**LEASE MANAGEMENT**

**College Name: KPR College of Arts Science and Research**

**College Code: bruaz**

**TEAM ID: NM2025TMID21504**

**TEAM MEMBERS:**

**Team Leader Name: PRASANTH L**

**Email: 23bai037@kprcas.ac.in**

**Team Member 1: YUVARAJ R**

**Email: 23bai060@kprcas.ac.in**

**Team Member 2:  JOSHNA ASHIKA M**

**Email: 23bai018@kprcas.ac.in**

**Team Member 3: JOVITA J**

**Email: 23bai019@kprcas.ac.in**

# 1.INTRODUCTION

## 1.1 Project Overview

The Lease Management System is a Salesforce-based application designed to streamline the processes associated with leasing real estate properties. It handles tenant management, lease contracts, payments, and communication with automation features such as flows, approval processes, and email alerts.



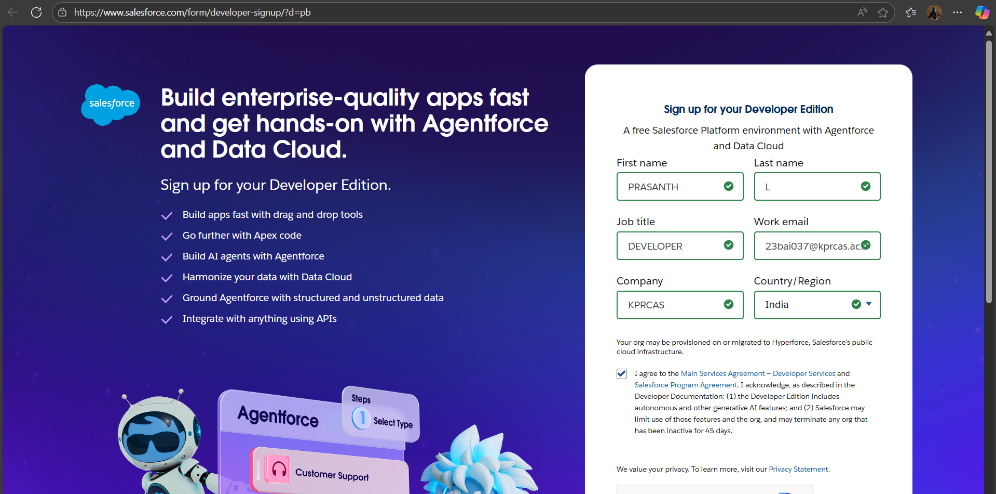
## 1.2 Purpose

The main objective of the project is to enable organizations to efficiently manage properties, tenants, and lease-related activities. It reduces manual intervention, improves accuracy, and ensures better compliance and communication.

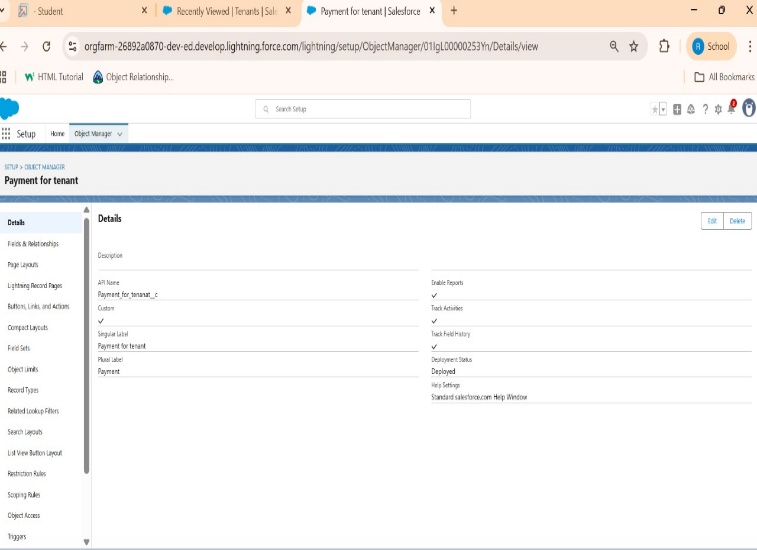
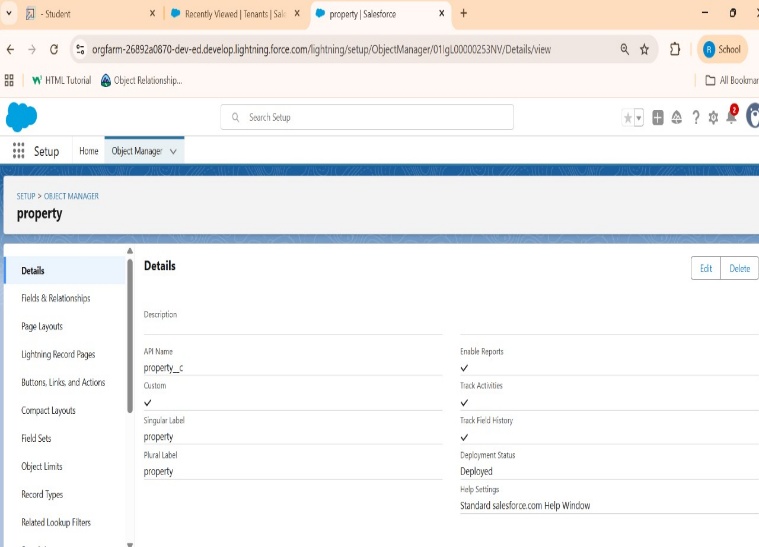
# DEVELOPMENT PHASE

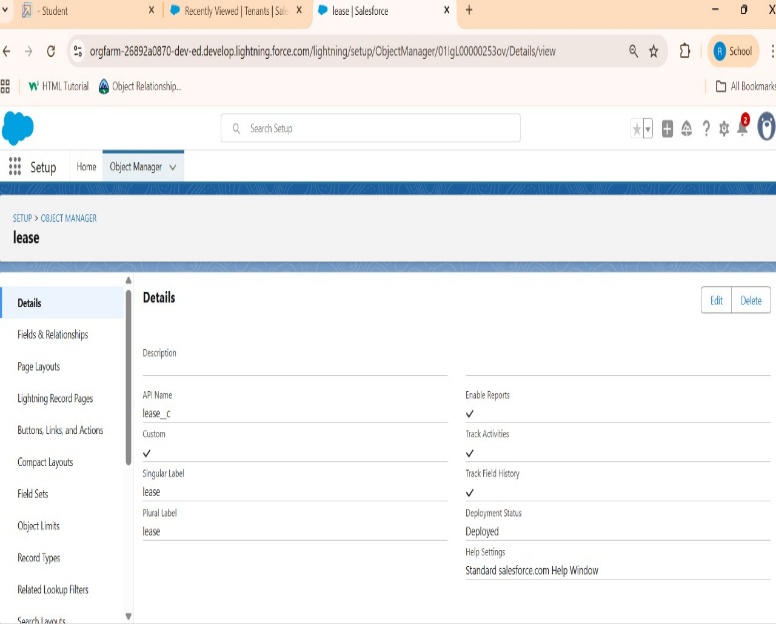
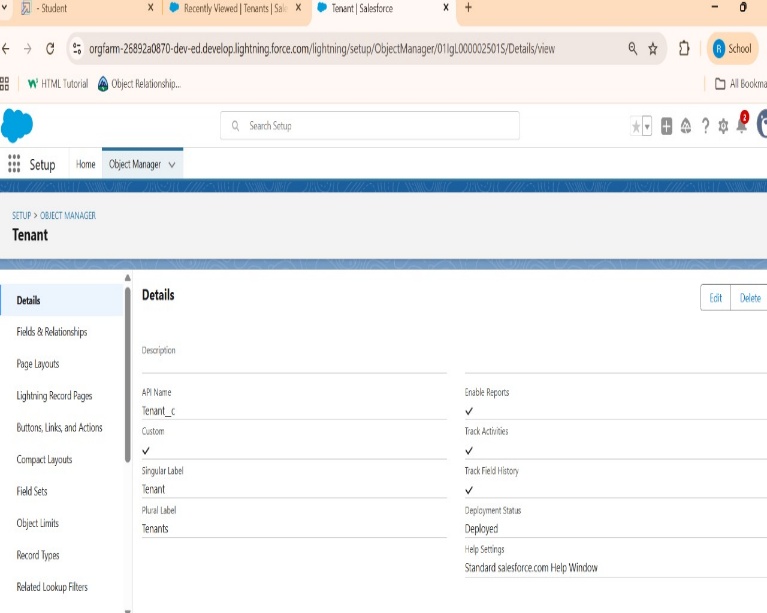
**Creating Developer Account:**

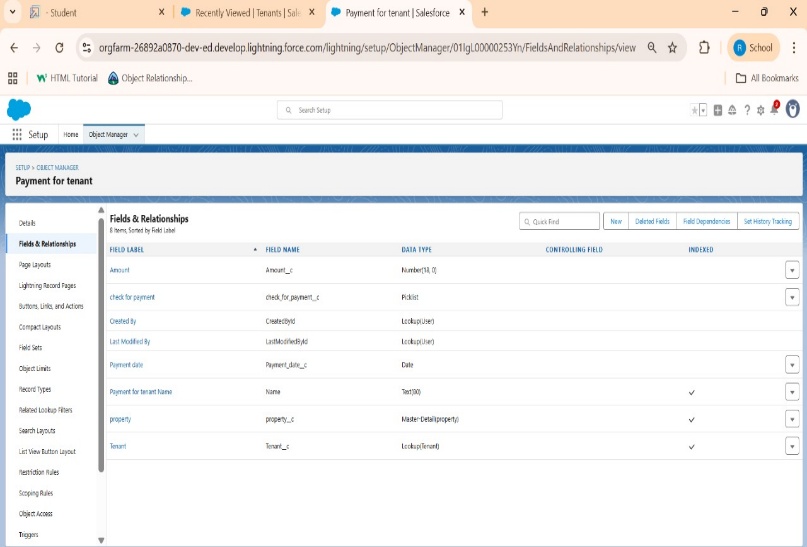
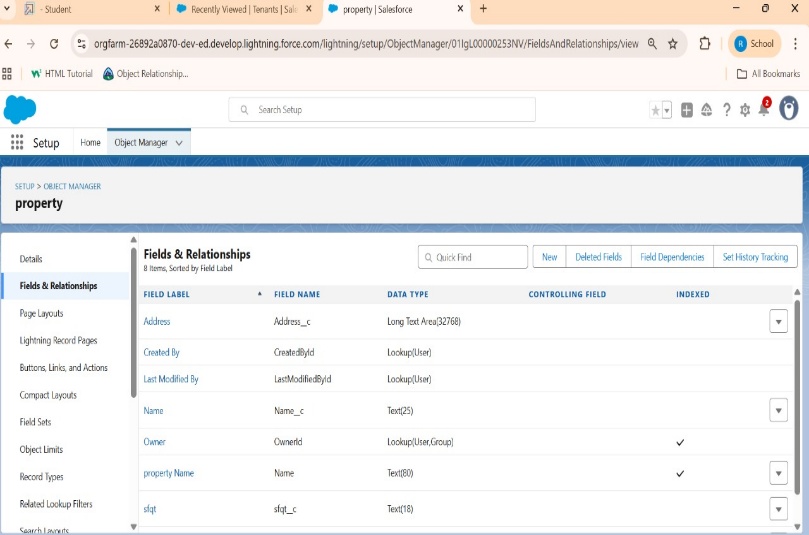
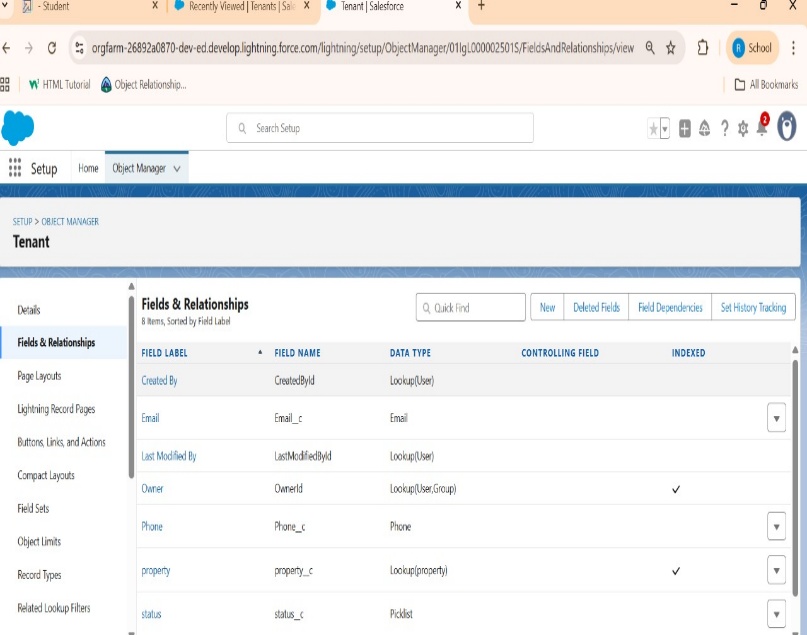
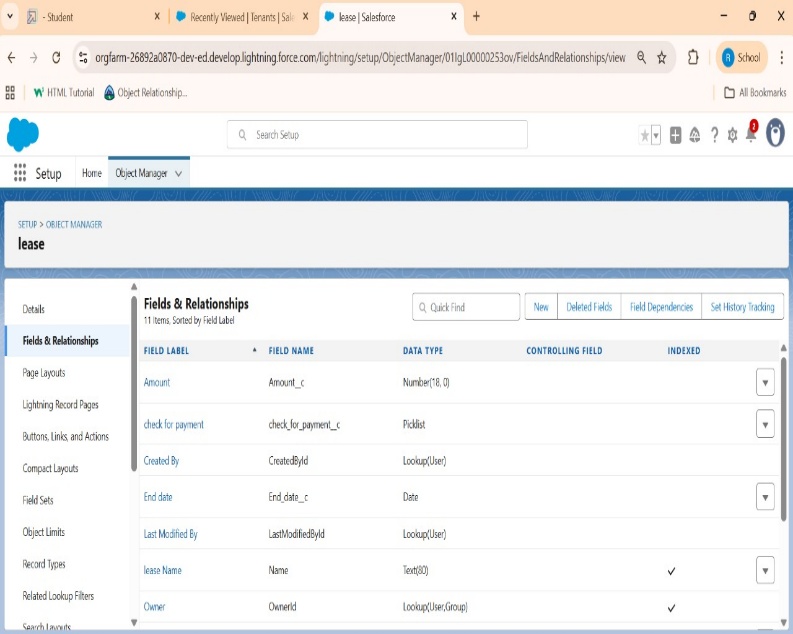
By using this URL [**- https://www.salesforce.com/form/developer-signup/?d=pb**](https://www.salesforce.com/form/developer-signup/?d=pb)



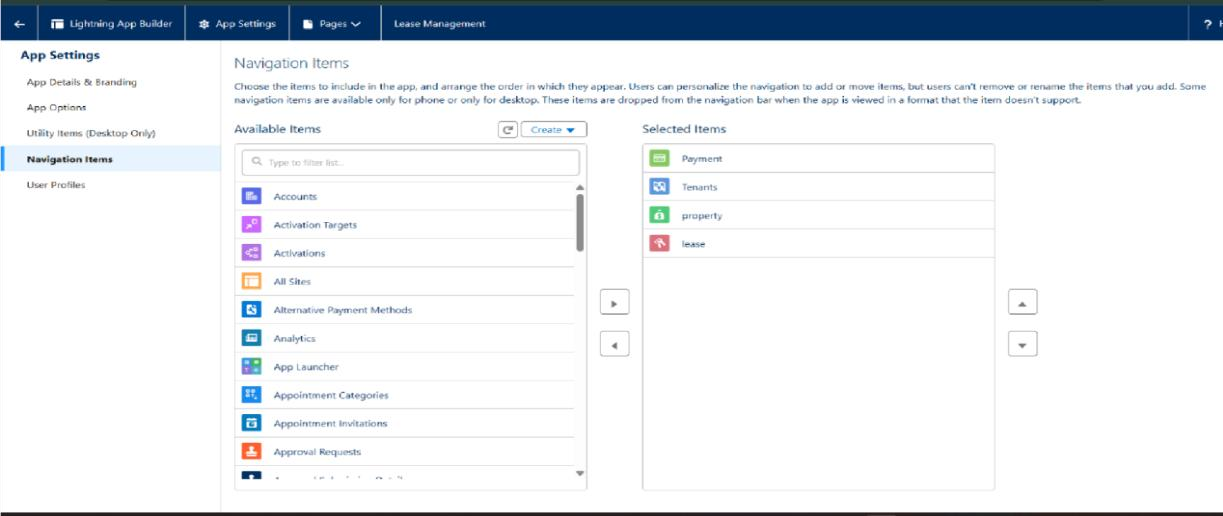
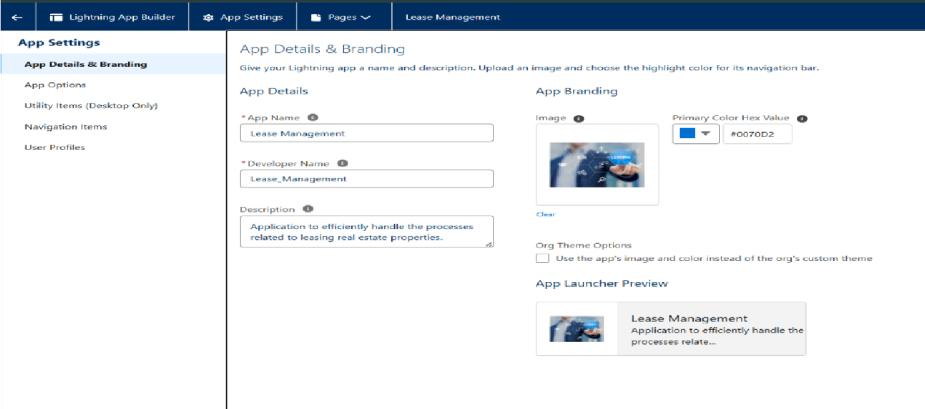
* Created objects: Property, Tenant, Lease, Payment

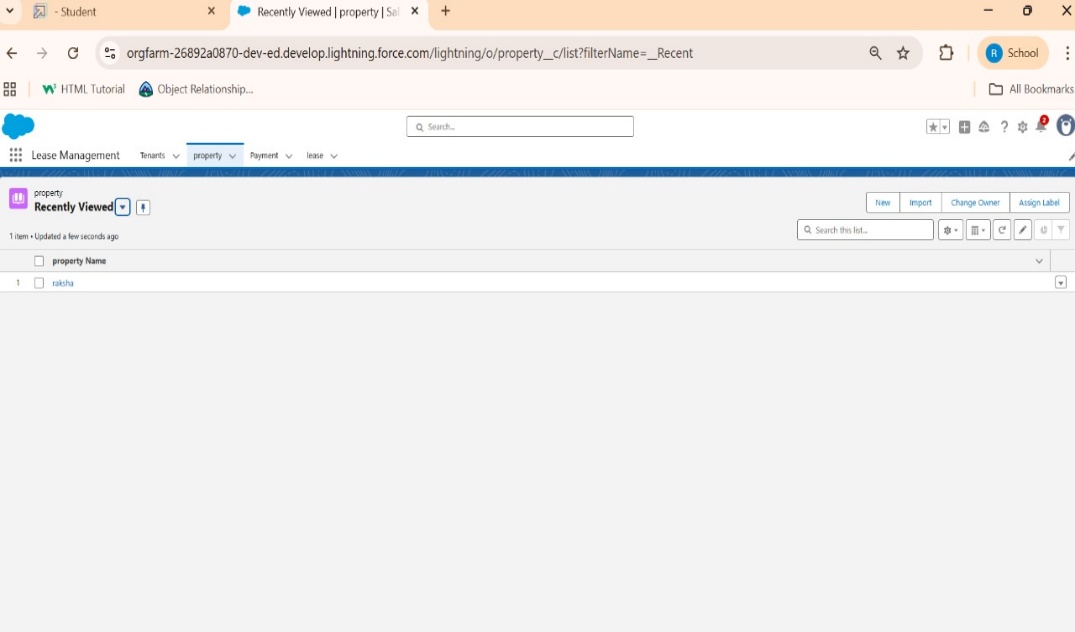


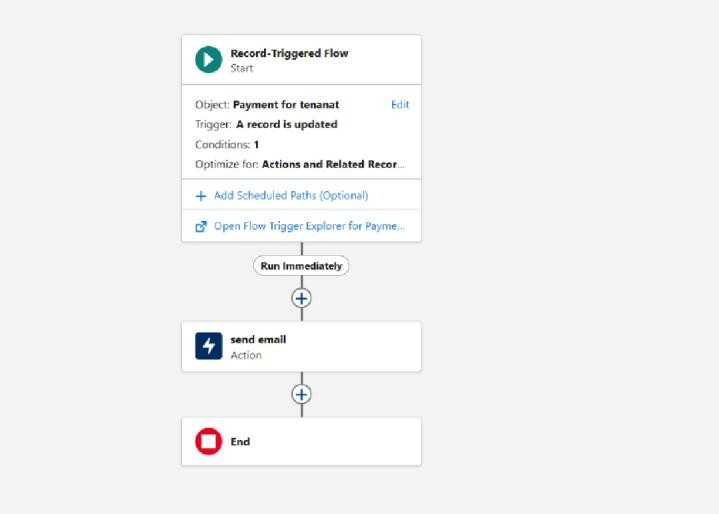


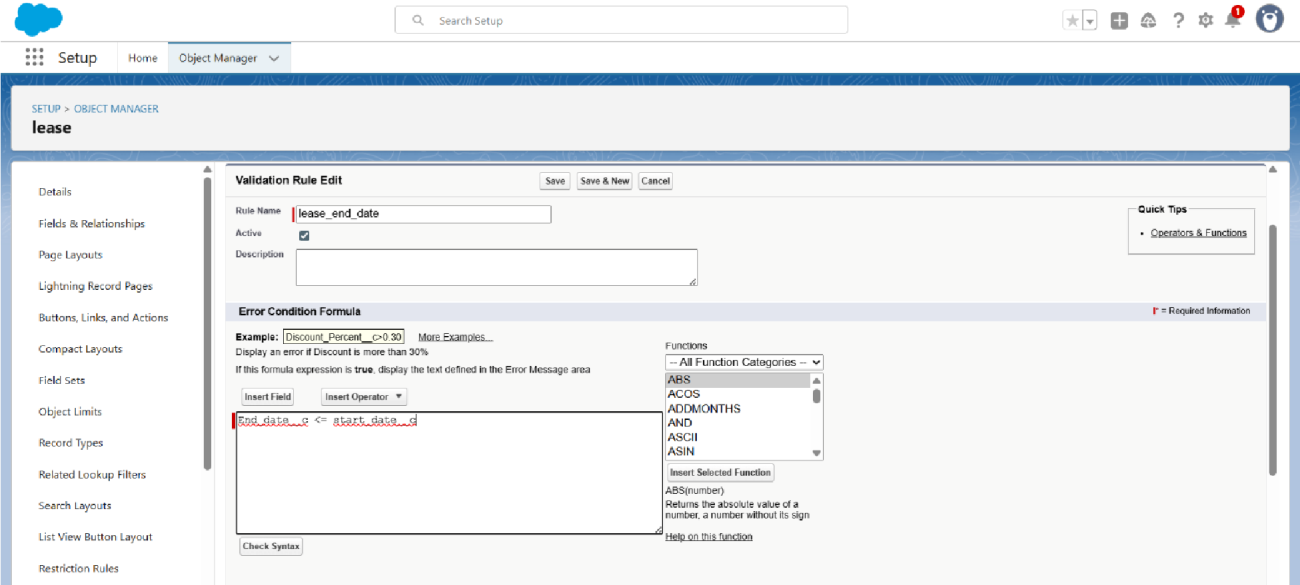
* Configured fields and relationships

* Developed Lightning App with relevant tabs

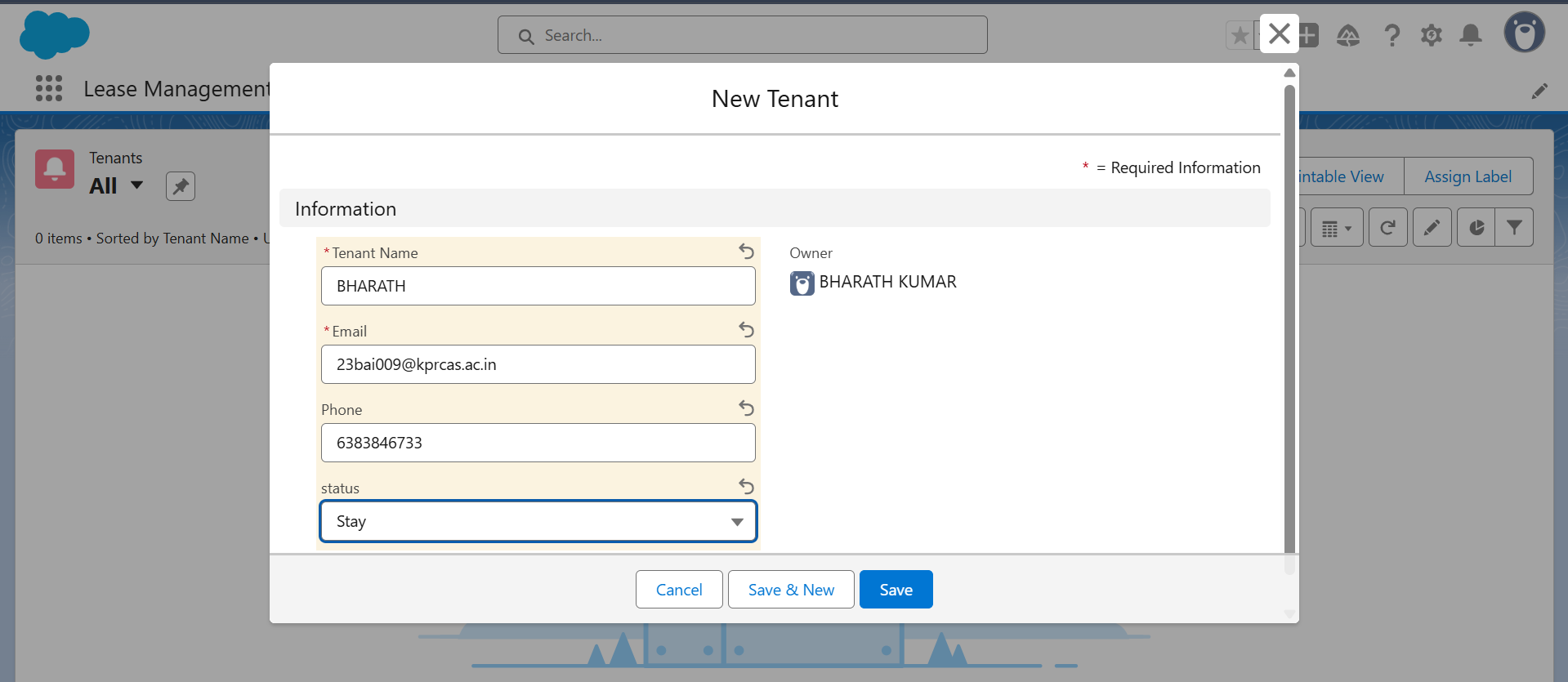




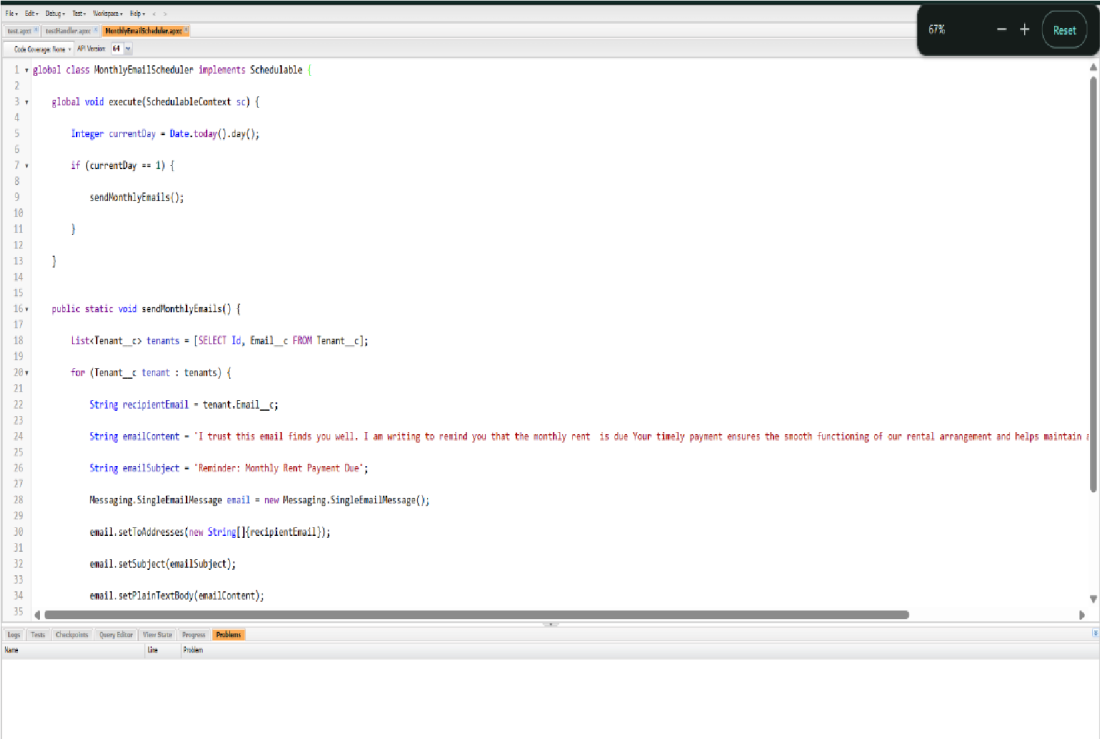
* Implemented Flows for monthly rent and payment success

* To create a validation rule to a Lease Object

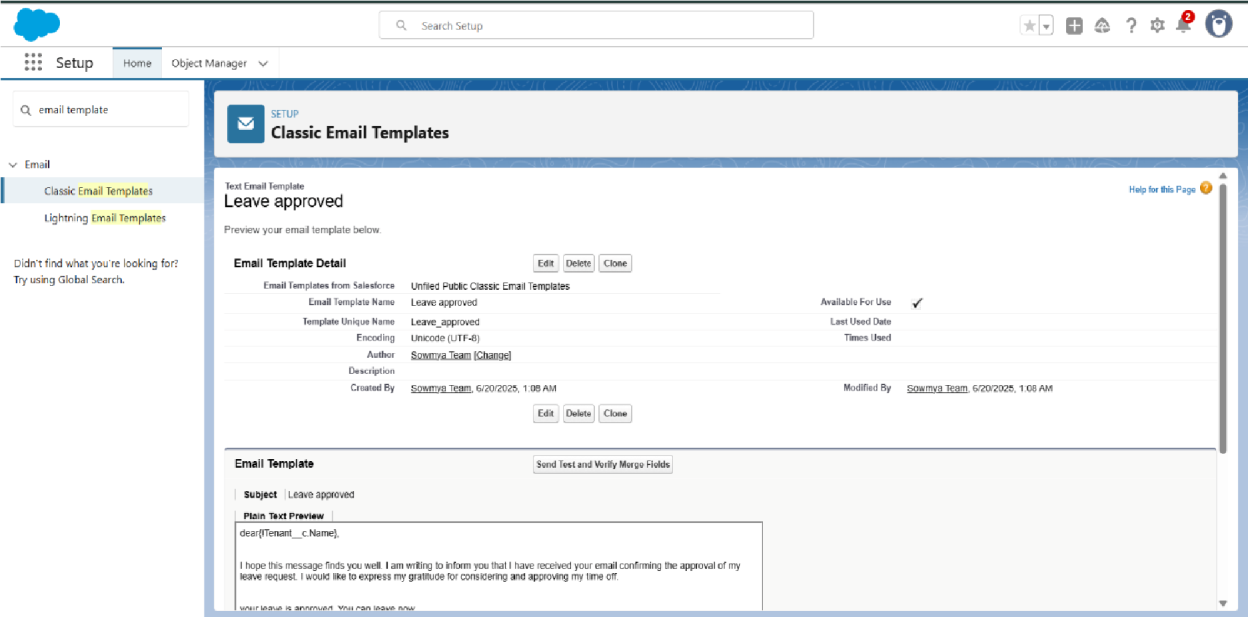
* Added Apex trigger to restrict multiple tenants per property

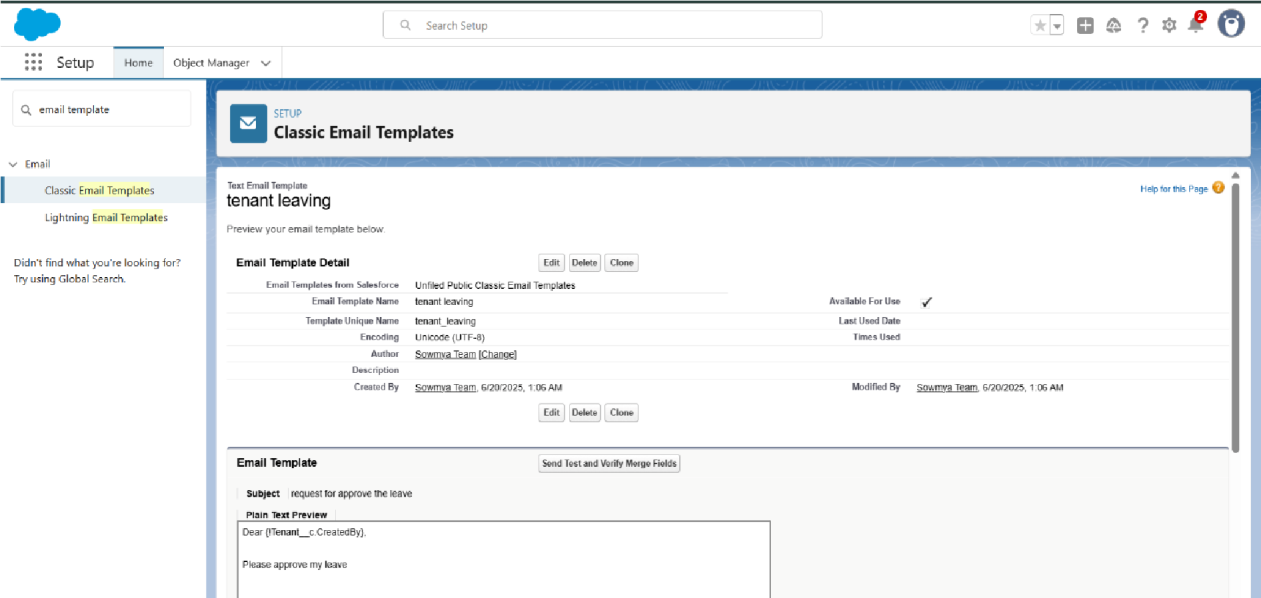


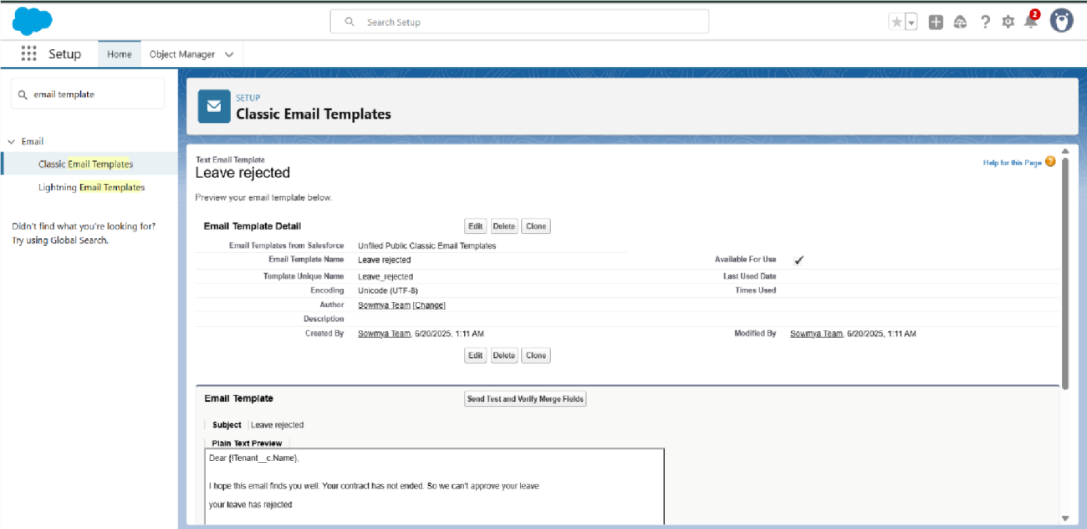
* Scheduled monthly reminder emails using Apex class

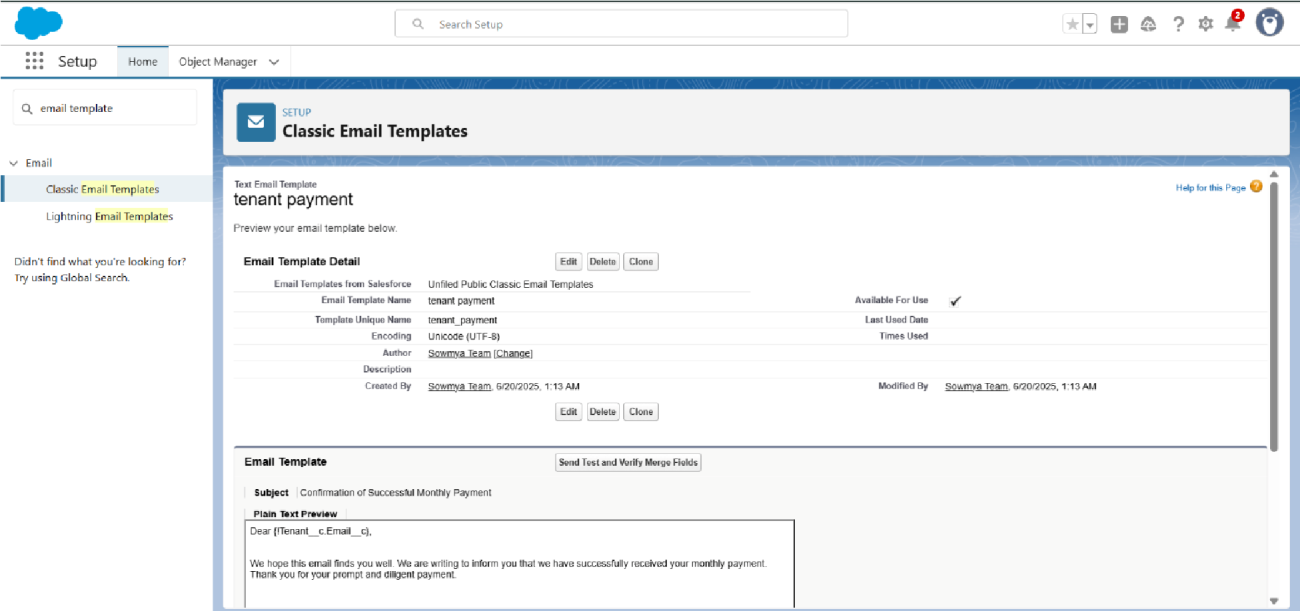
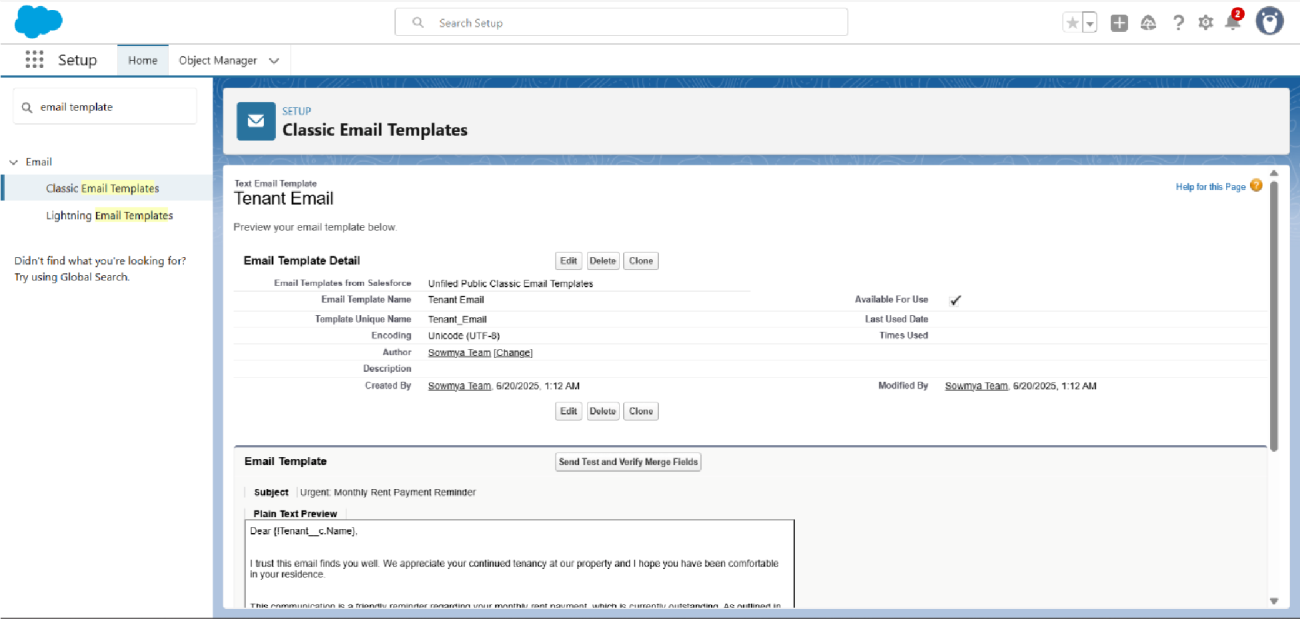


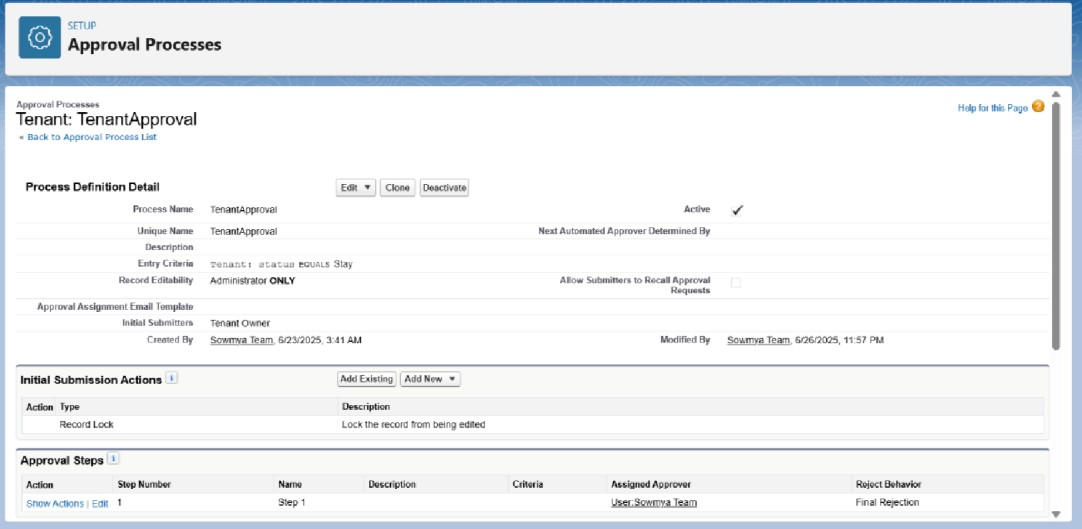
* Built and tested email templates for leave request, approval, rejection, payment, and reminders





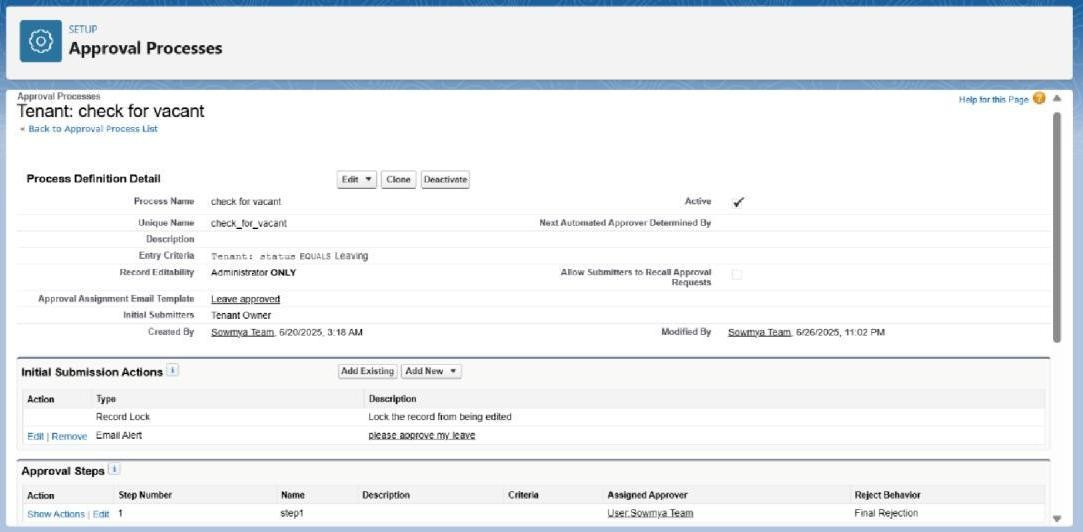


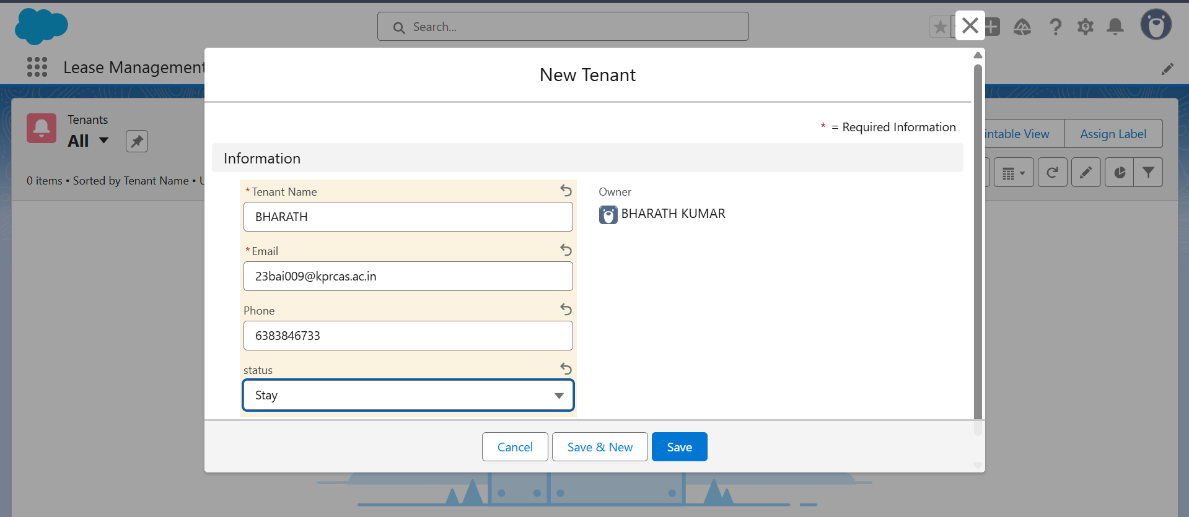




Approval Process creation

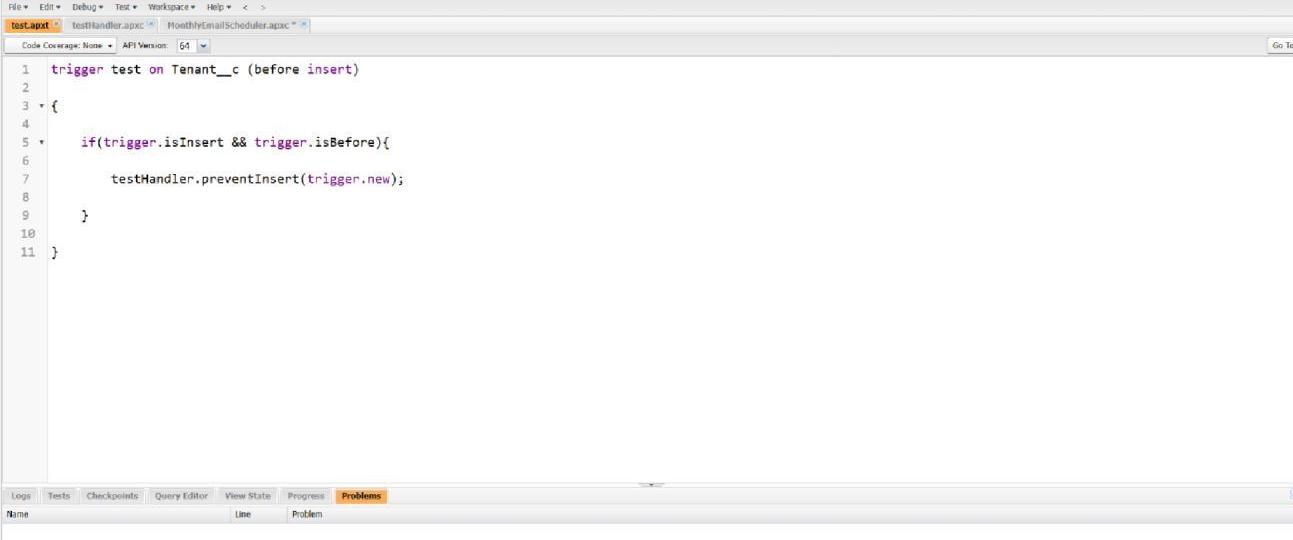
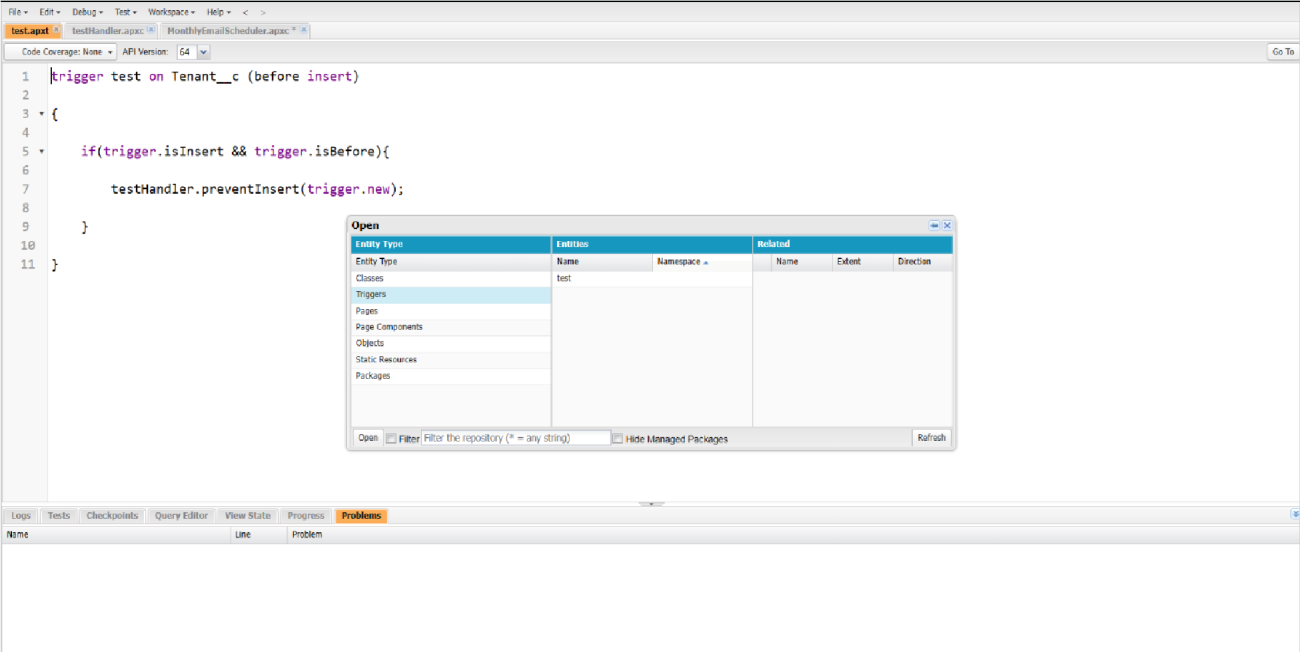
For Check for Vacant:



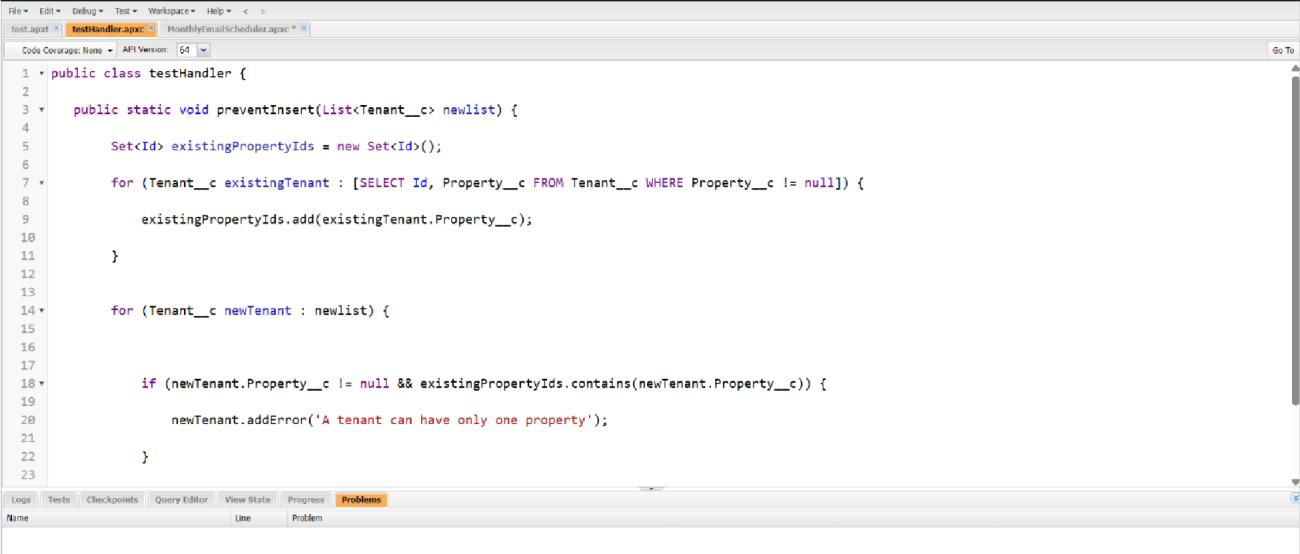
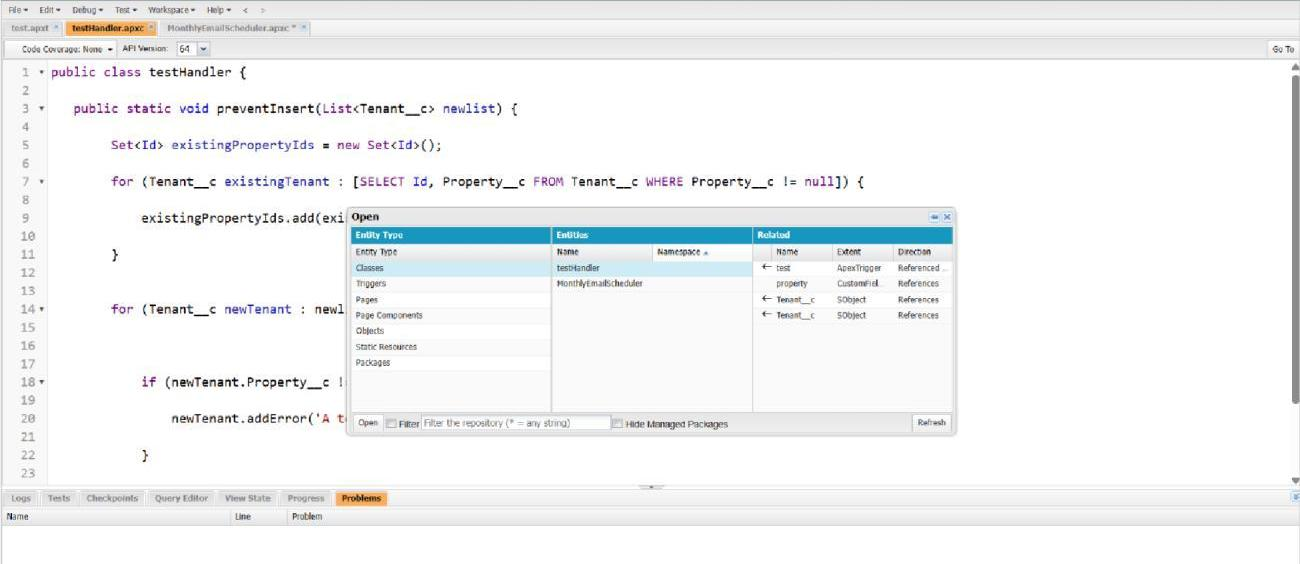


Apex Trigger

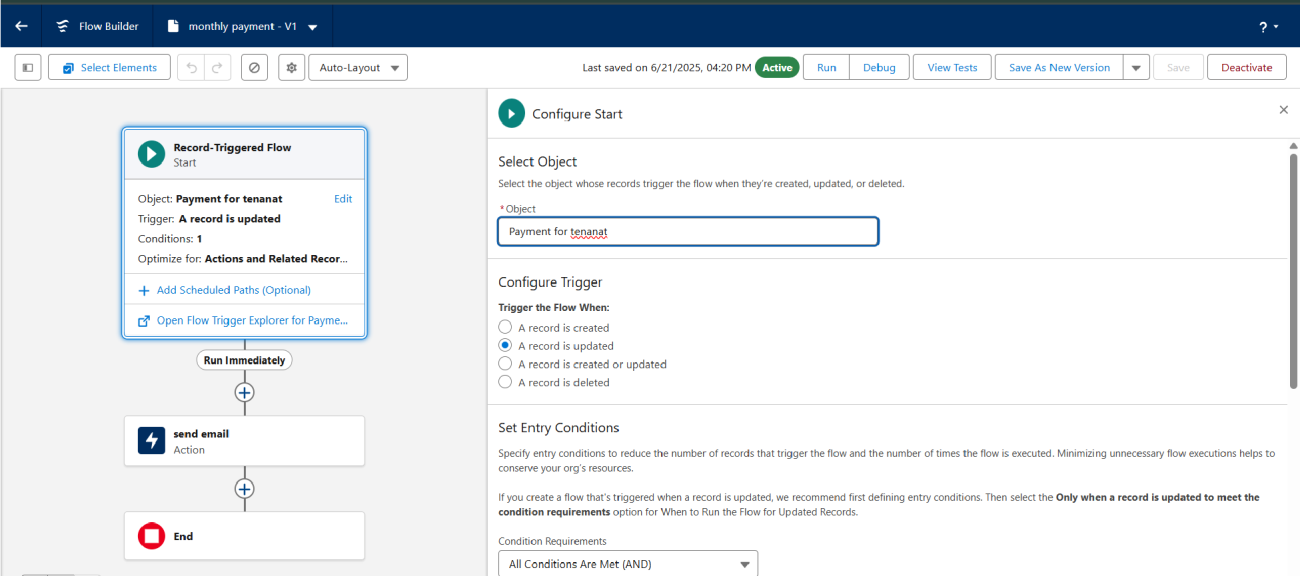
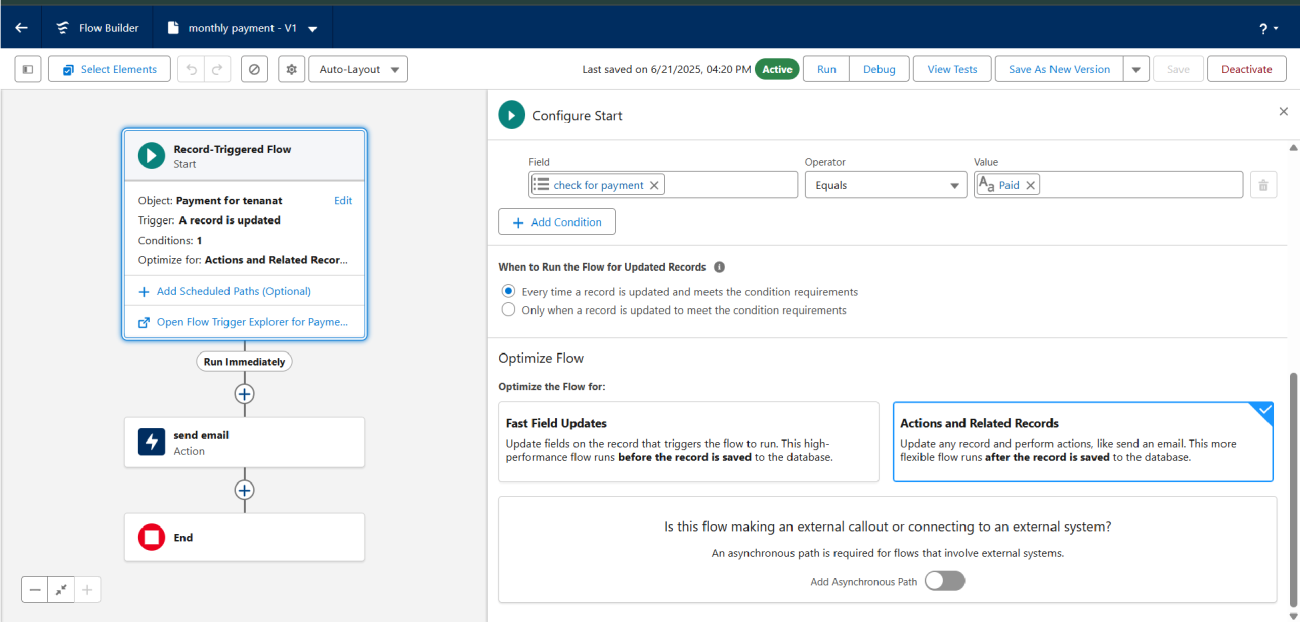
Create an Apex Trigger



Create an Apex Handler class

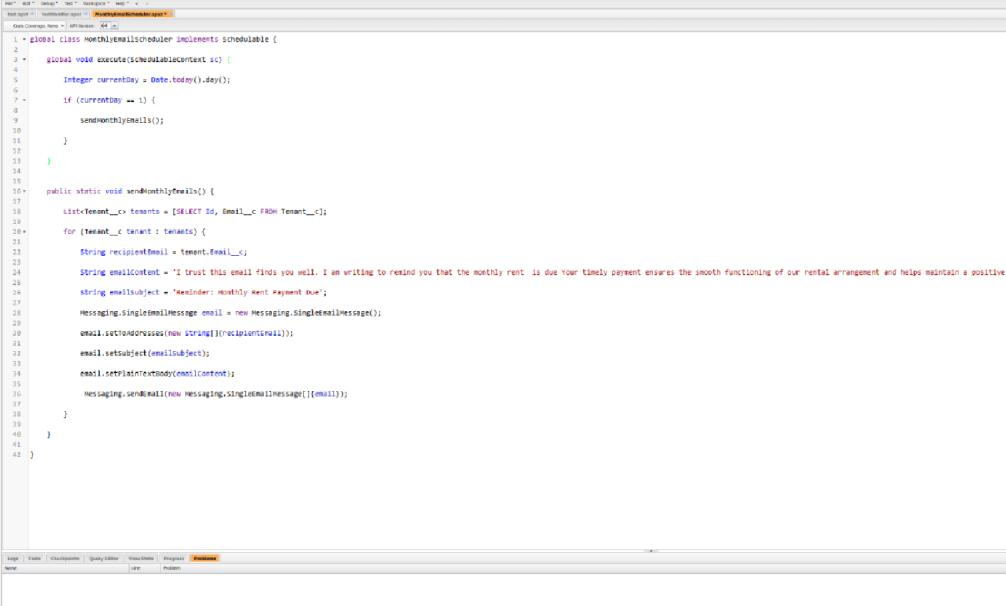
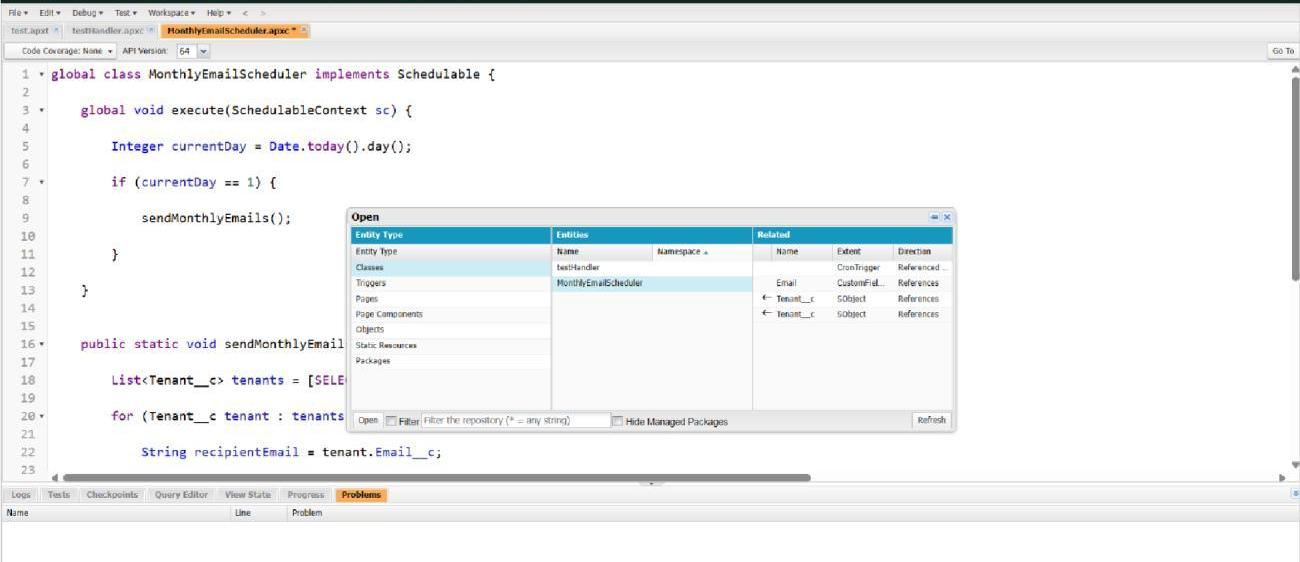


FLOWS

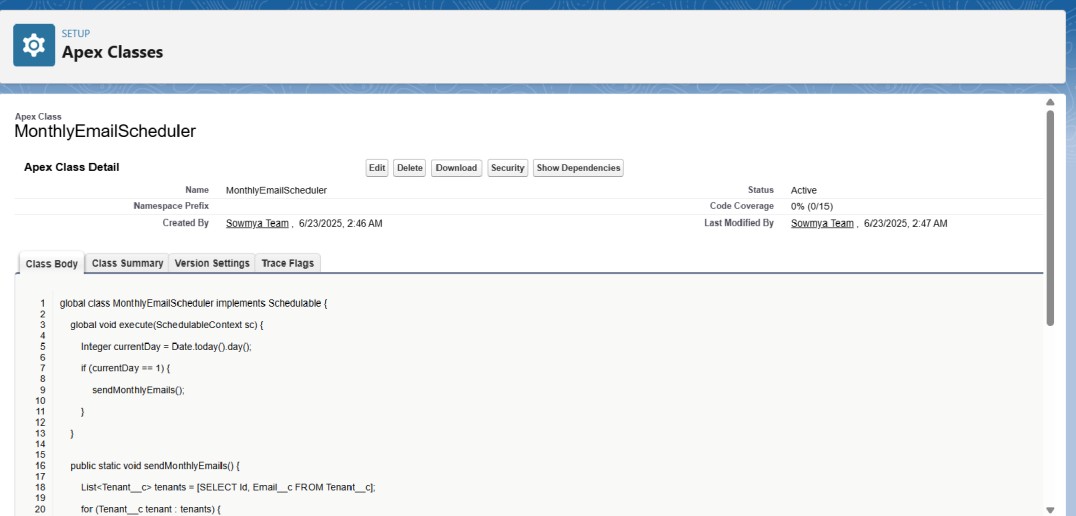


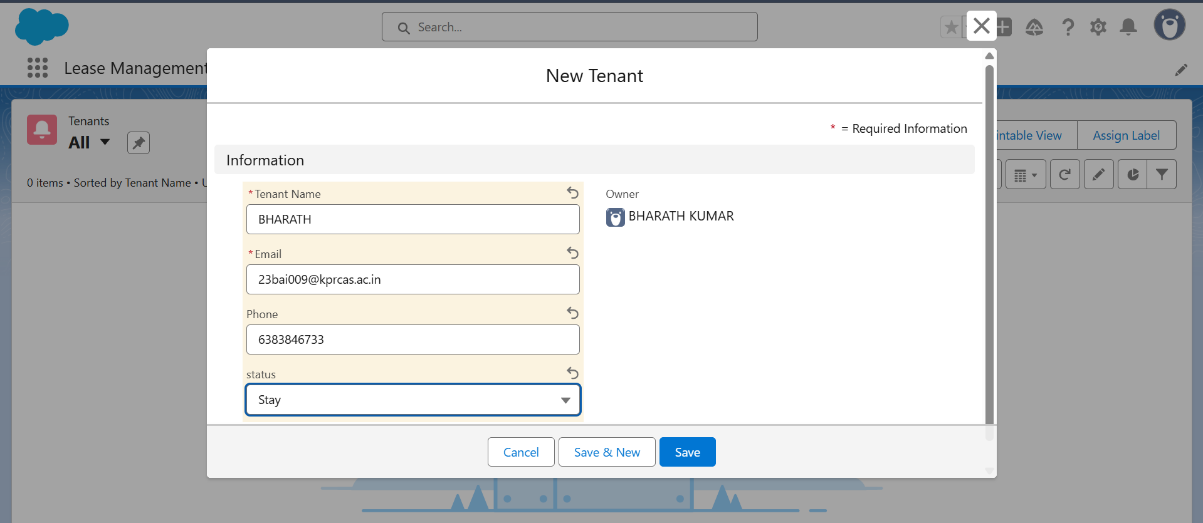
* Schedule class:

Create an Apex Class



Schedule Apex class



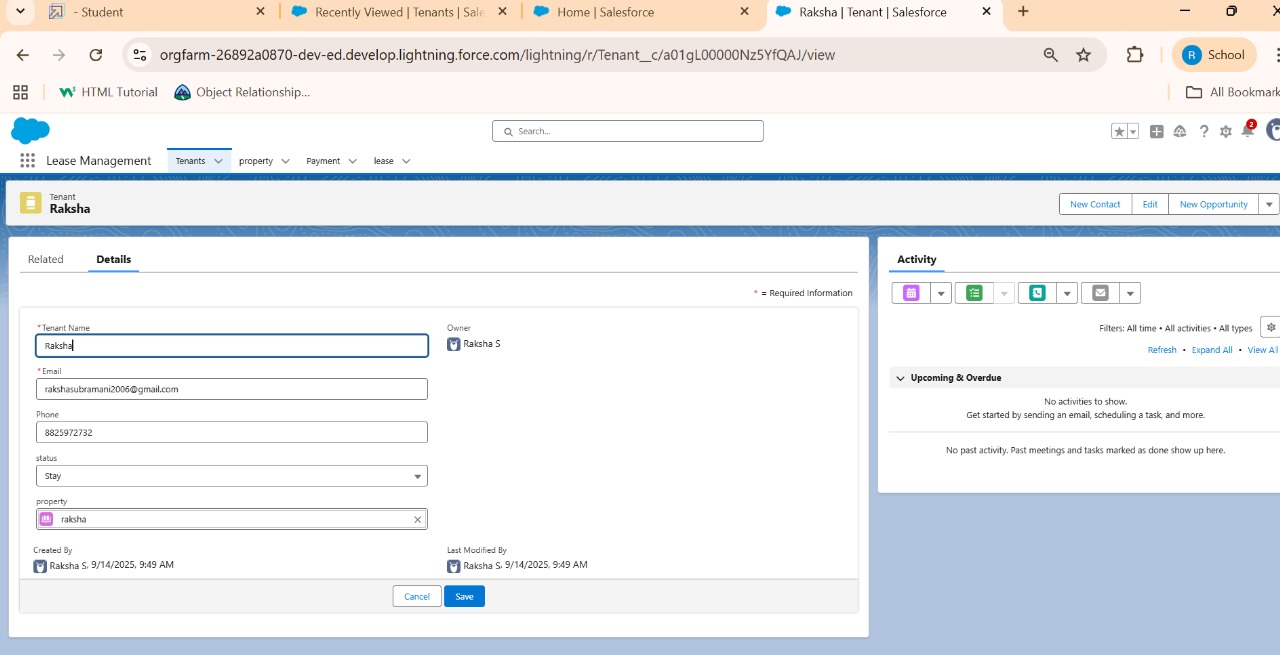


# FUNCTIONAL AND PERFORMANCE TESTING

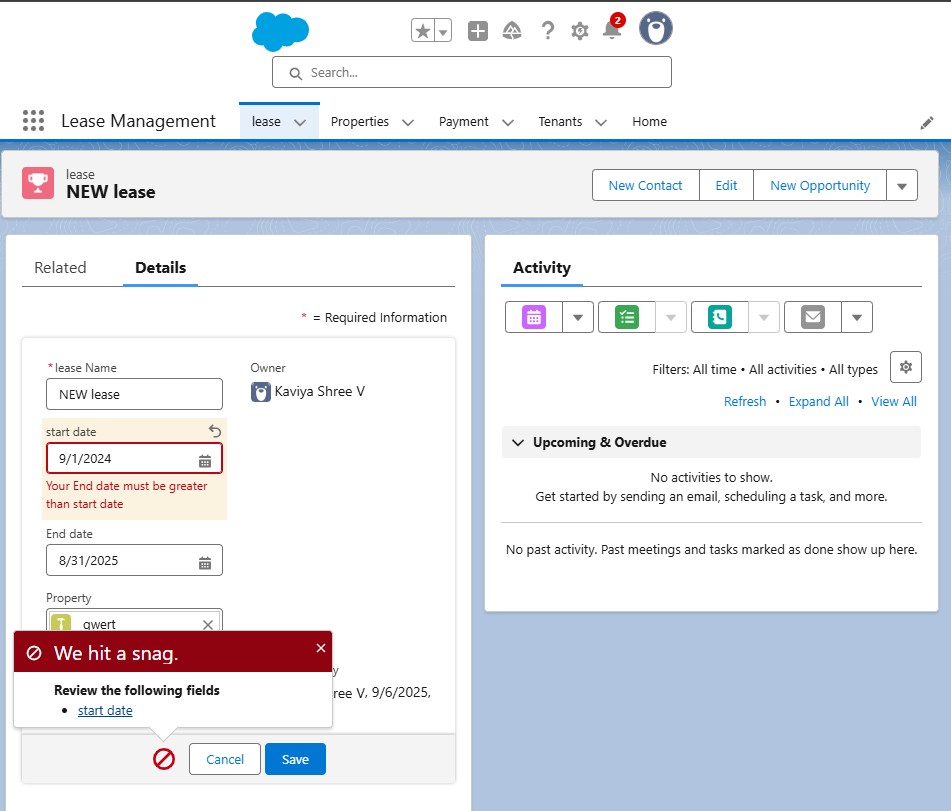
## Performance Testing

## 

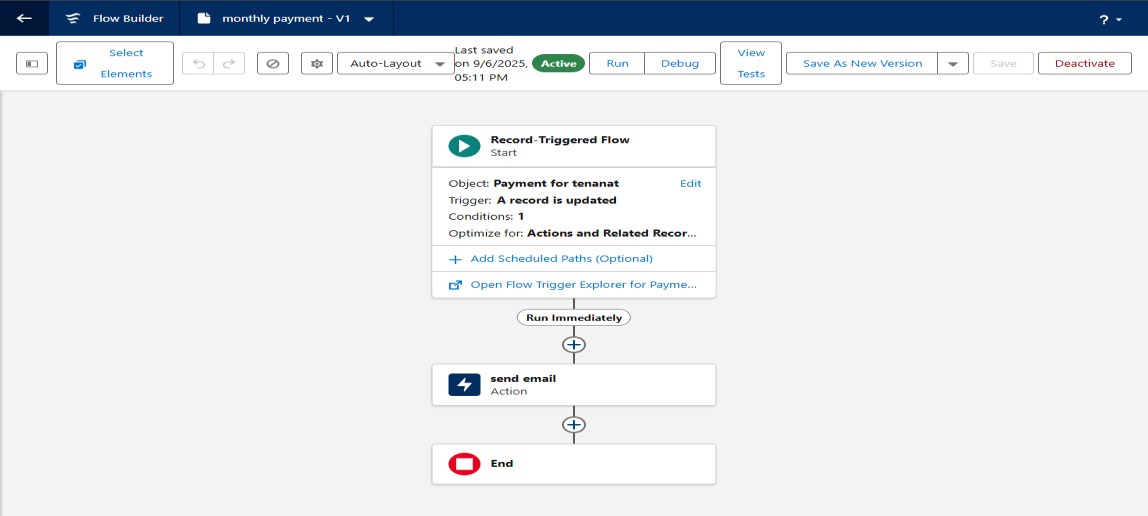
* Trigger validation by entering duplicate tenant-property records

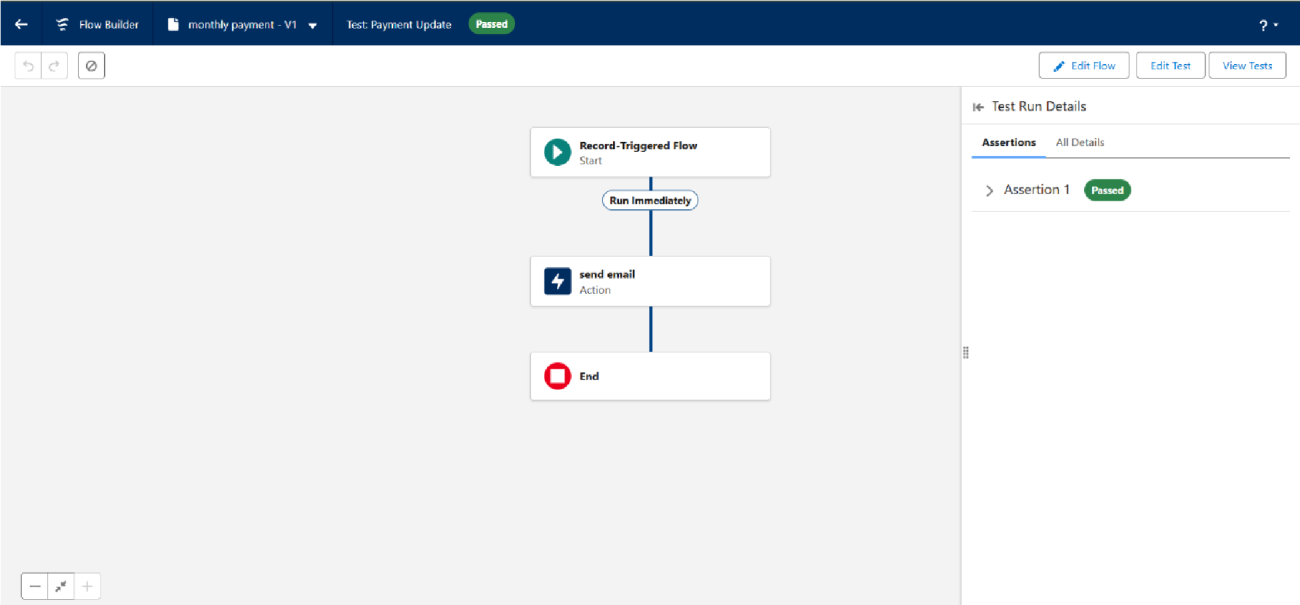


* Validation Rule checking

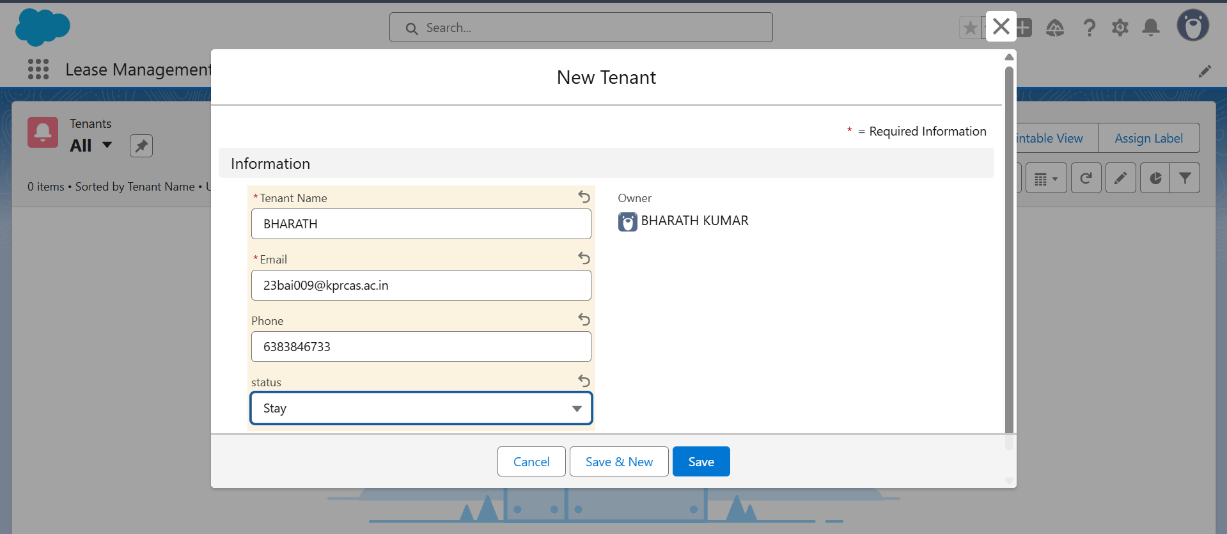


* Test flows on payment update



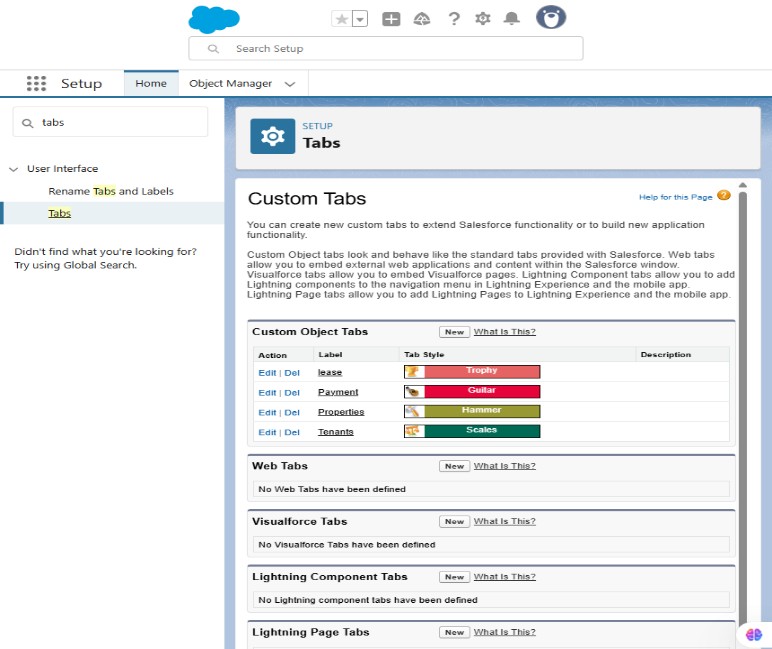


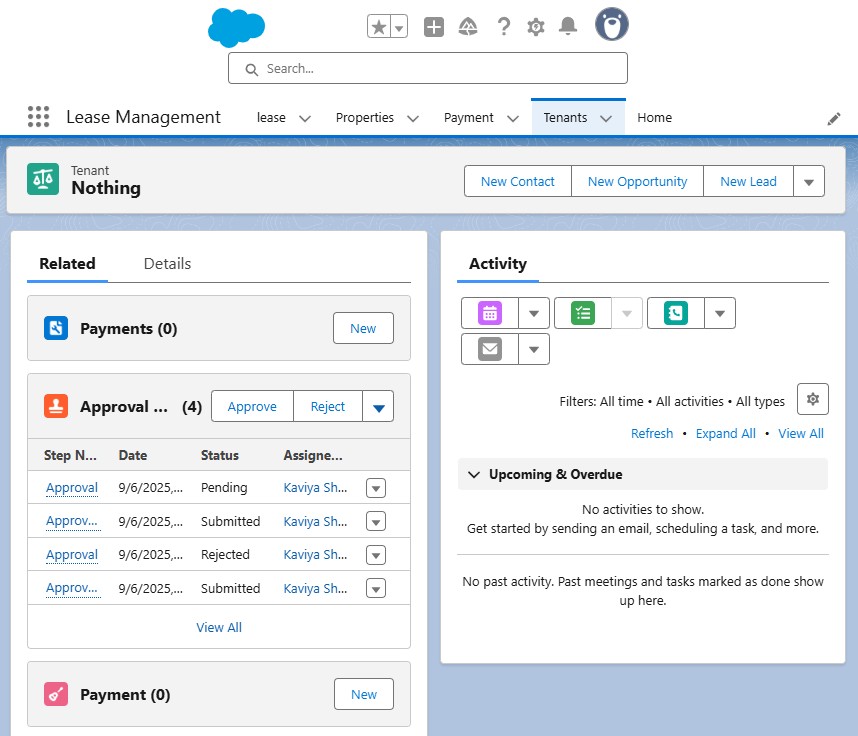
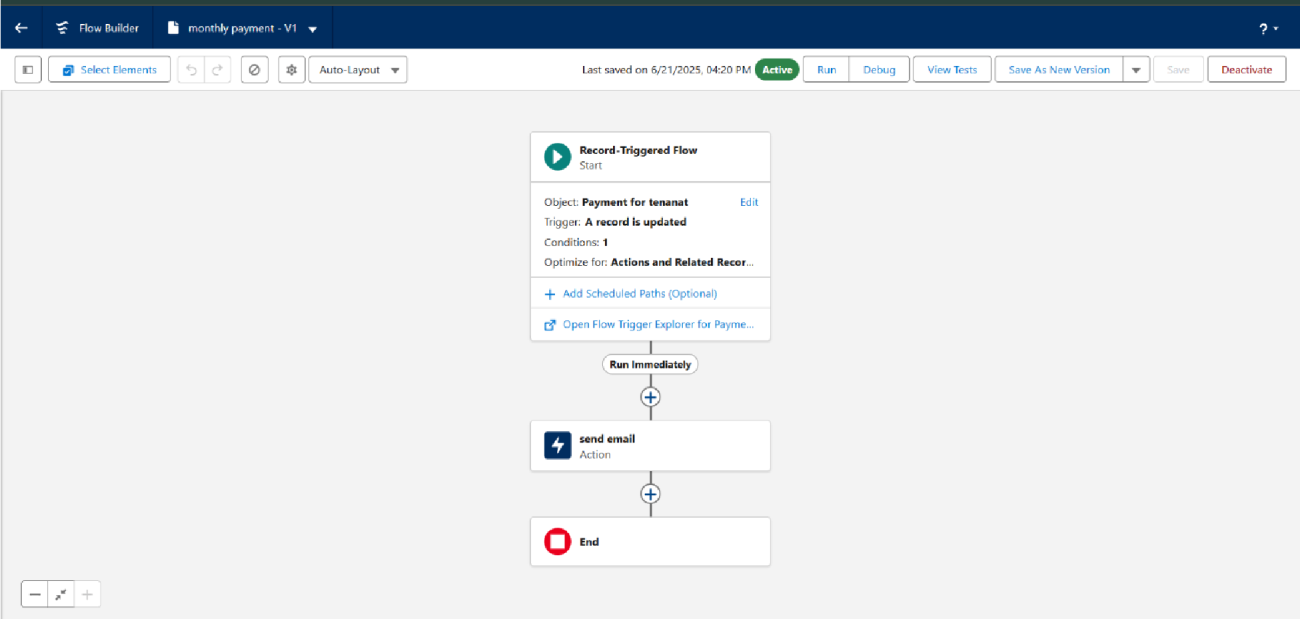
* Approval process validated through email alerts and status updates



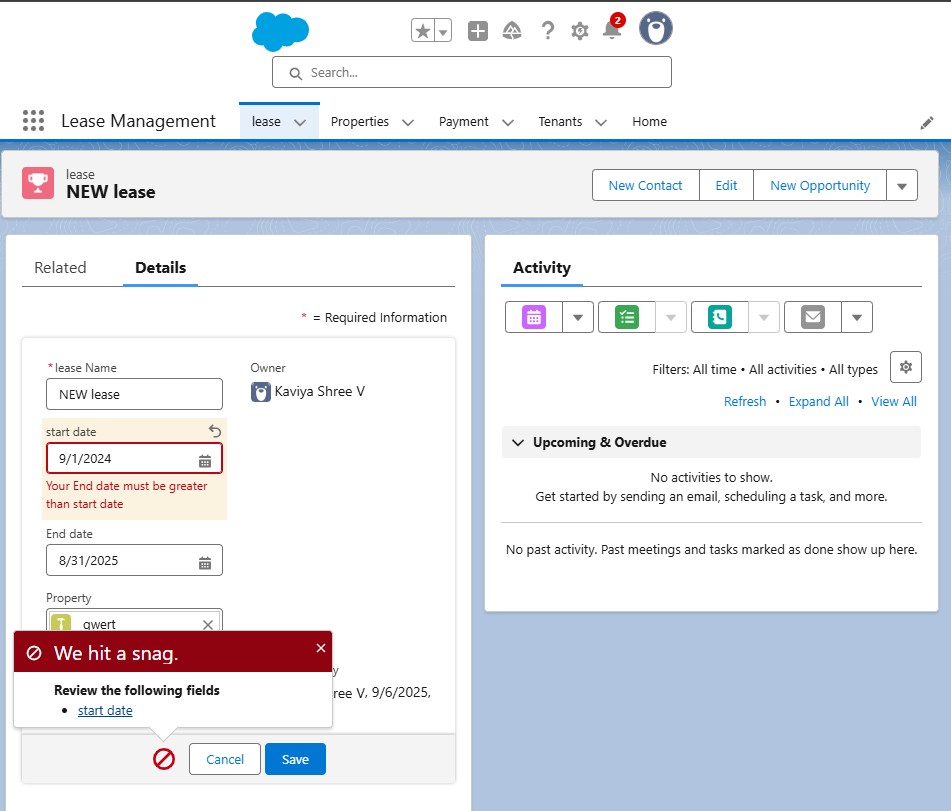
# RESULTS

## Output Screenshots

* Tabs for Property, Tenant, Lease, Payment

* Email alerts
* Flow runs

* Trigger error messages



# CONCLUSION

The Lease Management System successfully streamlines the operations of leasing through a structured, automated Salesforce application. It improves efficiency, communication, and data accuracy for both admins and tenants.

## APPENDIX

* **Source Code:** Provided in Apex Classes and Triggers

**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');

}

}

}

}

**MothlyEmailScheduler.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});

}

}

}