

LEASE MANAGEMENT

College Name: Sree Narayana Guru College

College Code: bru36

TEAM ID: NM2025TMID26637

TEAM MEMBERS: 4

- Team Leader Name: Sreelakshmi R

Email: sreelakshmiramesh296@gmail.com

- Team Member1: Sreeram R

Email: sreeramvava10@gmail.com

- Team Member: Suchithra S

Email: kshasubhi163@gmail.com

- Team Member: Suhaibshan S

Email: suhaibshan27@gmail.com

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.



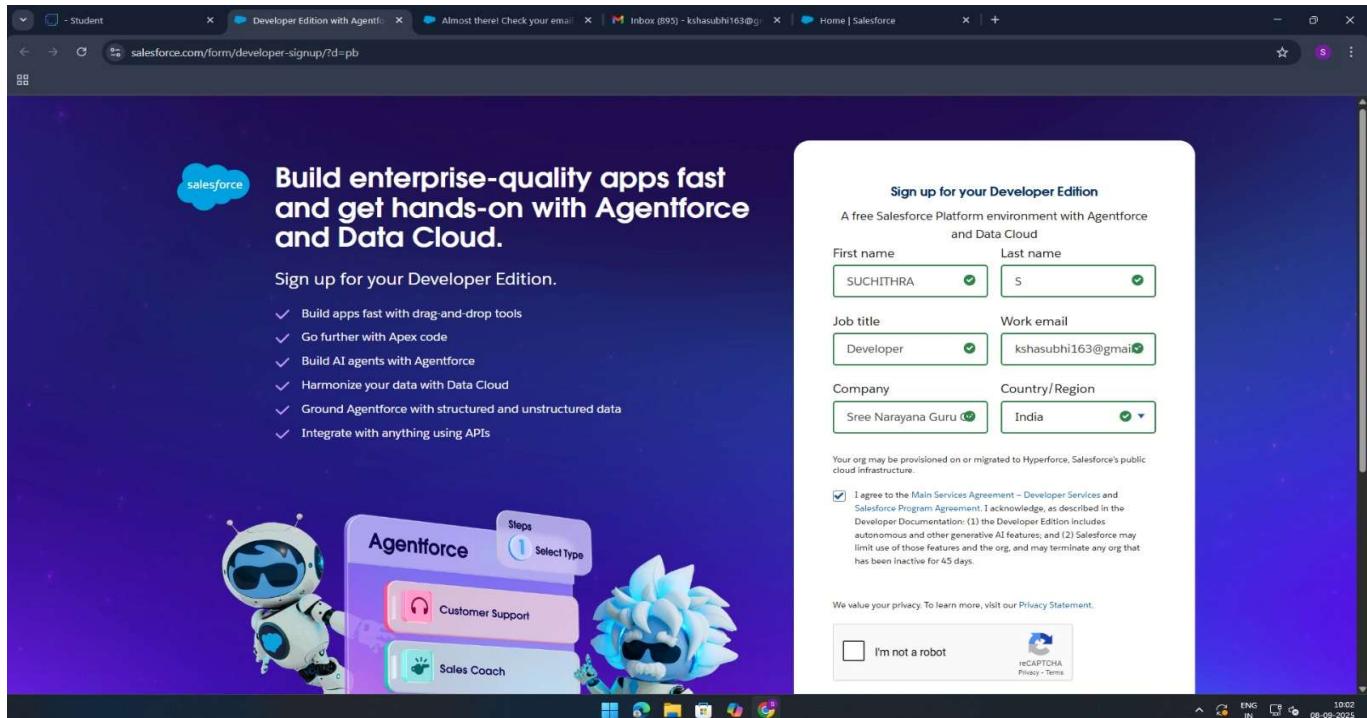
1.2 Purpose

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

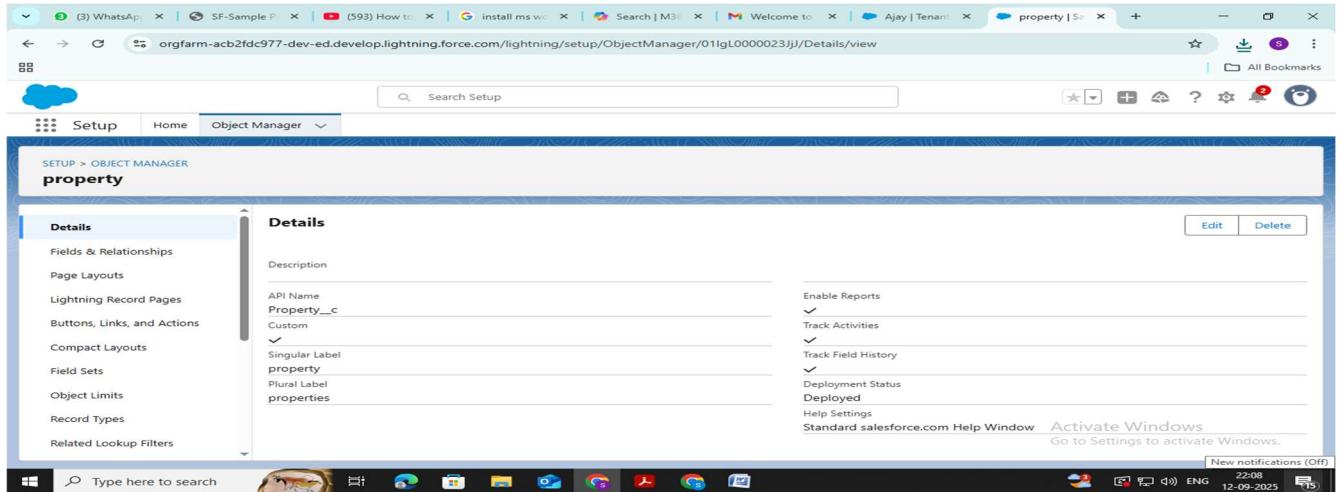
Creating Developer Account:

By using this URL - <https://www.salesforce.com/form/developer-signup/?d=pb>

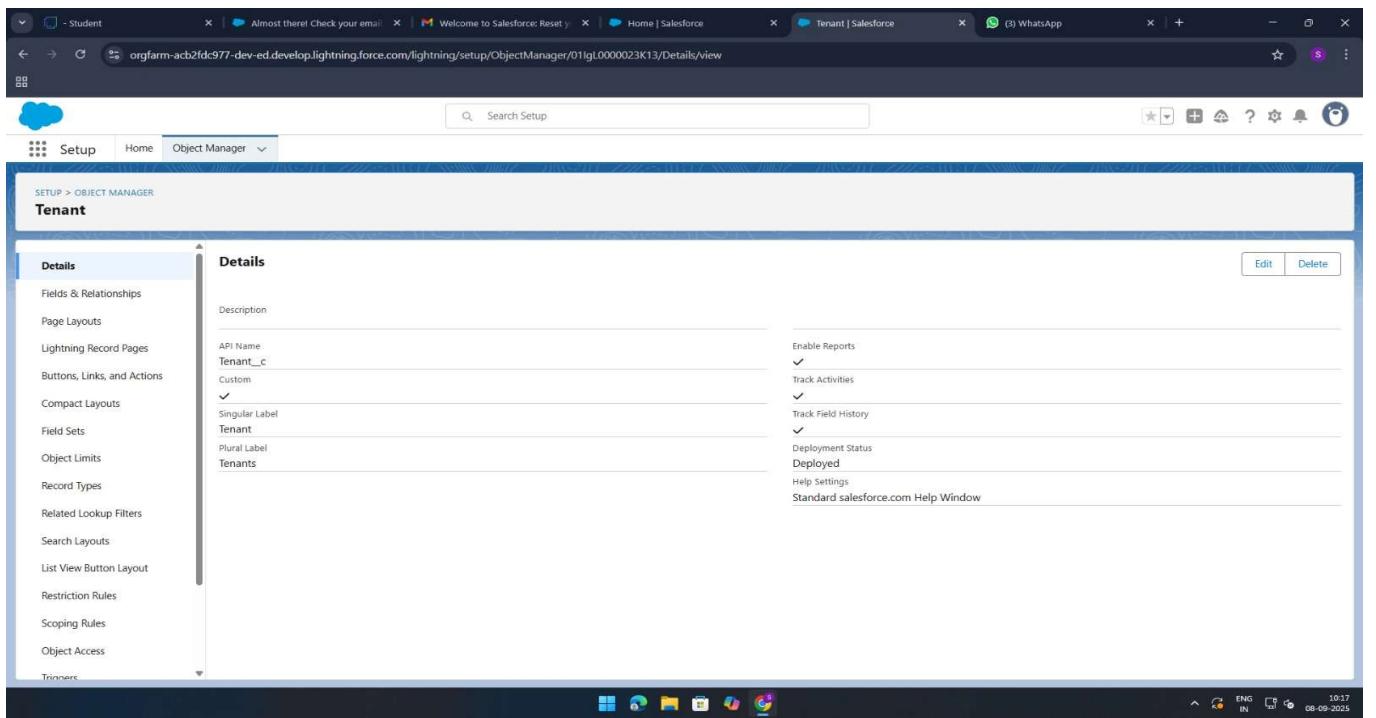


2.Created objects

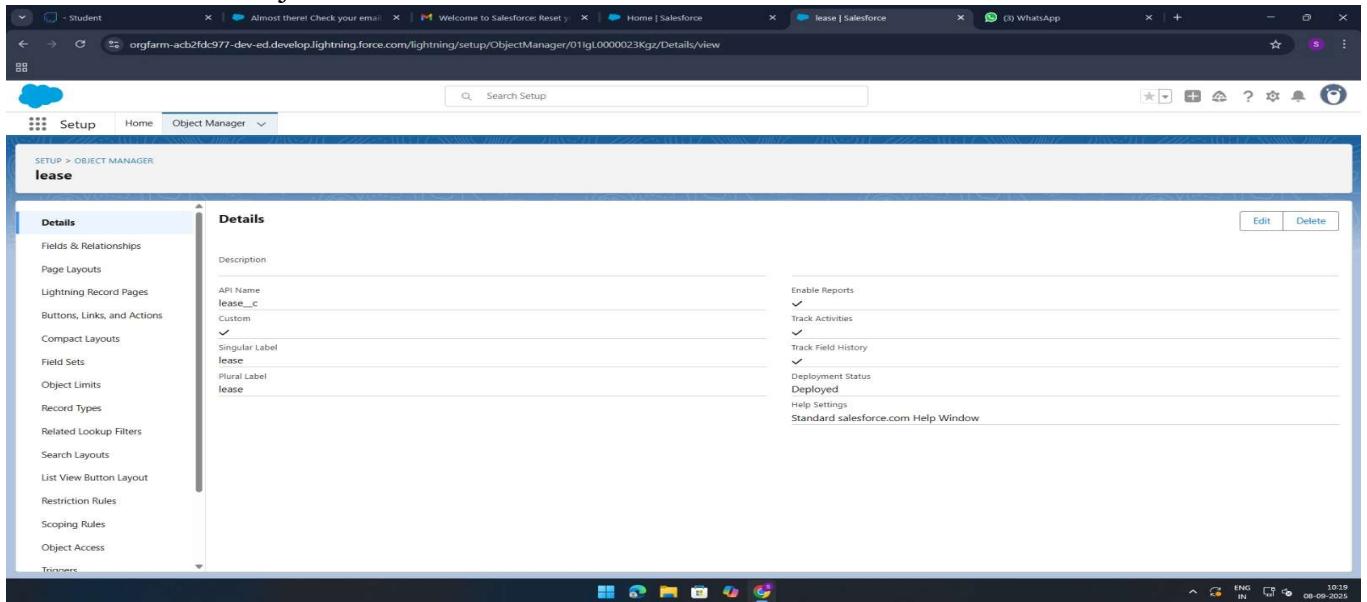
2.1 Create Property object



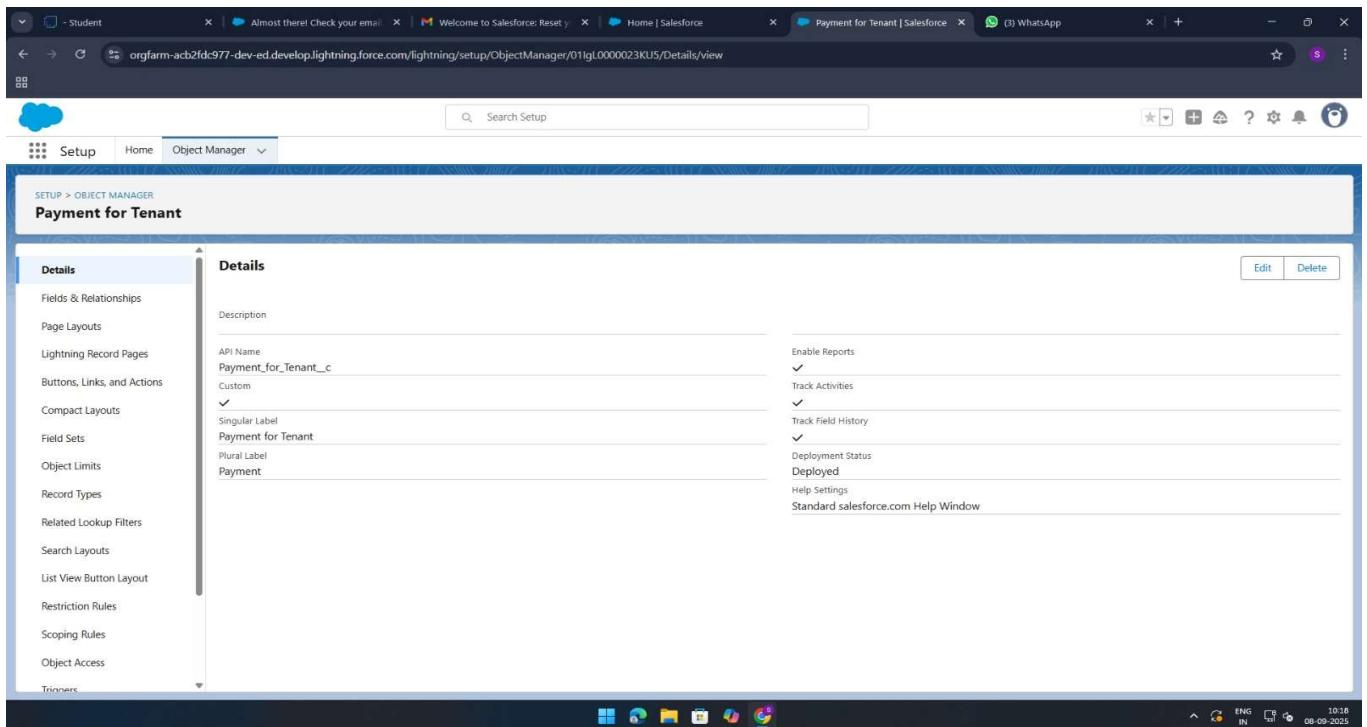
2.2 Create Tenant object



2.3 Create lease object



2.4 Create payment for tenant object



3.Tabs

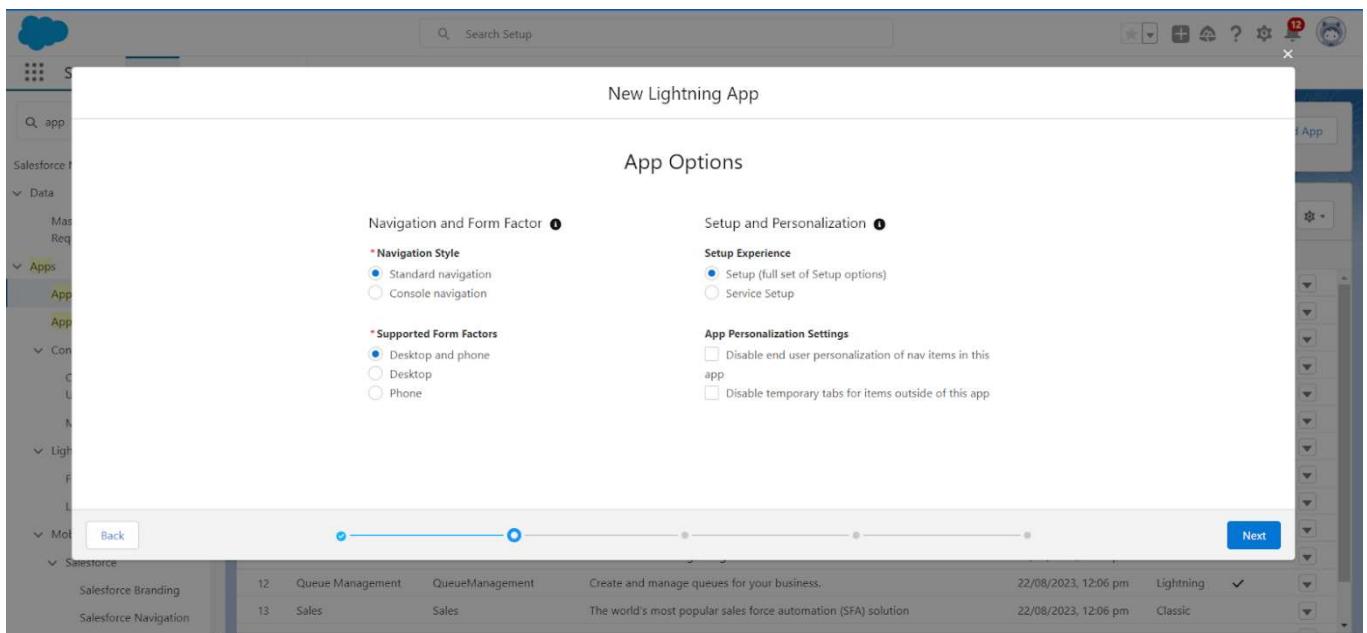
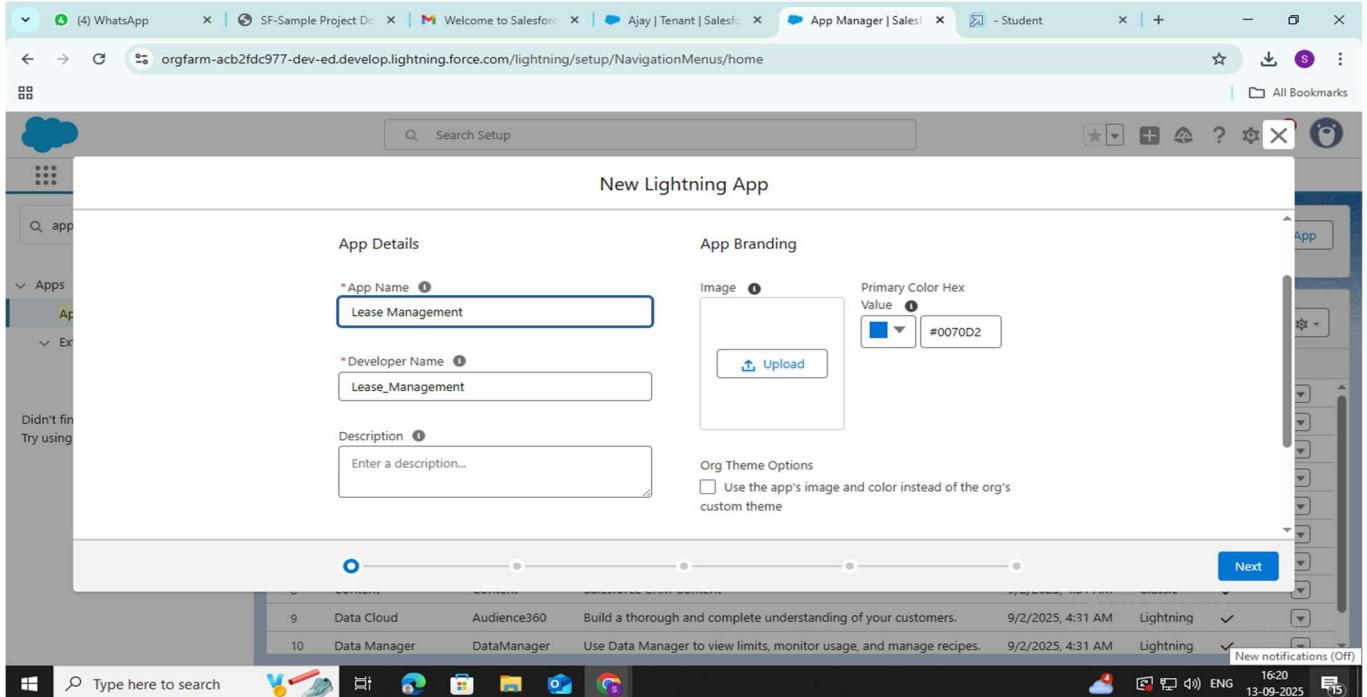
Creating custom tabs with tab styles

The screenshot shows the Salesforce Setup interface with the following details:

- Header:** Includes the Salesforce logo, a search bar labeled "Search Setup", and various navigation icons.
- Left Sidebar:** Shows a tree structure with "User Interface" expanded, "Rename Tabs and Labels" selected, and "Tabs" highlighted.
- Page Content:**
 - Custom Tabs:** A section with a sub-header "Custom Object Tabs". It contains a table with four rows:

Action	Label	Tab Style	Description
Edit Del	lease	Cup	
Edit Del	Payment	Books	
Edit Del	properties	Airplane	
 - Web Tabs:** A section stating "No Web Tabs have been defined".
 - Visualforce Tabs:** A section stating "No Visualforce Tabs have been defined".
 - Lightning Component Tabs:** A section with a "New" button and "What Is This?" link.

4.Developed Lightning App with relevant tabs



New Lightning App

Navigation Items

Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename the items that you add. Some navigation items are available only for phone or only for desktop. These items are dropped from the navigation bar when the app is viewed in a format that the item doesn't support.

Available Items

Selected Items

Back Next

13 Lightning Usage App LightningInstrumentation View Adoption and Usage Metrics for Lightning Experience 9/2/2025, 4:31 AM Lightning ✓
14 Marketing CRM Classic Marketing Track sales and marketing efforts with CRM objects. 9/2/2025, 4:31 AM Classic ✓
15 My Service Journey MSIAnn Discover new customer service capabilities. 9/2/2025, 4:31 AM Lightning ✓

Lightning App Builder App Settings Pages Lease Management

User Profiles

Choose the user profiles that can access this app.

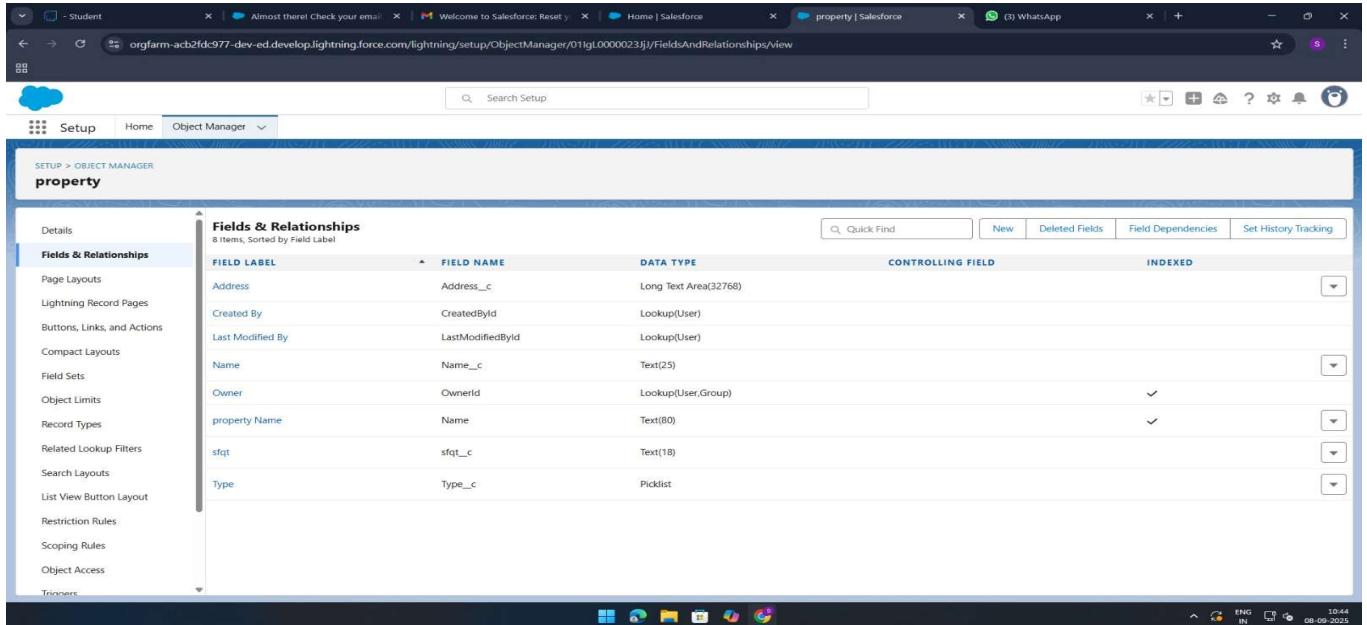
Available Profiles

Selected Profiles

System Administrator

5. Configured fields and relationships

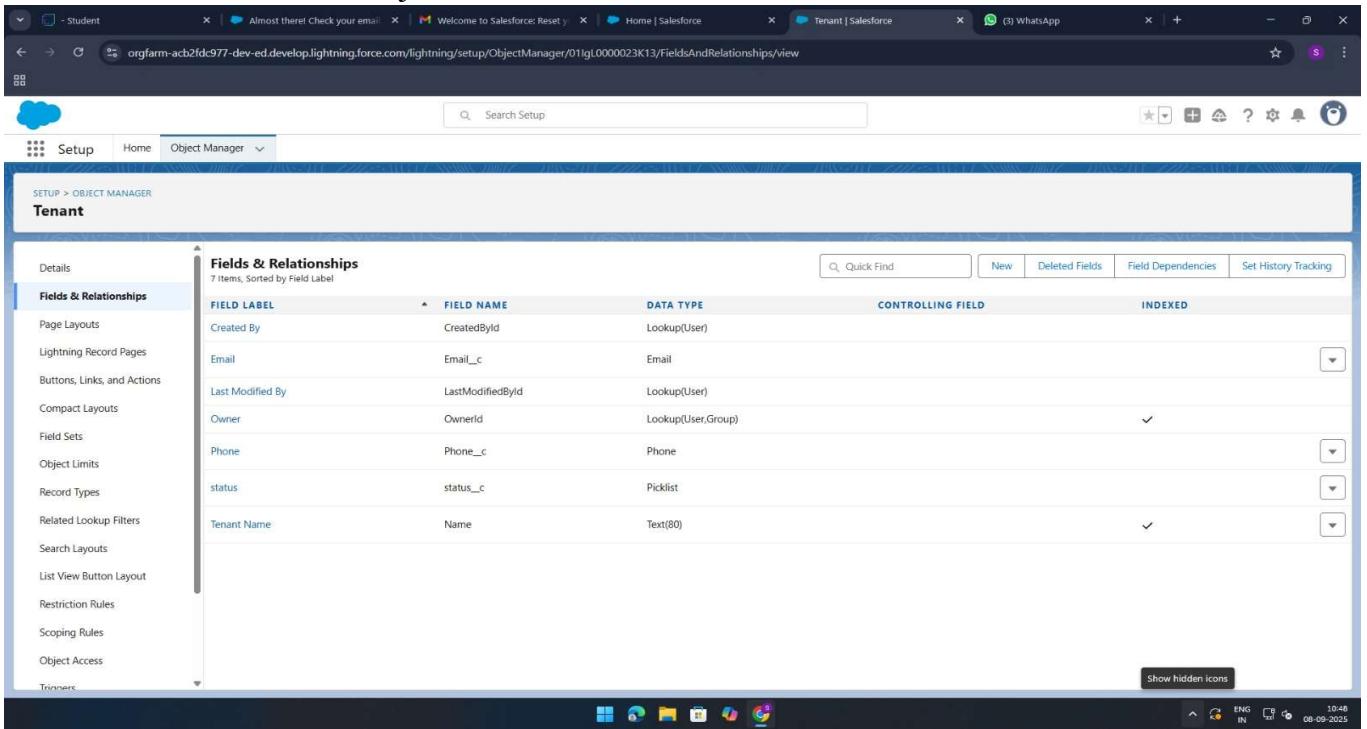
5.1 Create fields for Property object



The screenshot shows the Salesforce Object Manager interface for the 'property' object. The left sidebar lists various configuration options like Page Layouts, Lightning Record Pages, and Field Sets. The main area displays the 'Fields & Relationships' section with 8 items. The table shows the following fields:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address__c	Long Text Area(32768)		
Created By	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Name	Name__c	Text(25)		
Owner	OwnerId	Lookup(User,Group)		✓
property Name	Name	Text(80)		✓
sfqt	sfqt__c	Text(18)		
Type	Type__c	Picklist		

5.2 Create fields for Tenant object



The screenshot shows the Salesforce Object Manager interface for the 'Tenant' object. The left sidebar lists various configuration options. The main area displays the 'Fields & Relationships' section with 7 items. The table shows the following fields:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
Email	Email__c	Email		
Last Modified By	LastModifiedBy	Lookup(User)		
Owner	OwnerId	Lookup(User,Group)		✓
Phone	Phone__c	Phone		
status	status__c	Picklist		
Tenant Name	Name	Text(80)		✓

5.3 Create fields for lease object

The screenshot shows the Salesforce Object Manager interface for the 'lease' object. On the left, a sidebar lists various setup options like Page Layouts, Lightning Record Pages, and Field Sets. The main area displays a table titled 'Fields & Relationships' with the following data:

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Created By	CreatedBy	Lookup(User)		
End date	End_date_c	Date		
Last Modified By	LastModifiedBy	Lookup(User)		
lease Name	Name	Text(80)		✓
Owner	OwnerId	Lookup(User,Group)		✓
property	property_c	Lookup(property)		✓
start date	start_date_c	Date		

5.4 Create fields for Payment for tenant object

The screenshot shows the Salesforce Object Manager interface for the 'Payment for Tenant' object. On the left, a sidebar lists various setup options. The main area displays a table titled 'Fields & Relationships' with the following data:

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	CreatedBy	Lookup(User)		
Last Modified By	LastModifiedBy	Lookup(User)		
Payment date	Payment_date_c	Date		
Payment Name	Name	Text(80)		✓
property	property_c	Master-Detail(property)		✓

6. Validation Rule

- To create a validation rule to a Lease Object

The screenshot shows the 'Validation Rule Edit' screen for the 'lease' object. The 'Rule Name' is set to 'lease_end_date'. The 'Active' checkbox is checked. The 'Error Condition Formula' is defined as 'End_date__c > start_date__c'. A tooltip for the formula indicates it displays an error if the end date is more than 30% greater than the start date. The 'Functions' dropdown menu is open, showing various mathematical functions like ABS, ACOS, ADDMONTHS, ATAN, ASCII, ASIN, etc.

The screenshot shows the 'Validation Rules' list for the 'lease' object. There is one item listed: 'lease_end_date' with an error message 'Your End date must be greater than start date'. The rule is active and was modified by 'SUCHITHRA S.' on 9/13/2025 at 4:32 AM.

Rule Name	Error Location	Error Message	Active	Modified By
lease_end_date	start date	Your End date must be greater than start date	✓	SUCHITHRA S., 9/13/2025, 4:32 AM

7. Email Template

- Built and tested email templates for tenant leave , leave approval, rejection, monthly payment, and successful payment

7.1

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template name is 'Leave approved'. The details pane shows the following information:

Email Template from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Leave approved
Template Unique Name	Leave_approved
Encoding	Unicode (UTF-8)
Author	SUCHITHRA S [Change]
Description	
Created By	SUCHITHRA S, 9/7/2025, 10:40 PM
Modified By	SUCHITHRA S, 9/7/2025, 10:40 PM

The email template body contains the following text:

Subject: Leave approved
Plain Text Preview:
dear[Tenant__c_Name],
I hope this message finds you well. I am writing to inform you that I have received your email confirming the approval of my leave request. I would like to express my gratitude for considering and approving my time off.
your leave is approved. You can leave now

7.2

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template name is 'tenant leaving'. The details pane shows the following information:

Email Template from Salesforce	Unfiled Public Classic Email Templates
Email Template Name	Tenant_leaving
Template Unique Name	tenant_leaving
Encoding	Unicode (UTF-8)
Author	SUCHITHRA S [Change]
Description	
Created By	SUCHITHRA S, 9/7/2025, 10:38 PM
Modified By	SUCHITHRA S, 9/7/2025, 10:38 PM

The email template body contains the following text:

Subject: request for approve the leave
Plain Text Preview:
Dear ({Tenant__c_CreatedBy}),
Please approve my leave

7.3

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template 'Leave rejected' is selected. The template detail section shows the following details:

Email Template Detail	Unified Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	SUCHITHRA S [Change]
Description	
Created By	SUCHITHRA S 9/8/2025, 9:11 AM
Modified By	SUCHITHRA S 9/8/2025, 9:11 AM

The email template preview shows the subject 'Leave rejected' and the plain text content:

Subject : Leave rejected

Plain Text Preview

Dear [Tenant__c.Name],

I hope this email finds you well. Your contract has not ended. So we can't approve your leave
your leave has rejected

Activate Windows
Go to Settings to activate Windows.

Windows taskbar at the bottom with various icons and the date/time: 13-09-2025 22:55

7.4

The screenshot shows the Salesforce Setup interface with the 'Classic Email Templates' page open. The template 'Tenant Email' is selected. The template detail section shows the following details:

Email Template Detail	Unified Public Classic Email Templates
Email Template Name	Tenant_Email
Template Unique Name	Tenant_Email
Encoding	Unicode (UTF-8)
Author	SUCHITHRA S [Change]
Description	
Created By	SUCHITHRA S 9/8/2025, 9:15 AM
Modified By	SUCHITHRA S 9/8/2025, 9:15 AM

The email template preview shows the subject 'Urgent: Monthly Rent Payment Reminder' and the plain text content:

Subject : Urgent: Monthly Rent Payment Reminder

Plain Text Preview

Dear [Tenant__c.Name],

I trust this email finds you well. We appreciate your continued tenancy at our property and I hope you have been comfortable in your residence.

This communication is a friendly reminder regarding your monthly rent payment, which is currently outstanding. As outlined in our rental agreement, the payment is due. To ensure the smooth operation of our property management and to avoid any inconvenience, we kindly request you to settle the payment at your earliest convenience.

Activate Windows
Go to Settings to activate Windows.

Windows taskbar at the bottom with various icons and the date/time: 13-09-2025 23:01

8. Approval Process creation

8.1 Check for Vacant:

The screenshot shows the Salesforce Setup interface. In the left sidebar, under 'Data' > 'Process Automation', 'Approval Processes' is selected. A search bar at the top left contains 'approval'. The main content area is titled 'Approval Processes' and includes a 'Open Approvals App' button. Below it is a help panel with steps to implement approvals. A dropdown menu 'Manage Approval Processes For:' is set to 'Tenant'. The 'Active Approval Processes' section lists one process named 'check for vacant' with a process order of 1. The 'Inactive Approval Processes' section shows 'No approval processes available'.

8.2 approval process details

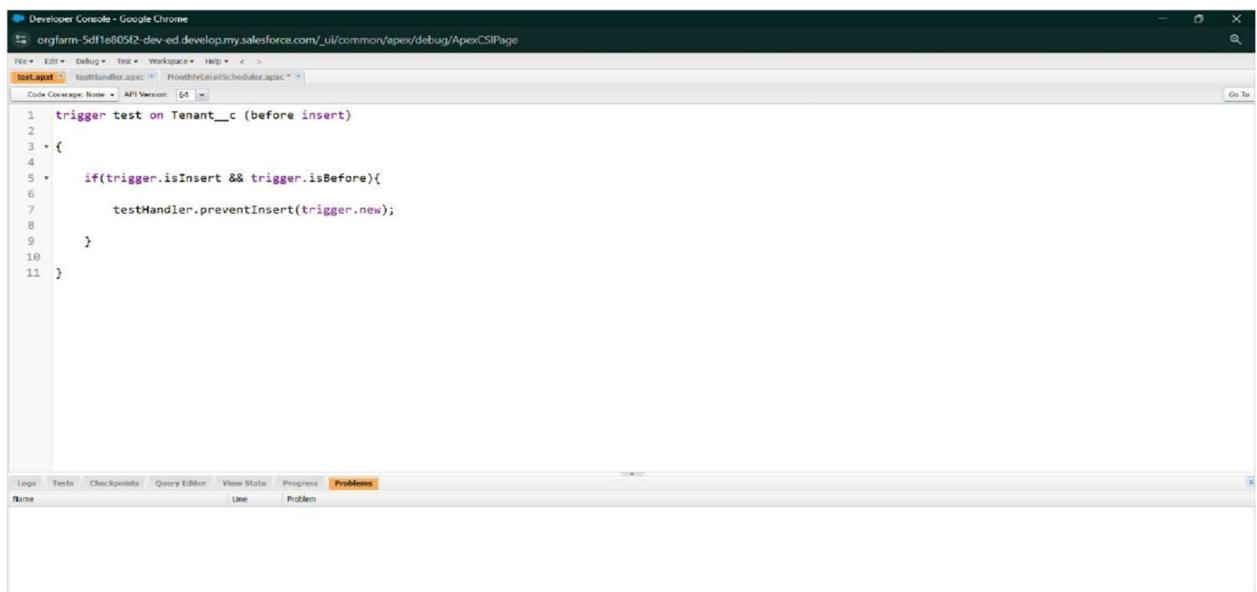
The screenshot shows the 'Approval Processes' detail page for the 'check for vacant' process. The process definition detail includes the following fields:

- Process Name: check for vacant
- Unique Name: check_for_vacant
- Description: check for vacant
- Entry Criteria: Tenants: status NOT EQUAL TO Leaving
- Record Editability: Administrator ONLY
- Approval Assignment Email Template: (empty)
- Initial Submitter: Tenant Owner
- Created By: SUCHITHRA.S (9/9/2025, 8:52 AM)
- Modified By: SUCHITHRA.S (9/9/2025, 9:50 AM)

The 'Initial Submission Actions' section contains a 'Record Lock' entry. The 'Approval Steps' section is empty. The 'Final Approval Actions' section is also empty. A note at the bottom right says 'Activate Windows' and 'Go to Settings to activate Windows.'

9.Apex Trigger

9.1 Create an Apex Trigger

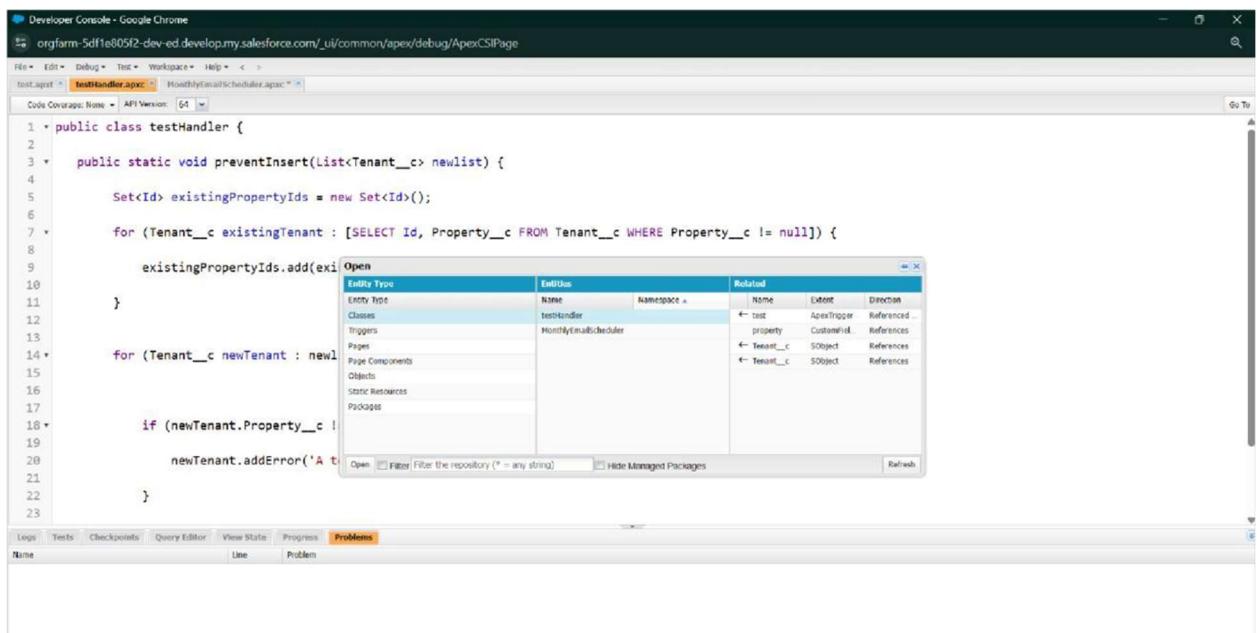


The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-5df1e805f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tab bar shows 'testHandler.apc' is selected. The code editor contains the following Apex trigger:

```
trigger test on Tenant__c (before insert)
{
    if(trigger.isInsert && trigger.isBefore){
        testHandler.preventInsert(trigger.new);
    }
}
```

The console interface includes tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is currently active.

9.2 Create an Apex Handler class



The screenshot shows the Salesforce Developer Console in Google Chrome. The URL is orgfarm-5df1e805f2-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The tab bar shows 'testHandler.apc' is selected. The code editor contains the following Apex Handler class:

```
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.Id);
        }
        for (Tenant__c newTenant : newList) {
            if (newTenant.Property__c != null) {
                newTenantaddError('A');
            }
        }
    }
}
```

A modal window titled 'Open' is displayed, listing dependencies for the class. The table shows:

Entity Type	Name	Namespace	Related
Classes	testHandler	MonthlyEmailScheduler	↳ test ApexTrigger References ↳ property CustomField References ↳ Tenant__c SObject References ↳ Tenant__c SObject References
Triggers			
Pages			
Page Components			
Objects			
Static Resources			
Packages			

The modal also includes 'Open', 'Filter', 'Hide Managed Packages', and 'Refresh' buttons.

Developer Console - Google Chrome
 orgfarm-5df1e805f2-dev-ed.develop.my.salesforce.com/ui/common/apex/debug/ApexCSIPage

```

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
14         for (Tenant__c newTenant : newList) {
15
16
17             if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
18
19                 newTenant.addError('A tenant can have only one property');
20
21             }
22
23         }
24     }
25
  
```

Logs Tests Checkpoints Query Editor View State Progress Problems

9.3 Testing the trigger

Lease Management Payment ▾

Tenants ▾ property lease ▾

New Tenant

* = Required Information

Tenant Name	chinnu	Owner	Sowmya Team
Email	chinnu@gmail.com		
Phone			
status	Stay		
property	Parkside	We hit a snag. Review the errors on this page. • A tenant can have only one property	
<input type="button" value="Cancel"/> <input type="button" value="Save & New"/> <input type="button" value="Save"/>			

10. Flows

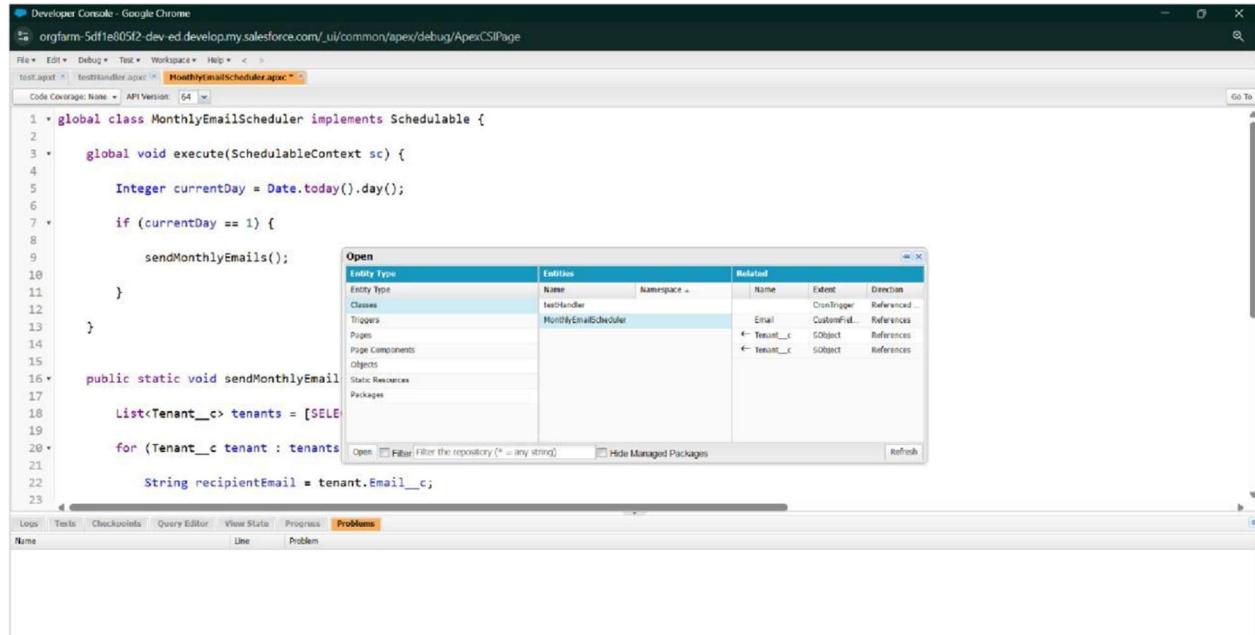
- Create flow for monthly payments

The screenshot shows the Flow Builder interface for a "monthly payment - V1" flow. The flow starts with a "Record-Triggered Flow" (Start) object set to trigger on "A record is updated" for the "Payment for Tenant" object. The flow then branches to an "End" node. In the middle pane, the "Configure Start" dialog is open, showing the selected object and trigger settings. The "Configure Trigger" section shows the trigger is set to "A record is updated". The "Set Entry Conditions" section contains a condition requirement "All Conditions Are Met (AND)" with a single condition: "Field check for payment Equals A Paid".

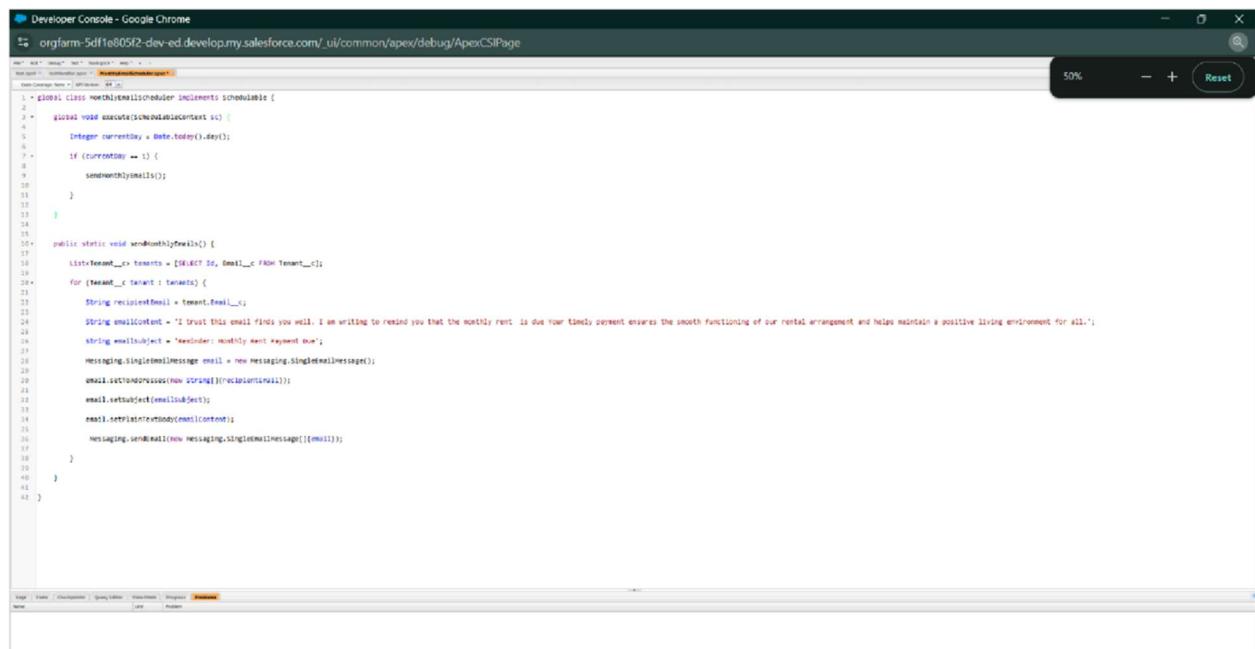
This screenshot is identical to the one above, showing the same flow configuration and dialog boxes. The flow structure, trigger settings, and entry conditions are all the same.

11.Schedule class:

11.1 Create an Apex Class



```
1 global class MonthlyEmailScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8
9             sendMonthlyEmails();
10        }
11    }
12
13 }
14
15 public static void sendMonthlyEmails() {
16
17     List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18
19     for (Tenant__c tenant : tenants) {
20
21         String recipientEmail = tenant.Email__c;
22
23     }
24 }
```



```
1 @isTest class MonthlyEmailScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8
9             sendMonthlyEmails();
10        }
11    }
12
13 }
14
15 public static void sendMonthlyEmails() {
16
17     List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18
19     for (Tenant__c tenant : tenants) {
20
21         String recipientEmail = tenant.Email__c;
22
23         String emailContent = 'I trust this email finds you well. I am writing to remind you that the monthly rent is due now. Timely payment ensures the smooth functioning of our rental arrangement and helps maintain a positive living environment for all.';
24
25         String emailSubject = 'Reminder: monthly rent payment due';
26
27         Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
28
29         email.setToAddresses(new String[]{recipientEmail});
30
31         email.setSubject(emailSubject);
32
33         email.setPlainTextEmailContent();
34
35         Messaging.SingleEmailMessage[] emails = new Messaging.SingleEmailMessage[1];
36
37         emails[0] = email;
38
39         Messaging.sendEmail(emails);
40
41     }
42 }
```

11.2 Schedule Apex class

The screenshot shows the Salesforce Setup Apex Classes page. The search bar at the top contains "apex". The left sidebar has sections for Email, Custom Code, Environments, and a "Didn't find what you're looking for?" section. Under Custom Code, Apex Classes is selected. The main area displays the Apex Class Detail for "MonthlyEmailScheduler". The class body contains the following code:

```
1 global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8
9     public static void sendMonthlyEmails() {
10        List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
11        for (Tenant__c tenant : tenants) {
12            ...
13        }
14    }
15
16    public static void sendMonthlyEmails() {
17        List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
18        for (Tenant__c tenant : tenants) {
19            ...
20        }
21    }
22}
```

The class summary table shows the following details:

Name	MonthlyEmailScheduler	Status	Active
Created By	Sowmya Team	Code Coverage	0% (0/15)
		Last Modified By	Sowmya Team
			6/23/2025, 2:47 AM

11.3 Testing approval process

The screenshot shows the Salesforce Lease Management page. A green success message at the top right says "Tenant was submitted for approval." The main area displays a "Lease Management" record for "Aswini". The "Details" tab is selected. The form fields include:

- Tenant Name: Aswini
- Email: aswiniamaraadi15@gmail.com
- Phone: (905) 223-5567
- status: Leaving
- property: Imran

Below the form, it shows "Created By: Sowmya Team, 6/26/2025, 6:05 AM" and "Last Modified By: Sowmya Team, 6/26/2025, 11:06 PM". On the right side, there is an "Activity" panel with a "Upcoming & Overdue" section which states "No activities to show. Get started by sending an email, scheduling a task, and more." The system status bar at the bottom shows "30°C Cloudy" and the date "27-06-2025".

The screenshot shows a Salesforce Lightning page titled "Process Instance Step". The main title is "Tenant Approval" with a status of "Approved". Below the title, there are four fields: "Submitter" (Sowmya Team), "Date Submitted" (Jun 27, 2025), "Actual Approver" (Sowmya Team), and "Assigned To" (Sowmya Team). A "Details" section contains "Approval Details" with fields: "Tenant Name" (Aswini), "Owner" (Sowmya Team), "property" (Imran), and "Assigned To" (Sowmya Team). To the right, a "Notifications" sidebar lists five recent notifications:

- Approval request for the tenant is approved Aswini (a few seconds ago)
- Approval request for the tenant is rejected Aswini (an hour ago)
- Approval request for the tenant is approved Aswini (an hour ago)
- Approval request for the tenant is approved Imran (an hour ago)
- Approval request for the tenant is approved Kiran (19 hours ago)

This screenshot shows the same notifications as the top one, but from a mobile device. The notifications are identical:

- Approval request for the tenant is approved niranjan (a few seconds ago)
- Approval request for the tenant is rejected niranjan (Jun 23, 2025, 4:29 PM)

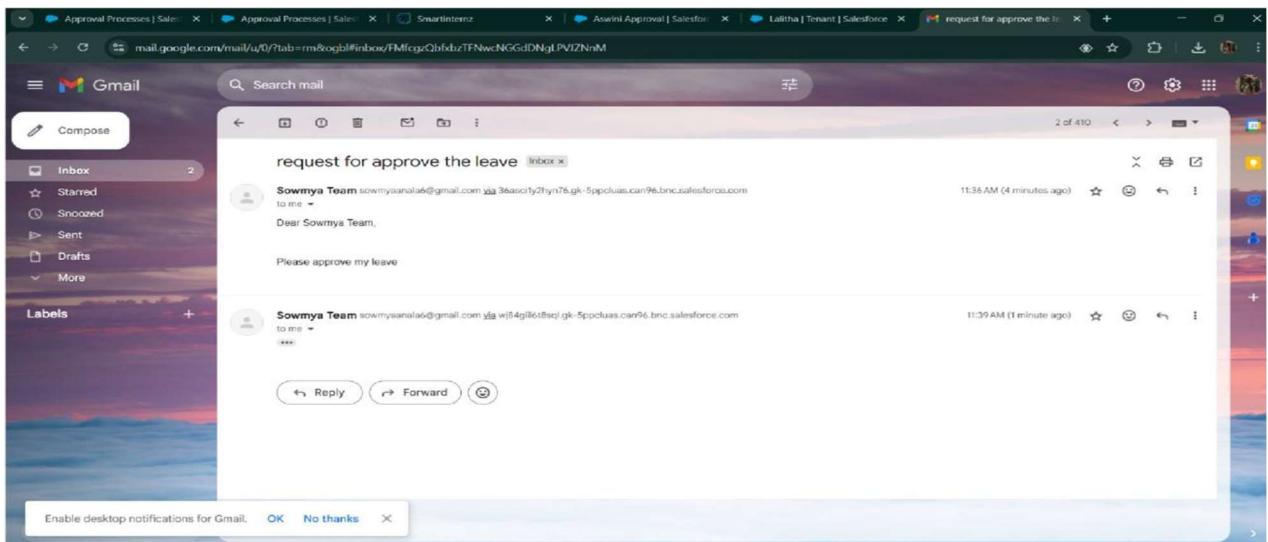
12.RESULTS

Output Screenshots

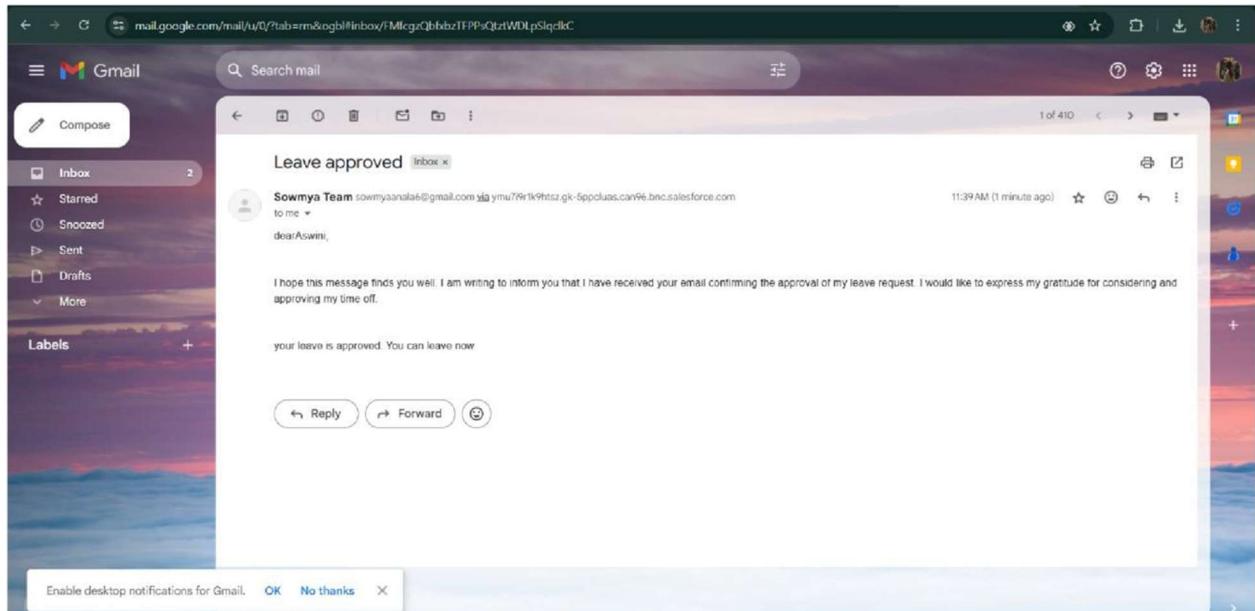
- Email alerts

Step Name	Date	Status	Assigned To	Actual Approver	Comments
1 Step 1	6/25/2025, 5:39 AM	Approved	Sowmya Team	Sowmya Team	approved
2 Approval Request Submitted	6/25/2025, 5:39 AM	Submitted	Sowmya Team	Sowmya Team	leaving
3 Step 1	6/23/2025, 3:59 AM	Rejected	Sowmya Team	Sowmya Team	Rejected
4 Approval Request Submitted	6/23/2025, 3:58 AM	Submitted	Sowmya Team	Sowmya Team	Leaving
5 Step 1	6/23/2025, 3:55 AM	Approved	Sowmya Team	Sowmya Team	Approved
6 Approval Request Submitted	6/23/2025, 3:55 AM	Submitted	Sowmya Team	Sowmya Team	leaving
7 Step 1	6/23/2025, 3:44 AM	Approved	Sowmya Team	Sowmya Team	Approval Approved
8 Approval Request Submitted	6/23/2025, 3:42 AM	Submitted	Sowmya Team	Sowmya Team	Leaving

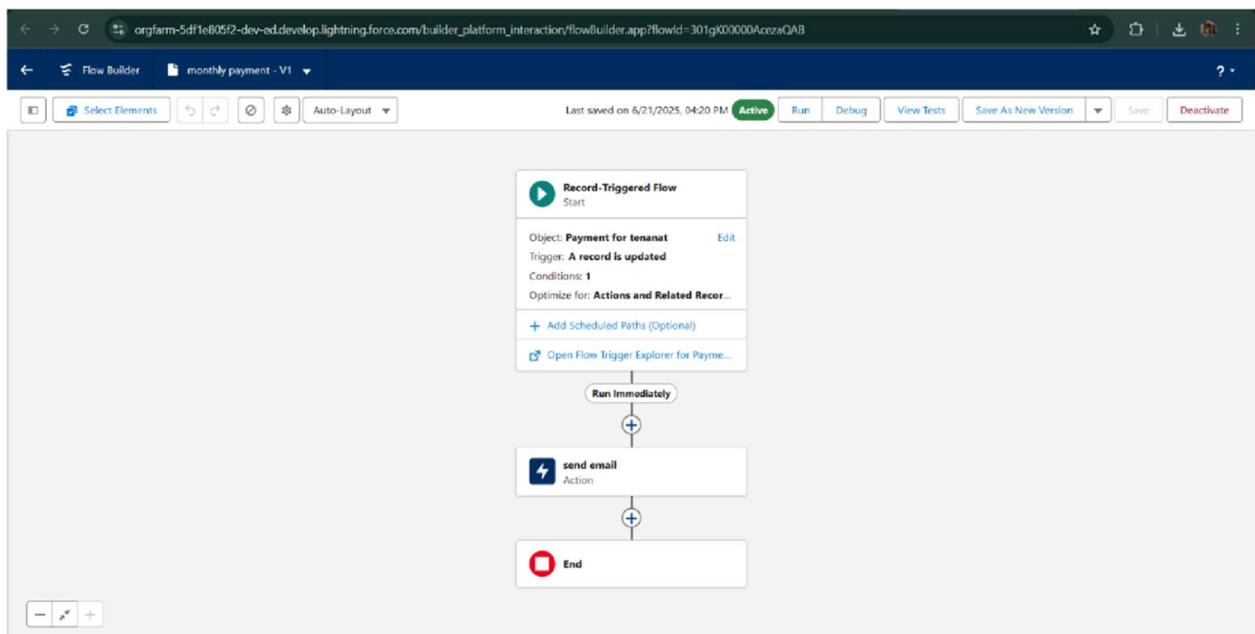
- Request for approve the leave



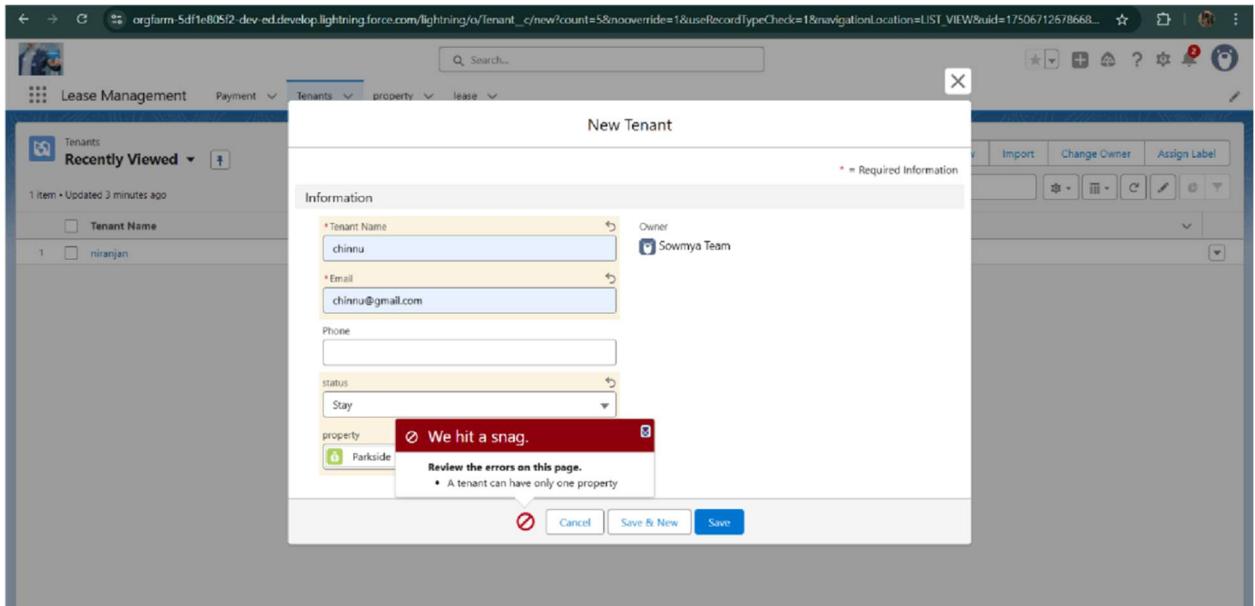
- Leave approved



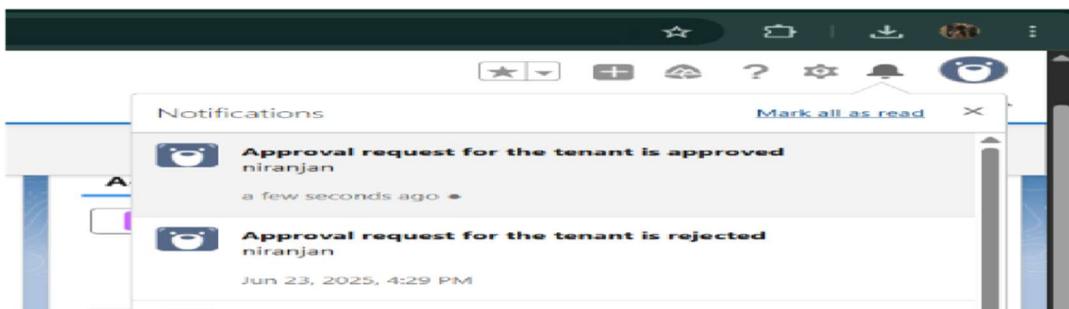
- Flow runs



- Trigger error messages



- Approval process notifications



13. 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)) {  
  
            newTenantaddError('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});
```

```
}
```

```
}
```

```
}
```