LEASE MANAGEMENT

Deepan S– 922222104006

Logesh S – 922222104026

Thirumani C – 922222104049

Panick Siranjan C – 922222104035

Project Overview

The **Lease Management System** is a Salesforce-based application designed to simplify and automate property lease operations. The project focuses on managing properties, tenants, leases, and payments efficiently through a centralized system.

This application enables property owners and administrators to track lease agreements, tenant details, and payment records. It ensures smooth communication between landlords and tenants through automated workflows, validation rules, approval processes, and scheduled notifications

Objectives

The main goal of the Lease Management System is to digitalize and automate the lease lifecycle, from property registration to payment collection and lease termination.

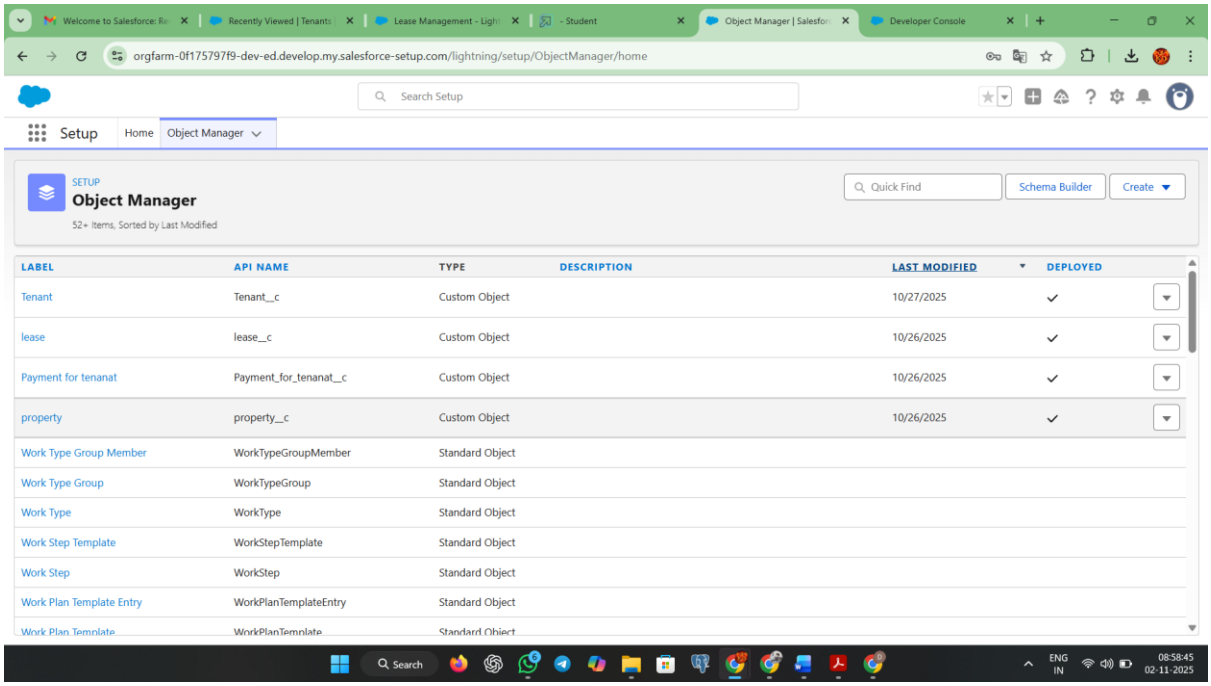
Business Goals:

- **Efficient Property Management:** Maintain detailed records of properties, including type, address, and square footage.
- **Tenant Information Management:** Store tenant contact details and lease status for easy access and updates.
- **Automated Payment Tracking:** Record and monitor monthly payments with automated email alerts for due and successful payments.
- **Streamlined Lease Operations:** Manage lease start and end dates, ensure data validation, and prevent overlapping tenancies.
- **Communication Automation:** Use email templates, flows, and Apex classes to automatically notify tenants regarding payment and lease status.

Salesforce Key Features and Concepts Utilized

The system utilizes several key Salesforce features to provide a robust and automated environment:

Custom Objects:
Created four main objects — *Property, Tenant, Lease,* and *Payment for Tenant* — to store and manage all data related to lease operation.

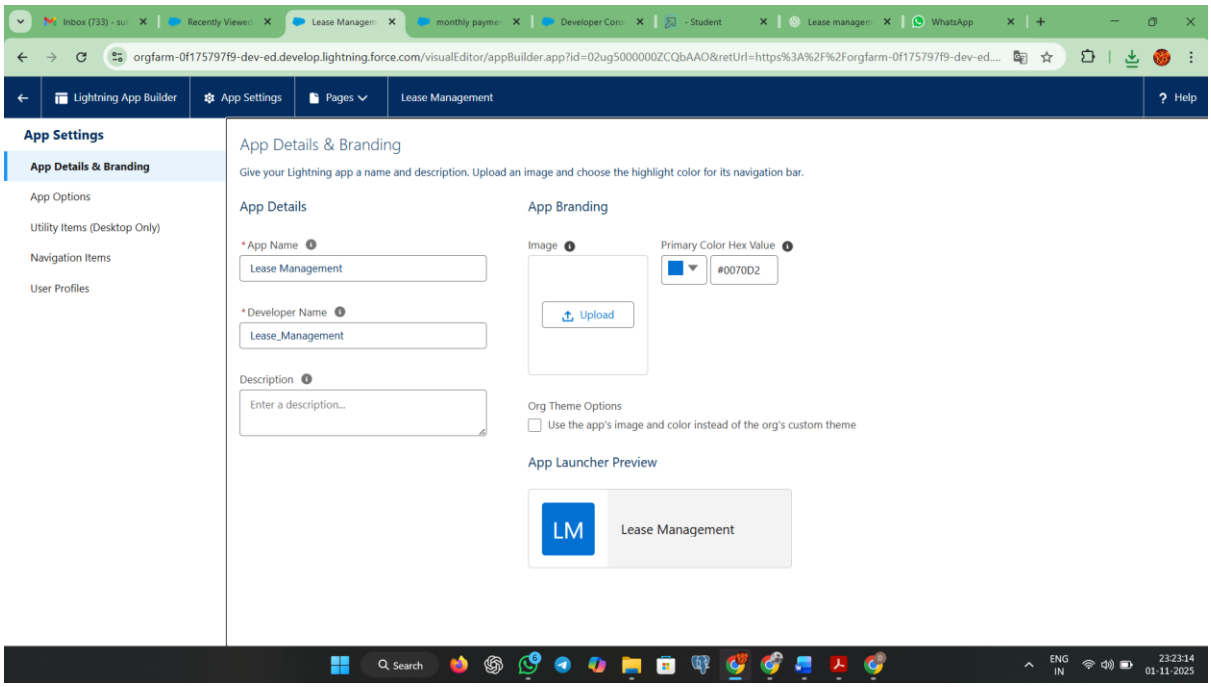


Custom Tabs:

Tabs were created for each object to ensure easy navigation within the Lightning App.

Lightning App:

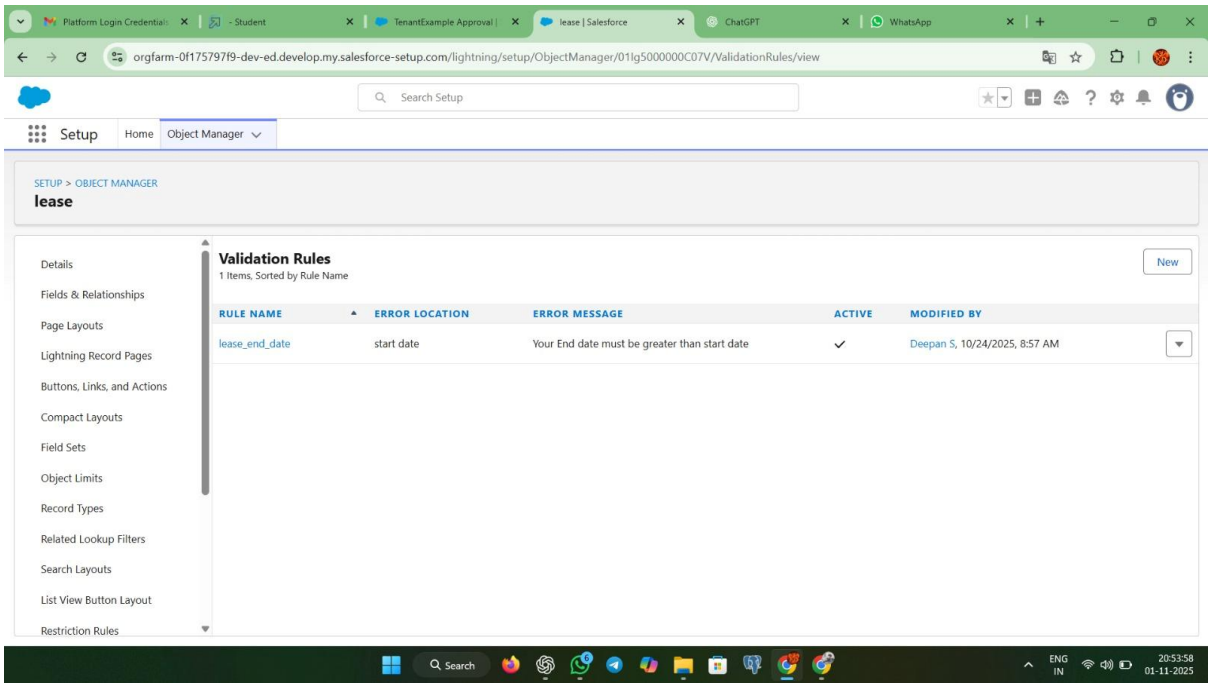
Developed a custom app named **Lease Management** to bring together all the objects and features under a unified interface.



Relationships:

- *Lookup Relationships* between Lease and Property, and Payment for Tenant and Tenant.
- *Master-Detail Relationship* between Property and Payment for Tenant.

Validation Rule:
Ensured logical consistency by validating that the *End Date* of a lease is greater than the *StartDate*.



Approval Process:
Implemented an approval workflow for tenant status ("Stay" or "Leaving") that triggers automated email alerts for approval or rejection.

When the tenant submits the request, an email using the **"Tenant Leaving"** template is sent for approval. If **approved**, a **"Leave Approved"** email confirms the tenant can vacate. If **rejected**, a **"Leave Rejected"** email notifies that the request is denied.

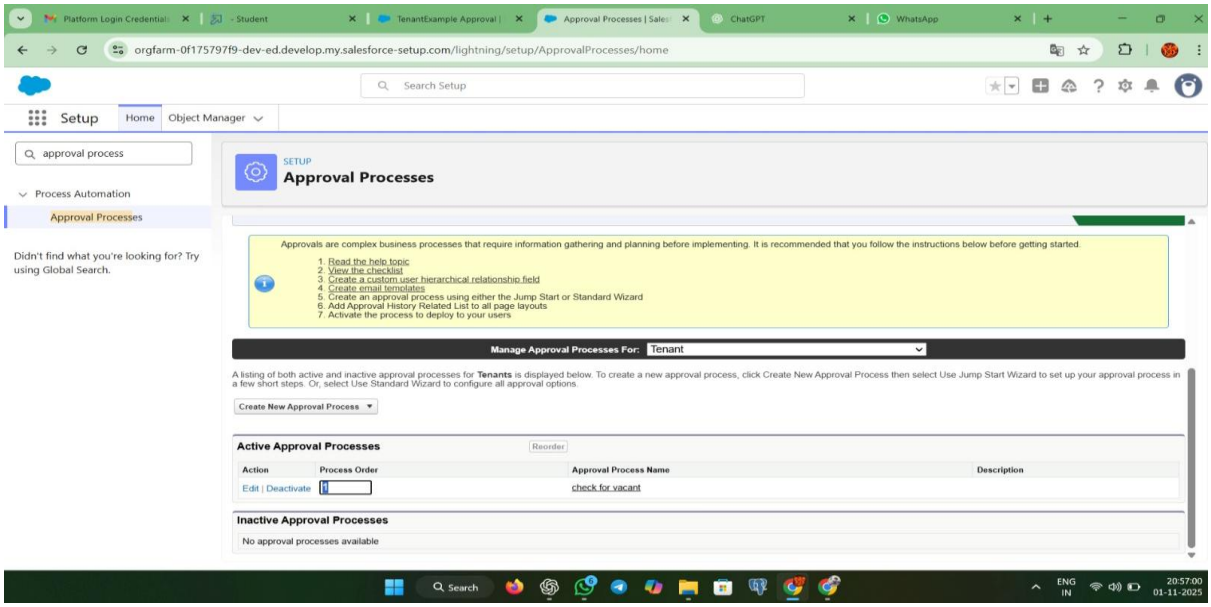
Email Templates Used Various email templates are used to automate communication:

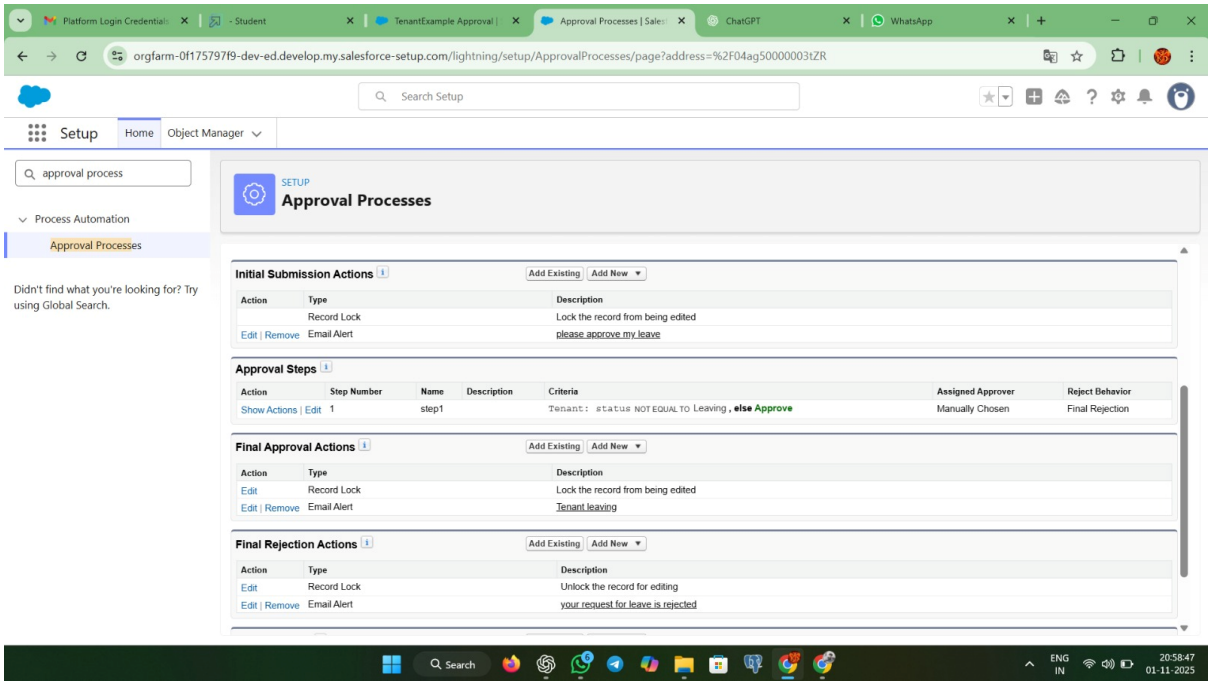
Tenant Leaving: Request for leave approval.

Leave Approved / Leave Rejected: Notify decision on the request.

Monthly Payment Reminder: Sends payment reminders and auto-confirmation when marked as "Paid" (via Flow).

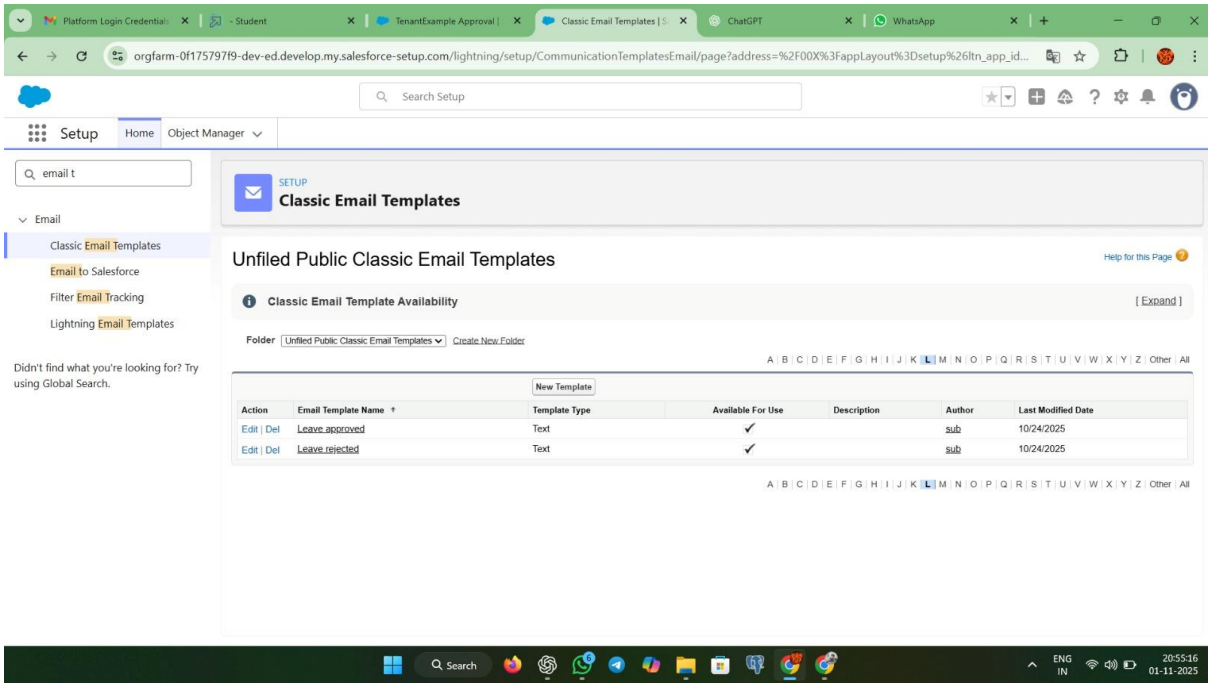
Successful Payment: Confirms the rent payment has been received

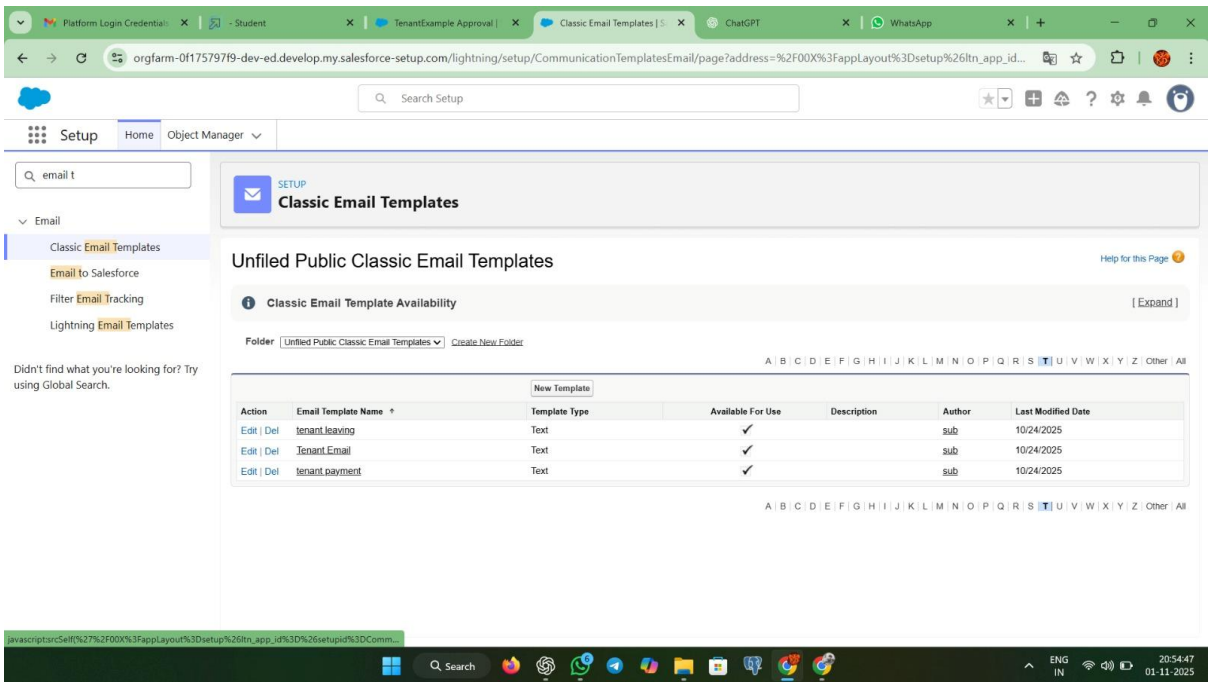




Email Templates:

In this project, various email templates are created to automate communication between tenants and property managers. These templates help in managing tenant requests, approvals, and payments efficiently without manual effort.

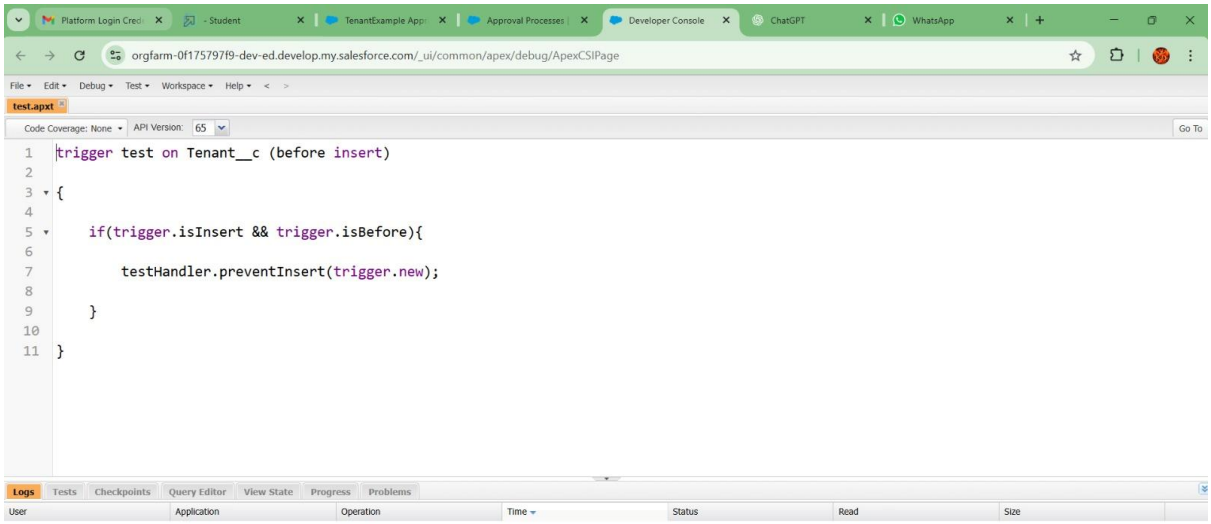


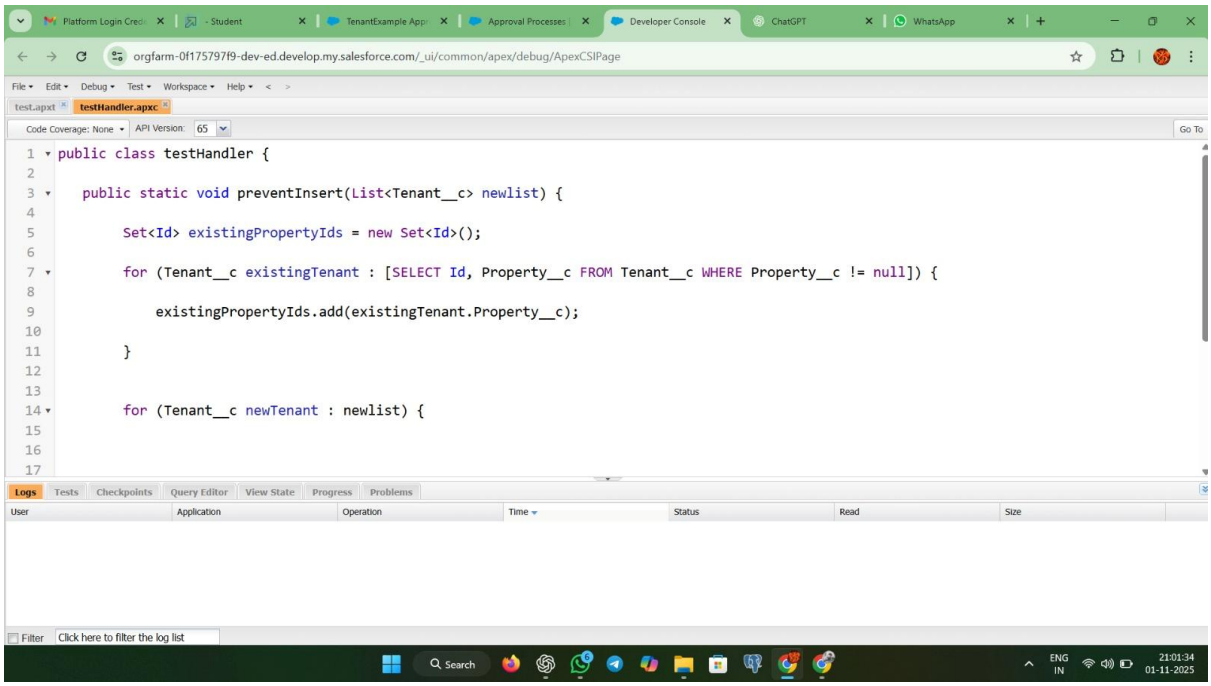


Apex trigger and Handler class:

Implemented business logic to prevent multiple tenants from being assigned to the same property. An **Apex Trigger** and **Handler Class** are created to maintain data integrity between tenants and properties. The trigger, named **"test"**, runs **before inserting** a new tenant record. It ensures that each property can only be linked to **one tenant at a time**.

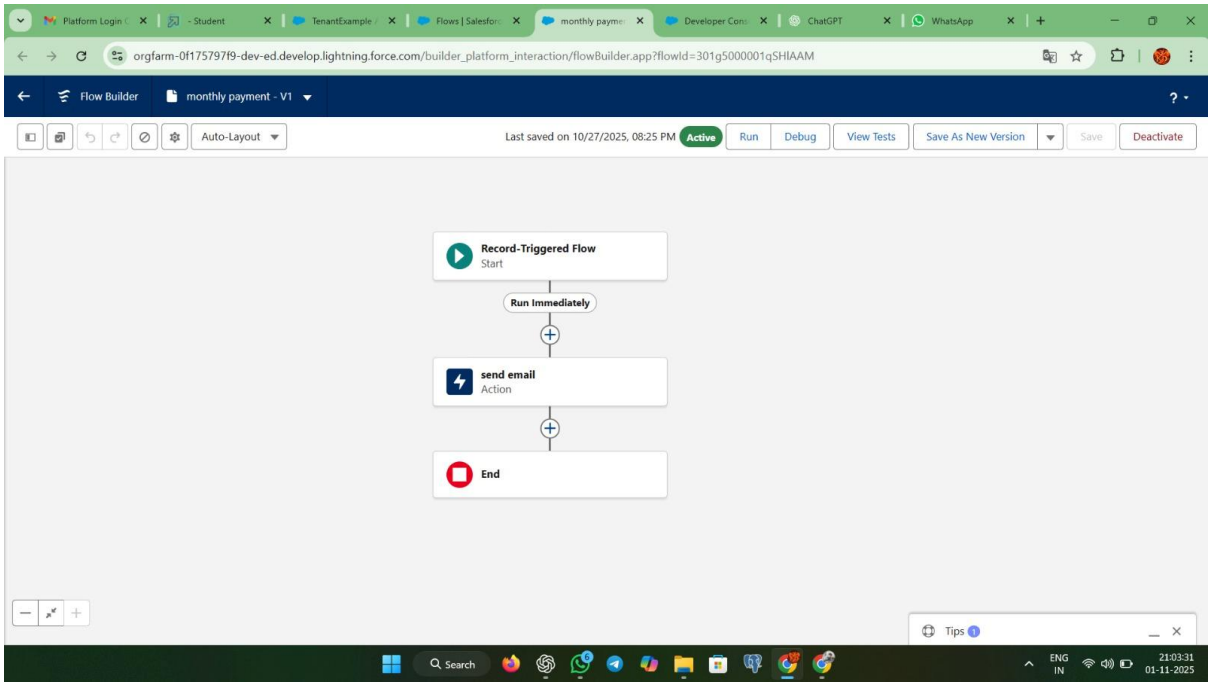
The **Handler Class (testHandler)** contains the main logic. It checks whether the property is already assigned to another tenant.

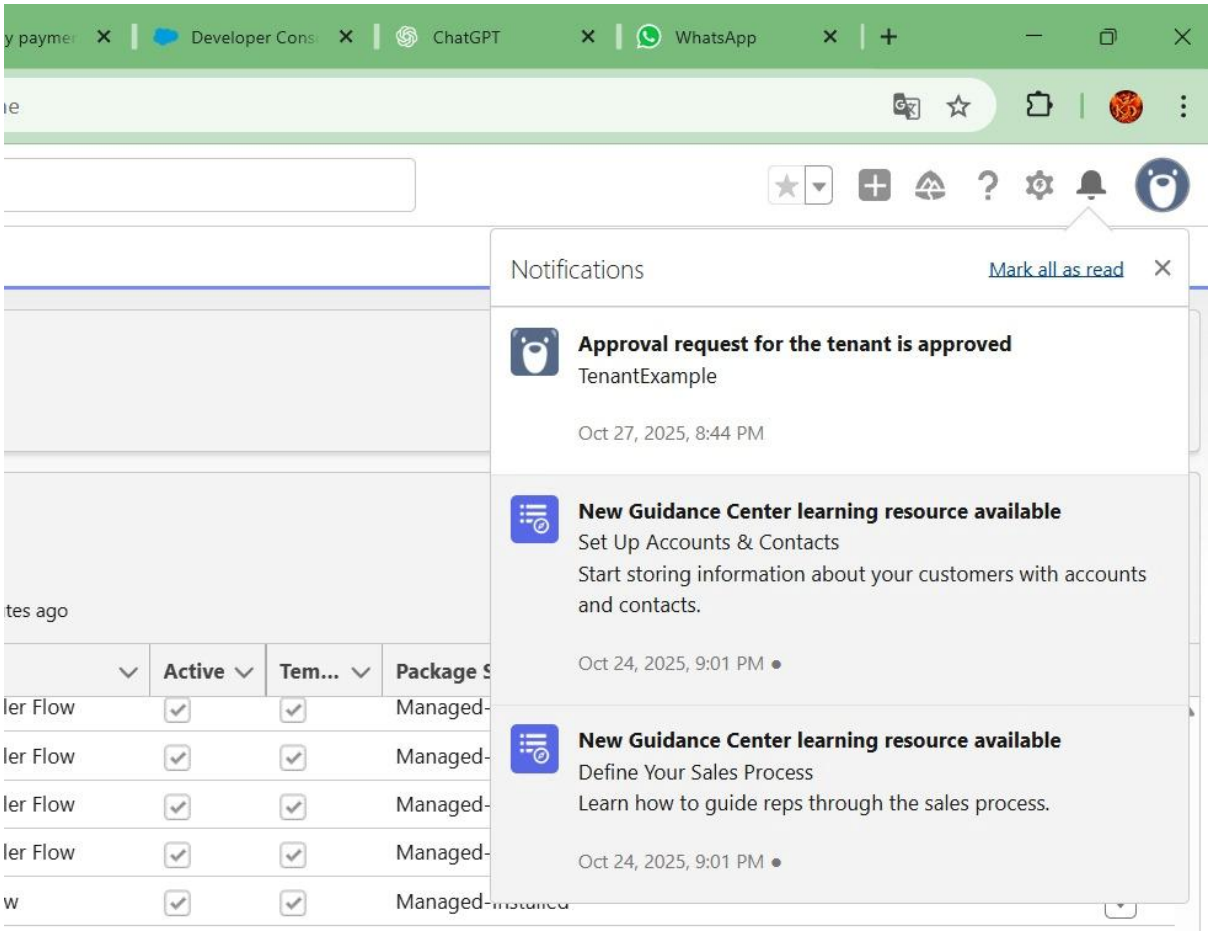




Flows and scheduled classes:

Automated monthly payment reminders using Record-Triggered Flows and an Apex Scheduler to send emails on the 1st of each month.


Flow Builder
monthly payment - V1
Auto-Layout
Last saved on 10/27/2025, 08:25 PM
Active
Run
Debug
View Tests
Save As New Version
Save
Deactivate
Record-Triggered Flow
Start
Run Immediately
send email
Action
End
Tips


Developer Cons
ChatGPT
WhatsApp
Notifications
Mark all as read
Approval request for the tenant is approved
TenantExample
Oct 27, 2025, 8:44 PM
New Guidance Center learning resource available
Set Up Accounts & Contacts
Start storing information about your customers with accounts and contacts.
Oct 24, 2025, 9:01 PM
New Guidance Center learning resource available
Define Your Sales Process
Learn how to guide reps through the sales process.
Oct 24, 2025, 9:01 PM
Active
Tem...
Package S
Managed-

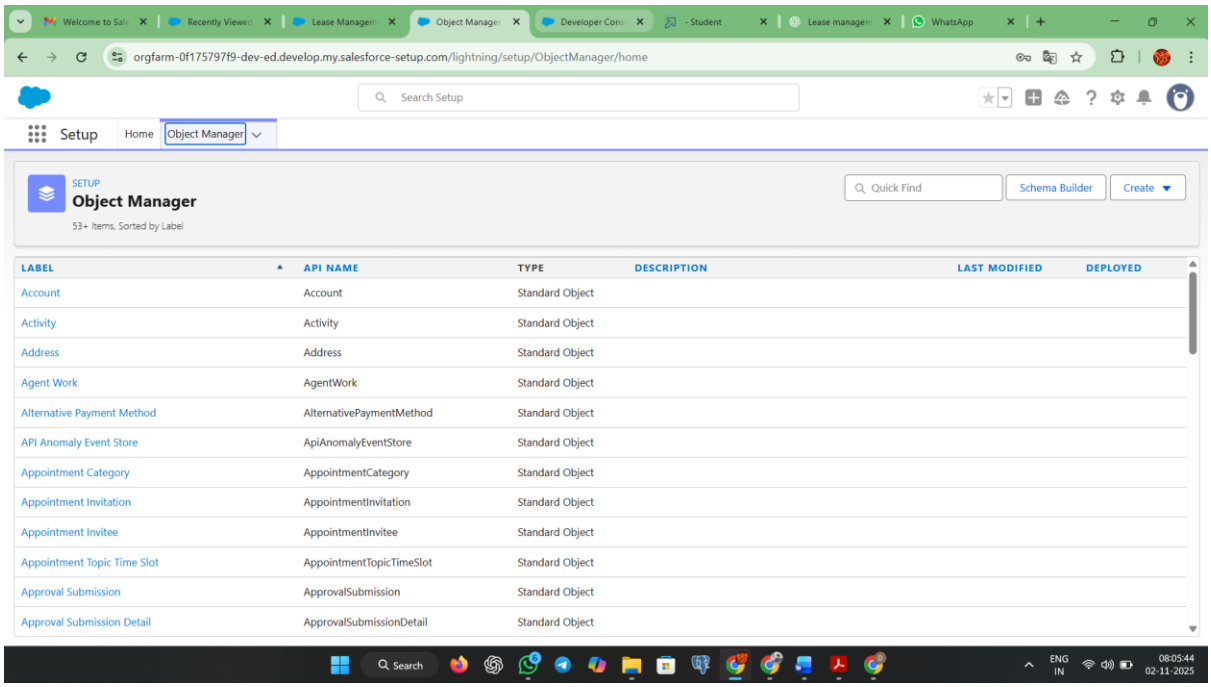
Detailed Steps to Solution Design

Step 1: Custom Object Creation

Created the following custom objects using the Object Manager:

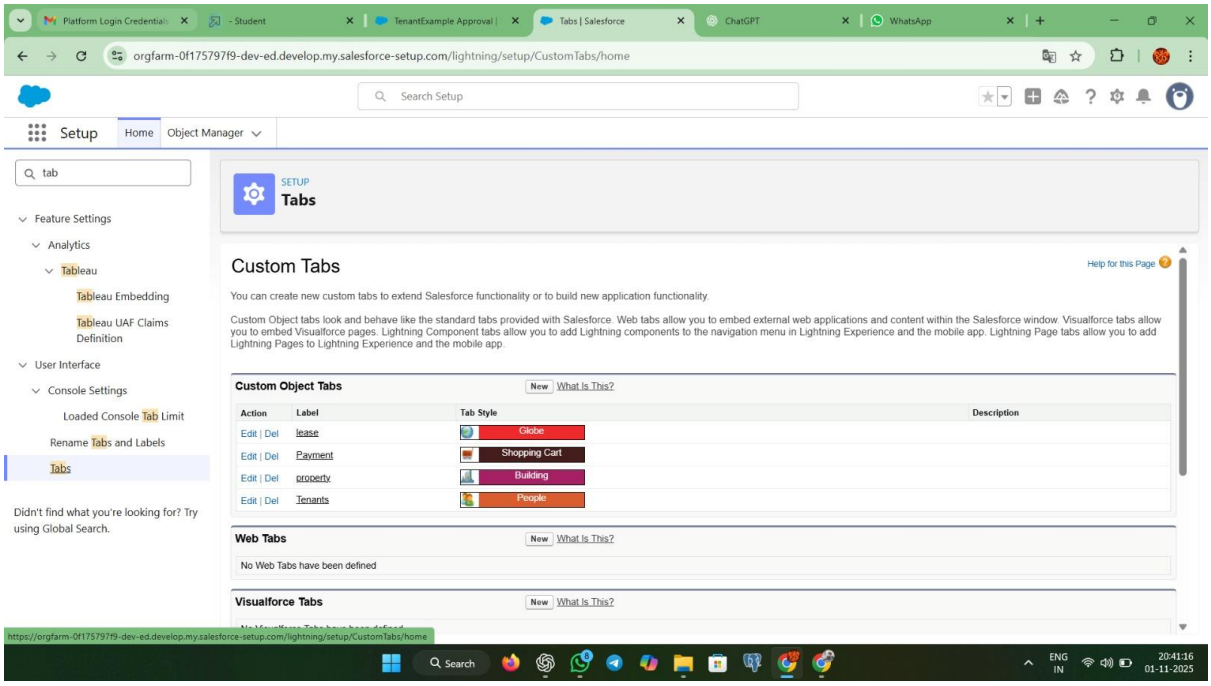
- **Property** – to store property details.
- **Tenant** – to record tenant information.
- **Lease** – to manage lease agreements.
- **Payment for Tenant** – to track rent payments.

Each object was configured with “Allow Reports,” “Track Field History,” “Allow Activities,” and “Allow Search.”



Step 2: Creating Custom Tabs

Custom tabs were created for all four objects (Property, Tenant, Lease, and Payment for Tenant) to allow easy navigation and data entry within the Lightning App.

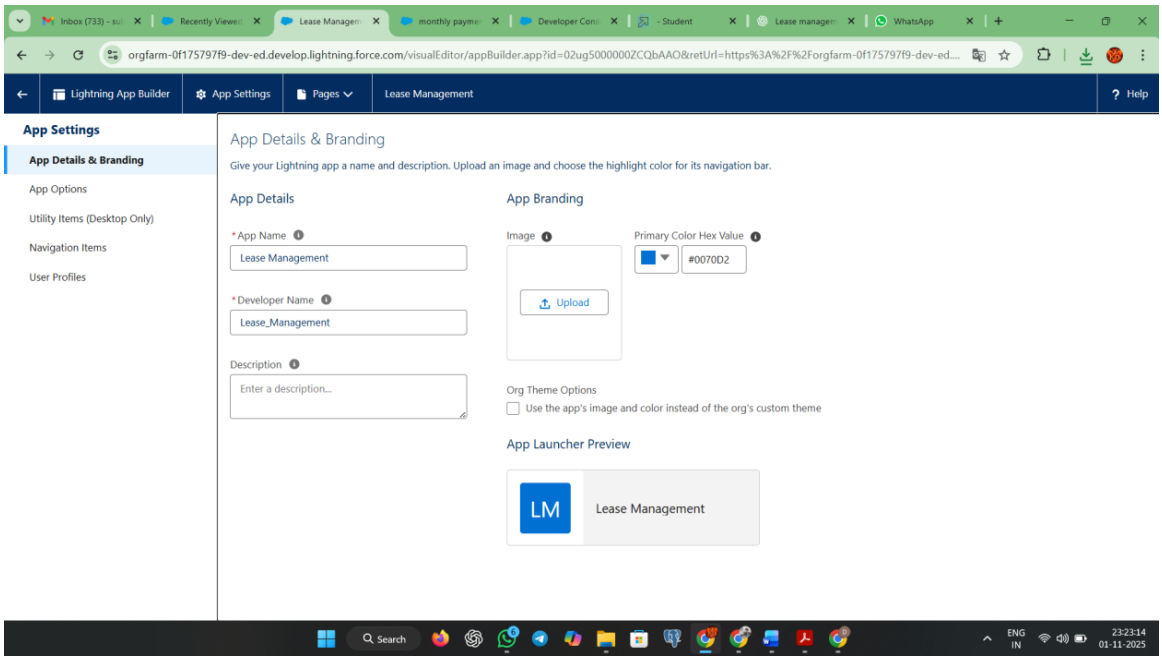


Step 3: Lightning App Creation

Developed a **Lightning App** named *Lease Management* with navigation items for:

- Property
- Tenant
- Lease
- Payment for Tenant

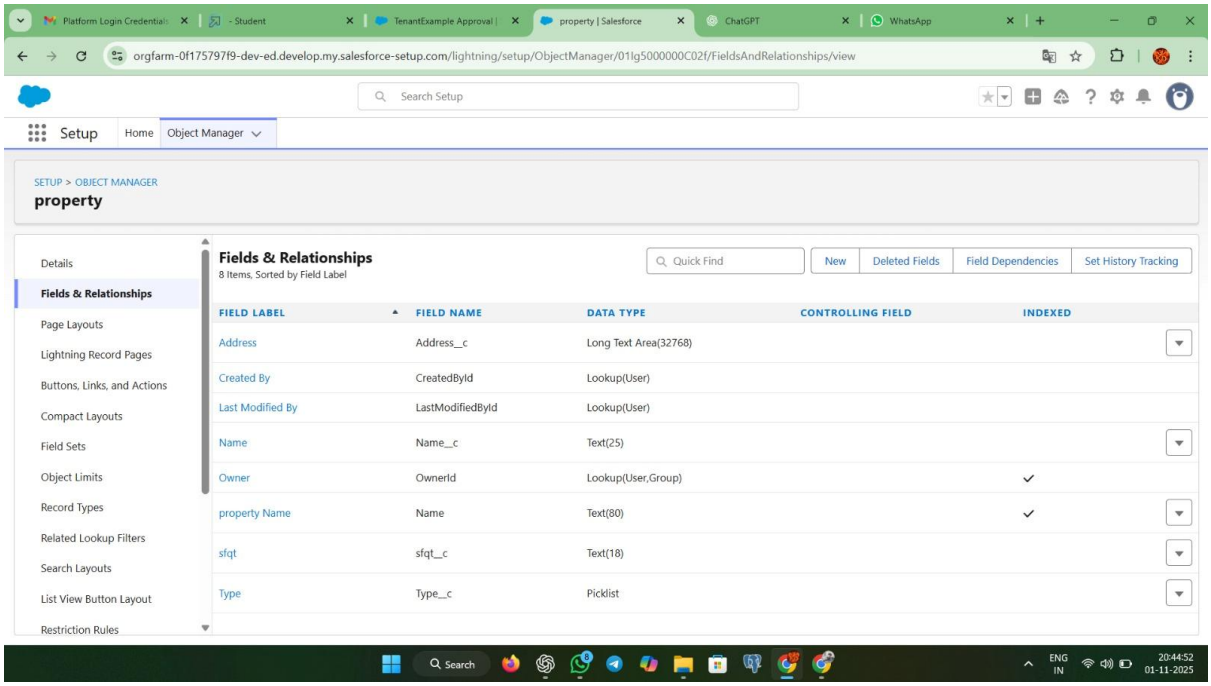
Set the navigation style as *Standard Navigation* and assigned the *System Administrator* profile for app access.



Step 4: Field Creation for Property Object

Added the following fields:

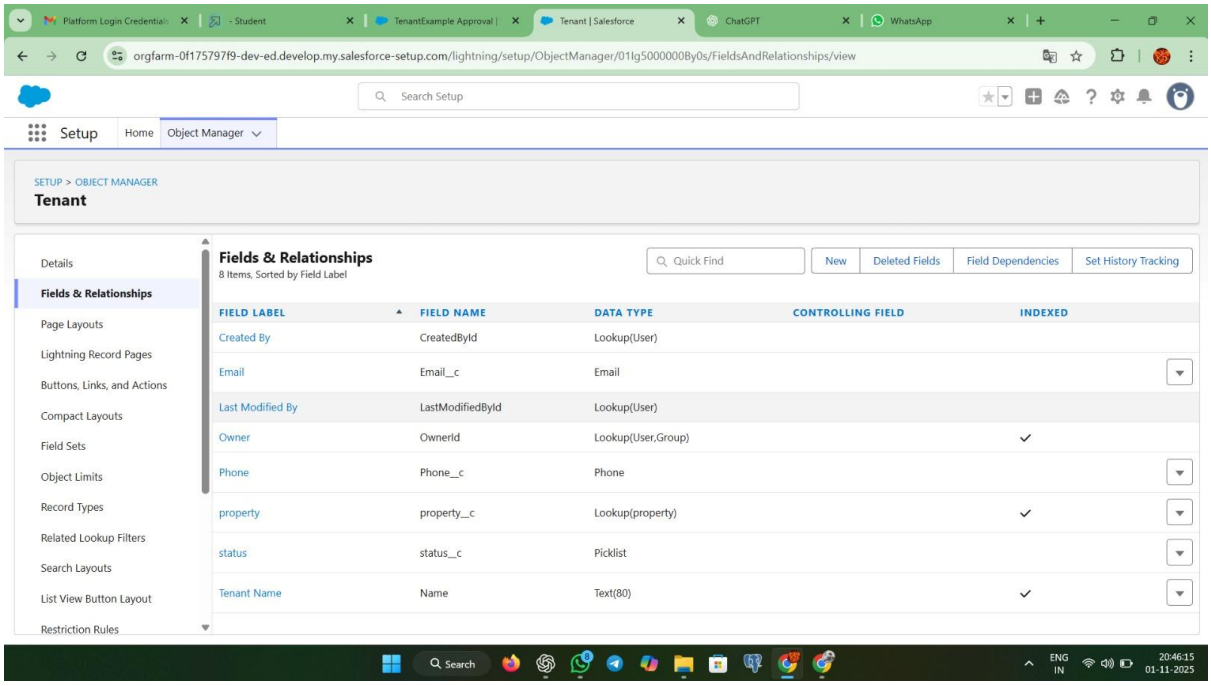
- **Name** (Text, Required)
- **Address** (Long Text Area)
- **Type** (Picklist: 1BHK, 2BHK, 3BHK)
- **Sqft** (Text, Length: 18)



Step 5: Field Creation for Tenant Object

Added key fields:

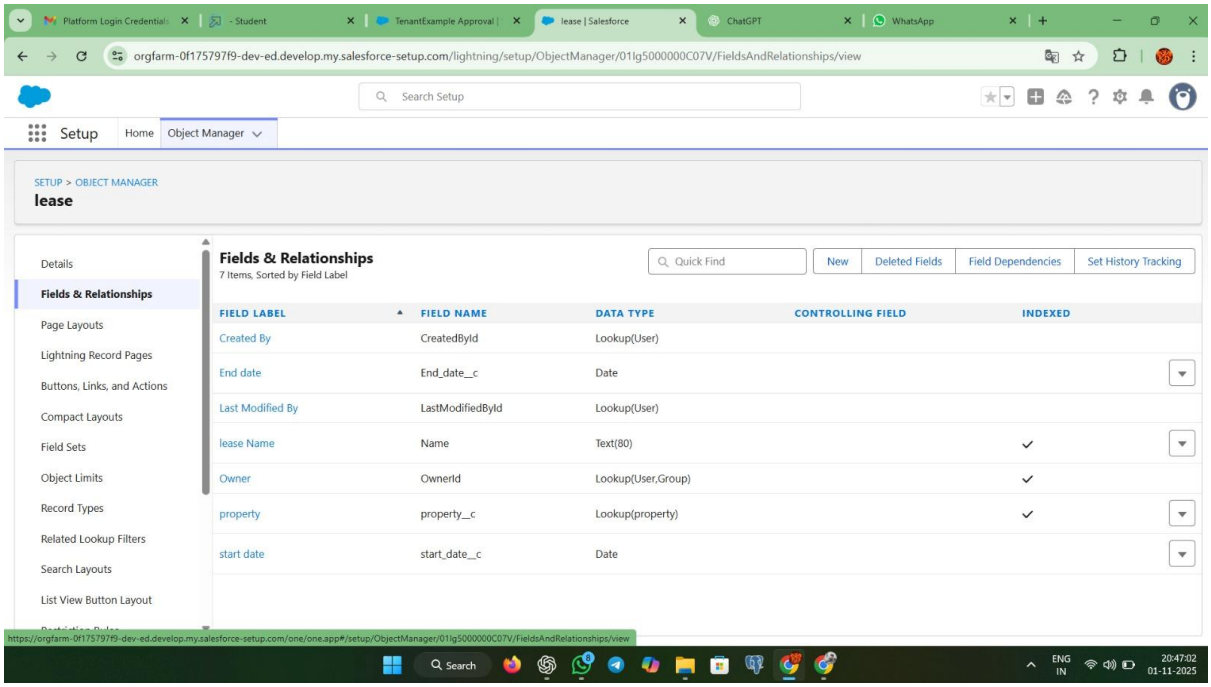
- **Email** (Email, Required)
- **Phone** (Phone)
- **Status** (Picklist: Stay, Leaving)



Step 6: Field Creation for Lease Object

Configured:

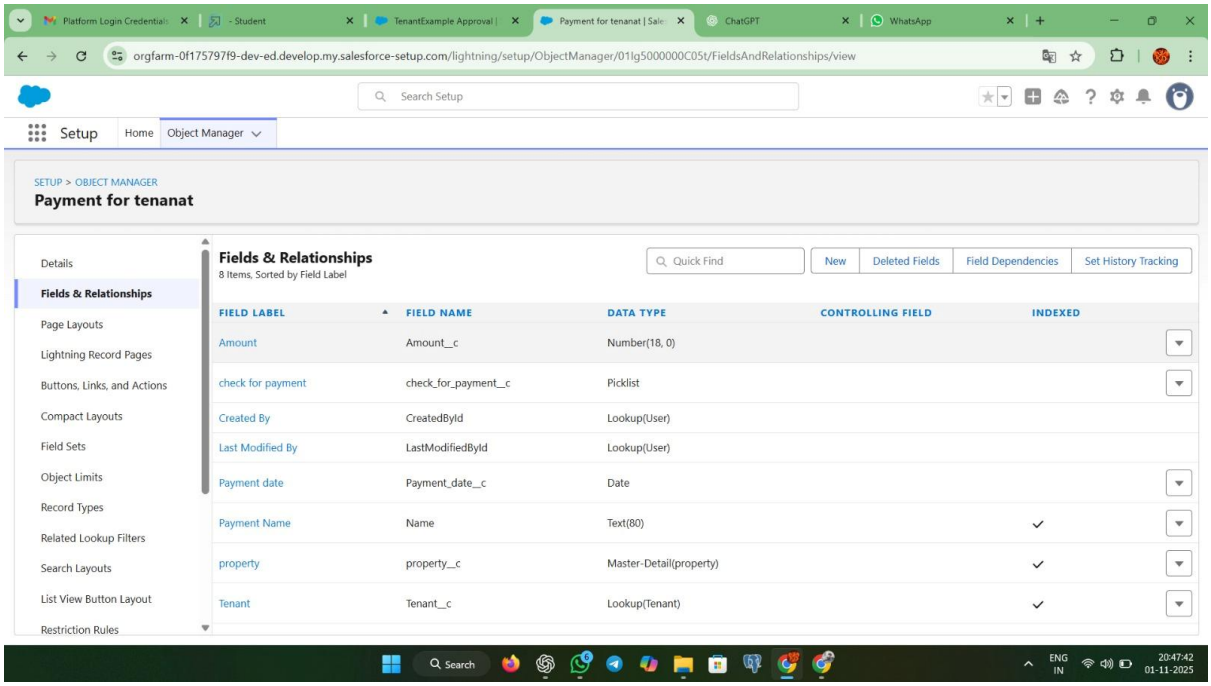
- **Start Date** (Date)
- **End Date** (Date)



Step 7: Field Creation for Payment for Tenant Object

Defined fields:

- **Payment Date** (Date)
- **Amount** (Number, Length: 18)
- **Check for Payment** (Picklist: Paid, Not Paid)



Step 8: Relationship Creation

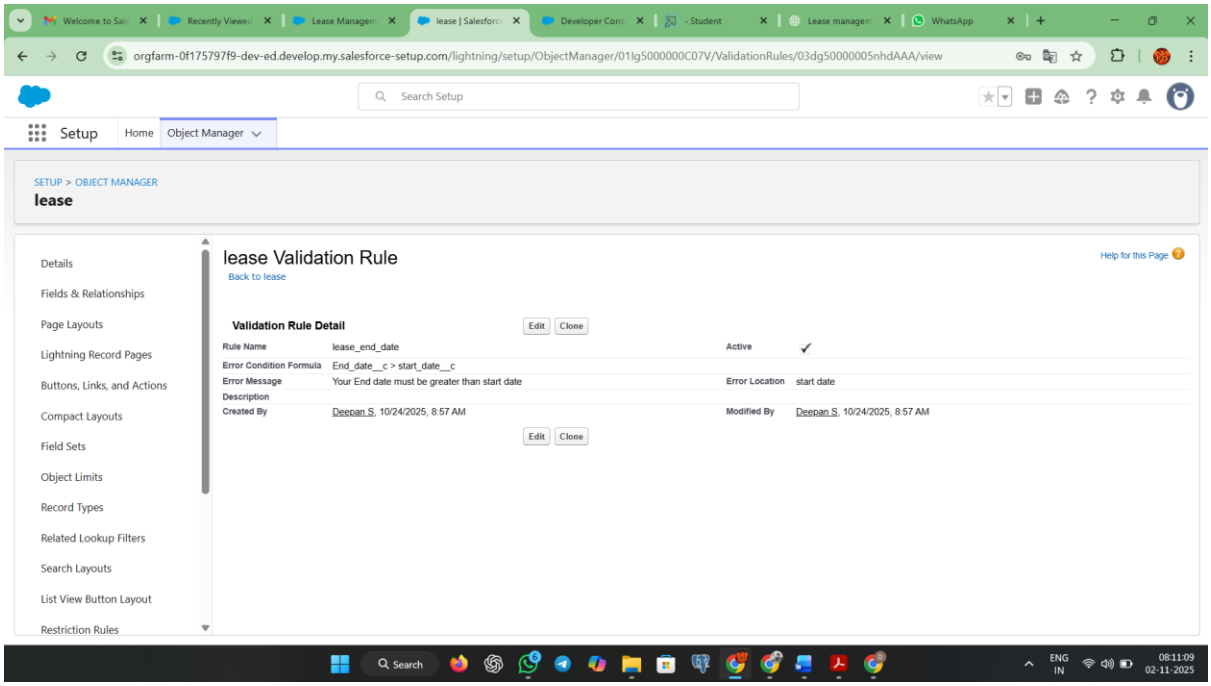
- **Lease → Property:** Lookup Relationship
- **Payment for Tenant → Tenant:** Lookup Relationship
- **Payment for Tenant → Property:** Master-Detail Relationship

These relationships ensure linked and consistent data across all objects.

Step 9: Validation Rule

Created a validation rule on **Lease Object**:

- **Rule Name:** lease_end_date
- **Condition:** End_date__c < Start_date__c
- **Error Message:** "Your End date must be greater than start date."
- **Error Location:** Field → Start Date



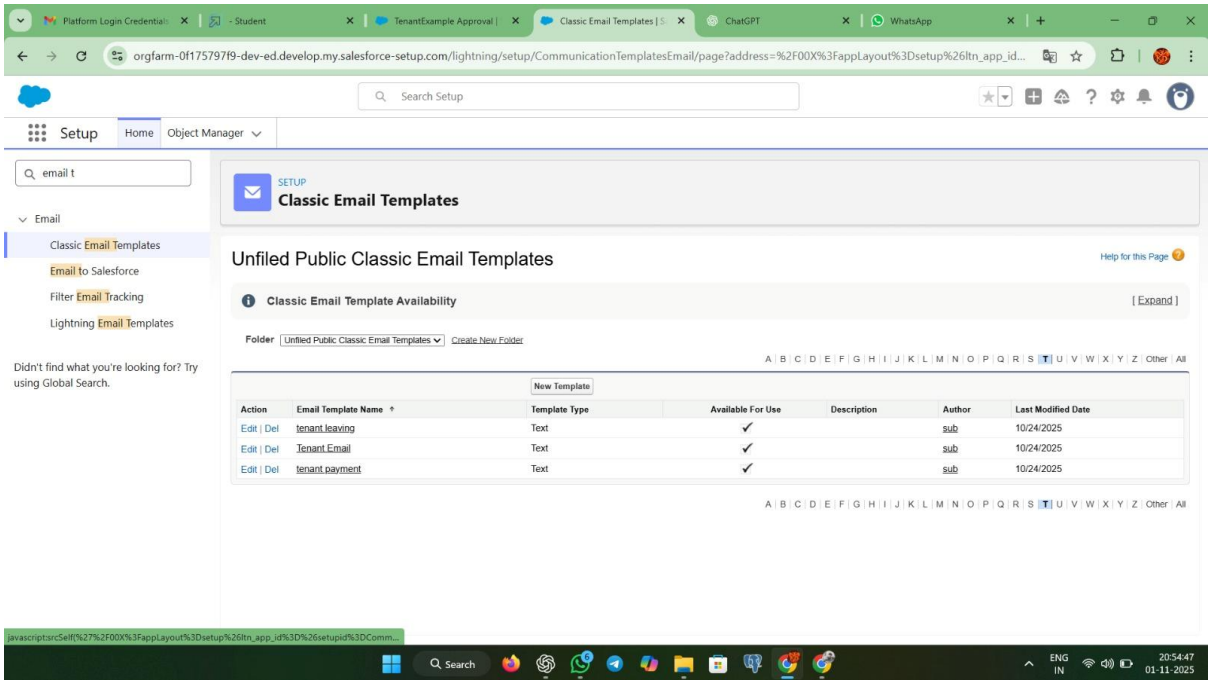
Step 10: Email Templates

Created multiple Classic Email Templates for communication:

Tenant Leaving – request approval for leave.

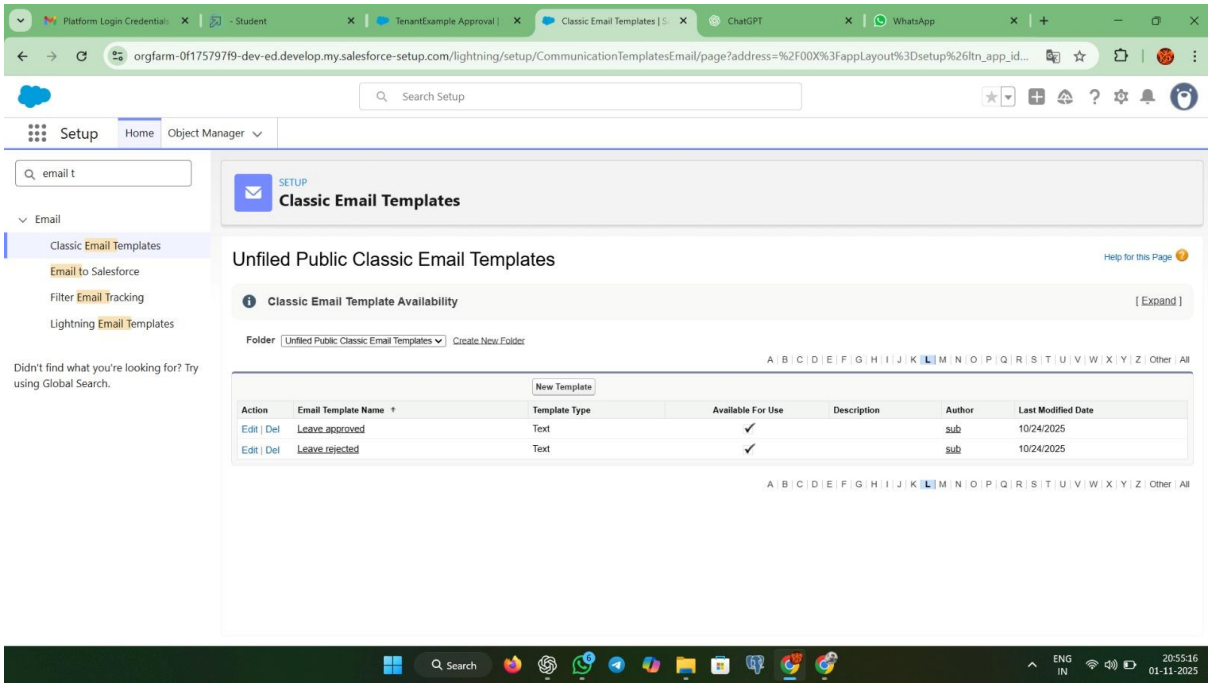
Leave Approved – confirmation email for approved leave.

Leave Rejected – notification for rejected leave.



Tenant Email – monthly rent payment reminder.

Tenant Payment – successful payment confirmation.

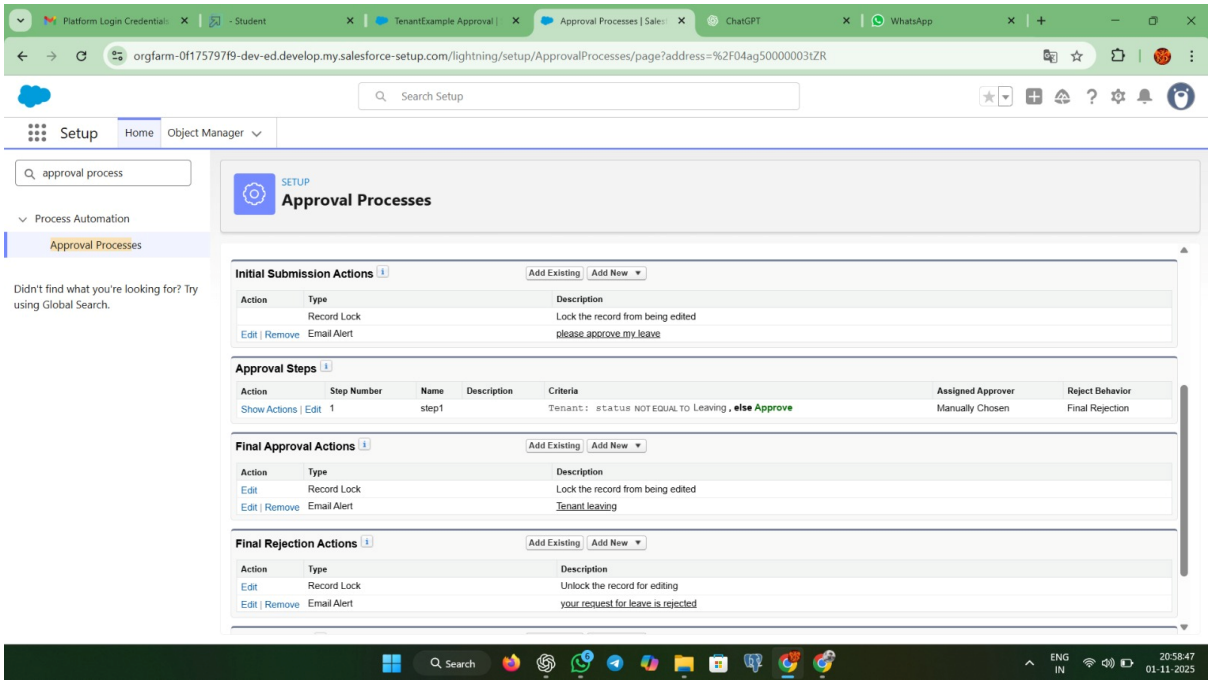


Step 11: Approval Process

Created an approval process named **Check for Vacant** for the *Tenant* object.
It automates email notifications and record updates during the approval lifecycle using the templates created above.

Actions:

- **Initial Submission:** Sends *Tenant Leaving* email.
- **Final Approval:** Sends *Leave Approved* email
- **Final Rejection:** Sends *Leave Rejected* email



Step 12: Apex Trigger and Handler Implementation

- Created an **Apex Trigger** and **Handler Class** to ensure one property is assigned to only one tenant.
- Prevents duplicate tenant assignments and maintains clean, accurate data.

Step 13: Scheduled Apex Class

- Implemented a **Scheduled Apex Class (MonthlyEmailScheduler)** to send rent reminder emails automatically on the first day of every month.
- This helps maintain regular rent payments and consistent communication with tenants.

Key Scenarios Addressed by Salesforce

- **Property Management:** Salesforce custom objects simplify property record maintenance.
- **Tenant Tracking:** Ensures every tenant is linked to a property with no duplication.
- **Lease Validation:** Validation rule ensures accurate date management.
- **Automated Communication:** Email templates and flows improve tenant engagement.
- **Approval Workflow:** Automates tenant status updates with admin approval.

Payment Tracking: Payments and reminders are handled via flows and scheduled emails

Conclusion

The **Lease Management System** project demonstrates the effective use of Salesforce to streamline rental operations. Through automation, relationship management, and workflow logic, the system enhances accuracy, transparency, and communication.

Key accomplishments include:

- End-to-end automation of lease, payment, and tenant processes.
- Secure and relational data structure.
- Automatic email notifications for tenants.
- Validation and approval systems for better control.
- Improved operational efficiency for property administrators.