Project Design Phase

Solution Architecture

Date	**01 NOV 2025**
Team ID	**NM2025TMID02942**
Title	**Medical Inventory System**
Maximum Marks	**4 Marks**

1. Overview

The **Medical Inventory Management System (MIMS)** is designed as a **modular, multi-tier architecture** that provides scalability, security, and real-time performance.
It connects various stakeholders (pharmacists, nurses, administrators, and suppliers) through a centralized digital platform for efficient tracking and management of medical inventory.

---

2. Architectural Goals

- Ensure **real-time visibility** of stock levels and transactions.
- Maintain **data accuracy** and **integrity** across all modules.
- Support **secure, role-based access** and **data protection**.
- Enable **easy scalability** for future expansion (multiple hospitals, departments).
- Facilitate **integration** with external hospital systems or supplier APIs.

---

3. System Architecture Overview

The system follows a **4-tier architecture**:

1. **Presentation Layer (Frontend)**
2. **Application / Business Logic Layer (Backend)**

3. **Database Layer (Data Storage)**
4. **Integration Layer (External Interfaces & APIs)**

---

4. Architecture Components

1. Presentation Layer (Frontend UI)

- **Purpose:** Provides a user-friendly interface for all users to interact with the system.
- **Users:** Pharmacists, Nurses, Procurement Officers, Administrators, Suppliers.
- **Key Features:**
    - Dashboard for stock overview and alerts
    - Forms for adding and updating inventory items
    - Role-based login interface
    - Reports and analytics visualization

Technologies (Example):

- HTML5, CSS3, JavaScript
- ReactJS / Angular / Vue.js for dynamic UI
- Bootstrap for responsive design

---

2. Application Layer (Backend / Business Logic)

- **Purpose:** Processes all inventory operations and enforces business rules.
- **Key Responsibilities:**
    - Handle requests from the frontend (e.g., add item, issue stock, generate report).
    - Manage authentication and user sessions.
    - Generate alerts (low stock, expiry notifications).
    - Process supplier orders and update status.

Technologies (Example):

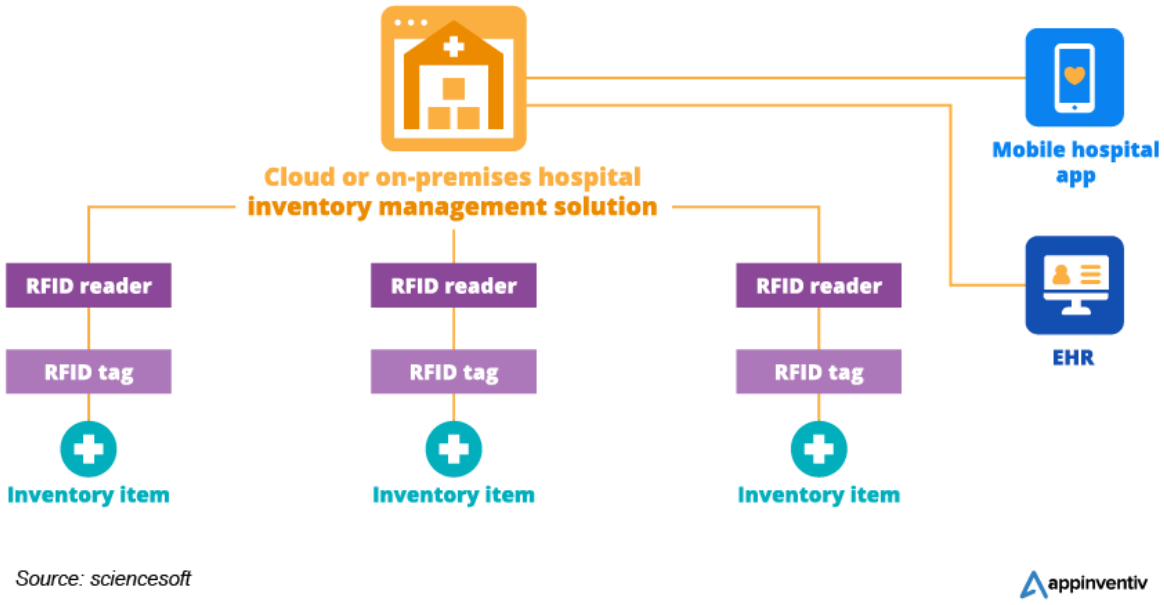
- Python (Django / Flask) or Node.js (Express)
- RESTful API design for modular communication
- Authentication: JWT / OAuth 2.0
- Scheduler for automated alerting and backups

---

3. Database Layer

- **Purpose:** Stores and manages all system data securely and consistently.
- **Data Entities:**
  - Users (Admins, Pharmacists, etc.)
  - Medical Items (Name, Category, Batch, Expiry, Quantity)
  - Transactions (Purchase, Issue, Return, Disposal)
  - Suppliers and Purchase Orders
  - Logs and Audit Trails

Example:



Source: sciencesoft

appinventiv