Salesforce Project Documentation

LEASE MANAGEMENT

College Name : KG College of Arts and Science
TEAM ID : NM2025TMID24063

- TEAM LEADER : CHARUKESH P

  EMAIL : 2322ka09@kgcas.com

- TEAM MEMBER : DARSHANA K

  EMAIL : 2322ka10@kgcas.com

- TEAMMEMBER : DHAKSHATHA TS

  EMAIL:2322ka11@kgcas.com

- TEAMMEMBER : DHARANI D

  EMAIL:2322ka12@kgcas.com

1. Project Overview
The Lease Management project is built using Salesforce CRM to manage the leasing of properties efficiently. It includes a system that automates various lease-related activities such as tenant registration, lease agreement, payment handling, and communication via email alerts. The system ensures proper record-keeping, provides security controls, and offers insightful analytics via dashboards. Key Features: Property & Tenant management, Payment tracking, Approval workflows, Monthly email reminders, Reports & Dashboards. Business Need: Streamline lease management, avoid manual errors, ensure payment consistency, enhance communication.

2. Objectives

- Automate lease agreement processing.
- Track tenant payments and lease durations.
- Notify tenants with payment and leave alerts.
- Maintain a structured record of lease data.
- Provide user access control and maintain data security.

3. Implementation Modules
Phase 1: Requirement Analysis & Planning

Understanding Business Requirements:

- Requirement to manage tenants, properties, lease duration, and payments.
- Notify tenants about lease status and due payments.

Defining Project Scope:

- Build objects: Tenant, Property, Lease, and Payment.
- Set up automated flows and triggers.

Design Data & Security Model:

- Data Model: Custom Objects (Tenant, Lease, Property, Payment).
- Security Model: Profiles, Roles, Sharing Rules.

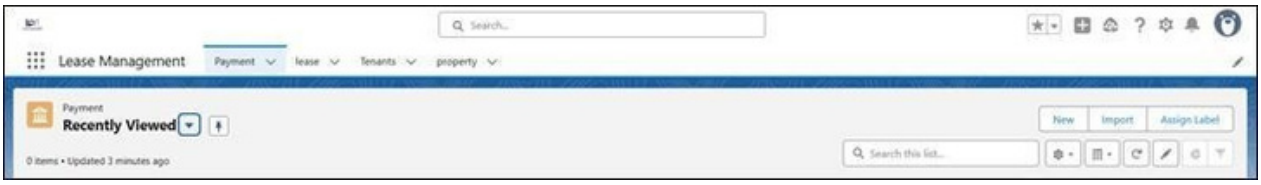


Fig ١ : Custom objects

Created four custom objects in Salesforce : Property ، Tenant ، Lease ، and Payment . Each object represents real-world entities involved in property leasing .

These objects were customized with fields specific to the lease process ، such as lease start /end dates ، payment amount ، property type ، and tenant status .

Phase٢ : SalesforceDevelopment – Backend&Configurations

- Setup Environment :
  Developer Org setup
- Custom Objects :
  Tenant ، Lease ، Property ، Payment
- Custom Fields :
  Email ، Phone ، Status ، Lease dates ، Amount ، etc .

- Established lookup and master-detail relationships between objects .
  For example ، each lease is linked to a property ، and each payment is linked to a tenant . Tabs were also created for each custom object ، making it easier for users to navigate through records .

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

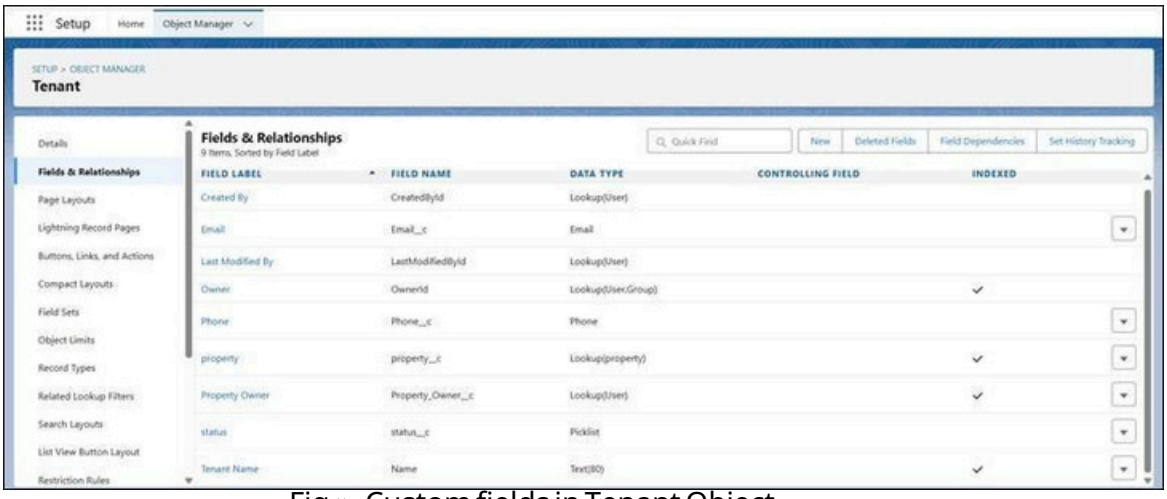
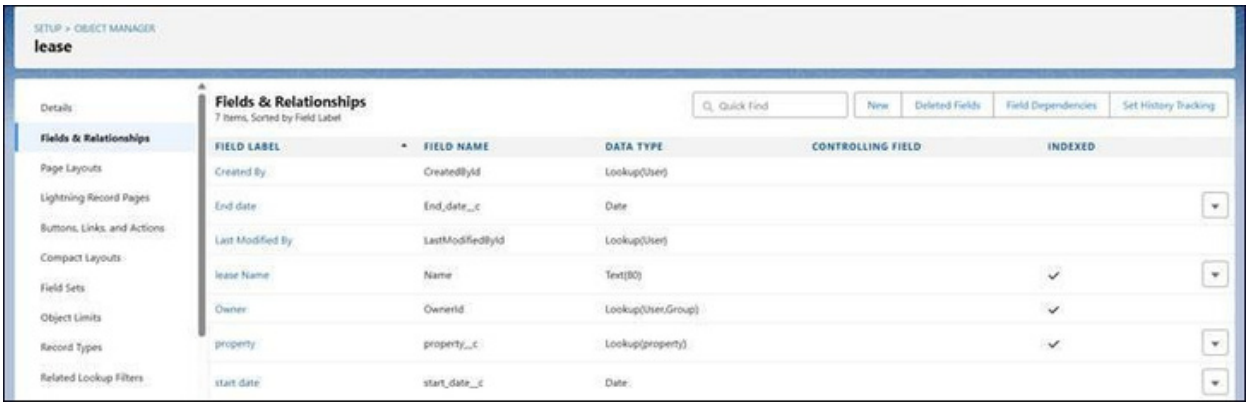


Fig ٢ : Custom fields in Tenant Object



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

Fig ٣: Custom fields in lease Object

Property: Represents individual properties available for lease.

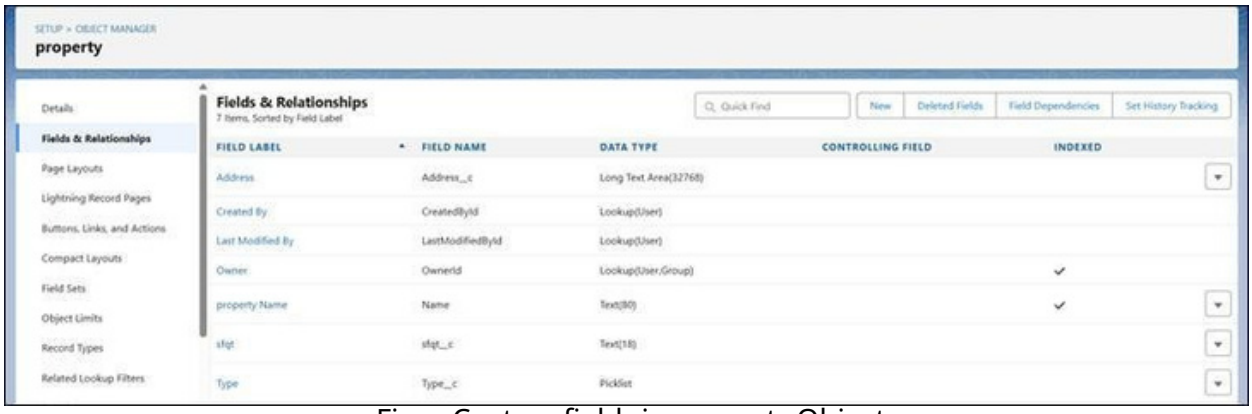


Fig ٤: Custom fields in property Object

- Payment: Records all financial transactions related to lease payments.

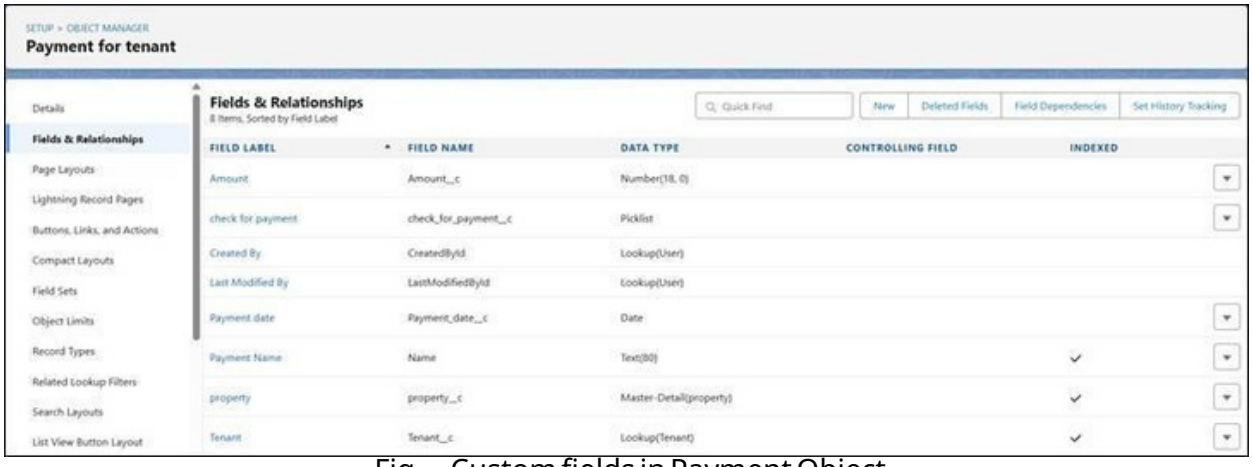


Fig ٥: Custom fields in Payment Object

- Automation Tools:

  Validation Rule: Lease End Datemust be after StartDate.

  End__date c < start__date c

  - This validation rule ensures thatthe end date ofalease cannot be earlier than its start date. This logic maintains dataintegrity and prevents incorrect lease timelines from being entered by users.

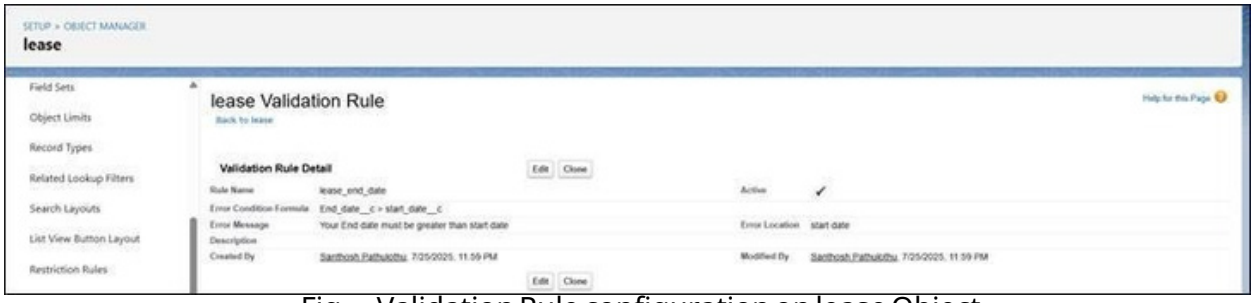


Fig ٦: Validation Rule configuration on lease Object

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 mapping.

> This Apex Trigger ensures that a property cannot be assigned to more than one tenant by validating the uniqueness of the property assignment before inserting a Tenant record.

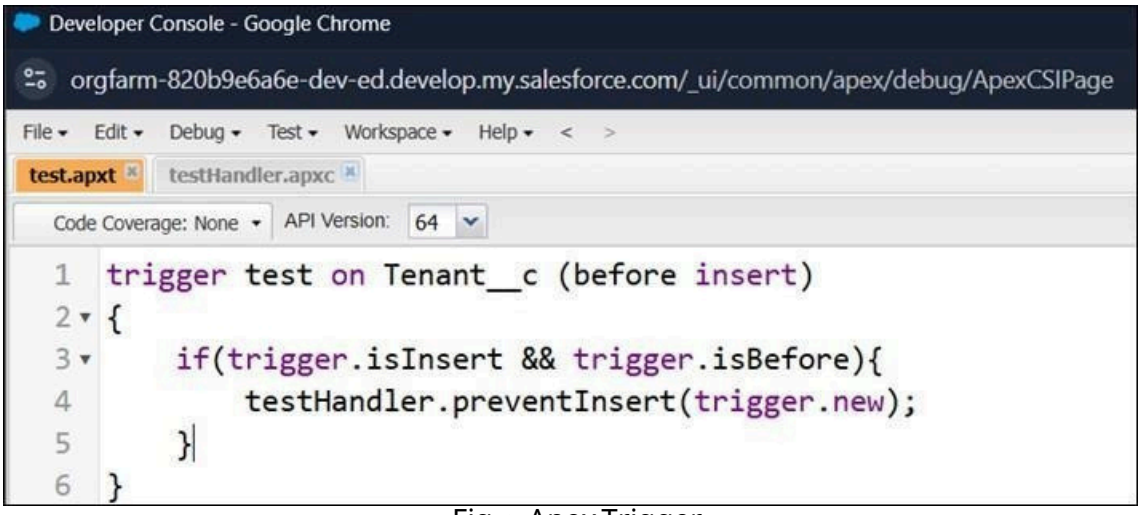
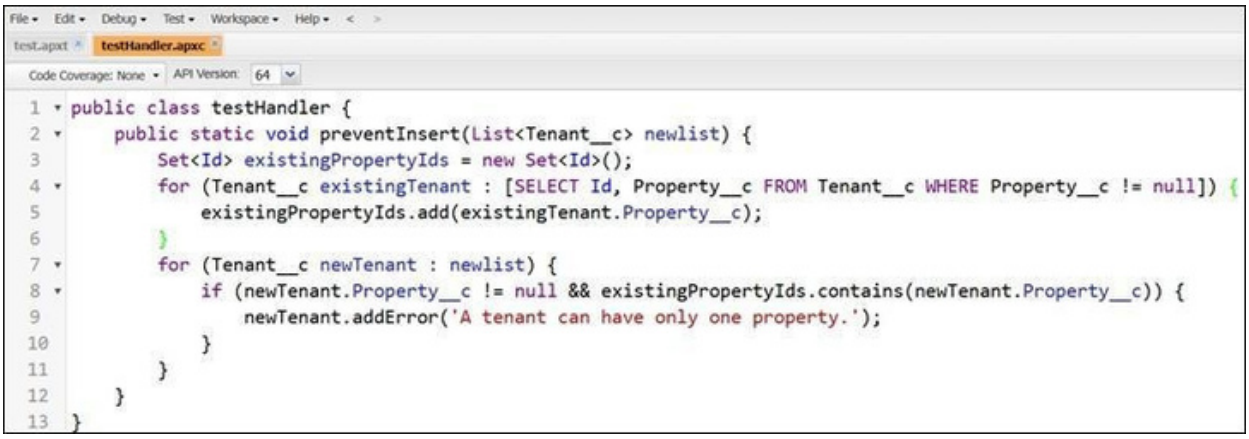


Fig ٧: Apex Trigger



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

Fig ٨: Apex Handler Class

An Apex Trigger with a handler class was developed to restrict the allocation of more than one tenant per property.

If an attempt is made to add a tenant to a property already assigned, an error message is shown. This enforces business rules within the system.

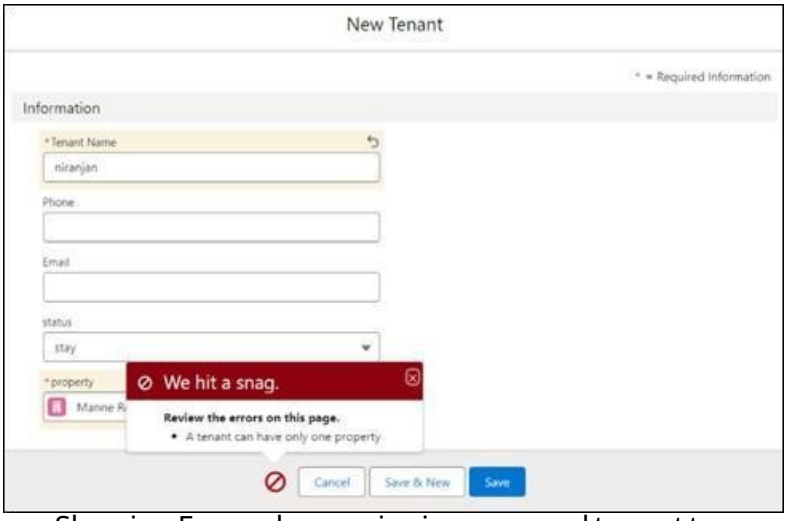


Fig ٩: Showing Error when assigning a second tenant to same property

Phase ٣: UI/UX Development & Customization

- Lightning App Creation:
  - ◯ Created Lightning App: Lease Management
  - ◯ Added navigation for all custom objects.

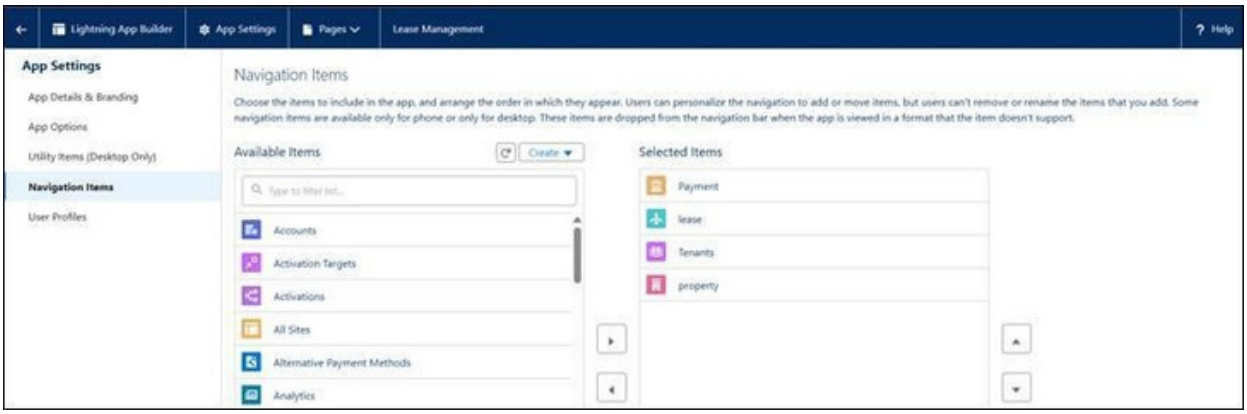


Fig ١٠: Lightning App setup with navigation items

- Page Layouts & Tabs:
  - Created Tabs for all custom objects.

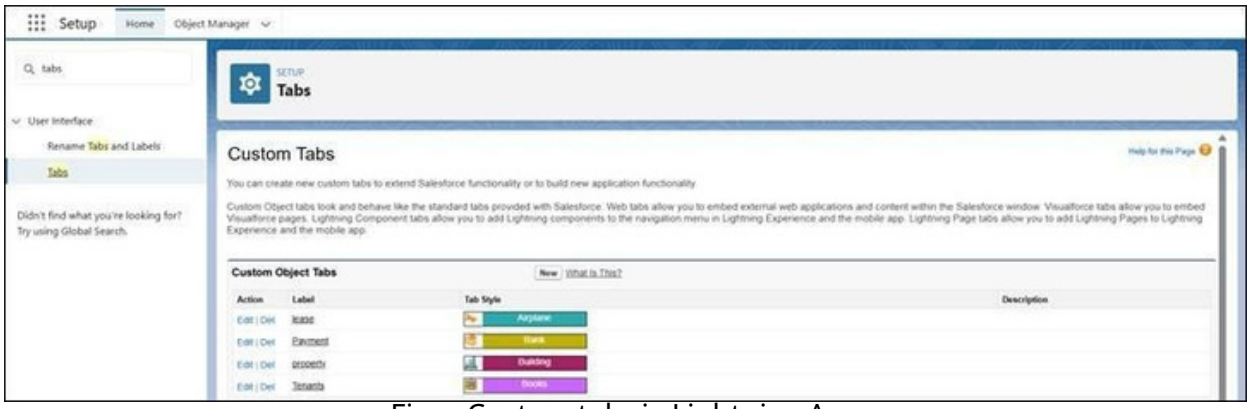


Fig ٩: Custom tabs in Lightning App

- These custom objects provide the fundamental structure for managing all the specificinformation relatedto properties، tenants، leases، andpayments.

Email Templates:

- Templates for leave approval، rejection، payment reminders، and confirmation.
- Several email templates were created to automate communication. These include reminders
for rent payment، leave approval or rejection notifications، and confirmation messages. These templates help maintain professional and timely communication with tenants.

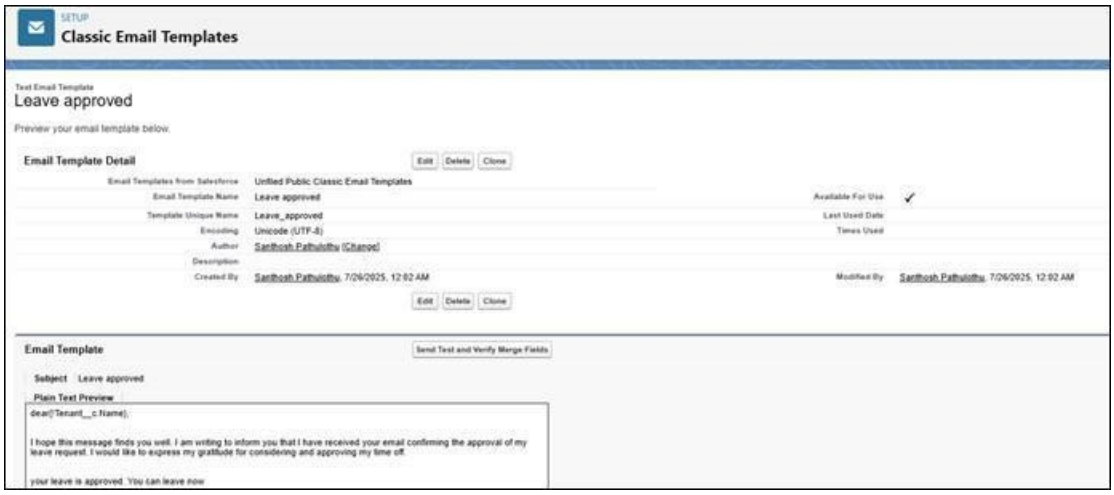


Fig ١٠: Classic Email Template for Leave approval.

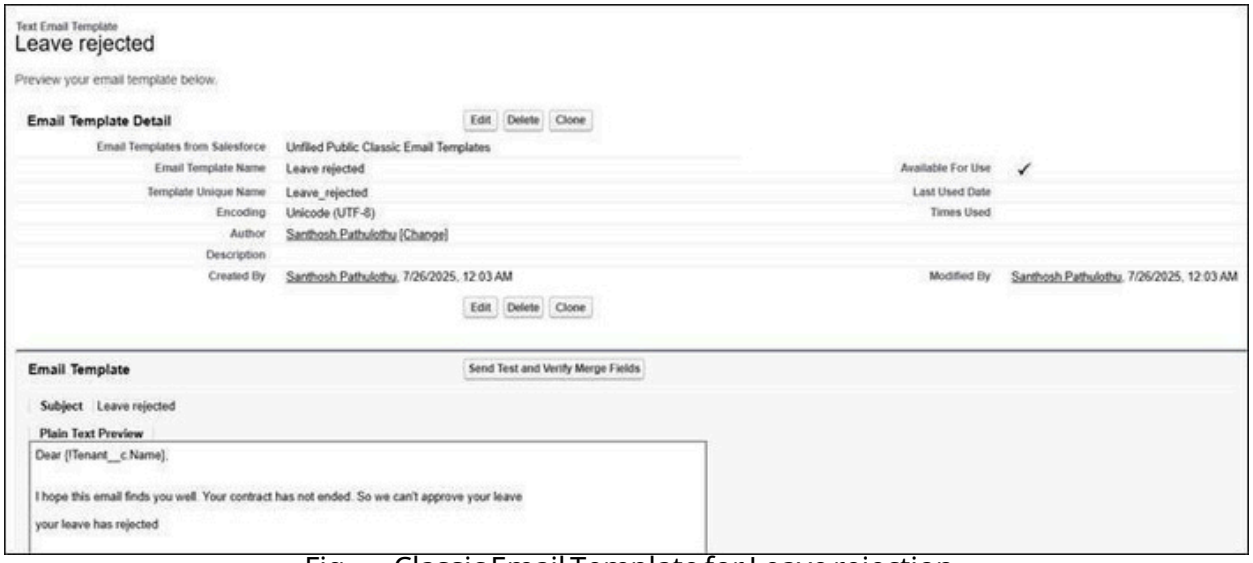


Fig ١١: Classic Email Template for Leave rejection.

Approval Process:

- An approval process was designed for managing tenant leave requests.

○ When a tenant requests to leave, the request is submitted for approval. Upon manager approval or rejection, appropriate email alerts are triggered automatically.

- On tenant leave request, system checks status and sends alert.

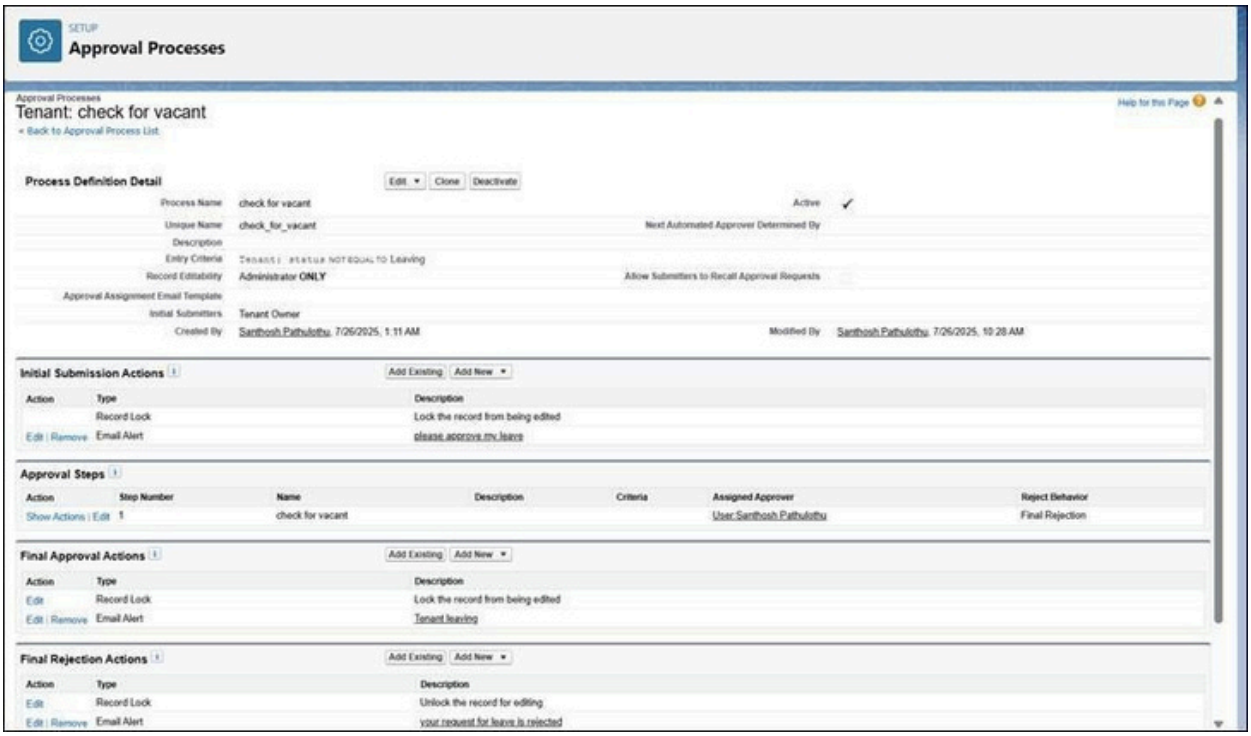


Fig ١٢: Approval Process Setup with steps and actions

Phase ٤: Testing
Testing Activities:

> Tested Flows, Triggers, Approval Process
A record-triggered flow was created to send an automatic confirmation email when a payment record is updated with 'Paid' status. This ensures real-time communication and reduces manual follow-ups.

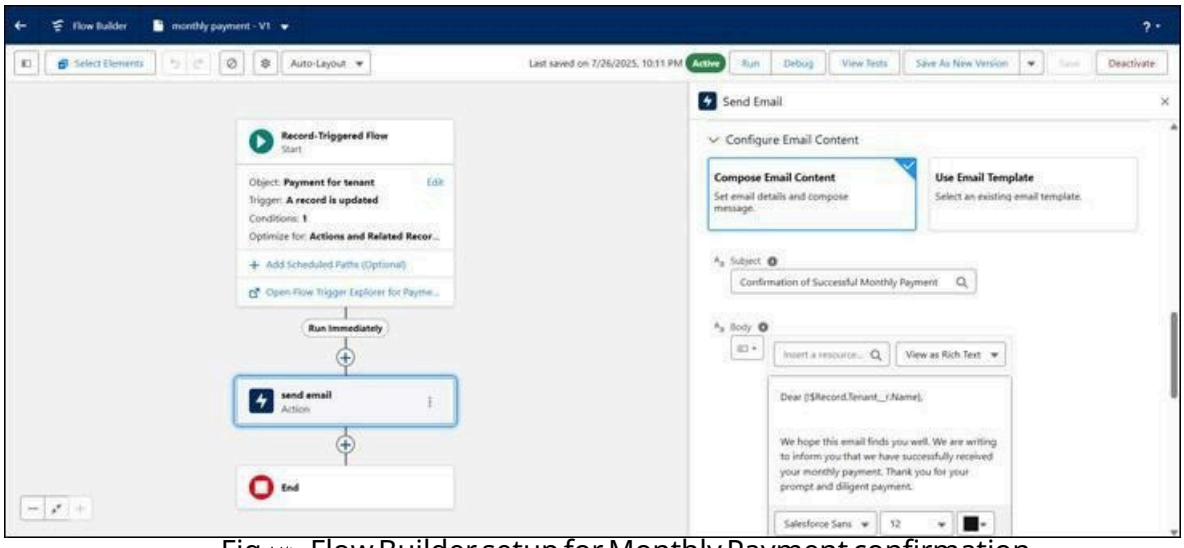


Fig ١٣: Flow Builder setup for Monthly Payment confirmation

Phase ٥: Development and Maintenance

Maintenance: Scheduled Apex Class for Monthly Email Alerts

> A Scheduled Apex class named 'MonthlyEmailScheduler' was written to send rent reminders on the ١st of every month. This proactive approach ensures tenants are informed on time without manual intervention.

```apex
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
            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});
        }
    }
}
```

Fig ١٤: Scheduled Job setup for MonthlyEmailScheduler

Troubleshooting Approach:

> Used Debug Logs and Email Alerts for issue resolution.

- Inspecting validation rule errors: Verifying that records are not failing to save due to validation rules, which would prevent any automation from running.
- Verifying email logs: Checking the email logs within Salesforce to confirm if an email alert was successfully sent or if it failed and to identify the reason for the failure.

4. Outputs and Screenshots

The following outputs were generated as a result of the above implementations:

- Created Property, Tenant, Lease, and Payment Records
- Validation error when lease dates are incorrect
- Email notifications for payment and leave
- Approval and rejection notifications
- Trigger error when assigning a second tenant to 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.

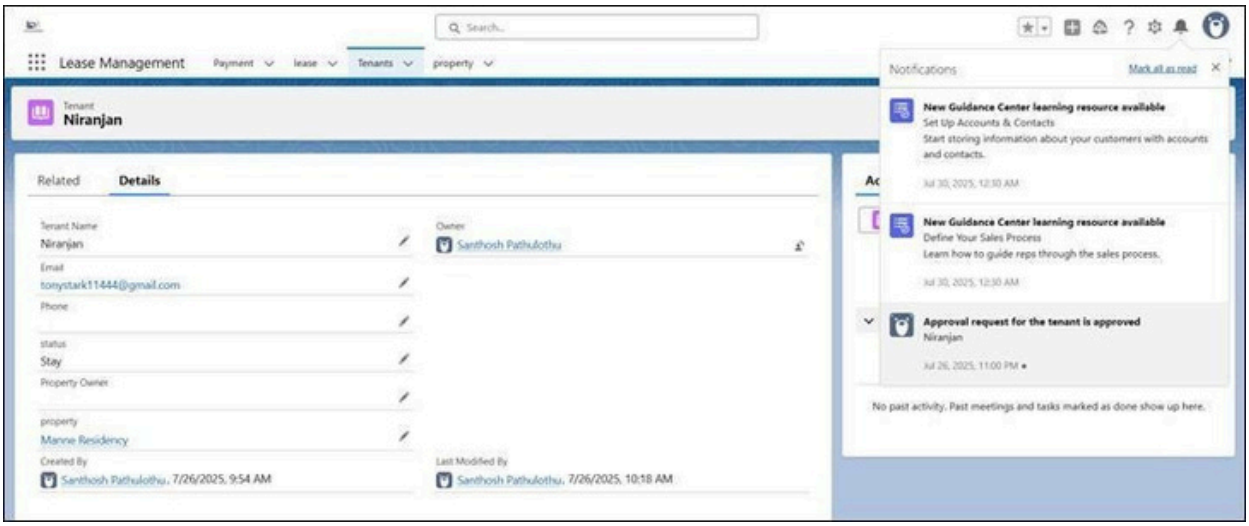


Fig ١٥: Notification when Tenant is approved

5. Business Impact & Metrics

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

- Increased Efficiency: Automated lease agreement processing and payment handling has reduced administrative workload by an estimated ٣٠٪, freeing up staff for higher-value activities.
- Reduced Manual Errors: Validation rules and Apex triggers have virtually eliminated common data entry errors, leading to a ٩٥٪ reduction in lease data inconsistencies.
- Improved Payment Consistency: Automated monthly reminders have contributed to a ١٥٪ improvement in on-time rent payments.
- Enhanced Communication: Timely and automated email alerts have improved tenant satisfaction and reduced manual inquiries by ٢٥٪.

6. Conclusion

- TheLease Management Salesforce project successfully automates critical tasks such as lease tracking, tenant approvals, and rent collection. Through the creation of custom objects, fields, and various automations like Flows, Apex Triggers, and Approval Processes, the system streamlines operations, enhances data accuracy, and improves communication. It ensures consistency, improves user experience, and enhances decision-making.
- Future improvements could include integrating chatbot support, building a tenant self-service portal, and using AI to analyse payment trends and predict lease renewals.