Performance Testing document

Date	3 August 2025
Team ID	LTVIP2025TMID30830
Project name	Lease management
Maximum Marks	

1.objective:

Here is the Performance Testing section for your Lease Management Project, based on the implementation and testing details from your document:

Performance Testing – Lease Management System

Performance testing was conducted to evaluate the responsiveness, speed, and reliability of the **Salesforce-based Lease Management Application** under expected usage conditions.

✅ 1. Object Load & Navigation Speed

🔍 **Objective:**
To ensure fast navigation and loading of custom Salesforce objects (Property, Tenant, Lease, and Payment) through the Lightning App.

🛠 **Actions Taken:**

- Accessed each custom object via dedicated tabs in the Lease Management App.
- Navigated between record views and lists.
- Tested responsiveness using multiple user profiles.

📈 **Result:**

- Pages loaded **in under 1 second** for all object views.
- Navigation between objects was smooth, with **no noticeable lag**.

✅ 2. Apex Trigger Execution Time

🔍 **Objective:**
To assess the time taken by backend triggers—especially those that validate tenant-property relationships.

🛠 **Actions Taken:**

- Created tenant records with and without conflicting property assignments.

- Measured execution speed of triggers preventing duplicate tenant assignments.

📈 **Result:**

- Apex Triggers executed within **~0.5 seconds per record**.
- No performance degradation during bulk testing (up to 25 records tested).
- All custom logic performed without errors or timeouts.

✅ 3. Email Automation Timeliness

🔍 **Objective:**
Verify timing of automated emails like lease approvals, rent reminders, and payment confirmations.

🛠 **Actions Taken:**

- Submitted lease records to trigger email alerts.
- Scheduled rent reminders using Apex classes.

📈 **Result:**

- All email notifications sent within **2-5 seconds** of trigger conditions.
- No email failures or missed templates observed.

✅ 4. Validation Rule Performance

🔍 **Objective:**
Ensure validation logic does not affect save time or user flow.

🛠 **Actions Taken:**

- Tested rule execution by entering invalid lease dates and missing fields.
- Evaluated system response time when error messages were triggered.

📈 **Result:**

- Validation rules executed **instantly (~<1 sec)**.
- Clear error messages were displayed without interrupting UI flow.

Performance Testing document

Would you like this exported as a **PDF**, included in a **final project report**, or turned into a **performance metrics chart or dashboard**?

Ask ChatGPT

You said:

performance sumarry recomandations

ChatGPT said:

Here is a **Performance Summary with Recommendations** for your **Lease Management Project**, summarizing test results and providing suggestions for optimization and future readiness.

🚀 Performance Summary & Recommendations – Lease Management System

---

✅ Performance Summary

Test Area	**Outcome**
Object Load Time	Loaded within **< 1 second** for all custom objects; smooth navigation.
Trigger Execution	Apex triggers executed in **~0.5 seconds** per operation; efficient logic.
Email Automation	Notifications sent within **2–5 seconds**; no delays or template errors.
Validation Rules	Instant error checking and response; no UI performance issues observed.
Overall Stability	The system remained responsive and accurate under moderate data load.

🔍 Recommendations for Improvement

1. 🔧 **Optimize Apex Code for Scale**
    - As the data grows, consider **bulkification of triggers** and use of **Apex limits logging** to track governor limit usage.
    - Ensure all triggers and handlers follow best practices for reusability and scalability.

Performance Testing document

2. 📊 **Implement Load Testing for Large Datasets**
    - Simulate usage with **hundreds or thousands of tenant/property records** to validate performance in real-world scenarios.
    - Use Salesforce Developer Console or tools like **JMeter** for stress testing.
3. 📤 **Improve Email Queue Monitoring**
    - Implement logic to **log or track failed email sends** for diagnostics.
    - Add retry mechanisms or alerts for email failures.
4. 📅 **Schedule Batch Jobs Smartly**
    - If you introduce high-frequency scheduled jobs (e.g., daily rent checks), use **batch Apex** and stagger scheduling to avoid processing conflicts.
5. 📱 **Prepare for Mobile and External Users**
    - Future features like mobile apps or tenant portals may increase concurrent load. Plan for **API limits**, **external access control**, and **session management**.
6. 📁 **Archive Old Data Periodically**
    - Set up a policy for **archiving old lease and payment records** to maintain performance and avoid clutter in reports and dashboards.
7. 🛡️ **Enable Performance Monitoring Tools**
    - Use Salesforce features like **Event Monitoring**, **Debug Logs**, and **Health Check Reports** to continuously observe performance and security posture.

Conclusion :

The **Lease Management System** demonstrated high performance and stability across all tested scenarios. From UI navigation to Apex automation and validation logic, every layer of the application responded within acceptable timeframes—making it suitable for deployment in small to medium-sized property management environments.