Project Design Phase

Solution Architecture

Date	06 November 2025
Team ID	F529277D47452DB9E7447BD087E08E5E
Project Name	Medical Inventory Management
Maximum Marks	4 Marks

Goals & requirements (short)

- Accurate real-time stock levels for medicines & supplies (batch, expiry, lot, serial).
- Automated reorder (min/max, EOQ, safety stock) + vendor integration.
- Traceability: receiving → storage → dispensing → disposal (audit trail).
- Integration with EHR/EMR, procurement, billing, and lab systems.
- Support barcode / RFID / IoT sensors (temperature, humidity).
- Secure + compliant (HIPAA/GDPR where applicable), role-based access.
- Scalable, highly-available, observable.

High-level architecture (summary)

1. Client layer: Web UI (inventory managers, pharmacists), Mobile apps (scanning), Kiosk/terminal UI.
2. API layer: Gateway + REST/GraphQL services.
3. Core services: Inventory service, Ordering/Replenishment service, Lot/Expiry service, Audit service, Supplier service.
4. Data layer: Relational DB for transactional data, Time-series DB for sensors, Data warehouse for analytics.
5. Integration & messaging: Event bus (Kafka/RabbitMQ), API adapters for EHR / suppliers, HL7/FHIR adapter.
6. Physical devices: Barcode/RFID readers, IoT sensors, smart shelves.
7. Cross-cutting: Auth/ZTNA, encryption, logging, monitoring, CI/CD.

(Architecture mapped to cloud on AWS/Azure/GCP is included below.)

Component breakdown

1. Client layer

- Web app (React/TS) for inventory management dashboards, inbound/outbound workflows.
- Mobile app (iOS/Android or PWA) for scanning barcodes/RFID and offline mode.
- Handheld scanners / kiosks — integrate via Bluetooth/USB or MQTT gateway.

2. API Gateway & Edge

- API Gateway (Auth enforcement, rate limiting).
- Protocols: REST for operational APIs, GraphQL for the UI if desired, gRPC for internal microservices.

3. Core microservices

- **Inventory Service** — core ledger of stock transactions (receipts, issues, transfers). Keeps per-location, per-lot, expiry, and status.
- **Lot & Expiry Service** — tracks lot numbers, expiry, recalls.
- **Replenishment / Procurement Service** — reorder rules (min/max, periodic review), PO creation, supplier selection.
- **Receiving Service** — checks inbound shipments against POs, QC, quarantine.
- **Dispense Service** — fulfill requests (ward, patient), supports barcode scanning for 2FA.
- **Audit & Compliance Service** — immutable log of all transactions (append-only).
- **Inventory Forecasting / ML** — demand forecasting and expiry risk; uses time-series and historical usage.
- **Integration Service / EHR Adapter** — transforms to/from HL7 / FHIR, syncs orders / patient medication needs.
- **Notification Service** — low stock alerts, expiry alerts, recall notifications (email/SMS/Push).

4. Data stores

- **Primary DB**: PostgreSQL/Cloud SQL / Azure SQL — ACID for transactions.
- **Cache**: Redis for hot lookups and locks.
- **Event Store / Stream**: Kafka or managed pub/sub for events (stock changed, PO created).
- **Timeseries DB**: InfluxDB / Prometheus / AWS Timestream for sensor data.
- **Data Warehouse**: Snowflake / BigQuery / Redshift for BI.
- **Object storage**: S3/GCS/Azure Blob for attachments, invoices, certificates.

5. Device & Edge connectivity

- Local gateway for barcode/RFID → forwards events to API (MQTT/HTTPS).

- IoT sensors stream to edge gateway → time-series DB / monitoring.

Data model (simplified ER)

- Item (item_id, sku, name, description, uom, category)
- Location (location_id, name, type:warehouse/ward/fridge)
- Lot (lot_id, item_id, lot_number, manufacture_date, expiry_date, quantity, status)
- StockTransaction (tx_id, item_id, lot_id, from_location, to_location, qty, tx_type, timestamp, user_id, reference_id)
- PurchaseOrder (po_id, supplier_id, items[], status)
- Supplier (supplier_id, name, contact)
- ReorderRule (item_id, location_id, min, max, reorder_qty, rule_type)
- AuditLog (log_id, entity, action, prev_state, new_state, user, timestamp)

Key flows (sequence)

Receiving & Putaway

1. Supplier sends shipment → Warehouse receiving clerk scans shipment.
2. System matches against PO (Receiving Service).
3. If mismatch → create exception task. If OK → create Lot record and StockTransaction (RECEIPT).
4. Putaway: move to location (shelf, cold chain) → StockTransaction (TRANSFER).

Dispensing to Ward/Patient

1. Ward places requisition (EHR or Inventory UI).
2. Dispense Service confirms availability by lot/expiry and reserves stock.
3. Pharmacist scans item & patient ID (2-factor verification), system records StockTransaction (ISSUE).
4. If partial, system updates lot quantities and triggers reorder if below threshold.

Replenishment

- Replenishment service periodically evaluates ReorderRules and forecasts. Generates PO and notifies procurement. Events published on event bus.

APIs (example endpoints)

- `POST /api/v1/receive` — submit received shipment (body: po_id, items[{sku, lot, qty}])
- `POST /api/v1/stock/transfer` — move stock between locations
- `POST /api/v1/dispense` — issue item to ward/patient (requires auth & patient_id)
- `GET /api/v1/stock?item_id=&location_id=` — current stock snapshot
- `GET /api/v1/lot/{lot_id}` — lot details (expiry, quantity)
- `POST /api/v1/reorder/trigger` — force reorder for item/location
- FHIR endpoints for medication supply: `POST /fhir/MedicationRequest` (adapter transforms)

All endpoints behind API Gateway with JWT/OAuth2 scopes.

Events / Topics (event-driven)

- `stock.received`
- `stock.issued`
- `stock.transferred`
- `stock.threshold_breach`
- `po.created`
- `lot.expiry_near` (7/30/90 days)
- `recall.issued`

Consumers: Analytics, Notification Service, External Integrations, Audit Service.

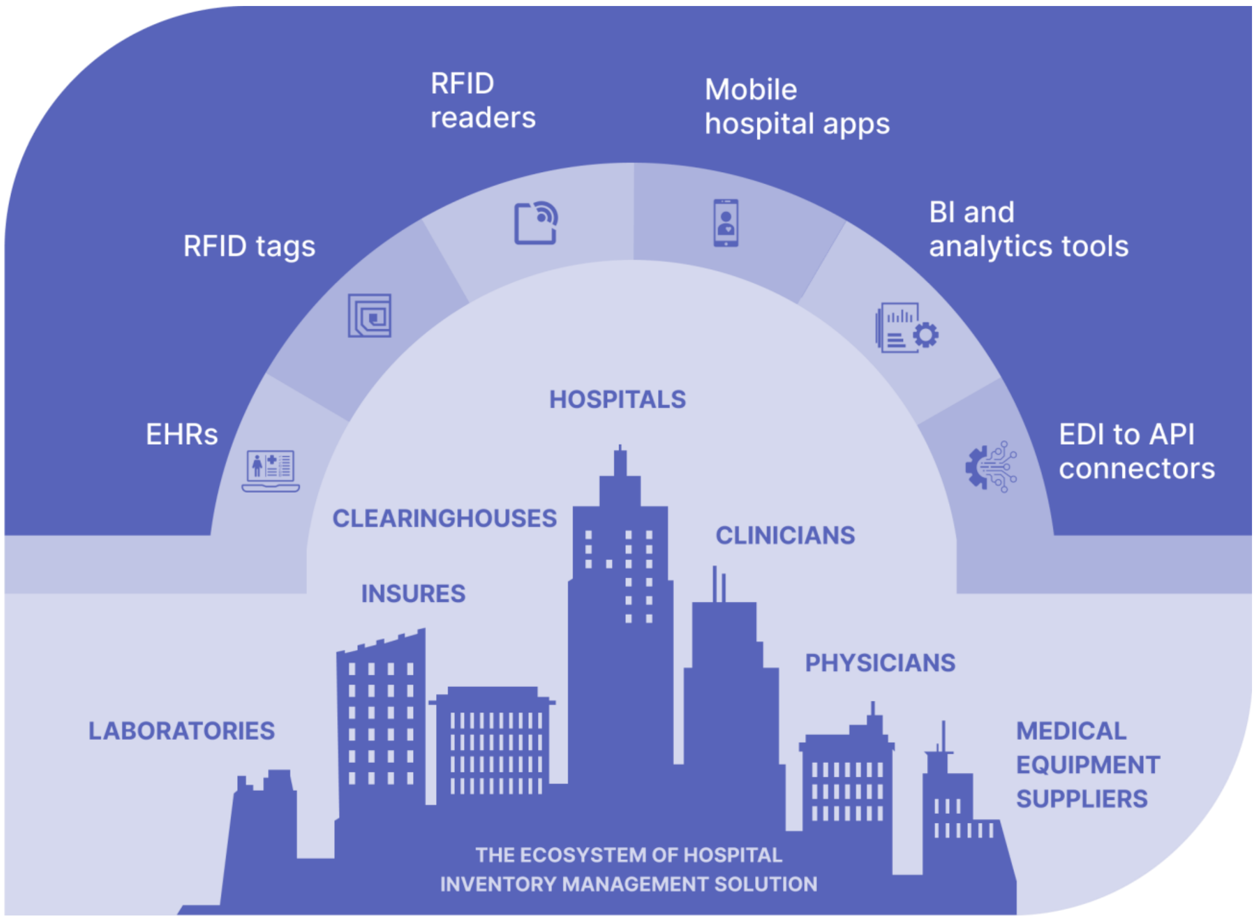
Security & Compliance

- Authentication: OAuth2 / OpenID Connect (Keycloak / Auth0 / AWS Cognito).
- Authorization: RBAC + attribute-based access (ABAC) for sensitive operations (dispense, write audits).
- Encryption: TLS in transit; database encryption at rest; HSM/KMS for key management.
- Auditability: Append-only audit logs with tamper-evident signatures (hash chaining).
- Data minimization: patient PHI separated; pseudonymize where possible.
- Access logging & SIEM integration (CloudWatch/ElasticSIEM/Splunk).
- Compliance checklist: HIPAA business associate agreements, data residency rules, retention & deletion policy.
- Physical security for cold-chain: signed logs + IoT sensor tamper detection.

Reliability, scaling & HA

- Stateless API services in Kubernetes with Horizontal Pod Autoscaling.

- PostgreSQL in multi-AZ with read replicas and automated failover (Patroni/RDS Multi-AZ).
- Kafka / managed pubsub for decoupling; compacted topics for stateful events.
- Redis cluster for caching & distributed locks (to prevent race conditions on inventory updates).
- Use optimistic concurrency control for stock updates plus idempotent operations (idempotency keys).
- Backup strategy: DB snapshots daily + WAL archiving; test restore monthly.
- Disaster recovery: RTO goal (e.g., 1-4 hours), RPO (e.g., 1 hour) depending on hospital SLAs.


RFID readers
Mobile hospital apps
RFID tags
BI and analytics tools
EHRs
EDI to API connectors
HOSPITALS
CLEARINGHOUSES
CLINICIANS
INSURES
PHYSICIANS
LABORATORIES
MEDICAL EQUIPMENT SUPPLIERS
THE ECOSYSTEM OF HOSPITAL INVENTORY MANAGEMENT SOLUTION