MDR Contact Tracing & Screening – Backend Documentation

The MDR Contact Tracing & Screening Backend System is designed to digitally track patient movements, contact events, shared equipment usage, and clinical parameters within a hospital environment. The system computes real-time Multidrug Resistant (MDR) pathogen risk scores completely in an offline-first environment using SQLite.

This backend ensures high reliability in low-connectivity environments, making it suitable for deployment in hospitals across Android, Windows, Linux, and macOS platform**s**. Cloud sync can be enabled later without altering the core architecture.

Flutter + SQLite (Offline-First Architecture)

This backend powers the **Digital Contact Tracing and MDR Screening System**, enabling hospitals to track patient movements, clinical parameters, exposure events, and compute multidrug-resistant pathogen risk scores in real time — entirely **offline** on Android/Windows/Linux.

- **Features**
  1. Offline-first backend using SQLite.
  2. QR-based patient identification.
  3. Logging of:
     - Patient details & MDR pathogen info
     - Movements across hospital wards
     - Equipment usage (shared devices)
     - Patient–patient contact events
     - Clinical observations
  4. MDR Risk Engine:
  5. Outbreak alert scanning (doctor dashboard).
  6. Patient-level risk overview.
  7. Prepared for ONNX-based machine learning (future upgrade).

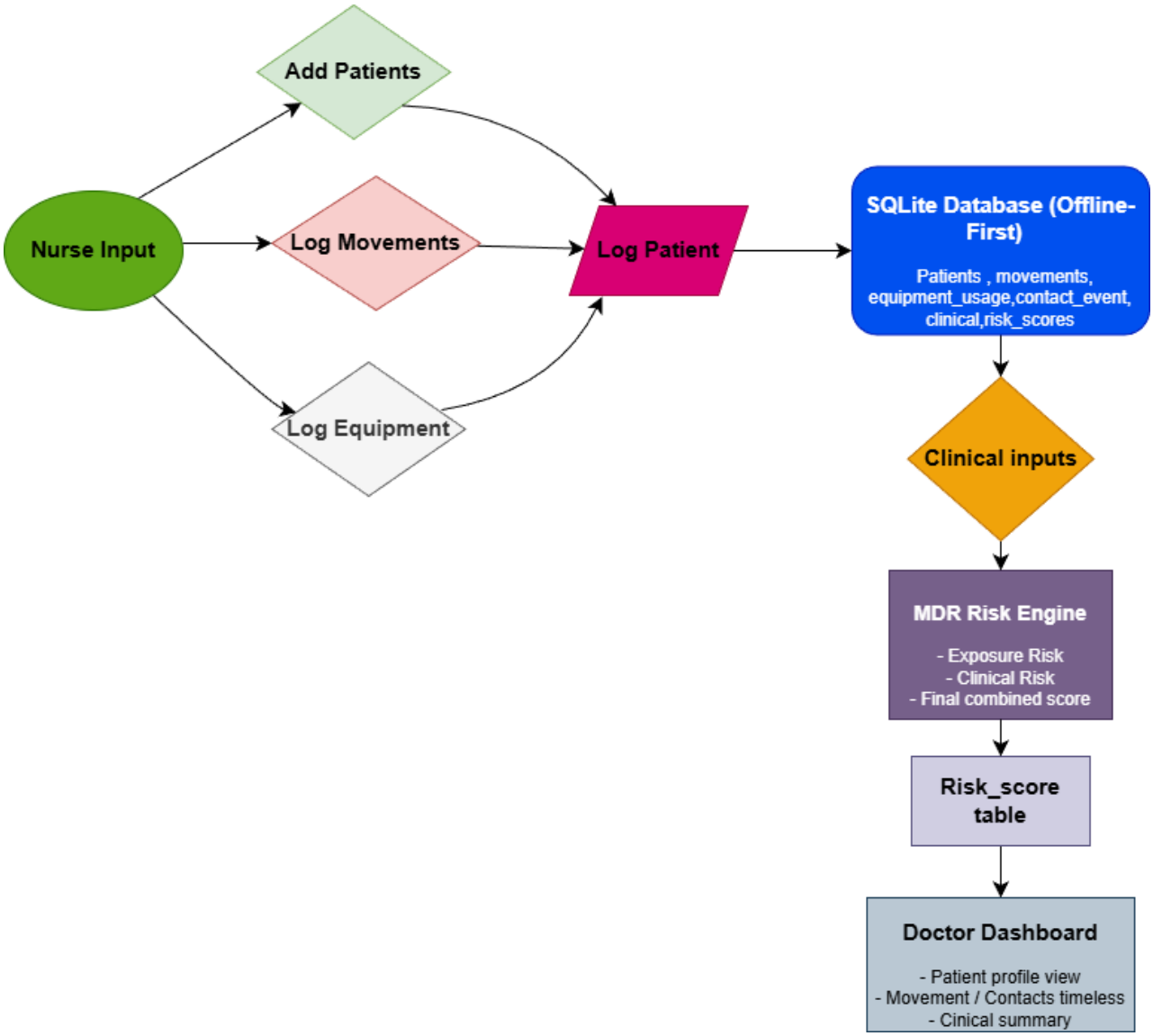
- **Backend Architecture**

  Lib/

  1.backend

  - db/ -> SQLite initialization & configuration
  - models/ -> Database table model
  - services/ -> Business logic & risk engines
  - utils/ -> Helper utilities & pathogen reference table

➢ **Flow Diagram**



This diagram shows the workflow of an offline MDR contact tracing and risk assessment system. Nurses add patient details, log their movements, and record shared equipment usage, and all this data is stored in a local SQLite database. Clinical inputs such as symptoms and test results are then processed by the MDR Risk Engine, which calculates exposure risk, clinical risk, and a final combined risk score. This score is saved in the risk score table and displayed on the doctor's dashboard to support quick medical decisions.

➢ **Database Overview**

The app uses **SQLite** (via sqflite / ffi_sqflite) with the following tables:

1. patients

emographics + MDR metadata
Fields:
id, name, age, ward, is_mdr_known, mdr_pathogen, mdr_syndrome, transmission_type, sync_status

2. movements

transitions
Fields:
patient_id, ward, time_in, time_out

3. equipment_usage

Logs shared device contact
Fields:
patient_id, equipment_name, timestamp, shared_with_others

4. contact_events

Logs direct patient–patient contact
Fields:
index_patient_id, contact_patient_id, start_time, end_time, location

5. clinical_observations

Stores clinical data used for ML risk
Fields:
temperature, comorbidities_score, previous_mdr_history, lab_mdr_positive, …

6. risk_scores

Stores output of the risk engine
Fields:
backend_exposure_risk, ml_clinical_risk, final_risk, explanation, calculated_at.

- **Backend Components**
  1. **Models (lib/backend/models/)**

     Each table has a model with:

     - Fields matching DB columns
     - toMap() for SQL inserts
     - fromMap() for reading records

     Models include:

     - Patient
     - Movement
     - EquipmentUsage
     - ContactEvent
     - ClinicalObservation
     - RiskScore

  2. **PatientService**

Used when nurses register new patients.

addPatient()

getPatientById()

getAllPatients()

3. **MovementService**

   Logs patient's movement across wards.

   addMovement()

   getMovementsForPatient()

4. **EquipmentService**

   Logs equipment usage.

   addUsage()

   getUsage()

5. **ContactService**

   Stores patient–patient contacts.

   addContactEvent()

   getContacts()

6. **ClinicalService**

   Saves clinical observations from nurses/doctors.

   insertObservation()

   getLatestForPatient()

- **MDR Risk Engine**

1. **Exposure Risk (Rule-Based)**

File: exposure_risk_engine.dart

2. **Clinical Risk (Logistic Model)**

   File: risk_service.dart

   Uses:

   - Age
   - Temperature
   - Comorbidity score
   - Previous MDR
   - Lab MDR+

- Total exposure duration (hours)

Formula is logistic-style (1 / (1 + exp(-z)))
Output: (0–1)

3. **Final Risk**

finalRisk = 0.7 * mlRisk + 0.3 * exposureRisk;

Weighted more towards clinical data.

- **ONNX Model Integration (Future)**

  File: mdr_onnx_model.dart

  Prepared to load:

  assets/models/mdr.onnx

  Supports 19 input features:

  timestamp,

  age,

  gender,

  ward,

  movement_path,

  equipment_used,

  sputum_result,

  The model is not yet active, but ready for drop-in replacement:

  final prob = await MdrOnnxModel().predictMdrProbability(inputs);

- **BackendTester**

  On app startup (non-web), it:

  - Inserts dummy patient
  - Adds dummy movements + contacts + equipment
  - Runs risk engine
  - Verifies DB integration

  This ensures the whole backend pipeline is working.

- **Data Flow**

  [NURSE]

Add Patient → PatientService

Log Movement → MovementService

Log Equipment → EquipmentService

Log Contact → ContactService

Add Clinical Data → ClinicalService

[DOCTOR]

Evaluate Risk → RiskService

Patient Overview → pulls all logs + risk

Outbreak Alerts → runs risk for all patients

Everything is stored locally in SQLite and can be synced to the cloud later.