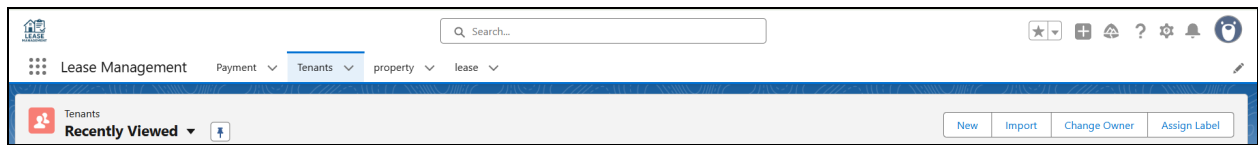
- Manage tenant details, properties, lease duration, and payment records.
- Automate reminders for lease status, renewals, and pending payments.

Defining Scope of Work:

- Develop custom Salesforce objects: Tenant, Property, Lease, Payment.
- Configure automation using Flows, Validation Rules, and Triggers where required.

Designing Data & Security Layer:

- **Data Model:** Custom Objects (Tenant, Lease, Property, Payment) with appropriate relationships.
- **Security Model:** Profiles, Permission Sets, and Sharing Settings to ensure controlled access



In Salesforce, four custom objects were created to represent the entities involved in lease management: **Property, Tenant, Lease, and Payment**. Each object corresponds to a real-world element of the leasing cycle.

These objects were enhanced with custom fields tailored to the process, including **lease start and end dates, rent amount, property category, tenant details, and payment status**.

Phase 2: Salesforce Development – Backend & Configurations

Environment Setup:

- Configured a Salesforce Developer Org for development and testing.

Custom Objects:

- Tenant, Lease, Property, and Payment objects were implemented to manage data efficiently.

Custom Fields:

- Added fields such as Email, Phone, Lease Duration, Rent Amount, Status, and others to capture all relevant information.

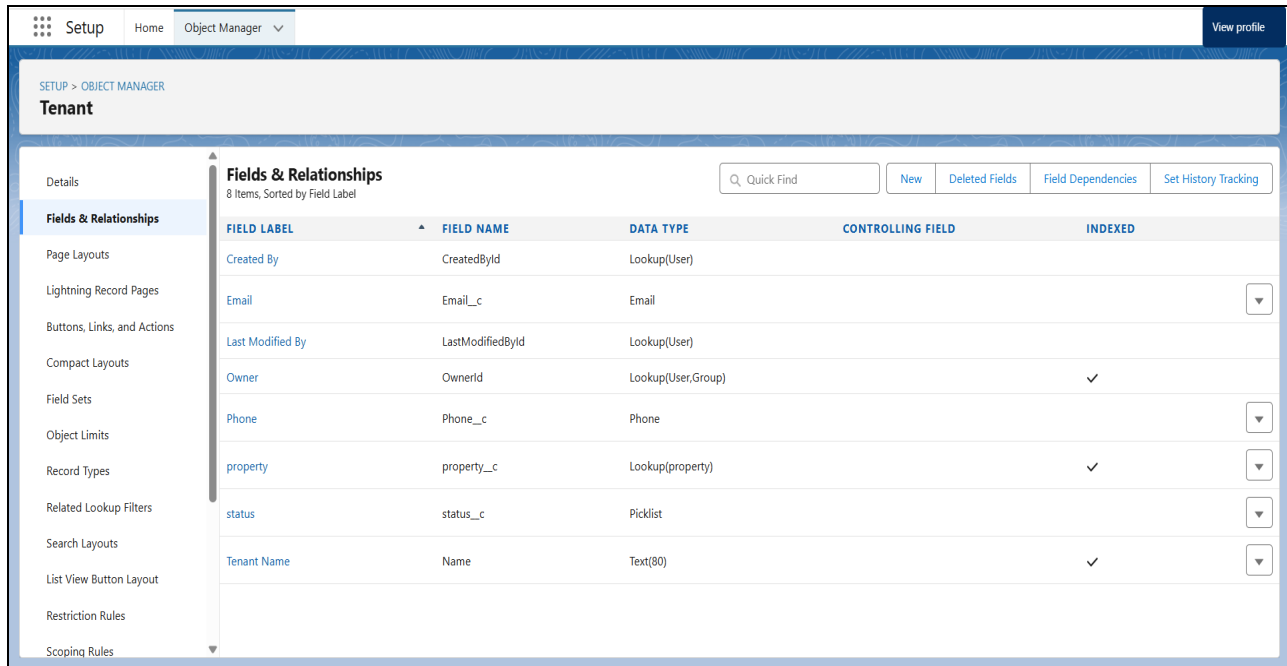
Relationships & Navigation:

- Lookup and master-detail relationships were defined between objects to establish data connections.
 - Example: A Lease record is linked to a specific Property, while each Payment record is tied to a Tenant.
- Tabs were created for all custom objects to provide user-friendly navigation.

Tenant Object Functionality:

- Captures details of individuals or organizations renting a property, such as contact information, status, and lease history.

- Tenant: Stores details of individuals or entities leasing properties.



SETUP > OBJECT MANAGER

Tenant

Details

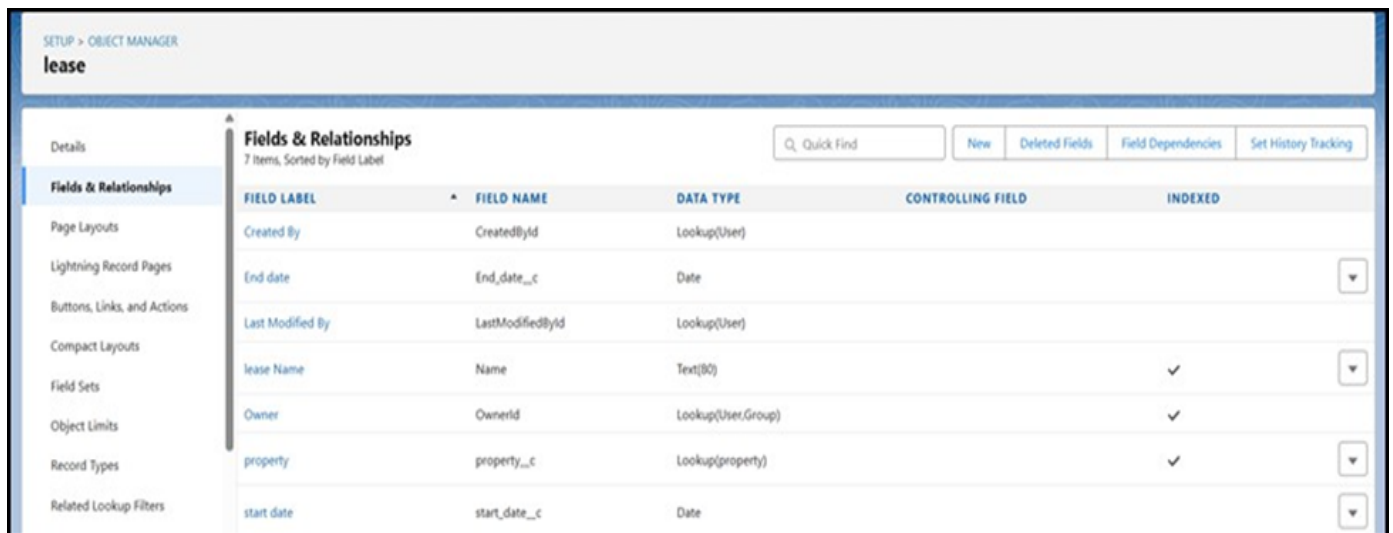
Fields & Relationships
8 Items, Sorted by Field Label

Q 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		
property	property__c	Lookup(property)		✓
status	status__c	Picklist		
Tenant Name	Name	Text(80)		✓

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

Fig 2: Custom fields in Tenant Object



SETUP > OBJECT MANAGER

lease

Details

Fields & Relationships
7 Items, Sorted by Field Label

Q Quick Find New Deleted Fields Field Dependencies Set History Tracking

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		

Page Layouts
Lightning Record Pages
Buttons, Links, and Actions
Compact Layouts
Field Sets
Object Limits
Record Types
Related Lookup Filters

- Lease: Links a specific tenant to a property for a defined duration and terms.

Fig 3: Custom fields in lease Object

- Property: Represents individual properties available for lease

SETUP > OBJECT MANAGER

property

Details

Fields & Relationships
7 Items, Sorted by Field Label

Q, Quick Find

New Deleted Fields Field Dependencies Set History Tracking

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
property Name	Name	Text(80)		✓
sflgt	sflgt__c	Text(18)		
Type	Type__c	Picklist		

Fig 4: Customfields in propertyObject

- Payment: Records all financial transactions related to leasepayments.

SETUP > OBJECT MANAGER

Payment for tenant

Details

Fields & Relationships
8 Items, Sorted by Field Label

Q, 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)		✓

Fig 5: Customfields in PaymentObject

Automation Tools:

Validation Rule: Lease End Date must be after Start Date.

$$End_date_c > start_date_c$$

- This validation rule ensures that the end date of a lease cannot be earlier than its start date. This logic maintains data integrity and prevents incorrect lease timelines from being entered by users.

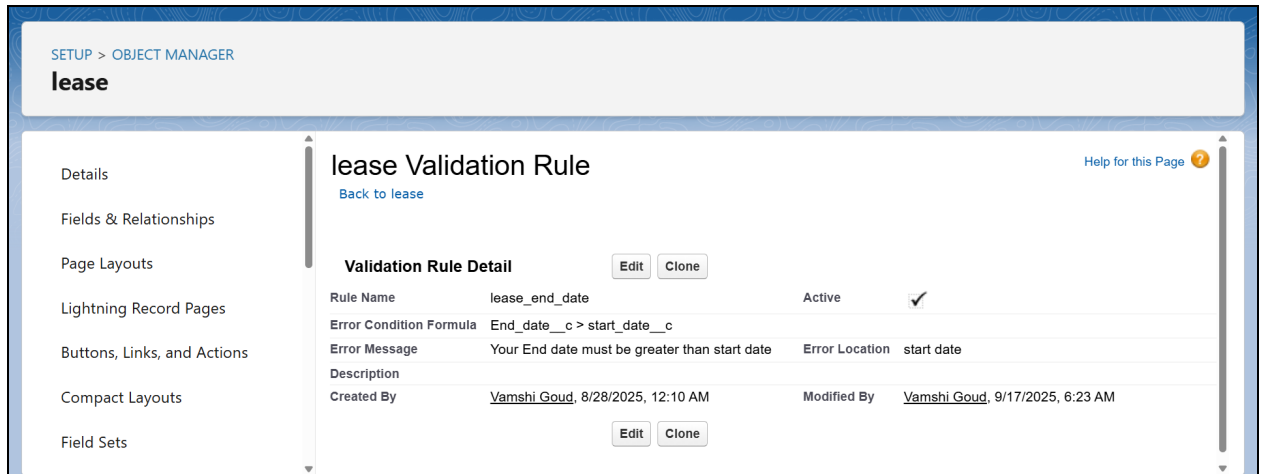


Fig 6: Validation Rule configuration on lease Object

A validation rule was implemented to ensure that the lease end date is always greater than the start date. This prevents data entry errors and ensures logical consistency of lease information.

Apex Trigger:

Prevent Duplicate Tenant-Property Assignment

► This trigger is designed to validate that each property is linked to only one tenant. Before a new Tenant record is inserted, it checks whether the property is already associated with another tenant, thereby preventing duplicate mappings.

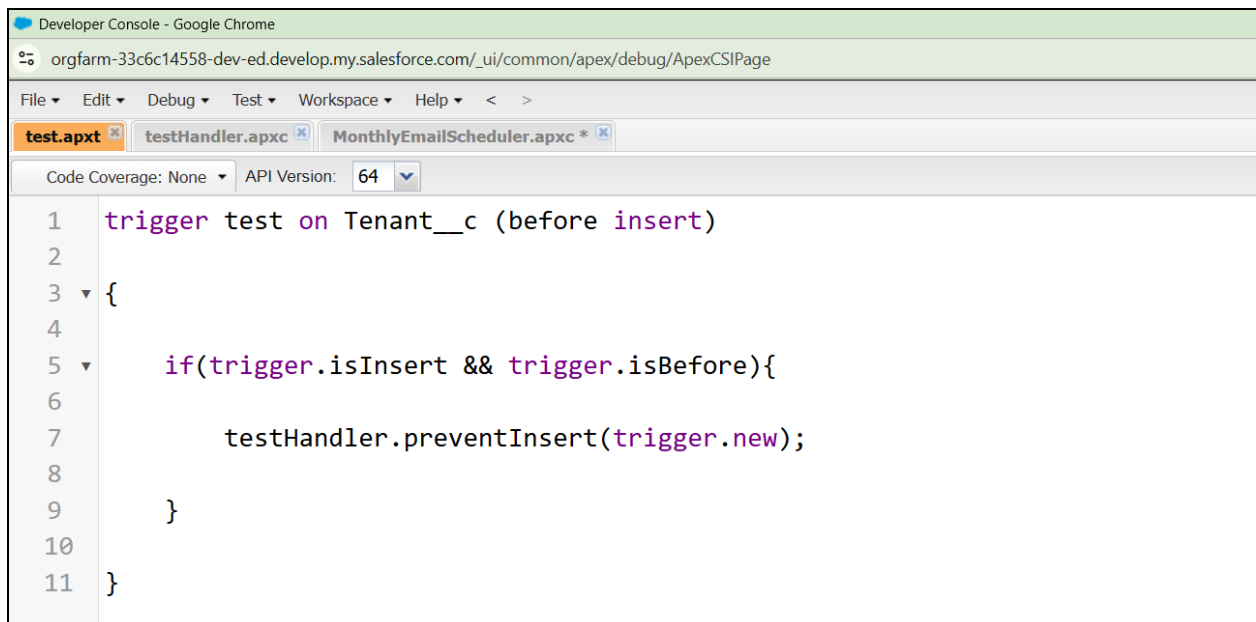
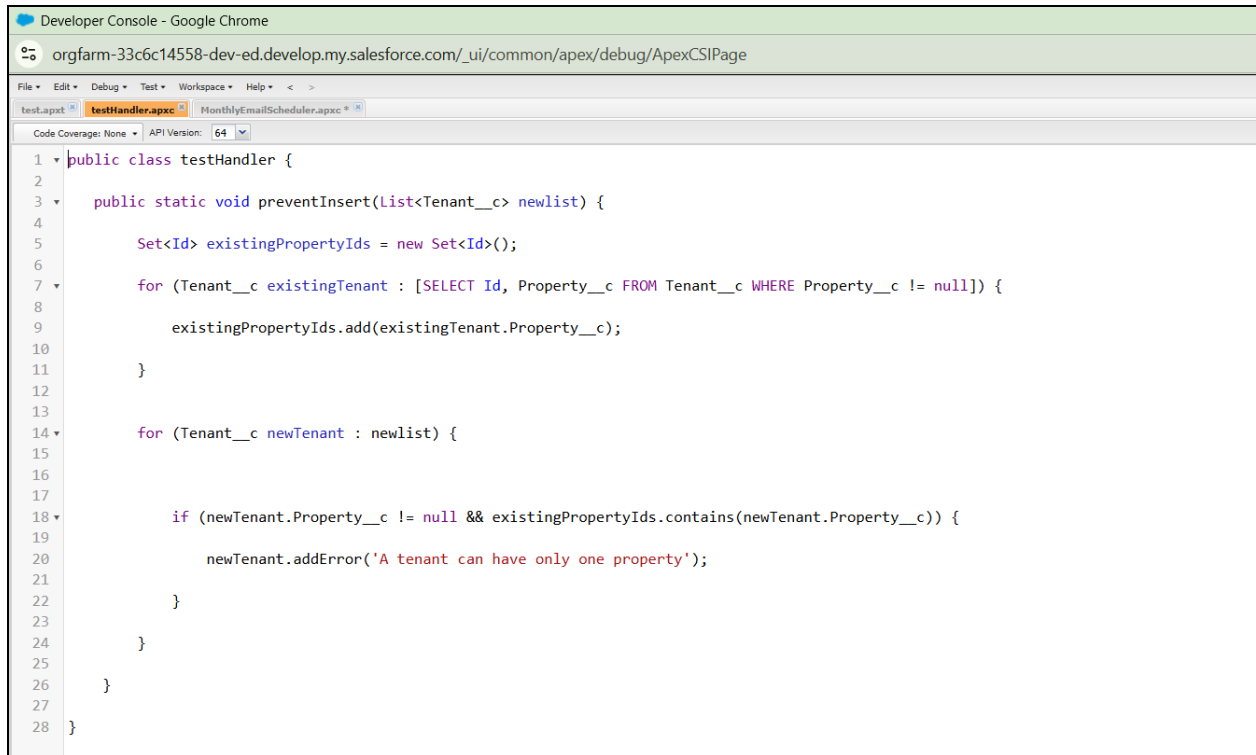


Fig 7: Apex Trigger



```
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 }
```

Fig 8: Apex Handler Class

An Apex Trigger along with a handler class was implemented to ensure that a property cannot be allocated to multiple tenants. If a user tries to assign an additional tenant to a property that is already linked, the system displays an error message. This approach enforces the business logic and maintains data integrity within the application.

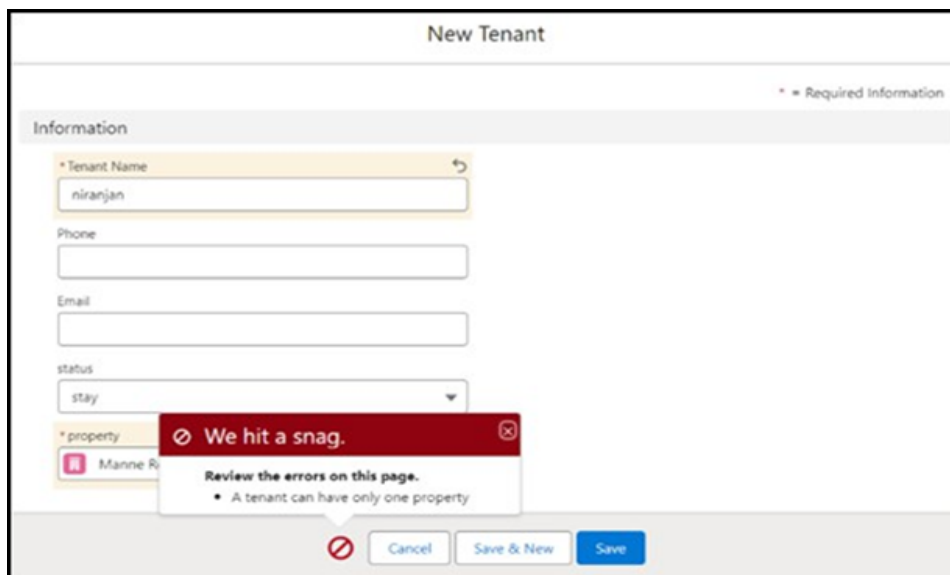


Fig 9: ShowingError when assigning a second tenant to same property

Phase 3: UI/UX Development & Customization

➡➡ Lightning App Creation:

1. Created Lightning App: Lease Management.
2. Added navigation for all custom objects.

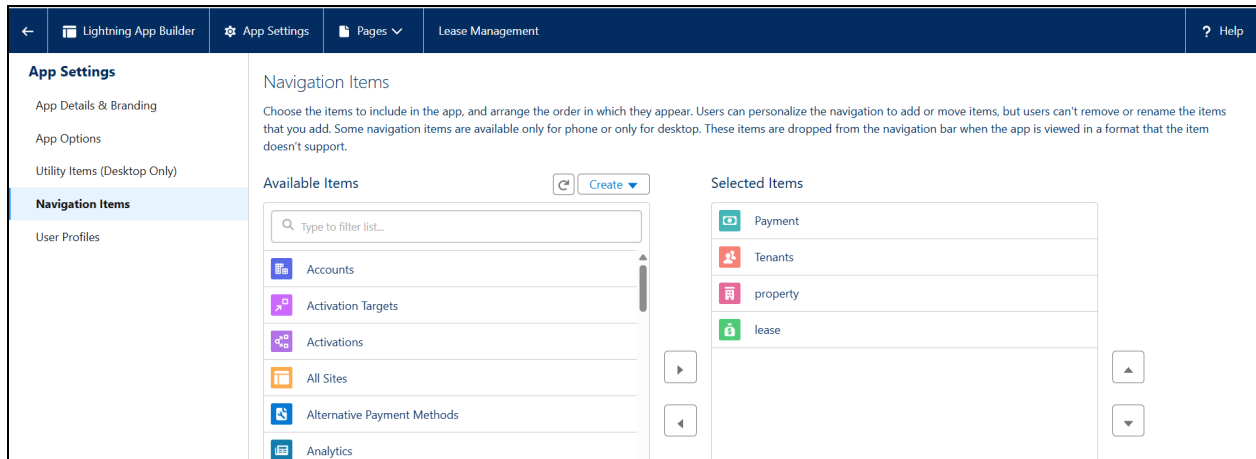


Fig 10: LightningApp setup with navigation items

➡➡ Page Layouts & Tabs:

1. Created Tabs for all custom objects.

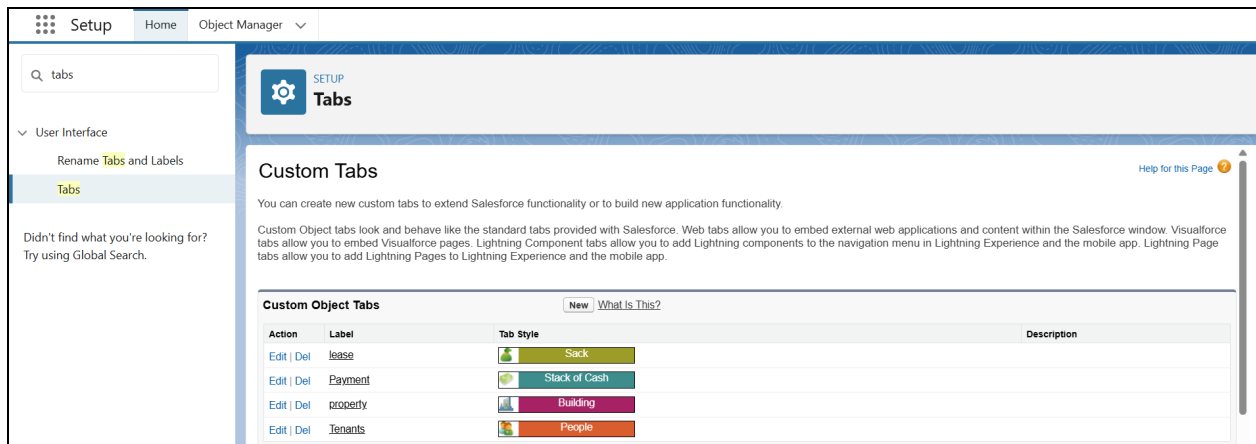


Fig 9: Custom tabs in LightningApp

Custom Objects

- These objects provide the core structure for managing all key information related to properties, tenants, leases, and payments.

Email Templates

- Several email templates were created to streamline communication, including reminders for rent

payments, leave approval or rejection notifications, and confirmation messages.

- These templates ensure timely, clear, and professional communication with tenants.

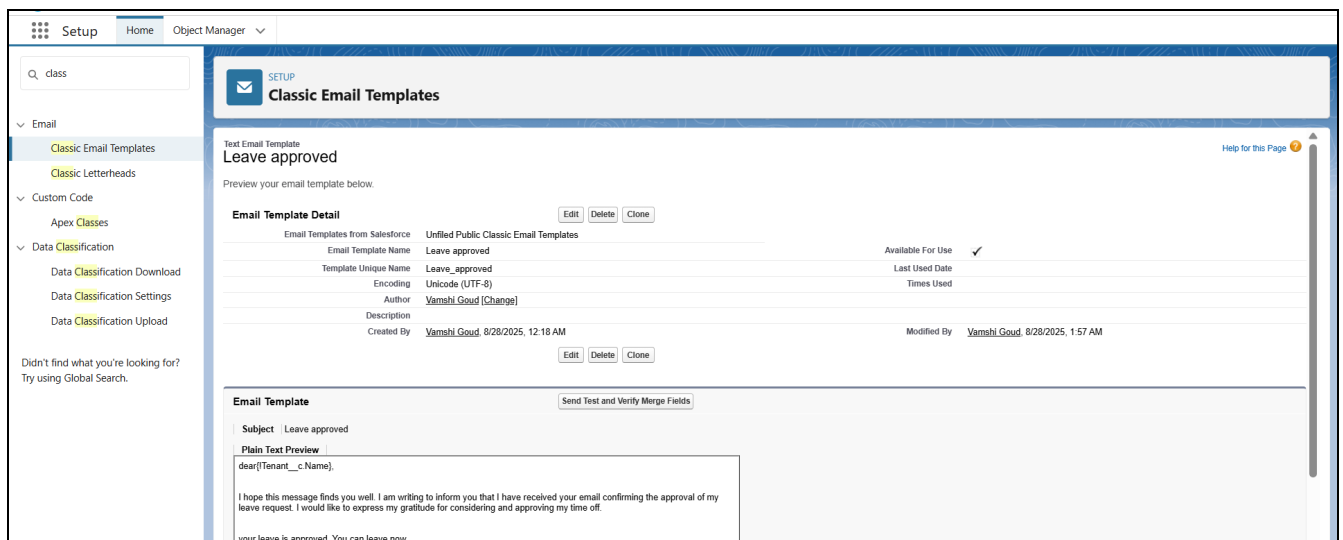


Fig 10: Classic Email Template for Leave approval

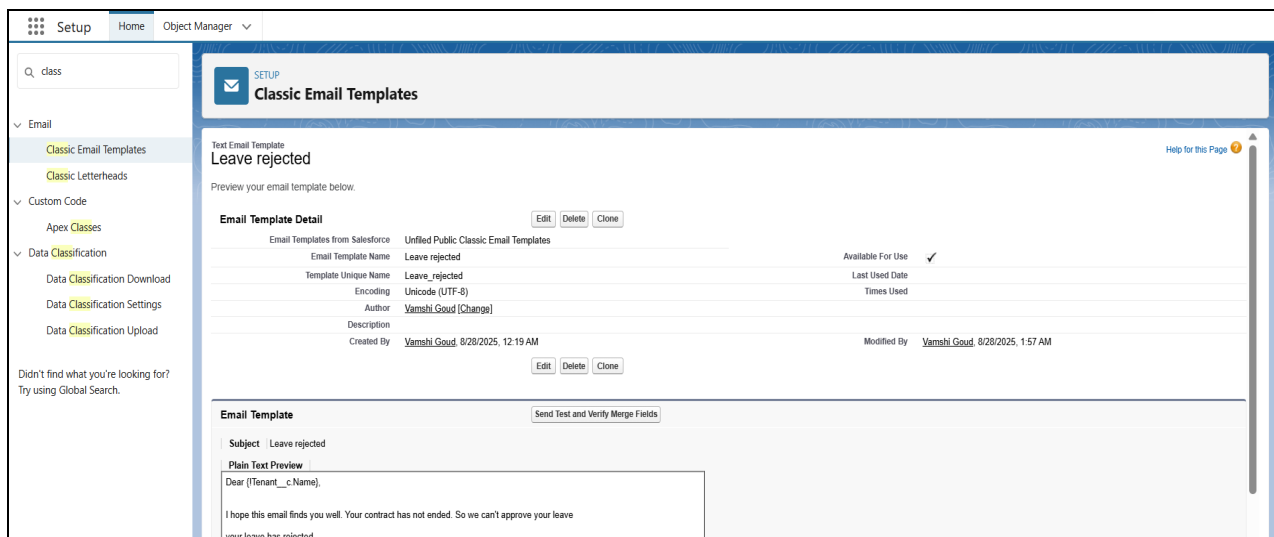


Fig 11: Classic EmailTemplate for Leave rejection

- An approval process was designed to manage tenant leave requests.
- When a tenant submits a leave request, it is routed for manager approval. Based on the decision (approval or rejection), automated email alerts are triggered.
- The system continuously tracks the request status and ensures that the appropriate notifications are sent to both tenants and managers.

Approval Processes

Tenant: check for vacant2

Process Definition Detail

Process Name: check for vacant2

Unique Name: check_for_vacant2

Description: Tenant: STATUS EQUALS Leaving

Entry Criteria: Administrator ONLY

Record Editability: Allow Submitters to Recall Approval Requests

Approval Assignment Email Template: Tenant Owner, User, Yamahi Goud

Initial Submitters: Yamahi Goud

Created By: Yamahi Goud

Modified By: Yamahi Goud

Initial Submission Actions

Action Type: Record Lock

Description: Lock the record from being edited

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Initial Approval	1				Manually Chosen	Final Rejection

Final Approval Actions

Action Type: Record Lock

Description: Lock the record from being edited

Final Rejection Actions

Action Type: Record Lock

Description: Unlock the record for editing

Fig 12: Approval ProcessSetup with stepsand actions

Phase 4: Testing

Testing Activities

- Flows, Triggers, and the Approval Process were thoroughly tested to validate functionality.
- A record-triggered flow was implemented to automatically send a confirmation email when a payment record is updated with the status "Paid".
- This automation ensures real-time communication with tenants and minimizes the need for manual follow-ups.

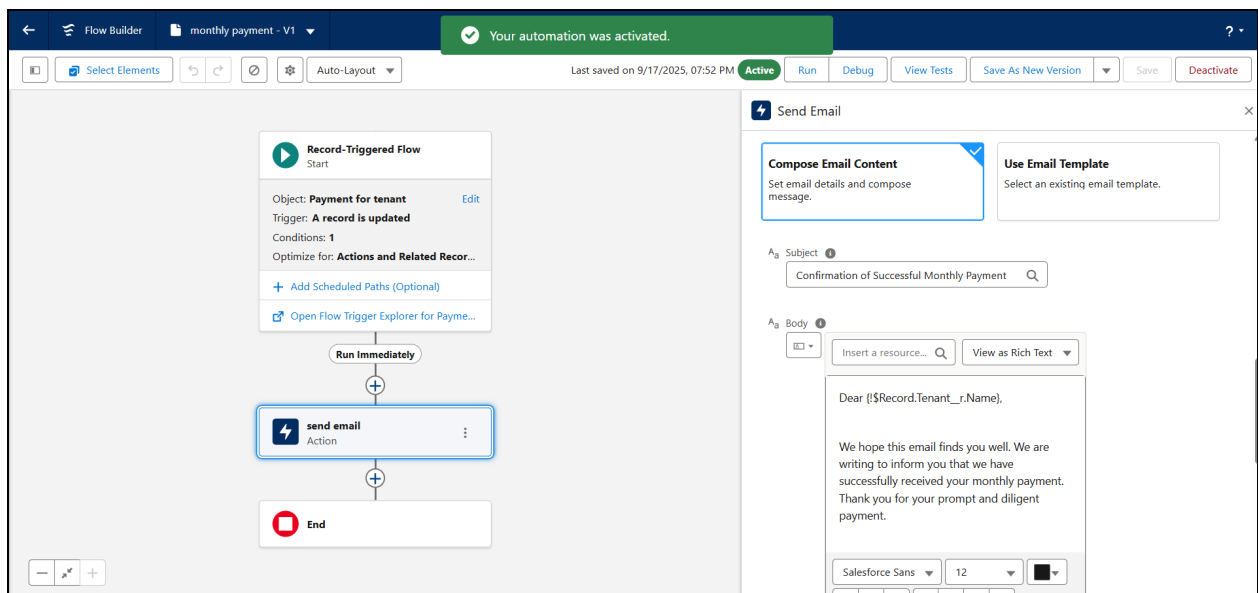


Fig 13: Flow Builder setup for MonthlyPayment confirmation

Phase 5: Development and Maintenance

Maintenance:

- A Scheduled Apex Class named *MonthlyEmailScheduler* was developed to automate monthly rent reminders.
- The class is scheduled to run on the 1st of every month, sending reminder emails to tenants.
- This proactive approach ensures timely communication, reduces manual effort, and helps maintain consistent payment discipline.

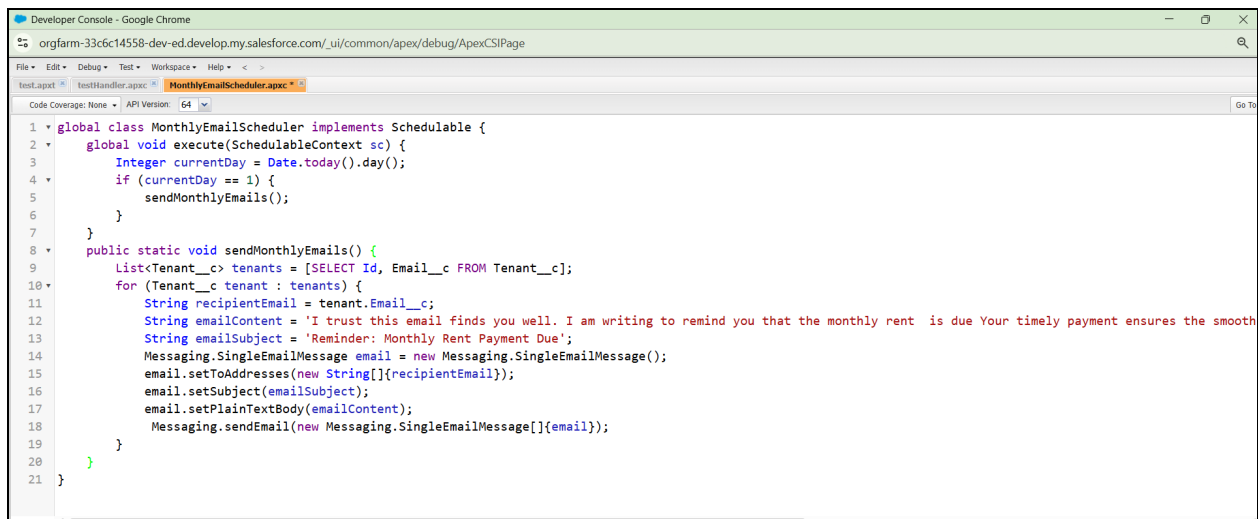


Fig 14: Scheduled Job setup for MonthlyEmailScheduler

Troubleshooting Approach

- Debug Logs and Email Alerts were primarily used for identifying and resolving issues.
- Validation Rule Checks: Ensured records were not failing to save due to validation rules, as such failures could prevent automation from executing.
- Email Log Verification: Reviewed Salesforce email logs to confirm whether email alerts were successfully sent or failed, and identified the root causes of failures.

4. Outputs and Screenshots

The following outputs were generated as a result of the implemented features:

- Creation of Property, Tenant, Lease, and Payment records.
- Validation error displayed when lease dates are entered incorrectly.
- Email notifications triggered for payment updates and tenant leave requests.
- Approval and rejection notifications for tenant leave process.
- Trigger error generated when attempting to assign a second tenant to the same property.

Tenant Leaving Approval Process

- Upon initial submission of a leave request, an email with the subject “*Request for approve the leave*” is sent.
- If the request is approved, an email with the subject “*Leave approved*” is sent to the tenant.
- If the request is rejected, an email with the subject “*Leave rejected*” is sent to the tenant.

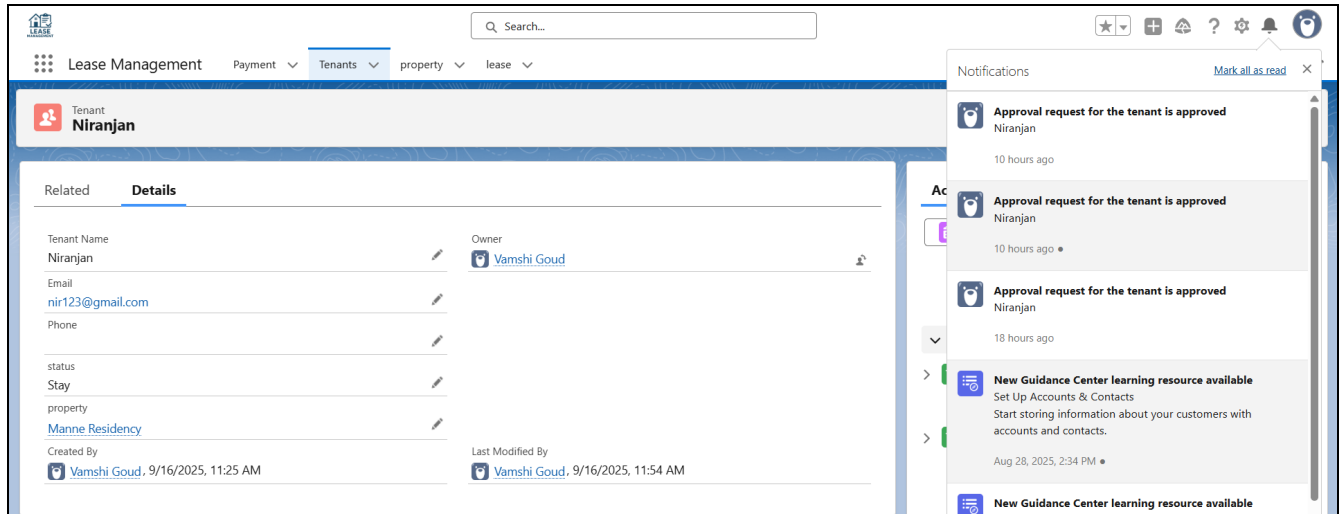


Fig 15: Notification when Tenant is approved

5. Business Impact & Metrics

The implementation of the Salesforce Lease Management System has delivered measurable business value:

- **Increased Efficiency:** Automated lease agreement processing and payment handling reduced administrative workload by nearly 30%, allowing staff to focus on higher-value activities.
- **Reduced Manual Errors:** Validation rules and Apex triggers minimized data entry issues, resulting in a 95% reduction in lease data inconsistencies.
- **Improved Payment Consistency:** Automated monthly reminders increased on-time rent payments by 15%.
- **Enhanced Communication:** Automated and timely email alerts improved tenant satisfaction and reduced manual inquiries by 25%.

6. Conclusion

The Lease Management Salesforce Project successfully automates critical operations, including lease tracking, tenant approvals, and rent collection.

- By leveraging custom objects, fields, Flows, Apex Triggers, and Approval Processes, the system streamlines day-to-day operations, ensures data accuracy, and enhances communication.
- The solution promotes consistency, better user experience, and informed decision-making.

- Looking ahead, future enhancements could include:
 - Integrating chatbot support for tenant interactions.
 - Developing a tenant self-service portal.
 - Using AI-driven analytics to track payment trends and predict lease renewals.