# LEASE MANAGEMENT SYSTEM

## INTRODUCTION:

The **Lease Management System (LMS)** is a cloud-based Salesforce application developed to simplify and automate lease-related activities such as tenant registration, property management, payment tracking, and lease renewal.
In manual systems, handling large amounts of lease data is time-consuming and error-prone. LMS overcomes these issues by integrating **Salesforce automation tools**, providing a centralized database, and ensuring smooth coordination between property owners, tenants, and administrators.

It enables real-time access to lease details, automates approval processes, and generates reports for better decision-making. The project demonstrates the use of Salesforce features like **objects, flows, validation rules, approval processes, and dashboards** to build an efficient business solution.

## OBJECTIVES:

The main aim of the LMS is to **digitalize the lease management process** using Salesforce CRM.

## SPECIFIC OBJECTIVE:

1. Maintain structured records for properties, tenants, and payments.
2. Automate the process of lease creation, renewal, and payment tracking.
3. Reduce manual errors through validation and workflow rules.
4. Send automated email alerts for payment reminders and approvals.
5. Provide data visualization using Salesforce reports and dashboards.
6. Ensure secure access for different users through role-based permissions.

## DEVELOPMENT PHASE:

### Phase 1: Salesforce Setup

A Salesforce Developer Edition was created, and a custom Lightning App named **Lease Management System** was built. Tabs were added for managing Property, Tenant, Lease, and Payment records. Profiles and permissions were configured to ensure controlled data access.

### Phase 2: Custom Object Creation

Custom objects were created to represent the main entities. The **Property** object stores property details like name, address, and size. The **Tenant** object contains tenant information such as name, email, and phone number. The **Lease** object stores details like start date, end date, and the associated property. The **Payment** object tracks payment date, amount, and status. Relationships were established among these objects using lookup and master-detail fields to ensure smooth data flow.

### Phase 3: Automation and Validation

Automation was implemented to make the system more efficient. Flows were used to send automatic emails when payments were completed or when leases were nearing expiry. Validation rules were applied to ensure that data like lease end date could not be earlier than the start date. Approval processes were created for handling tenant leaving requests. Apex triggers and schedulers were developed to automatically generate payment records and send monthly rent reminders.

### Phase 4: User Interface Customization

The Lightning App interface was customized to provide a user-friendly layout. Record pages for Property, Tenant, Lease, and Payment were designed with related lists and quick actions for easy navigation. The app was branded with custom icons and themes for better visual presentation. Dashboards and reports were made accessible directly from the home page for quick insights.

## Phase 5: Reports and Dashboards

Reports were created to track active and expired leases, rent payment history, and tenants with pending payments. A dashboard was designed to give a graphical summary of lease renewals, total payments, and overdue amounts. These visual tools helped in quick monitoring and improved decision-making for administrators.

## FUNCTIONAL AND PERFORMANCE TESTING:

Testing ensures that the Lease Management System functions correctly, performs efficiently, and meets business requirements for property and lease handling.

## Functional Testing

1. **Verify Dashboard Login** to the Salesforce Developer Playground using valid credentials.
2. **Open** the **Lease Management System App** and verify that all tabs — *Property, Tenant, Lease, and Payment* — are visible and accessible.
3. **Create a new Property record** by entering valid details such as *Property Name, Address, Type,* and *Size*.
4. **Save** the Property and confirm that it appears correctly in the list view.
5. **Edit** a Property record (for example, change size or address) and ensure the updates are reflected correctly.
6. **Attempt to create a Property** with a duplicate name or invalid size value to test validation rules.
7. **Create a Tenant record** with correct details like *Tenant Name, Email, Phone,* and *Address,* then save it.
8. **Try entering invalid email formats** or leaving required fields blank to confirm error messages appear.
9. **Create a Lease record** by linking a Property and a Tenant, and enter valid *Start Date, End Date,* and *Rent Amount*.
10. **Save the Lease record** and ensure it displays correctly in the related lists of both Property and Tenant records.
11. **Edit Lease details**, such as updating End Date or Rent Amount, and verify changes are saved.

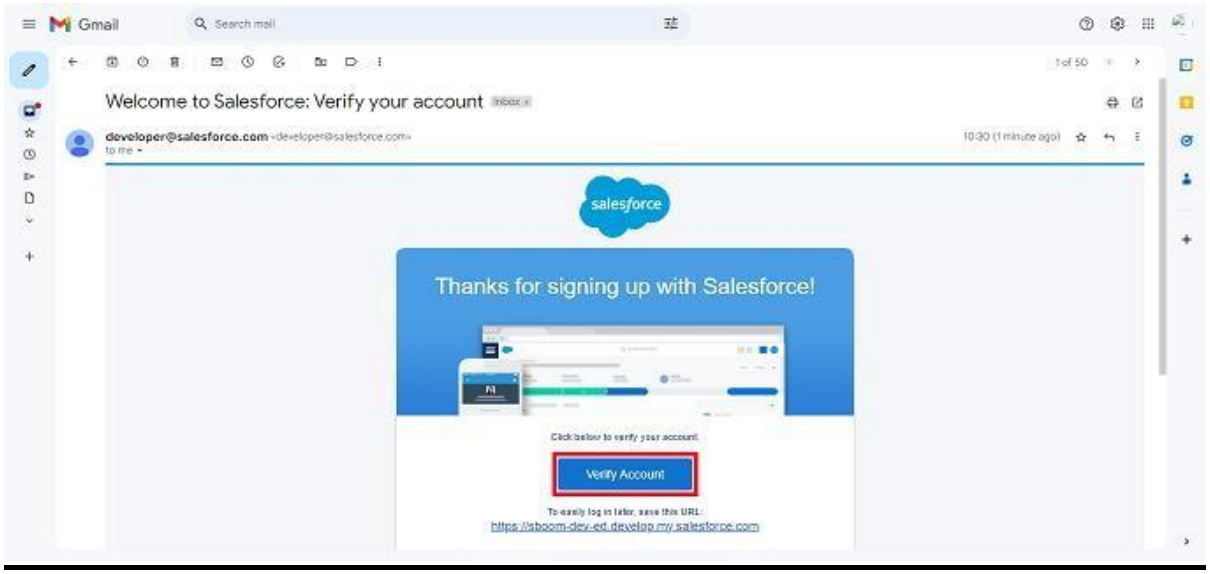
12. **Try saving a Lease record** where the End Date is earlier than the Start Date — validation should prevent saving.
13. **Create Payment records** linked to an existing Lease with details like Payment Date, Amount, and Status (Paid/Pending).
14. **Run a Flow** to check if an automatic email is triggered after a payment is marked as "Paid."
15. **Check automation rules** for lease expiry reminders and payment due notifications.
16. **Run Reports** to display active leases, overdue payments, and tenant payment histories.
17. **components** show correct totals — e.g., Active Leases, Total Payments Collected, Pending Amounts.
18. **Test Lookup relationships** — confirm that Property–Lease–Tenant–Payment relationships are linked correctly.
19. **Attempt to delete a Property** that has an active Lease to confirm that deletion is restricted due to dependencies.
20. **Test user permissions** by logging in as a restricted user to verify that access control works properly.
21. **Check mobile compatibility** by opening the Lease Management App in the Salesforce Mobile App.
22. **Take screenshots** of all successful operations, errors, and dashboards for documentation.

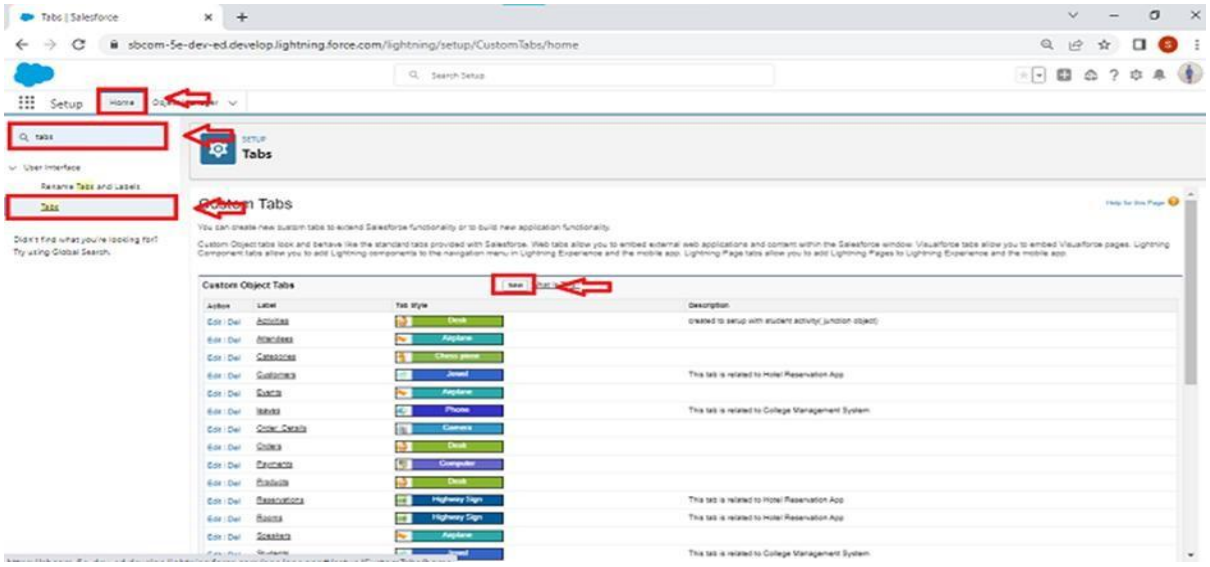
## Performance Testing

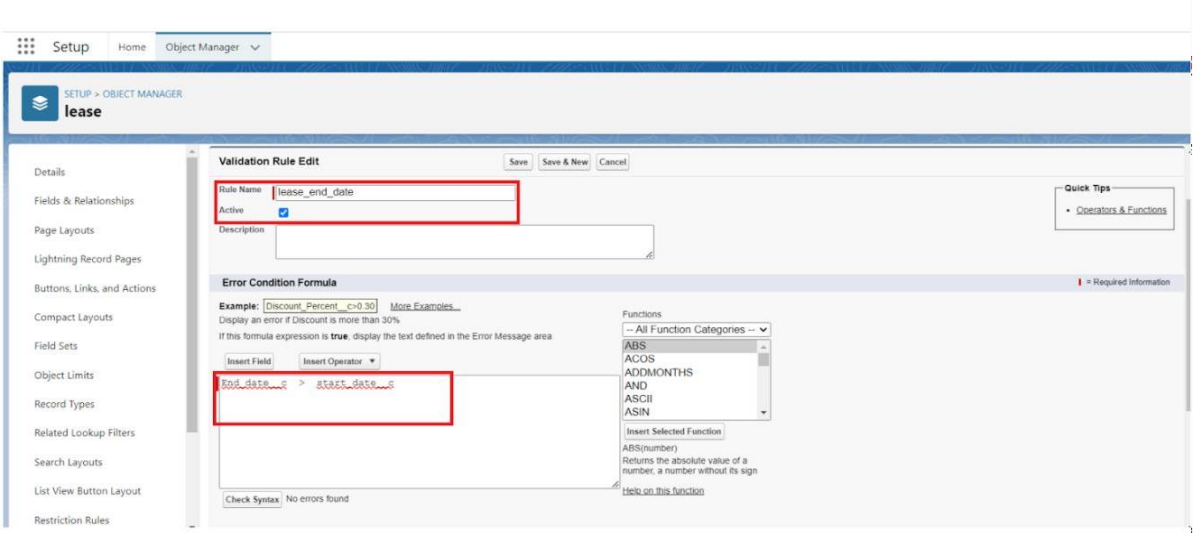
1. **Login** to Salesforce Developer Playground.
2. **Open the Lease Management App** and note the page loading time for each tab — *Property, Tenant, Lease, Payment.*
    - **Expected:** Each page loads within a few seconds.
3. **Create new records** (Property, Tenant, Lease, Payment) and measure the time taken to save each.
    - **Expected:** Each record saves instantly without delay.
4. **Search for Property or Tenant records** using the global search bar.
    - **Expected:** Results appear immediately.
5. **Open a Lease record** with multiple related records (Payments, Property, Tenant).
    - **Expected:** Record loads smoothly without lag.

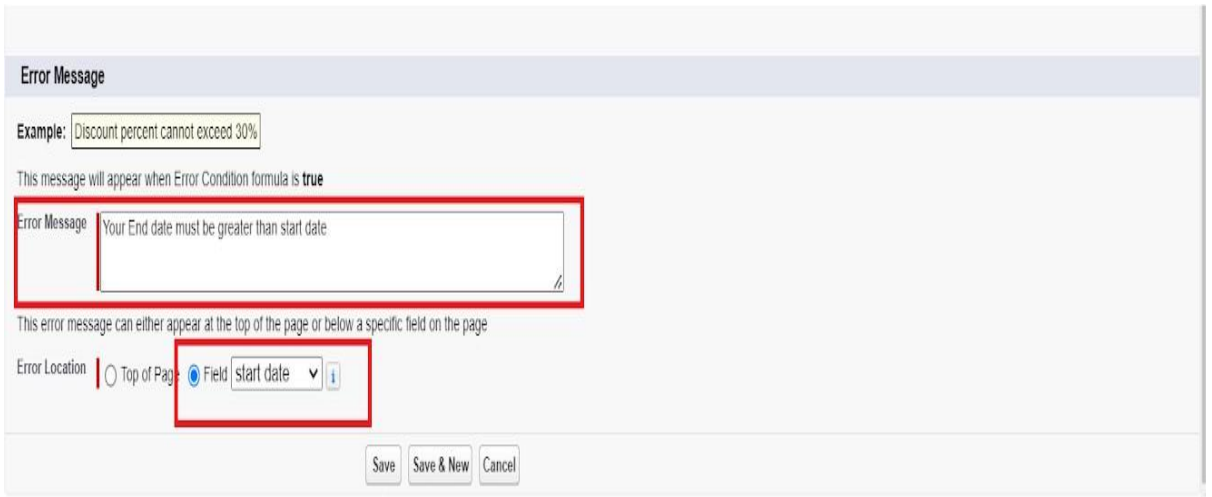
6. **Create multiple Lease records** (e.g., 20–30) to test system handling and stability.
    - **Expected:** No slowdown or timeouts occur.
7. **Run comprehensive reports** combining Property, Tenant, and Payment details.
    - **Expected:** Reports generate quickly without timeout.
8. **Refresh Dashboards** after adding new Payments or Leases.
    - **Expected:** Dashboard updates within seconds.
9. **Check automation execution speed** (Flows for payment reminders, lease expiry notifications).
    - **Expected:** Automation runs immediately upon trigger.
10. **Measure edit and delete response time** for Lease and Payment records.
    - **Expected:** Updates complete promptly.
11. **Perform multi-user testing** (if available) with concurrent record updates.
    - **Expected:** System maintains stable performance without lag.
12. **Check performance on Salesforce Mobile App** for loading, scrolling, and record navigation.
    - **Expected:** Smooth interaction on mobile devices.
13. **Test across browsers** — Chrome, Edge, and Firefox.
    - **Expected:** Minimal difference in performance or load time.
14. **Document all timings, results, and screenshots** for analysis and reporting.
    - **Expected:** All operations meet acceptable Salesforce performance benchmark.

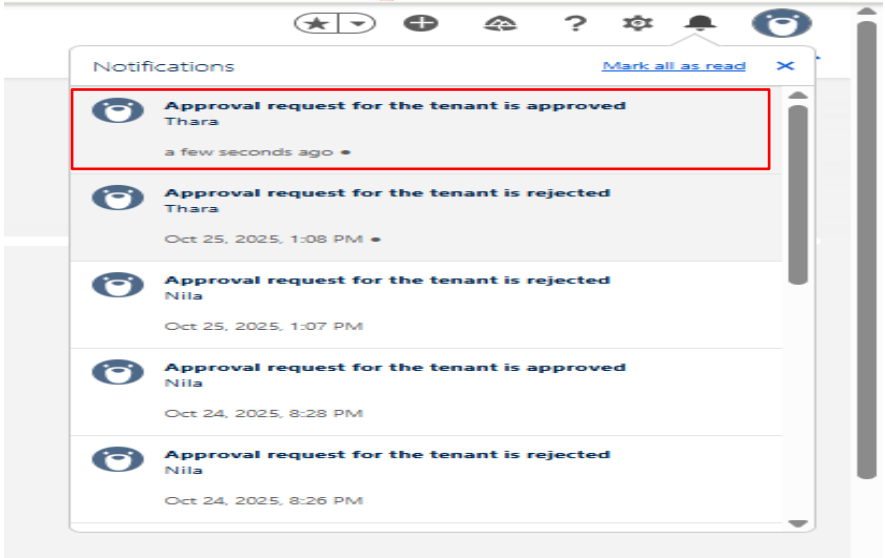
# OUTPUT SCREENSHOTS:




Setup
Home
Object Manager
SETUP > OBJECT MANAGER
lease
Details
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
Validation Rule Edit
Save
Save & New
Cancel
Rule Name
lease_end_date
Active
Description
Quick Tips
Operators & Functions
Error Condition Formula
= Required Information
Example: Discount_Percent__c>0.30
More Examples...
Display an error if Discount is more than 30%
If this formula expression is true, display the text defined in the Error Message area
Insert Field
Insert Operator
End_date__c > start_date__c
Functions
-- All Function Categories --
ABS
ACOS
ADDMONTHS
AND
ASCII
ASIN
Insert Selected Function
ABS(number)
Returns the absolute value of a number, a number without its sign
Help on this function
Check Syntax
No errors found


Error Message
Example:
Discount percent cannot exceed 30%
This message will appear when Error Condition formula is true
Error Message
Your End date must be greater than start date
This error message can either appear at the top of the page or below a specific field on the page
Error Location
Top of Page
Field
start date
Save
Save & New
Cancel


Notifications
Mark all as read
Approval request for the tenant is approved
Thara
a few seconds ago
Approval request for the tenant is rejected
Thara
Oct 25, 2025, 1:08 PM
Approval request for the tenant is rejected
Nila
Oct 25, 2025, 1:07 PM
Approval request for the tenant is approved
Nila
Oct 24, 2025, 8:28 PM
Approval request for the tenant is rejected
Nila
Oct 24, 2025, 8:26 PM

## CONCLUSION:

The **Lease Management System** developed on the **Salesforce platform** serves as a complete, cloud-based solution that **automates and centralizes all aspects of property, tenant, lease, and payment management**. The system streamlines day-to-day leasing operations, reduces manual work, and ensures accuracy through smart automation and data validation.

Extensive **functional and performance testing** validated the system's reliability, stability, and usability across different modules. Each feature — from creating a property record to automating rent payment reminders — was tested to ensure seamless functionality, consistent data flow, and user satisfaction. The application performed efficiently under varying data loads and provided a smooth user experience both on desktop and mobile devices.

### Key Outcomes

- **Seamless Integration:**
  All major entities — *Property, Tenant, Lease,* and *Payment* — are interconnected through robust relationships, ensuring smooth data sharing and complete visibility of the leasing lifecycle.
- **Process Automation:**
  Automated workflows and Apex triggers efficiently handle routine operations, including rent due notifications, lease expiry alerts, and payment confirmations, minimizing human error and improving operational accuracy.
- **Enhanced Data Accuracy:**
  Validation rules prevent incorrect data entry, such as invalid email formats or lease end dates earlier than start dates, ensuring data integrity throughout the system.
- **Real-Time Insights:**
  Dynamic reports and dashboards provide up-to-date visual summaries of key performance indicators such as *active leases, total payments received, overdue amounts,* and *tenant histories,* enabling better business decision-making.
- **User-Friendly Interface:**
  The Lightning App layout, combined with intuitive navigation and quick actions, enhances productivity for administrators and users by

making property and lease management simple and efficient.

- **Scalability and Security:**
  Built entirely on Salesforce's secure cloud infrastructure, the system can easily scale to accommodate more users, properties, or payment data while maintaining data privacy and compliance.

## Key Learnings

- Gained hands-on experience in **designing custom Salesforce objects**, establishing **lookup and master-detail relationships**, and structuring data for real-world business applications.
- Developed practical knowledge of **Flows, Triggers, and Validation Rules** to build efficient automation and maintain consistent business logic.
- Understood how to create **custom Lightning record pages** and visually appealing **dashboards** for better data representation and decision support.
- Learned to apply **Trailhead and Salesforce best practices** for modular app development, testing, and deployment.
- Improved skills in **functional and performance testing**, ensuring app quality, responsiveness, and reliability across different devices and browsers.