LEASE MANAGEMENT

**COLLEGE NAME**: Sri Ramakrishna College Of Arts And Science For Women

**COLLEGE CODE**: bru28 **TEAM ID**: NM2025TMID24172

**TEAM MEMBERS:**

**TEAM LEADER** : Priyavarshini S

**EMAIL ID** : [srcw2322k139@srcw.ac.in](mailto:srcw2322k112@srcw.ac.in)

**TEAM MEMBER1** : Nagalakshmi S

**EMAIL ID** : [srcw2322k132@srcw.ac.in](mailto:srcw2322k110@srcw.ac.in)

**TEAM MEMBER2** : Miruthika N

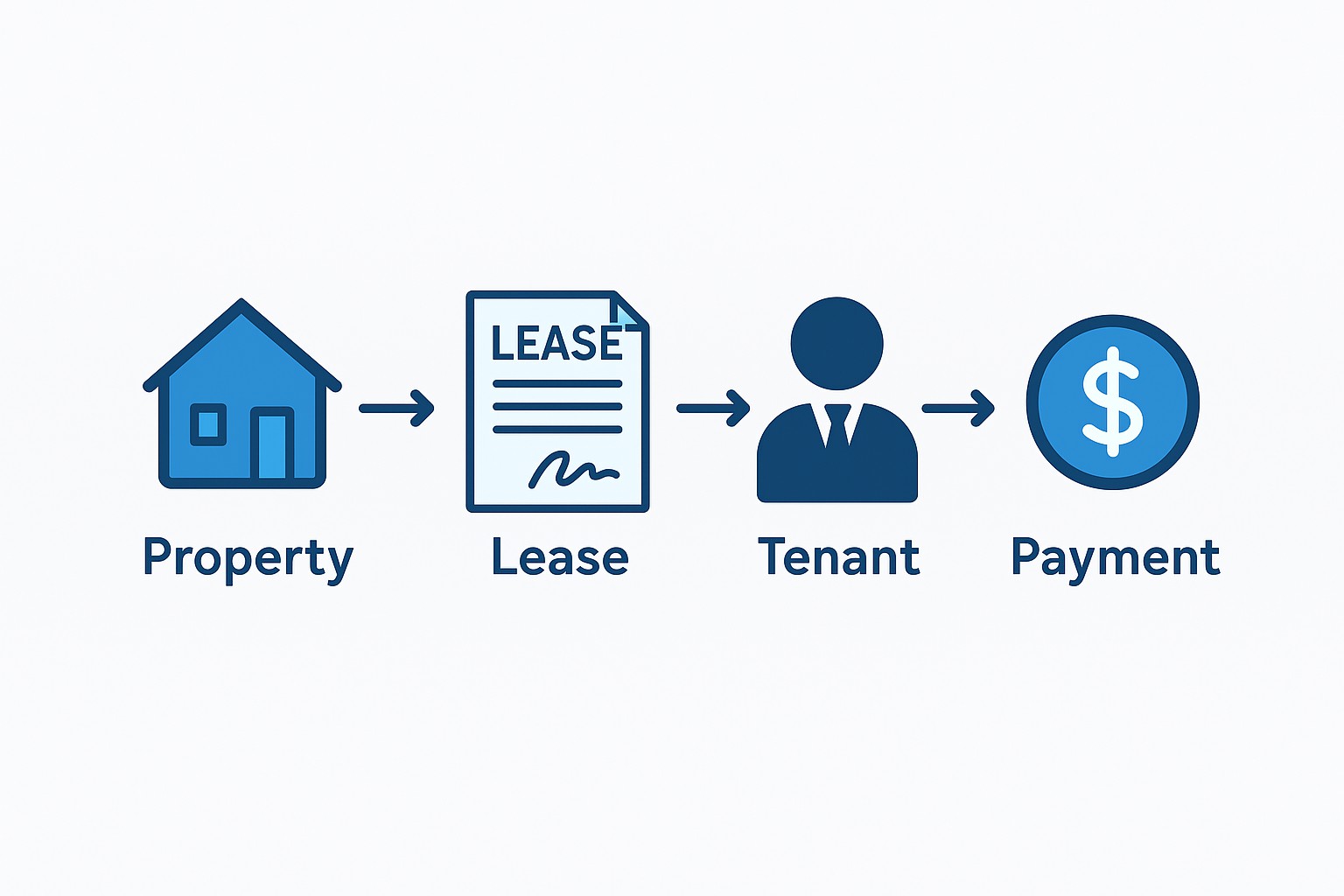
**EMAIL ID** : [srcw2322k129@srcw.ac.in](mailto:srcw2322k110@srcw.ac.in)

**TEAM MEMBER3** : Mohana P

**EMAIL ID** : [srcw2322k130@srcw.ac.in](mailto:srcw2322k110@srcw.ac.in)

**Introduction**

Lease Manage/ment is the process of handling agreements between a property owner (landlord) and a tenant. It involves maintaining property records, managing tenant details, tracking lease start and end dates, ensuring timely rent payments, and automating approvals. In Salesforce, lease management can be simplified using custom objects (Property, Tenant, Lease, Payment), workflows, approval processes, triggers, and email notifications.



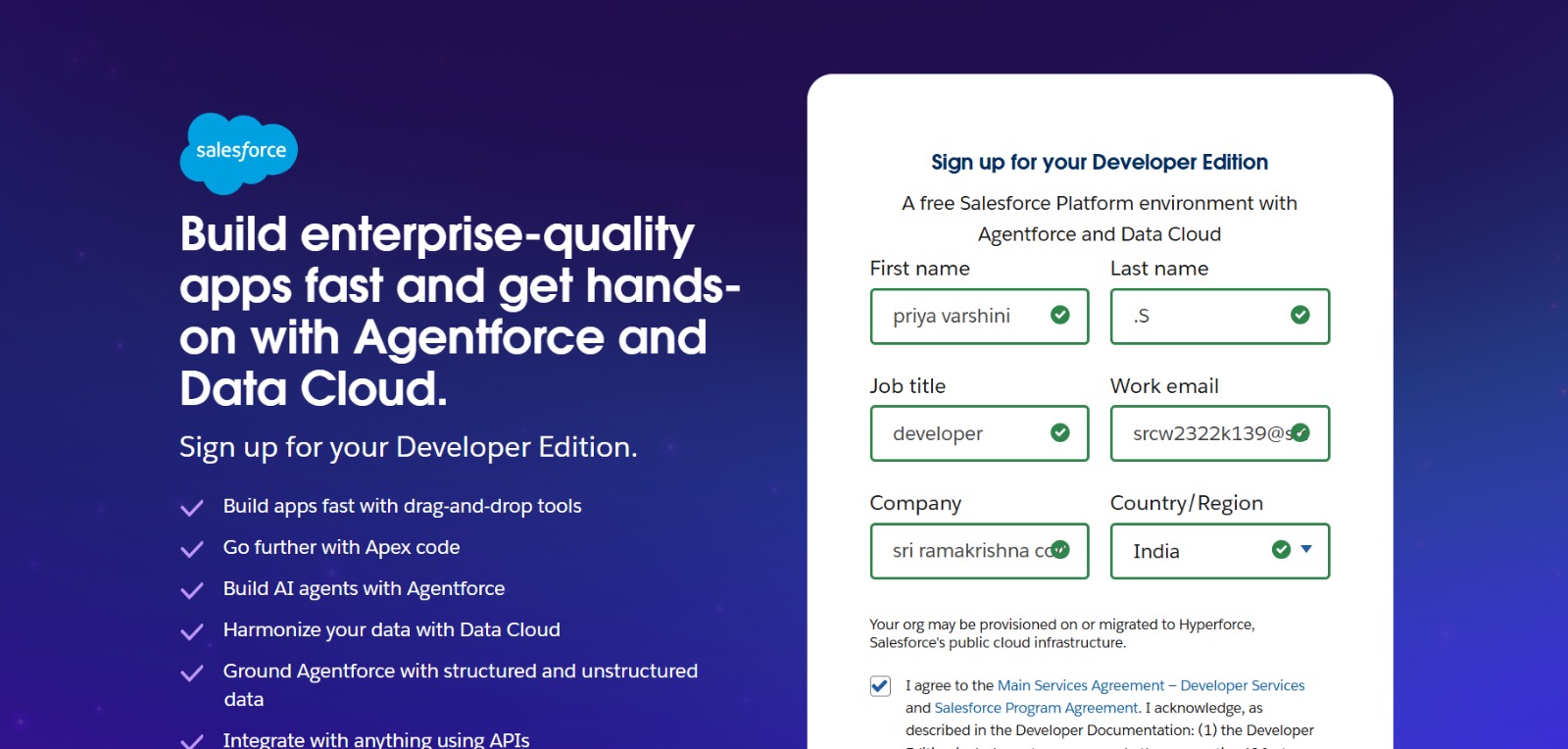
**Purpose**

* To centralize property, tenant, and lease information.
* To automate approval of lease requests and tenant leaving requests.
* To ensure compliance by validating lease dates (e.g., End Date > Start Date).
* To send timely notifications for approvals, rejections, and payments.
* To track monthly rent payments with reminders and receipts.
* To improve transparency between landlords, tenants, and administrators.

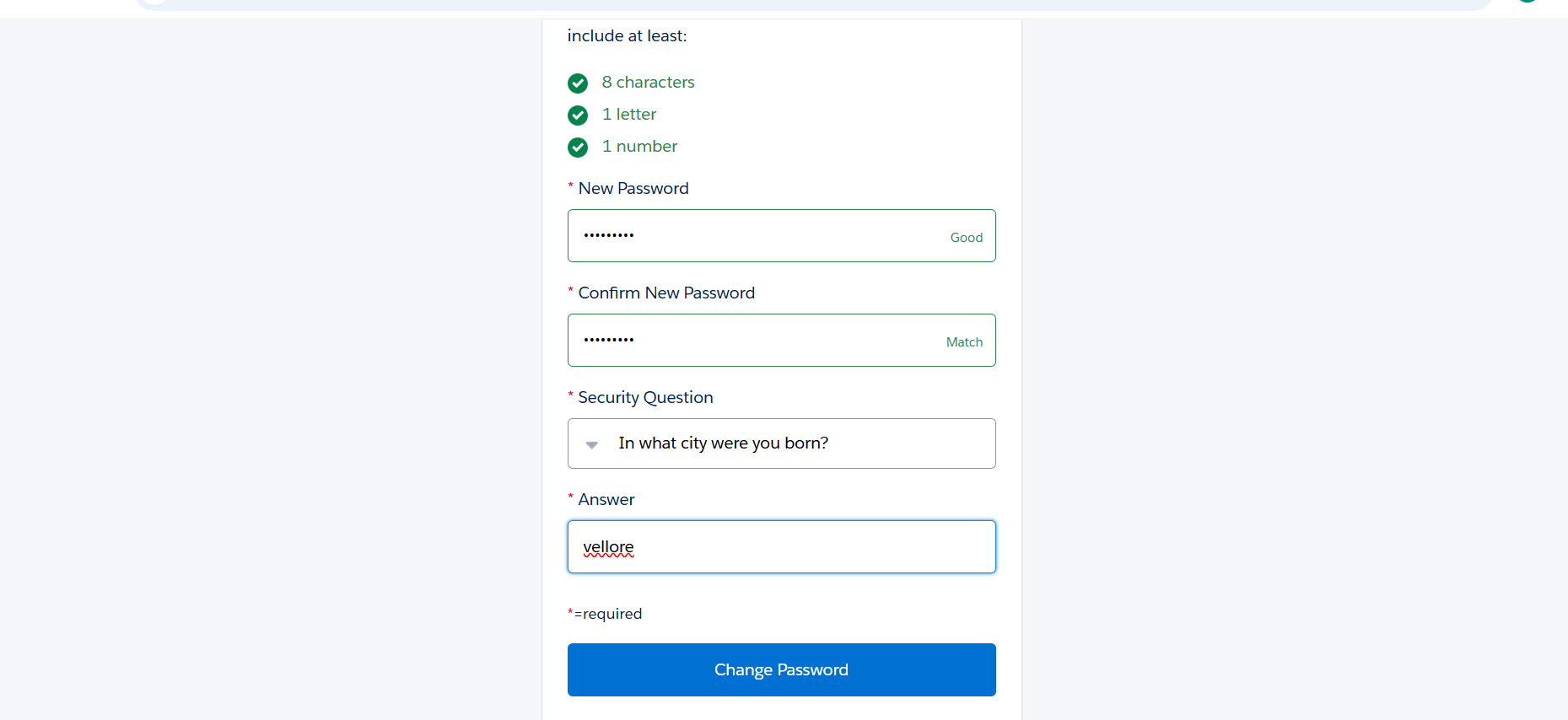
**DEVELOPMENT PHASE:**

The development phase of the Lease Management system was carried out in 11 structured milestones to ensure smooth progress and systematic completion. It began with creating the custom objects (Property, Tenant, Lease, and Payment) followed by setting up their tabs and a Lightning App to bring them together. Next, fields and lookup relationships were added to establish data connections, and validation rules like ensuring the end date is greater than the start date were implemented for accuracy. In the following milestones, email templates were designed for tenant leaving, approvals, rejections, and payments. Two approval processes were built to manage tenant requests and vacancy checks. The phase also included writing Apex triggers and handlers with test classes to automate critical actions, and flows for handling monthly payments. Finally, a scheduled class was developed for recurring tasks, completing the 11 milestones of development. Each step contributed to building a robust and automated lease management system.

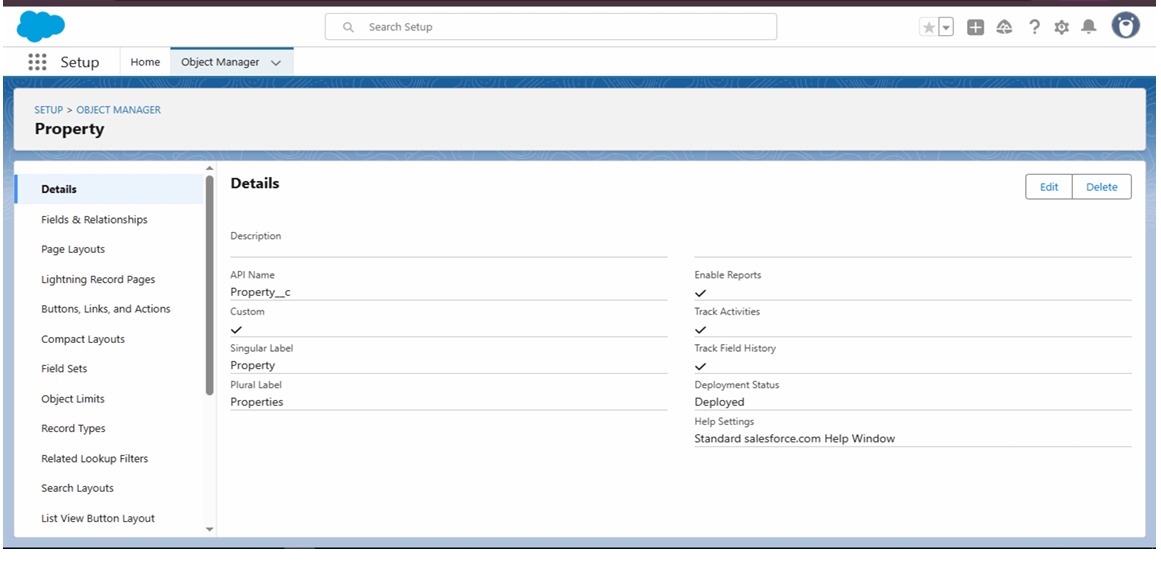
**Creating Developer Account:**

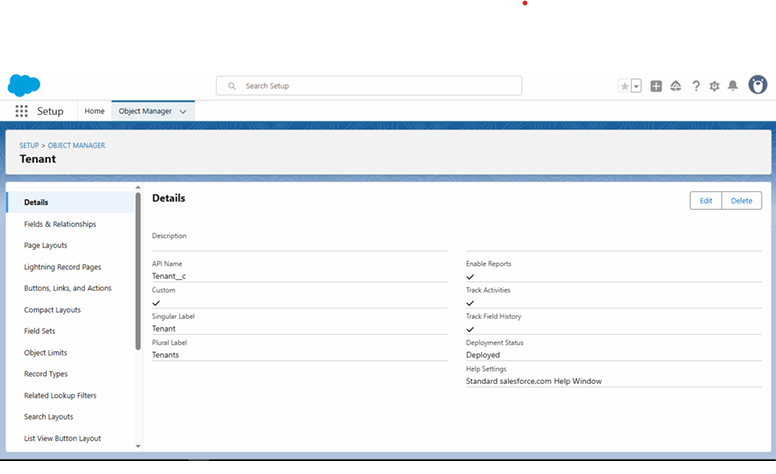


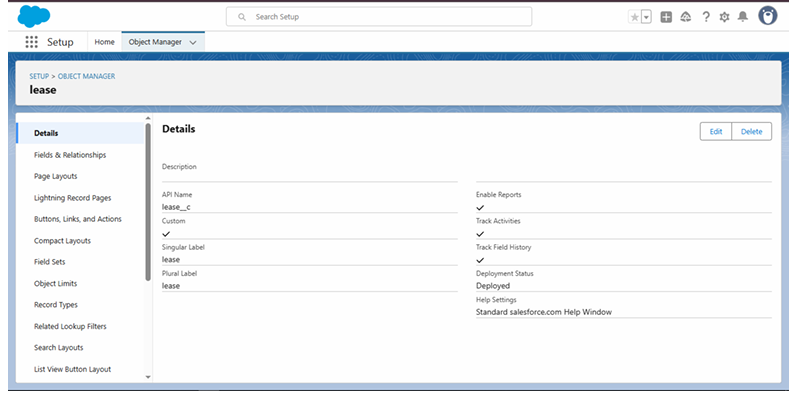
**Account Activation:**

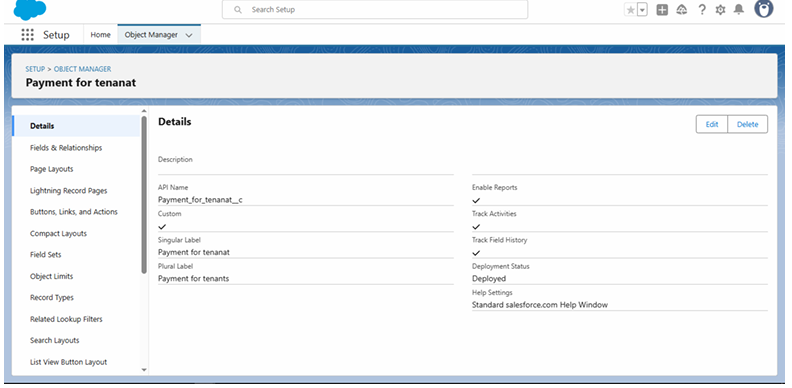
****

**OBJECT CREATION:**

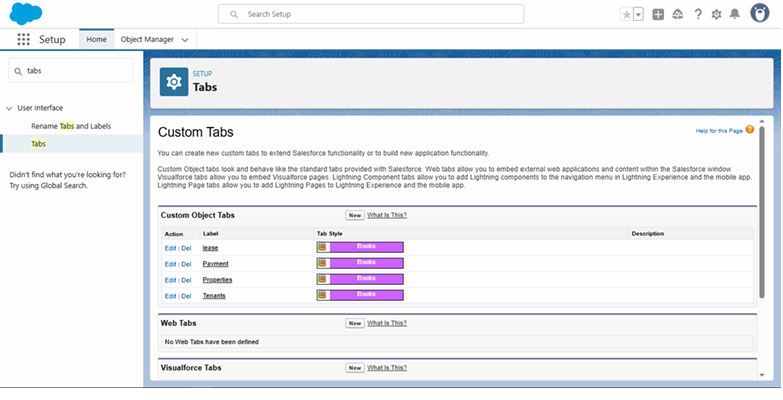
****

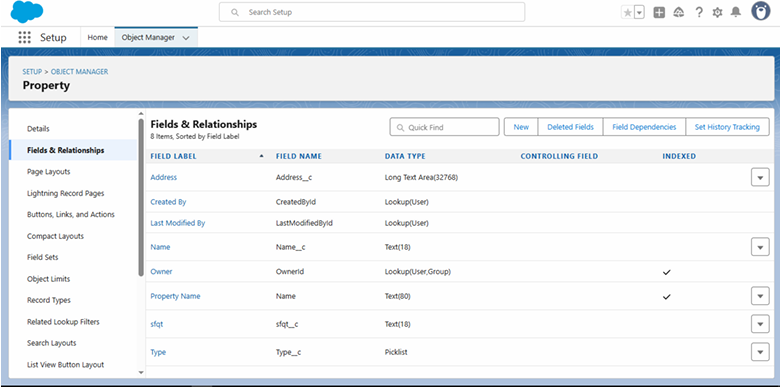
****

****

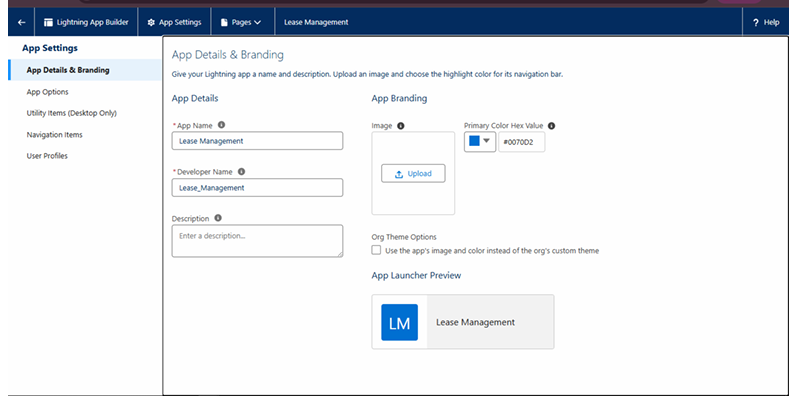
****

**TABS:**

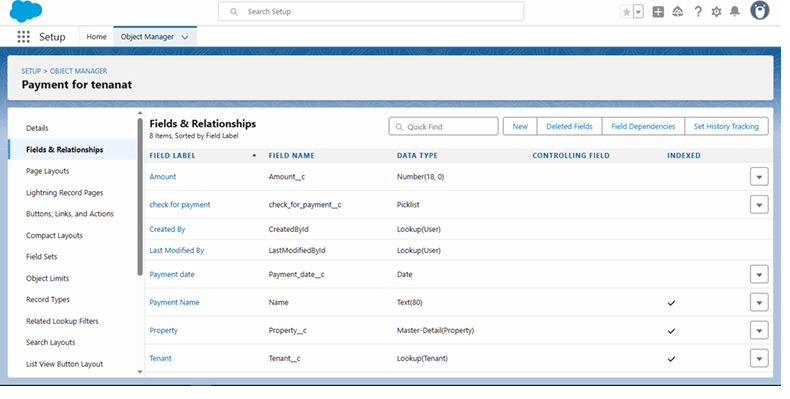
****

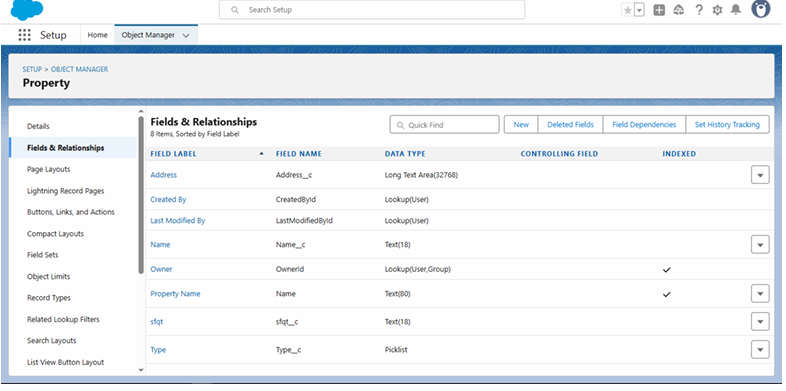
****

**THE LIGHTNING APP:**

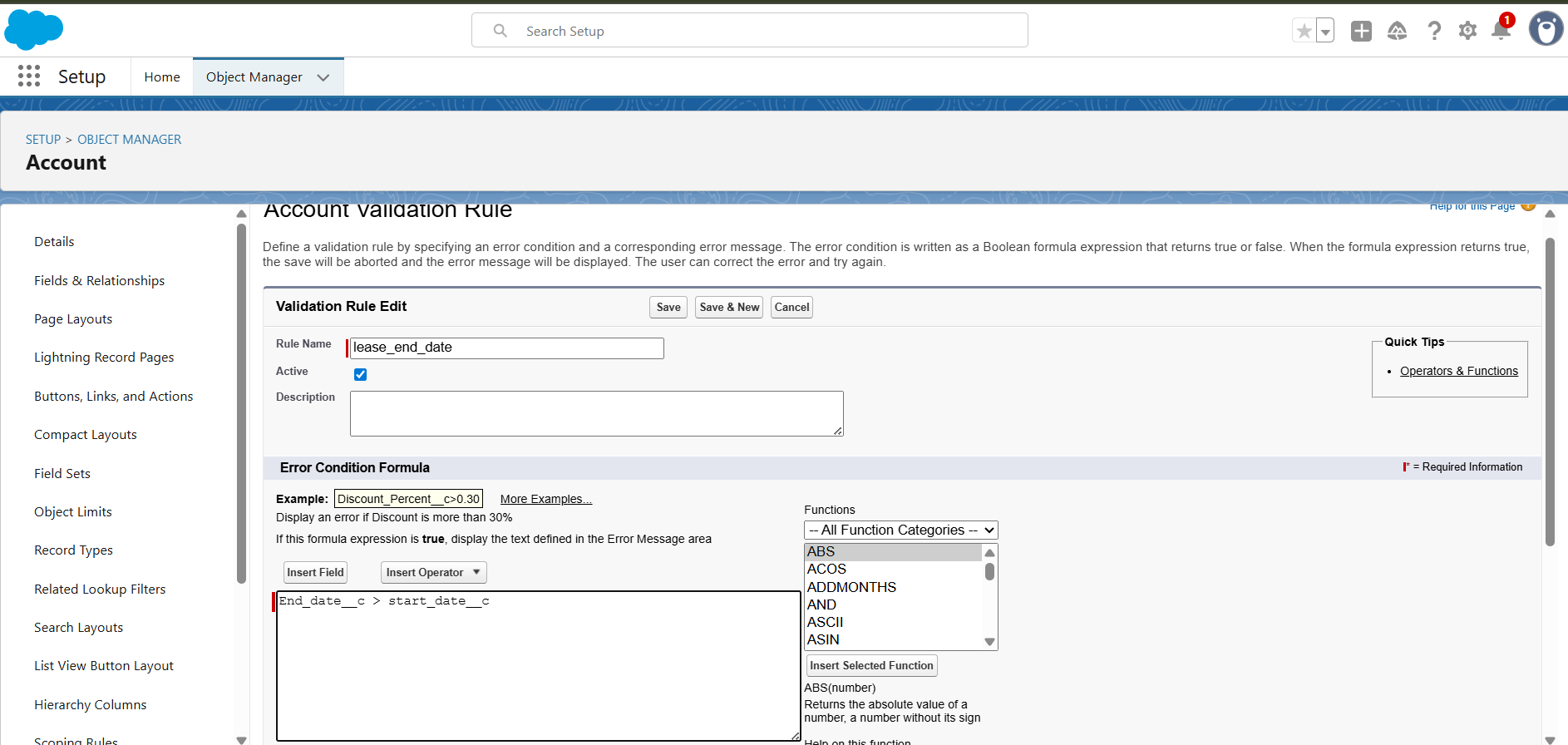
****

**FIELDS:**

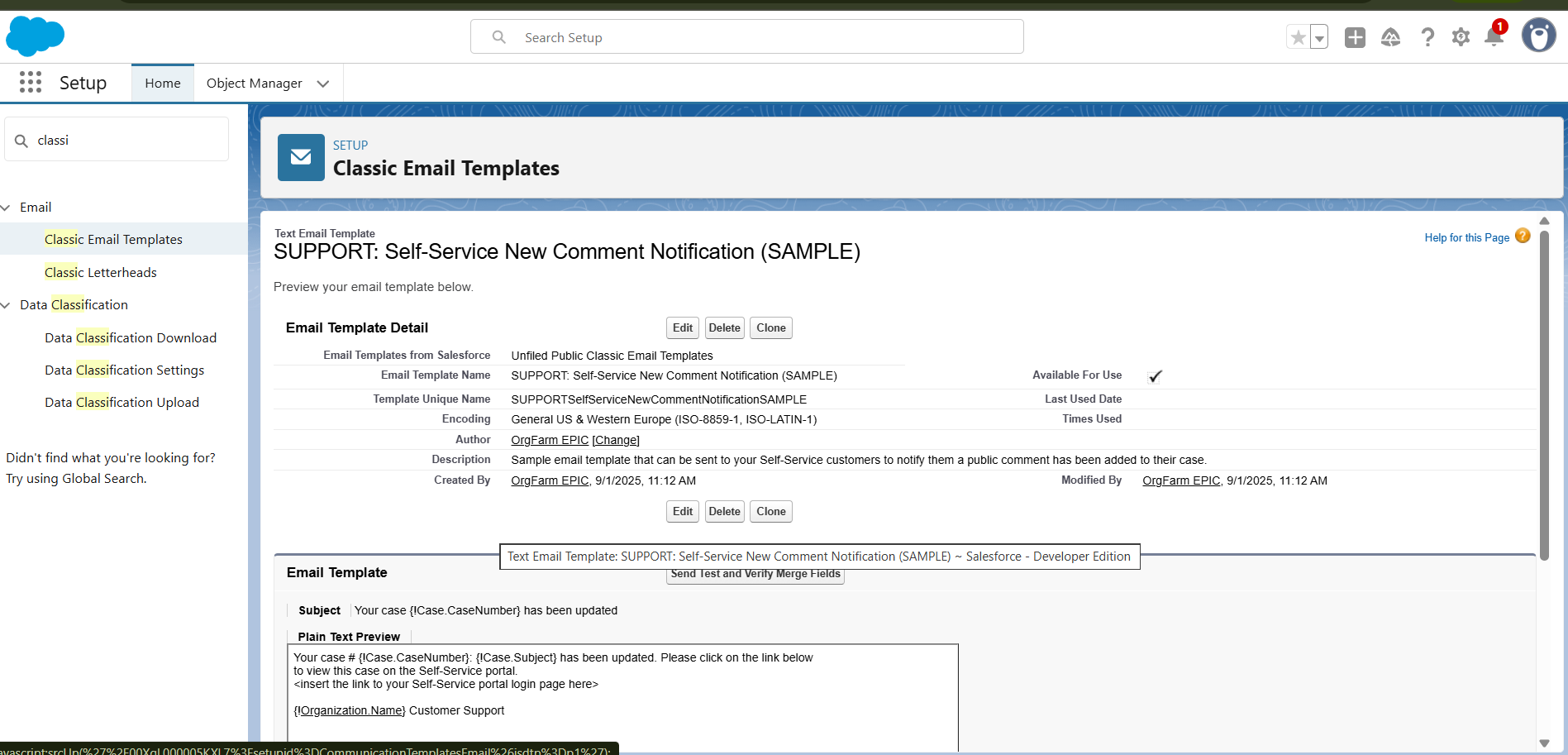
****

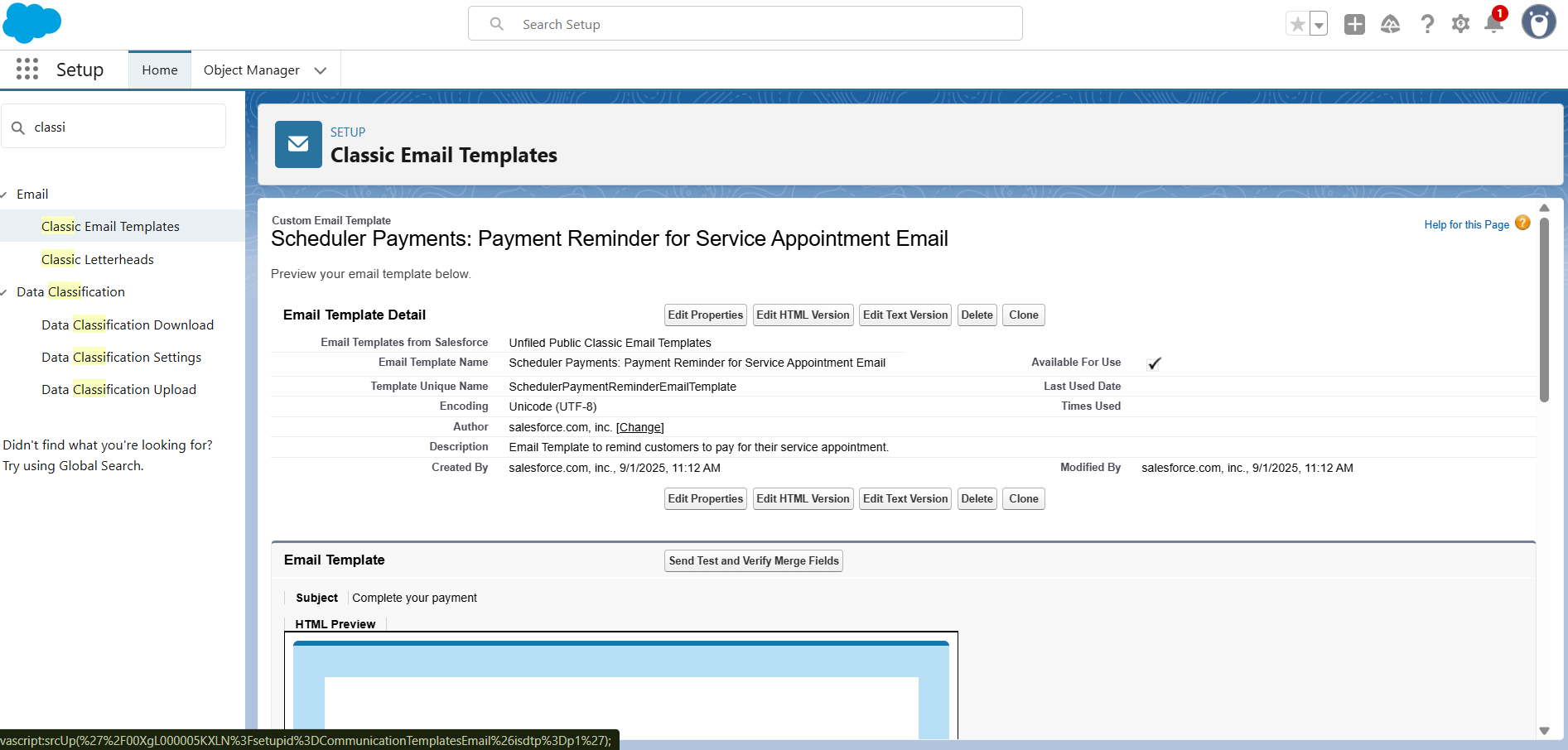


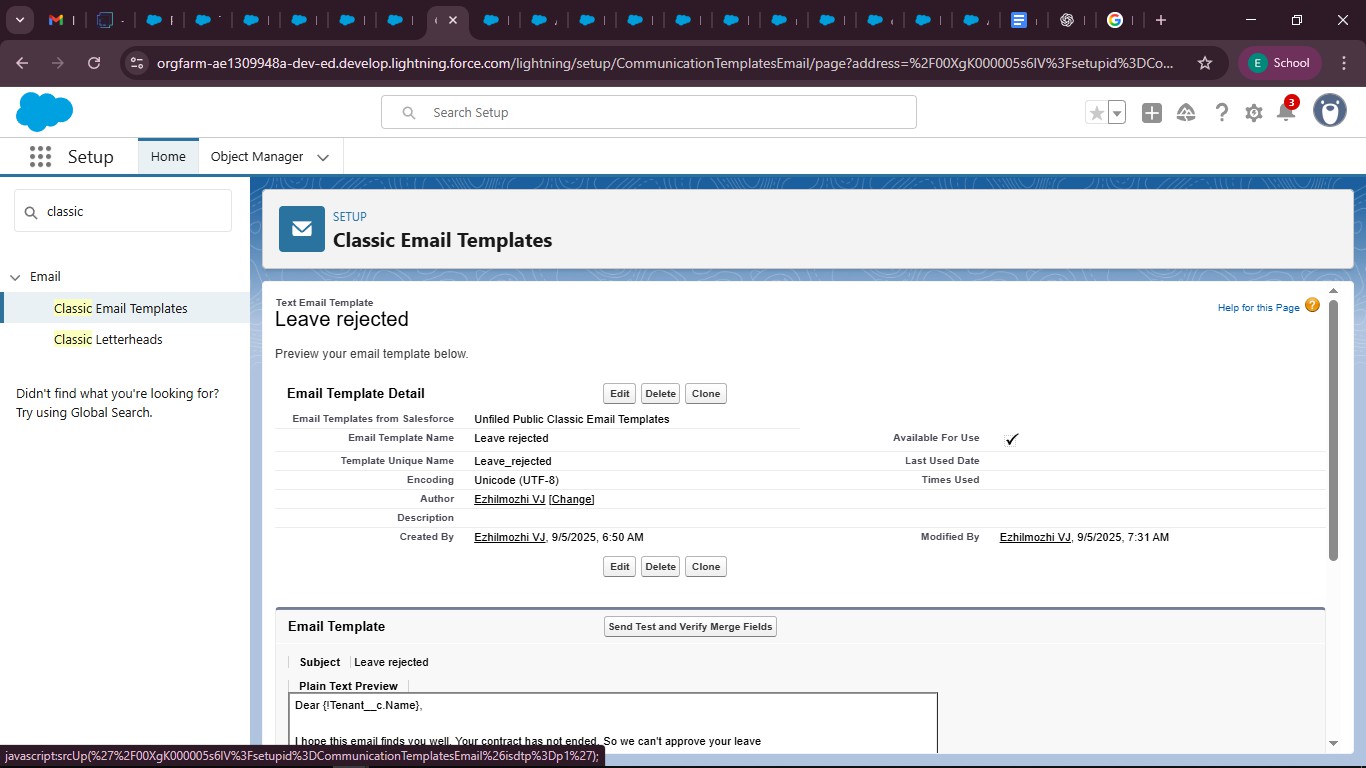
**VALIDATION RULE**:

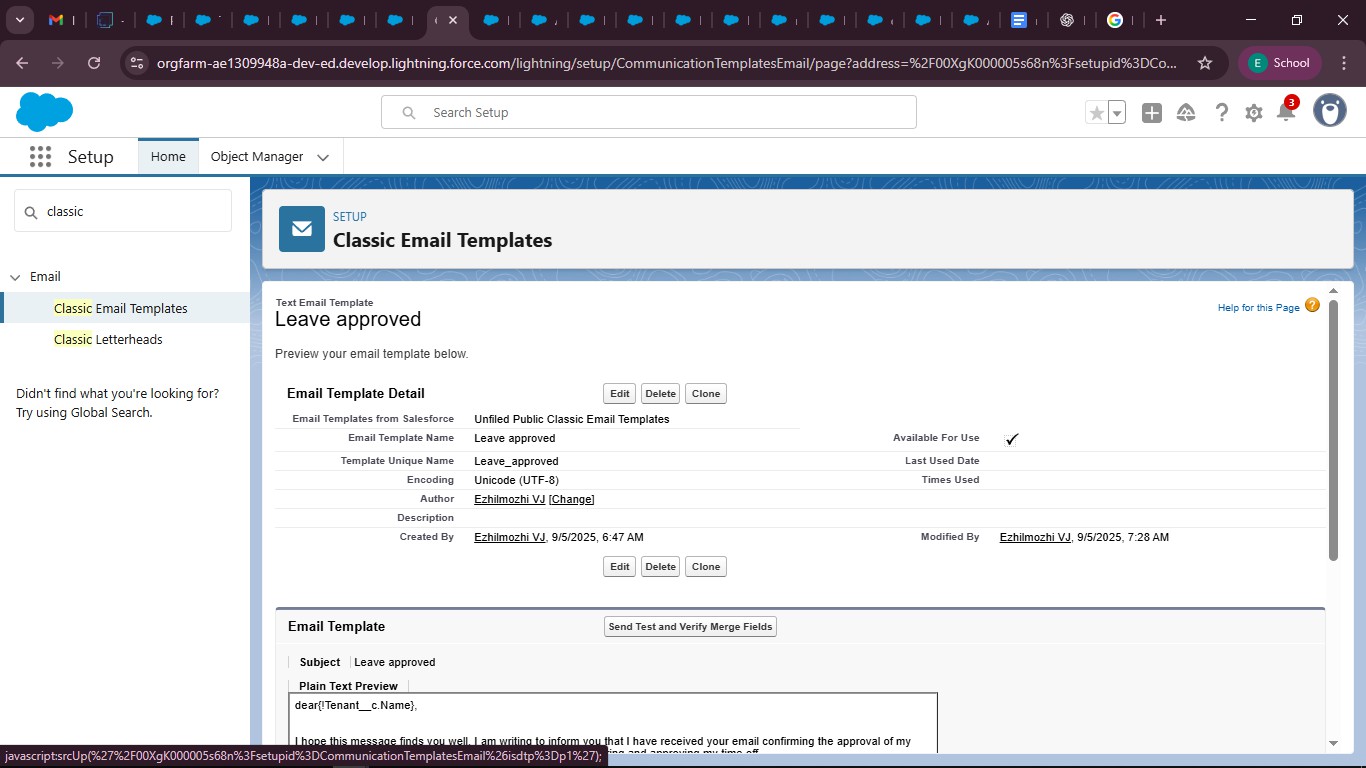


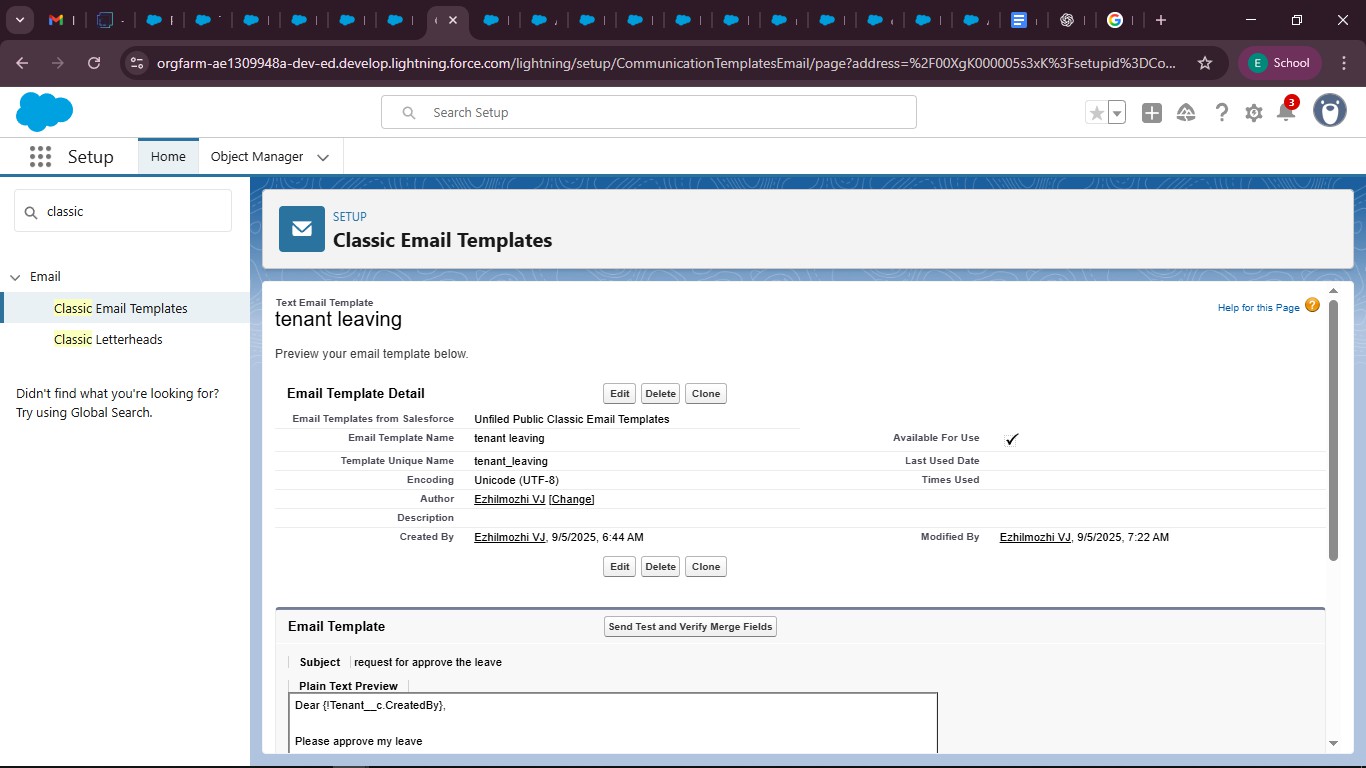
**EMAIL TEMPLATES**:





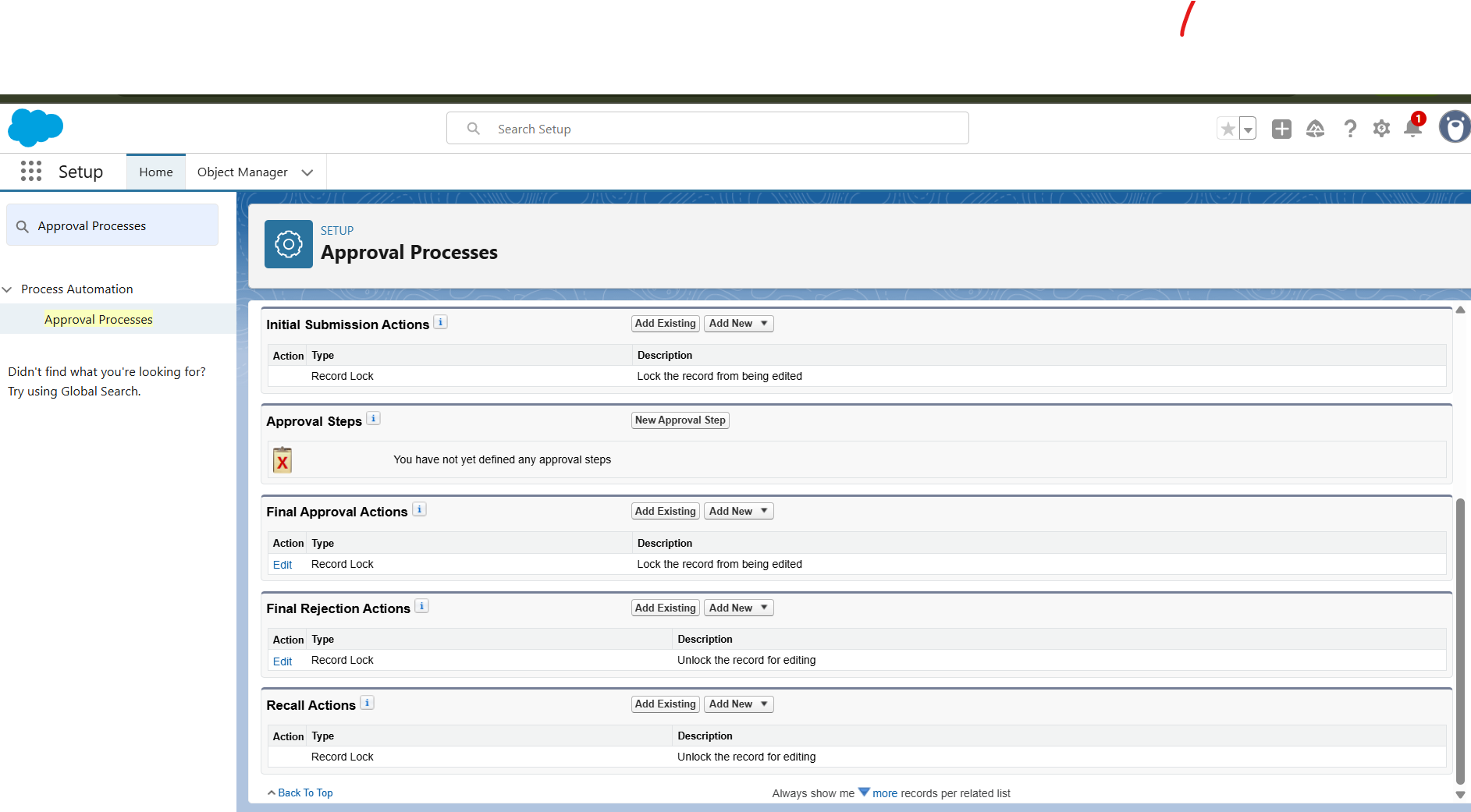




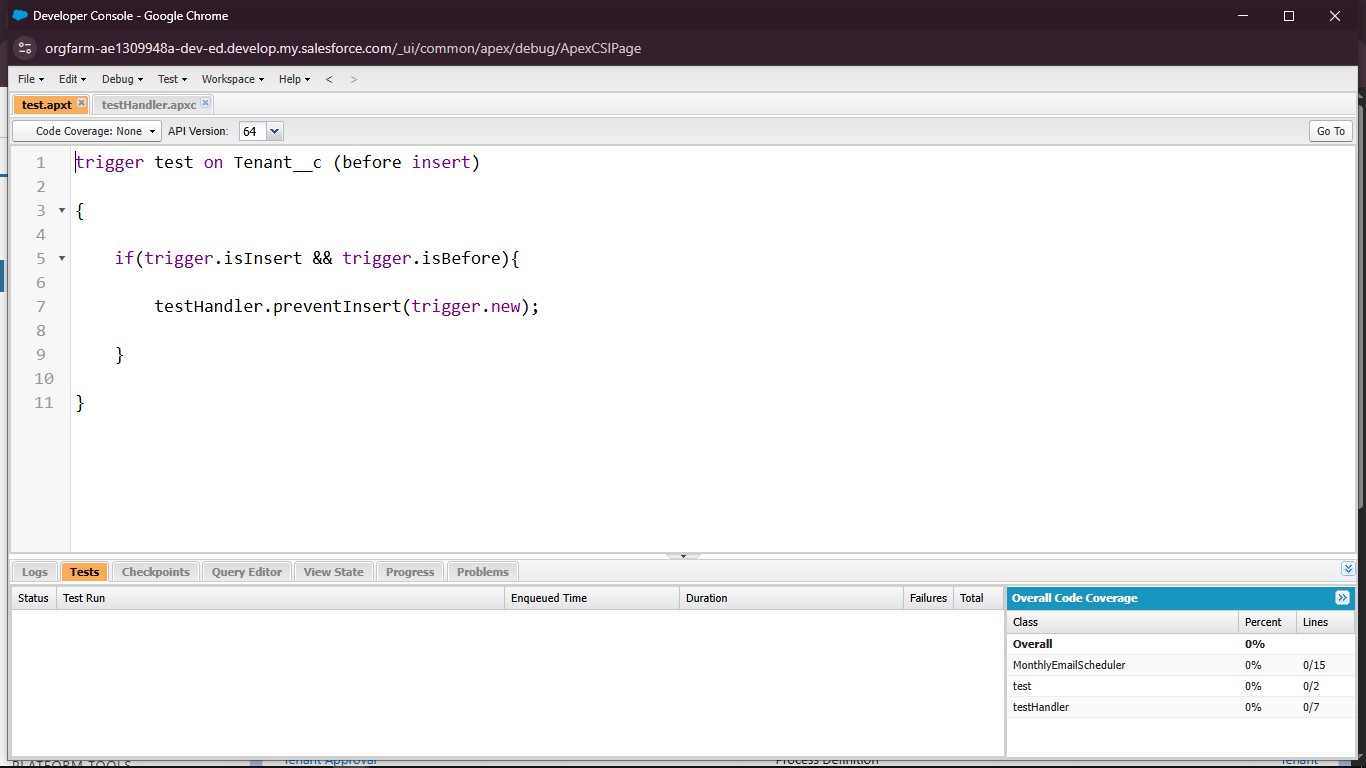


**APPROVAL PROCESS:**

****

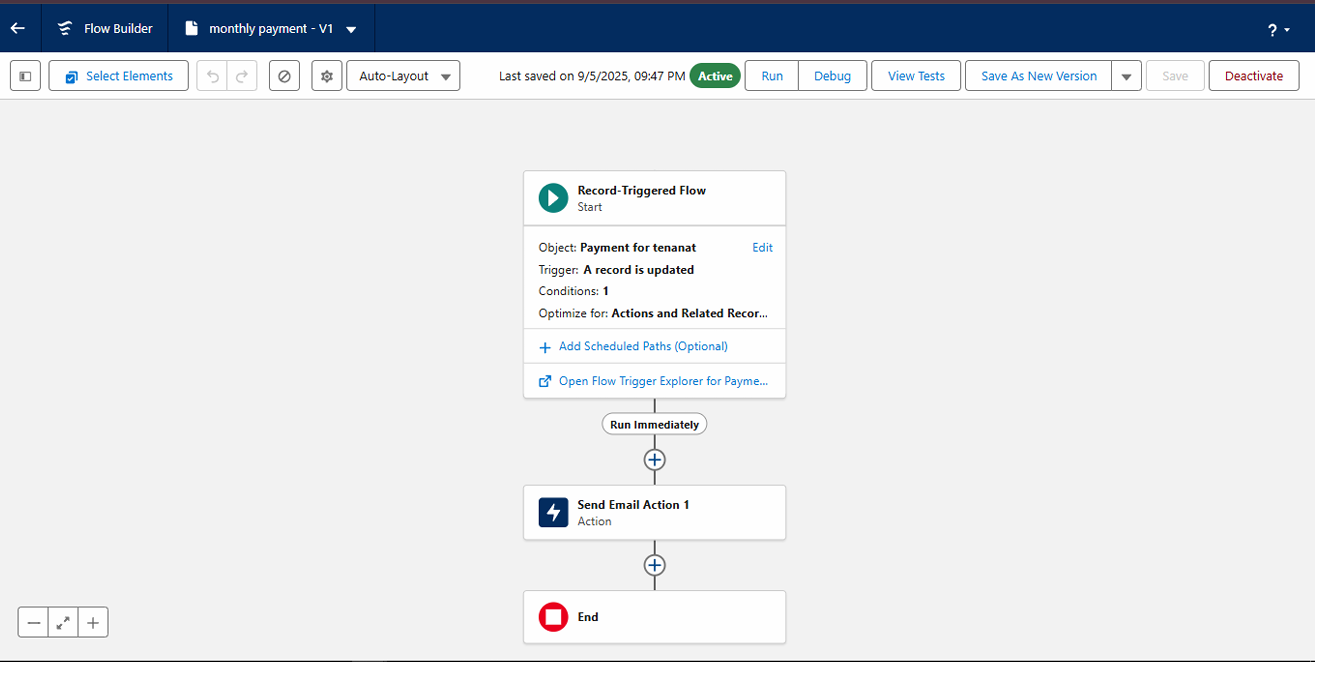


**APEX TRIGGER:**

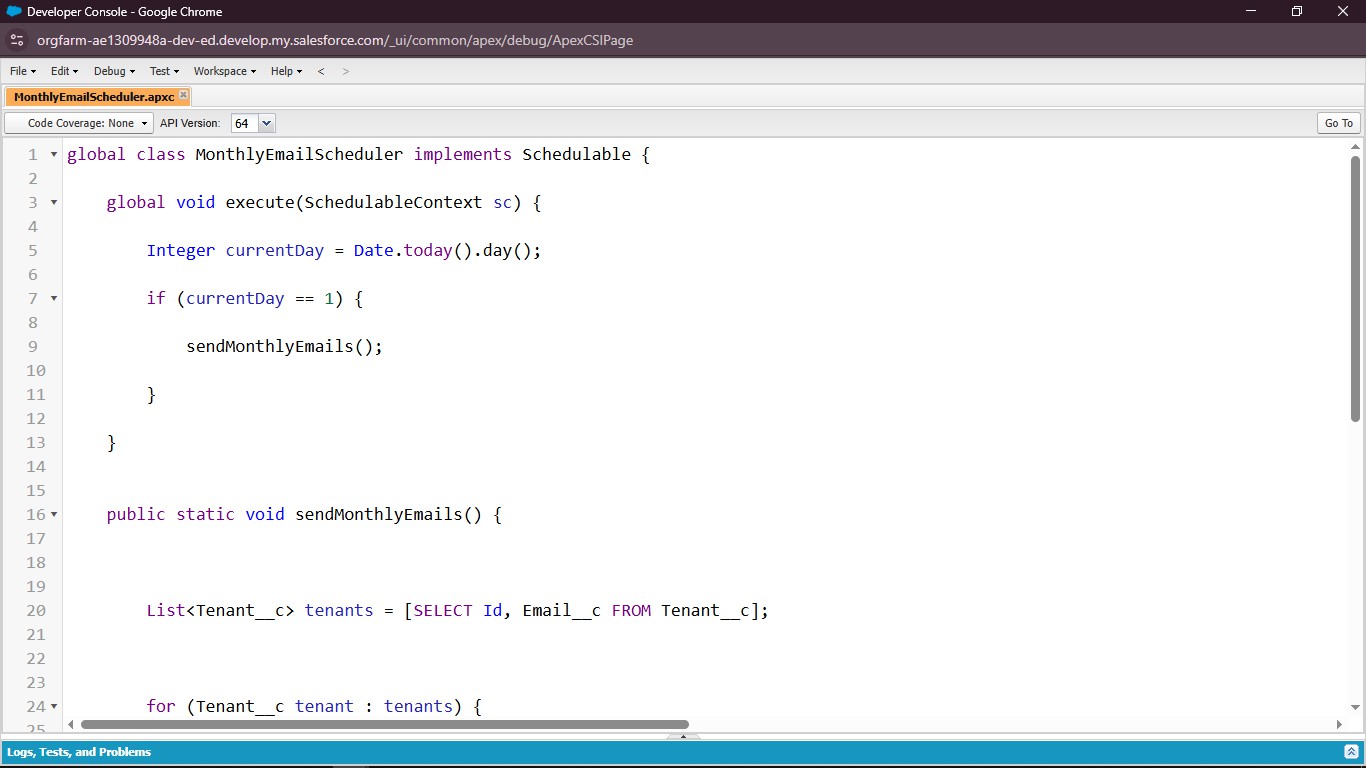
****

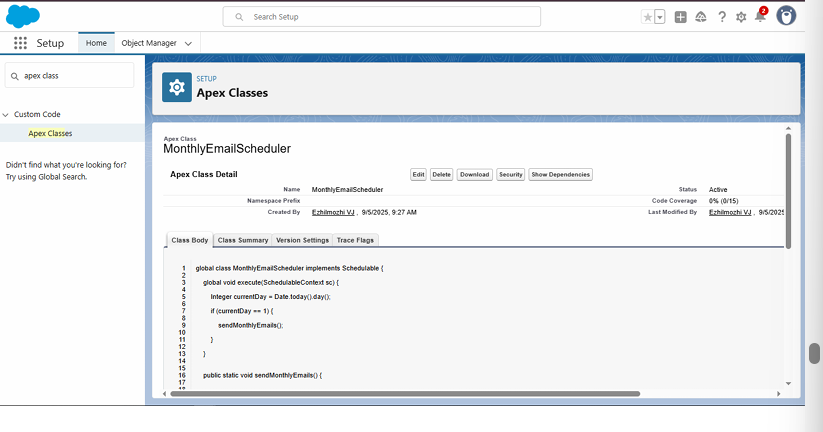


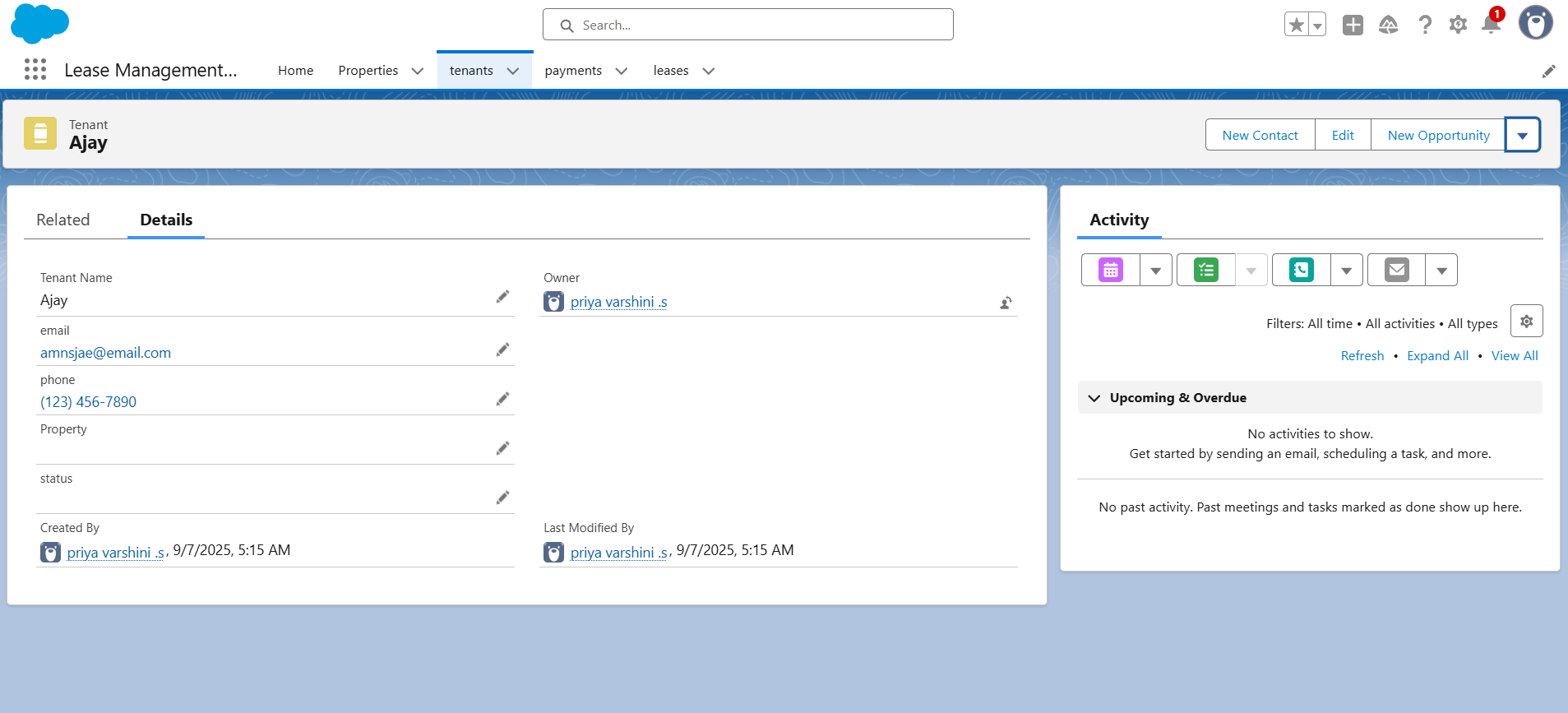
**FLOWS:**

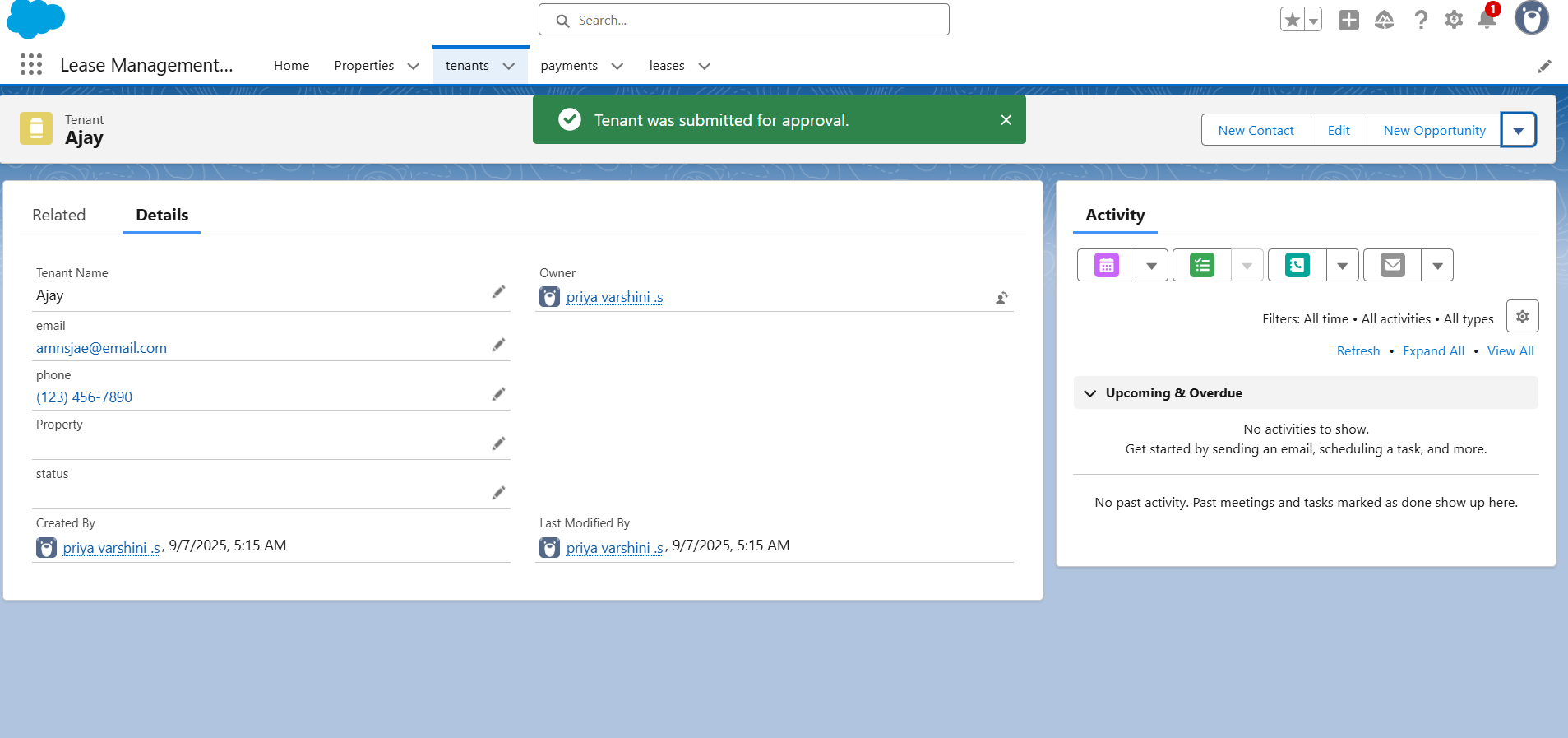
****

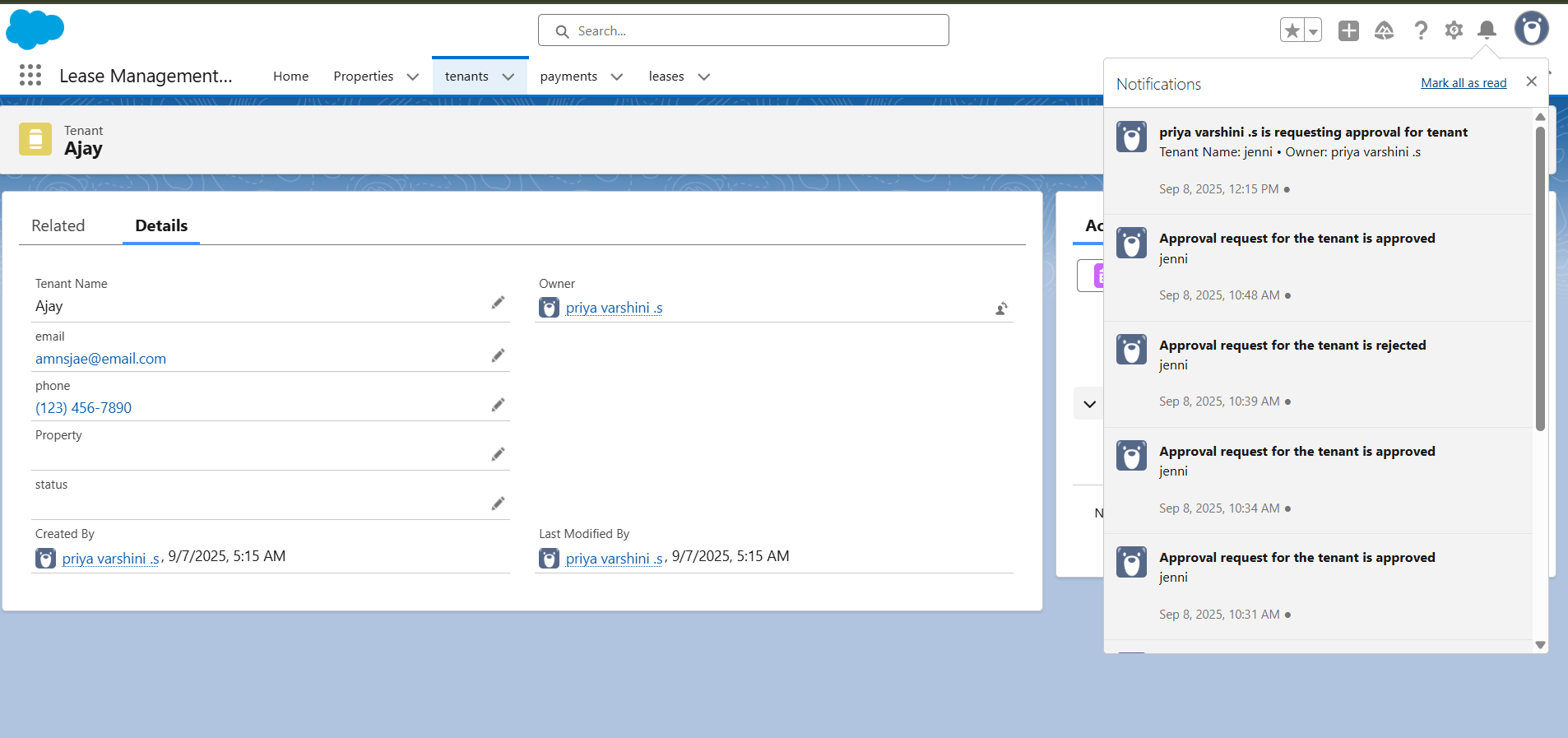
**SCHEDULE CLASSES:**

****



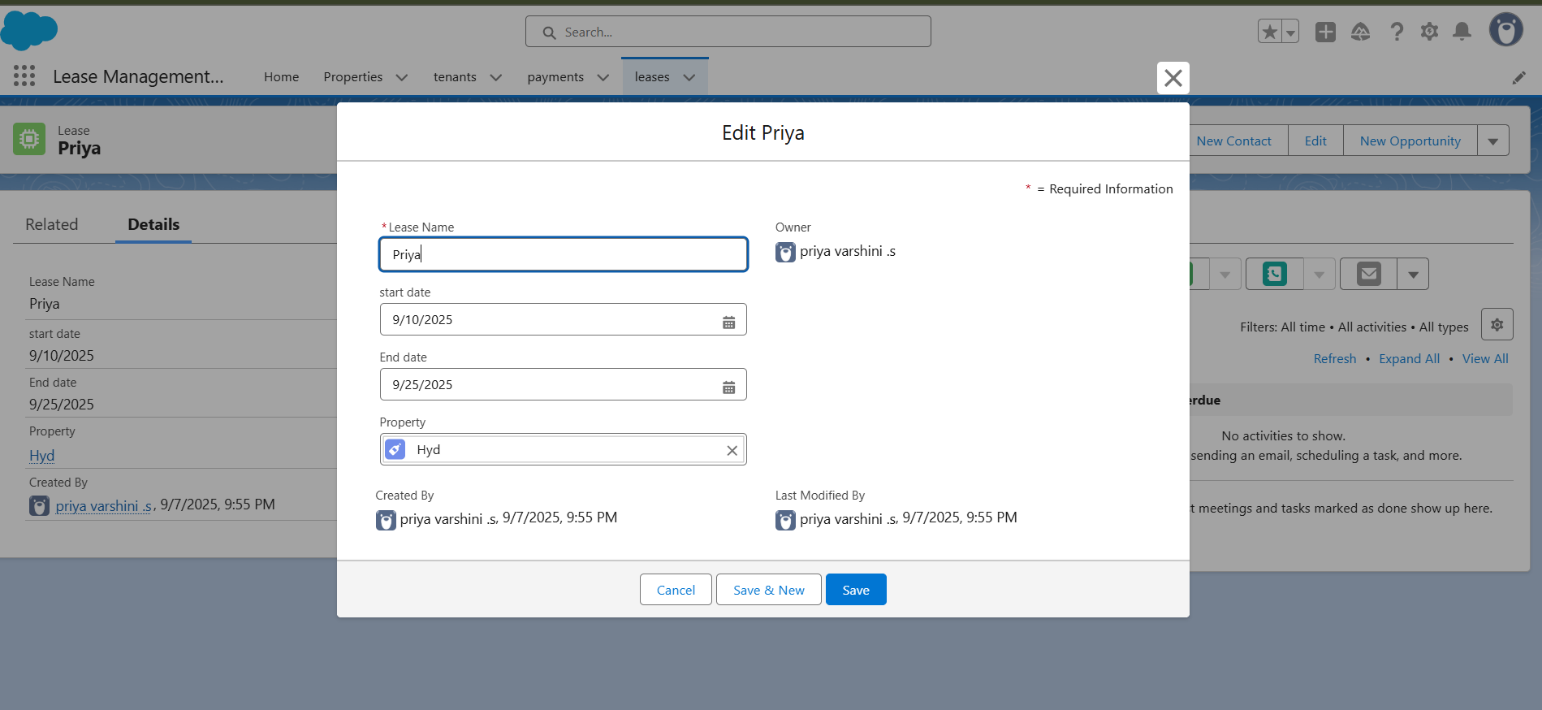




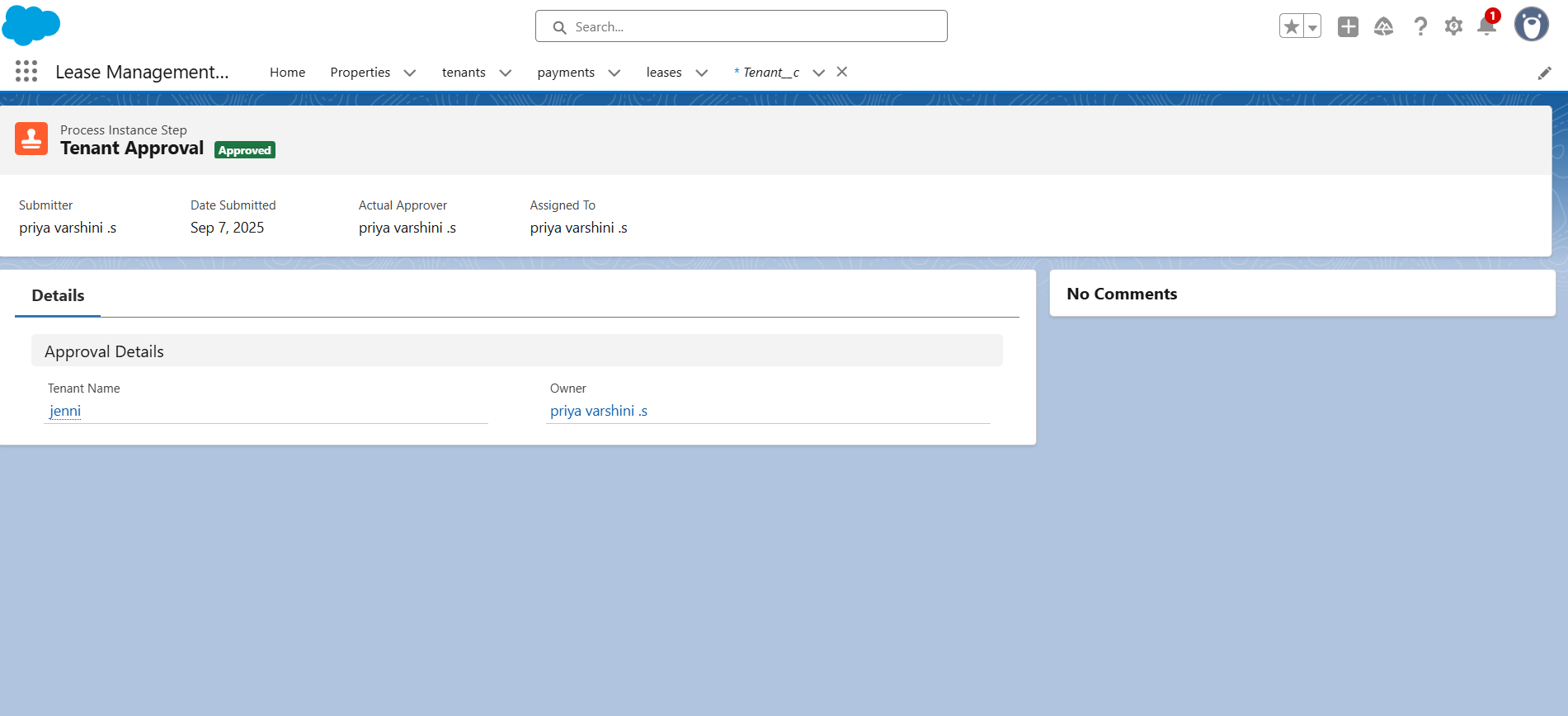
****

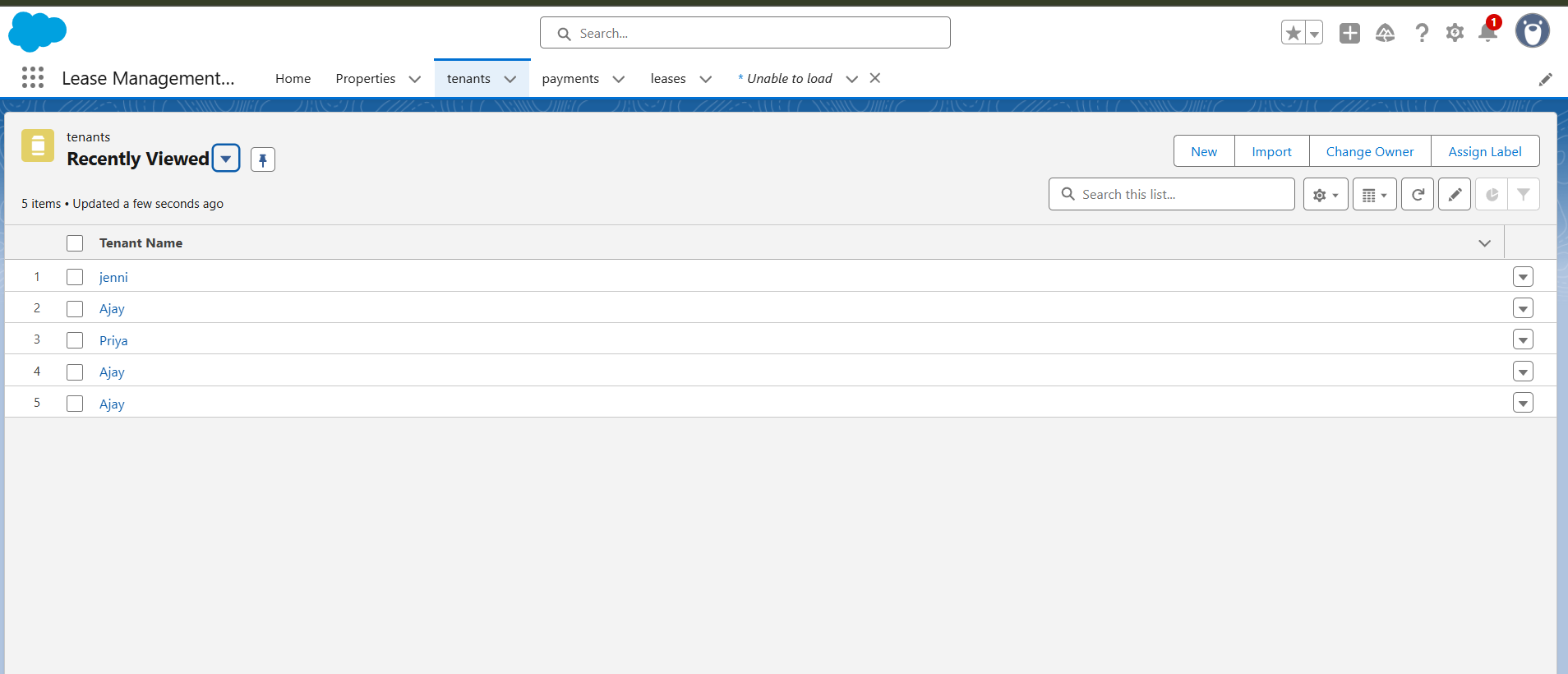
# FUNCTIONAL AND PERFORMANCE TESTING:

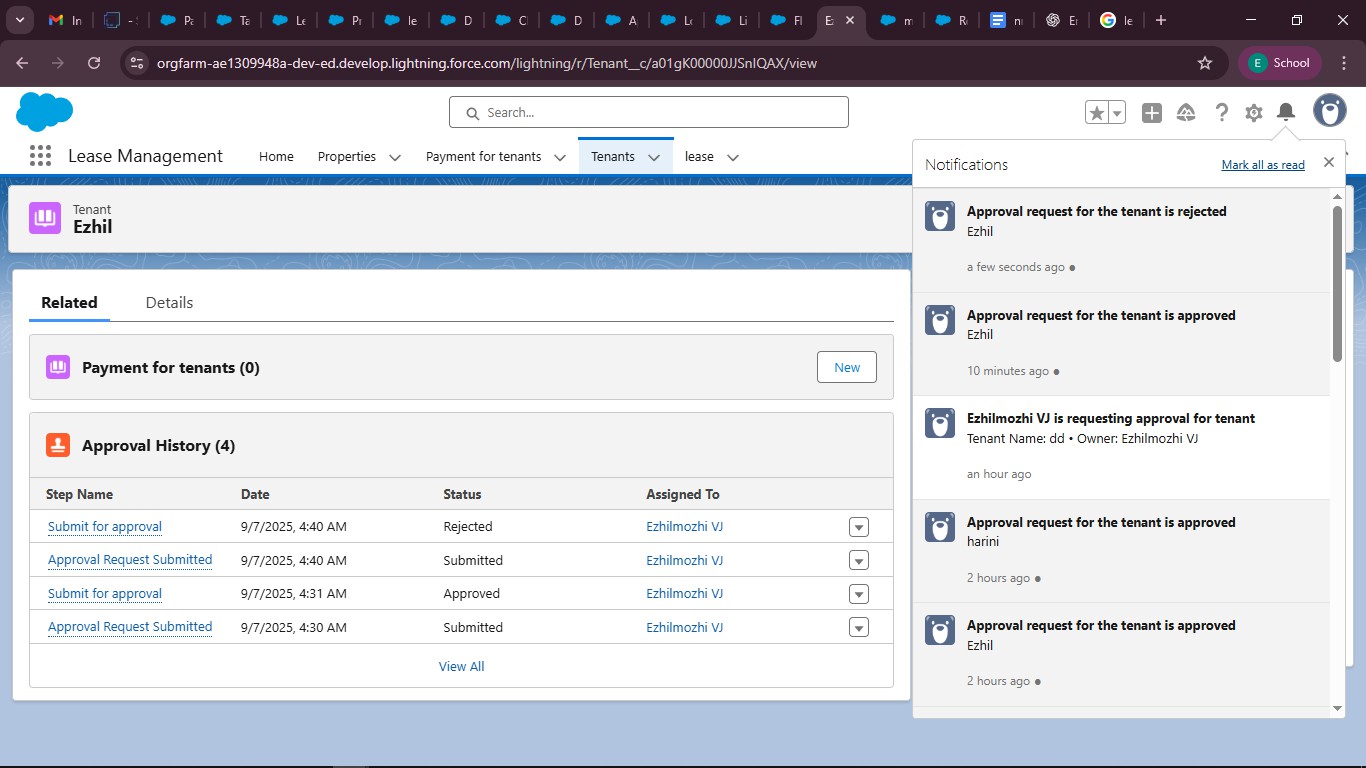
**Trigger validation by entering duplicate tenant property records**

****

**Approval process validated through email alerts and status updates:**

****

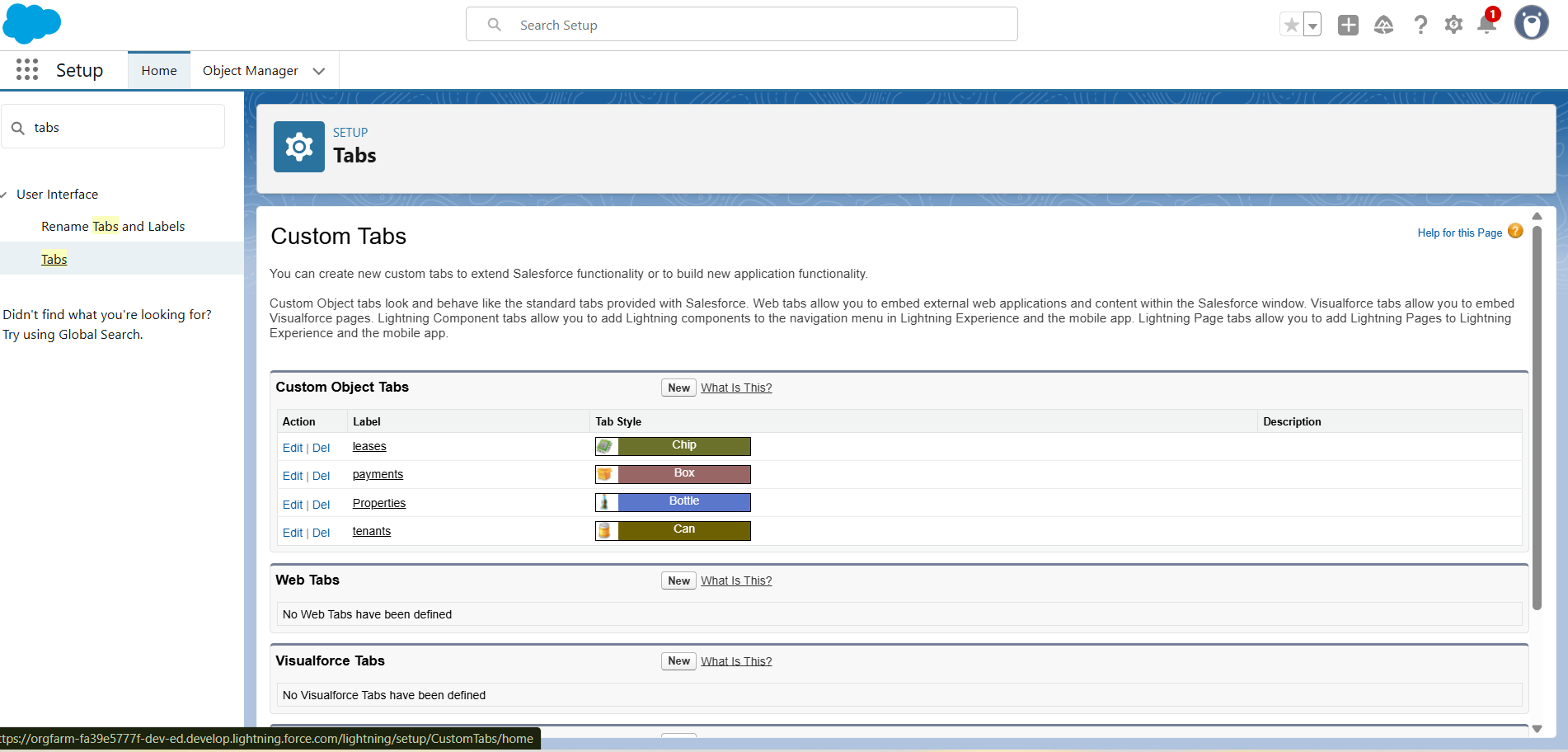
****



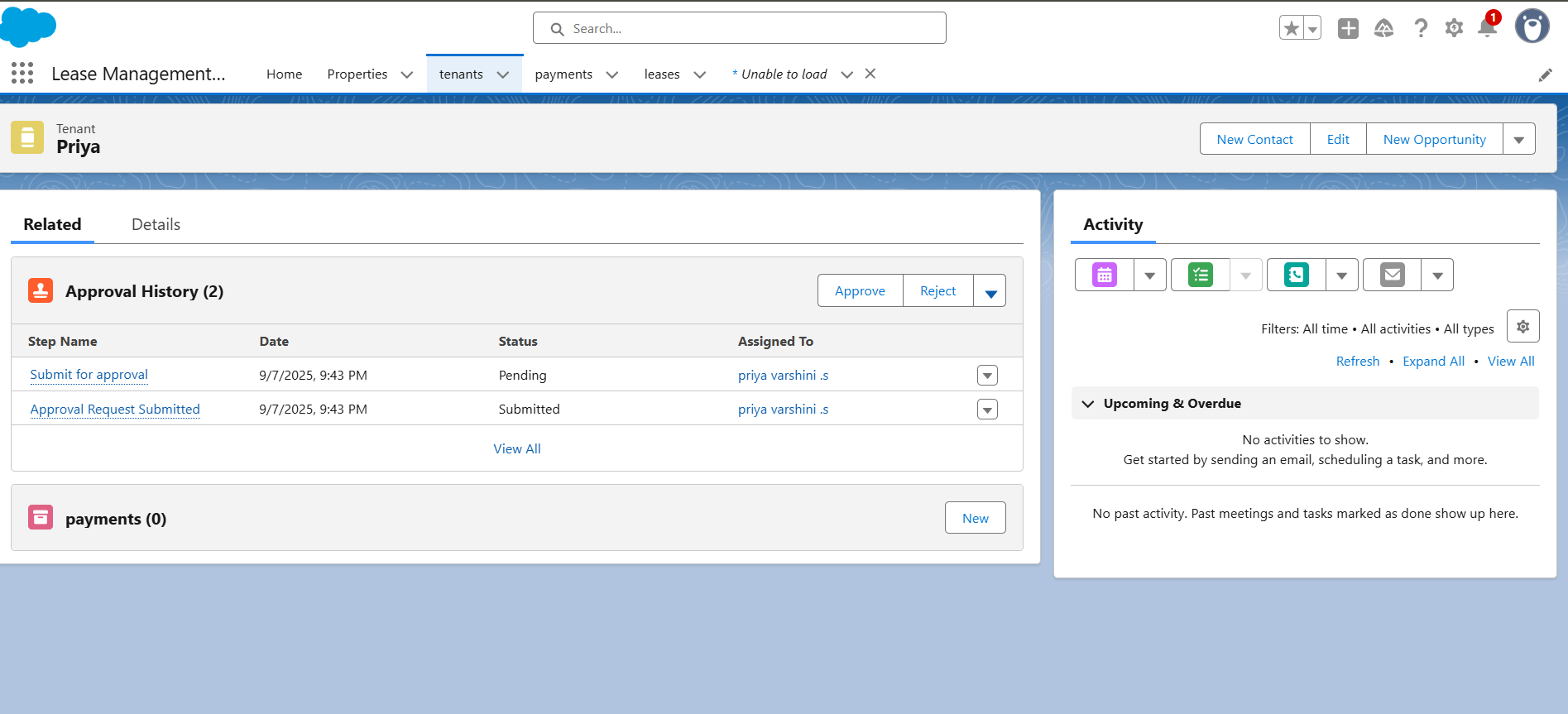
# RESULTS

Output Screenshots

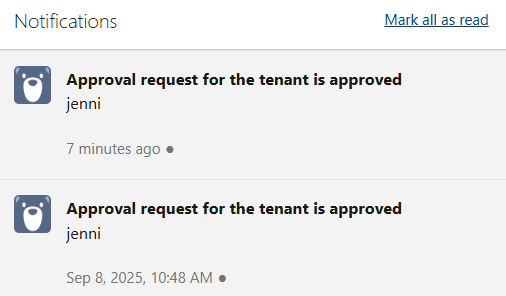
Tabs for Property, Tenants, Lease, Payment



**Email alerts**

****

**Approval process notifications**

****

**CONCLUSION**

Lease Management ensures smooth coordination between property owners, tenants, and administrators by centralizing records, automating approvals, and tracking payments. With the use of structured processes, validation rules, and timely notifications, it enhances transparency, reduces errors, and improves efficiency in managing property leases.

**APPENDIX:**

## Test.apxt:

trigger test on Tenant c (before insert)

{

if(trigger.isInsert && trigger.isBefore){ testHandler.preventInsert(trigger.new);

}

}

## testHandler.apxc:

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');

}

}

}

}

**MonthlyEmailScheduler.apxc:**

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});

}

}

}