

# **Lease Management**

**College code: BRU26**

**College Name: SNMV College of Arts and  
Science**

**TEAM ID: NM2025TMID20678**

**TEAM MEMBERS: 4**

- **Team Leader Name: Paramasivam A**

**Email: paramasivams971@gmail.com**

- **Team Member1: Satish k**

- **Email: [bscit311@gmail.com](mailto:bscit311@gmail.com)**

- **Team Member2: Dheenadhayalan J**

**Email: deenadhayalanj468@gmail.com**

- **Team Member3: Mansoor Ahamed N**

**Email: mansoor93631@gmail.com**

# 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.**

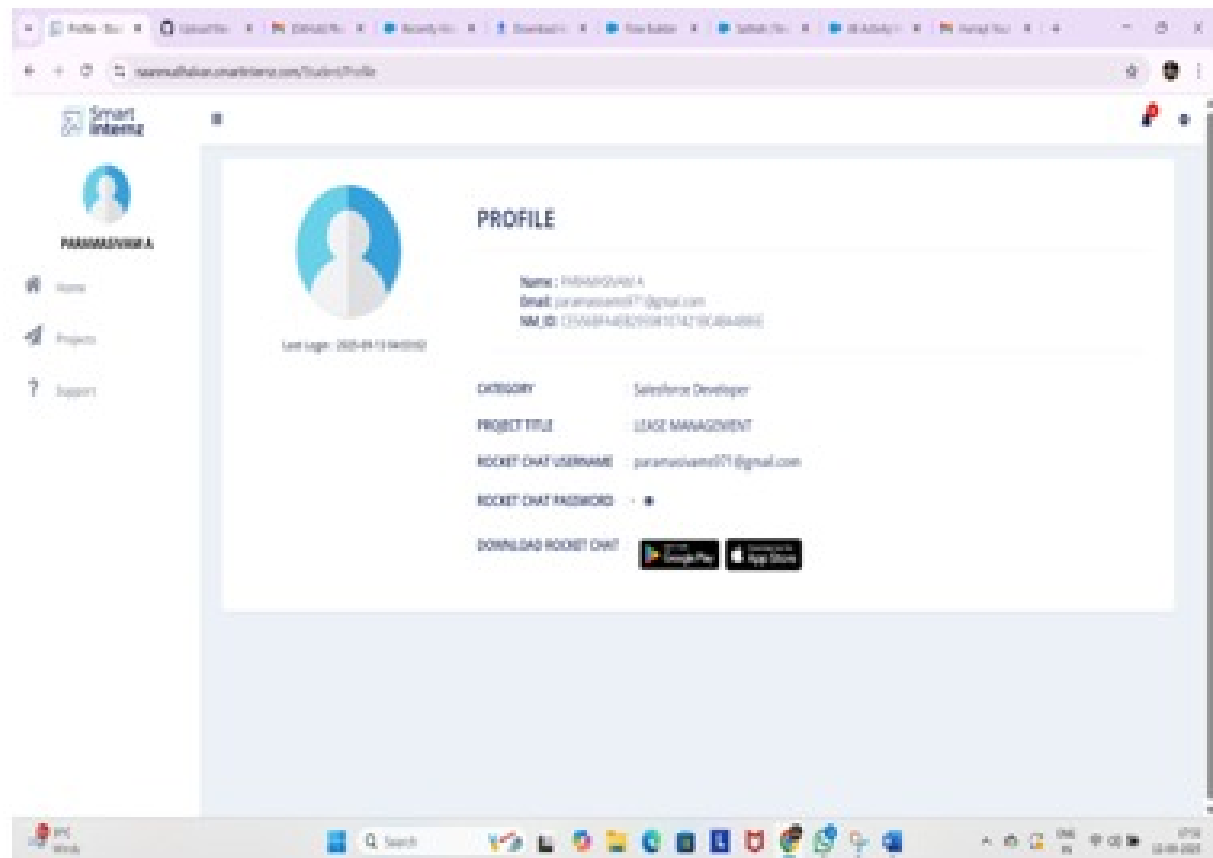


## 1.2 Purpose

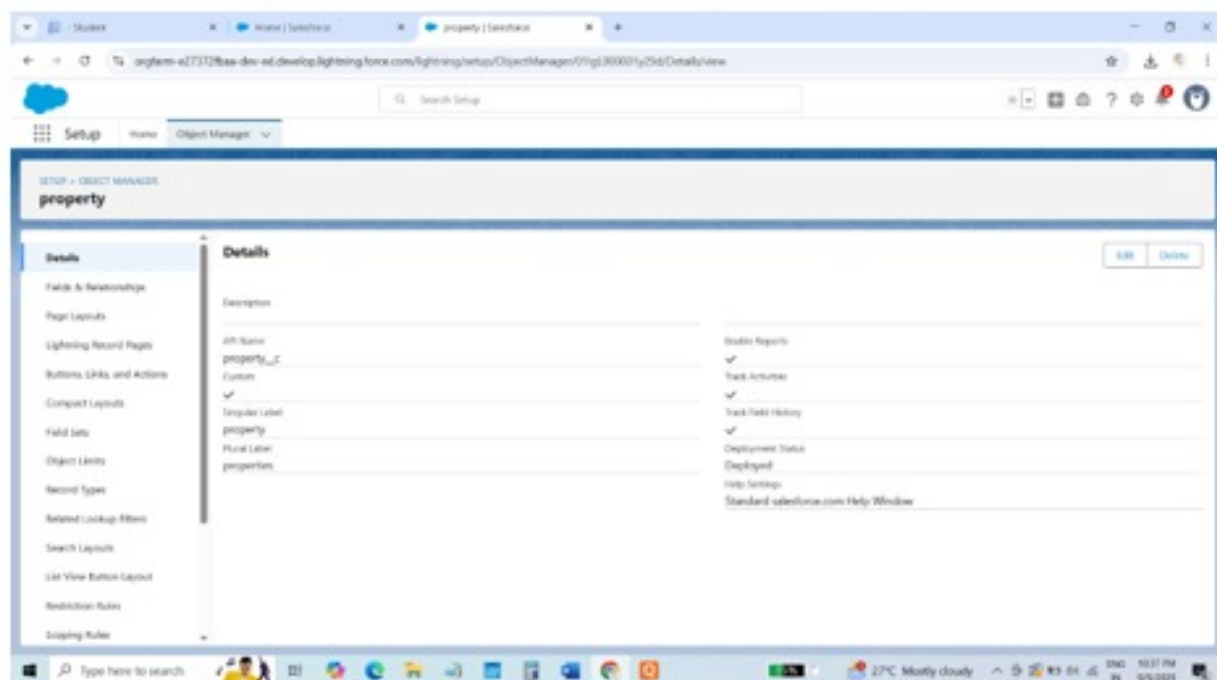
The main objective of the project is to enable organizations to efficiently manage properties, tenants, and lease-related activities.

# DEVELOPMENT PHASE

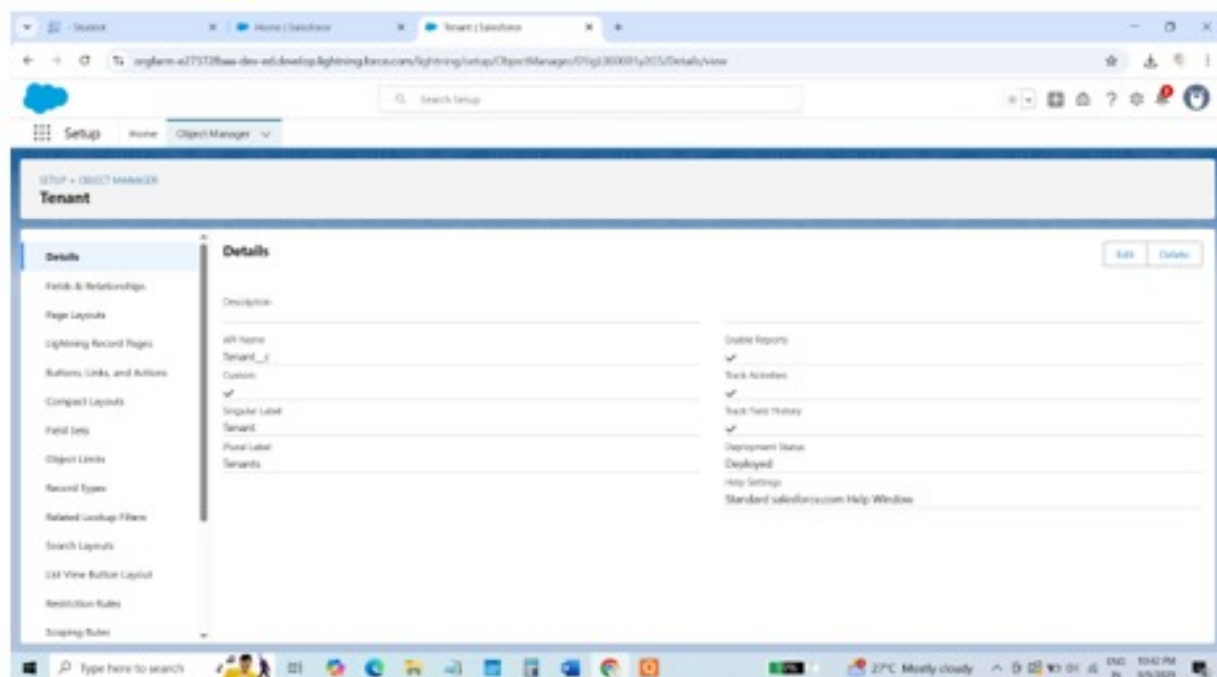
Creating Developer Account: By using this URL  
-<https://www.salesforce.com/form/developer-signup/?d=pb>



## ● Created objects: Property, Tenant, Lease ,Payment

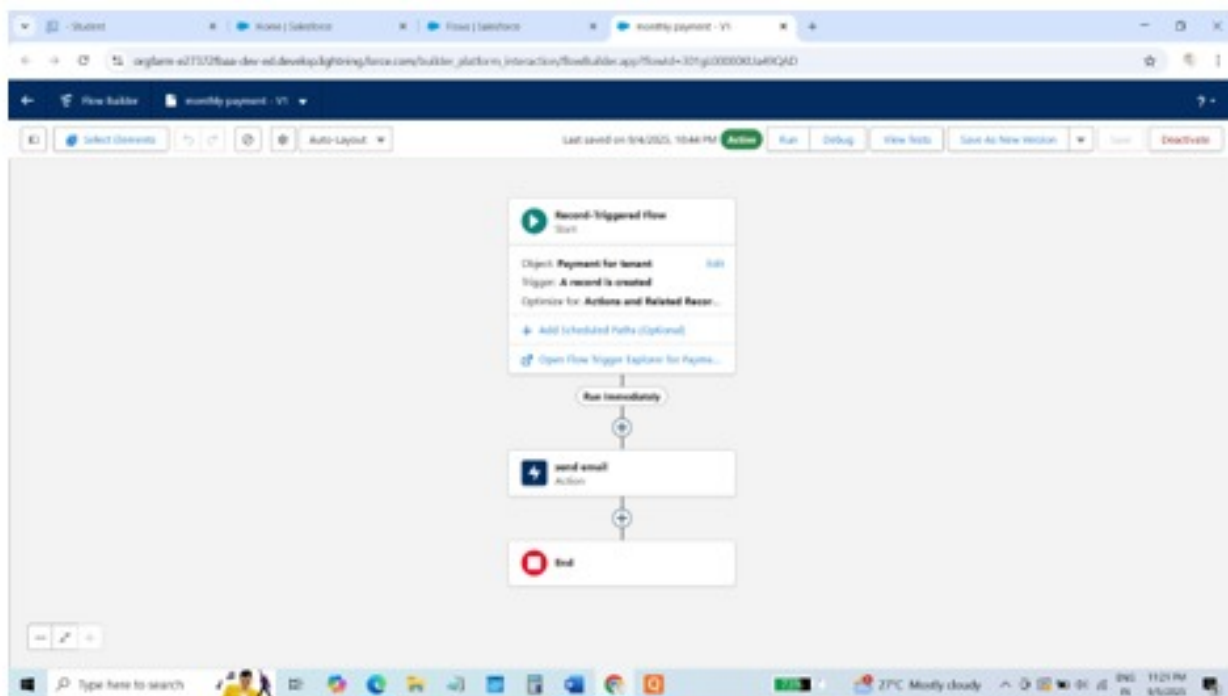


The screenshot shows the Salesforce Setup interface for the 'Property' object. The left sidebar contains a 'Details' menu with options like Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Links, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Sharing Rules. The main content area is titled 'Details' and includes a description field, API Name (property\_\_c), Custom checkbox, Singular Label (property), Plural Label (properties), and a list of features: Enable Reports (checked), Track Activities (checked), Track Field History (checked), Deployment Status (Deployed), and Help Settings (Standard Salesforce.com Help Window). Buttons for 'Edit' and 'Details' are in the top right.

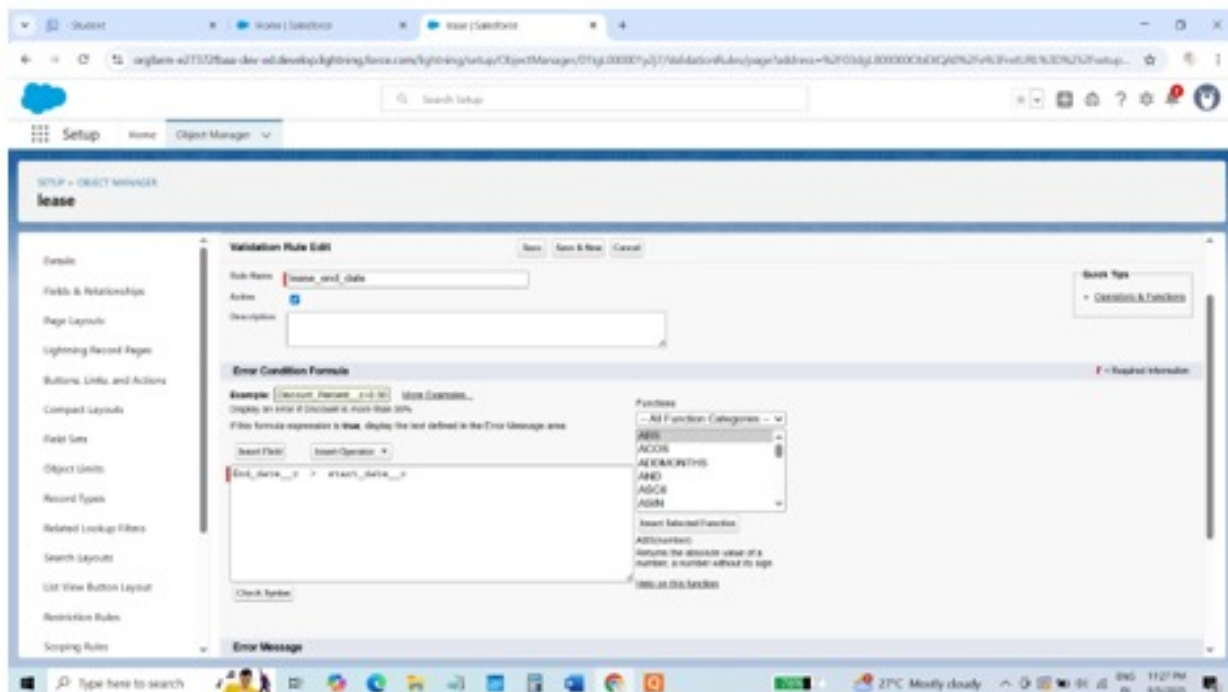


The screenshot shows the Salesforce Setup interface for the 'Tenant' object. The left sidebar contains a 'Details' menu with options like Fields & Relationships, Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Links, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, Restriction Rules, and Sharing Rules. The main content area is titled 'Details' and includes a description field, API Name (tenant\_\_c), Custom checkbox, Singular Label (tenant), Plural Label (tenants), and a list of features: Enable Reports (checked), Track Activities (checked), Track Field History (checked), Deployment Status (Deployed), and Help Settings (Standard Salesforce.com Help Window). Buttons for 'Edit' and 'Details' are in the top right.

## ● Implemented Flows for monthly rent and payment success



## ● To create a validation rule to a Lease Object



## • Developed Lightning App with relevant tabs

The screenshot displays the Lightning App Builder interface for configuring a Lightning App. The top navigation bar includes tabs for Lightning App Builder, App Settings, Pages, and Lease Management. The left sidebar shows the App Settings section with sub-tabs for App Details & Branding, App Options, Utility Items (Desktop Only), Navigation Items, and User Profiles. The main content area is titled "Navigation Items" and includes a description: "Choose the items to include in the app, and assign the roles in which they appear. Users can personalize the navigation to add or remove items, but users can't remove or rearrange 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 closed to a format that the device doesn't support."

The "Available Items" list on the left includes:

- Accounts
- Activation Targets
- Activations
- All Sites
- Alternative Payment Methods
- Analytics
- App Launcher
- Applicant Catalogs
- Applicant Installations
- Approval Requests
- Approved Submission Details

The "Selected Items" list on the right includes:

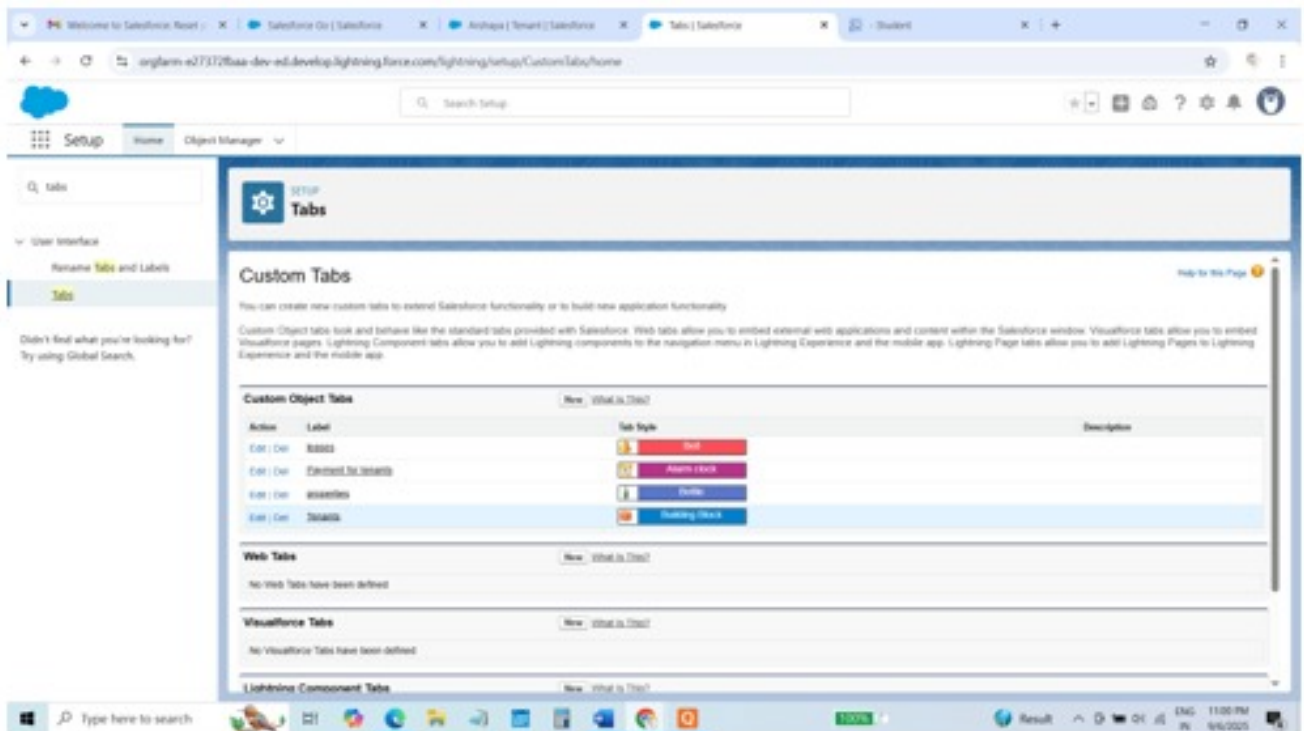
- Home
- Properties
- Payers for Brands
- Resorts
- Lease

The bottom screenshot shows the "App Details & Branding" configuration screen. It includes fields for App Name (Lease Management), Developer Name (Lease Management), and Description. The "App Branding" section shows the Primary Color Hex Value (#000000) and an Upload button for the app image. The "App Launcher Preview" shows a button labeled "LM" with the text "Lease Management".





## • Creating A Custom Tabs

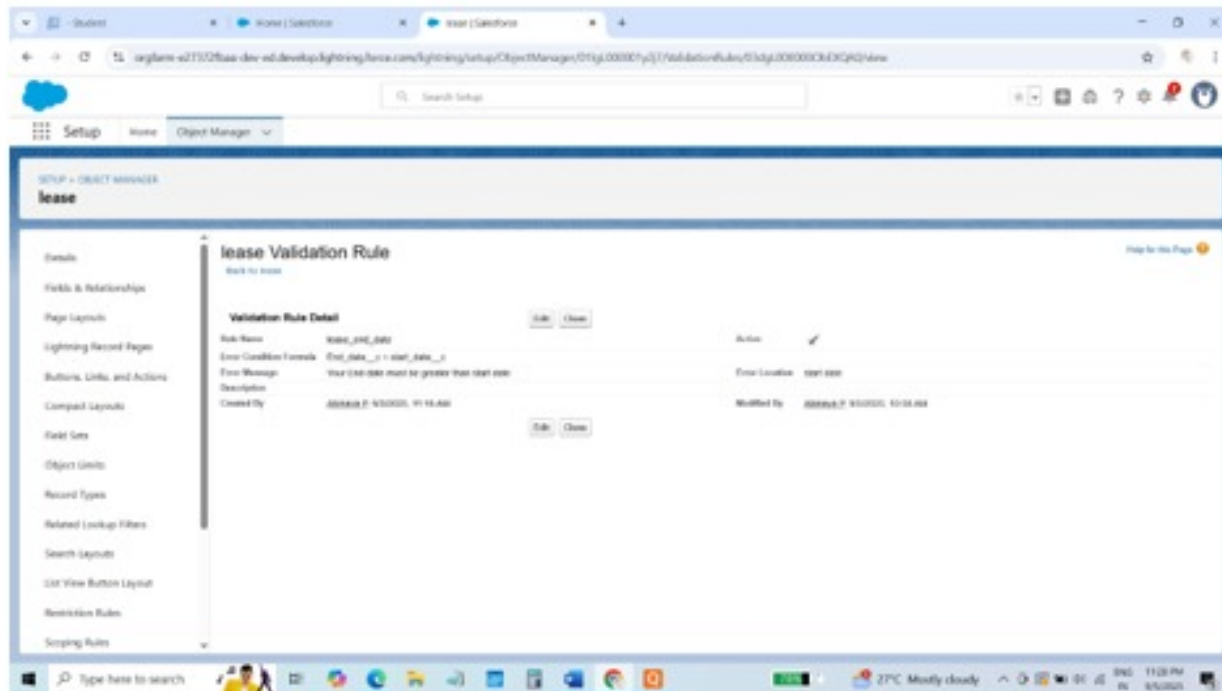


The screenshot shows the Salesforce Setup interface for Custom Tabs. The left sidebar contains navigation links for Setup, Home, and Object Manager. The main content area is titled 'Custom Tabs' and includes a 'Help for this Page' link. Below the title, there is a brief explanation of custom tabs and their types. The 'Custom Object Tabs' section contains a table with the following data:

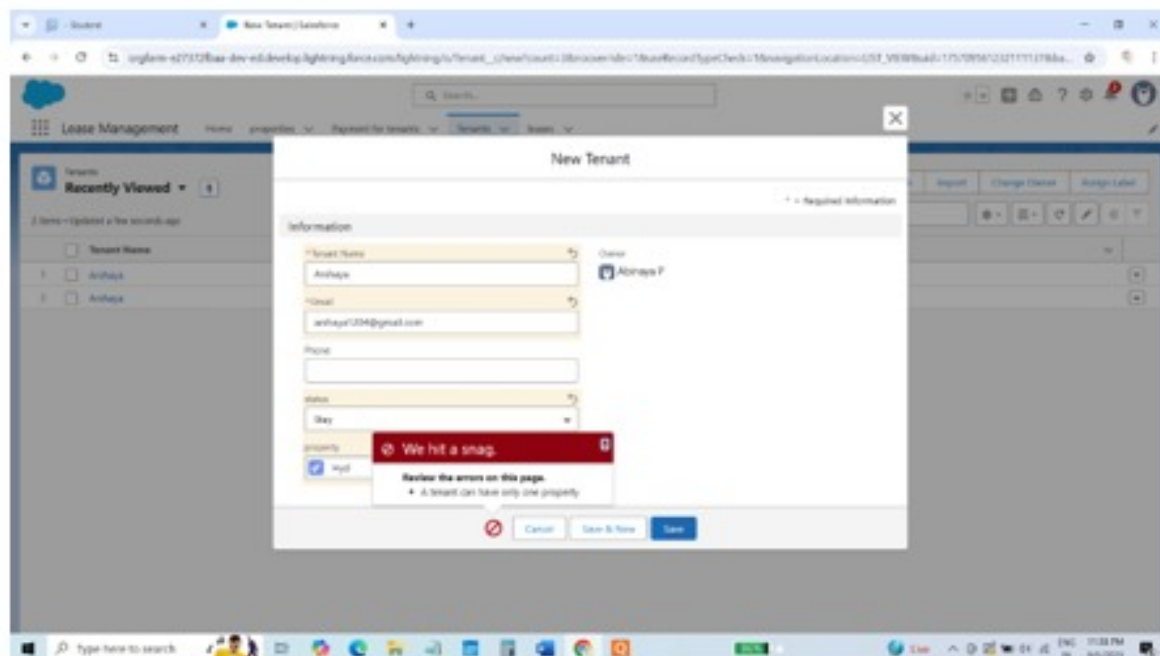
Action	Label	Tab Style	Description
<a href="#">Edit</a> / <a href="#">Del</a>	Home	Home	
<a href="#">Edit</a> / <a href="#">Del</a>	Standard for Objects	Standard	
<a href="#">Edit</a> / <a href="#">Del</a>	Analytics	Analytics	
<a href="#">Edit</a> / <a href="#">Del</a>	Mobile	Mobile	

The 'Web Tabs' and 'Visualforce Tabs' sections are currently empty, each with a 'New' button and a 'What is This?' link. The 'Lightning Component Tabs' section is also visible at the bottom.

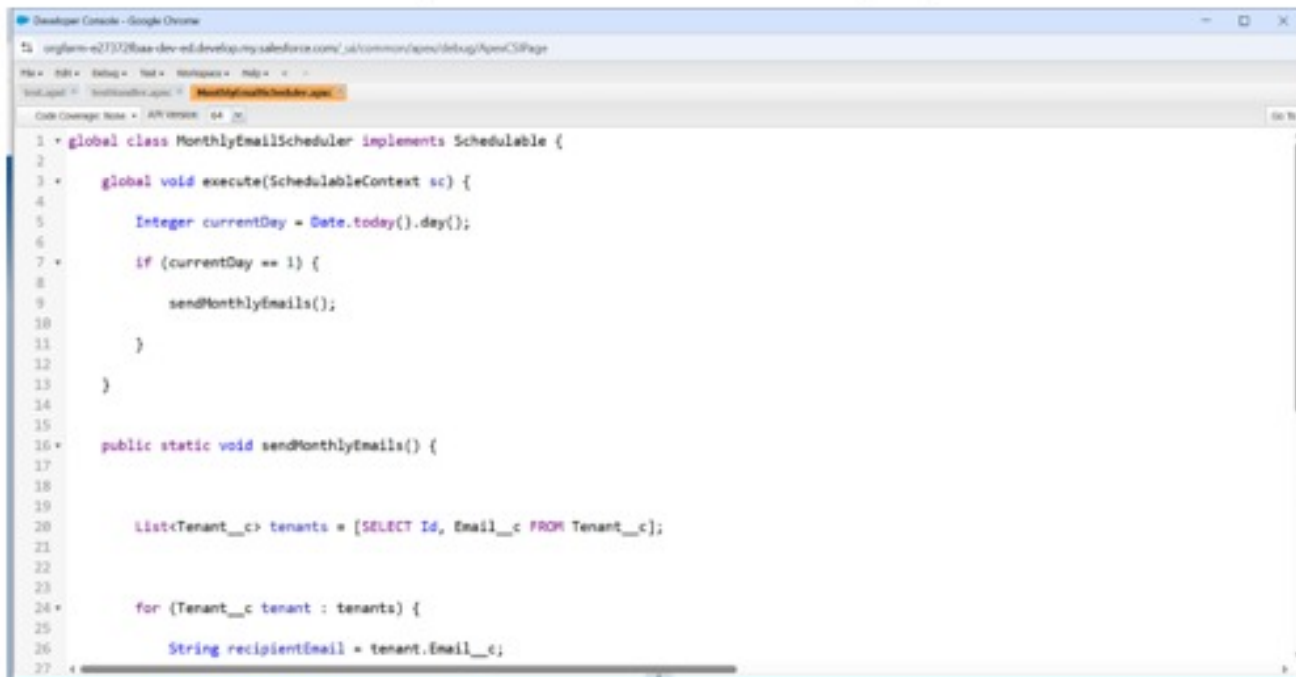




- Added Apex trigger to restrict multiple tenants per property

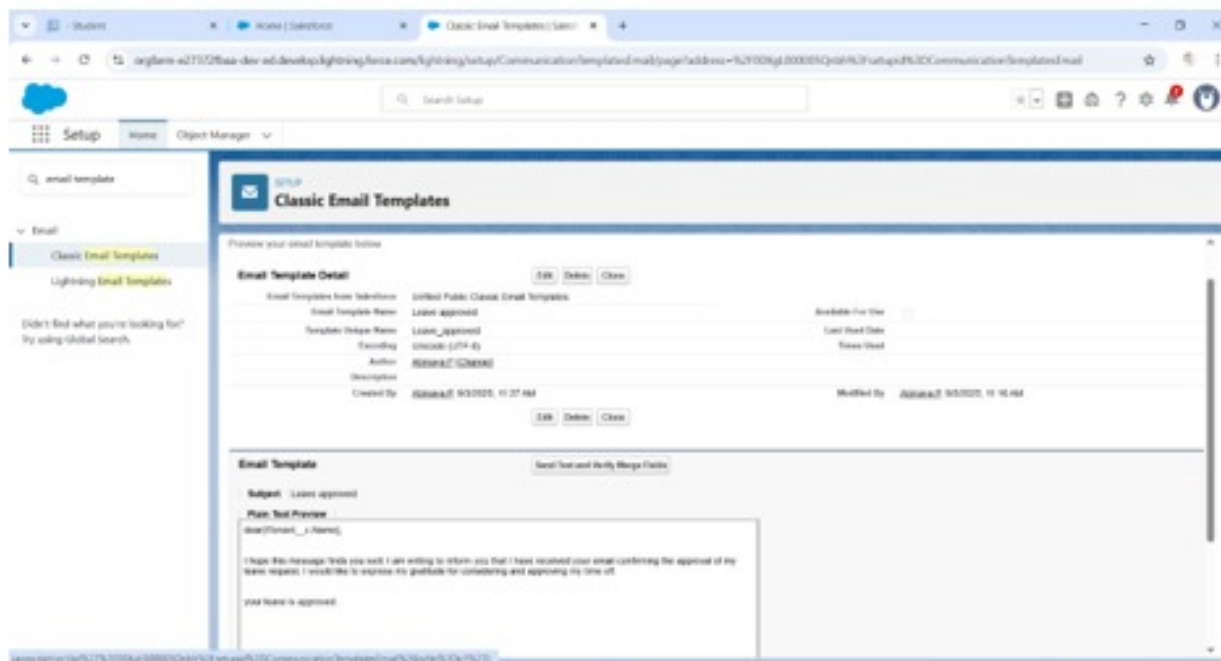


## • Scheduled monthly reminder emails using Apex class

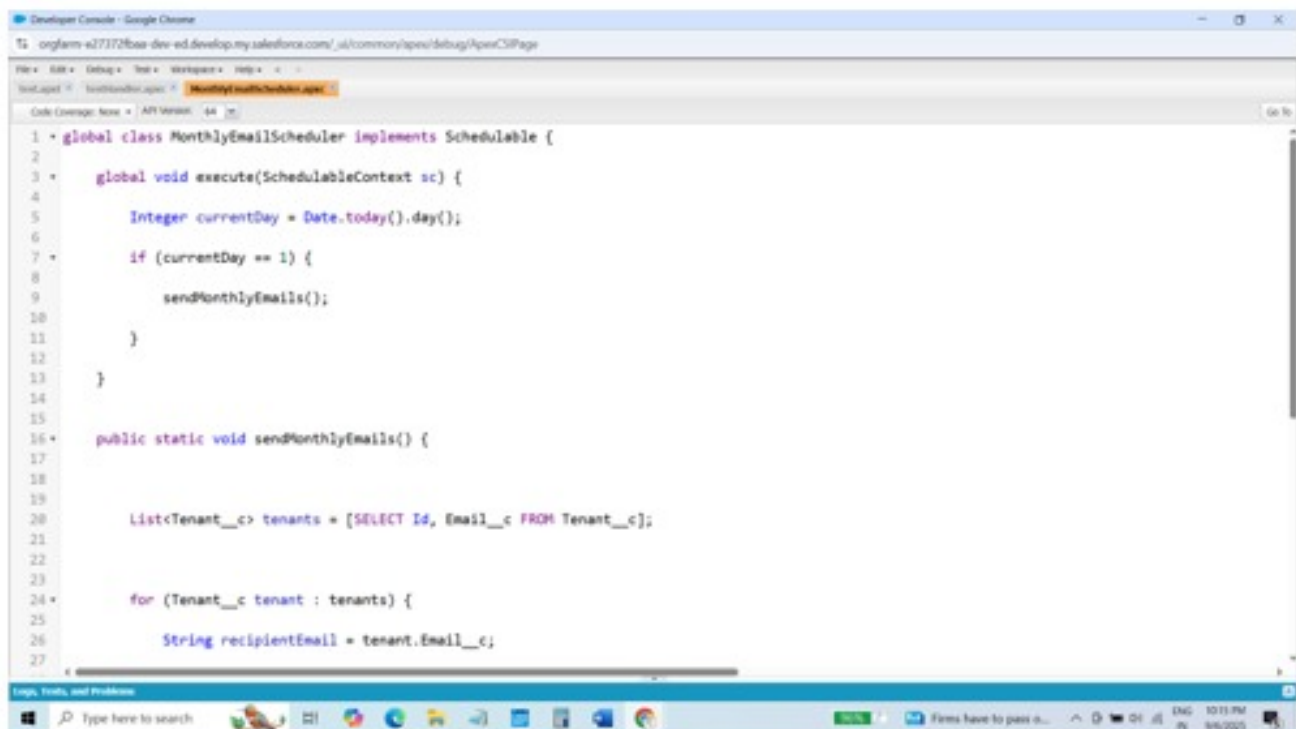


```
1 global class MonthlyEmailScheduler implements Schedulable {
2
3     global void execute(SchedulableContext sc) {
4
5         Integer currentDay = Date.today().day();
6
7         if (currentDay == 1) {
8             sendMonthlyEmails();
9         }
10    }
11
12
13 }
14
15
16 public static void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

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



## ● Schedule class: Create an Apex Class

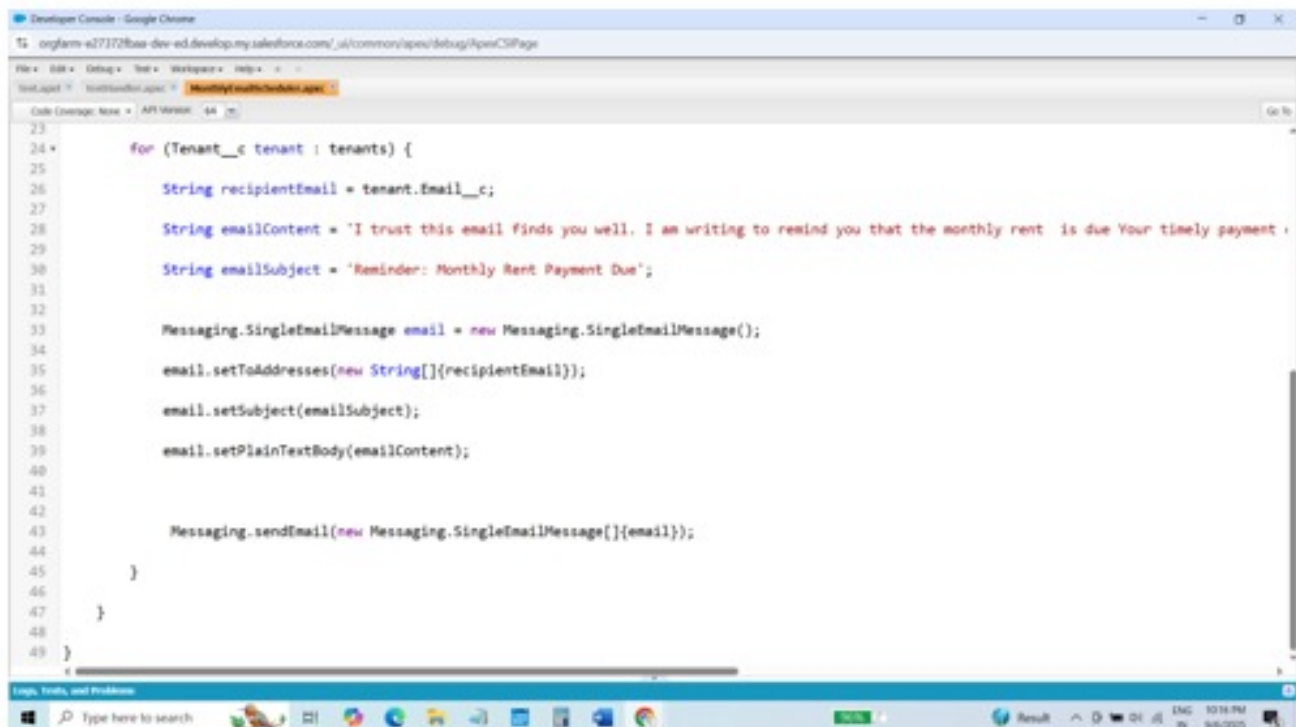


The screenshot shows the Salesforce Developer Console with the Apex class `MonthlyEmailScheduler` open. The class implements the `Schedulable` interface. The `execute` method checks if the current day is 1 and calls `sendMonthlyEmails`. The `sendMonthlyEmails` method queries the `Tenant__c` object and starts a loop to iterate over the results.

```

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             sendMonthlyEmails();
9         }
10
11     }
12
13 }
14
15
16 • public static void sendMonthlyEmails() {
17
18
19
20     List<Tenant__c> tenants = [SELECT Id, Email__c FROM Tenant__c];
21
22
23
24 •     for (Tenant__c tenant : tenants) {
25
26         String recipientEmail = tenant.Email__c;
27

```



The screenshot shows the continuation of the `MonthlyEmailScheduler` class. The loop in `sendMonthlyEmails` continues by creating a `Messaging.SingleEmailMessage` object, setting its properties (addresses, subject, body), and sending it via `Messaging.sendEmail`.

```

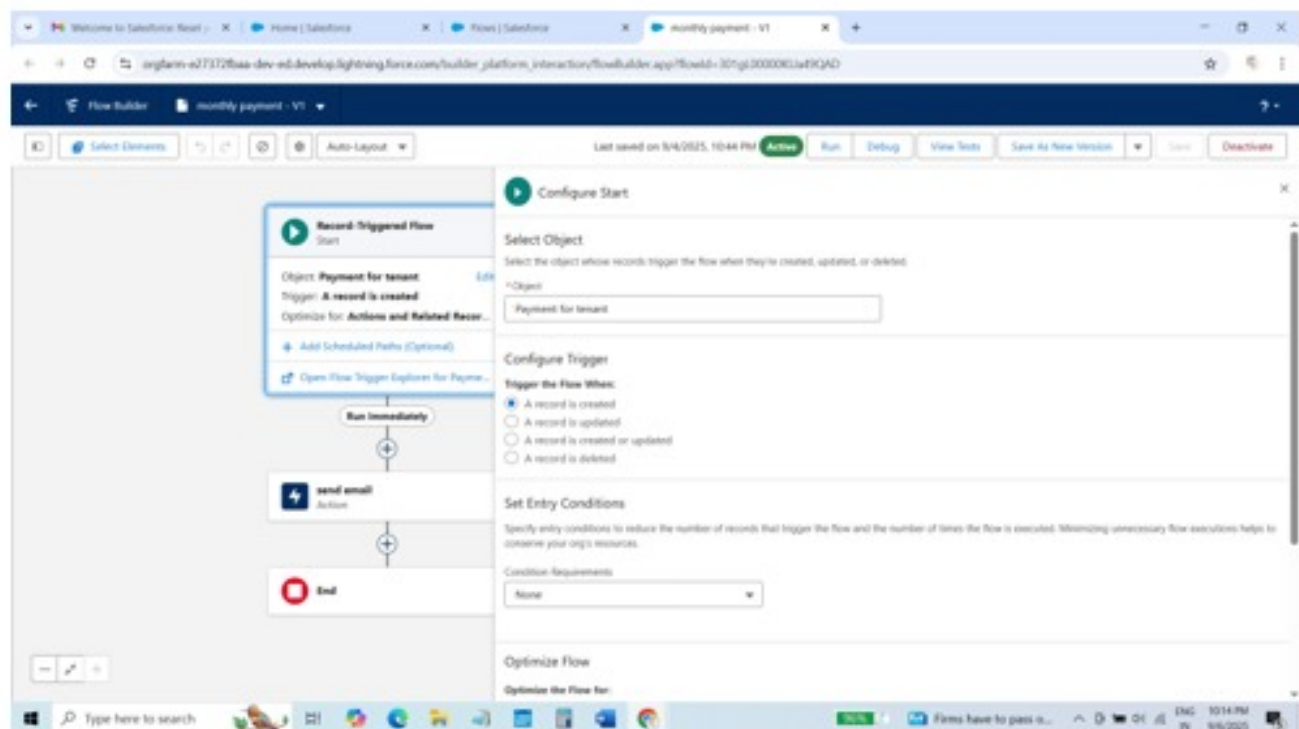
23
24 •     for (Tenant__c tenant : tenants) {
25
26         String recipientEmail = tenant.Email__c;
27
28         String emailContent = 'I trust this email finds you well. I am writing to remind you that the monthly rent is due Your timely payment';
29
30         String emailSubject = 'Reminder: Monthly Rent Payment Due';
31
32
33         Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
34         email.setToAddresses(new String[]{recipientEmail});
35         email.setSubject(emailSubject);
36         email.setPlainTextBody(emailContent);
37
38
39         Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
40
41
42
43     }
44
45 }
46
47
48
49 }

```

## Create an Apex Handler class

```
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         for (Tenant__c newTenant : newList) {
14
15
16
17
18             if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
19
20                 newTenant.addError('A tenant can have only one property');
21
22             }
23
24         }
25
26     }
27 }
```

## • FLOWS



## Submit for Approval

Comments

Leaving

Cancel

Submit