**Solution Architecture – Lease Management System**

**📅 Date:** 28 june 2025  
**📌 Team ID:** LTVIP2025TMID30244  
**📂 Project Name:** Lease Management System  
**🎯 Maximum Marks:** 4 Marks

**📌 Purpose of the Solution Architecture**

The architecture is designed to:

* Solve key business problems in lease administration, rent collection, and tenant communication.
* Define the technical structure and flow of data across all modules.
* Ensure scalability, availability, security, and performance of the application.
* Guide the development team through clearly defined components and services.

**🧱 Architecture Overview**

The Lease Management System follows a **3-tier architecture with microservices**, comprising the following layers:

**1. Presentation Layer**

* **Frontend (Web & Mobile):** React.js (Web), Flutter (Mobile)
* Role-based interfaces for Admin, Manager, and Tenant
* Responsive UI, secured by token-based authentication (JWT)

**2. Application Layer**

* **API Gateway:** Manages all incoming requests and routes them to services
* **Microservices Architecture:** Each function (e.g., Lease, Payments, Notifications) is a separate service built in Node.js/Python
* **Authentication & Authorization:** OAuth 2.0, JWT-based user session management

**3. Data Layer**

* **Database:** PostgreSQL for transactional data (leases, users, rent)
* **Cloud Storage:** AWS S3 / Google Cloud Storage for documents and receipts
* **Caching:** Redis to improve performance for frequently accessed data
* **Analytics Engine:** Generates reports and insights from usage and payment patterns

**🔁 Data Flow and Component Interaction**

less

CopyEdit

[User (Web/Mobile)]

↓

[React/Flutter Frontend]

↓

[API Gateway (Express.js)]

↓

┌───────────────────────────────┐

| Microservices (Node/Python) |

└───────────────────────────────┘

↓ ↓ ↓

[PostgreSQL] [Redis] [Cloud Storage]

↓ ↓

[Analytics/Reporting Engine] [Document Retrieval]

**🧩 Core Modules and Responsibilities**

| **Module** | **Description** |
| --- | --- |
| **User Management** | Registration, Login, Role Assignment |
| **Lease Module** | Add/Edit/Terminate lease, view timeline, attach documents |
| **Rent Management** | Set due dates, generate receipts, payment gateway integration |
| **Notification Service** | Sends reminders via email/SMS using Twilio or SendGrid |
| **Maintenance Module** | Raise and track service requests |
| **Analytics & Reports** | View lease trends, upcoming expiries, rent status |

**☁️ Infrastructure & Deployment**

* **Containerization:** Docker for all microservices
* **Orchestration:** Kubernetes (AWS EKS / Google GKE) for scaling
* **CI/CD Pipeline:** GitHub Actions or Jenkins for automated builds and deployment
* **Monitoring:** Prometheus + Grafana for system health and performance
* **Security:** HTTPS, IAM roles, database encryption (AES-256), firewall rules
* **Backup:** Daily backups to encrypted cloud storage

**📈 Development Phases**

| **Phase** | **Feature Set** |
| --- | --- |
| 1 | User Auth, Dashboard, Lease CRUD |
| 2 | Rent Module, Document Upload, Notifications |
| 3 | Tenant Portal, Reports, Maintenance Requests |
| 4 | Mobile App, Analytics Engine, ML Recommendations |

**✅ Benefits of the Architecture**

* **Scalable:** Easily supports hundreds of properties and users
* **Modular:** Each feature is independently developed and maintained
* **Secure:** Industry-grade data protection and user access control
* **Cloud-native:** Available 24/7 with global access
* **Future-Ready:** Compatible with AI/ML add-ons for predictive insights