# **Technical Brief: Cold Storage IoT Monitoring System**

## **1. Objective**

To build a robust system to capture, display, and analyze sensor data and equipment status from a cold storage facility. The system will include a real-time dashboard, a maintenance module, an alerting engine, and a reporting feature.

## **2. Data Points & Schema**

The following data points need to be captured and stored:

* **Cold Room Sensors:**
  + temperature (float, °C) - *Support for multiple sensors.*
  + relative\_humidity (float, %) - *Support for multiple sensors.*
* **Cooling System:**
  + compressor\_status (enum: ON, OFF, TRIPPED)
  + primary\_coolant\_pressure\_high (float, psi/bar)
  + primary\_coolant\_pressure\_low (float, psi/bar)
  + secondary\_coolant\_temp (float, °C)
  + secondary\_coolant\_level (float, %)
  + fan\_status (enum: ON, OFF, TRIPPED)
* **Defrosting System:**
  + heater\_status (enum: ON, OFF, TRIPPED)
  + heater\_temp (float, °C)
* **General Equipment:**
  + secondary\_coolant\_pump\_status (enum: ON, OFF, TRIPPED)
  + mains\_power\_status (enum: ON, OFF)
* **Door:**
  + door\_status (enum: OPEN, CLOSED)

## **3. Feature Breakdown & Logic**

### **Module 1: Dashboard & Real-time Monitoring (Frontend/Backend)**

* **UI:**
  + Display T & RH values. If multiple sensors exist, show an average and allow drill-down to individual sensor data. Use gauges or time-series graphs.
  + Utilize clear status indicators (e.g., green/red/orange toggles or badges) for all equipment statuses.
  + Implement a running counter for "Total Door Open Time" for the current 24-hour period (resets at midnight).
* **Backend:**
  + Develop a REST API endpoint (/api/v1/status/latest) to provide a single JSON object with the current state of all data points.
  + Implement WebSockets for pushing real-time data to the frontend to ensure the dashboard is live without constant polling.

### **Module 2: Maintenance & Equipment Logging (Backend/Database)**

* **Run Time Calculation:**
  + Log all equipment state changes to an EquipmentStatusLog table with equipment\_id, status, and timestamp.
  + A background job or trigger will calculate the duration between an ON and OFF/TRIPPED event and aggregate it into a TotalRunTime table (equipment\_id, total\_hours).
* **Servicing Logic:**
  + Create a ServiceRecords table: record\_id, equipment\_id, service\_date, notes, run\_time\_at\_service (float).
  + **"Run Time Since Last Service" Logic:** This is a computed value: CurrentTotalRunTime - run\_time\_at\_service from the latest service record for that equipment\_id.
  + **Maintenance Alert Logic:** Create a configurable rules engine. CREATE\_ALERT IF (NOW() - last\_service\_date) > X\_days OR (RunTimeSinceLastService > Y\_hours).
* **UI:**
  + A dedicated "Maintenance" section to view a list of all equipment.
  + Each item should display Total Run Time, Last Serviced Date, and Run Time Since Service.
  + Implement CRUD functionality for ServiceRecords via a form.

### **Module 3: Alerting Engine (Backend)**

* **Trigger Conditions:**
  1. **Equipment Trip/Breakdown:** When status changes to TRIPPED in EquipmentStatusLog, trigger an alert.
  2. **T & RH Thresholds:** On every new sensor reading, compare the value against configurable min\_threshold and max\_threshold values for that sensor type.
  3. **Door Open Threshold:** When door\_status changes to OPEN, start a timer. If the state remains OPEN for more than a configurable duration\_threshold\_seconds, trigger an alert. The timer must be reset upon a CLOSED event.
* **Implementation:**
  + Create an Alerts table: alert\_id, timestamp, type (e.g., TRIP, THRESHOLD), severity, message, is\_acknowledged (boolean).
  + Integrate with a notification service (e.g., SMTP for email, Twilio for SMS) to dispatch alerts.

### **Module 4: Records & Reporting (Backend/Frontend)**

* **UI:** A "Reports" section with date range selectors and dropdowns to filter by equipment or event type.
* **Backend API Endpoints:**
  + GET /api/v1/reports/runtime?equipment\_id=X&start\_date=Y&end\_date=Z
  + GET /api/v1/reports/events?type=[trip|threshold]&start\_date=Y&end\_date=Z
  + GET /api/v1/reports/servicing?equipment\_id=X
* **Functionality:** Allow exporting of query results as CSV. Data should be queried from the respective log tables (EquipmentStatusLog, Alerts, ServiceRecords).