

LEASE MANAGEMENT

College Name: Chikkanna Goverment Arts College

College Code: bru06

TEAM ID:

TEAM MEMBERS:

Team Leader Name: MANOJKUMAR R

Email : 701kingdark@gmail.com

Team Member 1: RAGAVI D

Email : saritharagavi6529@gmail.com

Team Member: BALAJI G

Email: balajibalaji75582005@gmail.com

Team Member: DHANUSH K

Email: dhanushdhanush2644@gmail.com

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.



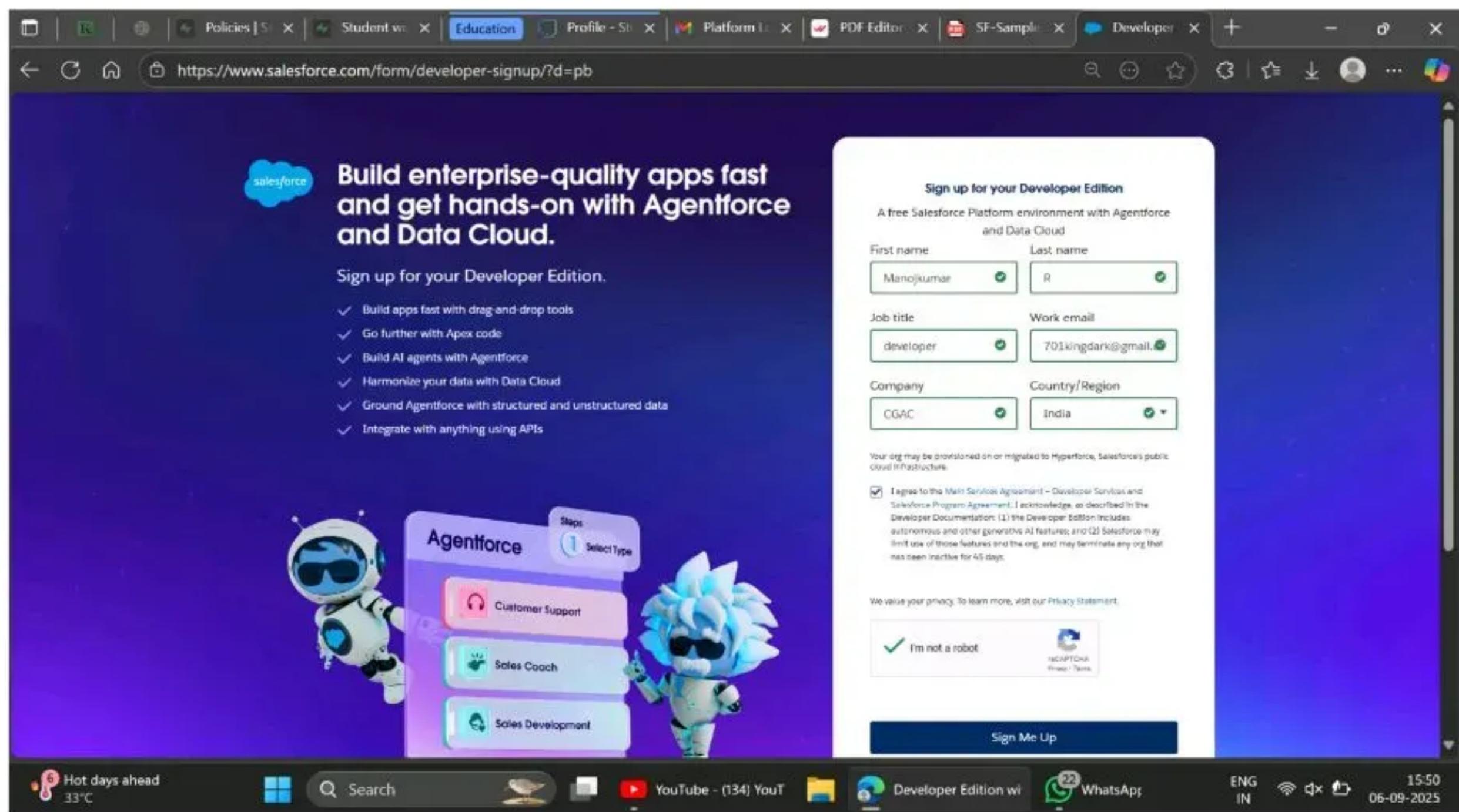
1.2 Purpose

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>



- Created objects: Property, Tenant, Lease, Payment

The screenshot shows the Salesforce Object Manager for the "property" object. The left sidebar lists various setup options: Fields & Relationships, 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, Restriction Rules, and Scoping Rules. The main area displays the "Details" tab for the "property" object. It shows the API Name as "property__c", the Singular Label as "property", and the Plural Label as "property". On the right, there are sections for "Enable Reports" (checked), "Track Activities" (checked), "Track Field History" (checked), "Deployment Status" (Deployed), and "Help Settings" (Standard salesforce.com Help Window). At the top right of the main area are "Edit" and "Delete" buttons.

The screenshot shows the Salesforce Setup interface for the 'Object Manager' section. The left sidebar lists various configuration options: Fields & Relationships, 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, Restriction Rules, and Scoping Rules. The main panel displays the 'Details' tab for the 'Tenant' object. The API Name is set to 'Tenants__c'. The Singular Label is 'Tenant' and the Plural Label is 'Tenants'. Under the 'Enable Reports' section, 'Track Activities' and 'Track Field History' are checked. Deployment Status is set to 'Deployed' and Help Settings point to the 'Standard salesforce.com Help Window'. The top right of the panel has 'Edit' and 'Delete' buttons.

The screenshot shows the Salesforce Setup interface for the 'Object Manager' section. The left sidebar lists various configuration options: Fields & Relationships, 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, Restriction Rules, and Scoping Rules. The main panel displays the 'Details' tab for the 'lease' object. The API Name is set to 'lease__c'. The Singular Label is 'lease' and the Plural Label is 'lease'. Under the 'Enable Reports' section, 'Track Activities' and 'Track Field History' are checked. Deployment Status is set to 'Deployed' and Help Settings point to the 'Standard salesforce.com Help Window'. The top right of the panel has 'Edit' and 'Delete' buttons.

Object Manager

Payment for tenantat

Details

Description	Enable Reports
API Name Payment_for_tenantat_c	✓
Custom	✓
Singular Label Payment for tenantat	Track Activities
Plural Label Payment	Track Field History
	✓
	Deployment Status Deployed
	Help Settings Standard salesforce.com Help Window

- Configured fields and relationships

Fields & Relationships

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Address	Address_c	Long Text Area(32768)		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Name	Name_c	Text(25)		
Owner	OwnerId	Lookup(User,Group)		✓
property Name	Name	Text(80)		✓
sfqt	sfqt_c	Text(18)		
Type	Type_c	Picklist		

https://orgfarm-bcd5b565ea-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgL000001yIUv/FieldsAnd...

Tenant

SETUP > OBJECT MANAGER

Fields & Relationships

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		
property	property_c	Lookup(property)		✓
Property Owner	Property_Owner_c	Lookup(User)		✓
status	status_c	Picklist		
Tenant Name	Name	Text(80)		✓

https://orgfarm-bcd5b565ea-dev-ed.develop.lightning.force.com/lightning/setup/ObjectManager/01lgL000001yIUv/FieldsAnd...

lease

SETUP > OBJECT MANAGER

Fields & Relationships

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

Setup > Object Manager

Payment for tenantat

Fields & Relationships

FIELD LABEL	FIELD NAME	DATA TYPE	CONTROLLING FIELD	INDEXED
Amount	Amount_c	Number(18, 0)		
check for payment	check_for_payment_c	Picklist		
Created By	CreatedById	Lookup(User)		
Last Modified By	LastModifiedById	Lookup(User)		
Payment date	Payment_date_c	Date		
Payment Name	Name	Text(80)		✓
Tenant	Tenant_c	Master-Detail(Tenant)		✓

Details

- 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
- Restriction Rules
- Scoping Rules

https://orgfarm-bcd5b565ea-dev-ed.develop.lightning.force.com/one/one...

- Developed Lightning App with relevant tabs

Click to go back, hold to see history

Lightning App Builder | App Settings | Pages | Lease Management

App Settings

App Details & Branding

Give your Lightning app a name and description. Upload an image and choose the highlight color for its navigation bar.

App Details

* App Name: Lease Management

* Developer Name: Lease_Management

Description: Enter a description...

App Branding

Image: Primary Color Hex Value: #0070D2

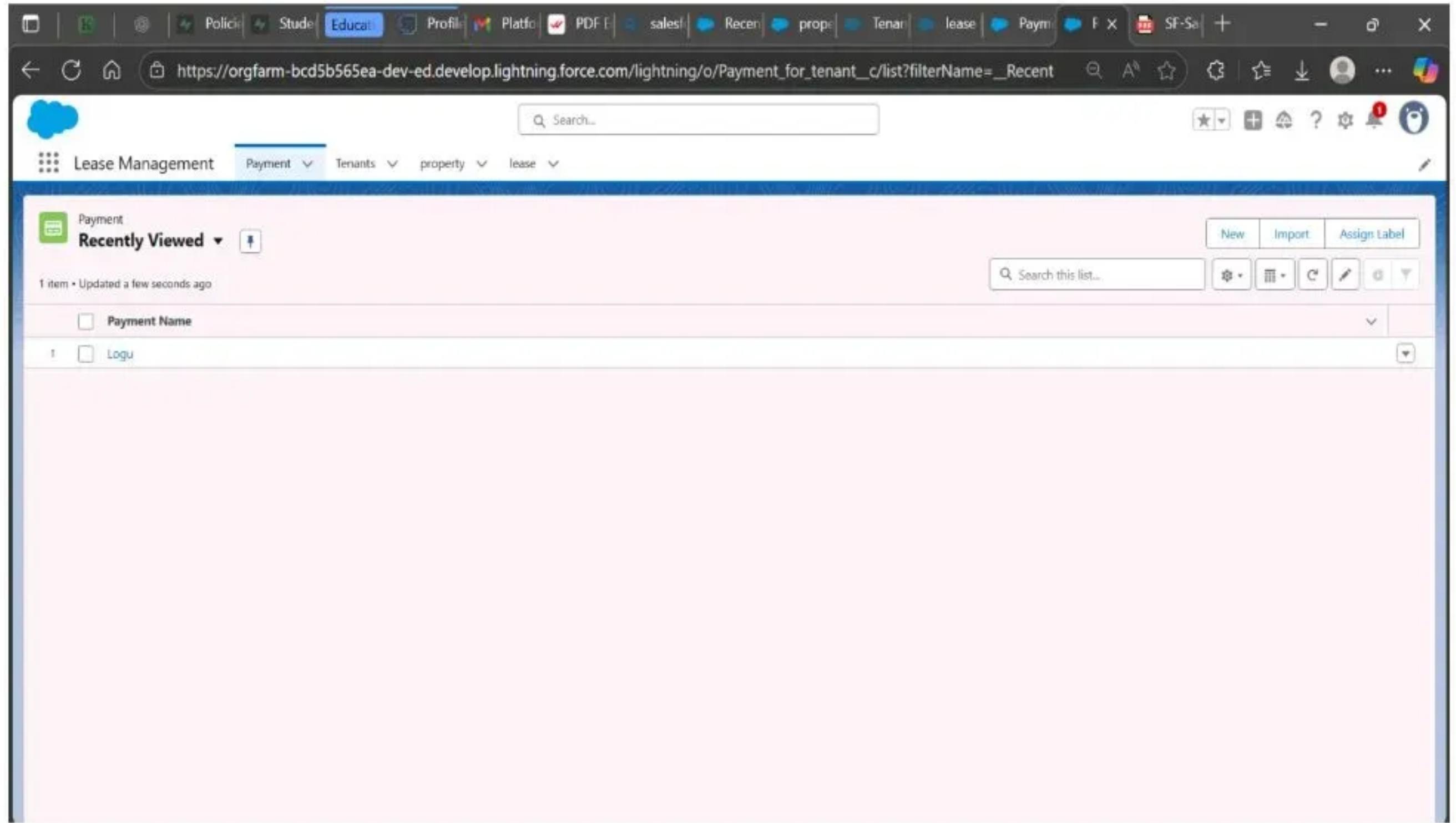
Org Theme Options: Use the app's image and color instead of the org's custom theme

App Launcher Preview

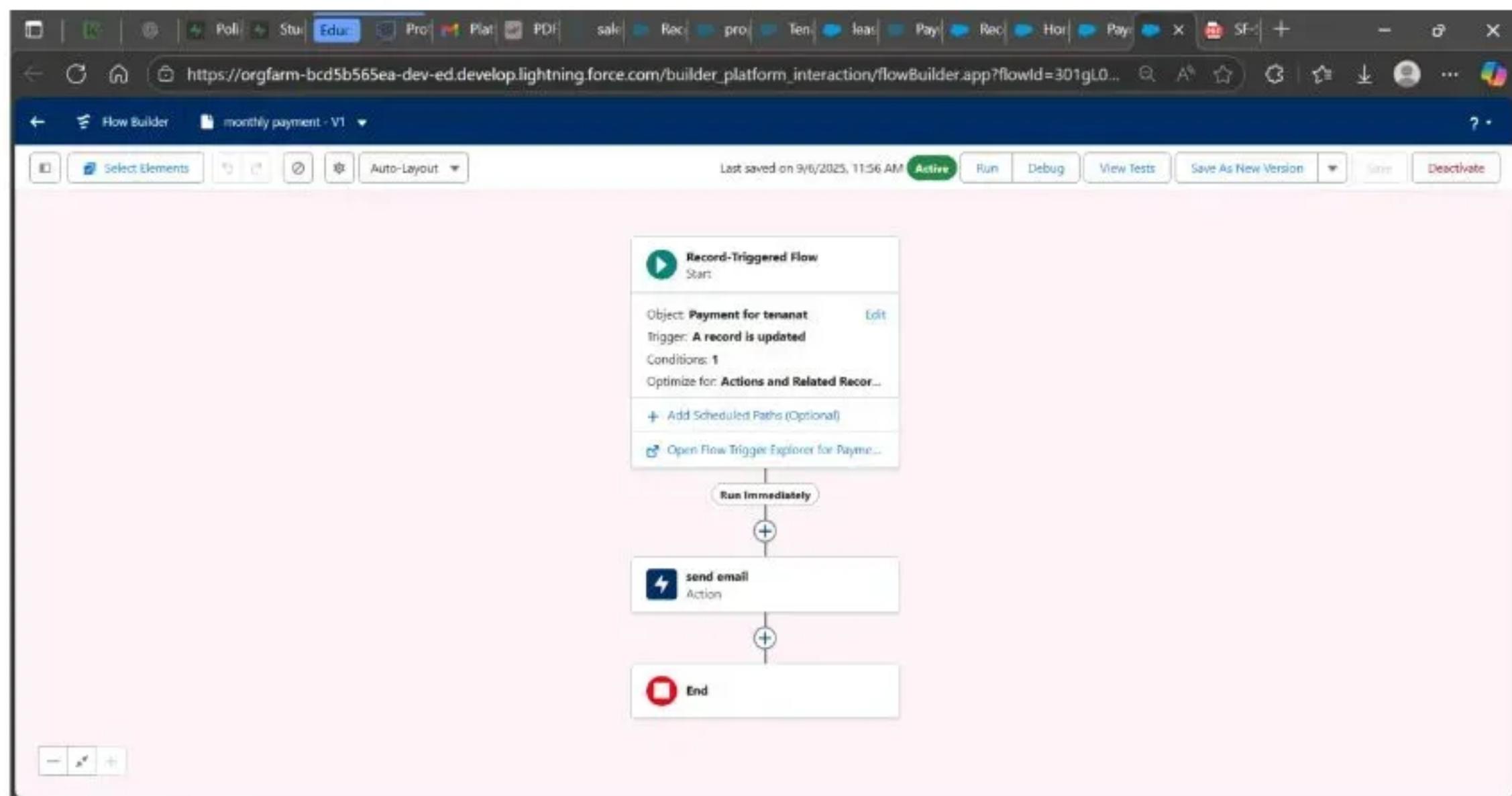
LM Lease Management

The screenshot shows the 'Navigation Items' section of the Lightning App Builder. On the left, a sidebar lists 'App Settings' (selected), 'App Details & Branding', 'App Options', 'Utility Items (Desktop Only)', and 'Navigation Items' (selected). The main area is titled 'Navigation Items' with a sub-instruction: 'Choose the items to include in the app, and arrange the order in which they appear. Users can personalize the navigation to add or move items, but users can't remove or rename 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 viewed in a format that the item doesn't support.' Below this is a 'Available Items' list containing various navigation items like Accounts, Activation Targets, Activations, All Sites, Alternative Payment Methods, Analytics, App Launcher, Appointment Categories, Appointment Invitations, Approval Requests, and Approval Submission Details. To the right is a 'Selected Items' list where 'Payment', 'Tenants', 'property', and 'lease' have been moved. Navigation arrows allow items to be moved between the two lists.

The screenshot shows the 'User Profiles' section of the Lightning App Builder. The sidebar on the left shows 'App Settings' (selected), 'App Details & Branding', 'App Options', 'Utility Items (Desktop Only)', and 'User Profiles' (selected). The main area is titled 'User Profiles' with the instruction: 'Choose the user profiles that can access this app.' Below this is a 'Available Profiles' list containing Analytics Cloud Integration User, Analytics Cloud Security User, Anypoint Integration, Authenticated Website, Authenticated Website, B2B Reordering Portal Buyer Profile, Contract Manager, Custom: Marketing Profile, Custom: Sales Profile, Custom: Support Profile, and Customer Community Login User. To the right is a 'Selected Profiles' list where 'System Administrator' has been added. Navigation arrows allow profiles to be moved between the two lists.



- Implemented Flows for monthly rent and payment success



- To create a validation rule to a Lease Object

The screenshot shows the Salesforce Setup interface under the Object Manager section for the 'lease' object. A validation rule named 'lease_end_date' is being created. The formula is set to check if the end date is less than the start date. The formula editor shows the expression `End_date__c < start_date__c`. A dropdown menu for functions lists 'ABS' as the selected function.

Validation Rule Edit

Validation Rule Detail

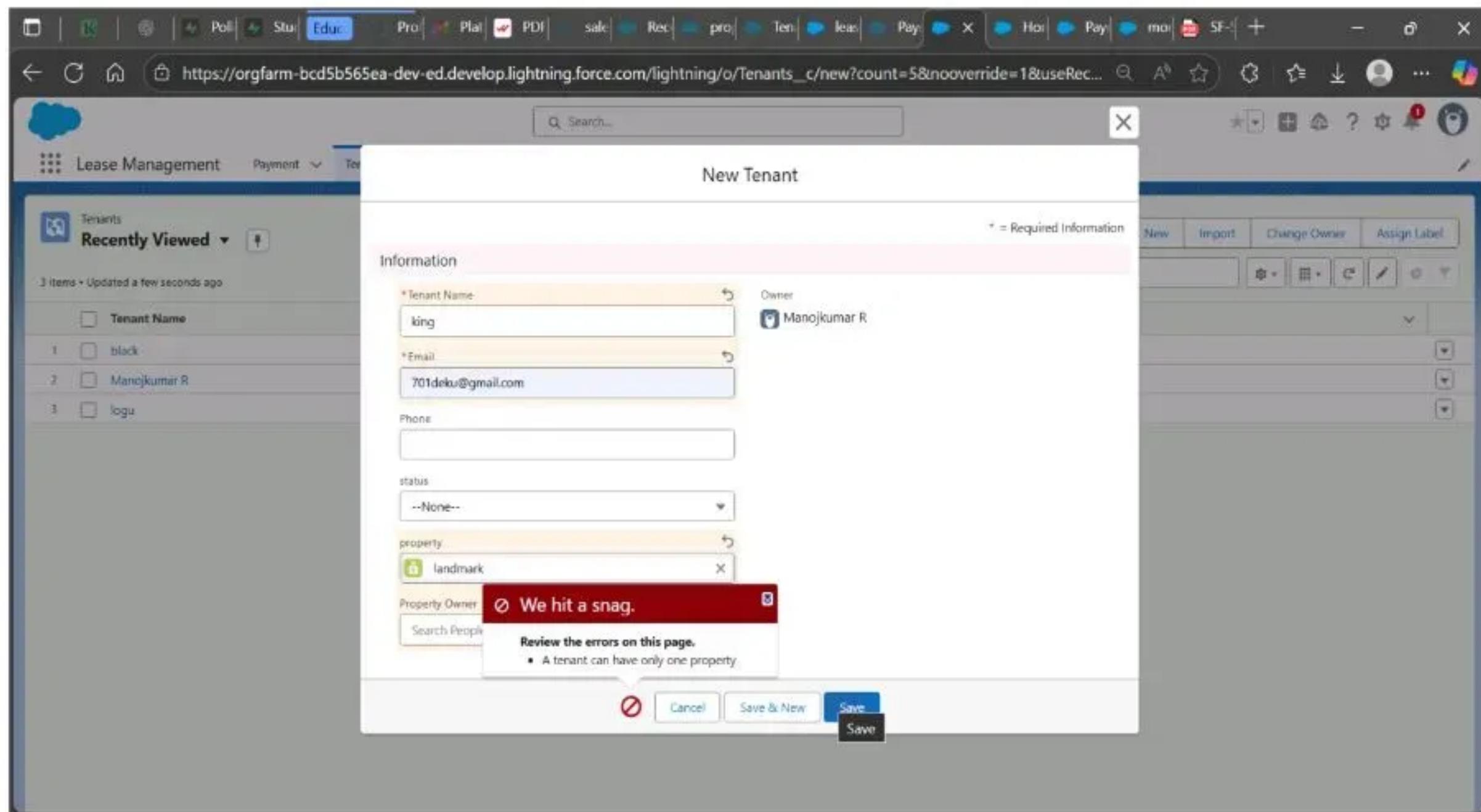
Field	Value
Rule Name	lease_end_date
Active	<input checked="" type="checkbox"/>
Error Condition Formula	<code>End_date__c < start_date__c</code>
Description	(empty)
Created By	Manojkumar R. 9/5/2025, 12:17 AM
Modified By	Manojkumar R. 9/5/2025, 12:38 AM

The screenshot shows the validation rule detail page for the 'lease' object. The rule 'lease_end_date' is listed with its formula and error message. The formula is `End_date__c < start_date__c`, and the error message is 'Your End date must be greater than start date'. The rule is active and was created by Manojkumar R. on 9/5/2025 at 12:17 AM, last modified by the same user on 9/5/2025 at 12:38 AM.

Validation Rule Detail

Field	Value
Rule Name	lease_end_date
Error Condition Formula	<code>End_date__c < start_date__c</code>
Error Message	Your End date must be greater than start date
Description	(empty)
Created By	Manojkumar R. 9/5/2025, 12:17 AM
Modified By	Manojkumar R. 9/5/2025, 12:38 AM

- Added Apex trigger to restrict multiple tenants per property

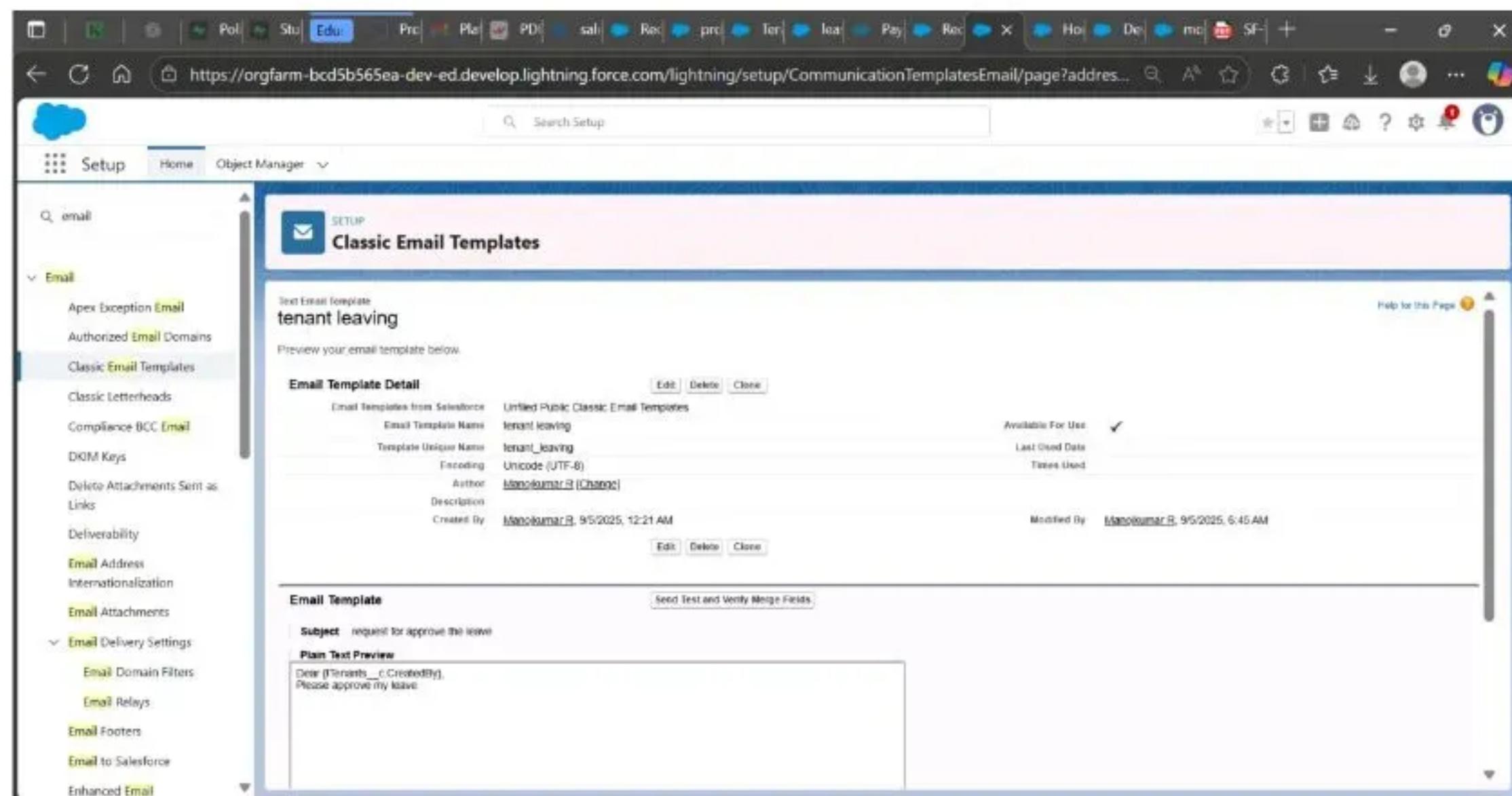
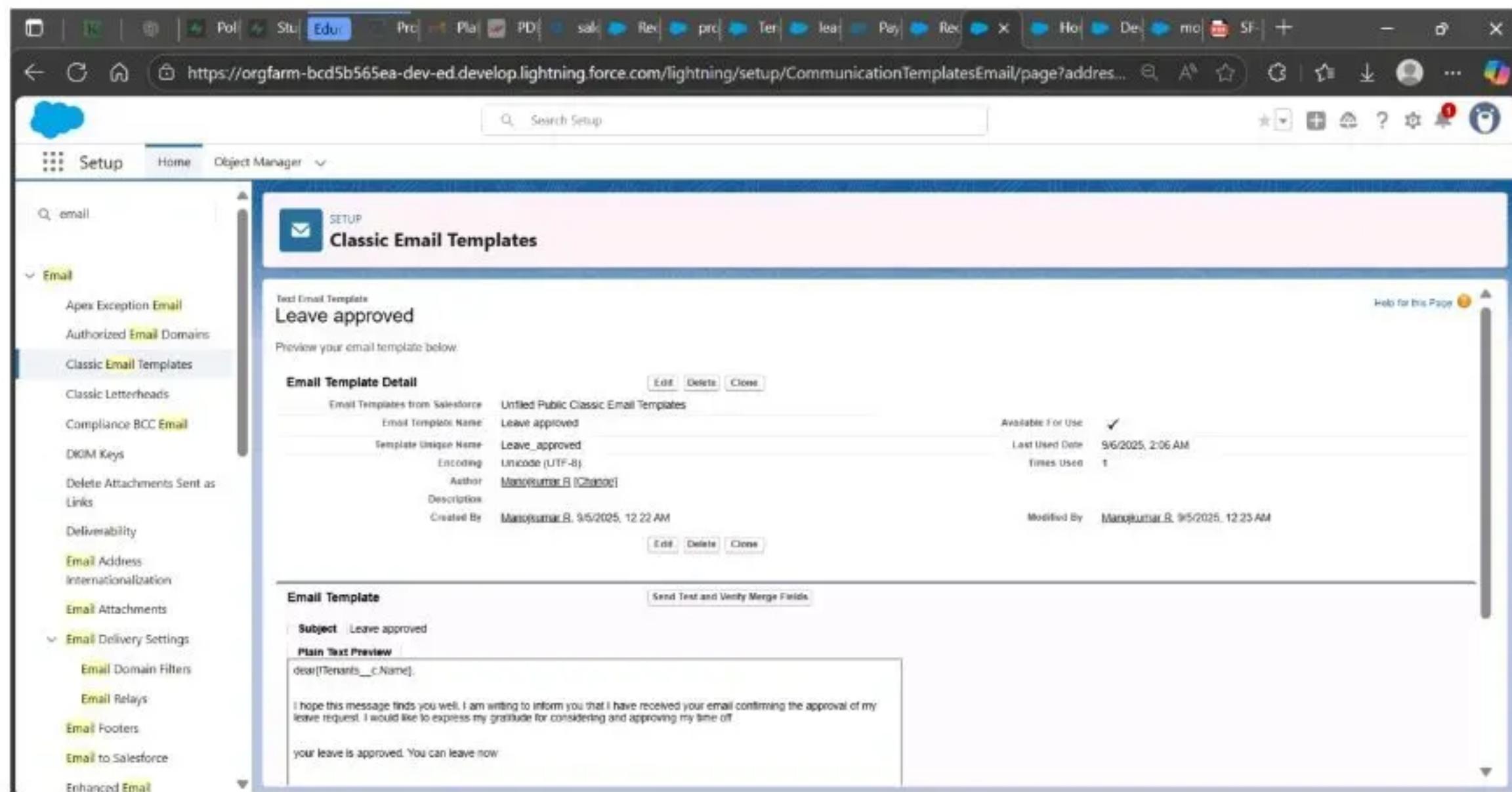


- Scheduled monthly reminder emails using Apex class

```
1 global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8     public static void sendMonthlyEmails() {
9         List<Tenants__c> tenants = [SELECT Id, Email__c FROM Tenants__c];
10        for (Tenants__c tenant : tenants) {
11            String recipientEmail = tenant.Email__c;
12            String emailContent = 'I trust this email finds you well. I am writing to remind you that the monthly rent is due Your time';
13            String emailSubject = 'Reminder: Monthly Rent Payment Due';
14            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
15            email.setToAddresses(new String[]{recipientEmail});
16            email.setSubject(emailSubject);
17            email.setPlainTextBody(emailContent);
18            Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
19        }
20    }
21 }
```

The screenshot shows the Salesforce developer console with the URL https://orgfarm-bcd5b565ea-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The code editor displays the Apex class "MonthlyEmailScheduler.apxc". The class implements the "Schedulable" interface and contains a scheduled method "execute" that checks if the current day is 1. If so, it calls the static method "sendMonthlyEmails". This method queries all "Tenants__c" records and loops through them to create a single email message for each, setting the recipient to the "Email__c" field, the subject to "Reminder: Monthly Rent Payment Due", and the body to a placeholder message. Finally, it uses the "Messaging.sendEmail" method to send the emails.

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



The screenshot shows the Salesforce Setup interface with the URL https://orgfarm-bcd5b565ea-dev-ed.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=tenant_leaving. The left sidebar is expanded, showing various email-related settings under the 'Email' category. The main content area displays the 'Classic Email Templates' page for a template named 'tenant leaving'. The 'Email Template Detail' section shows the following details:

Email Templates from Salesforce	Unified Public Classic Email Templates
Email Template Name	tenant leaving
Template Unique Name	tenant_leaving
Encoding	Unicode (UTF-8)
Author	Manojkumar.B [Change]
Description	
Created By	Manojkumar.B, 9/5/2025, 12:21 AM
Modified By	Manojkumar.B, 9/5/2025, 6:40 AM

The 'Email Template' section shows the subject: "request for approve the leave". The 'Main Text Preview' shows the following text:

Dear {[Tenants__c.CreatedBy]},
Please approve my leave

The screenshot shows the Salesforce Setup interface with the URL https://orgfarm-bcd5b565ea-dev-ed.lightning.force.com/lightning/setup/CommunicationTemplatesEmail/page?address=Tenant_Email. The left sidebar is expanded, showing various email-related settings under the 'Email' category. The main content area displays the 'Classic Email Templates' page for a template named 'Tenant Email'. The 'Email Template Detail' section shows the following details:

Email Templates from Salesforce	Unified Public Classic Email Templates
Email Template Name	Tenant Email
Template Unique Name	Urgent_Monthly_Rent_Payment_Reminder
Encoding	Unicode (UTF-8)
Author	Manojkumar.B [Change]
Description	
Created By	Manojkumar.B, 9/5/2025, 12:27 AM
Modified By	Manojkumar.B, 9/5/2025, 12:27 AM

The 'Email Template' section shows the subject: "Urgent Monthly Rent Payment Reminder". The 'Main Text Preview' shows the following text:

Dear {[Tenants__c.Name]},
I trust this email finds you well. We appreciate your continued tenancy at our property and I hope you have been comfortable in your residence.
This communication is a friendly reminder regarding your monthly rent payment, which is currently outstanding. As outlined in our rental agreement, the payment is due. To ensure the smooth operation of our property management and to avoid any

Email Template Detail

- Email Template Name: tenant payment
- Template Unique Name: tenant_payment
- Encoding: Unicode (UTF-8)
- Author: Manokumar_B [Change]
- Description: Created By: Manokumar_B, 9/5/2025, 12:28 AM
- Modified By: Manokumar_B, 9/5/2025, 12:28 AM

Email Template

Plain Text Preview:

Subject: Confirmation of Successful Monthly Payment

Dear {Tenants__c.Email__c},

We hope this email finds you well. We are writing to inform you that we have successfully received your monthly payment. Thank you for your prompt and diligent payment.

- Approval Process creation

For Tenant Leaving:

Process Definition Detail

- Process Name: TenantApproval
- Unique Name: TenantApproval
- Description: Tenant: status EQUALS Stay
- Entry Criteria: Record Editability: Administrator ONLY
- Approval Assignment Email Template: Tenant Owner
- Initial Submitter: Created By: Manokumar_B, 9/6/2025, 1:00 AM
- Modified By: Manokumar_B, 9/6/2025, 2:20 AM

Initial Submission Actions

Action Type	Description
Record Lock	Lock the record from being edited

Approval Steps

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions	Edit	1	step 1		User: Manokumar_B	Final Rejection

For Check for Vacant:

The screenshot shows the Salesforce Setup interface with the URL <https://orgfarm-bcd5b565ea-dev-ed.develop.lightning.force.com/lightning/setup/ApprovalProcesses/page?address=%2F04agL...>. The page title is "Approval Processes". The process detail shows:

- Process Name:** check for vacant
- Unique Name:** check_for_vacant
- Description:** Tenant: status EQUALS Leaving
- Entry Criteria:** Tenant: status EQUALS Leaving
- Record Editability:** Administrator ONLY
- Next Automated Approver Determined By:** None
- Approval Assignment Email Template:** tenant_leaving
- Initial Submitters:** Tenant Owner
- Created By:** Manokumar R, 9/6/2025, 1:18 AM
- Modified By:** Manokumar R, 9/6/2025, 2:21 AM

Initial Submission Actions:

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

Approval Steps:

Action	Step Number	Name	Description	Criteria	Assigned Approver	Reject Behavior
Show Actions	Edit	1	step 1		User: Manokumar R	Final Rejection

● Apex Trigger

Create an Apex Trigger

The screenshot shows the Salesforce Developer Console with the URL https://orgfarm-bcd5b565ea-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The trigger code is:

```

1 trigger test on Tenants__c (before insert)
2
3 {
4     if(trigger.isInsert) {
5         testHandler.prev
6     }
7 }

```

A modal window titled "Open" is displayed, listing entities such as Classes, Triggers, Pages, etc. The "Triggers" option is selected.

The screenshot shows the Salesforce Apex code editor with the URL https://orgfarm-bcd5b565ea-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The code editor displays the following Apex trigger:

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

Create an Apex Handler class

The screenshot shows the Salesforce Apex code editor with the URL https://orgfarm-bcd5b565ea-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The code editor displays the following Apex class:

```
public class testHandler {
    public static void preventInsert(List<Tenants__c> newlist) {
        Set<Id> existingIds = new Set<Id>();
        for (Tenants__c newTenant : newlist) {
            existingIds.add(newTenant.getPropertyId());
        }
        for (Tenants__c newTenant : newlist) {
            if (newTenant.getPropertyId() != null && existingIds.contains(newTenant.getPropertyId())) {
                newTenant.addError('A tenant can have only one property');
            }
        }
    }
}
```

A modal dialog box titled "Open" is displayed over the code editor, listing entities and their properties. The "Entity Type" column includes "Classes", "Triggers", "Pages", "Page Components", "Objects", "Static Resources", and "Packages". The "Entities" column lists "DoNothingClass", "MonthlyEmailScheduler", and "testHandler". The "Related" column lists "Name", "Extent", and "Direction".

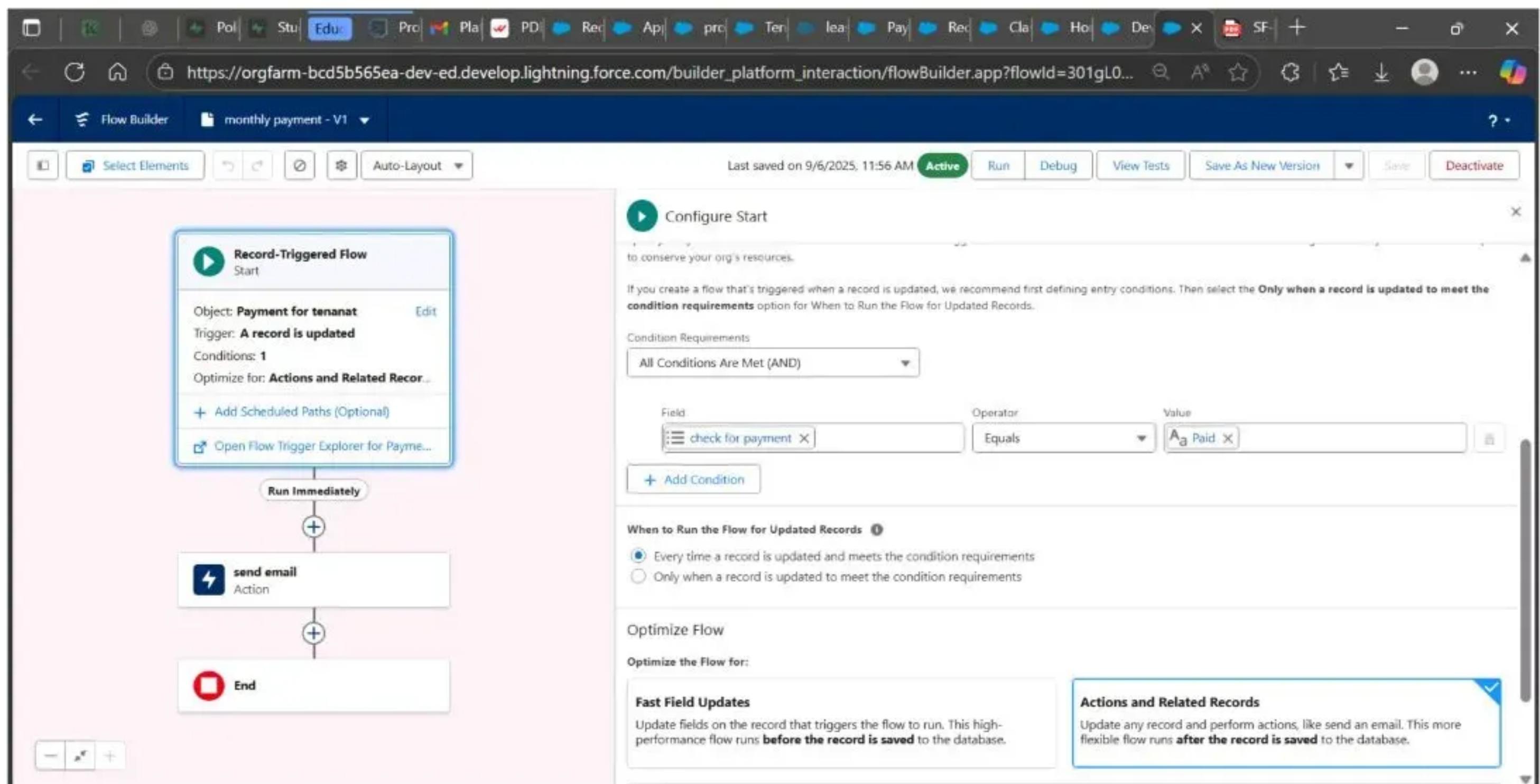
```

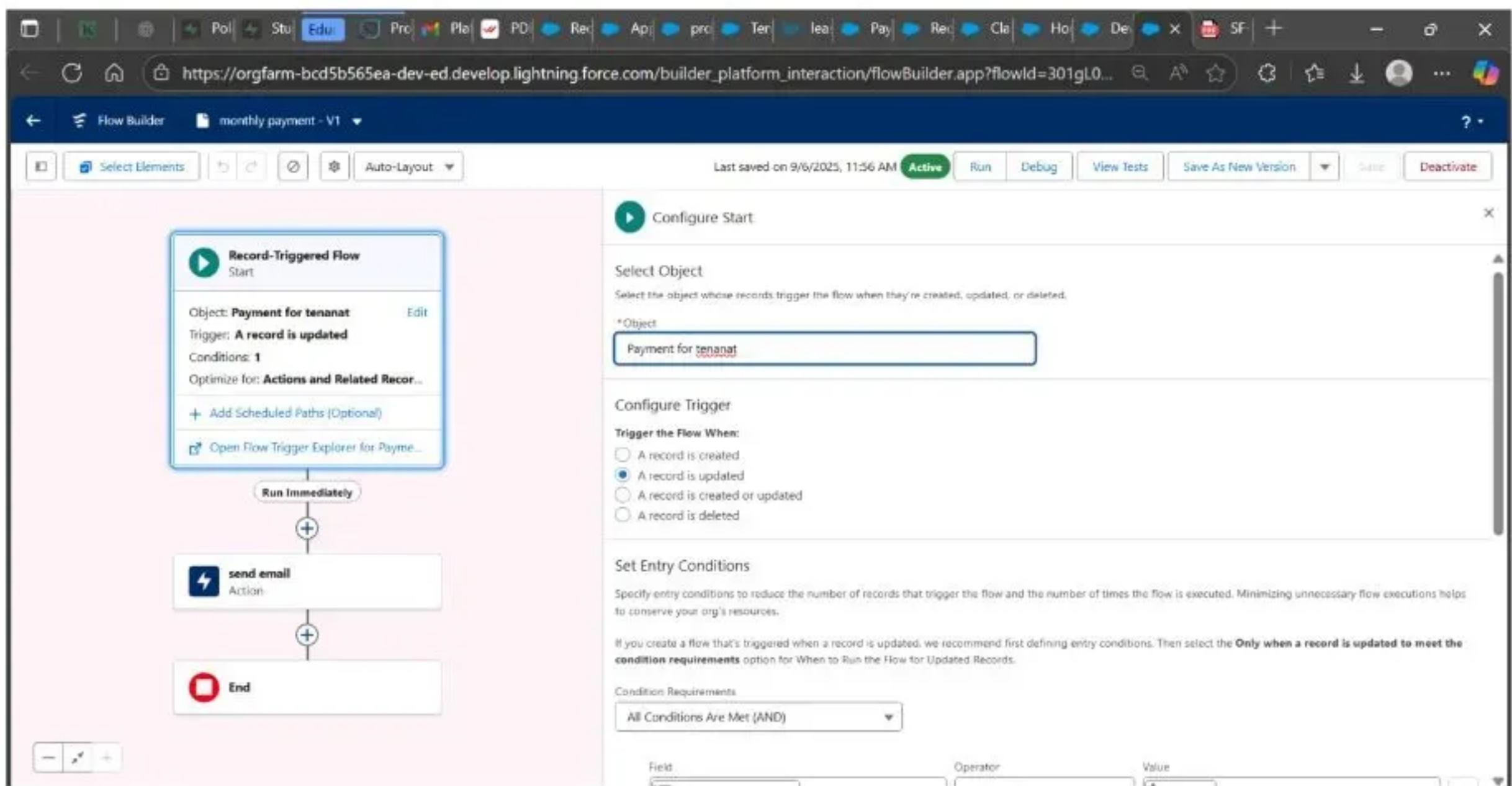
1 public class testHandler {
2
3     public static void preventInsert(List<Tenants__c> newList) {
4
5         Set<Id> existingPropertyIds = new Set<Id>();
6
7         for (Tenants__c existingTenant : [SELECT Id, Property__c FROM Tenants__c WHERE Property__c != null]) {
8
9             existingPropertyIds.add(existingTenant.Property__c);
10
11        }
12
13
14        for (Tenants__c newTenant : newList) {
15
16
17            if (newTenant.Property__c != null && existingPropertyIds.contains(newTenant.Property__c)) {
18
19                newTenantaddError('A tenant can have only one property');
20
21            }
22
23        }
24
25    }
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259

```

Logs, Tests, and Problems ①

● FLOWS





- Schedule class:
Create an Apex Class

```

1 * global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8     public static void sendMonthlyEmails() {
9         List<Tenants__c> tenants = [SELECT Id, Email__c FROM Tenants__c];
10        for (Tenants__c tenant : tenants) {
11            String recipientEmail = tenant.Email__c;
12            String emailContent = 'I trust this email finds you well. I am writing to remind you that the monthly rent is due. Your time';
13            String emailSubject = 'Reminder: Monthly Rent Payment Due';
14            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
15            email.setToAddresses(new String[]{recipientEmail});
16            email.setSubject(emailSubject);
17            email.setPlainTextBody(emailContent);
18            Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
19        }
20    }
21 }

```

The screenshot shows the Salesforce Apex code editor with the URL https://orgfarm-bcd5b565ea-dev-ed.develop.my.salesforce.com/_ui/common/apex/debug/ApexCSIPage. The code editor displays the `MonthlyEmailScheduler.apxc` file. The code implements a scheduled apex class that sends monthly rent reminder emails to tenants.

```
1 * global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8     public static void sendMonthlyEmails() {
9         List<Tenants__c> tenants = [SELECT Id, Email__c FROM Tenants__c];
10        for (Tenants__c tenant : tenants) {
11            String recipientEmail = tenant.Email__c;
12            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.';
13            String emailSubject = 'Reminder: Monthly Rent Payment Due';
14            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
15            email.setToAddresses(new String[]{recipientEmail});
16            email.setSubject(emailSubject);
17            email.setPlainTextBody(emailContent);
18            Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
19        }
20    }
21 }
```

Schedule Apex class

The screenshot shows the Salesforce Lightning setup page with the URL <https://orgfarm-bcd5b565ea-dev-ed.develop.lightning.force.com/lightning/setup/ApexClasses/page?address=%2F01pgL00000...>. The page displays the `MonthlyEmailScheduler` Apex class under the `Apex Classes` section. The class is active and was created by Manojkumar R on 9/5/2025 at 10:35 AM.

Apex Class Detail

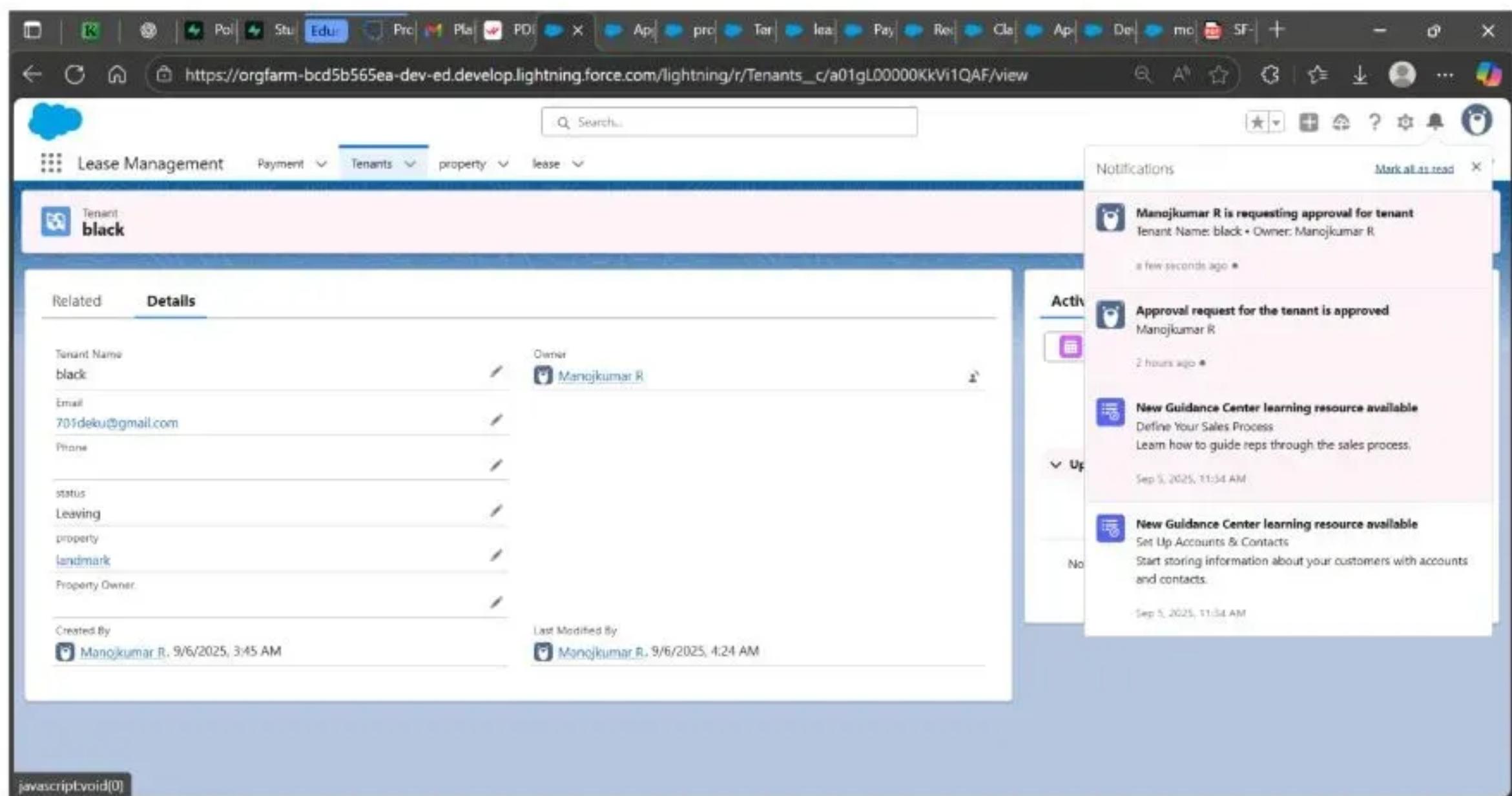
Name	Namespace Prefix	Status	Last Modified By
MonthlyEmailScheduler		Active	Manojkumar R
		Code Coverage: 0% (0/14)	9/5/2025, 10:42 AM

Class Body

```
1 global class MonthlyEmailScheduler implements Schedulable {
2     global void execute(SchedulableContext sc) {
3         Integer currentDay = Date.today().day();
4         if (currentDay == 1) {
5             sendMonthlyEmails();
6         }
7     }
8
9     public static void sendMonthlyEmails() {
10        List<Tenants__c> tenants = [SELECT Id, Email__c FROM Tenants__c];
11        for (Tenants__c tenant : tenants) {
12            String recipientEmail = tenant.Email__c;
13            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.';
14            String emailSubject = 'Reminder: Monthly Rent Payment Due';
15            Messaging.SingleEmailMessage email = new Messaging.SingleEmailMessage();
16            email.setToAddresses(new String[]{recipientEmail});
17            email.setSubject(emailSubject);
18            email.setPlainTextBody(emailContent);
19            Messaging.sendEmail(new Messaging.SingleEmailMessage[]{email});
20        }
21    }
22 }
```

This screenshot shows a Salesforce Lightning interface for a tenant record. The top navigation bar includes tabs like 'Lease Management', 'Payment', 'Tenants', 'property', and 'lease'. The main content area displays a 'Details' tab for a tenant named 'black'. The tenant's email is listed as '701deku@gmail.com'. The owner is 'Manojkumar R'. The status is 'Leaving'. The right sidebar contains an 'Activity' section with a button labeled 'Submit for Approval' highlighted in blue.

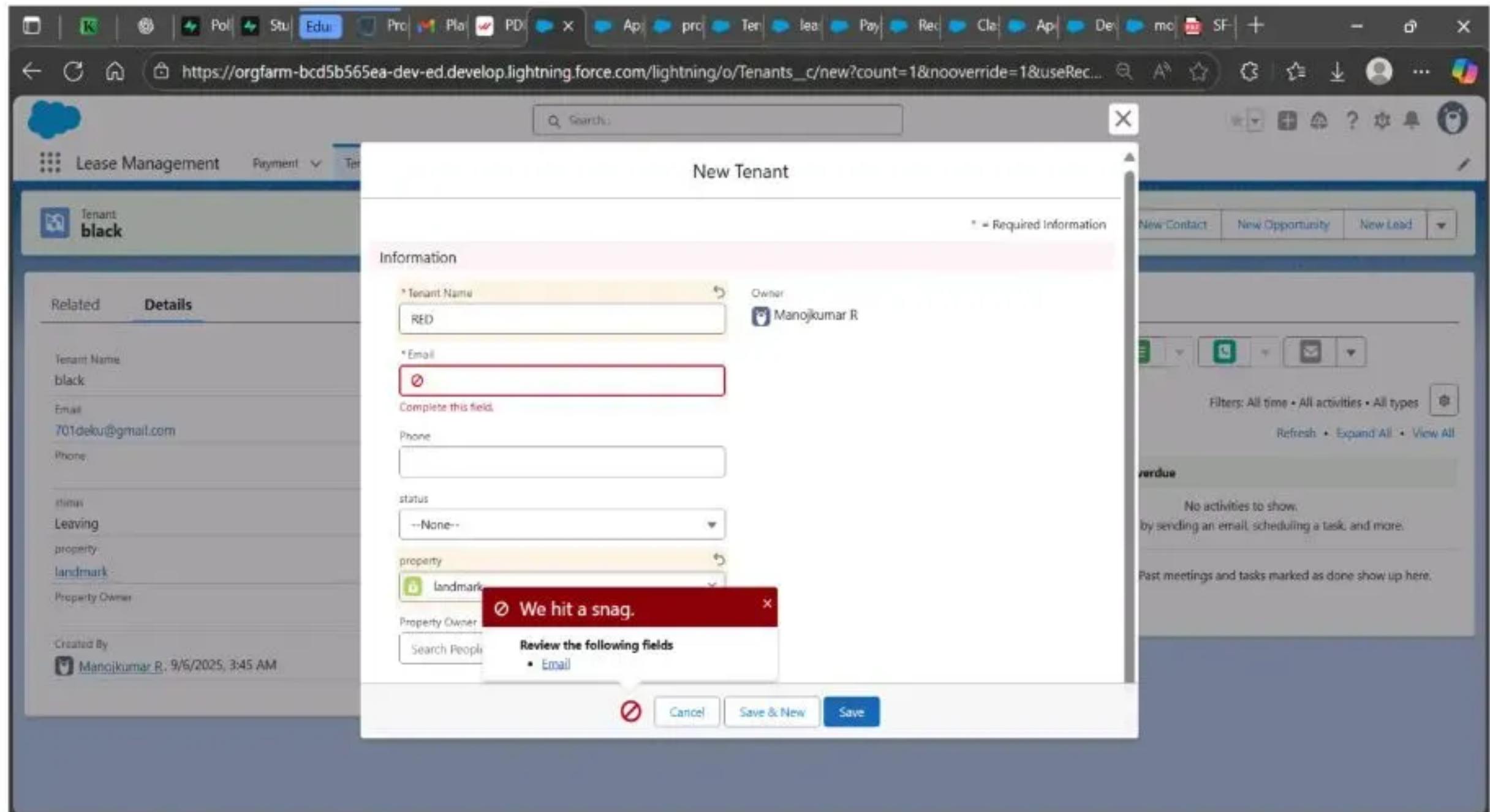
This screenshot shows the same Salesforce Lightning interface after the approval step. A green success message at the top states 'Tenant was submitted for approval.' The rest of the page remains identical to the first screenshot, showing the tenant record details and the activity sidebar.



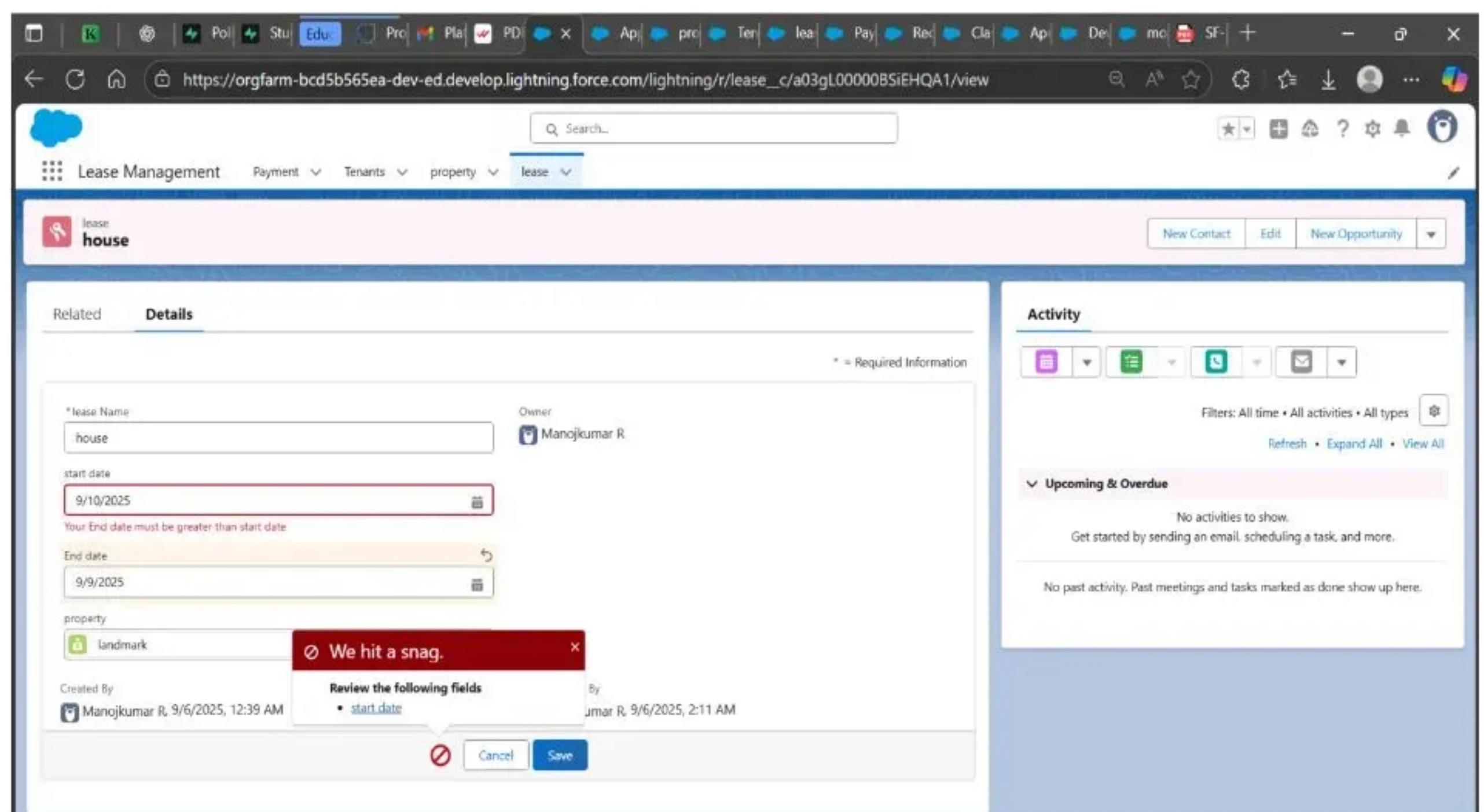
FUNCTIONAL AND PERFORMANCE TESTING

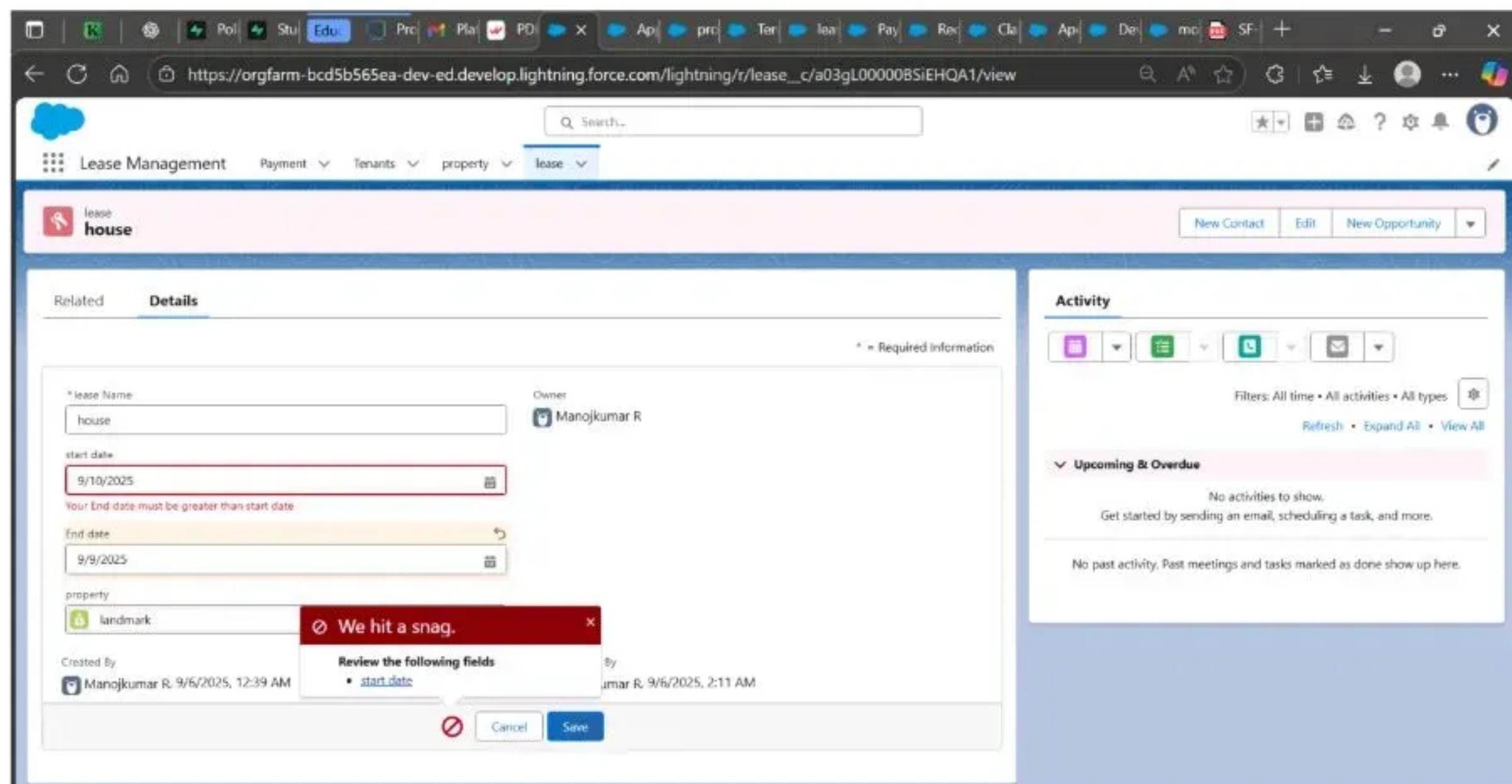
Performance Testing

- Trigger validation by entering duplicate tenant-property records

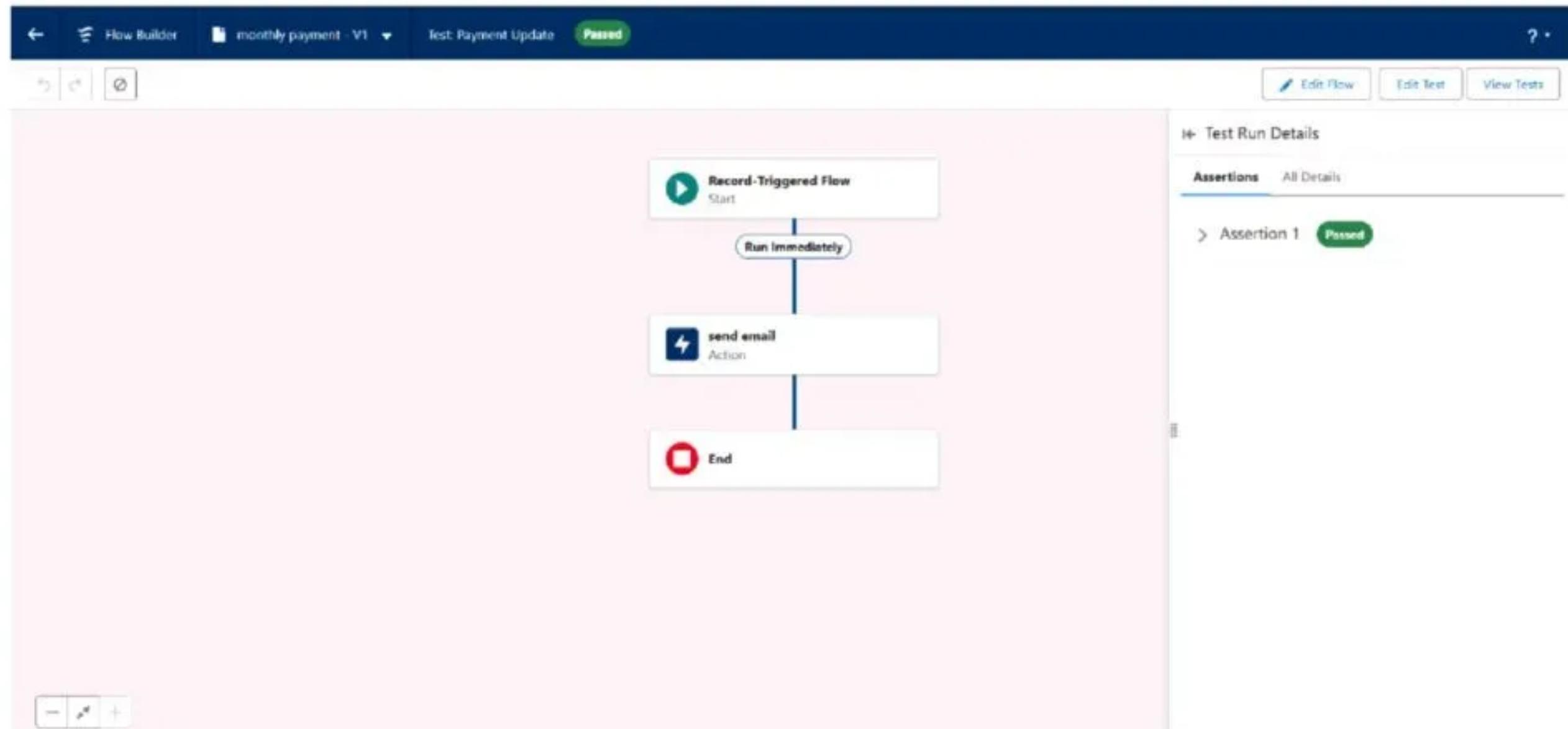


● Validation Rule checking





- Test flows on payment update



- Approval process validated through email alerts and status updates

The screenshot shows a Salesforce Lightning interface for a 'Lease Management' application. The main area displays a 'Tenant' record for 'black'. The 'Details' tab is selected, showing fields such as Tenant Name (black), Email (701deku@gmail.com), Phone, status (Leaving), property (landmark), and Property Owner. It also shows the Created By (Manojkumar R) and Last Modified By (Manojkumar R) information. On the right side, there is a 'Notifications' sidebar with several activity items:

- Manojkumar R is requesting approval for tenant black • Owner: Manojkumar R. (a few seconds ago)
- Approval request for the tenant is approved • Manojkumar R. (2 hours ago)
- New Guidance Center learning resource available • Define Your Sales Process (Sep 5, 2025, 11:34 AM)
- New Guidance Center learning resource available • Set Up Accounts & Contacts (Sep 5, 2025, 11:34 AM)

This screenshot shows the same Salesforce Lightning interface for a 'Lease Management' application, but with a different view. The main area displays a 'Tenant' record for 'black'. The 'Related' tab is selected, showing sections for 'Payments (0)', 'Approval History (2)', and 'Payment (0)'. The 'Approval History' section contains two entries:

Step Name	Date	Status	Assigned To
step 1	9/6/2025, 4:36 AM	Approved	Manojkumar R
Approval Request Submitted	9/6/2025, 4:25 AM	Submitted	Manojkumar R

On the right side, there is an 'Activity' sidebar with a toolbar, filters (All time, All activities, All types), and sections for 'Upcoming & Overdue' and 'Upcoming & Overdue' (both showing no activities).

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');

            }
        }

    }
}

```

MothlyEmailScheduler.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});  
}  
}  
}
```