

# **SHIVANI ENGINEERING COLLEGE**

Department of Artificial Intelligence and Data Science (AIDS)

**PROJECT TITLE:**

## **LEASE MANAGEMENT SYSTEM**

**Submitted by (Team Members):**

1. Sivaranjani C (Team Leader)
2. Nirmala S
- 3 . Nivetha N
4. Sarathi I

**Guided by:**

Mr. Jayashree K

**Platform:** Salesforce Developer

**Date of Submission:** 01/11/2025

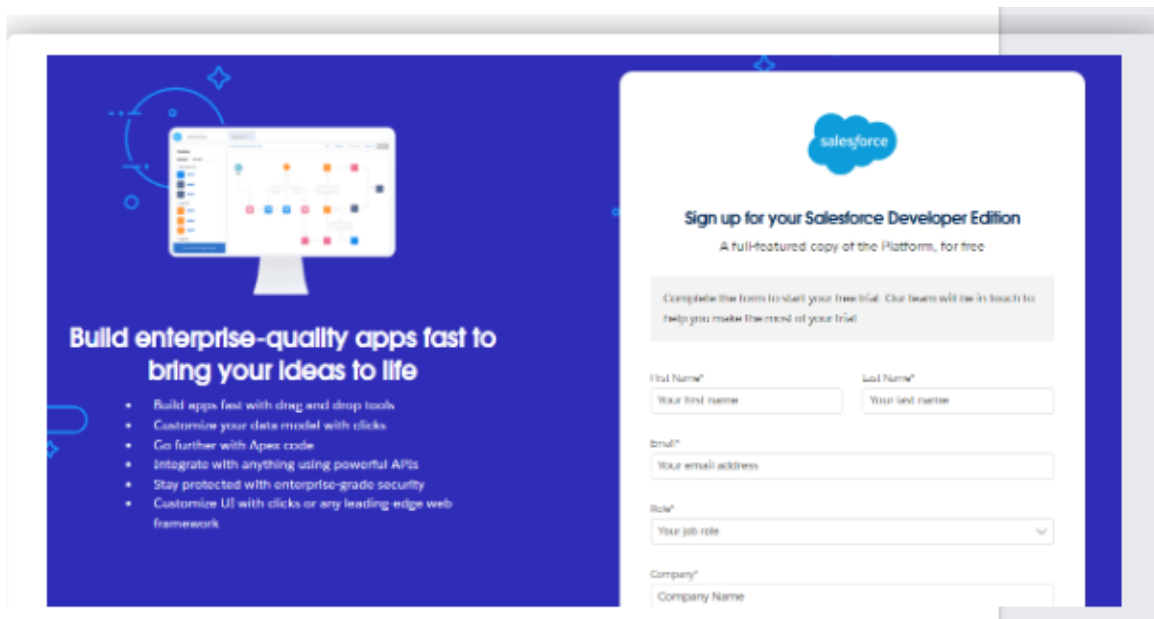
## **Project Description**

A lease management project involves creating a system or application to efficiently handle the processes related to leasing real estate properties, equipment, or other assets.

The goal is to streamline and automate various tasks associated with lease agreements, ensuring accurate record-keeping, compliance with regulations, and effective communication between parties involved.

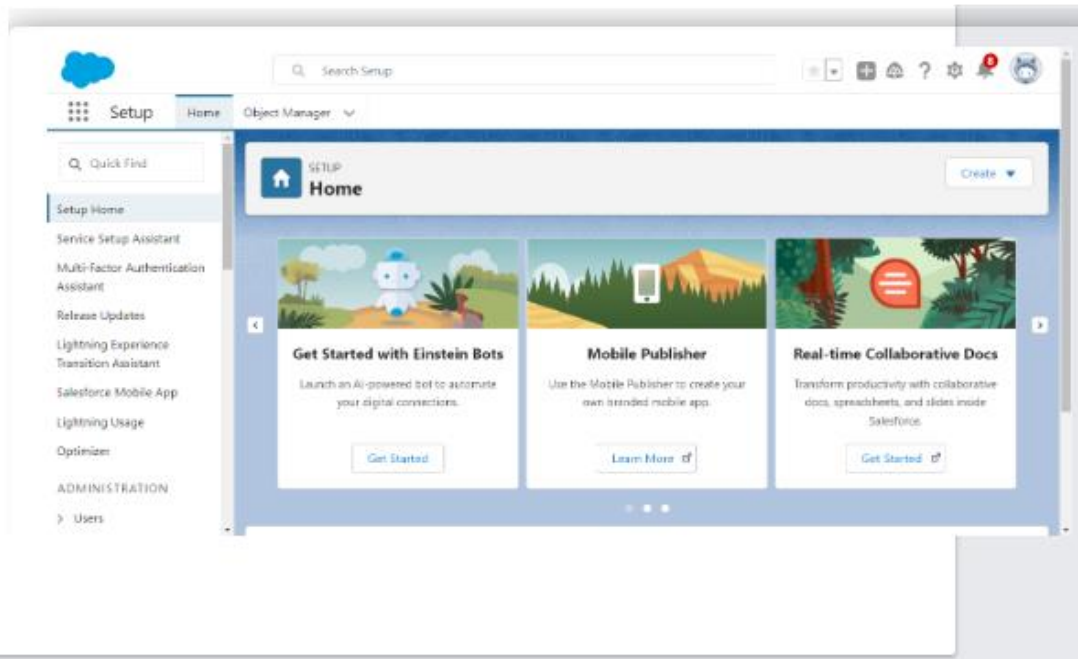
## **Creating Developer Account**

In this step, a new Salesforce Developer Account is created using a valid email address. This account provides access to the development environment for project setup.



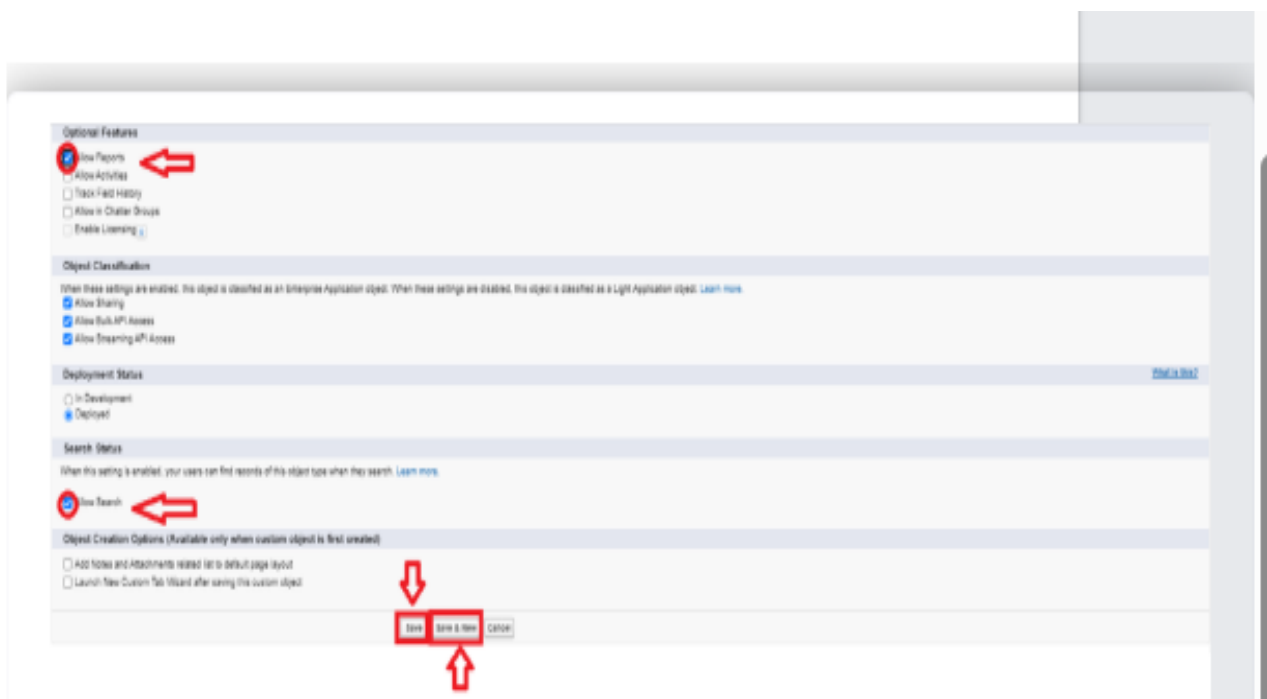
## **Account Activation**

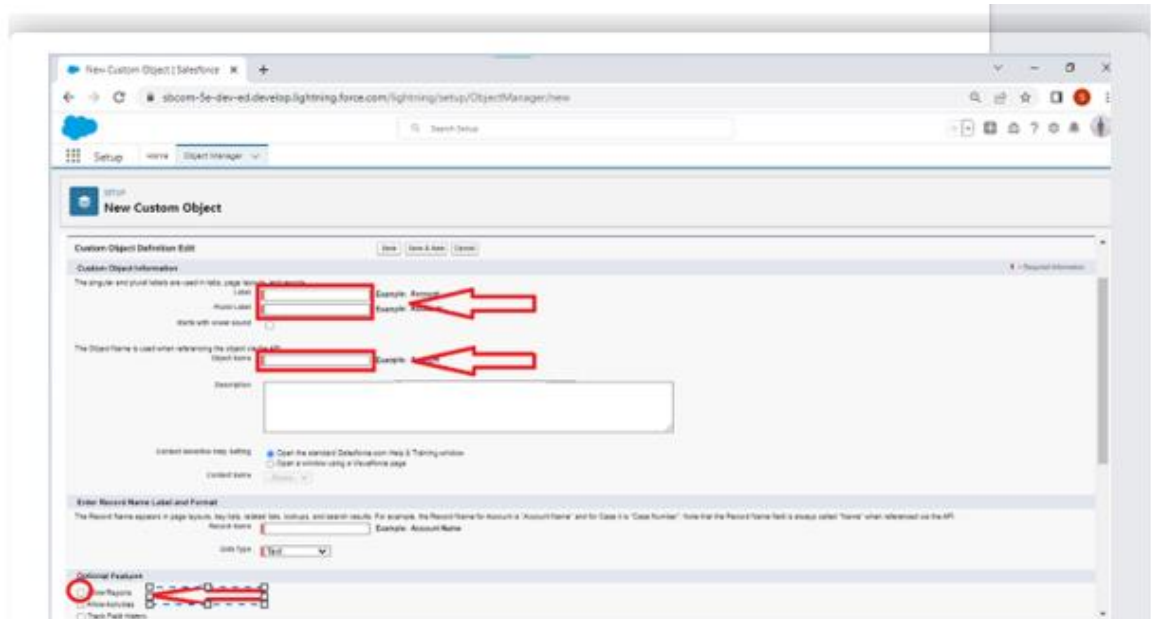
Activate the Salesforce Developer Account by verifying the registered email and completing the activation process.



## Create Property Object

Create a new custom object named “Property” to store details related to lease properties such as property name, type, and location.





## Create Tenant Object

Create a “Tenant” custom object to manage tenant information including name, contact, and lease duration.

## Create Payment Object

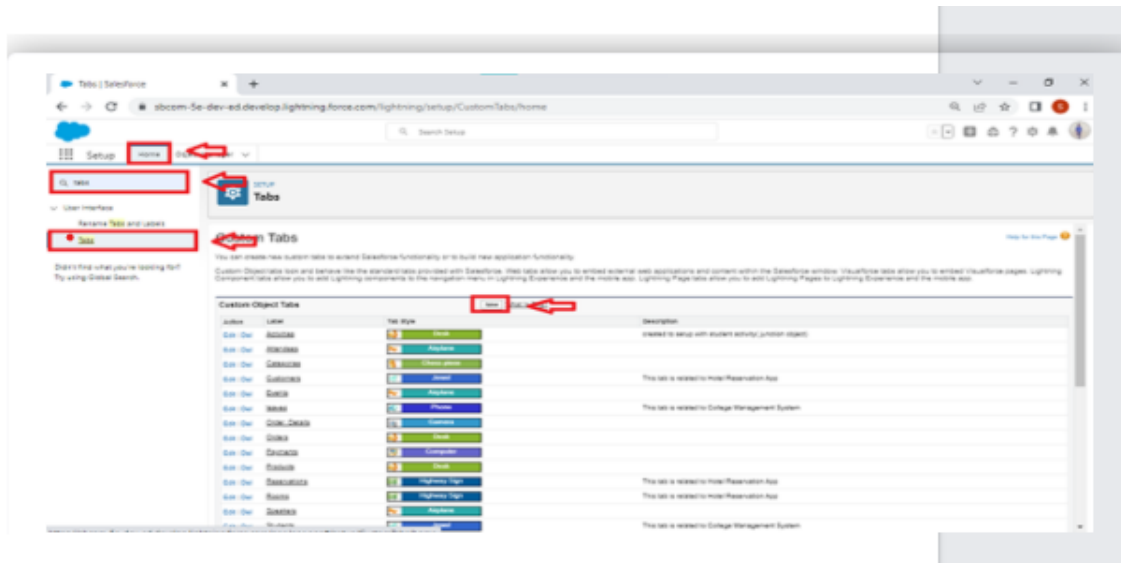
Design a “Payment” object to handle rent payments, due dates, and payment status tracking.

## Create Lease Object

Establish a “Lease” object to connect Property and Tenant objects and manage lease terms, start and end dates.

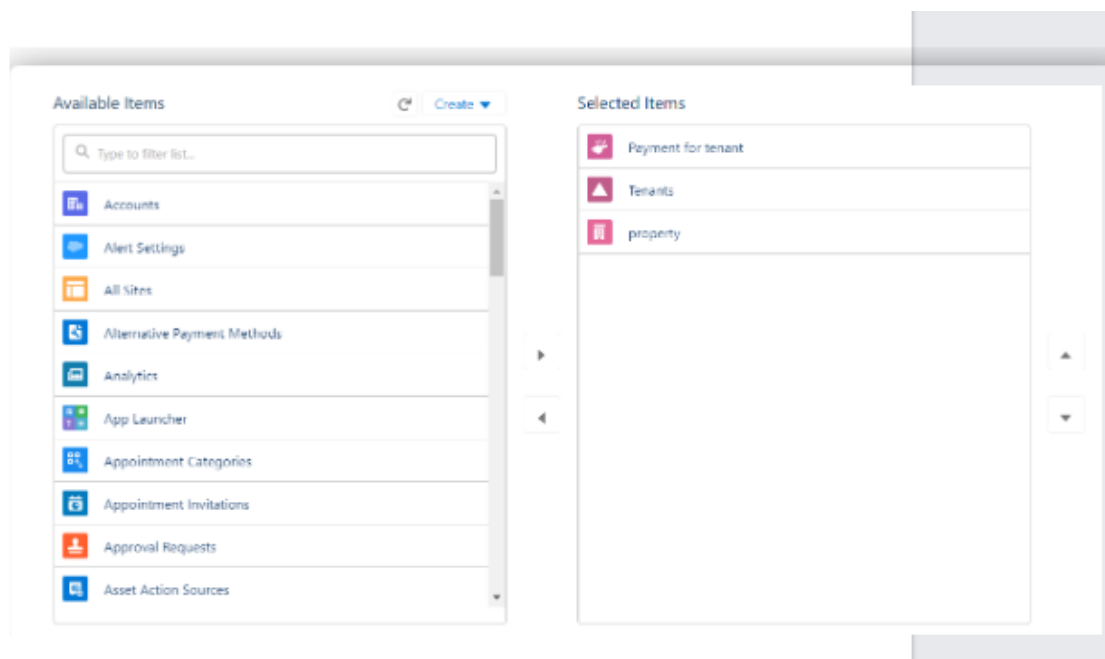
## Creating a Custom Tab

Add custom tabs for each created object to make them accessible in the Salesforce app navigation bar.



## Lightning App Creation

Create a custom Lightning App that includes all the custom objects and tabs for better project navigation. **Fields Creation**



# Fields Creation

In Salesforce, **Fields** are used to store different types of data inside each **Object**. Each object in the **Lease Management System** has specific fields that help to manage properties, tenants, leases, and payments efficiently.

The screenshot shows the Salesforce Setup - Object Manager interface for creating a new field. The left sidebar contains a navigation menu with the following items: Details, Fields & Relationships (selected), Page Layouts, Lightning Record Pages, Buttons, Links, and Actions, Compact Layouts, Field Sets, Object Limits, Record Types, Related Lookup Filters, Search Layouts, List View Button Layout, and Restriction Rules. The main content area is titled "Step 2: Enter the details" and "Step 2 of 4". It contains the following fields and options:

- Field Label:  (highlighted with a red box)
- Length:  (highlighted with a red box)
- Field Name:  (highlighted with a red box)
- Description:
- Help Text:
- Required: ☒ Always require a value in this field in order to save a record
- Unique: ☐ Do not allow duplicate values
  - ☐ Treat "ABC" and "abc" as duplicate values (case insensitive)
  - ☐ Treat "ABC" and "abc" as different values (case sensitive)
- External ID: ☐ Set this field as the unique record identifier from an external system
- Auto add to custom report type: ☒ Add this field to existing custom report types that contain this entity

At the bottom right, there are buttons for "Previous", "Next", and "Cancel". A red arrow points to the "Next" button.

# Email Template

An **Email Template** in Salesforce is a pre-designed message format that allows users to send standardized emails automatically or manually. It saves time, ensures consistency, and helps communicate important information like **lease confirmation**, **payment receipts**, and **reminders** to tenants

## 1. Tenant Leaving Notification

Sends an alert to the admin when a tenant requests to vacate the property.

## 2. Lease Approved

Notifies the tenant that their lease request has been approved successfully.

## 3. Lease Rejection Email

Informs the tenant that their lease application has been rejected due to missing details or eligibility.

## 4 . Monthly Payment Reminder

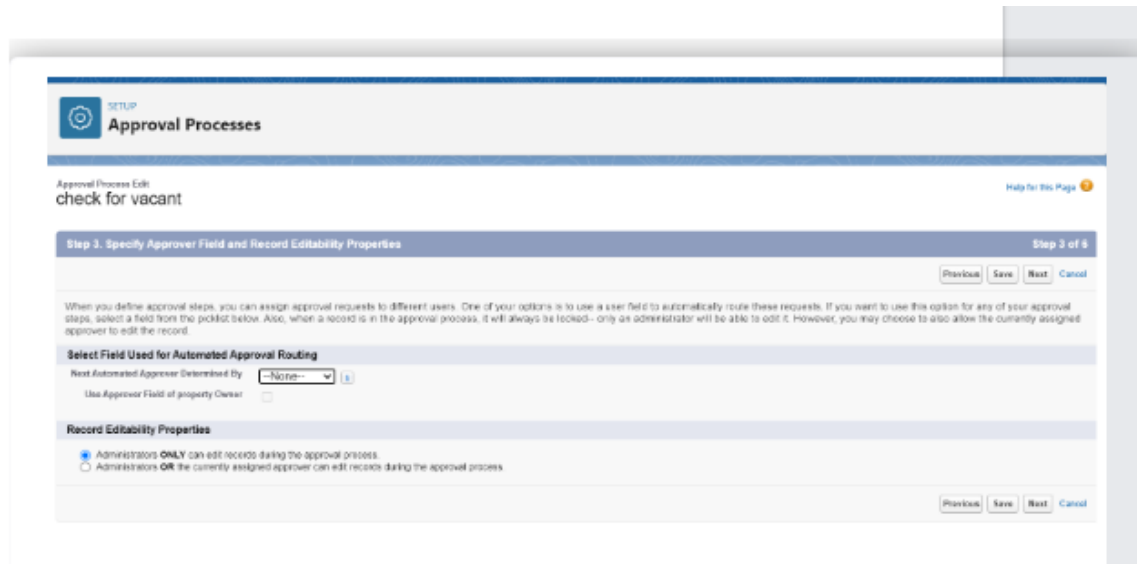
Automatically reminds the tenant each month about the upcoming rent payment due date.

## 5. Successful Payment Confirmation

Sends a thank-you message and confirmation once the tenant's rent payment is received successfully.

# Approval Process

An **Approval Process** automates how records are approved in Salesforce. In this project, an approval process is created for the **Lease or Payment** object to ensure that certain records (for example, rent amount > ₹50,000) need admin or manager approval before final confirmation.



The screenshot shows the Salesforce Setup page for Approval Processes. The page title is "Approval Processes" and the sub-header is "Approval Process Edit: check for vacant". The page is at Step 3 of 4, titled "Step 3. Specify Approver Field and Record Editability Properties".

When you define approval steps, you can assign approval requests to different users. One of your options is to use a user field to automatically route these requests. If you want to use this option for any of your approval steps, select a field from the picklist below. Also, when a record is in the approval process, it will always be locked - only an administrator will be able to edit it. However, you may choose to also allow the currently assigned approver to edit the record.

**Select Field Used for Automated Approval Routing**

Next Automated Approver Determined By:  [+](#)

Use Approver Field of property Owner: ☐

**Record Editability Properties**

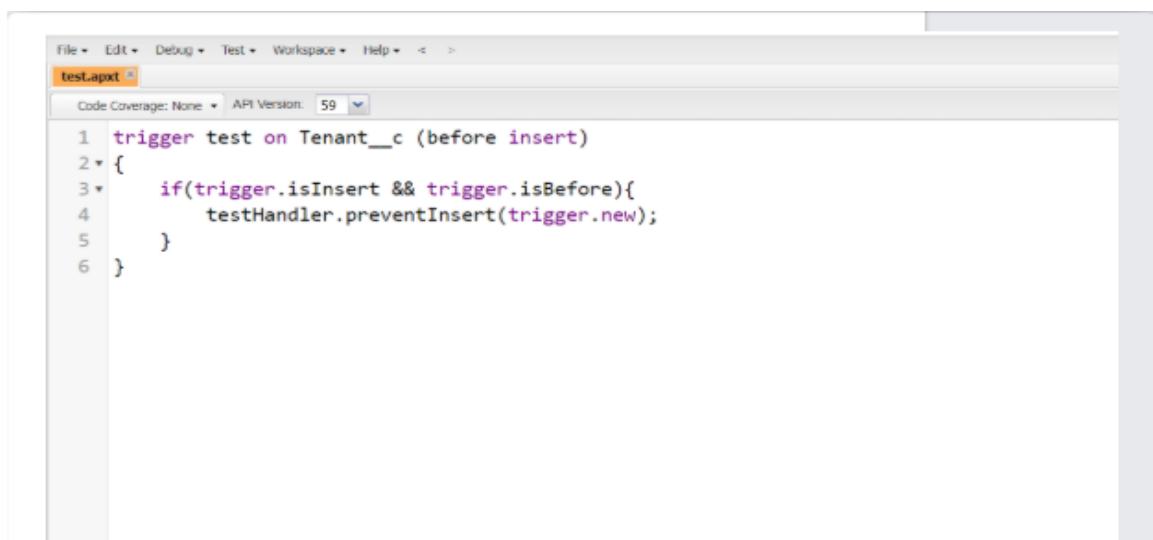
☒ Administrators **ONLY** can edit records during the approval process.

☐ Administrators **OR** the currently assigned approver can edit records during the approval process.

Buttons: Previous, Save, Next, Cancel

# Apex Trigger

An **Apex Trigger** is used to perform automatic actions when a record is created or updated. In this project, a trigger is written on the **Payment** object to automatically update the **Lease Status** once the payment is completed.



```
1 trigger test on Tenant__c (before insert)
2 {
3     if(trigger.isInsert && trigger.isBefore){
4         testHandler.preventInsert(trigger.new);
5     }
6 }
```

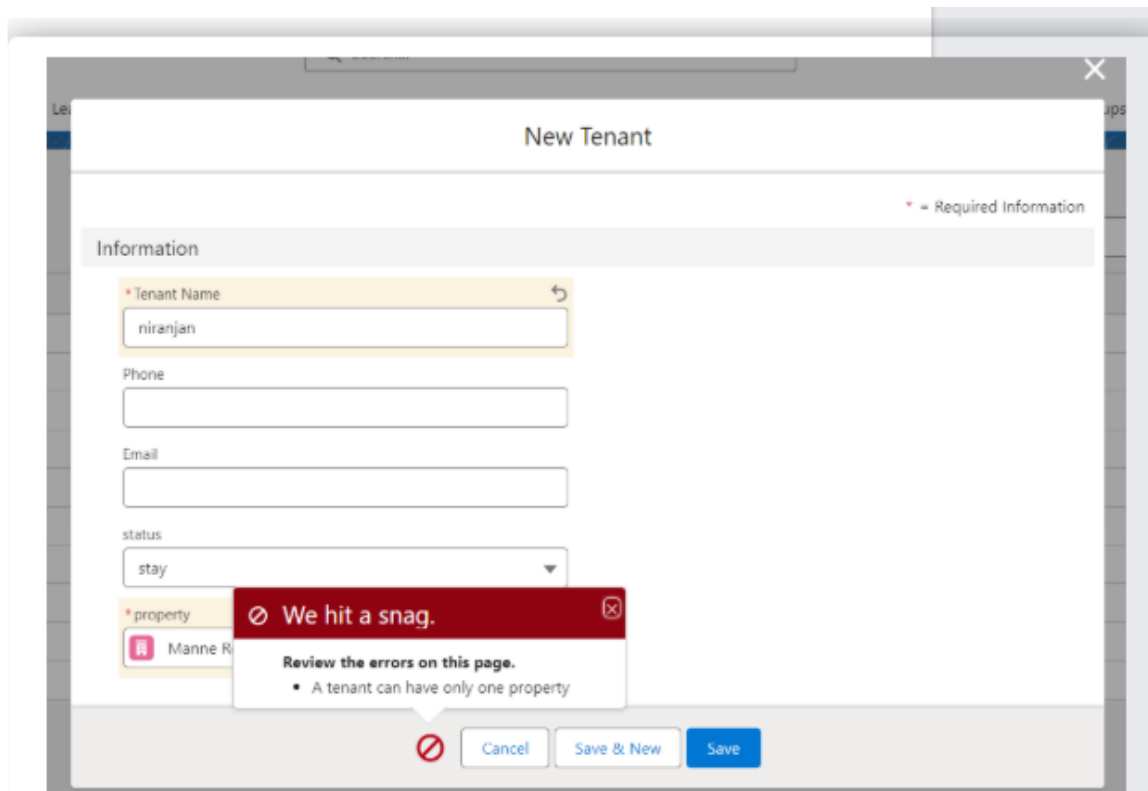


## Apex logic:

```
:
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');
            }
        }
    }
}
```

## Testing the Trigger:



The screenshot shows a web form titled "New Tenant" with a close button (X) in the top right corner. A legend indicates that a red asterisk (\*) denotes "Required Information". The form contains several fields: "Tenant Name" (required, with a refresh icon, containing the text "niranjan"), "Phone", "Email", "status" (a dropdown menu currently showing "stay"), and "property" (required, with a red error icon, containing the text "Manne R"). A red error message box is overlaid on the form, stating "We hit a snag." and "Review the errors on this page." with a list of errors: "A tenant can have only one property". At the bottom of the form, there are three buttons: a red "Cancel" button, a "Save & New" button, and a "Save" button.

## Flow Creation

A **Flow** is created to automate record updates or send email alerts without coding.

In this project, a **Record-Triggered Flow** is designed on the **Tenant** or **Payment** object to send an automatic email when a new payment is recorded.

## Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements

All Conditions Are Met (AND) ▼

Field	Operator	Value
check_for_paymet_c	Equals ▼	paid

+ Add Condition

✕

**When to Run the Flow for Updated Records** ⓘ

☒ Every time a record is updated and meets the condition requirements  
☐ Only when a record is updated to meet the condition requirements

**\* Optimize the Flow for:**

**Fast Field Updates**  
 Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

**Actions and Related Records**  
 Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

## Schedule Apex Class

A **Scheduled Apex Class** is used to run Apex code automatically at a specific time (daily, weekly, monthly).

For example — sending payment reminders, lease expiry alerts, or auto-updating records every morning.

**Schedule Apex**

Schedule an Apex class that implements the `Schedulable` interface to be automatically executed on a weekly or monthly interval.

Save Cancel

Job Name: MonthlyEmailScheduler

Apex Class: MonthlyEmailScheduler

Schedule Apex Location

Frequency: ☒ Weekly ☒ Monthly

On day: 1 of every month On: (Sun, 1st) of every month

Start: 04/12/2023 End: 04/01/2024

Preferred Start Time: 9:00 am

Exact start time will depend on job queue activity.

Save Cancel

## Conclusion

The *Lease Management System* project was successfully created using Salesforce. This project demonstrates how different Salesforce features such as **Objects, Tabs, Validation Rules, Email Templates, Approval Process, Flows, and Apex Triggers** can be used to automate and simplify lease management tasks. The system ensures efficient handling of tenant records, payments, and lease approvals with minimal manual work. Overall, this project improves productivity, reduces human error, and provides a digital solution for real-time property management.