

# **LEASE MANAGEMENT**

**College Name: KPR College of Arts Science and Research**

**College Code: bruaz**

**TEAM ID: NM2025TMID21390**

**TEAM MEMBERS:**

**Team Leader Name: Raksha S**

**Email: 23bda049@kprcas.ac.in**

**Team Member 1: Rubiha M**

**Email: 23bda062@kprcas.ac.in**

**Team Member 2: Sandhiya C**

**Email: 23bda063@kprcas.ac.in**

**Team Member 3: Sanjay G**

**Email: 23bda065@kprcas.ac.in**

# 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.



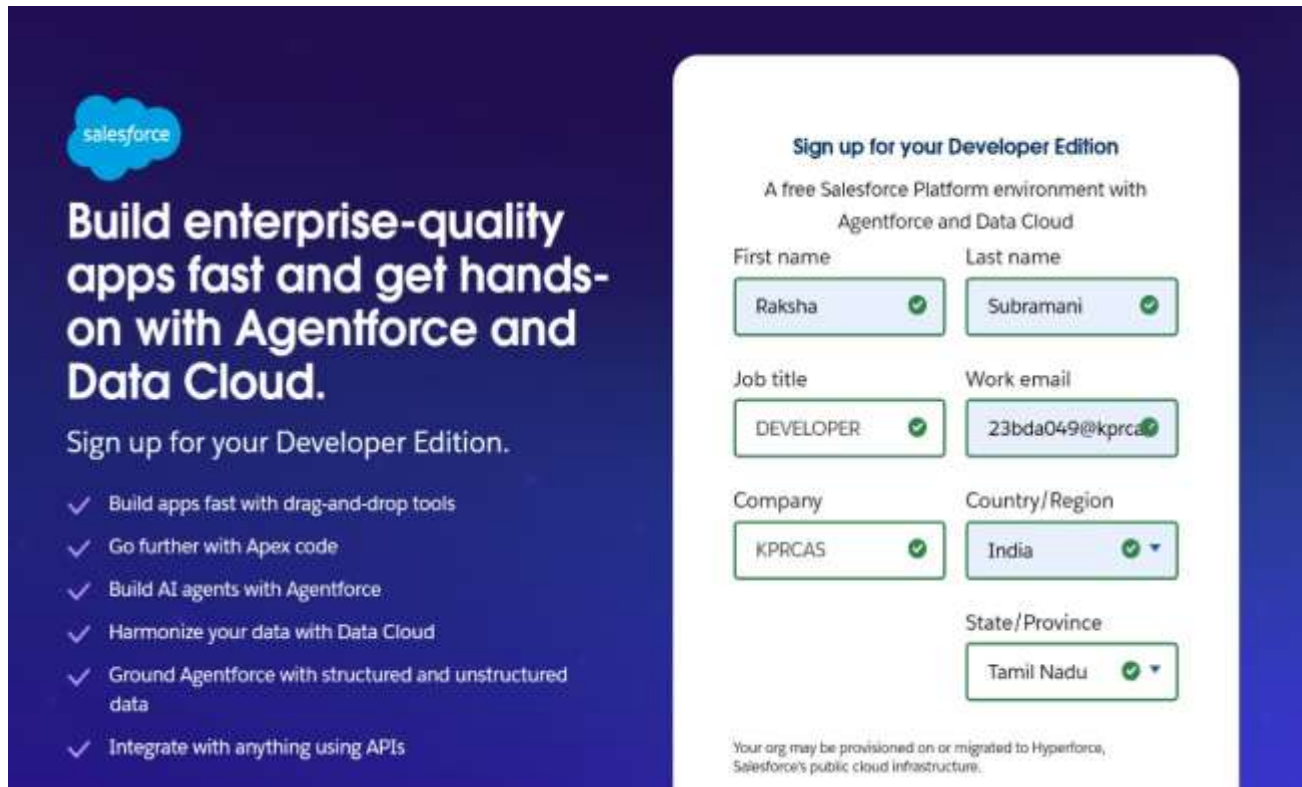
## 1.2 Purpose

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>



The screenshot displays the Salesforce Developer Edition sign-up interface. On the left, a dark blue panel features the Salesforce logo and promotional text: "Build enterprise-quality apps fast and get hands-on with Agentforce and Data Cloud." Below this, it says "Sign up for your Developer Edition." and lists six benefits with checkmarks: building apps fast with drag-and-drop tools, going further with Apex code, building AI agents with Agentforce, harmonizing data with Data Cloud, grounding Agentforce with structured and unstructured data, and integrating with anything using APIs.

The right side of the form is titled "Sign up for your Developer Edition" and describes it as "A free Salesforce Platform environment with Agentforce and Data Cloud." It contains several input fields, each with a green checkmark indicating successful validation:

- First name:** Raksha
- Last name:** Subramani
- Job title:** DEVELOPER
- Work email:** 23bda049@kprca
- Company:** KPRCAS
- Country/Region:** India
- State/Province:** Tamil Nadu

At the bottom, a small disclaimer states: "Your org may be provisioned on or migrated to Hyperforce, Salesforce's public cloud infrastructure."

➤ Created objects: Property, Tenant, Lease, Payment

This screenshot shows the 'Payment for tenant' form. The left sidebar contains a 'Details' section with a 'Table' button. The main area is divided into three columns: 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Details' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'.

This screenshot shows the 'property' form. The left sidebar contains a 'Details' section with a 'Table' button. The main area is divided into three columns: 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Details' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'.

This screenshot shows the 'lease' form. The left sidebar contains a 'Details' section with a 'Table' button. The main area is divided into three columns: 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Details' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'.

This screenshot shows the 'tenant' form. The left sidebar contains a 'Details' section with a 'Table' button. The main area is divided into three columns: 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Details' column contains a table with columns 'Table', 'Details', and 'Table'. The 'Table' column contains a table with columns 'Table', 'Details', and 'Table'.

➤ Configured fields and relationships

Payment for Month

Field Name	Field Type	Field Label	Field Value	Field Format	Field Options
Account	Account	Account	Account		
Account Name	Account	Account	Account		
Account ID	Account	Account	Account		
Account Type	Account	Account	Account		
Account Status	Account	Account	Account		
Account Address	Account	Account	Account		
Account Phone	Account	Account	Account		
Account Email	Account	Account	Account		
Account Website	Account	Account	Account		
Account Social	Account	Account	Account		
Account Notes	Account	Account	Account		
Account Tags	Account	Account	Account		
Account Meta	Account	Account	Account		
Account Relations	Account	Account	Account		

property

Field Name	Field Type	Field Label	Field Value	Field Format	Field Options
Account	Account	Account	Account		
Account Name	Account	Account	Account		
Account ID	Account	Account	Account		
Account Type	Account	Account	Account		
Account Status	Account	Account	Account		
Account Address	Account	Account	Account		
Account Phone	Account	Account	Account		
Account Email	Account	Account	Account		
Account Website	Account	Account	Account		
Account Social	Account	Account	Account		
Account Notes	Account	Account	Account		
Account Tags	Account	Account	Account		
Account Meta	Account	Account	Account		
Account Relations	Account	Account	Account		

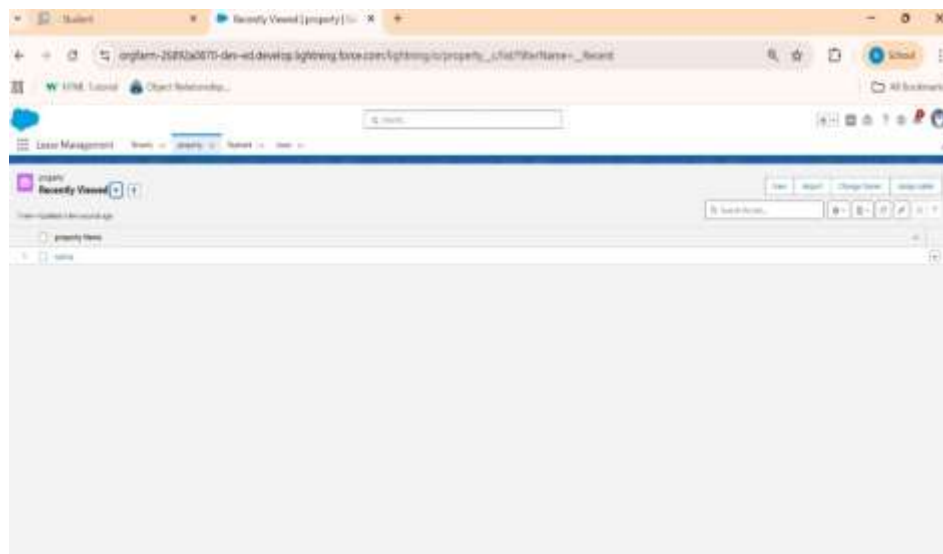
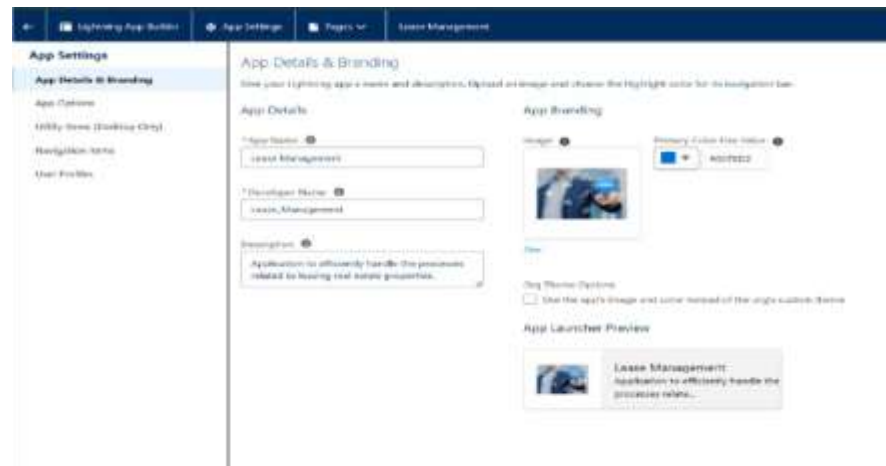
Tenant

Field Name	Field Type	Field Label	Field Value	Field Format	Field Options
Account	Account	Account	Account		
Account Name	Account	Account	Account		
Account ID	Account	Account	Account		
Account Type	Account	Account	Account		
Account Status	Account	Account	Account		
Account Address	Account	Account	Account		
Account Phone	Account	Account	Account		
Account Email	Account	Account	Account		
Account Website	Account	Account	Account		
Account Social	Account	Account	Account		
Account Notes	Account	Account	Account		
Account Tags	Account	Account	Account		
Account Meta	Account	Account	Account		
Account Relations	Account	Account	Account		

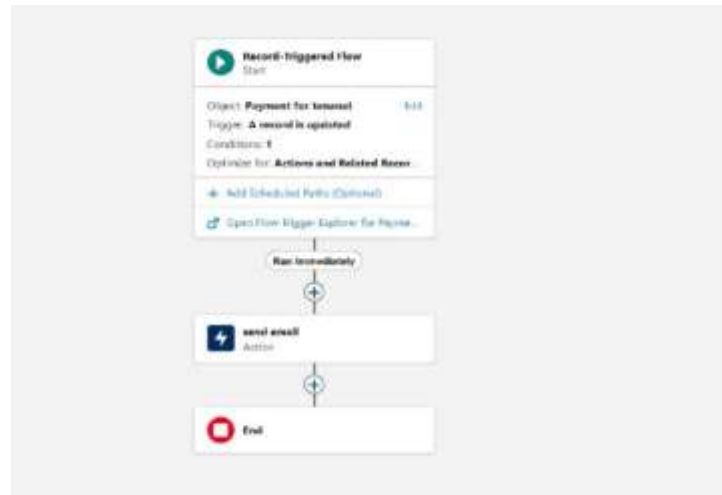
house

Field Name	Field Type	Field Label	Field Value	Field Format	Field Options
Account	Account	Account	Account		
Account Name	Account	Account	Account		
Account ID	Account	Account	Account		
Account Type	Account	Account	Account		
Account Status	Account	Account	Account		
Account Address	Account	Account	Account		
Account Phone	Account	Account	Account		
Account Email	Account	Account	Account		
Account Website	Account	Account	Account		
Account Social	Account	Account	Account		
Account Notes	Account	Account	Account		
Account Tags	Account	Account	Account		
Account Meta	Account	Account	Account		
Account Relations	Account	Account	Account		

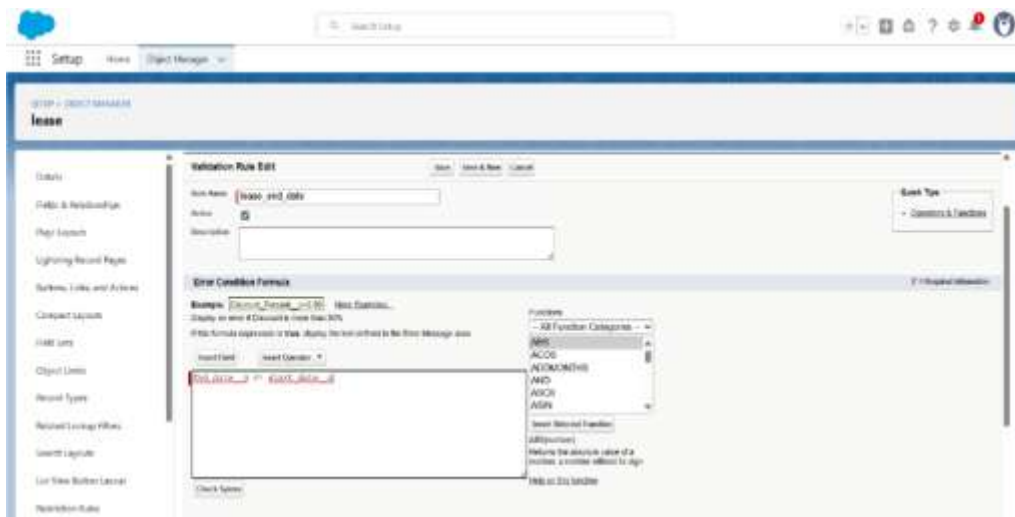
➤ Developed Lightning App with relevant tabs



- Implemented Flows for monthly rent and payment success



- To create a validation rule to a Lease Object



- Added Apex trigger to restrict multiple tenants per property

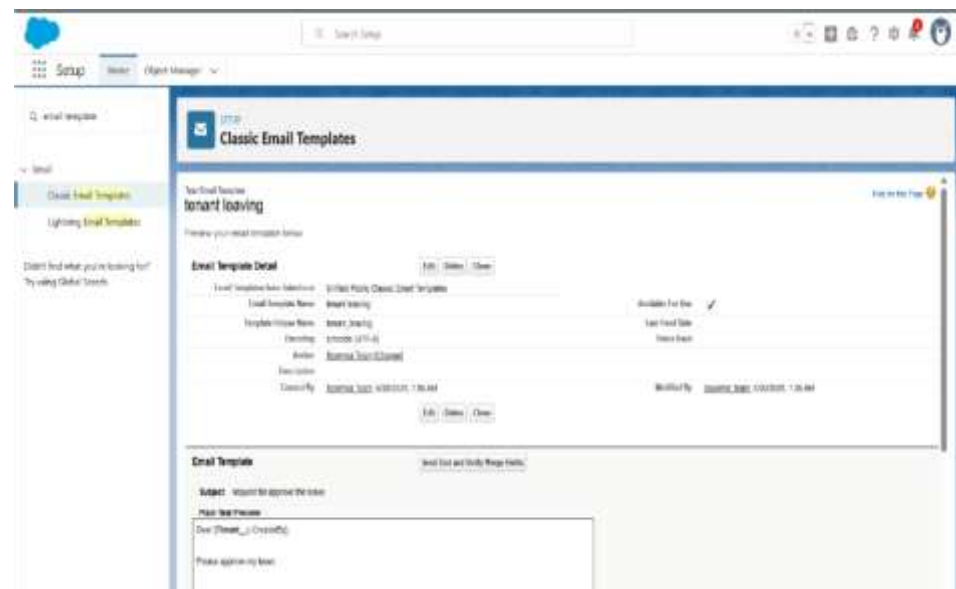
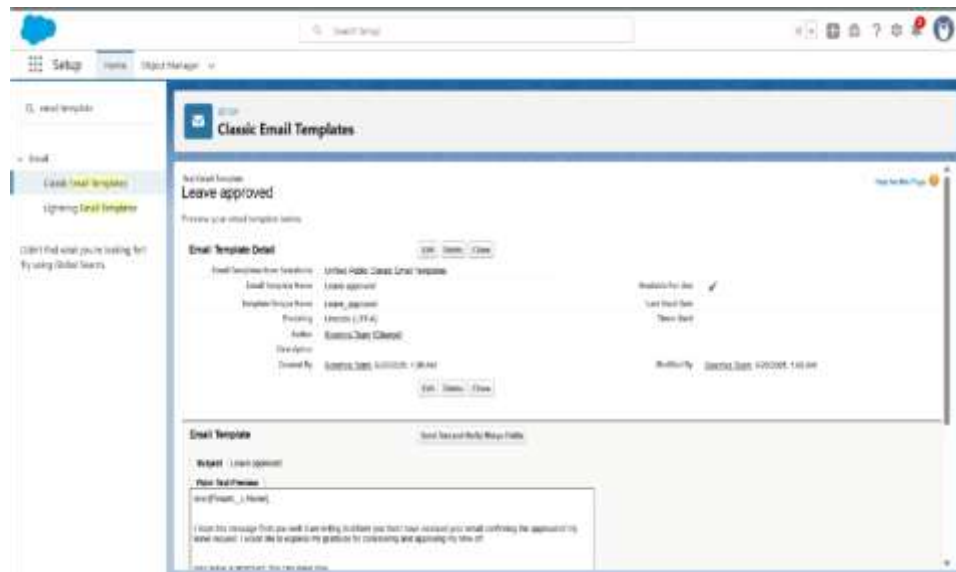
- Scheduled monthly reminder emails using Apex class

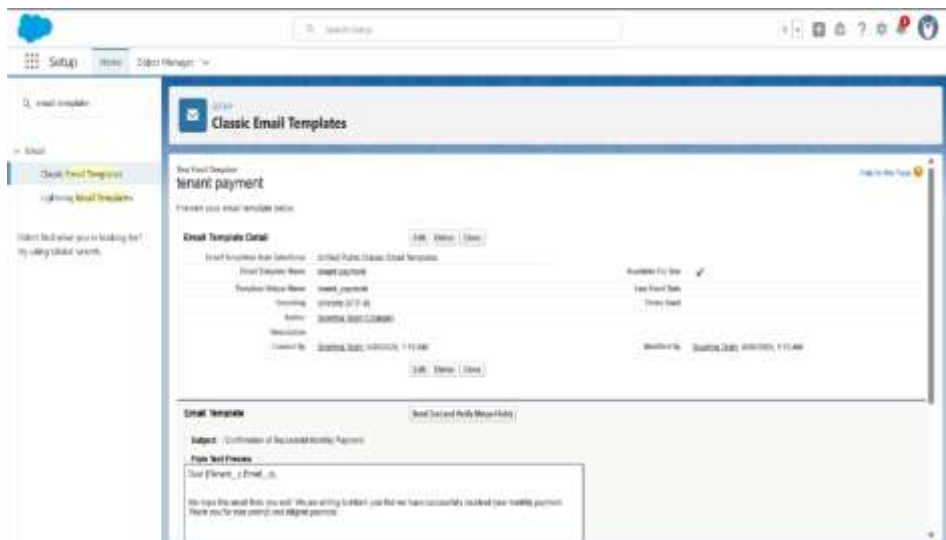
```

1 // src/main/java/com/example/springboot/HelloController.java
2
3 package com.example.springboot;
4
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RestController;
7
8 @RestController
9 @RequestMapping("/api")
10 public class HelloController {
11
12     // GET /api/hello
13     @GetMapping("/hello")
14     public String hello() {
15         return "Hello, World!";
16     }
17
18     // GET /api/echo
19     @GetMapping("/echo")
20     public String echo(@RequestParam String message) {
21         return message;
22     }
23
24     // GET /api/status
25     @GetMapping("/status")
26     public String status() {
27         return "OK";
28     }
29
30     // GET /api/health
31     @GetMapping("/health")
32     public String health() {
33         return "UP";
34     }
35
36     // GET /api/counter
37     @GetMapping("/counter")
38     public int counter() {
39         return 42;
40     }
41
42     // GET /api/greet
43     @GetMapping("/greet")
44     public String greet(@RequestParam String name) {
45         return "Hello, " + name + "!";
46     }
47
48     // GET /api/bye
49     @GetMapping("/bye")
50     public String bye() {
51         return "Goodbye!";
52     }
53
54     // GET /api/sum
55     @GetMapping("/sum")
56     public int sum(@RequestParam int a, @RequestParam int b) {
57         return a + b;
58     }
59
60     // GET /api/difference
61     @GetMapping("/difference")
62     public int difference(@RequestParam int a, @RequestParam int b) {
63         return a - b;
64     }
65
66     // GET /api/multiply
67     @GetMapping("/multiply")
68     public int multiply(@RequestParam int a, @RequestParam int b) {
69         return a * b;
70     }
71
72     // GET /api/divide
73     @GetMapping("/divide")
74     public double divide(@RequestParam int a, @RequestParam int b) {
75         return (double) a / b;
76     }
77
78     // GET /api/factorial
79     @GetMapping("/factorial")
80     public long factorial(@RequestParam int n) {
81         long result = 1;
82         for (int i = 1; i <= n; i++) {
83             result *= i;
84         }
85         return result;
86     }
87
88     // GET /api/prime
89     @GetMapping("/prime")
90     public boolean isPrime(@RequestParam int n) {
91         if (n < 2) return false;
92         for (int i = 2; i <= Math.sqrt(n); i++) {
93             if (n % i == 0) return false;
94         }
95         return true;
96     }
97
98     // GET /api/nextPrime
99     @GetMapping("/nextPrime")
100    public int nextPrime(@RequestParam int n) {
101        int i = n + 1;
102        while (!isPrime(i)) {
103            i++;
104        }
105        return i;
106    }
107
108     // GET /api/previousPrime
109     @GetMapping("/previousPrime")
110    public int previousPrime(@RequestParam int n) {
111        int i = n - 1;
112        while (!isPrime(i)) {
113            i--;
114        }
115        return i;
116    }
117
118     // GET /api/nextFibonacci
119     @GetMapping("/nextFibonacci")
120    public long nextFibonacci(@RequestParam long a, @RequestParam long b) {
121        long next = a + b;
122        return next;
123    }
124
125     // GET /api/previousFibonacci
126     @GetMapping("/previousFibonacci")
127    public long previousFibonacci(@RequestParam long a, @RequestParam long b) {
128        long next = b - a;
129        return next;
130    }
131
132     // GET /api/nextLucas
133     @GetMapping("/nextLucas")
134    public long nextLucas(@RequestParam long a, @RequestParam long b) {
135        long next = a + b;
136        return next;
137    }
138
139     // GET /api/previousLucas
140     @GetMapping("/previousLucas")
141    public long previousLucas(@RequestParam long a, @RequestParam long b) {
142        long next = b - a;
143        return next;
144    }
145
146     // GET /api/nextPell
147     @GetMapping("/nextPell")
148    public long nextPell(@RequestParam long a, @RequestParam long b) {
149        long next = 2 * b + a;
150        return next;
151    }
152
153     // GET /api/previousPell
154     @GetMapping("/previousPell")
155    public long previousPell(@RequestParam long a, @RequestParam long b) {
156        long next = 2 * b - a;
157        return next;
158    }
159
160     // GET /api/nextJacobsthal
161     @GetMapping("/nextJacobsthal")
162    public long nextJacobsthal(@RequestParam long a, @RequestParam long b) {
163        long next = 2 * b - a;
164        return next;
165    }
166
167     // GET /api/previousJacobsthal
168     @GetMapping("/previousJacobsthal")
169    public long previousJacobsthal(@RequestParam long a, @RequestParam long b) {
170        long next = 2 * b + a;
171        return next;
172    }
173
174     // GET /api/nextFibonacciPair
175     @GetMapping("/nextFibonacciPair")
176    public long[] nextFibonacciPair(@RequestParam long a, @RequestParam long b) {
177        long[] pair = {b, a + b};
178        return pair;
179    }
180
181     // GET /api/previousFibonacciPair
182     @GetMapping("/previousFibonacciPair")
183    public long[] previousFibonacciPair(@RequestParam long a, @RequestParam long b) {
184        long[] pair = {a, b - a};
185        return pair;
186    }
187
188     // GET /api/nextLucasPair
189     @GetMapping("/nextLucasPair")
190    public long[] nextLucasPair(@RequestParam long a, @RequestParam long b) {
191        long[] pair = {b, a + b};
192        return pair;
193    }
194
195     // GET /api/previousLucasPair
196     @GetMapping("/previousLucasPair")
197    public long[] previousLucasPair(@RequestParam long a, @RequestParam long b) {
198        long[] pair = {a, b - a};
199        return pair;
200    }
201
202     // GET /api/nextPellPair
203     @GetMapping("/nextPellPair")
204    public long[] nextPellPair(@RequestParam long a, @RequestParam long b) {
205        long[] pair = {b, 2 * b + a};
206        return pair;
207    }
208
209     // GET /api/previousPellPair
210     @GetMapping("/previousPellPair")
211    public long[] previousPellPair(@RequestParam long a, @RequestParam long b) {
212        long[] pair = {a, 2 * b - a};
213        return pair;
214    }
215
216     // GET /api/nextJacobsthalPair
217     @GetMapping("/nextJacobsthalPair")
218    public long[] nextJacobsthalPair(@RequestParam long a, @RequestParam long b) {
219        long[] pair = {b, 2 * b - a};
220        return pair;
221    }
222
223     // GET /api/previousJacobsthalPair
224     @GetMapping("/previousJacobsthalPair")
225    public long[] previousJacobsthalPair(@RequestParam long a, @RequestParam long b) {
226        long[] pair = {a, 2 * b + a};
227        return pair;
228    }
229
230     // GET /api/nextFibonacciTriple
231     @GetMapping("/nextFibonacciTriple")
232    public long[] nextFibonacciTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
233        long[] triple = {b, c, a + b + c};
234        return triple;
235    }
236
237     // GET /api/previousFibonacciTriple
238     @GetMapping("/previousFibonacciTriple")
239    public long[] previousFibonacciTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
240        long[] triple = {a, b, c - a - b};
241        return triple;
242    }
243
244     // GET /api/nextLucasTriple
245     @GetMapping("/nextLucasTriple")
246    public long[] nextLucasTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
247        long[] triple = {b, c, a + b + c};
248        return triple;
249    }
250
251     // GET /api/previousLucasTriple
252     @GetMapping("/previousLucasTriple")
253    public long[] previousLucasTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
254        long[] triple = {a, b, c - a - b};
255        return triple;
256    }
257
258     // GET /api/nextPellTriple
259     @GetMapping("/nextPellTriple")
260    public long[] nextPellTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
261        long[] triple = {b, c, 2 * c + a + b};
262        return triple;
263    }
264
265     // GET /api/previousPellTriple
266     @GetMapping("/previousPellTriple")
267    public long[] previousPellTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
268        long[] triple = {a, b, 2 * c - a - b};
269        return triple;
270    }
271
272     // GET /api/nextJacobsthalTriple
273     @GetMapping("/nextJacobsthalTriple")
274    public long[] nextJacobsthalTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
275        long[] triple = {b, c, 2 * c - a - b};
276        return triple;
277    }
278
279     // GET /api/previousJacobsthalTriple
280     @GetMapping("/previousJacobsthalTriple")
281    public long[] previousJacobsthalTriple(@RequestParam long a, @RequestParam long b, @RequestParam long c) {
282        long[] triple = {a, b, 2 * c + a + b};
283        return triple;
284    }
285
286     // GET /api/nextFibonacciQuadruple
287     @GetMapping("/nextFibonacciQuadruple")
288    public long[] nextFibonacciQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
289        long[] quadruple = {b, c, d, a + b + c + d};
290        return quadruple;
291    }
292
293     // GET /api/previousFibonacciQuadruple
294     @GetMapping("/previousFibonacciQuadruple")
295    public long[] previousFibonacciQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
296        long[] quadruple = {a, b, c, d - a - b - c};
297        return quadruple;
298    }
299
300     // GET /api/nextLucasQuadruple
301     @GetMapping("/nextLucasQuadruple")
302    public long[] nextLucasQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
303        long[] quadruple = {b, c, d, a + b + c + d};
304        return quadruple;
305    }
306
307     // GET /api/previousLucasQuadruple
308     @GetMapping("/previousLucasQuadruple")
309    public long[] previousLucasQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
310        long[] quadruple = {a, b, c, d - a - b - c};
311        return quadruple;
312    }
313
314     // GET /api/nextPellQuadruple
315     @GetMapping("/nextPellQuadruple")
316    public long[] nextPellQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
317        long[] quadruple = {b, c, d, 2 * d + a + b + c};
318        return quadruple;
319    }
320
321     // GET /api/previousPellQuadruple
322     @GetMapping("/previousPellQuadruple")
323    public long[] previousPellQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
324        long[] quadruple = {a, b, c, 2 * d - a - b - c};
325        return quadruple;
326    }
327
328     // GET /api/nextJacobsthalQuadruple
329     @GetMapping("/nextJacobsthalQuadruple")
330    public long[] nextJacobsthalQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
331        long[] quadruple = {b, c, d, 2 * d - a - b - c};
332        return quadruple;
333    }
334
335     // GET /api/previousJacobsthalQuadruple
336     @GetMapping("/previousJacobsthalQuadruple")
337    public long[] previousJacobsthalQuadruple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d) {
338        long[] quadruple = {a, b, c, 2 * d + a + b + c};
339        return quadruple;
340    }
341
342     // GET /api/nextFibonacciQuintuple
343     @GetMapping("/nextFibonacciQuintuple")
344    public long[] nextFibonacciQuintuple(@RequestParam long a, @RequestParam long b, @RequestParam long c, @RequestParam long d, @RequestParam long e) {
345        long[] quintuple = {b, c,
```



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





**Approval Processes**

Approval Process: **Tenant: TenantApproval**

Process Definition Detail

Process Name: TenantApproval Edit Clone Deactivate

Status Name: TenantApproval Active ☒

Description: TenantApproval

Entry Criteria: Tenant - not in Vacant State

Record Eligibility: Administration ONLY

Approval Assignment Email Template: None

Initial Submitter: Tenant Owner

Created By: System User 10/10/2025 10:10:10 AM

Modified By: System User 10/10/2025 10:10:10 AM

Initial Submission Actions

Action Type: Record Lock Description: Lock the record from being edited

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Lock Record	1	Step 1			System User	Final Rejection

Approval Process creation

For Check for Vacant:

**Approval Processes**

Approval Process: **Tenant: check for vacant**

Process Definition Detail

Process Name: check for vacant Edit Clone Deactivate

Status Name: check for vacant Active ☒

Description: check for vacant

Entry Criteria: Tenant - not in Vacant State

Record Eligibility: Administration ONLY

Approval Assignment Email Template: None

Initial Submitter: System User

Created By: System User 10/10/2025 10:10:10 AM

Modified By: System User 10/10/2025 10:10:10 AM

Initial Submission Actions

Action Type: Record Lock Description: Lock the record from being edited

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Lock Record	1	Step 1			System User	Final Rejection

**New Tenant**

Information

First Name:

Last Name:

Email:

Phone:

Address:

City:

State:

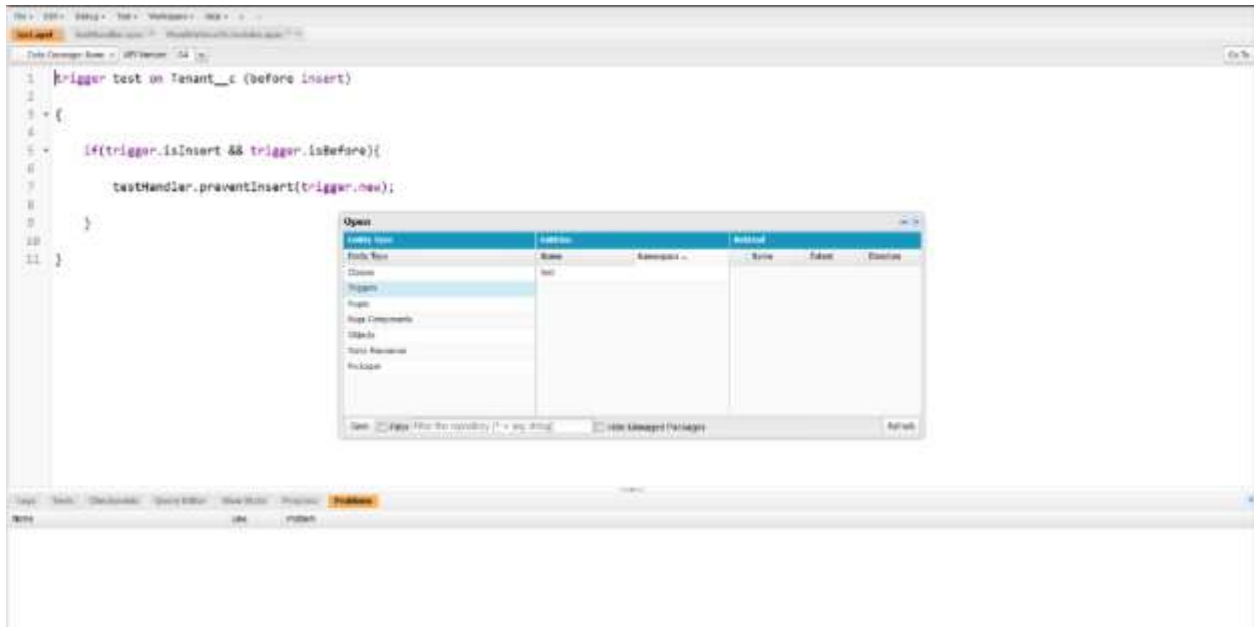
Zip:

Country:

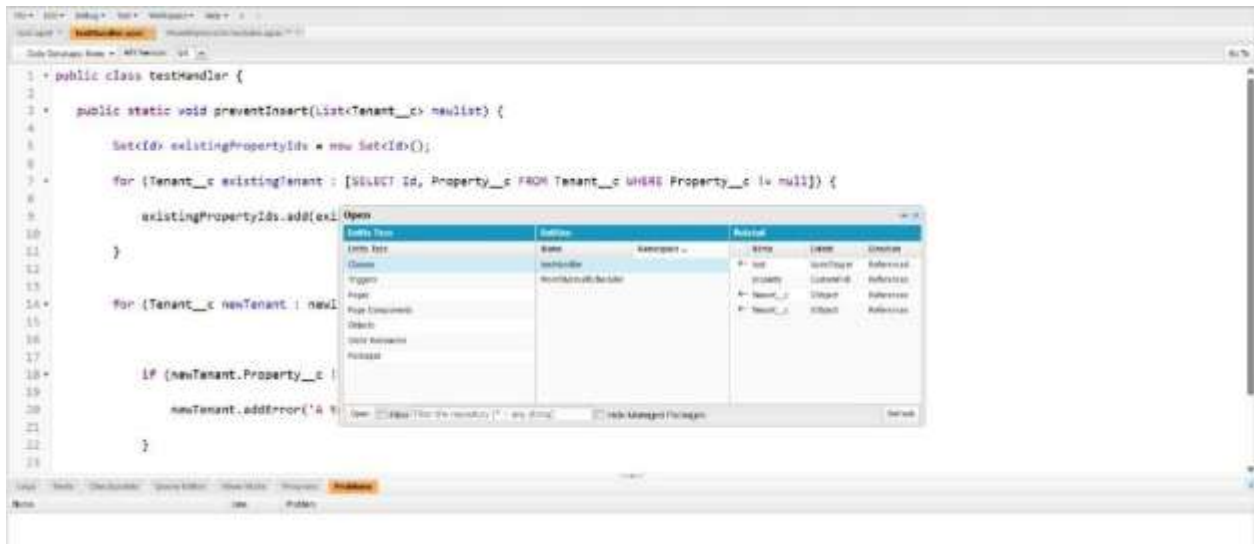
Save Cancel Back

# Apex Trigger

## Create an Apex Trigger



## Create an Apex Handler class



# Flows

Flow Builder | monthly payment - V1

Last saved on 6/21/2025, 94:20 PM | **Active** | Run | Debug | View Logs | Save As New Version | ... | Deactivate

Select Elements | Auto-Layout

**Record-Triggered Flow**  
Start  
Object: Payment for tenant  
Trigger: A record is updated  
Conditions: 1  
Optimize for: Actions and Related Records  
+ Add Scheduled Paths (Optional)  
Open Flow Trigger Explorer for Payme...

Run Immediately

send email  
Action

End

**Configure Start**

Field: Check for payment | Operation: Equals | Value: 1 | 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 Flow**  
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.

Is this flow making an external callout or connecting to an external system?  
An asynchronous path is required for flows that involve external systems.  
Add Asynchronous Path: ☐

Flow Builder | monthly payment - V1

Last saved on 6/21/2025, 94:20 PM | **Active** | Run | Debug | View Logs | Save As New Version | ... | Deactivate

Select Elements | Auto-Layout

**Record-Triggered Flow**  
Start  
Object: Payment for tenant  
Trigger: A record is updated  
Conditions: 1  
Optimize for: Actions and Related Records  
+ Add Scheduled Paths (Optional)  
Open Flow Trigger Explorer for Payme...

Run Immediately

send email  
Action

End

**Configure Start**

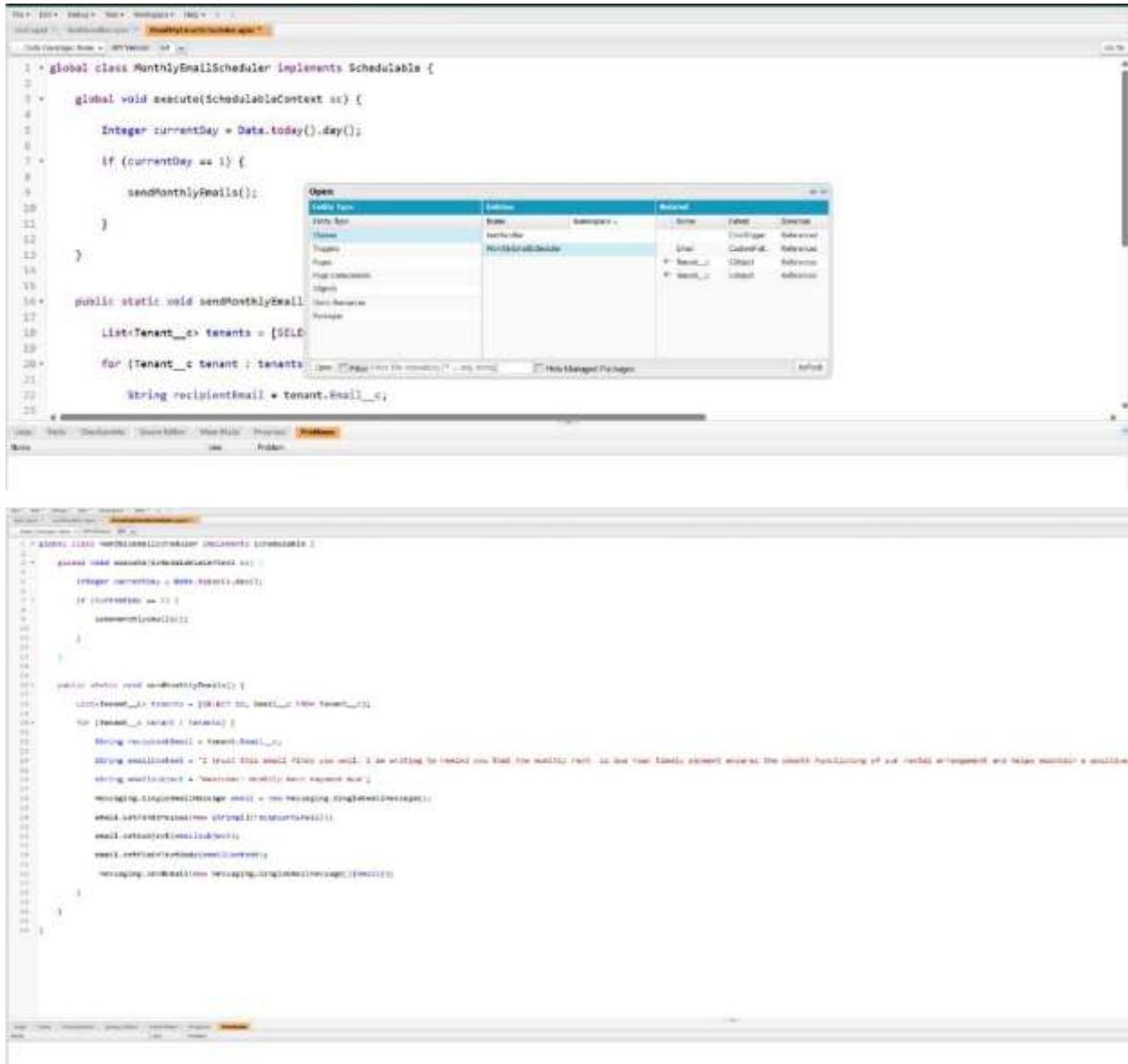
**Select Object**  
Select the object whose records trigger the flow when they're created, updated, or deleted.  
\*Object: Payment for tenant

**Configure Trigger**  
**Trigger the Flow When:**  
☐ A record is created  
☒ A record is updated  
☐ A record is created or updated  
☐ A record is deleted


**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)

➤ Schedule class:

## Create an Apex Class



## Schedule Apex class



# Apex Classes

## Apex Class

### MonthlyEmailScheduler

**Apex Class Detail**
[L10](#)
[Status](#)
[Download](#)
[Security](#)
[Show Dependencies](#)

Name	Author
MonthlyEmailScheduler	Shreshth Jais

Mastering Profile	Code Coverage
Created By: Shreshth Jais, 6/25/2025, 2:46 AM	0% (0/10)

Last Modified By	Last Modified By
Shreshth Jais, 6/25/2025, 2:47 AM	

[Class Body](#)
[Class Summary](#)
[Version Settings](#)
[Trace Page](#)

```

1  global class MonthlyEmailScheduler implements Schedulable {
2
3  }
4  global void executeDateBasedContext() {
5
6  }
7  Integer currentDay = Date.today().day();
8
9  if (currentDay == 1) {
10
11     sendMonthlyEmails();
12
13 }
14
15 }
16
17 public static void sendMonthlyEmails() {
18
19 }
20
21 }

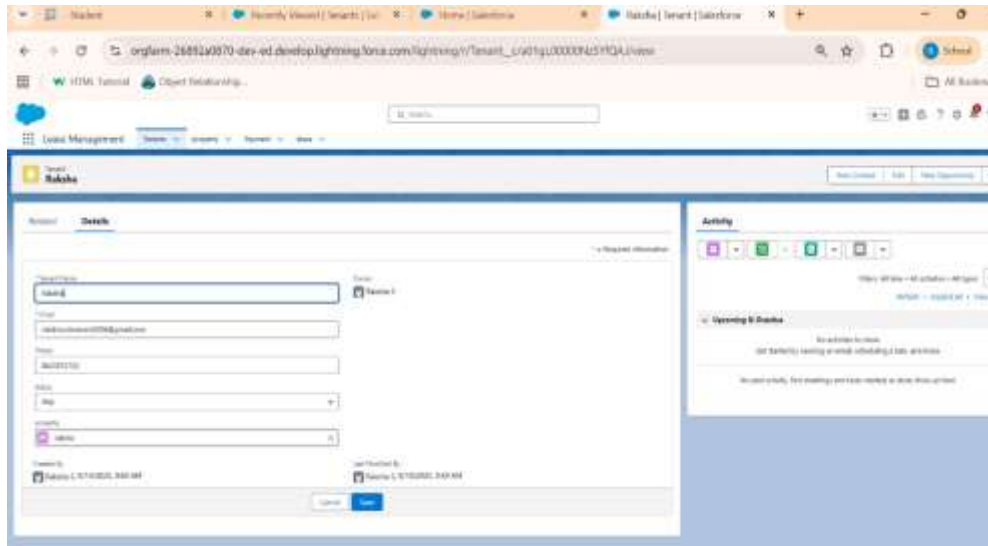
```



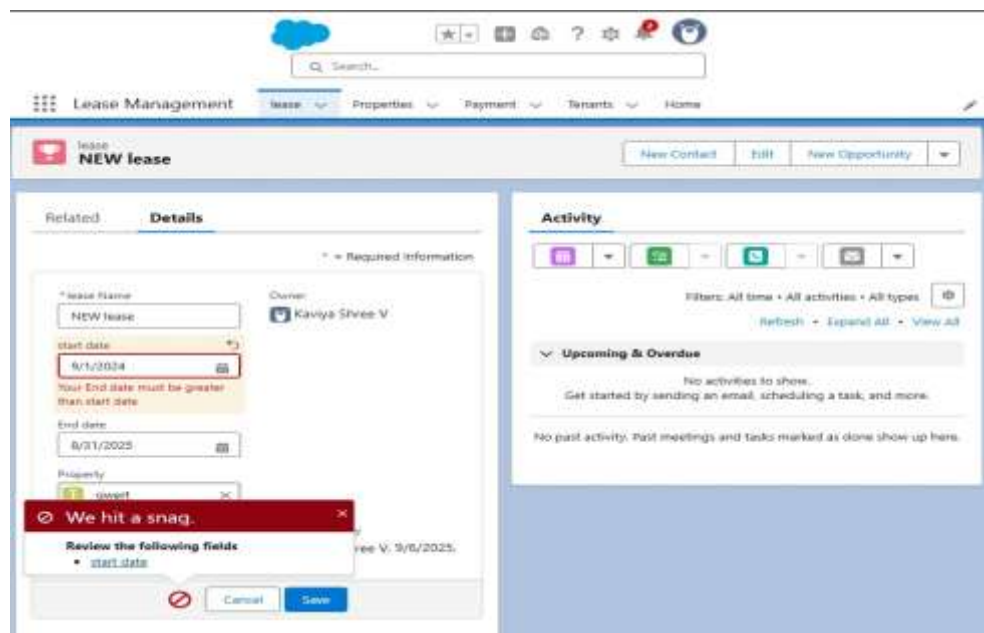
# FUNCTIONAL AND PERFORMANCE TESTING

## Performance Testing

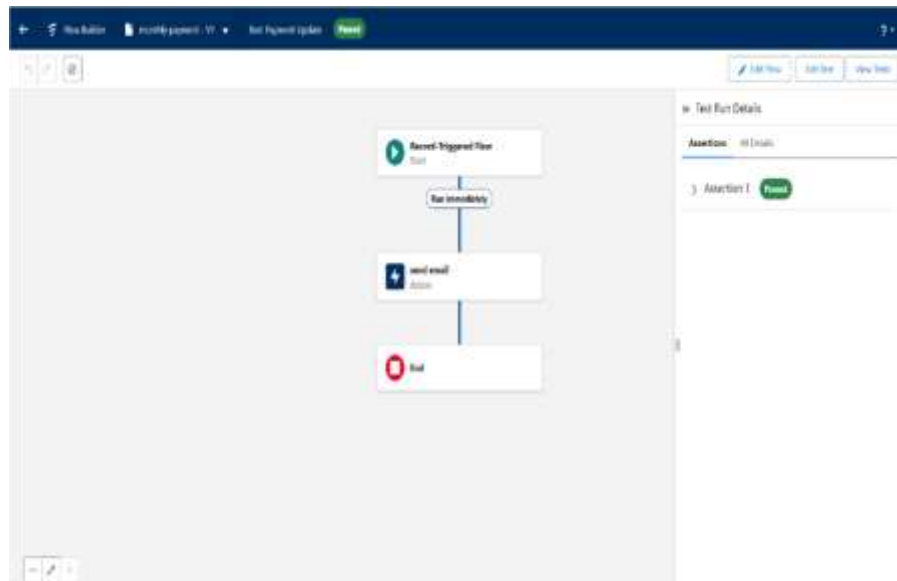
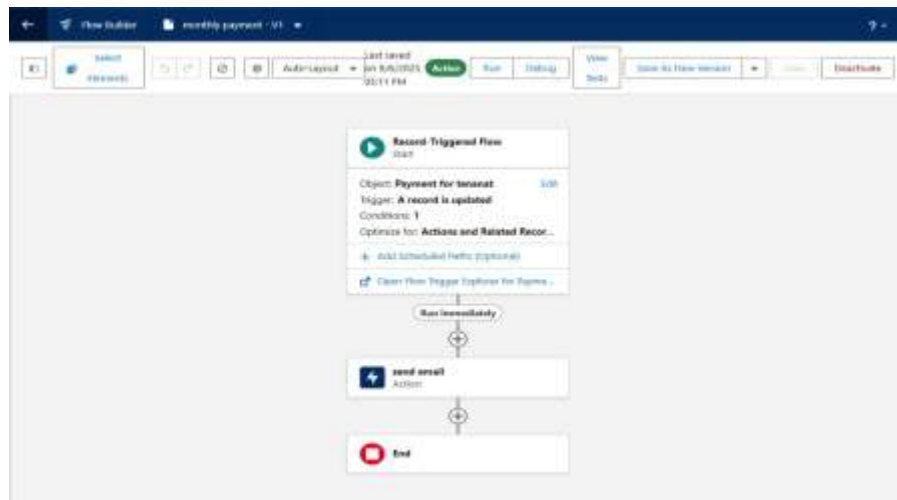
- Trigger validation by entering duplicate tenant-property records



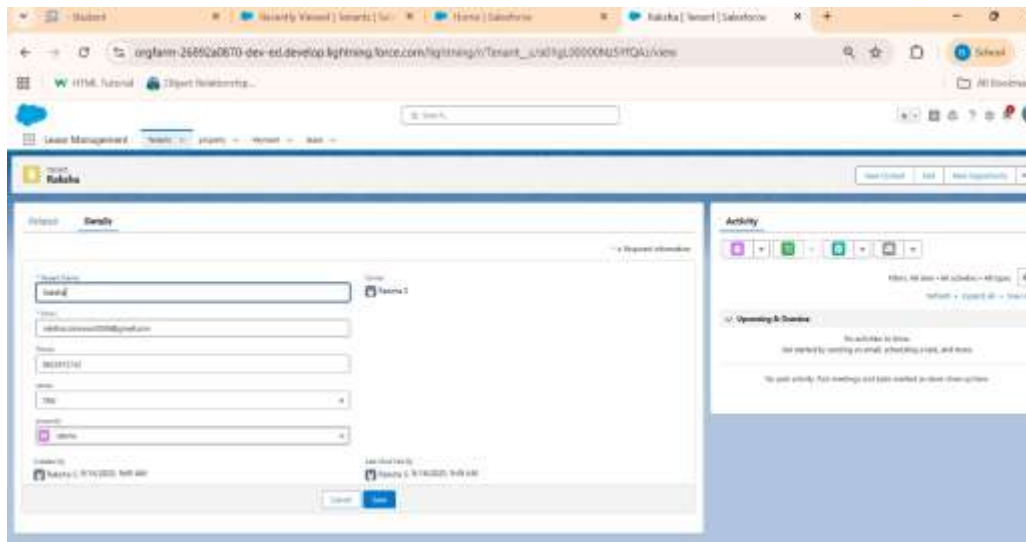
- Validation Rule checking



➤ Test flows on payment update



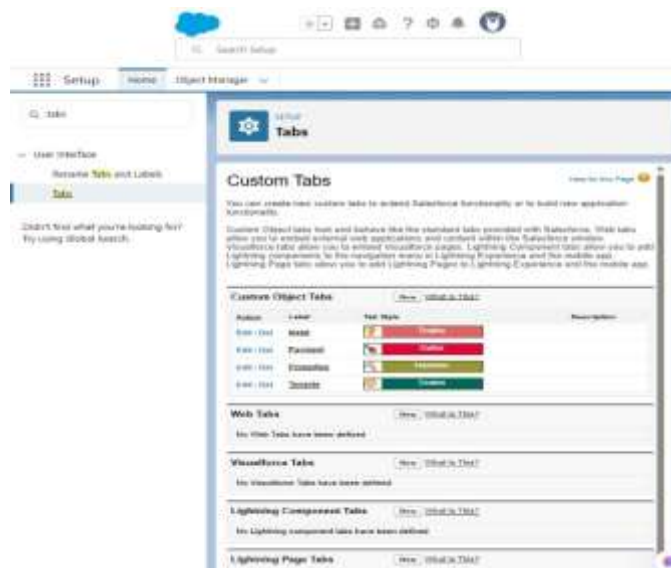
- Approval process validated through email alerts and status updates



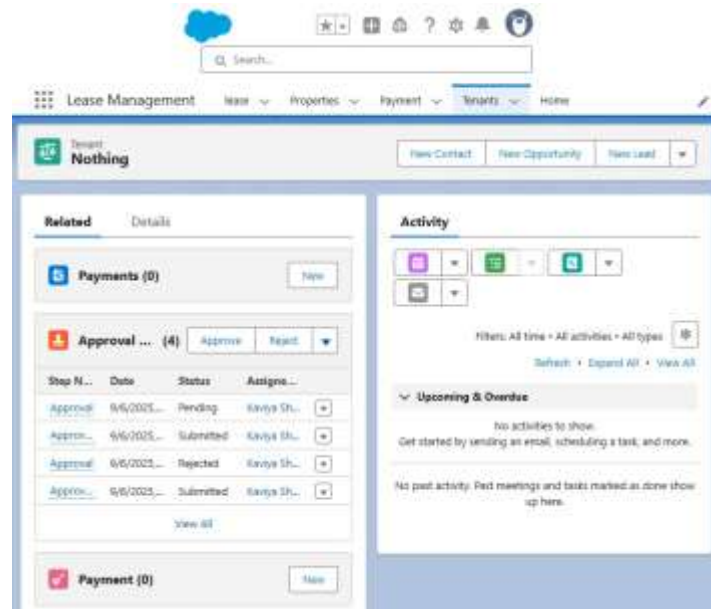
## RESULTS

### Output Screenshots

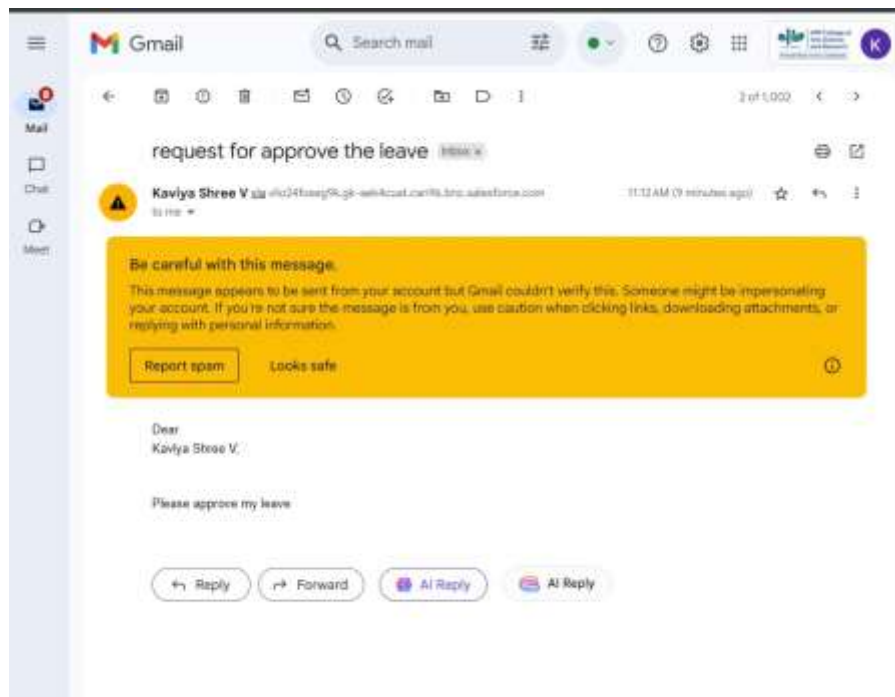
- Tabs for Property, Tenant, Lease, Payment



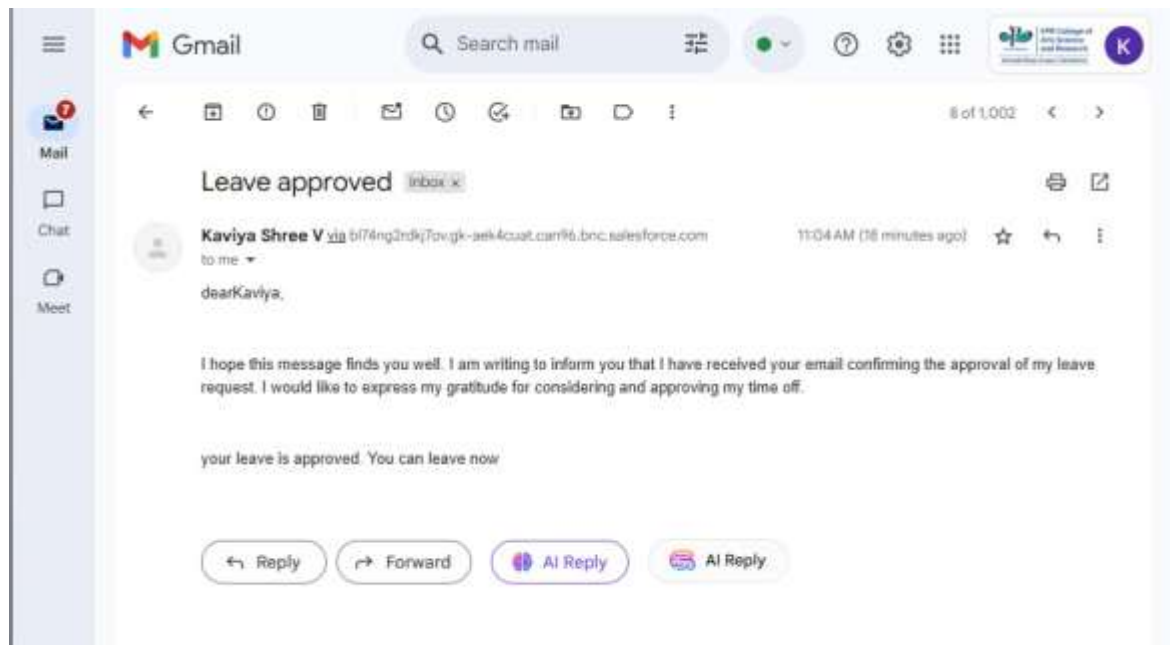
➤ Email alerts



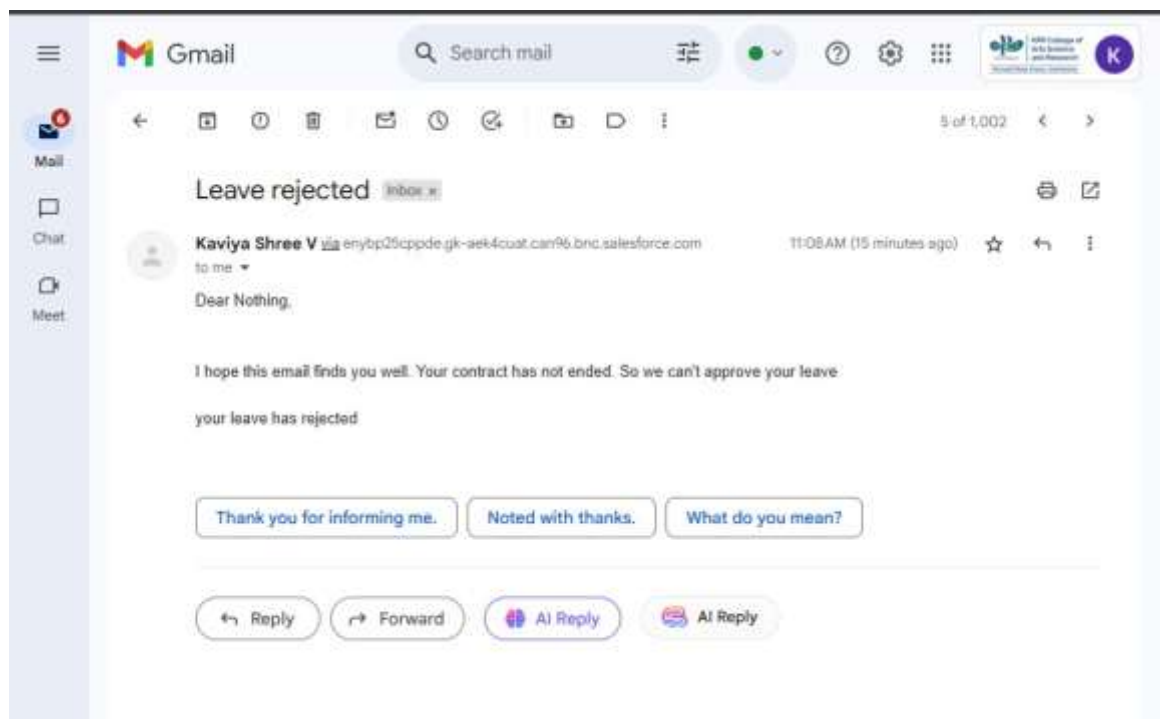
➤ Request for approve the leave



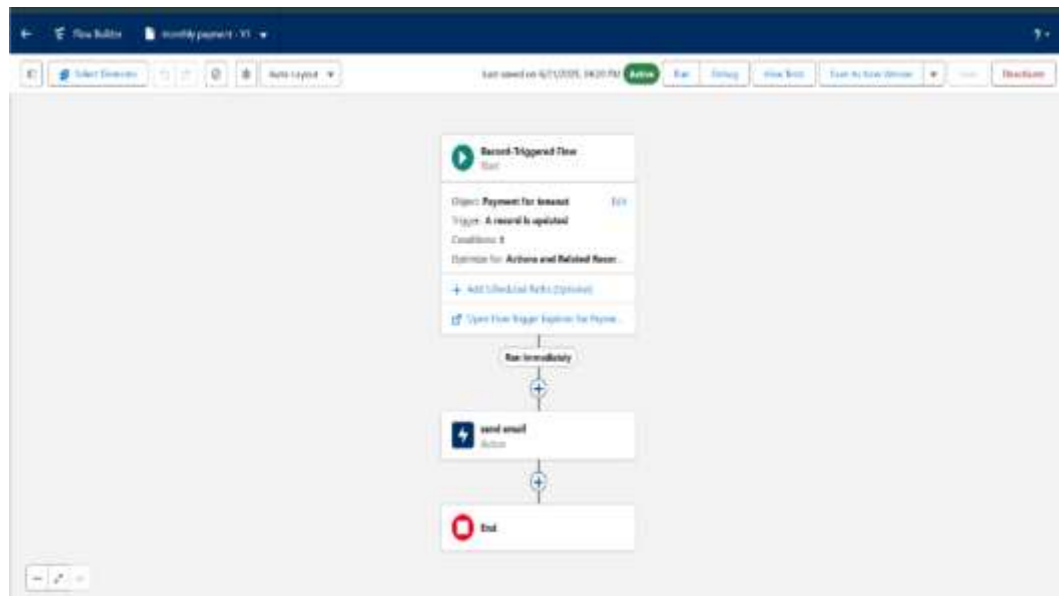
➤ Leave approved



➤ Leave rejected



➤ Flow runs



➤ Trigger error messages

The screenshot shows a web application interface for 'Lease Management'. The main section is titled 'NEW lease' and contains a form with fields for 'lease Name', 'start date', 'End date', and 'Property'. The 'start date' field is highlighted with a red border and a message: 'Your End date must be greater than start date'. A red error message box is displayed at the bottom of the form, stating 'We hit a snag. Review the following fields: start date'. The 'Activity' section on the right shows a list of activities with filters and a 'Refresh' button. The bottom of the page has a 'Cancel' button and a 'Save' button.

## 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))
    { newTenant.addError('A tenant can have only one property');
    }
}
}
}
}
}

```

**MonthlyEmailScheduler.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});
}
}
```