Project Design Phase

Solution Architecture

Date: 10 November 2025

Project Name: Medical Inventory Management on Salesforce

Max Marks / Priority: High (mission-critical)

Goals of the Architecture

Ensure accurate, real-time inventory levels for medical supplies across locations.

Prevent data loss / accidental deletion of critical inventory or product records.

Support automated reorder, stock movements, expirations and recalls.

Provide auditable history and role-based access for compliance (HIPAA / regulatory where applicable).

Integrate with external procurement/ERP systems and barcode scanners.

Key Components

Salesforce Data Model (Custom Objects & Key Fields)

Product__c

Name, SKU__c, Category__c, Unit_of_Measure__c, Active__c

Inventory_Location__c

Name, Location_Type__c (Warehouse, Clinic), Address__c

Inventory_Balance__c

Product__c (lookup), Location__c (lookup), Quantity__c (number), Reorder_Level__c, Expiry_Date__c, Batch__c

Stock_Movement__c

Movement_Type__c (Receive, Issue, Transfer, Adjustment), Product__c, From_Location__c, To_Location__c, Quantity__c, Performed_By__c, Related_PO__c, Related_Consumption__c, Date__c

Purchase_Order__c (or integrate with ERP)

Supplier__c, Line_Items (child), Status__c, Expected_Delivery__c

Supplier__c

Inventory_Incident__c (for damaged/expired items)

Audit_Log__c (or use Field History / Shield Event Monitoring for compliance)

Automation & Logic

Flows (Record-Triggered Flows) for:

Updating Inventory_Balance__c when Stock_Movement__c is created.

Auto-create Reorder Tasks when Quantity <= Reorder_Level__c.

Scheduled Apex for:

Daily expiry checks & notifications.

Reconciliation jobs with external systems.

Apex Triggers/Handlers for complex validations or high-performance bulk updates.

Platform Events for real-time integration (incoming shipments, scanner events).

Einstein/Reports & Dashboards for stock, usage trends, critical items.

Integrations

Inbound / Outbound APIs:

REST API endpoints for barcode scanners and mobile apps to call (create Stock_Movement__c).

Middleware (MuleSoft / Dell Boomi / custom) to sync Purchase Orders, Suppliers and Inventory with ERP.

Third-Party:

Barcode scanners or mobile apps (offline sync capability).

Optional HL7 adapters if integrating with clinical systems (use middleware).

UI / UX

Lightning Record Pages for Product / Inventory with related lists and quick actions (Receive, Transfer, Issue).

Lightning Web Components (LWC) for scanning UI and quick stock adjustment screens.

Mobile: Salesforce Mobile or a lightweight mobile app that calls Salesforce REST.

Data Protection & Compliance

Field-level encryption for sensitive fields.

Sharing model: private for inventory with role hierarchy and sharing rules for visibility by facility.

Audit: Field History Tracking (key fields) + Event Monitoring (for high-security orgs).

Development Phases

1. Design & Data Model — create objects, fields, and relationships.

2. Basic Automation — flows to maintain balances and reorder alerts.

3. Integrations — create REST resources & middleware mappings for ERP/POs.

4. Mobile & Scanner — implement LWC/mobile endpoints for barcode scanning and offline sync.

5. Compliance & Security — implement sharing, permissions, encryption, audit logging.

6. Testing — unit tests, UAT with clinics, performance testing for bulk loads.

7. Go-Live & Monitoring — monitor jobs, dashboards, error handling, operational runbook.

Solution Architecture Description (short)

The system uses custom Salesforce objects to model products, inventory locations, and balances. All physical movements are captured as Stock_Movement__c records; creation of those records updates Inventory_Balance__c through declarative Flows or Apex to guarantee immediate, consistent balance changes. Barcode/mobile clients create stock movements through secure REST endpoints or Platform Events for near real-time updates. External ERP systems exchange purchase orders and receipts using middleware to ensure reconciled inventory across systems. Role-based access and audit logging ensure regulatory accountability. Scheduled reconciliation jobs detect mismatches and

expirations; alerts and dashboards keep operators informed of low stock or expired supplies.

Example — Textual Architecture

User / Scanner (mobile LWC)

→ Salesforce REST / Platform Event → Stock_Movement__c (Receive/Issue)

→ Record-Triggered Flow or Apex Handler

→ update Inventory_Balance__c (increment/decrement)

→ If Quantity__c <= Reorder_Level__c → create Reorder Task / notify Procurement

→ Middleware sync with ERP for POs and receipts (bi-directional)

→ Dashboards & Reports (Inventory Health, Expiry, Movements)

(Background: Scheduled Apex for expiry checks & reconciliation; Shield/Event logs for audit)

Prevent accidental deletion (pattern)

Use Validation (before delete) or Apex trigger to prevent deletion of Product__c or Inventory_Location__c if related Inventory_Balance__c / Stock_Movement__c exist.

Alternatively use a soft-delete pattern (set Active__c = false) and prevent physical deletion via profiles.

Sample Apex trigger (prevent Product deletion if inventory exists)

```
trigger PreventProductDelete on Product__c (before delete) {
    Set<Id> prodIds = new Set<Id>();
```

```
    for (Product__c p : Trigger.old) prodIds.add(p.Id);
    Integer relatedCount = [SELECT COUNT() FROM Inventory_Balance__c WHERE
Product__c IN :prodIds];
    if (relatedCount > 0) {
        for (Product__c p : Trigger.old) {
            // addError prevents delete and surfaces message in UI/API
            p.addError('Cannot delete product. There are inventory balances or transactions
associated.');
        }
    }
}
```

> Note: For bulk safety change COUNT query to map counts by ID; include unit tests (≥75% coverage) and bulkified handling.

Monitoring, Logging & Error Handling

Use Platform Events + Dead Letter Queue for integration errors.

Scheduled reconciliation jobs to surface discrepancies.

Alerts via Chatter/email for critical low stock or failed integrations.

Use Data Loader + Mass Update jobs for migrations; keep backups.

Non-Functional Considerations

Performance: bulkify triggers, avoid synchronous external calls in transactions. Use Queueable / Platform Events for heavy processing.

Scalability: partition inventory by location and archive old batch/expired entries.

Availability: configure retry logic for middleware; design idempotent