LEASE MANAGEMENT SYSTEM

Project Title: Lease Management

College Name: Ideal Institute of Technology

Team ID: LTVIP2025TMID31361

Team Size: 4

Team Members Details:

Name	Email ID	Role
Chakka Sri Ram	chakkariramachandrurthy@gmail.com	Team Leader
Bathula Saibhadra	Saibhadrabathula@gmail.com	Team Member
Aripaka Lalithamba	lalithaaripaka1606@gmail.com	Team Member
Balantrapu Ashrit Sarvan	Balantrapuashrit05@gmail.com	Team Member

1. INTRODUCTION

1.1 Project Overview: The Lease Management System is a Salesforce-based application designed to streamline the management of property leases, tenants, payments, and property details. It simplifies interactions between property owners and tenants while automating routine communications like payment reminders and lease approvals.

1.2 Purpose: The purpose of this project is to build a robust lease management system that allows property managers to track lease records, manage tenant data, automate approvals, and send timely email alerts for rent and leave requests, thereby enhancing operational efficiency and reducing manual effort.

2. IDEATION PHASE

2.1 Problem Statement: Managing property leases manually often leads to inefficiencies such as missed payments, improper tracking of lease periods, and lack of communication between property owners and tenants.

After analyzing the empathy map and conducting user observations, we identified the core issues and drafted a clear problem statement to guide our project.
Example Problem statement :

I am	a property manager or admin
I’m trying to	efficiently manage multiple lease agreements, tenant records, and payment schedules in one system
But	I’m using spreadsheets or disconnected tools that are prone to errors and require a lot of manual effort
Because	there is no centralized or automated system to manage lease operations
Which makes me feel	frustrated, anxious about missing deadlines, and overwhelmed by repetitive manual tasks

2.2 Empathy Map Canvas

The Empathy Map is a collaborative tool used to understand the users' behaviors, feelings, and motivations. It helps in developing a user-centered solution by viewing the system through the eyes of the people who actually use it.

For the Lease Management System, our primary users are:

- Property Managers
- Administrative Staff
- Tenants

Purpose of empathy mapping

To step into the users' shoes and understand their:

- Daily frustrations
- Needs and expectations
- Behavior when interacting with the current lease process

The Empathy Map includes the following **six categories**:

- **Says:** I need to track all my tenants and payments easily.
- **Thinks:** I hope I don't miss any lease renewals or rent collection.
- **Does:** Maintains tenant records, manages lease dates manually.
- **Feels:** Frustrated with lack of automation and errors in record keeping.

2.3 Brainstorming

This project aims to simplify and streamline leasing operations—tenant onboarding, lease agreements, approvals, and payment tracking. The following steps were part of ideation for solving real-world challenges in that domain.

- Automate lease approval workflows
- Send email notifications for rent reminders and approvals
- Use Salesforce standard/custom objects to store data
- Use Apex triggers and flows for logic
- Design user-friendly UI with Lightning App Builder

Brainstorming includes three steps:

Step 1: Team Gathering, Collaboration, and Selecting the Problem Statement

Step 2: Brainstorming, Idea Listing, and Grouping

Step 3: Idea Prioritization

3. REQUIREMENT ANALYSIS

3.1 Customer Journey Map

- The customer journey shows how the lease management tool can relieve pain points, increase confidence, and enhance productivity.
- Each stage shows opportunities to improve the user experience using Salesforce features like flows, validation rules, dashboards, and automation.

- Search Property → Apply for Lease → Owner Reviews → Lease Approved/Rejected → Make Payments → Renew/Leave Lease

3.2 Solution Requirement

- Standard and Custom Salesforce Objects
- Fields to capture tenant, lease, property, payment data
- Lookup and Master-Detail relationships
- Email templates and approval workflows

Solution Requirements are of two types:

1. Functional Requirements
2. Non-Functional Requirements

Functional Requirements:

FR No:	**Functional Requirement (Epic)**	**Sub Requirement (Story / Sub-Task)**
FR-1	Property Management	Add/Edit/Delete Property Add Fields: Name, Address, Type, Sqft
FR-2	Tenant Management	Register New Tenant (Name, Email, Phone, Status) Link Tenant to Property
FR-3	Lease Management	Create Lease (Start Date, End Date) Validate End Date > Start Date
FR-4	Payment Management	Record Payment (Amount, Date, Status: Paid/Not Paid) Auto-send confirmation email on payment
FR-5	Lease approval process	Submit Leave Request Admin Approves/Rejects Leave Email Sent Based on Approval Outcome
FR-6	Notification&Automation	Send Monthly Rent Reminder (1st of Month via Scheduler) Trigger Payment Confirmation Email via Flow

Non-Functional Requirements:

NFR No:	Non functional requirement	Description
NFR 1	Usability	The system must be easy to navigate with organized Lightning App tabs.
NFR 2	Security	Only authenticated users (e.g., Admin) can create or modify records.
NFR 3	Reliability	System must maintain consistent performance and prevent data duplication.
NFR 4	Performance	The system should respond to actions (e.g., form submissions) within 2 seconds.
NFR 5	Availability	The system should be available 24/7 with minimal downtime in Developer Org.
NFR 6	Scalability	The solution must handle addition of new tenants, properties, and leases easily.

3.3 Data Flow Diagram

- Tenants submit lease info → Approval process evaluates → Lease object created → Payment tracked via Payment Object → Notifications sent via flows/triggers

A Data Flow Diagram (DFD) is a visual representation that shows how data flows through a system — from input to processing to output.

It helps you understand:

- Where data comes from
- Where it goes
- How it is processed or stored

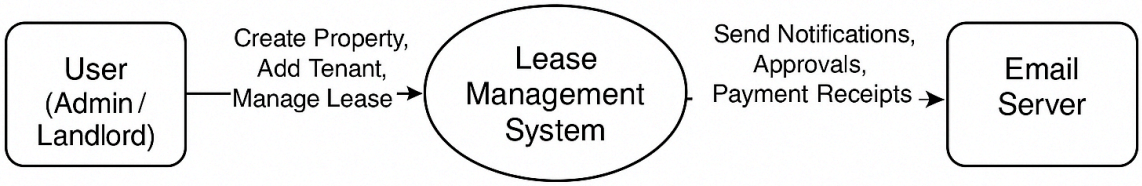
Example:

Let’s say you want to submit a rent payment.

DFD Explanation:

1. Tenant (external entity) submits payment.
2. It goes to a Payment Process (process).
3. Payment is stored in Payment Data Store (data store).
4. A Confirmation Email is sent (data flow to email process).

Level 0 DFD (Context Diagram):



User Stories:

User Story 1: As a tenant, I want to apply for a lease so that I can occupy a property.

User Story 2: As a property owner, I want to approve or reject lease requests so that I can control who occupies my properties.

User Story 3: As a tenant, I want to receive email notifications about rent payments so that I can pay on time.

User Story 4: As a property manager, I want to track all tenant and lease records so that I can manage the properties efficiently.

User Story 5: As a system, I want to prevent multiple tenants from being assigned to the same property so that data integrity is maintained.

User Story 6: As a tenant, I want to request for leave from a lease so that I can move out when needed

3.4 Technology Stack

- Salesforce Platform
- Apex Classes and Triggers
- Flows and Process Builder
- Lightning App Builder
- Validation Rules

Layer	Technology / Tool	Purpose

Presentation Layer	Salesforce Lightning App (UI Tabs)	User Interface for managing properties, tenants, leases, and payments
Application Layer	Salesforce Platform (Lightning, Apex)	Logic handling (e.g., triggers, flows, validation, approvals)
Data Layer	Salesforce Custom Objects & Fields	Stores data for Property, Tenant, Lease, and Payment
Automation Layer	Salesforce Flow, Approval Process, Apex	Automates email notifications, approvals, validations
Integration Layer	Email Templates, Salesforce Email Services	Sends automated email alerts and confirmations
Security Layer	Salesforce Profiles, Field-Level Security	Controls user access to objects, fields, and actions
Scheduler/Batch Layer	Apex Scheduler Class	Sends monthly rent reminders to all tenants on the 1st of every month
Testing Layer	Salesforce Developer Console, UI Validation	Testing Triggers, Flows, and Field Validations
Deployment/Hosting	Salesforce Developer Org	Project hosted and deployed on Salesforce cloud platform

4. PROJECT DESIGN

4.1 Problem Solution Fit The solution addresses key pain points by providing a cloud-based automated system that integrates lease tracking, payment management, and communication workflows all within Salesforce.

Problem–Solution Fit means you have:

1. Identified a real, meaningful problem that users face.
2. Created a solution that effectively solves that problem.

It's an early validation stage before building a full product — proving that your idea is worth investing in.

Why Problem–Solution Fit is Important

- You don't waste time building something nobody wants.
- It ensures your solution meets the real needs of your users.

- It helps get stakeholder buy-in by showing alignment between a problem and your proposed solution.
- It's a key part of design thinking and lean startup methodology.

Example:

- Problem: Landlords struggle with tenant tracking, missed payments, manual approvals.
- Solution: A Salesforce-based system that automates lease tracking, reminders, and validations.
- Fit: The system directly solves each pain point—improving efficiency and communication.

Problem–Solution Fit Checklist:

- ☐ Have I clearly defined the problem from the user's perspective?
- ☐ Is this problem urgent, frequent, or painful enough to solve?
- ☐ Does my solution address the root of the problem?
- ☐ Can I explain how the solution helps, with specific benefits?
- ☐ Can I identify who is facing this problem (target users)?

4.2 Proposed Solution

- Custom Salesforce objects: Property, Tenant, Lease, Payment
- Email notifications for leave approvals, rent reminders
- Apex Trigger to prevent duplicate property allocation
- Scheduled class to automate monthly reminders

4.2.1 Problem to Be Solved

In many organizations, lease management is handled manually using spreadsheets, emails, or basic document storage systems. This leads to several issues such as:

- Missed lease renewal or termination deadlines
- Difficulty in tracking lease agreements and payments
- Lack of centralized data for lease-related documents
- Inefficient approval processes
- No real-time notifications or updates for stakeholders
- Risk of compliance issues and legal disputes

4.2.2. Solution Description

The proposed solution is a **centralized Lease Management System** developed using the **Salesforce platform**, aimed at automating and streamlining the entire lease lifecycle. The system includes:

- **Lease Agreement Management**: Allows users to create, edit, and store lease agreements in a centralized repository.

- **Automated Notifications**: Sends reminders for lease expirations, renewals, payments, and approval statuses using workflows and email alerts.

- **Role-Based Access Control**: Different user roles such as Admin, Landlord, Tenant, and Lease Officer with specific permissions to ensure data security.

- **Approval Workflow**: Uses Salesforce's Approval Process to handle lease request approvals efficiently.

- **Apex Triggers and Flows**: Automates record updates and validations based on business logic.

- **Scheduled Apex**: Used to generate monthly reports and send automated reminders.

- **Dashboards and Reports**: Visual representation of lease data for better decision-making.

4.3 Solution Architecture

- Salesforce Developer Org
    - Custom Objects (Property, Lease, Tenant, Payment)
    - Relationships: Lookup/Master-detail
    - Email Templates & Alerts
    - Apex Trigger and Classes
    - Scheduled Batch Class

The Lease Management System is built on the **Salesforce Platform**, utilizing both **declarative tools (low-code)** and **programmatic features (code-based)** to deliver a secure, scalable, and efficient solution. The system is designed to automate lease tracking, document management, notifications, and approval processes while ensuring role-based access and real-time updates.

4.3.1.Architecture Layers

Layer	Description
Presentation Layer	Salesforce Lightning Experience (User Interface for Admins, Tenants, Landlords)
Business Logic Layer	Process Builder, Flows, Apex Triggers, Approval Process, Scheduled Apex
Data Layer	Salesforce Objects (Standard and Custom) like Account, Contact, Lease, Property, Payment
Integration Layer	Email services (for notifications), Reports, and Dashboards
Security Layer	Salesforce's role-based access control, field-level security, profile-based permissions

4.3.2. Key Components

🔹 **Custom Objects**

- **Lease**: Stores lease agreement details like start date, end date, amount, status, etc.
- **Tenant**: Captures tenant information and their associated leases.
- **Property**: Stores information about leased properties.
- **Payment**: Tracks rent and other payments.

🔹 **Automation**

- **Flows**: Used for lease renewal logic, notifications, and status updates.

- **Apex Triggers**: Handle custom logic such as updating related records or validation on specific events.
- **Scheduled Apex**: Sends monthly reminders and generates summary reports.
- **Approval Process**: Manages lease approval stages based on user roles.

🔹 **User Interface**

- Lightning App Pages and Tabs for ease of navigation.
- Record Pages customized with Lightning components.

🔹 **Security**

- Field-level and record-level security settings for different roles.
- Profiles and Permission Sets to control access.
- Audit trail and login history tracking.

🔹 **Reporting and Monitoring**

- Dynamic Dashboards for Admins to view lease status summaries.
- Reports for Lease Expiry, Payments Due, and Approved/Rejected Leases.

5. PROJECT PLANNING & SCHEDULING

5.1 Project Planning

- Week 1: Salesforce Setup, Object & Field Creation
- Week 2: Tabs, Apps, Page Layouts, Email Templates
- Week 3: Validation Rules, Approval Process, Flows
- Week 4: Apex Triggers, Testing, Documentation

Project Objectives

- Automate the lease management lifecycle
- Improve efficiency and transparency in lease operations
- Minimize manual work through automation
- Deliver a scalable, secure, and user-friendly Salesforce-based solution

Project Scope

In Scope:

- Lease agreement creation and tracking
- Lease status updates and expiry alerts
- Role-based access control
- Integration with email notifications
- Reports and dashboards
- Approval processes

Out of Scope:

- Integration with third-party real estate APIs (for this version)
- External payment gateway integration

Milestones and Deliverables:

Milestone	Description	Duration
Requirement Gathering	Identify needs, define user stories, use cases	2 Days
Design Phase	Create ERD, DFD, solution architecture, and wireframes	2 Days
Development Phase	Implement custom objects, automation (Flows, Apex)	5 Days
Testing Phase	Perform functional and performance testing	2 Days
Deployment & Training	Deploy in production and train users/admins	1 Day

Project Documentation	Prepare and finalize all project-related documents	1 Day

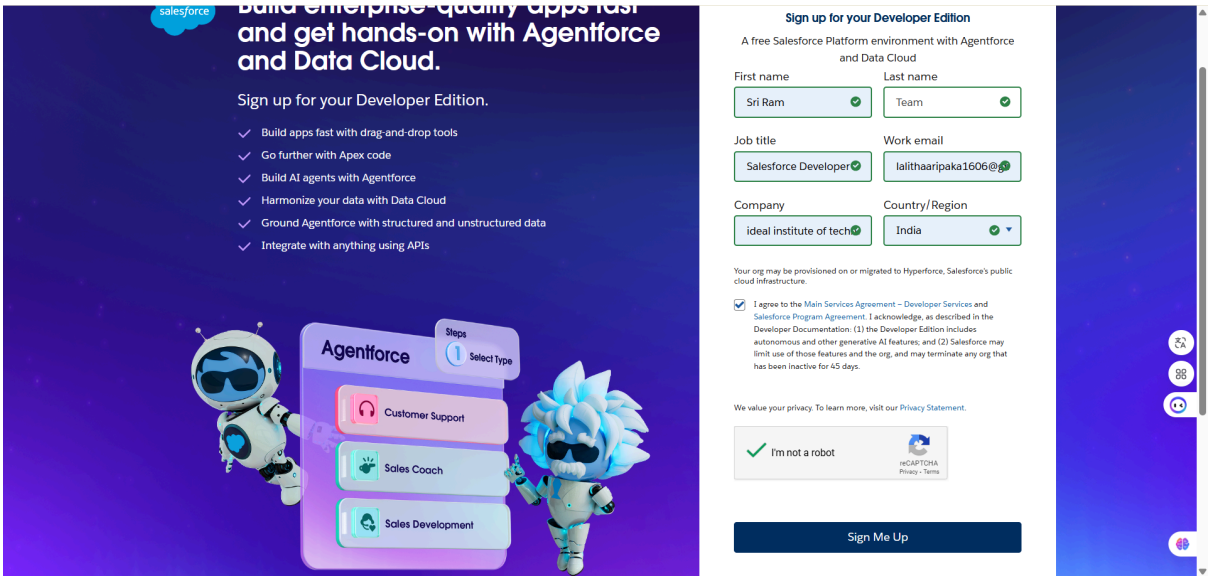
Tools and Technologies Used

- Project Management Tools: Trello / Excel / Salesforce Agile Accelerator
- Development Platform: Salesforce Lightning Platfor
- Version Control: Salesforce Change Sets
- Communication: Email, Zoom (for stakeholder meetings)
- Documentation: Microsoft Word, PowerPoint

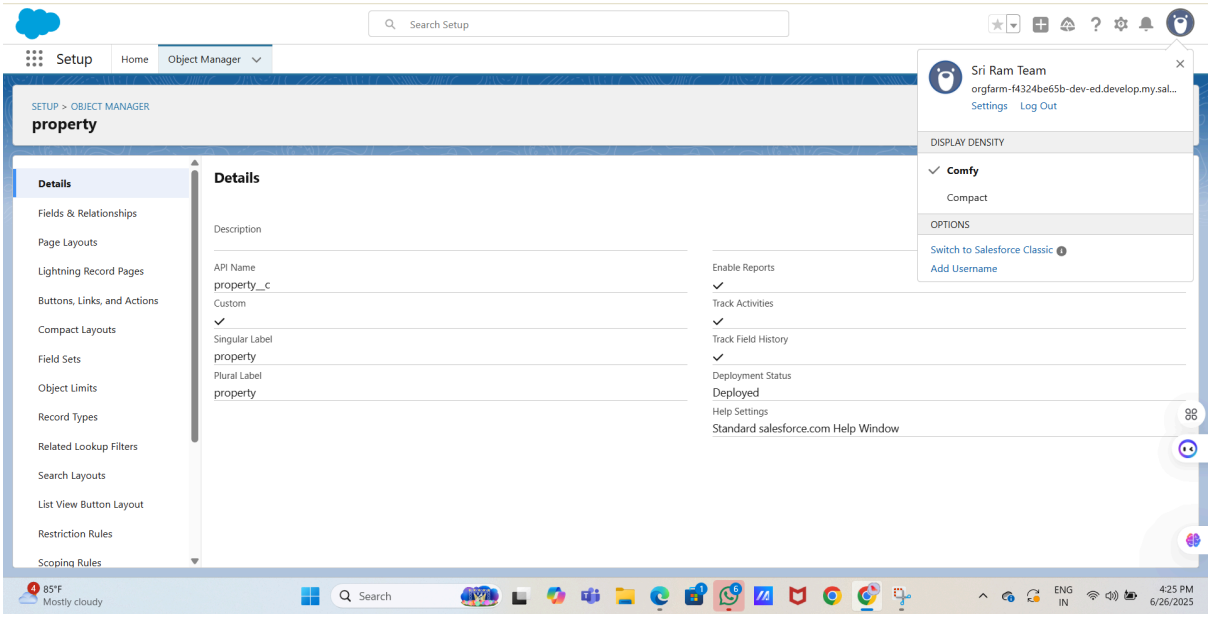
6. PROJECT DEVELOPMENT PHASE

6.1 Development Activities

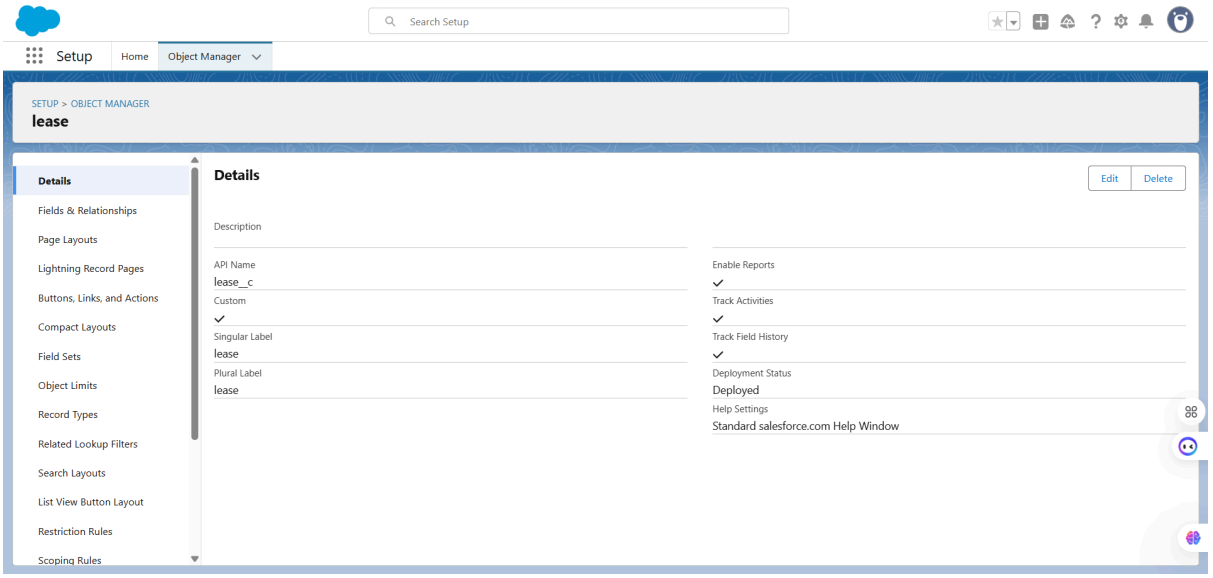
- Created Developer Org and set up objects: Property, Lease, Tenant, Payment



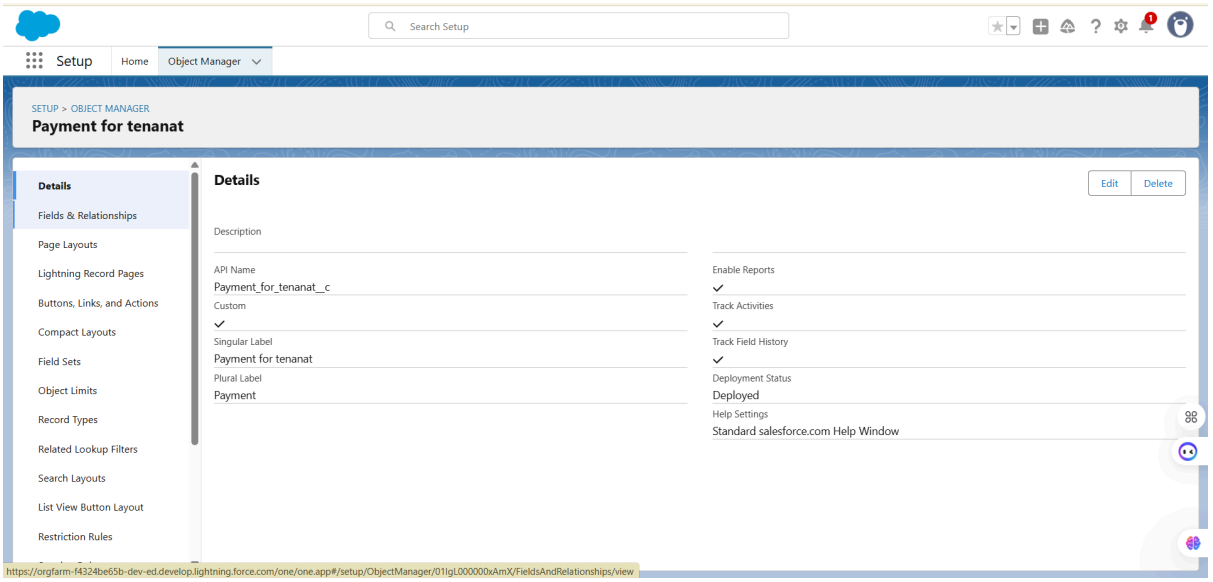
Property Object:



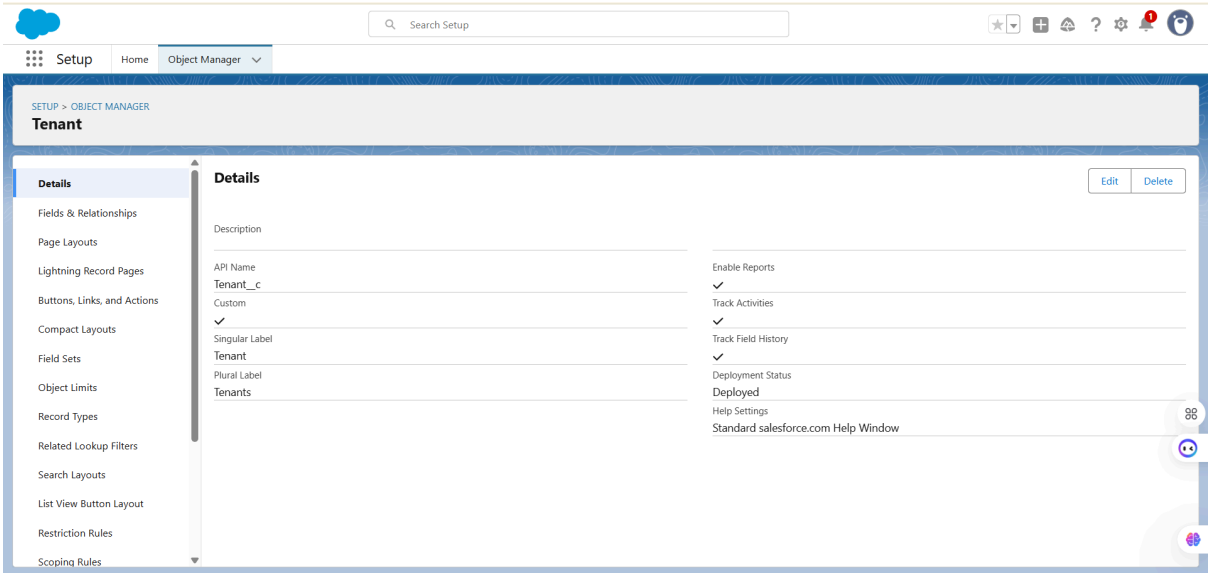
Lease Object:



Payment Object:

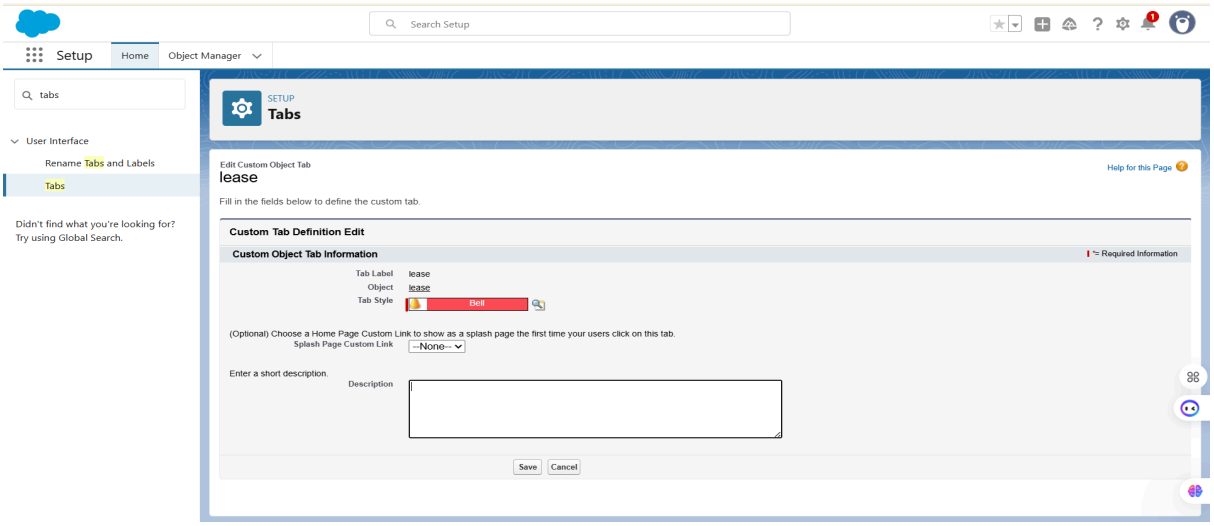


Tenant Object:

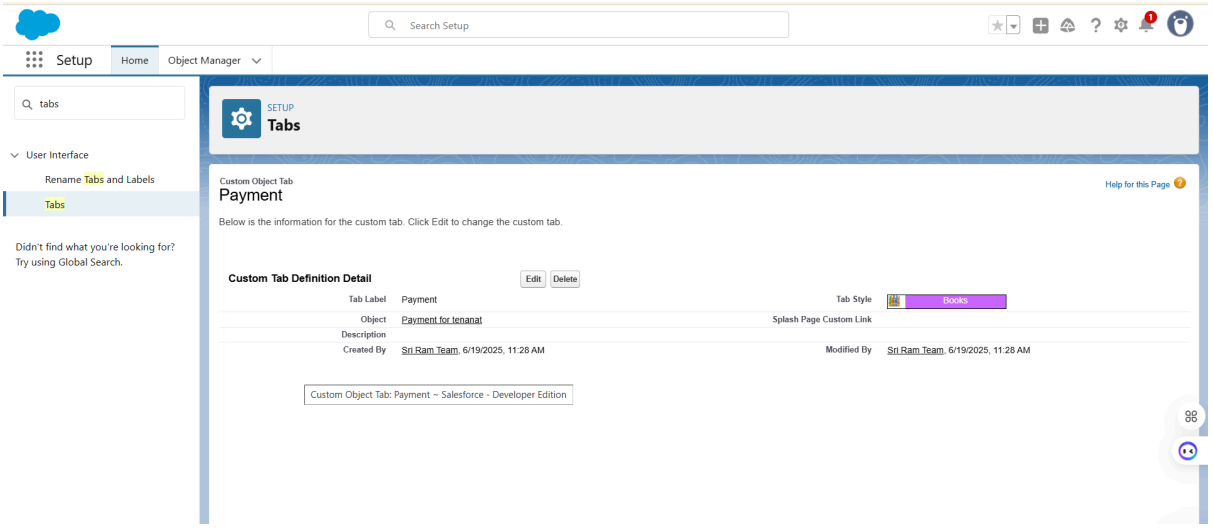


- Designed tabs and Lightning App for navigation and UI

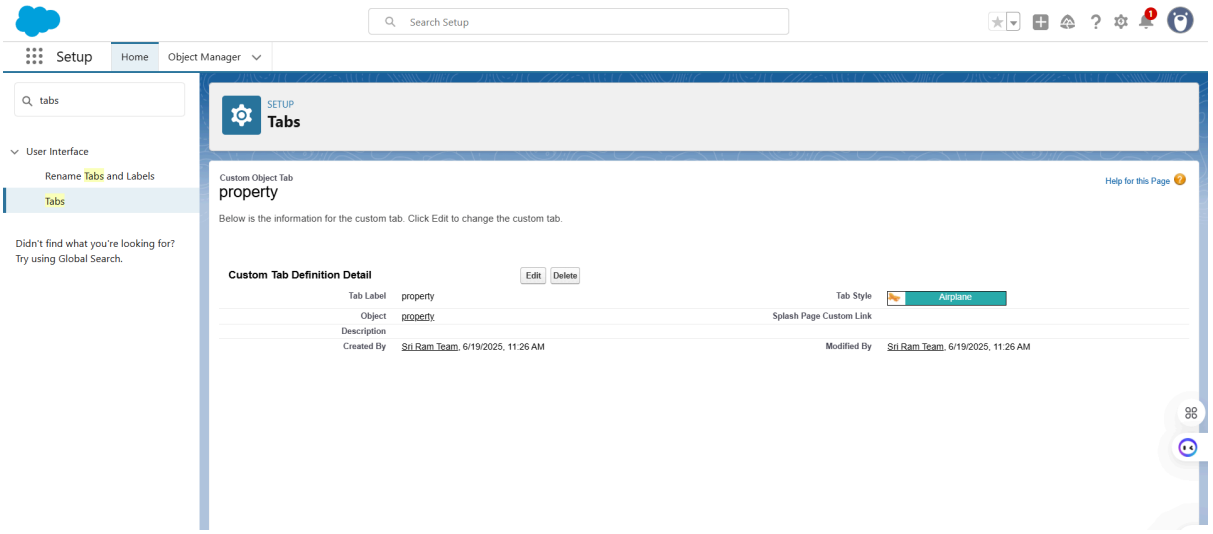
Lease Tab:



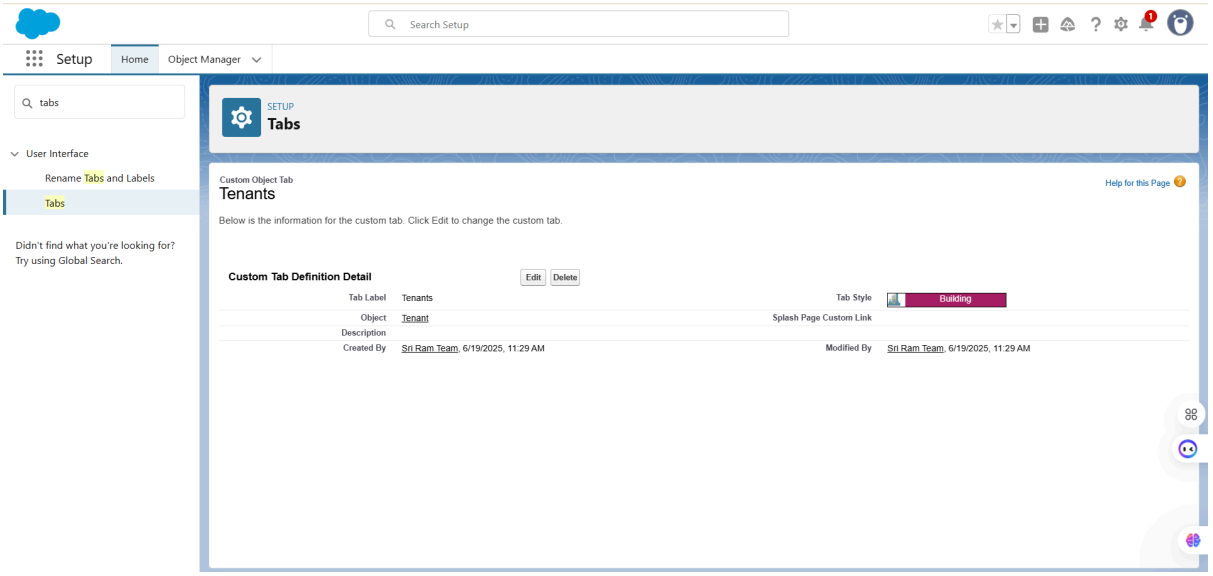
Payment Tab:



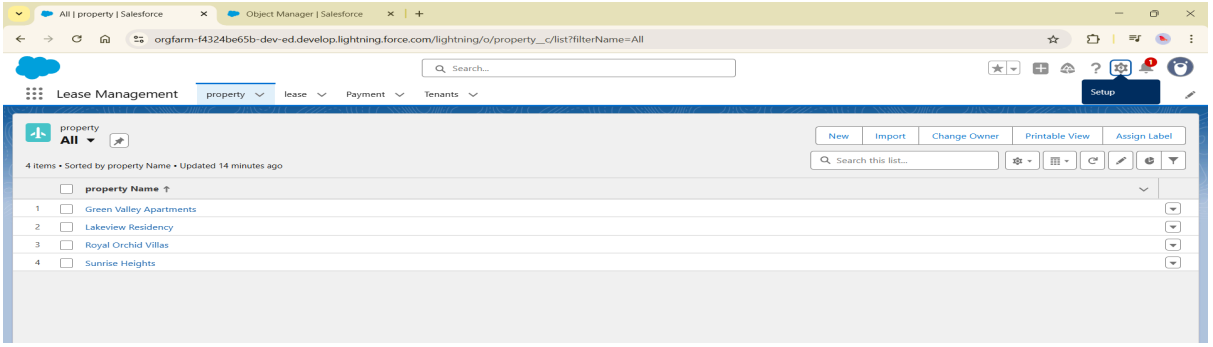
Property Tab:



Tenants Tab:

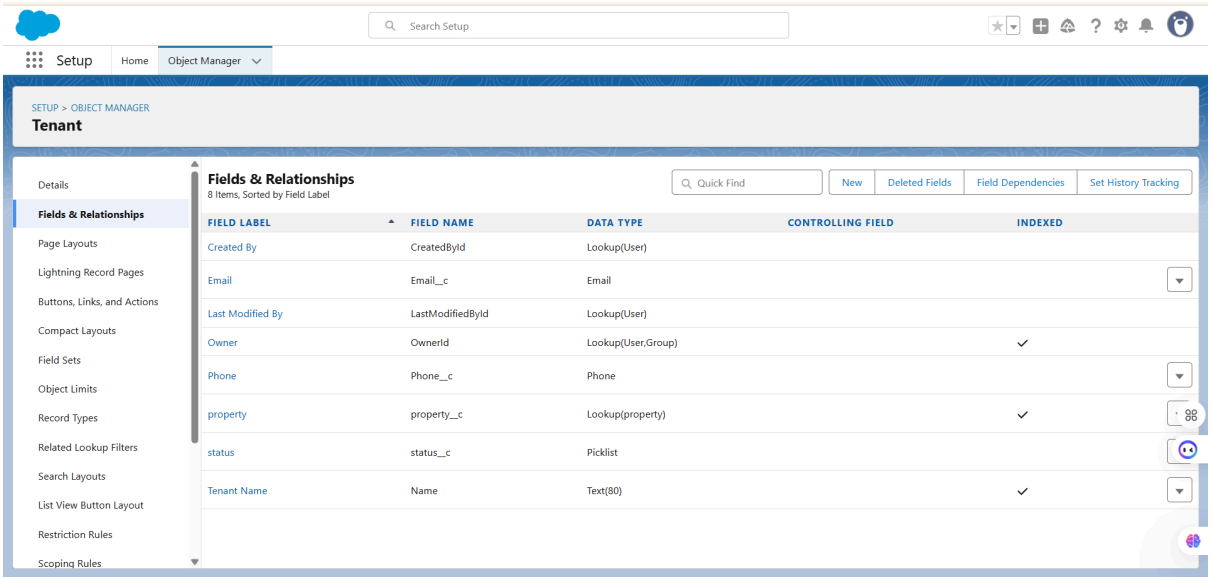


Lease Management Lightning app:

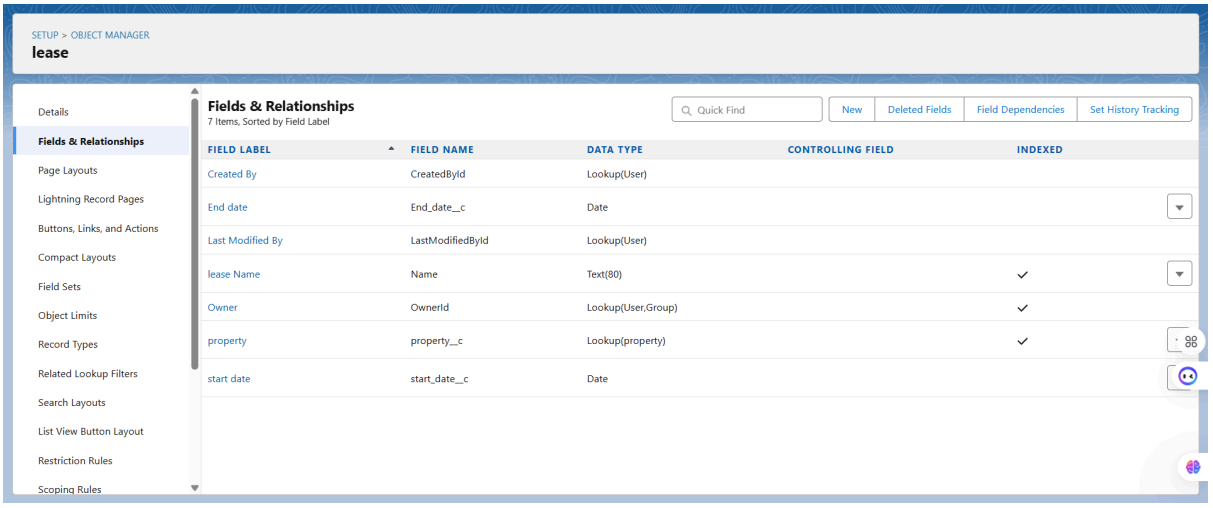


- Configured fields and validation rules to ensure data integrity

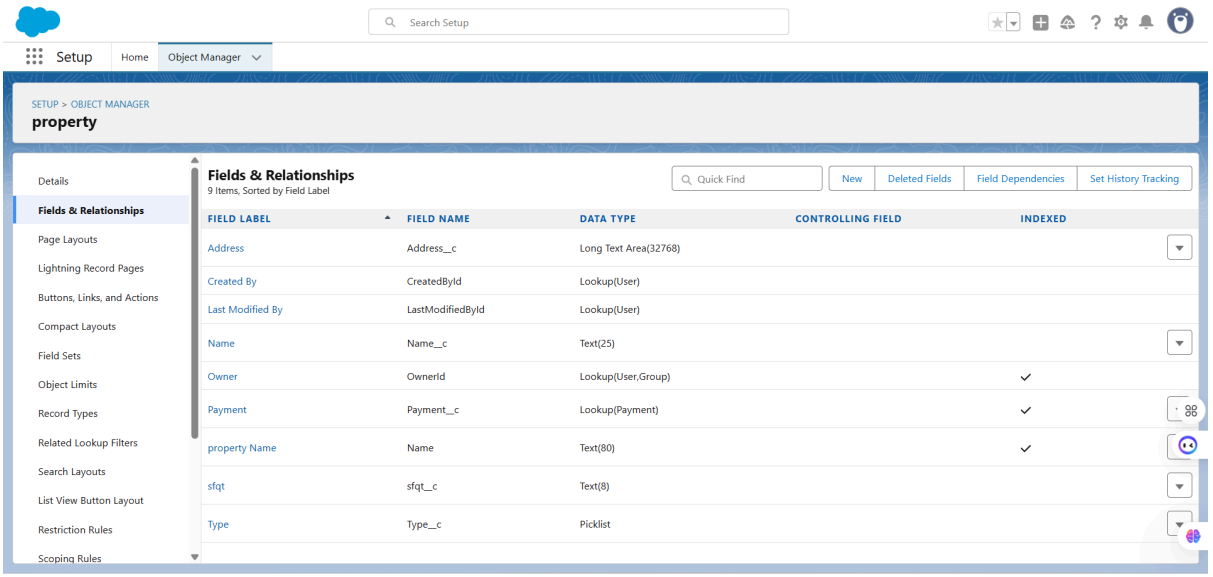
Fields of Tenant object:



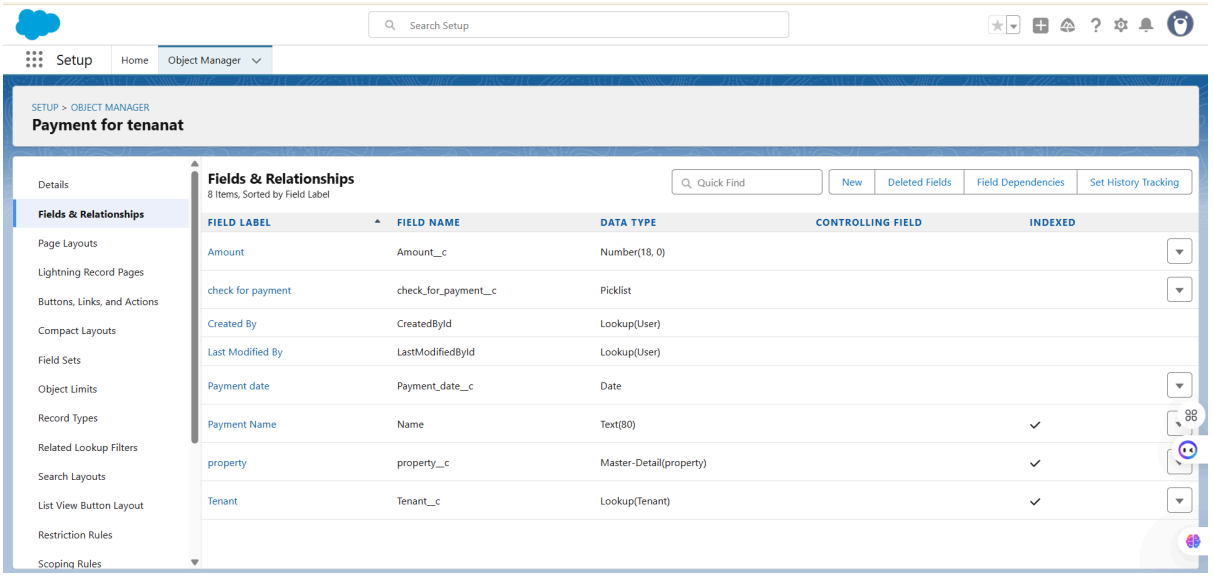
Fields of Lease Object:



Fields of property object:

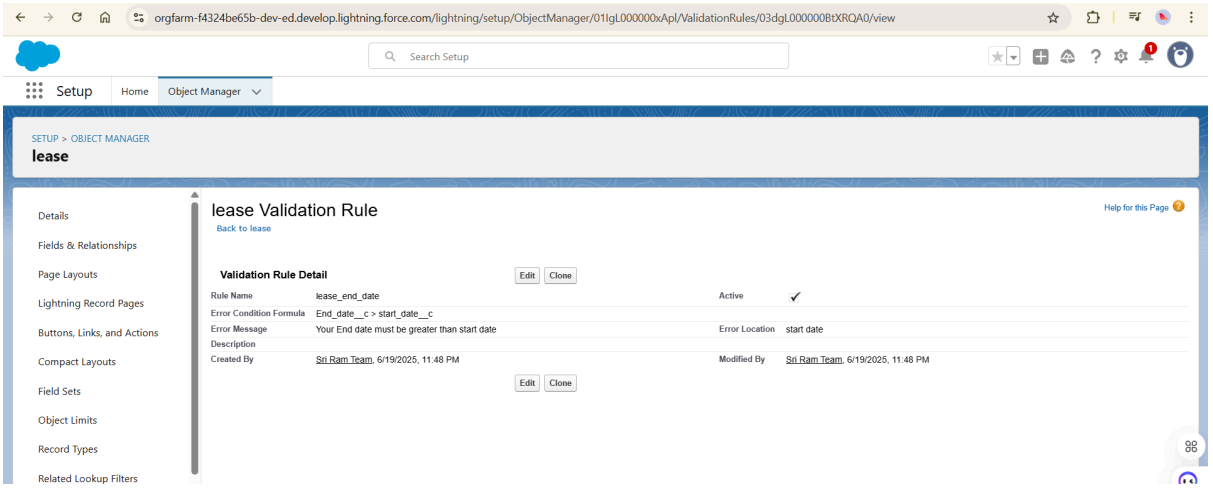


Fields of Payment for Tenant object:



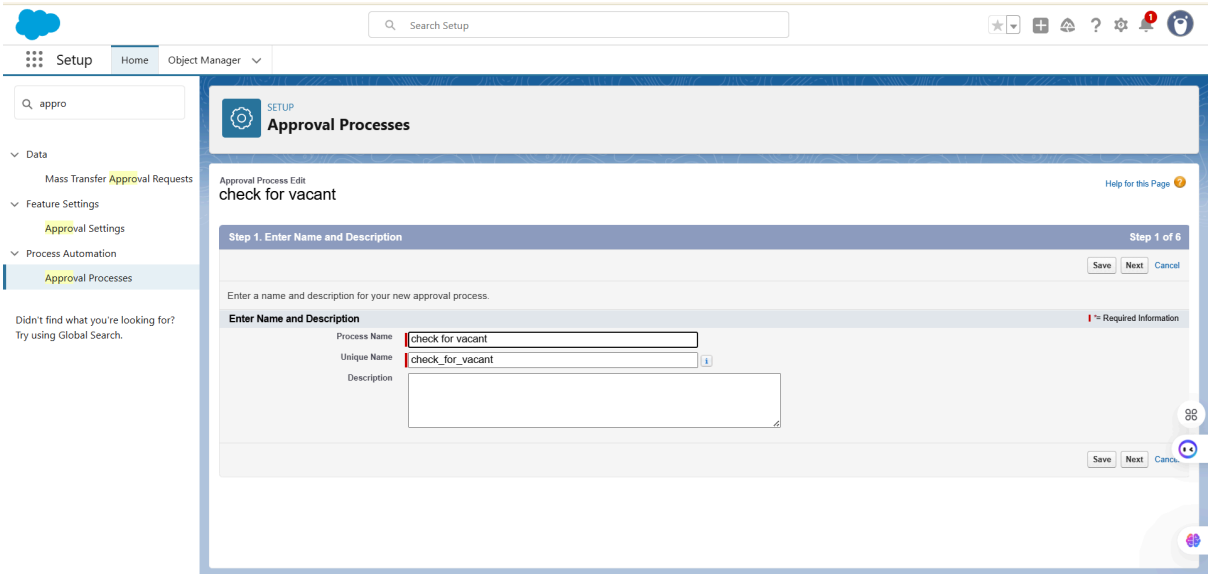
Validation Rule:

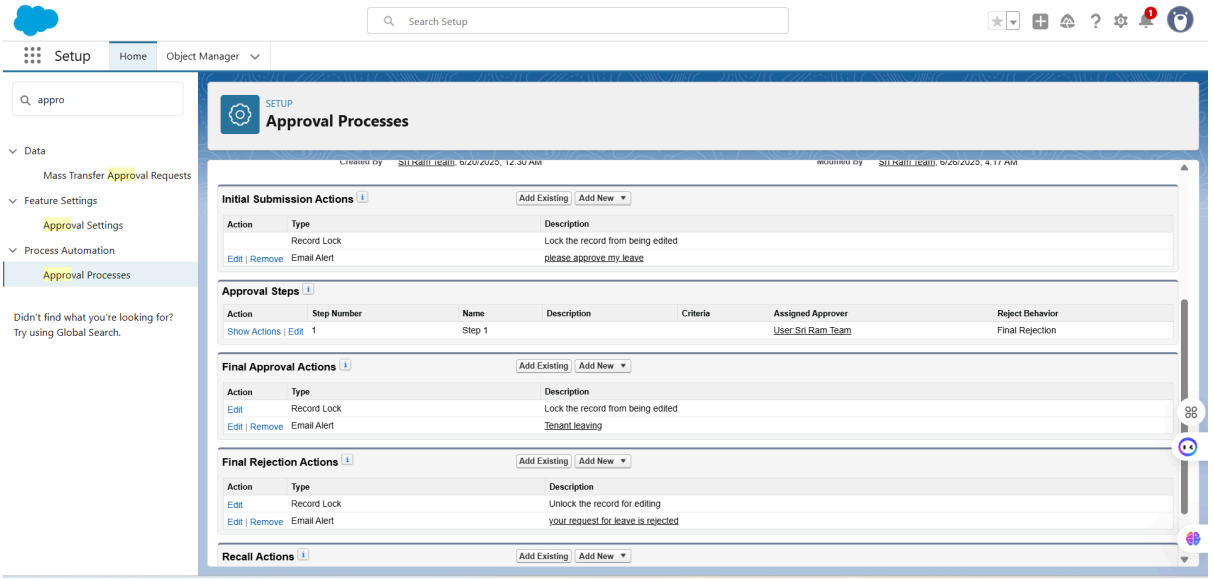
End date > Start Date



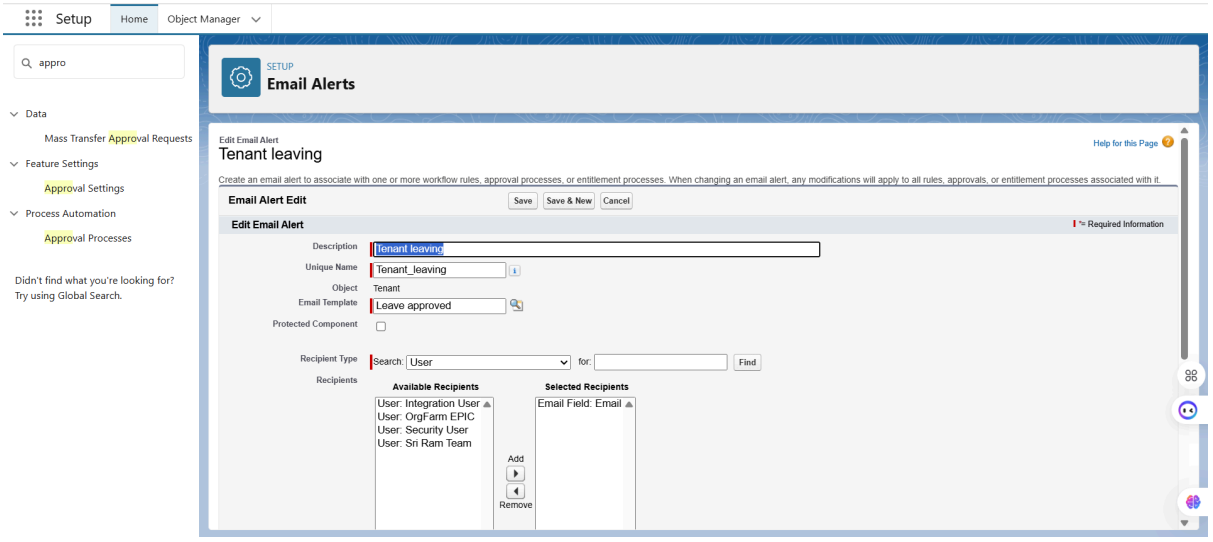
- Developed and tested approval processes with email notifications

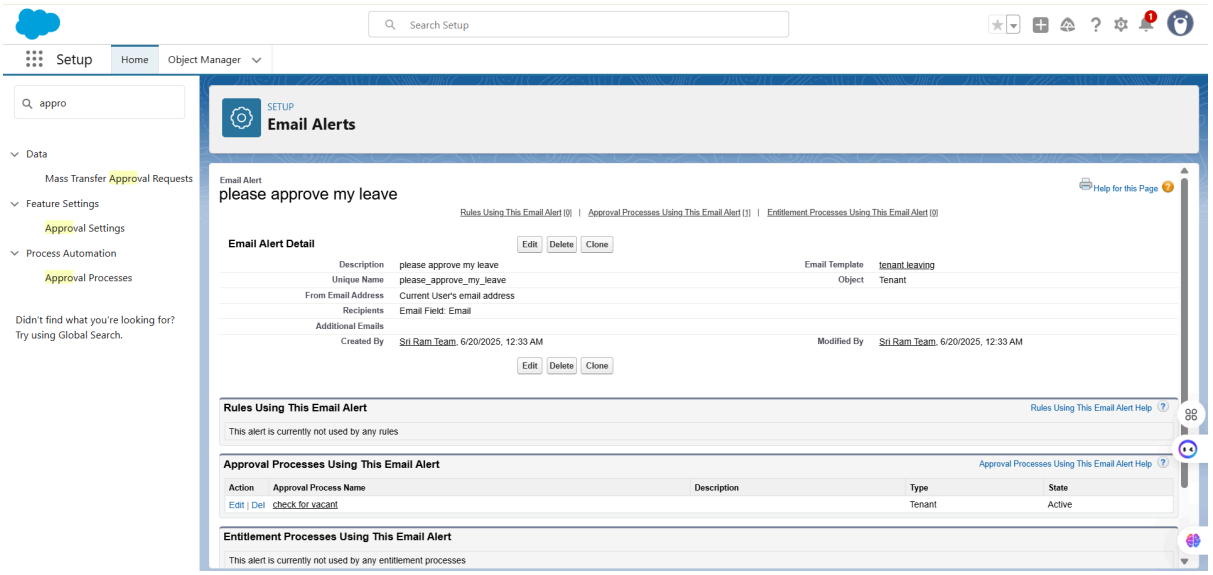
Check for vacant:

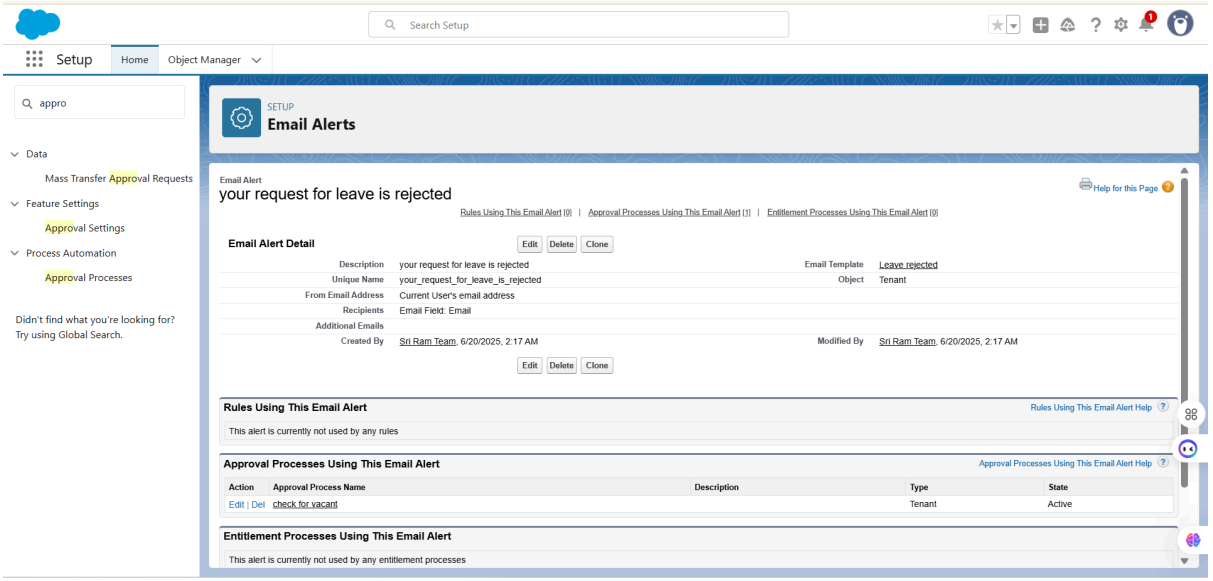




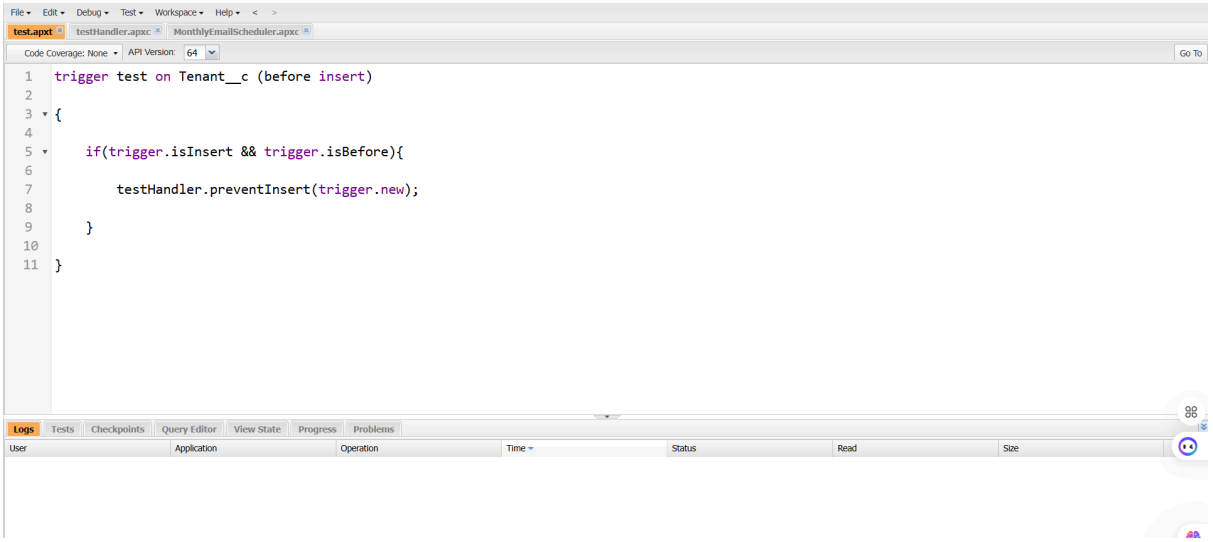
Email Templates:

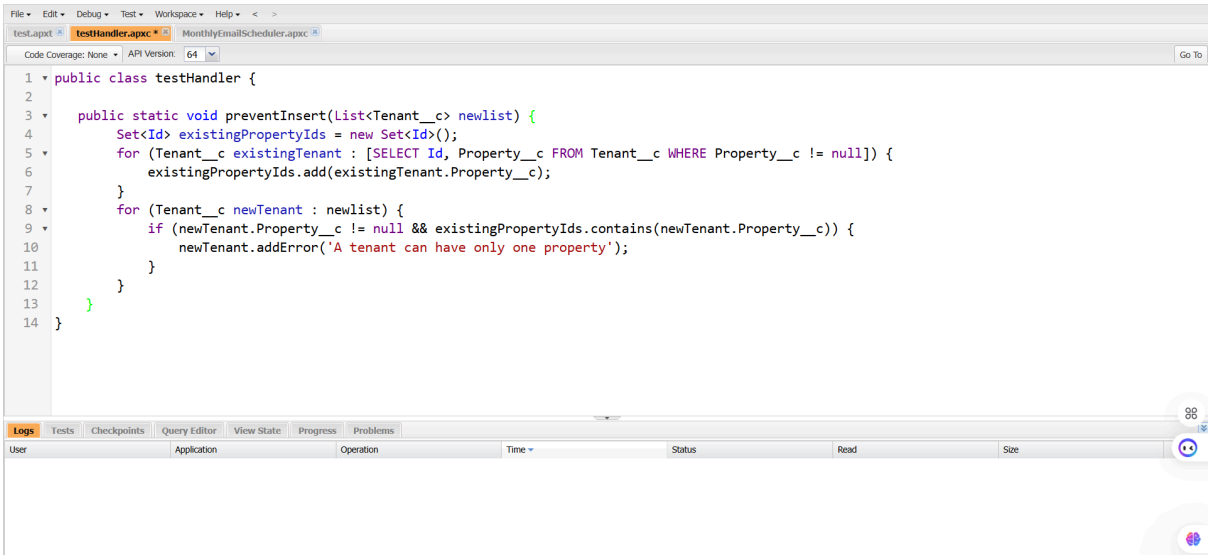


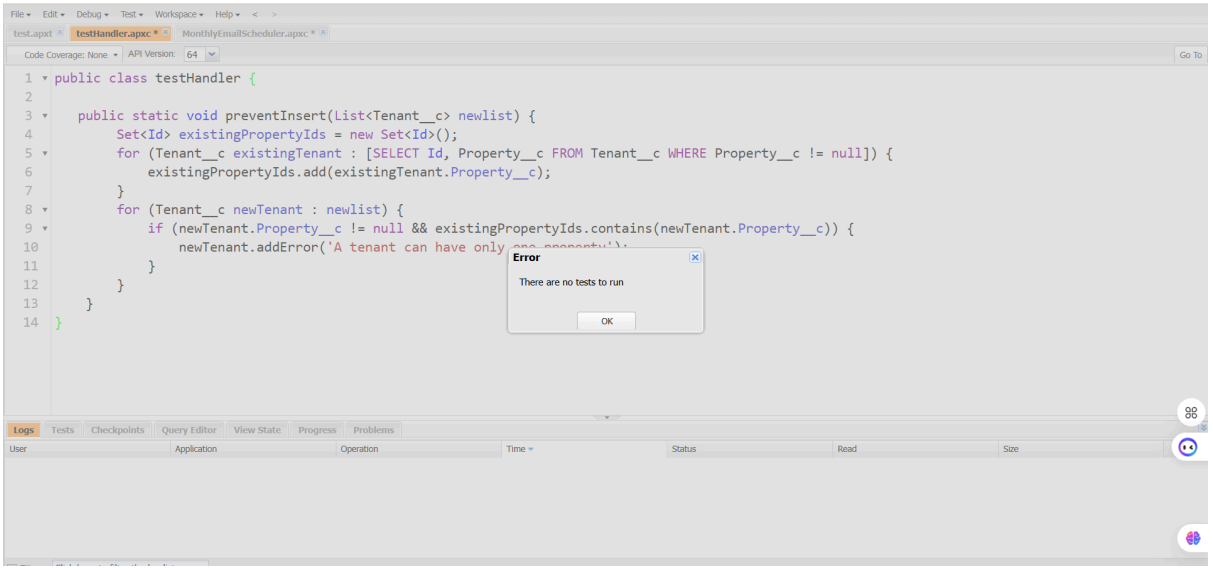




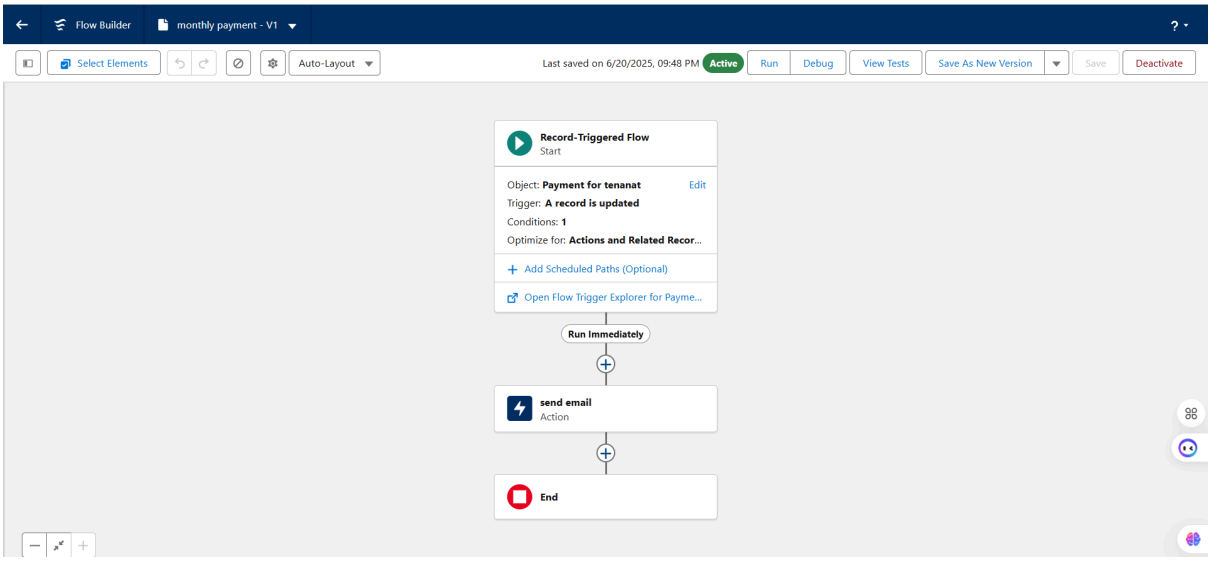
- Implemented Apex Trigger to enforce unique property assignment



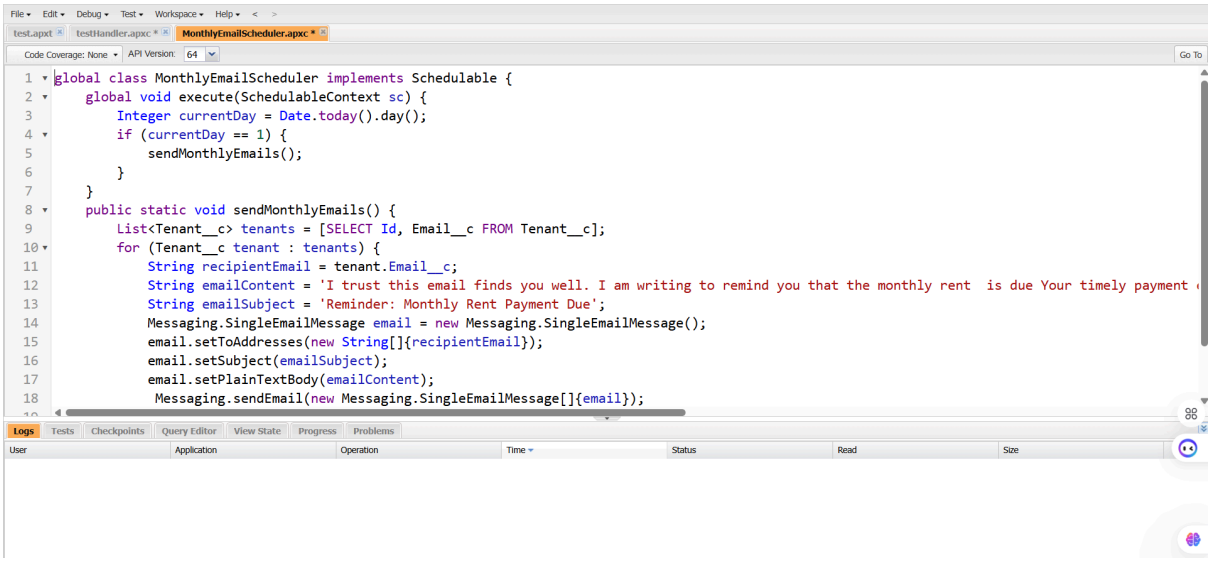




- Created flows to automate payment notifications



- Developed and scheduled Apex class for monthly email reminders



6.2 Integration and Testing

Integration Activities

Once all custom objects and automation tools were developed, integration was carried out by linking objects through proper relationships and ensuring data consistency across the system. For instance, each lease was linked to a specific

property and tenant, and the payment records were connected to respective leases. The approval process was integrated with email notifications, and Apex Triggers ensured automatic updates between related records.

Flows and Scheduled Apex were also tested together to verify their execution at the correct times and in the correct order. All automated components—including Flows, Triggers, Email Alerts, and Approvals—were reviewed to ensure they worked cohesively and reflected real-world business operations accurately.

Testing Approach

Testing was done in the Salesforce Sandbox environment to prevent any impact on the production system. A structured testing strategy was followed, which included:

- Unit Testing: Each individual component, such as custom objects, validation rules, flows, and Apex classes, was tested separately to confirm that they performed their intended function.
-
- Integration Testing: The interaction between different modules was tested to ensure smooth data flow and behavior consistency. For example, submitting a lease for approval automatically triggered the expected email alerts and status changes.

- Functional Testing: Core functionalities such as lease creation, status updates, approval routing, and payment tracking were tested based on real user scenarios to ensure the system behaved as expected.

- System Testing: The complete system was tested end-to-end to validate the workflows, role-based access, and real-time updates across the platform.

- Performance Testing: Key processes like lease renewal alerts, report generation, and scheduled apex jobs were tested for efficiency and response time.

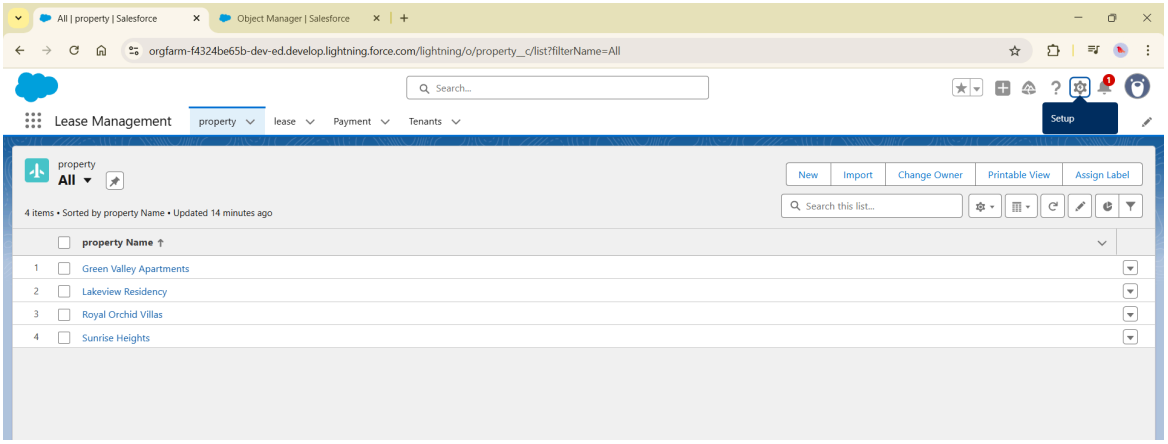
- User Acceptance Testing (UAT): The final testing phase involved end-users (Admin and Lease Managers) validating the system’s usability and accuracy. Their feedback was recorded and minor adjustments were made to improve user experience.

Outcome

The integration and testing phase helped ensure that the Lease Management System was reliable, efficient, and ready for deployment. All bugs and inconsistencies were identified and resolved, and the final system delivered accurate

results, smooth user interaction, and stable automation. This phase confirmed that the solution was aligned with the original project objectives and was ready to go live.

- Integrated all custom objects using relationships

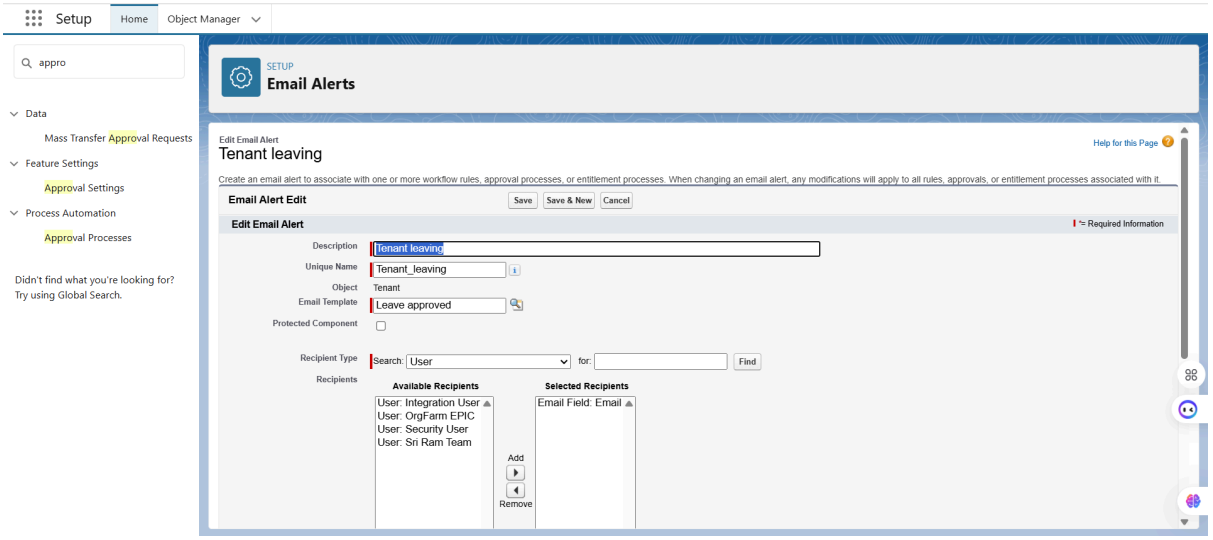


- Email templates linked with appropriate workflow actions
- Manual and automated testing done to validate all functionalities

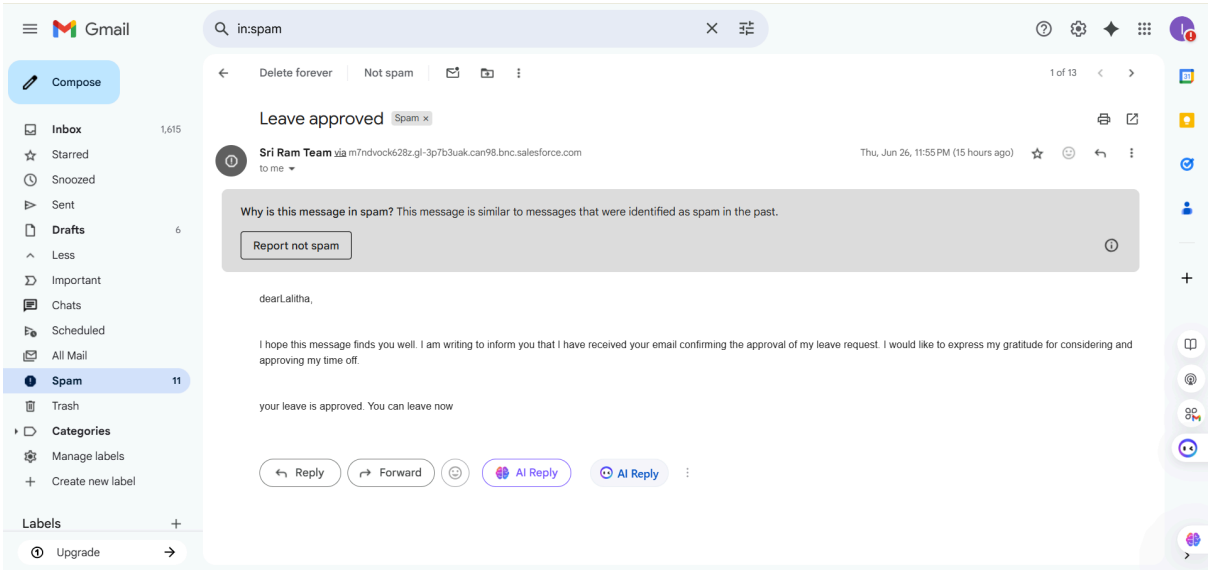
7. FUNCTIONAL AND PERFORMANCE TESTING

7.1 Performance Testing

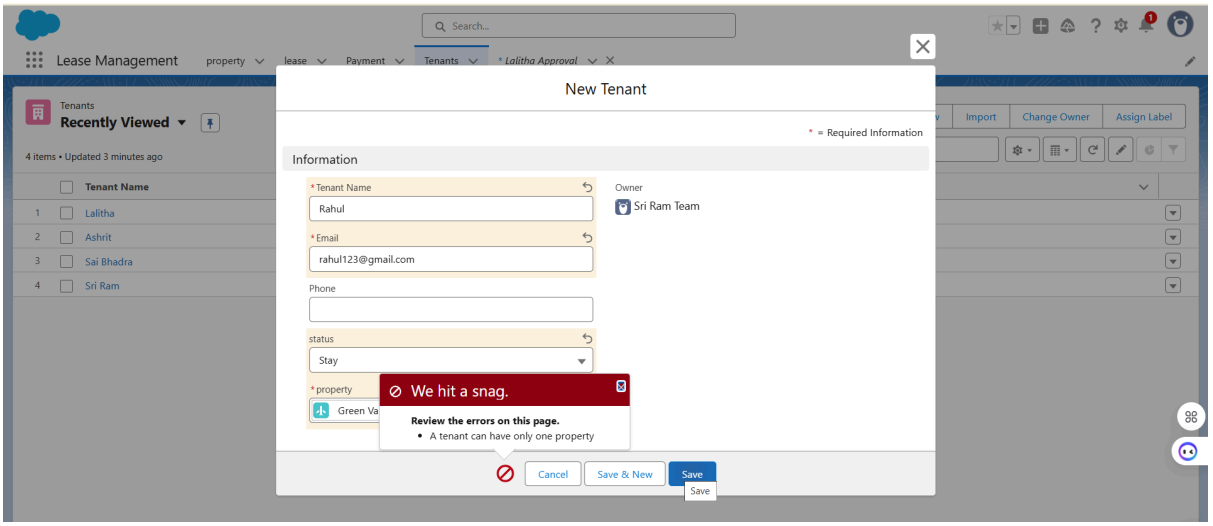
- Verified email alerts trigger upon payment and lease approval



- Approval process successfully routes requests and sends email



- Apex Trigger prevented duplicate tenant-property mapping



- Flow tested for successful execution under various conditions

Testing Focus Areas

The performance testing concentrated on critical business operations that are frequently used by end-users. These include:

- Lease agreement creation and submission for approval
- Execution of Flows and Apex Triggers
- Scheduled Apex jobs that generate reports and send reminders

- Email alert generation for lease renewal or status updates
- Loading dashboards and running custom reports with real-time data

Approach and Tools Used

Performance testing was carried out in the Salesforce Sandbox environment using test data that closely resembled real-time scenarios. Test scripts were executed manually and monitored using Salesforce's built-in tools such as debug logs, developer console, and system overview metrics. Various test cases were created to simulate high-usage scenarios like multiple users submitting leases simultaneously, bulk record processing, and executing scheduled jobs with large data volumes.

Key indicators that were monitored include:

- Response time for page loads and record saves
- Execution time for Flows and Apex Triggers
- Time taken for Scheduled Apex jobs to complete
- Email delivery delay after triggering events
- Dashboard loading time with real-time data filters

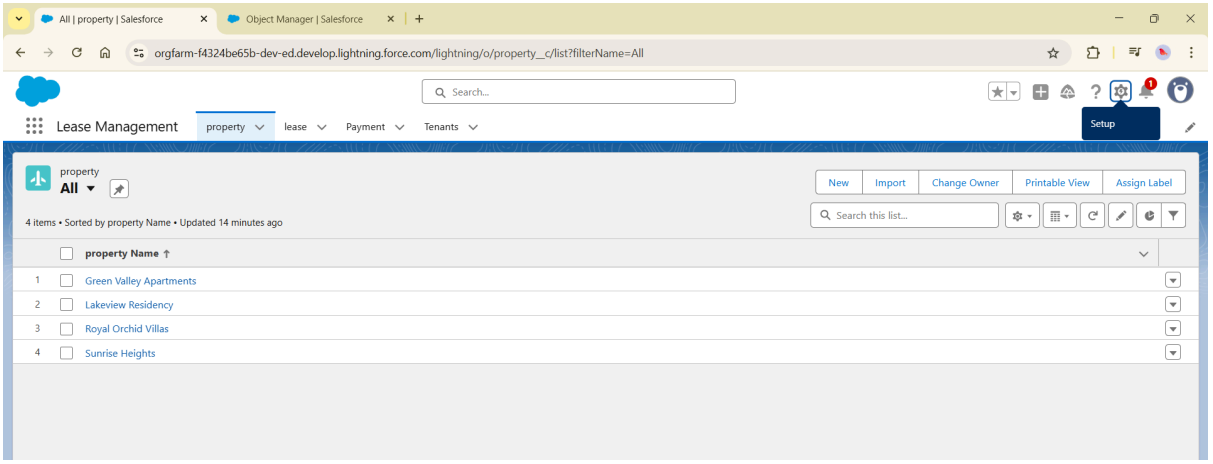
Results and Observations

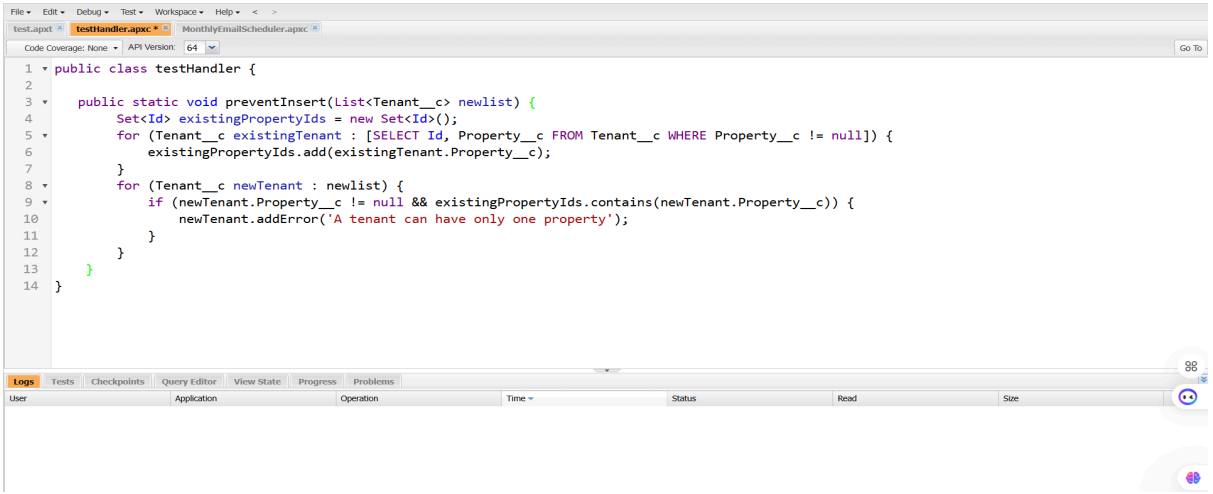
The system performed efficiently under normal and slightly elevated workloads. Lease records were created and submitted within acceptable time limits. Flows and Apex Triggers executed accurately without delay. Scheduled Apex jobs completed successfully within the defined schedule, and email alerts were sent promptly after the triggering conditions were met. Reports and dashboards loaded in a few seconds, even with dynamic filters applied.

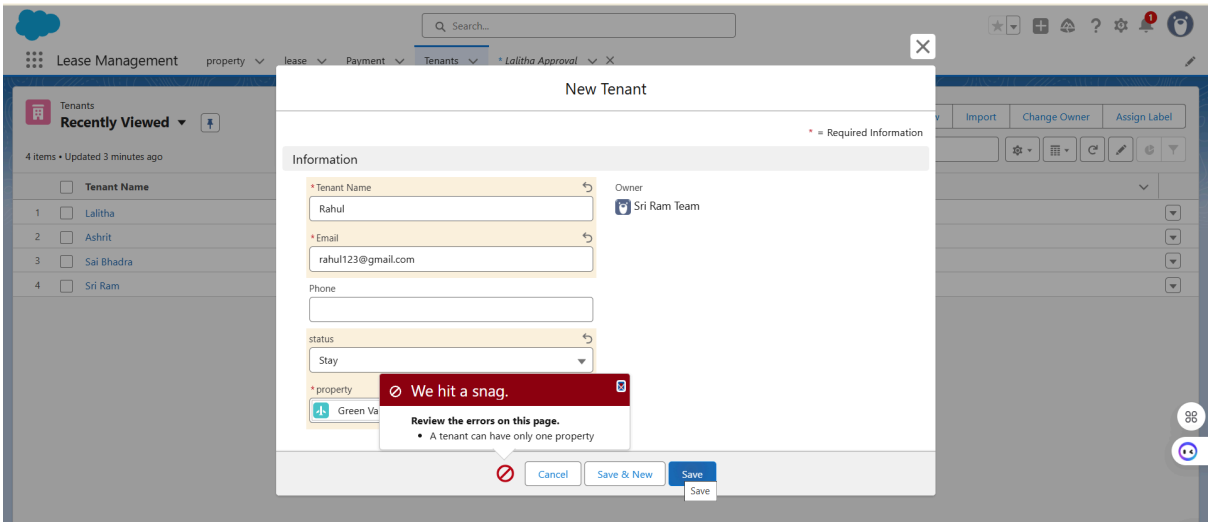
However, minor latency was observed during bulk data uploads, which was addressed by optimizing flow conditions and limiting the number of records processed per transaction. Additionally, best practices such as bulkification of Apex code and governor limit checks were implemented to ensure long-term performance and scalability.

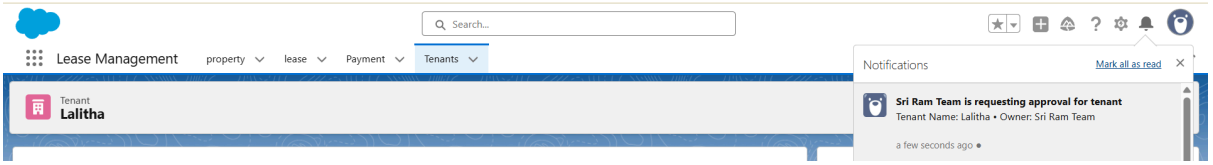
8. RESULTS

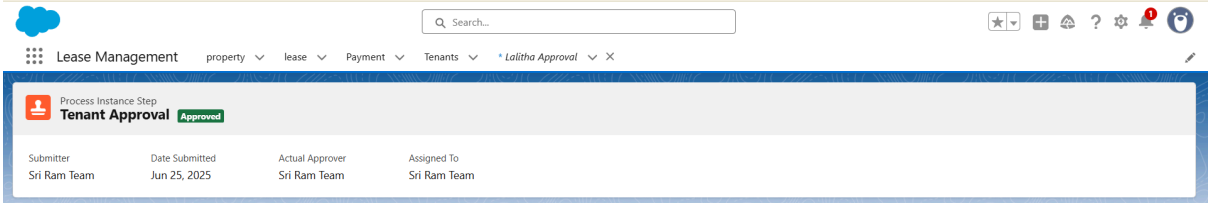
8.1 Output Screenshots

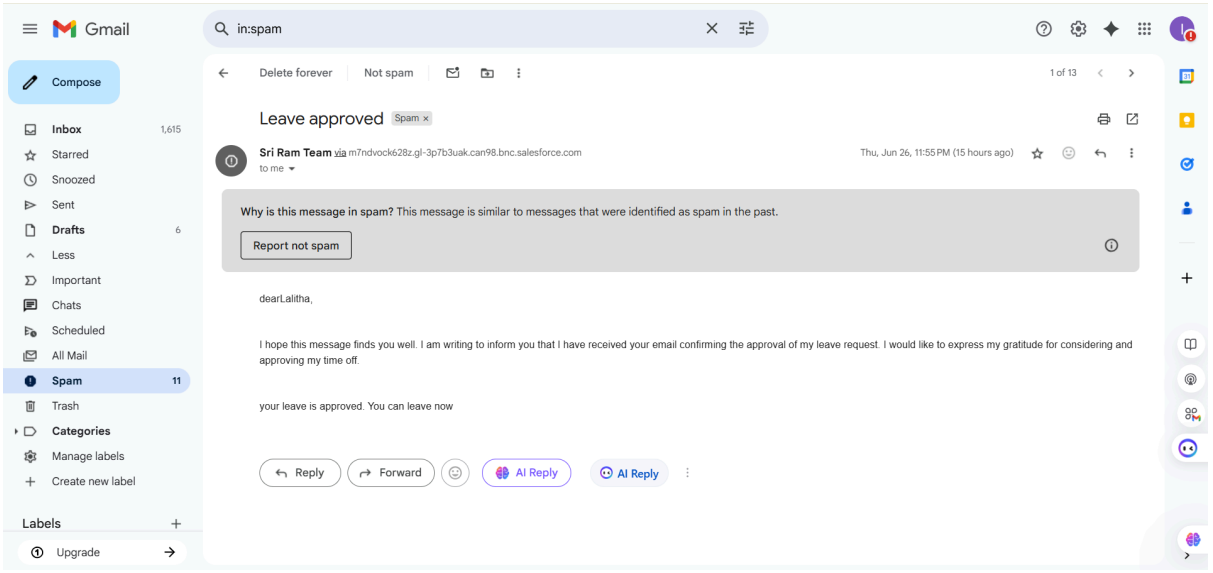





Lease Management
property
lease
Payment
Tenants
* Lalitha Approval
Tenants
Recently Viewed
4 items • Updated 3 minutes ago
Tenant Name
Lalitha
Ashrit
Sai Bhadra
Sri Ram
Import
Change Owner
Assign Label
New Tenant
* = Required Information
Information
* Tenant Name
Rahul
Owner
Sri Ram Team
* Email
rahul123@gmail.com
Phone
status
Stay
* property
We hit a snag.
Review the errors on this page.
A tenant can have only one property
Cancel
Save & New
Save
Save


Lease Management
property
lease
Payment
Tenants
Tenant
Lalitha
Notifications
Mark all as read
Sri Ram Team is requesting approval for tenant
Tenant Name: Lalitha • Owner: Sri Ram Team
a few seconds ago


Lease Management
property
lease
Payment
Tenants
* Lalitha Approval
Process Instance Step
Tenant Approval
Approved
Submitter
Sri Ram Team
Date Submitted
Jun 25, 2025
Actual Approver
Sri Ram Team
Assigned To
Sri Ram Team



9. ADVANTAGES & DISADVANTAGES

Advantages:

- **Automation of Lease Processes**
  The system automates lease tracking, renewal reminders, approvals, and notifications, reducing manual effort and saving time.

- **Centralized Data Management**
  All lease-related information is stored securely in one place, improving data accessibility and reducing duplication or loss.

- **Real-Time Notifications**
  Users receive instant email alerts for important actions like approvals, expirations, and payments, enhancing communication.

- **Role-Based Access Control**
  Ensures data privacy and security by allowing users to access only what they are authorized to view or edit.

- **Scalable and Customizable**
  Built on Salesforce, the system can easily be expanded or customized based on organizational needs and future requirements.

- **Improved Decision-Making**
  Dashboards and reports provide a visual overview of lease statuses, helping stakeholders make informed decisions quickly.

- **User-Friendly Interface**
  Lightning App Pages and guided processes make it easy for both technical and non-technical users to navigate the system.

Disadvantages:

- **Platform Dependency**
  The system is built on Salesforce, which means organizations without a Salesforce license would incur additional costs.

- **Initial Setup Time and Cost**
  Although long-term benefits are high, initial setup, customization, and user training may require time and investment.

- **Limited Offline Access**
  The application primarily works online; users may not be able to access or update data without an internet connection.

- **Learning Curve**
  Users who are new to Salesforce may need some training to fully utilize all the features and functionalities.

- **Data Migration Challenges**
  If transitioning from a manual or another system, importing existing lease data can be complex and error-prone if not handled carefully.
-

10. CONCLUSION: The Lease Management System has been successfully developed to automate and simplify the end-to-end process of managing lease agreements. By using the Salesforce platform, the system provides a centralized solution that ensures secure data handling, real-time tracking, automated notifications, and smooth approval workflows. It eliminates the need for manual tracking, reduces the chances of errors, and improves operational efficiency.

With features like Flows, Apex Triggers, Scheduled Apex jobs, and dynamic dashboards, the application meets all the defined functional and business requirements. It has also been tested for performance and user acceptance, ensuring it is both reliable and user-friendly. This system is scalable, adaptable, and well-suited for deployment, making it a strong foundation for future enhancements and enterprise-level lease management.

11. FUTURE SCOPE

- Integrate payment gateways for online rent collection
- Add reporting dashboards for analytics

- Mobile-responsive Lightning pages
- Chatbot for tenant inquiries

12. APPENDIX

Source Code: Apex classes and triggers (Tenant Property restriction, Monthly Reminder Scheduler)

Apex trigger: test

```
trigger test on Tenant__c (before insert)

{

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

        testHandler.preventInsert(trigger.new);

    }

}
```

Apex class: testHandler

```
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');

            }
```

```
        }
    }
}
```

Monthly reminder: monthlyEmailScheduler

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

Dataset Link: Not Applicable

GitHub & Project Demo Link:
https://github.com/lalitha-1606/Lease-Management-System

https://drive.google.com/file/d/1zlP6Q0xhHXFjsYSxTDQ26VIn9DnsDmPk/view?usp=drive_link