
1️⃣ High-level architecture

Style: Modular, service-oriented (can be microservices or well-separated modules in a monolith).

Core layers:

- **Device & edge integration layer**
- **Core domain services**
- **Biometric & analytics services**
- **Admin & configuration services**
- **APIs & integration layer**
- **Infrastructure & cross-cutting services**

2️⃣ Core services and responsibilities

A. Device gateway service

Purpose: Single entry point for all hardware (tags, readers, gates, alarms, scanners).

- **Protocols supported:**
  - MQTT over TLS
  - HTTPS (REST/Webhook)
  - WebSocket (optional)
- **Responsibilities:**
  - Authenticate devices (certificates, tokens).
  - Normalize incoming messages:
    - Tag sightings
    - Tamper events
    - Gate scan events
    - Alarm status
  - Route messages to:
    - RTLS service
    - Event processing service
    - Audit log service

B. RTLS & location service

Purpose: Maintain real-time view of infant/mother tag locations.

- **Inputs:**
  - Tag sightings from ceiling readers (tag ID, reader ID, RSSI, timestamp).
- **Responsibilities:**

    - Map reader ID → zone/room.
    - Maintain in-memory state:
        - `tag_id → current_zone, last_seen_at, signal_strength`.
    - Generate events:
        - Zone enter/exit
        - Tag missing (not seen for X seconds)
    - Provide APIs:
        - `GET /location/tag/{id}`
        - `GET /location/infant/{infantId}`
        - `GET /location/zone/{zoneId}`

C. Infant & mother management service

Purpose: Master data for infants, mothers, and their relationships.

- **Entities:**
    - Infant
    - Mother
    - InfantTag
    - MotherTag
    - PairingRecord
- **Responsibilities:**
    - Register infant and mother.
    - Assign tags.
    - Pair infant ↔ mother.
    - Validate pairing on request (e.g., from gate service).
    - Expose APIs:
        - `POST /infants`
        - `POST /mothers`
        - `POST /pairings`
        - `GET /pairings/{infantId}`

D. Gate authorization service

Purpose: Decide if a movement through a gate is allowed.

- **Inputs:**
    - From gate terminal:
        - Infant tag ID
        - Mother tag ID (or staff ID)
        - Gate ID
        - Reason code
- **Logic:**
    - Validate:
        - Infant tag is active and assigned.
        - Mother tag matches paired mother for that infant (if mother present).
        - Staff ID is valid and authorized for that action.

        - Gate is allowed for that movement type.
    - Check infant’s current zone vs allowed path.
    - Decision:
        - Authorized → return OK, log movement.
        - Denied → return error, trigger potential alert.
- **Outputs:**
    - MovementLog entry.
    - Optional event to alarm service if suspicious.
- **APIs:**
    - `POST /gate/authorizeMovement`
    - `GET /gate/movements?filters…`

E. Tamper & security event service

Purpose: Central brain for security-critical events.

- **Inputs:**
    - Tamper events from tags.
    - Unauthorized movement attempts.
    - Zone violations (infant near exit without authorization).
    - Device health anomalies.
- **Responsibilities:**
    - Classify events (INFO/WARN/CRITICAL).
    - Trigger:
        - Alarm controller commands.
        - Notifications to nurse/security dashboards.
    - Maintain event history.
- **APIs:**
    - `GET /events?type=…`
    - `POST /events/manualTrigger` (for manual alarms)

F. Alarm orchestration service

Purpose: Coordinate sirens, strobes, and alarm states.

- **Inputs:**
    - Commands from tamper & event service.
    - Manual triggers from UI.
- **Responsibilities:**
    - Maintain alarm state per zone/hospital:
        - `IDLE`, `ACTIVE`, `SILENCED`, `RESET_PENDING`.
    - Send commands to alarm controller nodes:
        - `RAISE_ALARM`, `SILENCE`, `TEST`, `RESET`.
    - Log all alarm actions.
- **APIs:**
    - `POST /alarms/raise`
    - `POST /alarms/silence`

    - `GET /alarms/status`

G. Biometric service

Purpose: Handle footprint templates and matching.

- **Inputs:**
    - From footprint scanner:
        - Infant ID
        - Biometric template (or image)
- **Responsibilities:**
    - Enrollment:
        - Store template linked to infant.
        - Check for duplicates (de-duplication).
    - Verification:
        - Compare new template against stored one(s).
    - Provide:
        - `POST /biometric/enrollInfant`
        - `POST /biometric/verifyInfant`
        - `POST /biometric/checkDuplicate`
- **Implementation:**
    - Can call external biometric engine or internal library.
    - Store templates in secure DB or dedicated store.

H. User & role management service

Purpose: Manage staff accounts, roles, permissions.

- **Entities:**
    - User (nurse, doctor, security, admin)
    - Role (NURSE, SECURITY, ADMIN, BIOMED, IT)
    - Permissions (view, control, configure)
- **Responsibilities:**
    - Authentication (integrate with hospital SSO/LDAP if needed).
    - Authorization (RBAC).
    - Audit of logins and critical actions.
- **APIs:**
    - `POST /auth/login`
    - `GET /users`
    - `POST /roles`

I. Configuration & asset management service

Purpose: Manage static and semi-static configuration.

- **Entities:**
    - Hospital
    - Ward
    - Zone
    - Gate
    - Reader
    - Device (tag, scanner, alarm node)
- **Responsibilities:**
    - Store topology:
        - Which readers belong to which zones.
        - Which gates connect which zones.
    - Device registry:
        - Device ID, type, firmware version, status.
    - Provide configuration to devices (via gateway).
- **APIs:**
    - `GET /config/zones`
    - `GET /config/devices`
    - `POST /config/devices`

J. Audit & logging service

Purpose: Immutable record of all critical actions.

- **Logs:**
    - Movements
    - Pairings
    - Alarms
    - Logins
    - Config changes
    - Device events
- **Storage:**
    - Append-only log store (e.g., separate DB or log system).
    - Retention policies per regulation.
- **APIs:**
    - `GET /audit?filters…`

3️⃣ API gateway & external integration

API gateway:

- Single entry for:
    - Web dashboards
    - Mobile apps
    - Hospital HIS/EMR integration
- Responsibilities:
    - Authentication/authorization.

    - Rate limiting.
    - Routing to internal services.
    - API versioning.

External integration:

- **HIS/EMR:**
    - Sync patient/mother demographics.
    - Optional: admit/discharge events.
- **Alarm systems:**
    - Integrate with existing hospital alarm/notification systems (e.g., nurse call).

4️⃣ Data flow examples

A. Infant tamper event

1. Infant tag detects tamper → sends event via reader/gateway.
2. Device gateway receives → forwards to tamper & security event service.
3. Event service:
    - Creates security event.
    - Notifies alarm orchestration service.
4. Alarm service:
    - Sends `RAISE_ALARM` to alarm controller node.
    - Notifies dashboards via WebSocket.
5. Audit service logs all steps.

B. Infant movement through gate

1. Gate terminal scans infant tag, mother tag, staff ID.
2. Gate terminal calls `POST /gate/authorizeMovement`.
3. Gate authorization service:
    - Validates pairing, staff, gate, zones.
    - Logs movement.
    - Returns `AUTHORIZED` or `DENIED`.
4. Gate terminal:
    - Shows result.
    - If denied and suspicious → event sent to tamper & security event service.

5️⃣ Technology stack suggestions

- **Language:** Java/Kotlin, C#, Go, or Node.js—whatever your team prefers.
- **API:** REST + WebSocket; optionally GraphQL for dashboards.

- **Messaging:** MQTT (devices), internal message bus (e.g., Kafka/RabbitMQ) for events.
- **DB:** PostgreSQL for core data; Redis for real-time state; object store for logs/images.
- **Deployment:** Kubernetes or VM-based, on-prem (hospital data center) or private cloud.
