

# **TITLE: - LEASE MANAGEMENT USING SALESFORCE**

**College Name:** Navarasam Arts and Science College for Women

**College Code:** bru3a

## **Team Members:**

**Team Leader Name:** Shanmuga Sree C

**E-mail:** [shannmugasree093@gmail.com](mailto:shannmugasree093@gmail.com)

**Team Member1:** Monisha V

**E-mail:** [monishav25806@gmail.com](mailto:monishav25806@gmail.com)

**Team Member2:** Vaishnavi V

**E-mail:** [v43810051@gmail.com](mailto:v43810051@gmail.com)

**Team Member3:** Sandhiya Devi E

**E-mail:** [sarankumar112233sa@gmail.com](mailto:sarankumar112233sa@gmail.com)

## **INTRODUCTION**

### **Project Overview:**

The Lease Management project develops a Lease Management system on the Salesforce platform to handle agreements for properties and assets. Salesforce objects and workflows are used to track lease details, monitor payments, and manage compliance. The system provides alerts, reports, and dashboards to support decision-making. It ensures a structured approach to managing the complete lease lifecycle.

**Purpose:** This project provides a system to manage lease agreements effectively. It automates key tasks to minimize errors and support compliance.

### **Objectives:**

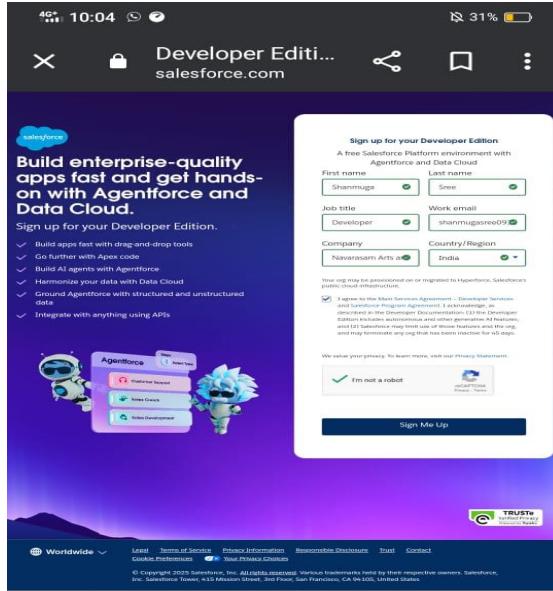
- Streamline Lease Agreements:** Automate the creation, renewal, and termination of lease contracts.
- Ensure Accuracy in Records:** Maintain centralized, accurate, and up-to-date information on all leases.
- Improve Communication:** Facilitate better collaboration between property, owners, tenants, and managers.
- Optimize Payment Tracking:** Automate rent/lease payment schedules, reminders, and receipts.
- Provide Real-Time Insights:** Generate reports and dashboards for better decision-making and performance monitoring.



## 1.DEVELOPMENT PHASE

## 1.1 Creating Developer Account in Salesforce:

By using this URL- <https://developer.salesforce.com/signup>

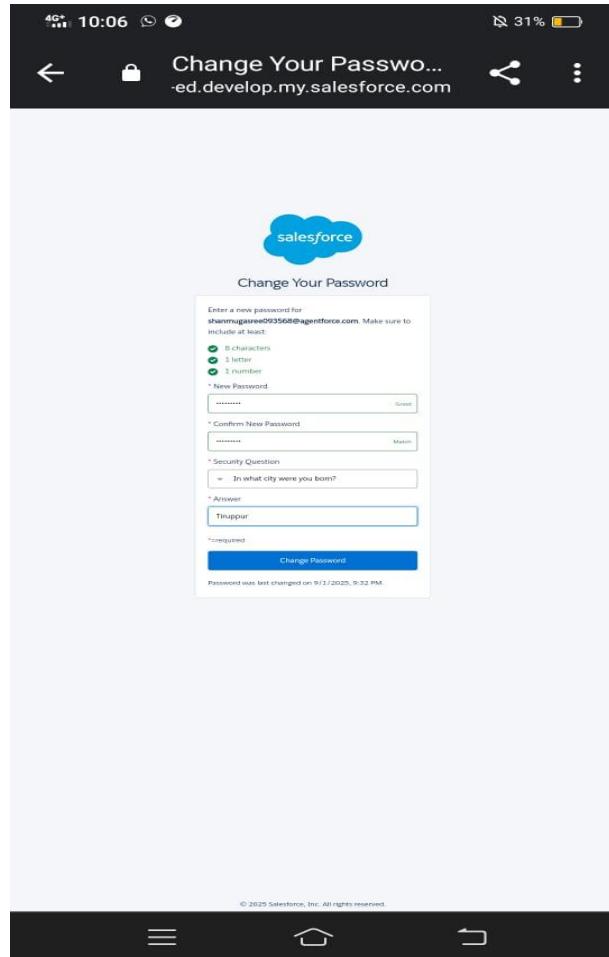


□ Enter essential details to sign up in developer edition as shown in image,

1. Name – First Name & Last Name.
2. Job title – Developer.
3. E-mail.
4. Company Name or College Name.

## 1.2 Account Activation

Verifying the e-mail account to reset the password for activating salesforce account.



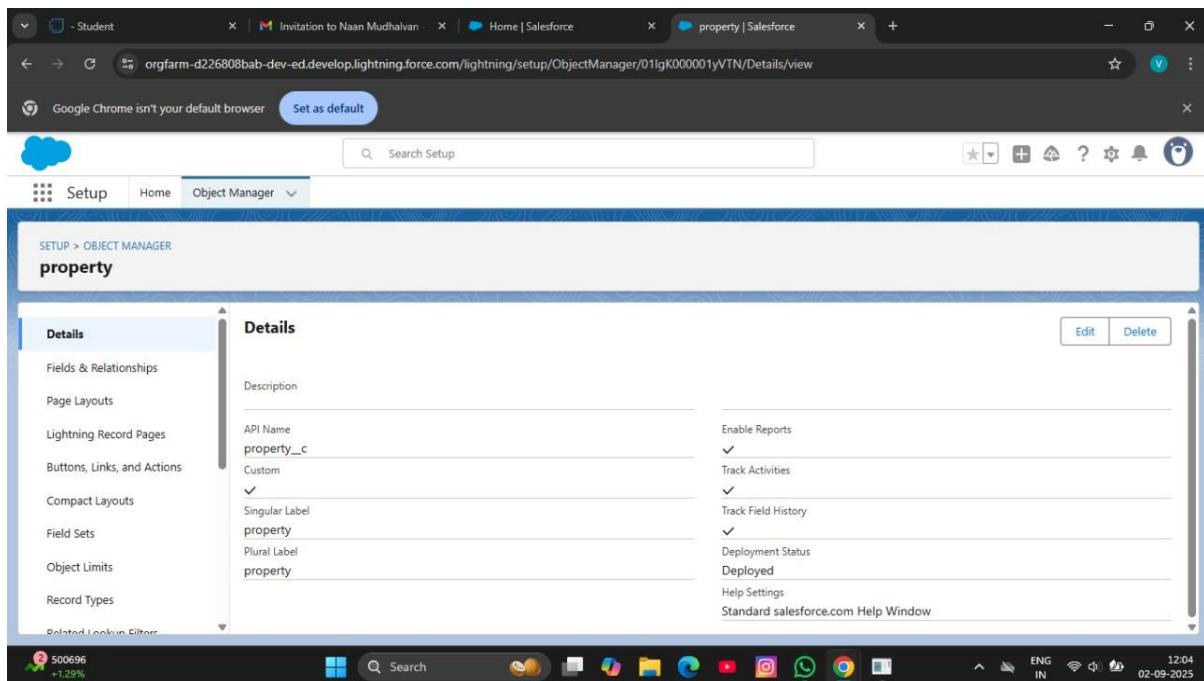
- Set the new password.
- Confirm the new password.
- Set answer for security question.
- Now you will redirect it to the salesforce setup page.

## 2. OBJECT CREATION PHASE

## To create an object:

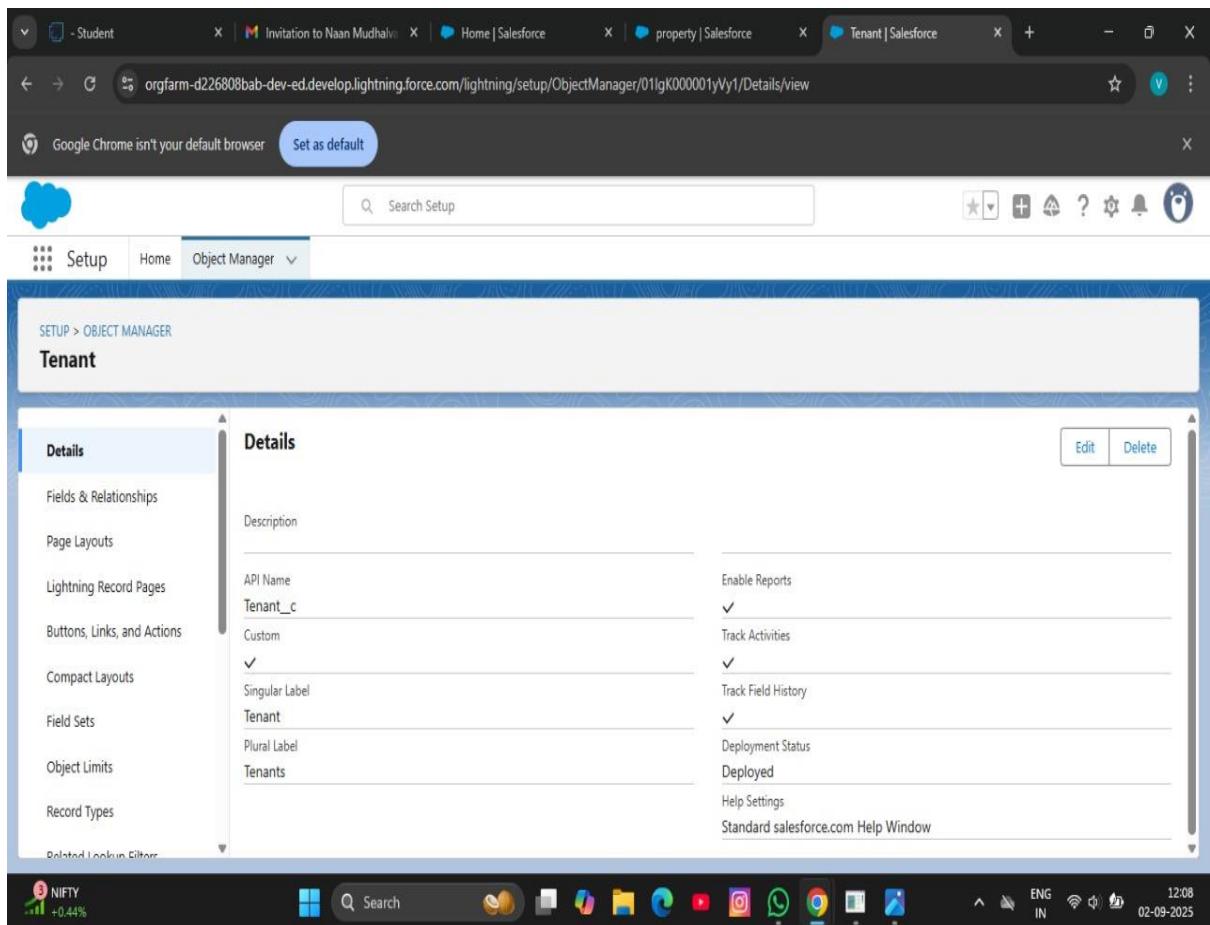
From the setup page > Click on Object Manager > Click on Create > Click on Custom Object.

### 2.1 Creating Property Object



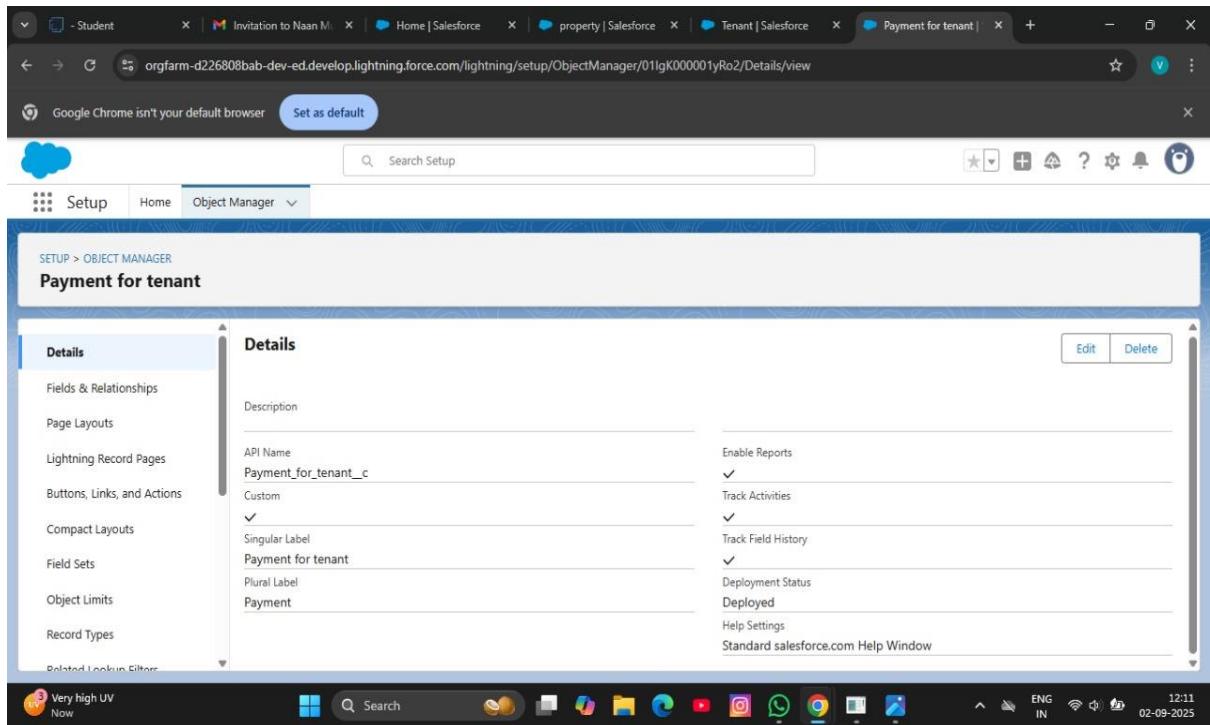
- **Created object property.**
- It stores details of properties available for lease, such as location and type. It helps identify assets in the system. Properties are linked to lease records for tracking.

### 2.2 Creating Tenant Object



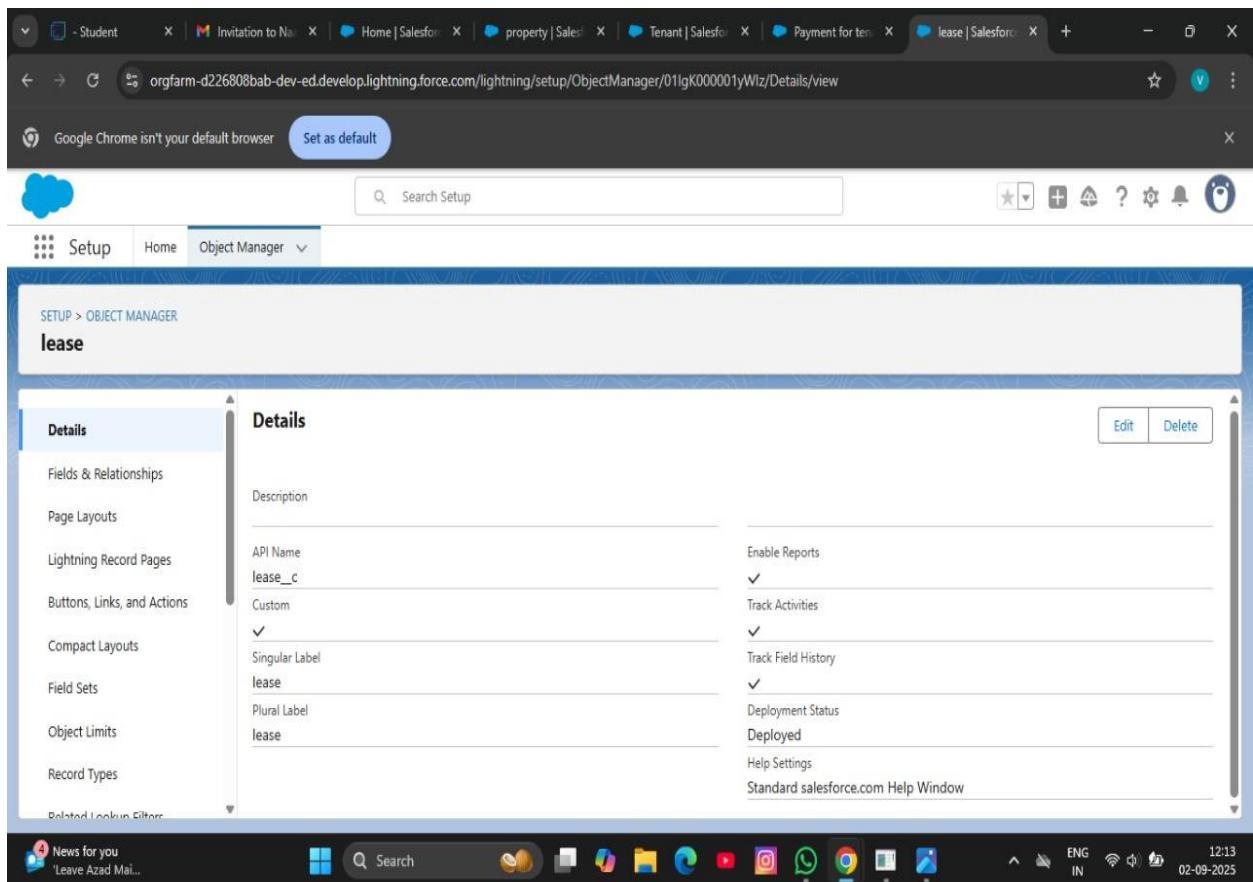
- **Created object Tenant.**
- It holds tenant information including name, contact, and lease history. It allows easy management of client details. Tenants are connected to properties and leases for clarity.

## 2.3 Creating Payment Object



- **Created object Payment for tenant.**
- It records all payment transactions related to leases. It keeps track of due amounts, status, and history. Payments are tied to leases for accurate financial management.

## 2.4 Creating Lease Object

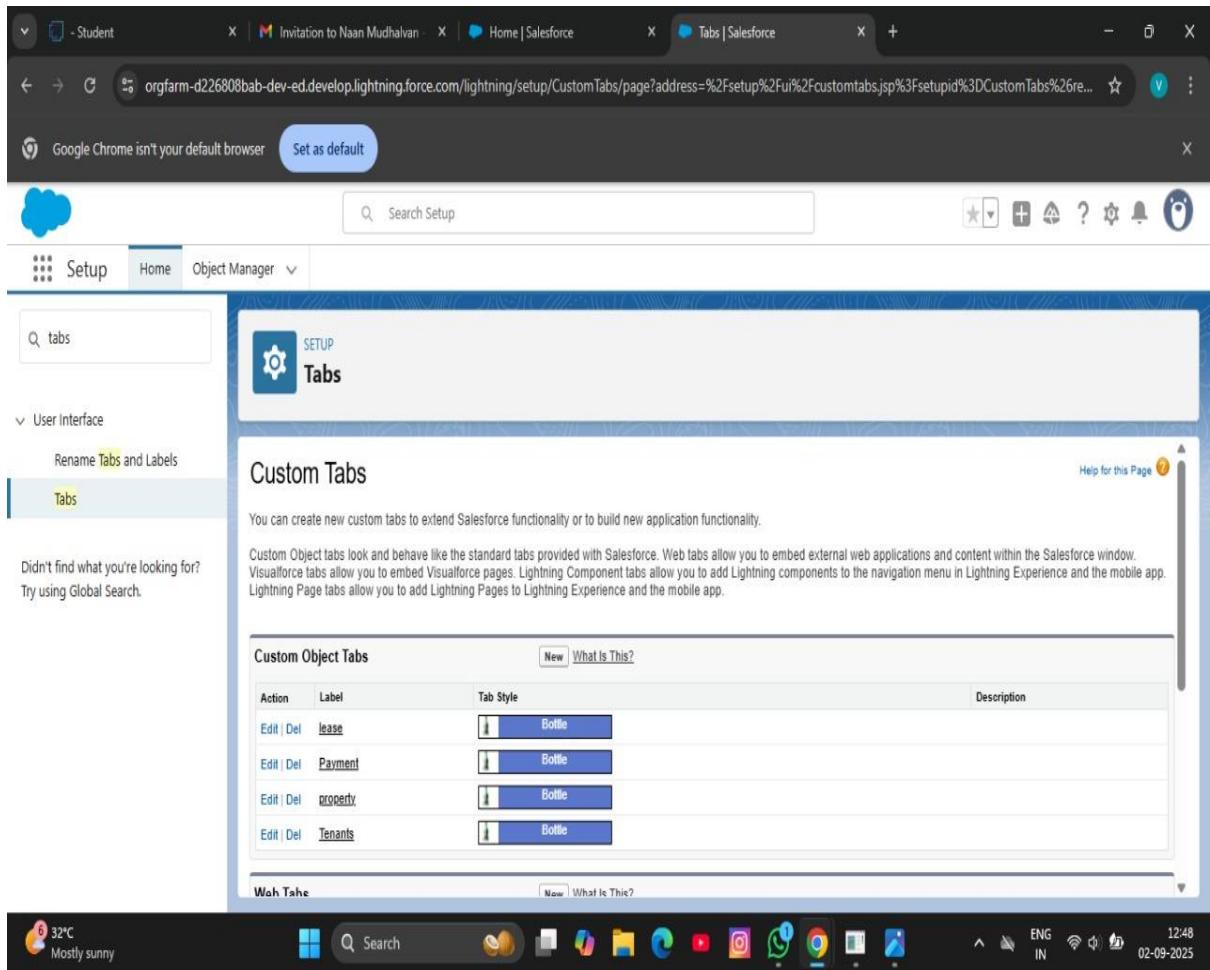


- **Created object lease.**
- It manages agreements between tenants and properties with start/end dates and terms. It acts as the central link between tenants, properties, and payments. This ensures proper lifecycle tracking.

### 3.TAB CUSTOMIZATION PHASE

**To create customized tab:**

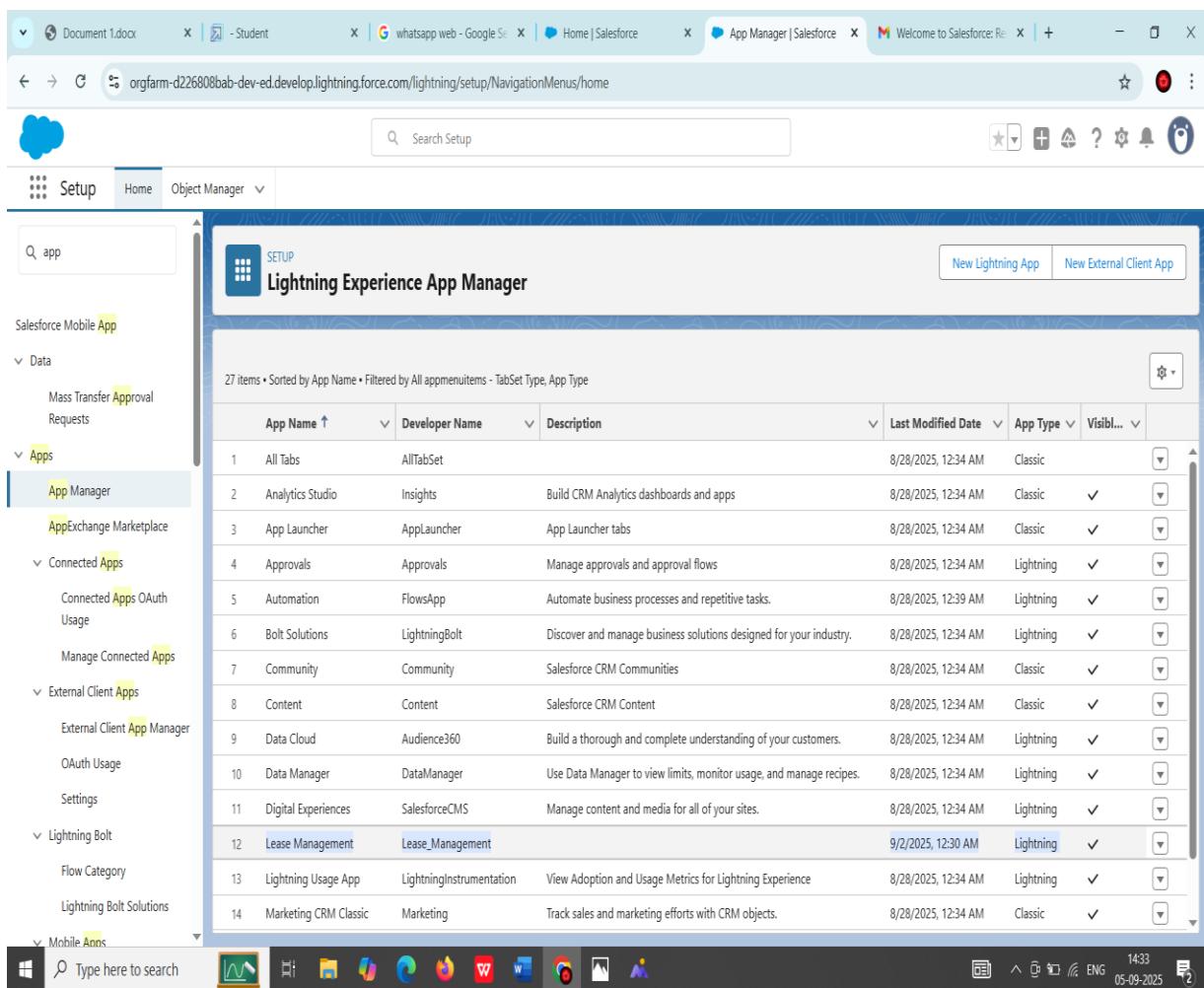
- From the setup page > **type Tabs in Quick Find bar** > click on tabs > New.
- Select object > **Tab Style**.



- Select the desired tab styles for the objects lease, Payment, property, Tenants.

## 4. APPLICATION DEVELOPMENT PHASE

- Creating Lightning app for Lease Management



- A Lightning App was created to bring all lease-related objects like Property, Tenant, Lease, and Payment under one workspace.
- It provides users with an organized view and easy navigation between records.
- This improves efficiency by managing all lease processes within a single app.

## 5.FIELDS AND RELATIONSHIPS PHASE

### 5.1 Creating Fields for Property Object

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

**Setup** > **OBJECT MANAGER** > **property**

**Fields & Relationships** (9 items, Sorted by Field Label)

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	property__c	Lookup(property)	✓	
property Name	Name	Text(80)	✓	
sfqt	sfqt__c	Text(18)		
Type	Type__c	Picklist		

- Created field labels of **name**, **address**, **type**, **sfqt** for the property object.
- Custom fields were added to capture details like property size, type, and availability. These fields help maintain accurate property records. They also support linking with lease agreements.

## 5.2 Creating Fields for Tenant Object

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

- Tab Bar:** Includes "Student", "Invitation to Naan Mudhalvan", "Home | Salesforce", and "Tenant | Salesforce".
- Header:** Shows "orgfarm-d226808bab-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgK00001yVY1/FieldsAndRelationships/view". A message says "Google Chrome isn't your default browser Set as default".
- Toolbar:** Includes a cloud icon, a search bar with "Search Setup", and various icons for navigation and settings.
- Breadcrumb:** "SETUP > OBJECT MANAGER".
- Section:** "Tenant".
- Left Sidebar:** "Fields & Relationships" is selected. Other options include "Page Layouts", "Lightning Record Pages", "Buttons, Links, and Actions", "Compact Layouts", "Field Sets", "Object Limits", and "Record Types".
- Table:** "Fields & Relationships" table with 7 items, sorted by Field Label.

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		
- Bottom Bar:** Weather (32°C, Mostly sunny), Search, and various system icons. Language is set to ENG IN. Date is 02-09-2025.

- Created field labels of **Email**, **Phone**, **status** for the Tenant object.
- Fields were created to store tenant information such as contact details, ID proof, and lease preferences. This ensures complete tenant profiles. The fields also help in tracking tenant history.

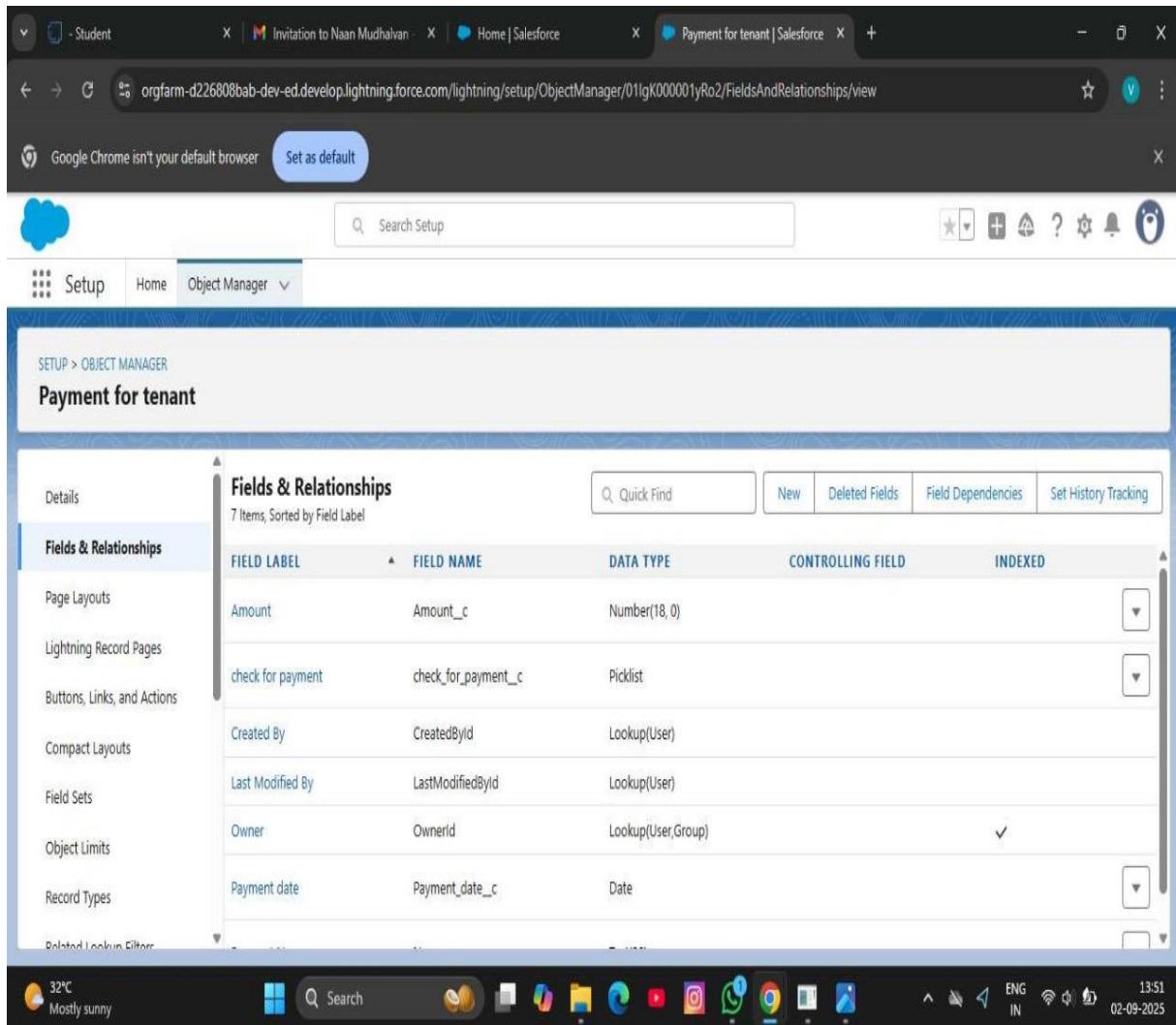
### 5.3 Creating Fields for Lease Object

The screenshot shows a browser window with multiple tabs open, including 'Student', 'Invitation to Naan Mudhalvan', 'Home | Salesforce', and 'Lease | Salesforce'. The main content is the 'Object Manager' for the 'Lease' object. The left sidebar lists various setup options like 'Fields & Relationships', 'Page Layouts', 'Lightning Record Pages', etc. The main area displays a table titled 'Fields & Relationships' with 6 items. The table columns are 'FIELD LABEL', 'FIELD NAME', 'DATA TYPE', 'CONTROLLING FIELD', and 'INDEXED'. The data is as follows:

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

- Created field labels of **start date**, **End date** for the lease object.
- Custom fields were designed to record lease terms, duration, and agreement status. These fields act as the backbone of lease tracking. They also connect with payment schedules and property details.

## 5.4 Creating Fields for Payment for Tenant Object



- Created field labels of **Payment date**, **Amount**, **check for payment** for the payment for tenant object.
- Custom fields were designed to record lease terms, duration, and agreement status. These fields act as the backbone of lease tracking. They also connect with payment schedules and property details.

## 5.5 Creation of Lookup Fields

- **Creation of Lookup Field on Lease Object**

1. Click on Object Manager >> (**Lease**) >> click on object.

2. Click on “**Fields & Relationships**” >> New
3. Select lookup relationship
4. Select the related object “**property**” and click next.
5. Field Name: property
6. Field label: Auto generated
7. Next >> Next >> Save.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for Setup, Home, and Object Manager. The main content area is titled "SETUP > OBJECT MANAGER" and shows the "lease" object. On the left, a sidebar menu is open under the "Fields & Relationships" section, listing options like Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, and Record Types. The main table displays seven items, sorted by Field Label. Each row contains three columns: the field name, its data type, and its description. For example, the "lease Name" field is of type "Text(80)". The table also includes a "Quick Find" search bar and buttons for "New", "Deleted Fields", "Field Dependencies", and "Set History Tracking".

- The lookup field connects Lease with Property or Tenant, ensuring each lease record is linked to the right details.
- **Creation of Lookup Field on Payment Object**

1. Click on Object Manager >> (**Payment**) >> click on object.
2. Now click on “**Fields & Relationships**” >> New
3. Select lookup relationship
4. Select the related object “**Tenant**” and click next.

5. Field Name: Tenant
6. Field label: Auto generated
7. Next >> Next >> Save.

The screenshot shows the Salesforce Object Manager Fields & Relationships page for the Payment object. The left sidebar has 'Fields & Relationships' selected. The main area displays a table of fields:

Salesforce Result Code	\$ResultCode	Picklist
Status	Status	Picklist
Tenant	Tenant_c	Lookup(Tenant)
Total Applied	TotalApplied	Currency(16, 2)
Total Refund Applied	TotalRefundApplied	Roll-Up Summary (Refund Line Payment)
Total Refund Unapplied	TotalRefundUnapplied	Roll-Up Summary (Refund Line Payment)
Total Unapplied	TotalUnapplied	Currency(16, 2)
Type	Type	Picklist

At the bottom of the page, there are standard Salesforce navigation buttons: New, Deleted Fields, Field Dependencies, and Set History Tracking. The status bar at the bottom right shows system information: ENG IN, 14:58, 02-09-2025.

- The lookup field ties Payments to a Lease, so all monthly or one-time payments can be tracked under the correct agreement.
  - **Creation of Lookup Field on property Object**

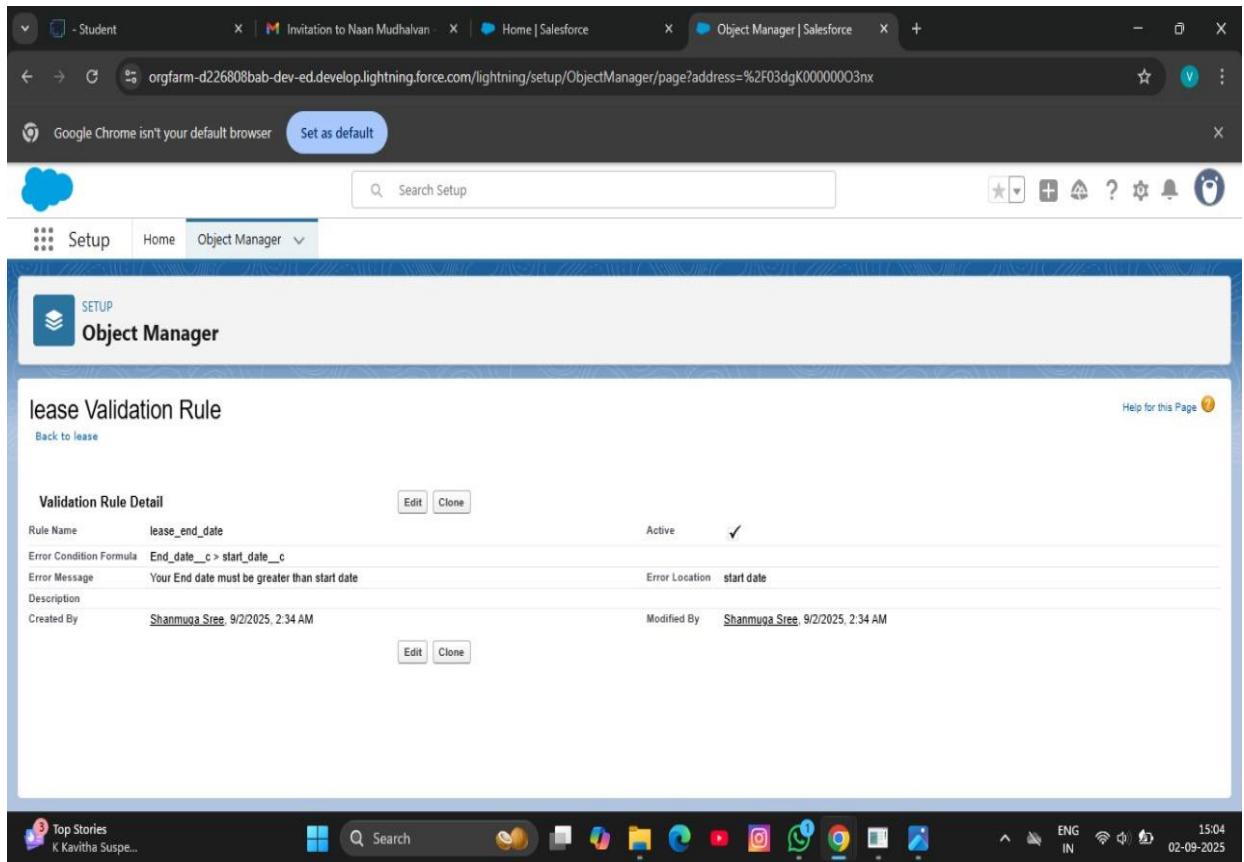
1. Click on Object Manager >>(property)>>click on object.
2. Now click on “Fields & Relationships” >> New
3. Select lookup relationship
4. Select the related object “property” and click next.
5. Field Name: property
6. Field label: Auto generated

## 7. Next >> Next >> Save.

The screenshot shows the Salesforce Object Manager interface. The top navigation bar includes tabs for 'Student', 'Invitation to Naan Mudhalvan', 'Home | Salesforce', and 'property | Salesforce'. A message 'Google Chrome isn't your default browser' with a 'Set as default' button is displayed. The main area shows the 'property' object details. On the left, a sidebar lists various setup options like 'Fields & Relationships', 'Page Layouts', 'Lightning Record Pages', etc. The 'Fields & Relationships' section is currently selected, displaying a table of fields. The table has columns for 'Field Label', 'Type', and 'Description'. Fields listed include 'Last Modified By', 'Name', 'Owner', 'property', 'property Name', 'sfqt', and 'Type'. The 'Owner' field is highlighted with a checkmark. The bottom of the screen shows the Windows taskbar with various pinned icons and system status indicators.

- The lookup field associates Property with Lease, helping to identify which property is currently leased.

## 6.VALIDATION RULE PHASE



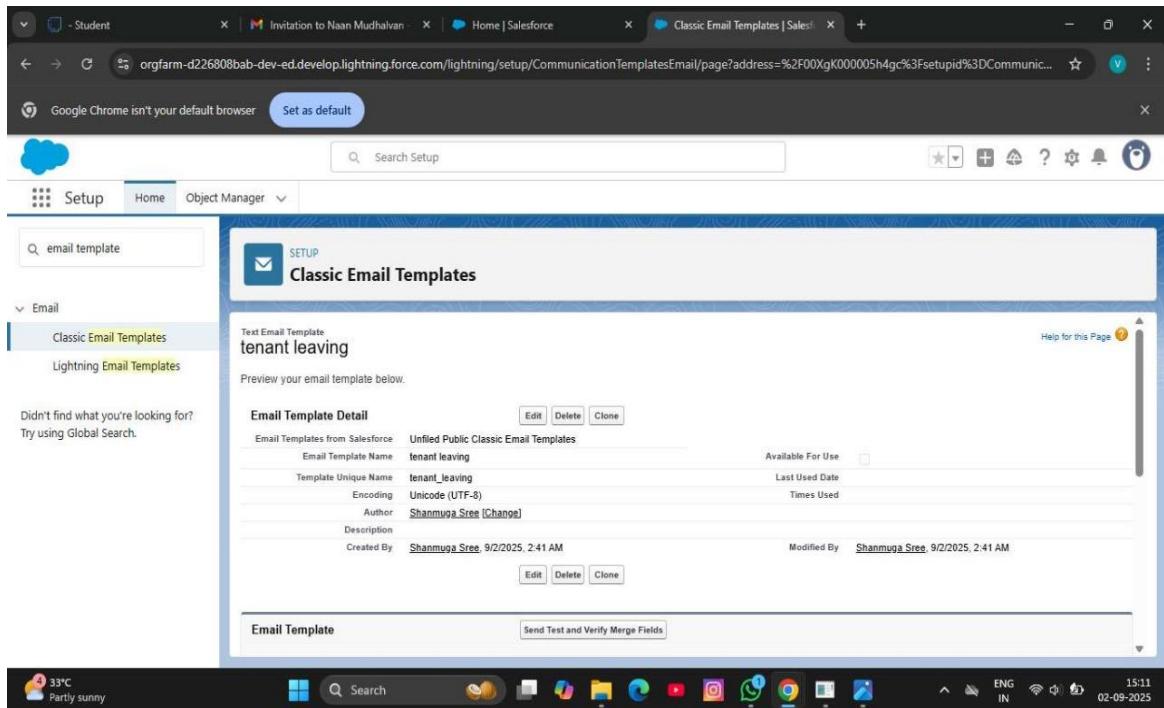
## □ Created Validation Rule to a Lease Object.

Validation rules ensure data accuracy by restricting invalid entries. In this project, they are applied to lease records, such as checking correct start and end dates, or preventing duplicate property assignments. This helps maintain reliable and error-free lease information.

## 7. EMAIL TEMPLATES PHASE

- Creating Email Template for Tenant Leaving
- Creating Email Template for Leave Approved
- Creating Email Template for rejection for leave

- Creating Email Template for Monthly payment
- Creating Email Template for successful payment



- Email templates were created to send automated notifications for **leave approvals and rejections**.
- This ensures clear communication and quick updates to users without manual effort.

The screenshot shows the 'Classic Email Templates' page in Salesforce. The template name is 'Leave approved'. The details are as follows:

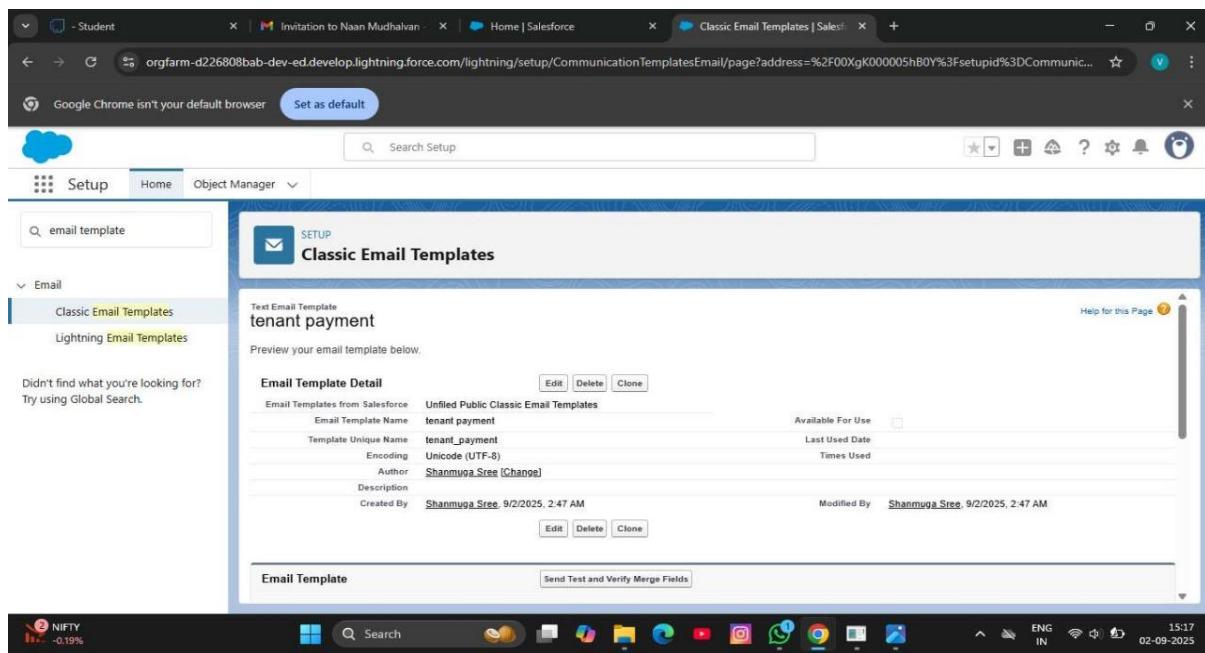
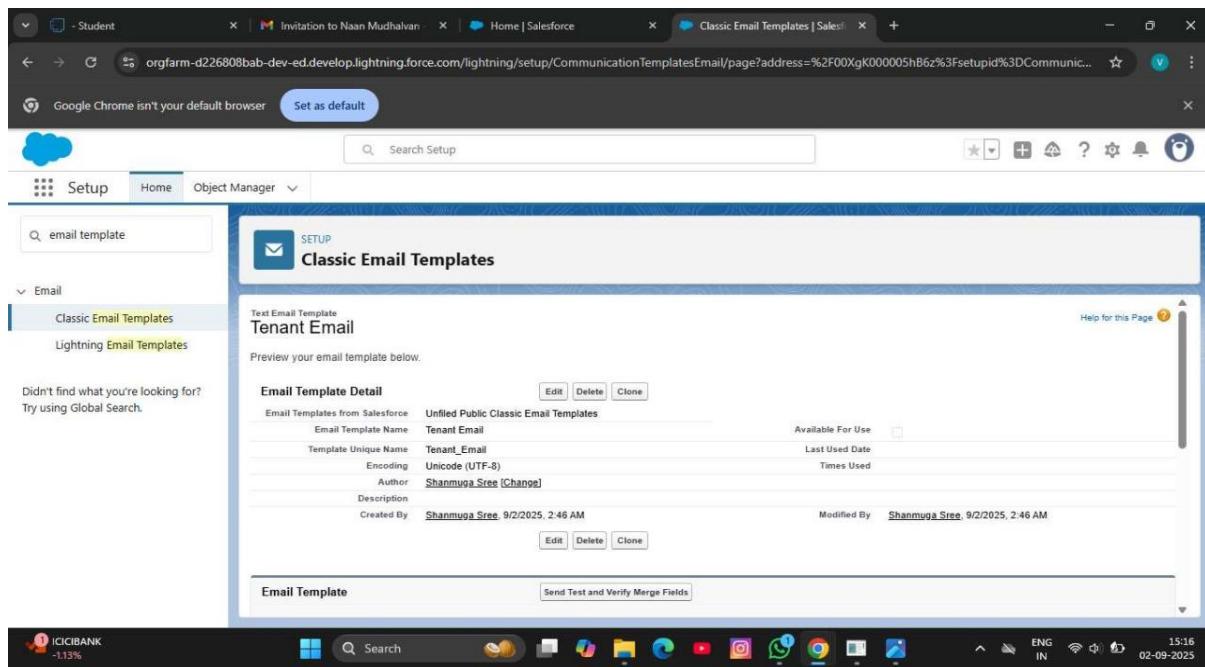
Email Template Detail	Unfiled Public Classic Email Templates
Email Template Name	Leave approved
Template Unique Name	Leave_approved
Encoding	Unicode (UTF-8)
Author	Shanmuga Sree [Change]
Description	
Created By	Shanmuga Sree, 9/2/2025, 2:43 AM
Modified By	Shanmuga Sree, 9/2/2025, 2:43 AM

At the bottom, there are 'Edit', 'Delete', and 'Clone' buttons.

The screenshot shows the 'Classic Email Templates' page in Salesforce. The template name is 'Leave rejected'. The details are as follows:

Email Template Detail	Unfiled Public Classic Email Templates
Email Template Name	Leave rejected
Template Unique Name	Leave_rejected
Encoding	Unicode (UTF-8)
Author	Shanmuga Sree [Change]
Description	
Created By	Shanmuga Sree, 9/2/2025, 2:44 AM
Modified By	Shanmuga Sree, 9/2/2025, 2:44 AM

At the bottom, there are 'Edit', 'Delete', and 'Clone' buttons.



## 8. APPROVAL PROCESSES PHASE

Approval processes streamline authorization for important actions, like approving lease agreements or payment adjustments. They define a sequence of

steps where managers or admins must review and approve requests before they are finalized, ensuring compliance and control.

## 8.1 Create Approval Process For check for vacant

## 8.2 Initial Submission Action

## 8.3 Final Approval Action

## 8.4 Final Rejection Action

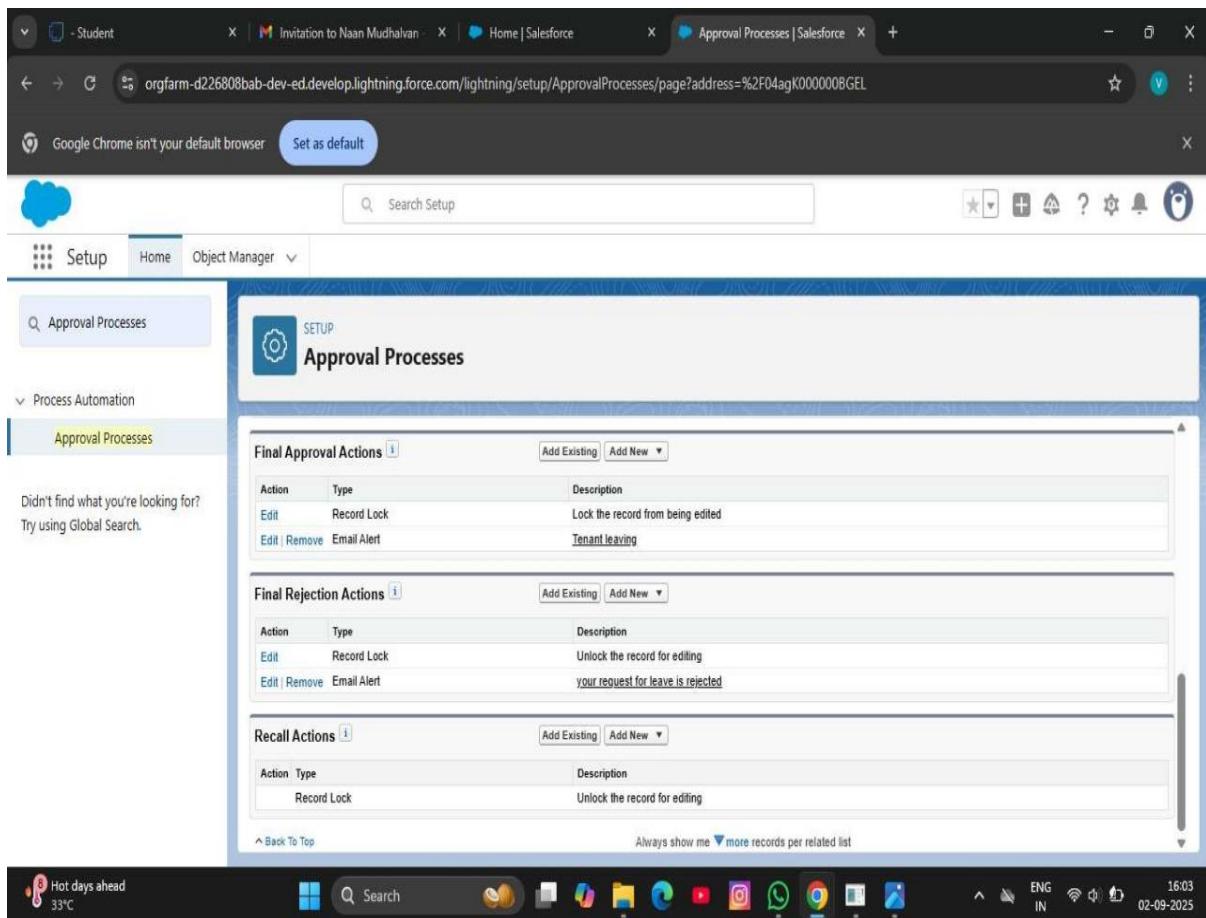
The screenshot shows the Salesforce Setup interface for creating an Approval Process. The process is titled "check for vacant". The "Process Definition Detail" section includes:

- Process Name: check for vacant
- Unique Name: check\_for\_vacant
- Description: Tenant: status NOT EQUAL TO Leaving
- Entry Criteria: Tenant: status NOT EQUAL TO Leaving
- Record Editability: Administrator ONLY
- Active: checked
- Next Automated Approver Determined By: [empty]
- Allow Submitters to Recall Approval Requests: unchecked

The "Initial Submission Actions" section contains:

Action	Type	Description
Record Lock	Lock	Lock the record from being edited
Email Alert	Email	please approve my leave

The "Approval Steps" section is empty, showing the message: "You have not yet defined any approval steps".



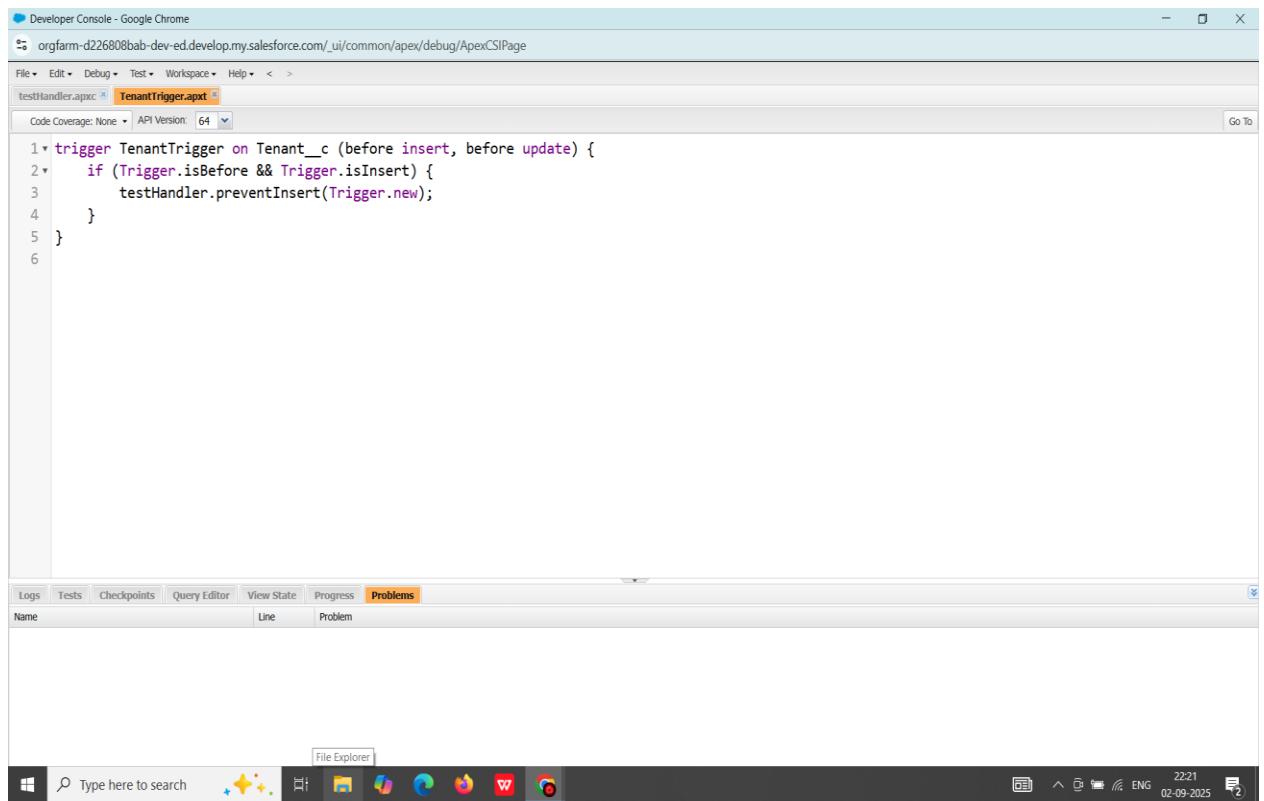
- Initial submission actions are the actions that occur when a user first submits a record for approval.
- Final Approval actions are the actions that occur when a record is approved from all the approval steps.
- Final Rejection actions are the actions that occur when a record is rejected from any of the approval steps.

## 9. APEX TRIGGER PHASE

- To create an Apex Trigger:

From the developer console >> Click on the file > New > Apex Trigger.

## 9.1 Creating an Apex Trigger



The screenshot shows the Salesforce Developer Console interface. The title bar reads "Developer Console - Google Chrome" and the URL is "orgfarm-d226808bab-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage". The menu bar includes File, Edit, Debug, Test, Workspace, Help, and several navigation icons. A tab labeled "testHandler.apxc" is open, showing the code for a trigger:

```
trigger TenantTrigger on Tenant__c (before insert, before update) {
    if (Trigger.isBefore && Trigger.isInsert) {
        testHandler.preventInsert(Trigger.new);
    }
}
```

The code editor shows "Code Coverage: None" and "API Version: 64". Below the editor, there are tabs for Logs, Tests, Checkpoints, Query Editor, View State, Progress, and Problems. The Problems tab is selected, showing a single entry: "Line 1 Problem". The bottom of the screen shows the Windows taskbar with the File Explorer icon highlighted, along with other pinned application icons like Edge, File Explorer, and others.

## 9.2 Creating an Apex Handler class

Developer Console - Google Chrome  
orgfarm-d22680bab-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >  
**testHandler.apxc** TenantTrigger.apxt  
Code Coverage: None API Version: 64 Go To

```
1 * public class testHandler {
2     public static void preventInsert(List<SObject> newList) {
3
4         // Cast SObject list to Tenant__c
5         List<Tenant__c> tenants = (List<Tenant__c>)newList;
6
7         Set<Id> propertyIds = new Set<Id>();
8
9         // Collect property Ids from incoming tenants
10        for (Tenant__c t : tenants) {
11            if (t.Property__c != null) {
12                propertyIds.add(t.Property__c);
13            }
14        }
15
16        // Query existing tenants with those properties
17        Map<Id, Tenant__c> propertyTenantMap = new Map<Id, Tenant__c>();
18        for (Tenant__c existing : [
19            SELECT Id, Property__c
20            FROM Tenant__c
21        ]) {
22            propertyTenantMap.put(existing.Id, existing);
23        }
24
25        // Now validate
26        for (Tenant__c t : tenants) {
27            if (t.Property__c != null && propertyTenantMap.containsKey(t.Property__c)) {
28                taddError('A property can have only one tenant.');
29            }
30        }
31    }
32 }
33 }
```

Logs Tests Checkpoints Query Editor View State Progress **Problems**

Name Line Problem

Type here to search Microsoft Edge 2221 ENG 02-09-2025

Developer Console - Google Chrome  
orgfarm-d22680bab-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage

File Edit Debug Test Workspace Help < >  
**testHandler.apxc** TenantTrigger.apxt  
Code Coverage: None API Version: 64 Go To

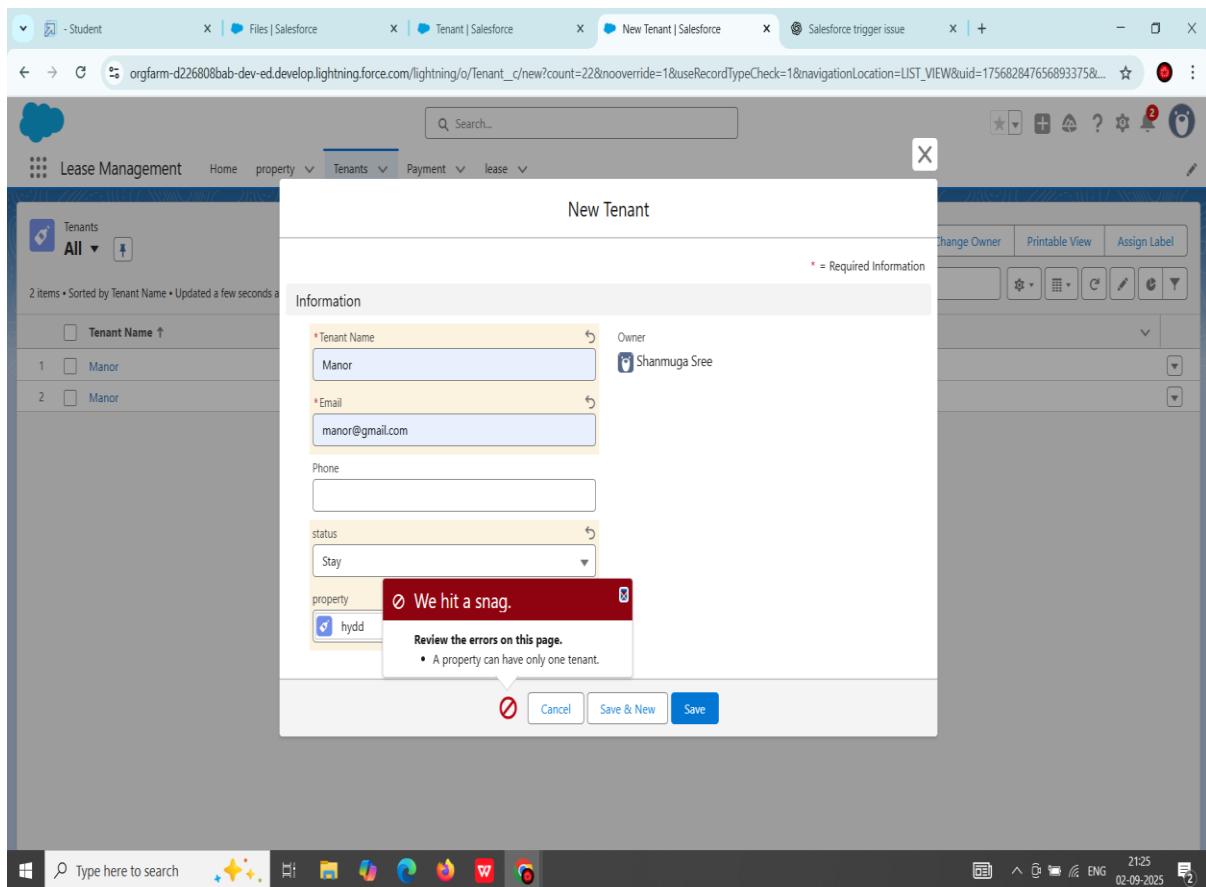
```
15
16        // Query existing tenants with those properties
17        Map<Id, Tenant__c> propertyTenantMap = new Map<Id, Tenant__c>();
18        for (Tenant__c existing : [
19            SELECT Id, Property__c
20            FROM Tenant__c
21            WHERE Property__c IN :propertyIds
22        ]) {
23            propertyTenantMap.put(existing.Id, existing);
24        }
25
26        // Now validate
27        for (Tenant__c t : tenants) {
28            if (t.Property__c != null && propertyTenantMap.containsKey(t.Property__c)) {
29                taddError('A property can have only one tenant.');
30            }
31        }
32    }
33 }
```

Logs Tests Checkpoints Query Editor View State Progress **Problems**

Name Line Problem

Type here to search Microsoft Edge 2225 ENG 02-09-2025

## 9.3 Testing the Trigger



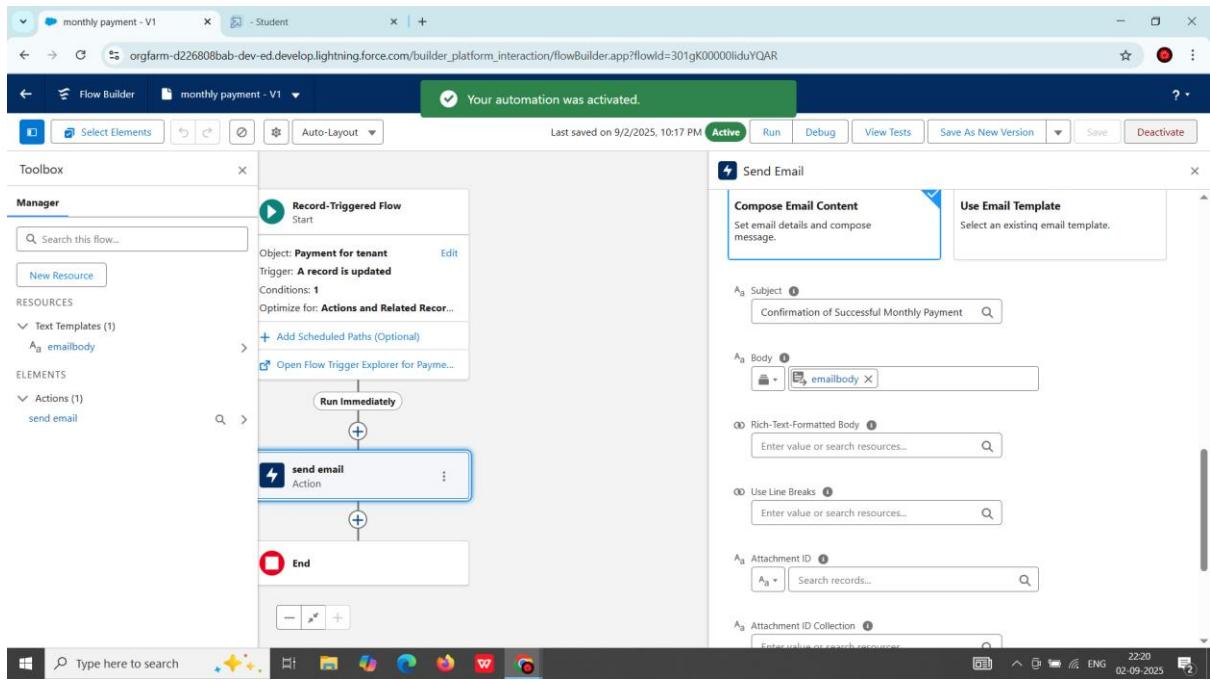
- The trigger ensures that a tenant can be associated with only one property, preventing multiple property allocations for the same tenant.

## 10. FLOW CREATION PHASE

### Creating Flow for Monthly Payment

**To Create Flow:**

- From the setup page > **type Flow in Quick Find bar** > click on flow > New flow.



- The flow records monthly payments automatically. It also reminds tenants about due payments on time.
- Flows automate repetitive tasks such as monthly payment reminders, updating records, or sending email notifications.

## 11. SCHEDULE CLASS PHASE

- Creating Apex Class**
- Creating Schedule Apex Class**

Developer Console - Google Chrome

File Edit Debug Test Workspace Help < >

testHandler.apxc TenantTrigger.apxc MonthlyEmailScheduler.apxc

Code Coverage: None API Version: 64 Go To

```
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  
16 public static void sendMonthlyEmails() {  
17  
18  
19  
20 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Type here to search 22:38 02-09-2025

Developer Console - Google Chrome

File Edit Debug Test Workspace Help < >

testHandler.apxc TenantTrigger.apxc MonthlyEmailScheduler.apxc

Code Coverage: None API Version: 64 Go To

```
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  
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 is appreciated.';  
29  
30     String emailSubject = 'Reminder: Monthly Rent Payment Due';  
31  
32  
33     Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();  
34  
35     email.setToAddresses(new String[]{recipientEmail});  
36  
37     email.setSubject(emailSubject);  
38  
39 }
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Type here to search 22:38 02-09-2025

Developer Console - Google Chrome  
 orgfarm-d22680bab-dev-ed.develop.my.salesforce.com/\_ui/common/apex/debug/ApexCSIPage

```

File Edit Debug Test Workspace Help < >
testHandler.apxc TenantTrigger.apxt MonthlyEmailScheduler.apxc
Code Coverage: None API Version: 64 Go To
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 is appreciated.';
29
30 String emailSubject = 'Reminder: Monthly Rent Payment Due';
31
32
33 Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
34
35 email.setToAddresses(new String[]{recipientEmail});
36
37 email.setSubject(emailSubject);
38
39 email.setPlainTextBody(emailContent);
40
41
42 Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
43
44
45 }
46
47
  
```

Logs Tests Checkpoints Query Editor View State Progress Problems

Name Line Problem

Type here to search 22:38 02-09-2025

Document 1.docx whatsapp web Home Salesforce Apex Classes Login Salesforce Welcome to Sales Project overview - Student

orgfarm-d22680bab-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/home

Setup Home Object Manager

apex

**Apex Classes**

Apex Code is an object oriented programming language that allows developers to develop on-demand business applications on the Lightning Platform.

**Percent of Apex Used: 0.04%**  
 You are currently using 2,178 characters of Apex Code (excluding comments and @isTest annotated classes) in your organization, out of an allowed limit of 6,000,000 characters. Note that the amount in use includes both Apex Classes and Triggers defined in your organization.

Estimate your organization's code coverage | Compile all classes | View: All | Create New View

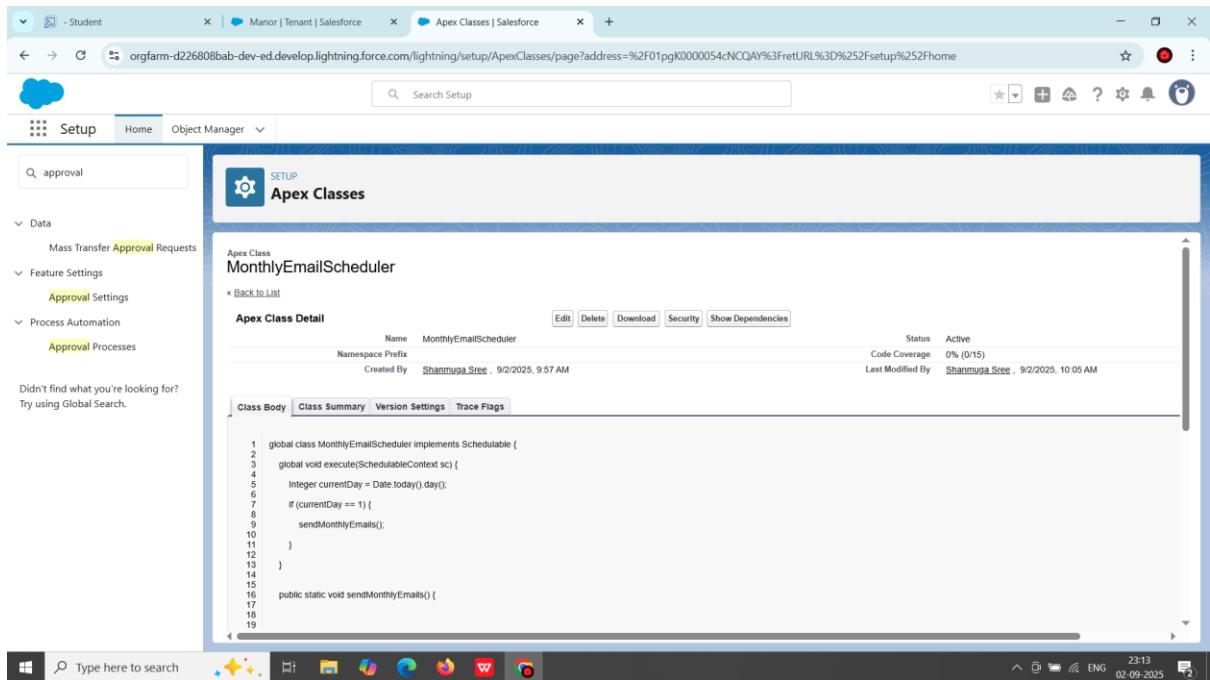
Action	Name	Namespace Prefix	Api Version	Status	Size Without Comments	Last Modified By	Has Trace Flags
Edit   Del   Security	MonthlyEmailScheduler		64.0	Active	1,125	Shanmuga Sree	9/2/2025, 10:05 AM
Edit   Del   Security	testHandler		64.0	Active	881	Shanmuga Sree	9/2/2025, 8:49 AM

**Dynamic Apex Classes**

Dynamic Apex extends your programming reach by interacting with Lightning Platform components.

Class Name	Namespace Prefix	Api Version	Created By	Last Modified By
No records to display.				

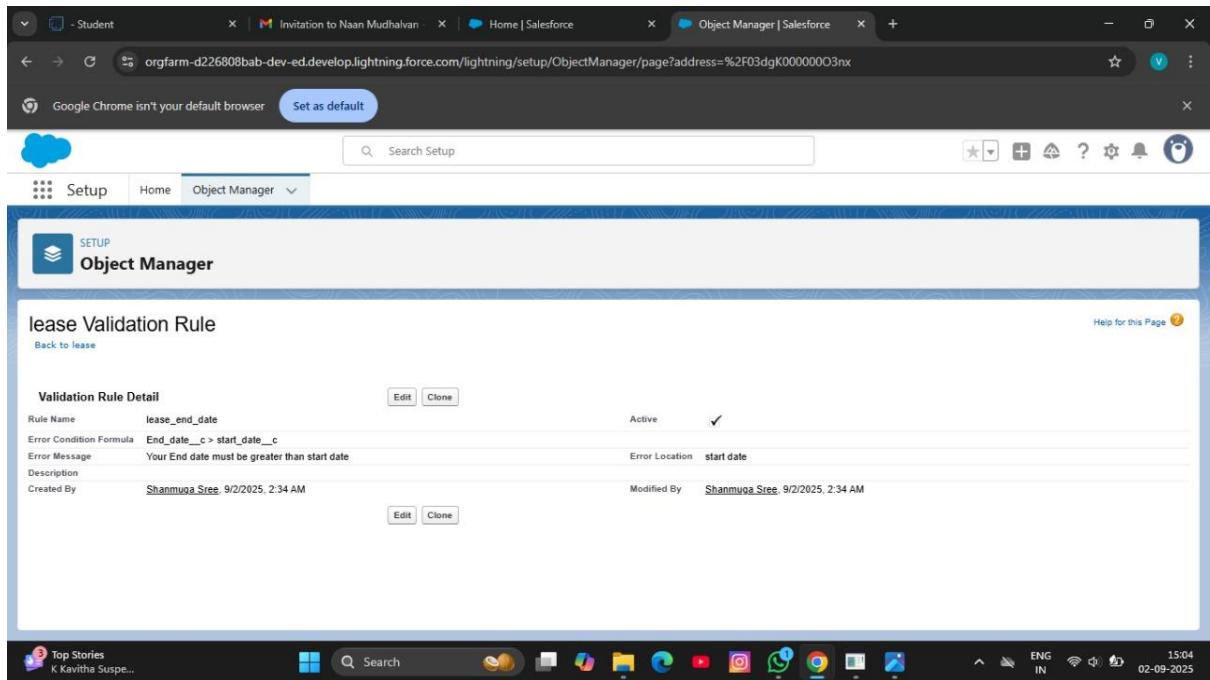
Type here to search 21:38 05-09-2025



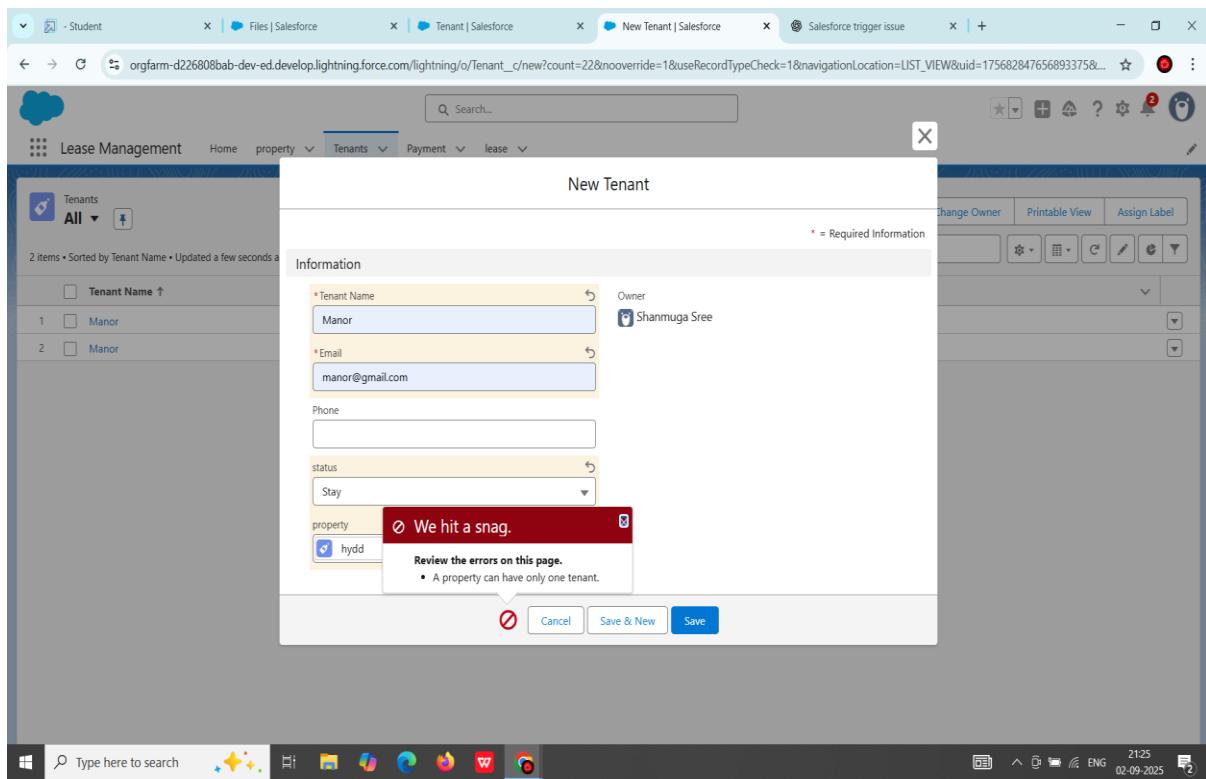
## TESTING APPROACH

- This project was tested mainly for validation rules, approval processes, flows, triggers, and reports. The testing ensured that data is accurate, automation works correctly, and system behavior meets requirements. Issues (snags) found during testing were corrected to improve reliability.

- **Functional Testing:**



- The validation rule was tested to ensure correct data entry. In this example, the system successfully prevented saving a record if the lease end date was earlier than the start date, proving that functional testing was achieved.
- **Performance Testing:**



- It verified that system responses were quick, and reports generated without delay for sample data, which proved that performance testing was achieved.
- During testing, an Apex trigger was executed to check business logic. At this stage, a snag was encountered, showing that the validation correctly restricted invalid data entry. This confirms that the system behaves as expected under rule conditions.

## FUTURE ENHANCEMENTS

- **Mobile App Integration:** Provide mobile access for tenants and property owners.
- **Online Payment Gateway:** Enable secure digital rent and lease payments.

- **AI Suggestions:** Use AI to recommend lease renewals, detect risks, and give smart insights.
- **Chatbot Integration:** Add a chatbot for quick tenant and landlord support.
- **Advanced Reporting:** Build interactive dashboards for better decision-making.

## CONCLUSION

This project shows how Salesforce tools like custom objects, validation rules, flows, and triggers can make lease management easier. Using Salesforce, the system became more accurate, faster, and ready to handle future needs.