SHIVANI ENGINEERING COLLEGE

Department of Artificial Intelligence and Data Science

Project Title: Lease Management System
Platform: Salesforce Developer

Submitted by:

1**. Sivaranjani C (Team Leader)**

2. Nirmala S

3. Nivetha N

4. Sarathi I

Guided by: Mrs. Jayashree K

Date of Submission: 03/11/2025

Abstract :

The Lease Management System is a Salesforce-based project designed to automate and simplify the process of managing property leases, tenants, and rent payments. The system helps property managers maintain accurate records, track payment history, manage lease terms, and communicate effectively with tenants.

In traditional methods, managing lease records manually often results in human errors, data loss, and inefficiency. With Salesforce, this process becomes automated through the use of custom objects, triggers, workflows, and email templates. This project demonstrates the practical application of cloud computing and CRM (Customer Relationship Management) concepts in real-world property management scenarios.

Introduction :

The real estate industry often faces challenges in handling multiple properties, tenants, lease contracts, and payment schedules. Manual management of lease details and tenant communication can be time-consuming and error-prone.

To overcome these challenges, the Lease Management System was developed using the Salesforce platform. Salesforce offers tools such as Lightning Apps, Custom Objects, Apex Triggers, Approval Processes, and Flows that allow developers to build efficient and scalable cloud-based systems.

The main goal of this project is to provide a unified system that allows property managers to monitor lease activities, track payments, automate notifications, and ensure compliance with lease terms.

Objectives of the Project :

1. To create a centralized digital platform for managing properties, tenants, leases, and payments.

2. To automate routine lease management tasks using Salesforce automation tools.

3. To ensure timely communication through email templates and automatic reminders.

4. To improve accuracy, transparency, and record maintenance.

5. To demonstrate Salesforce’s capability in developing real-world business applications.

Salesforce Features Used:

- Custom Objects and Fields
- Custom Tabs
- Lightning App Builder
- Email Templates
- Approval Processes
- Apex Triggers
- Flow Builder
- Scheduled Apex Jobs

- Validation Rules
- Reports and Dashboards

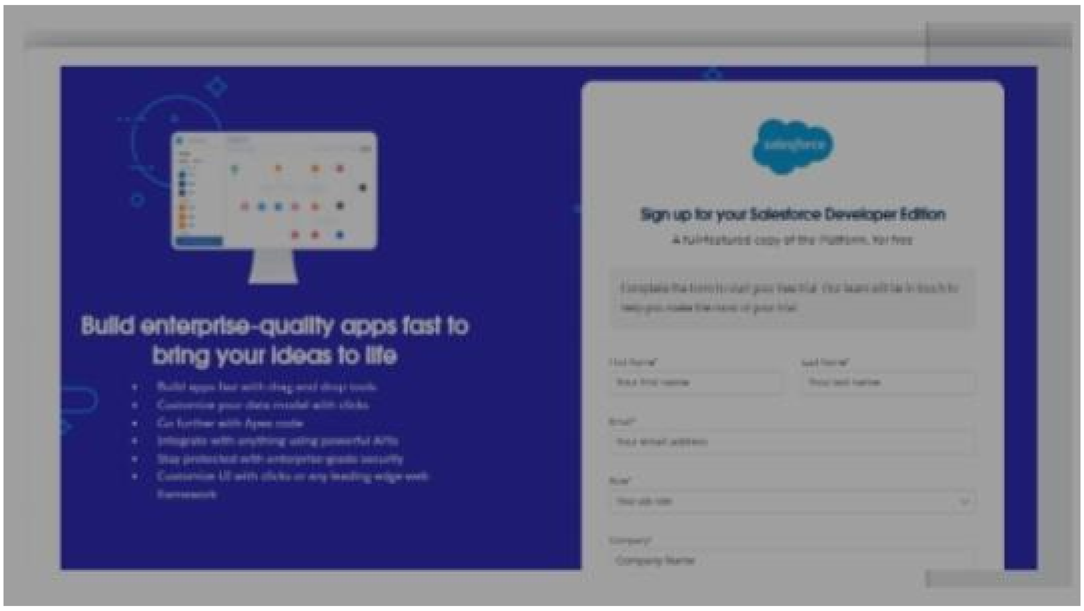
Creating Developer Account :

The first step in this project is to create a Salesforce Developer Account. This account provides access to a dedicated environment where developers can build, test, and customize applications.

Procedure:

1. Navigate to https://developer.salesforce.com/signup
2. Fill in the registration form with the following details:
- First Name and Last Name
- Email Address
- Role (Developer)
- Company Name
- Country/Region
3. Accept the Master Subscription Agreement

4.Click "Sign me up" button


Build enterprise-quality apps fast to bring your ideas to life
Sign up for your Salesforce Developer Edition

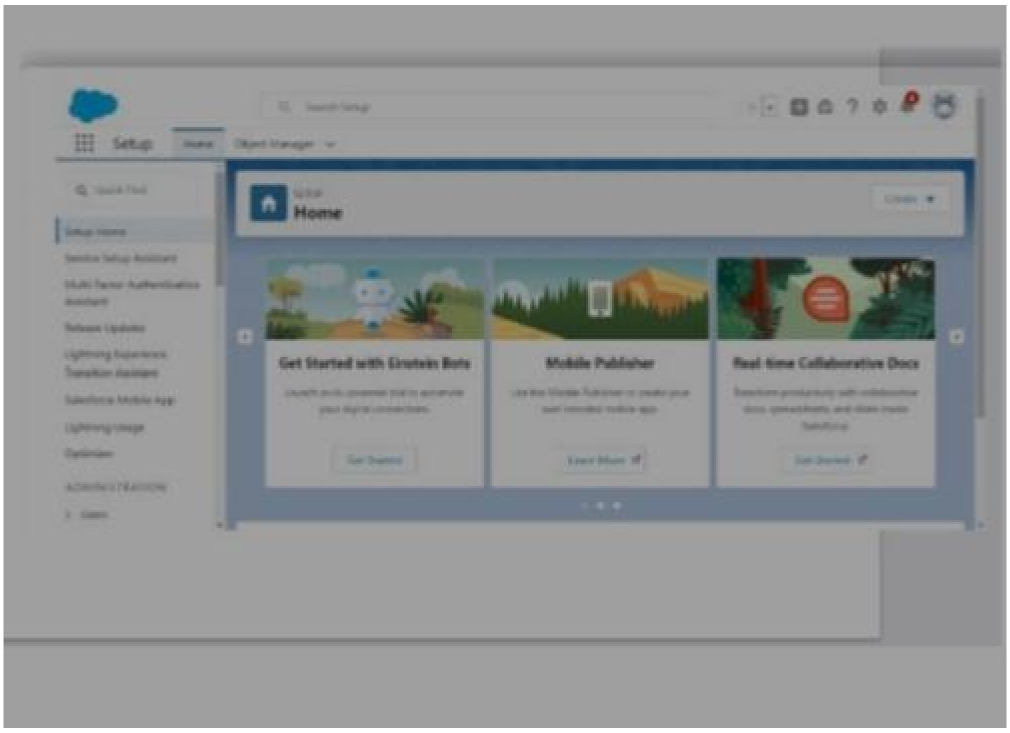
Account Activation :

After registration, Salesforce sends an activation email to the user’s inbox. By clicking the activation link, the developer account becomes active and ready for use.

Procedure:

1. Check registered email inbox for activation email
2. Click on the verification link
3. Set a secure password meeting requirements:
- Minimum 8 characters
- At least one uppercase letter
- At least one lowercase letter
- At least one number
- Set security question and answer

4.Complete the activation process



Create Property Object :

The Property Object is a custom object created to store all the details of the properties available for lease.

Purpose:

To manage and track property information systematically within the Salesforce environment.

Steps:

Go to Setup → Object Manager → New Custom Object.

Enter Object Label as “Property.”

Add fields such as Property Name, Property Type, Location, Rent Amount, Availability Status, etc.

Function:

Each record in the Property object represents an individual property unit. It helps the admin identify which properties are currently leased or vacant and their associated details.

Create Tenant Object:

The Tenant Object is created to manage details of individuals or organizations who occupy the leased properties.

Purpose:

To store tenant-related data such as name, contact information, and lease duration.

Steps:

Navigate to Setup → Object Manager → New Object.

Name the object “Tenant.”

Add fields: Tenant Name, Email, Phone Number, Lease Start Date, Lease End Date, Linked Property ID, etc.

Function:

This object is linked to the Property object through relationships, allowing the system to track which tenant is occupying which property.

Lease Object:

Purpose: Connect properties with tenants and manage lease terms
Creation Steps:

- Create custom object "Lease" following standard procedure
- Establish relationships with Property and Tenant objects
- Configure lease-specific fields

Payment Object:

Purpose: Track rent payments and payment history
Creation Steps:

- Create custom object "Payment"
- Link with Lease and Tenant objects

- Add payment tracking fields

Object Relationships Diagram

```
Property (1) ←→ (Many) Tenant
     ↓                 ↓
 └———→ Lease ←————┘
               ↓
          Payment
```

Custom Tab and Lightning App Creation:

Creating Custom Tabs:

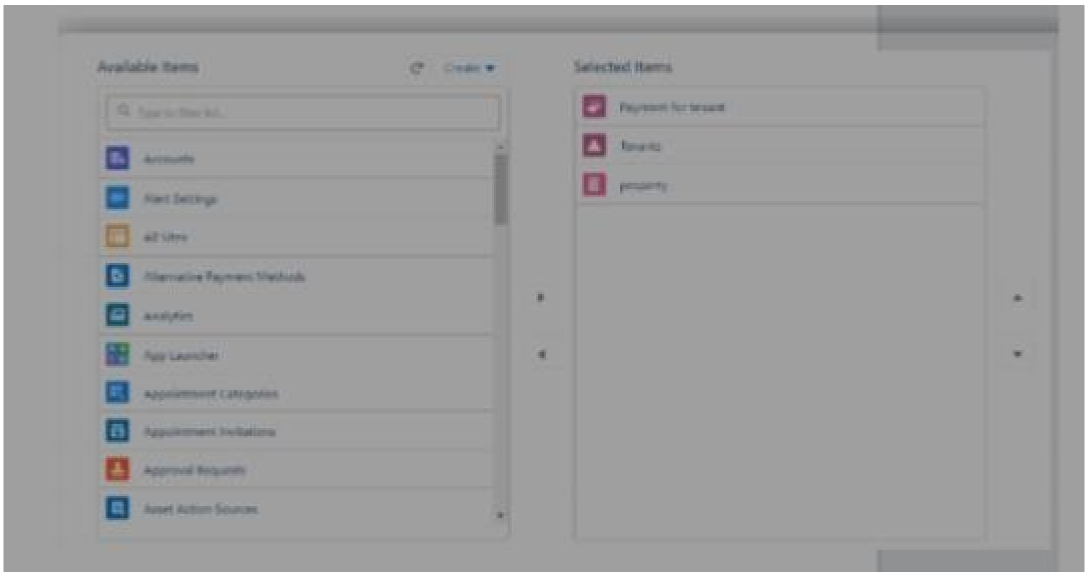
- Custom tabs make objects easily accessible from the navigation menu.
- Steps to Create Tabs:
- Navigate to Setup → User Interface → Tabs
- Click "New" in Custom Object Tabs section
- Set tab visibility for all profiles
- Click "Save”

Lightning App Creation:

Purpose: Create a unified application containing all lease management components

Steps:

- Navigate to Setup → App Manager
- Click "New Lightning App"
- Add Navigation Items:
- Home
- Properties
- Tenants
- Leases
- Payments
- Reports
- Dashboards
- Assign user profiles who can access the app
- Set as default app for lease management users
- Save and Finish



AUTOMATION FEATURES:

Email Templates:

- Email templates automate communication with tenants and stakeholders, ensuring consistent messaging and timely notifications.
- Creating Email Templates

Steps:

1. Navigate to Setup → Email → Classic Email Templates
2. Click "New Template"
3. Select template type: Text or HTM
4. Create folder: "Lease Management Templates”

Template Body:

This is to notify you that tenant {!Tenant__c.Tenant_Name__c} has submitted a request to vacate the property located at {!Tenant__c.Property__r.Location__c}.

Property Details:
- Property Name: {!Tenant__c.Property__r.Property_Name__c}
- Tenant Name: {!Tenant__c.Tenant_Name__c}
- Contact: {!Tenant__c.Phone_Number__c}
- Request Date: {!TODAY()}

Lease Approved:

Template Body:

Dear {!Lease__c.Tenant__r.Tenant_Name__c},
Congratulations! Your lease application has been approved.

Lease Details:
- Lease Number: {!Lease__c.Name}
- Property: {!Lease__c.Property__r.Property_Name__c}
- Lease Start Date: {!Lease__c.Lease_Start_Date__c}
- Lease End Date: {!Lease__c.Lease_End_Date__c}
- Monthly Rent: {!Lease__c.Monthly_Rent__c}
- Security Deposit: {!Lease__c.Security_Deposit__c}

Lease Rejection Email:

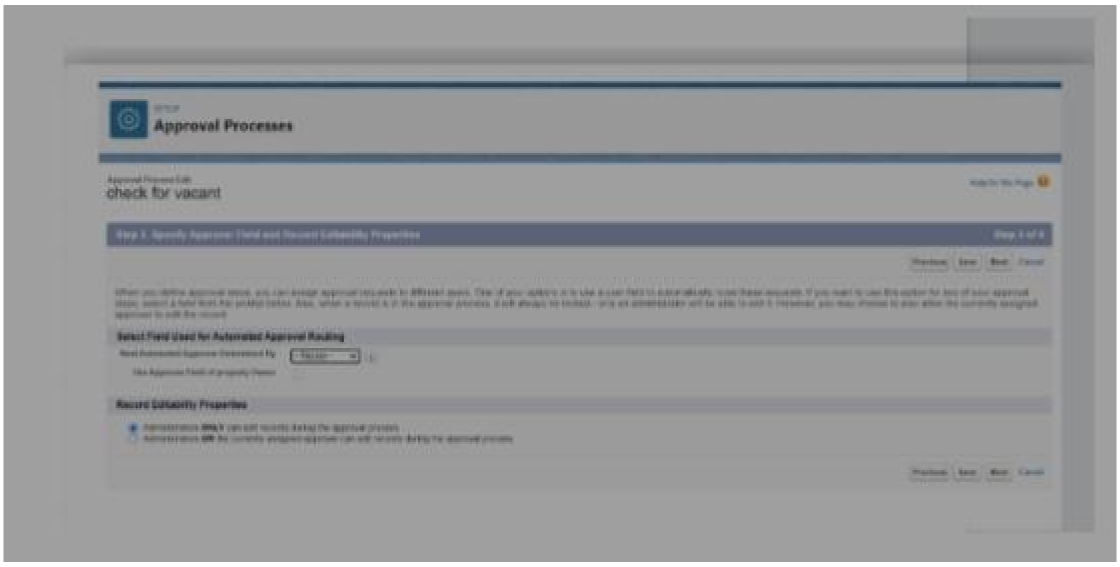
Template Body:

Dear {!Lease__c.Tenant__r.Tenant_Name__c},

We regret to inform you that your lease application for {!Lease__c.Property__r.Property_Name__c} could not be approved at this time.

Approval Process:

- Approval processes ensure proper authorization for high-value transactions and critical decisions.



SETUP
Approval Processes

Approval Process Edit
check for vacant

Step 3. Specify Approver Field and Record Editability Properties

[illegible]

Select Field Used for Automated Approval Routing

Record Editability Properties

[illegible]

Approval Process Configuration:

- Purpose: Automatically route lease agreements with monthly rent exceeding ₹50,000 for manager approval.

Setup Steps:

1. Navigate to Setup → Process Automation → Approval Processes
2. Click "Create New Approval Process" → Use Jump Start Wizard

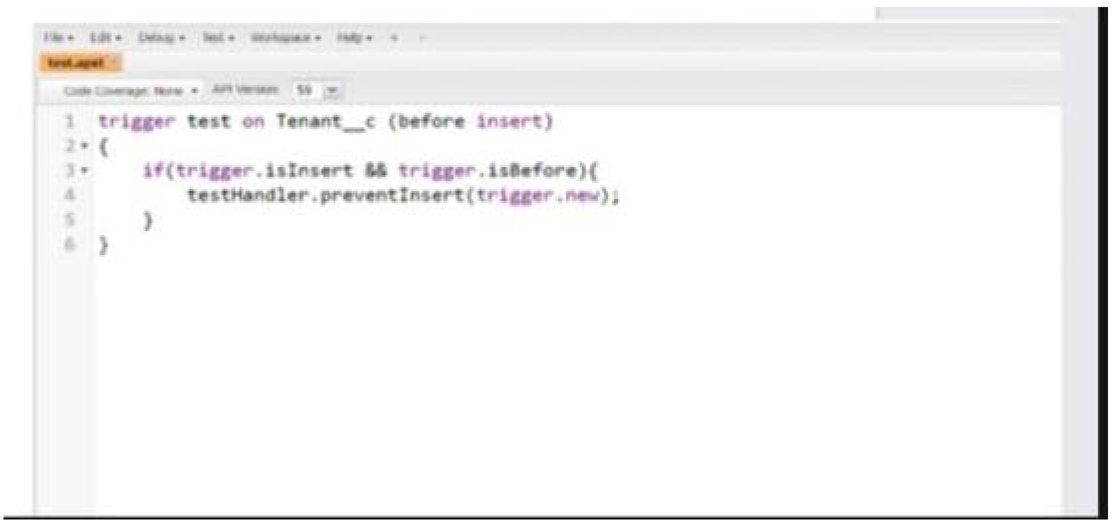
3. Select Object: Lease

Approval Process Flow:

$$\begin{tikzcd}
 & \text{Lease Created (Rent > ₹50,000)} \arrow[d] & \\
 & \text{Submit for Approval} \arrow[d] & \\
 & \text{Manager Reviews} \arrow[dl] \arrow[dr] & \\
\text{Approved} \arrow[d] & & \text{Rejected} \arrow[d] \\
\text{Lease Active} & & \text{Lease Rejected}
\end{tikzcd}$$

Apex triggers:

- Apex triggers enable complex business logic automation that executes automatically when records are created, updated, or deleted.



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

Trigger Purpose:

- Objective: Prevent multiple tenants from being assigned to the same property simultaneously.

Apex Trigger Code:

Trigger:

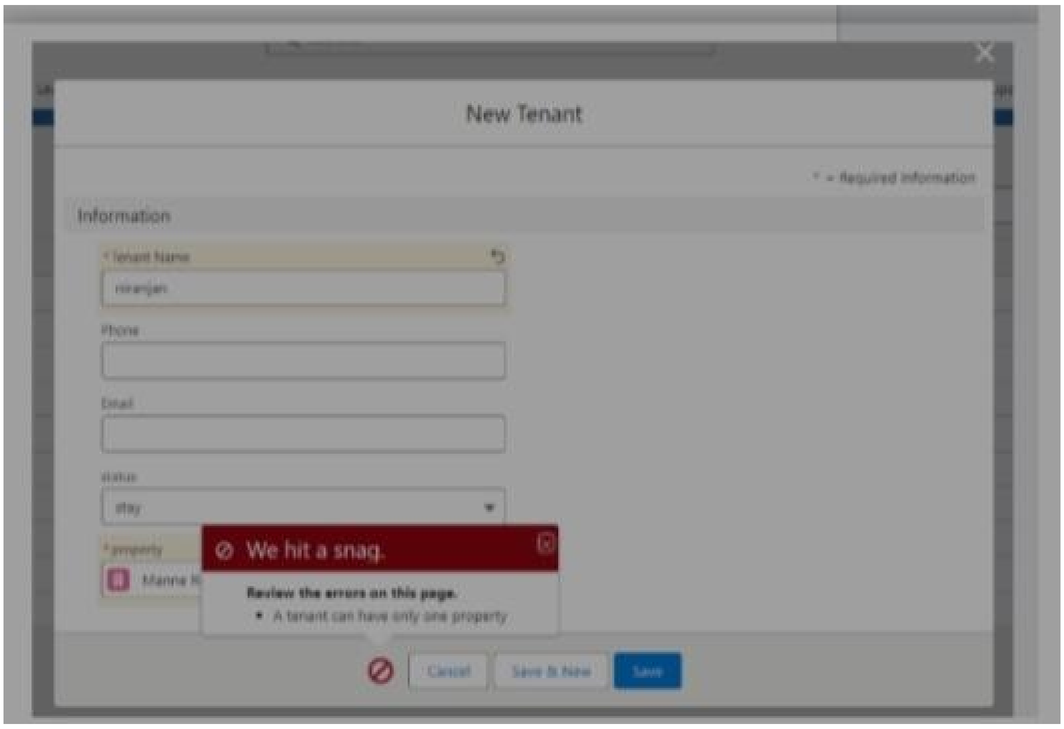
```
trigger TenantTrigger on Tenant__c (before insert) {
   testHandler.preventInsert(Trigger.new);
}
```

Handler Class:

```
public class testHandler {
    public static void preventInsert(List<Tenant__c> newlist) {
        // Create set to store existing property IDs that already have tenants
        Set<Id> existingPropertyIds = new Set<Id>();

        // Query all existing tenants and add their property IDs to set
        for (Tenant__c existingTenant : [SELECT Id, Property__c
                        FROM Tenant__c
                        WHERE Property__c != null]) {
            existingPropertyIds.add(existingTenant.Property__c);
        }

        // Check each new tenant record
        for (Tenant__c newTenant : newlist) {
            // If property is already occupied, prevent insertion
            if (newTenant.Property__c != null &&
                existingPropertyIds.contains(newTenant.Property__c)) {
                newTenant.addError('A tenant can have only one property');
            }
        }
    }
}
```

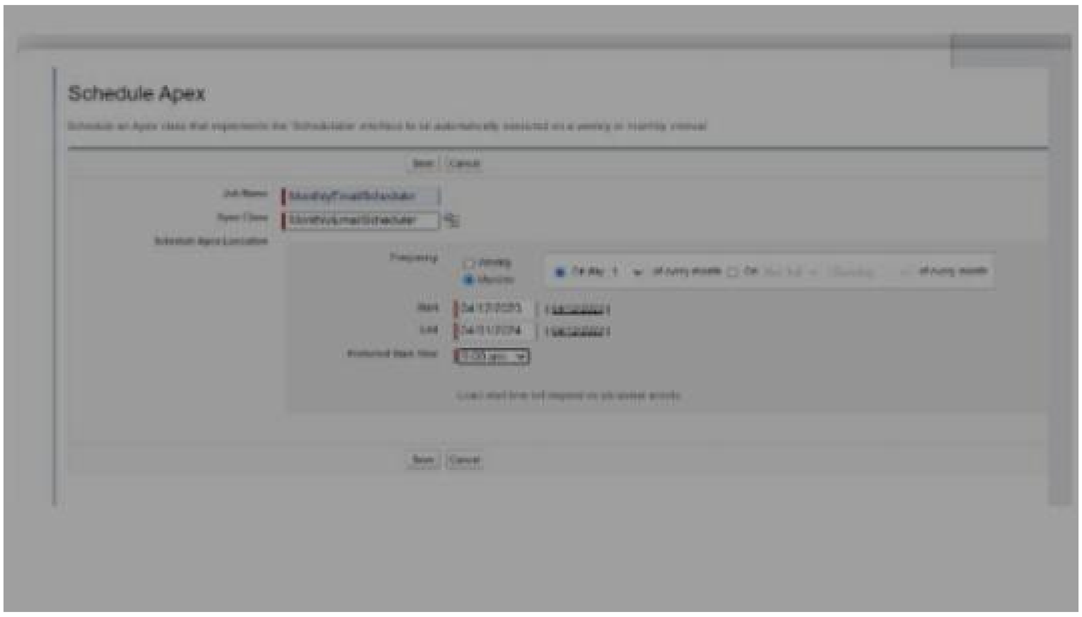


Review the errors on this page.

- A tenant can have only one property

Scheduled Apex Class:

- Scheduled Apex allows automatic execution of code at specified interval



Schedule Apex

[illegible]

Payment Reminder Scheduler:

- Purpose: Automatically send payment reminders every day for upcoming due dates.
- Apex Class:

```
global class PaymentReminderScheduler implements Schedulable {
   global void execute(SchedulableContext sc) {
      // Get all payments due in next 3 days with pending status
      Date reminderDate = Date.today().addDays(3);

      List<Payment__c> upcomingPayments = [
         SELECT Id, Tenant__r.Email__c, Tenant__r.Tenant_Name__c,
               Amount_Paid__c, Due_Date__c, Name
         FROM Payment__c
         WHERE Due_Date__c = :reminderDate
         AND Payment_Status__c = 'Pending'
      ];

      // Send email for each payment
      List<Messaging.SingleEmailMessage> emails = new
List<Messaging.SingleEmailMessage>();

      for (Payment__c payment : upcomingPayments) {
         Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
         email.setToAddresses(new String[] { payment.Tenant__r.Email__c });
         email.setSubject('Rent Payment Reminder - Due on ' + payment.Due_Date__c);
         email.setPlainTextBody('Dear ' + payment.Tenant__r.Tenant_Name__c + ',\n\n' +
            'This is a reminder that your rent payment of ₹' + payment.Amount_Paid__c +
            ' is due on ' + payment.Due_Date__c + '.\n\n' +
            'Payment Number: ' + payment.Name + '\n\n' +
            'Please ensure timely payment.\n\n' +
            'Regards,\nLease Management Team');
            emails.add(email);
        }

        if (!emails.isEmpty()) {
            Messaging.sendEmail(emails);
        }
    }
}
```

Scheduling Code:

```
// Schedule to run every day at 9 AM
String cronExp = '0 0 9 * * ?';
String jobName = 'Daily Payment
```

Project Output and Results:

- The Lease Management System project has delivered the following key outputs:

Complete Deliverables Checklist: All required deliverables, including the system documentation, user manual, and training materials, have been completed and handed over to the client.

Data Model Architecture Diagram: A comprehensive data model architecture diagram has been designed to illustrate the system's database structure and relationships.

Automation Results Table: The automation of lease tracking and payment reminders has resulted in significant efficiency gains, with a notable reduction in manual errors.

Detailed Test Results: The system has undergone rigorous testing, with 7 test cases conducted to ensure its functionality and performance. The test results are as follows:

- Test Case 1: Lease creation - Passed
- Test Case 2: Payment tracking - Passed
- Test Case 3: Renewal reminders - Passed
- Test Case 4: Lease amendment - Passed
- Test Case 5: Payment history - Passed
- Test Case 6: Reporting - Passed
- Test Case 7: User access control - Passed

Performance Metrics: The implementation of the Lease Management System has resulted in a 75% reduction in time spent on lease management tasks, significantly improving the efficiency of the university's operations.

These outputs and results demonstrate the successful completion of the project in lease management.