Device Management Dashboard (Azure):

Dashboard Workflow (Step-by-Step)

1. **User opens the dashboard (App Service + React)**
    - Authenticated user (via Azure AD B2C) opens the dashboard.
2. **Frontend queries device list via API**
    - REST API (via Azure API Management) retrieves all devices from Azure IoT Hub or Cosmos DB.
3. **Metadata is pulled from Device Twins / DB**
    - Azure Function pulls the latest metadata: Device ID, current firmware version, geographic location, last active timestamp.
4. **Health & status shown using real-time metrics**
    - Device Twin properties (or reported telemetry) are used to show real-time indicators.
5. **Filtering/sorting handled client-side or via API**
    - User can filter by location, status, version; backend supports query parameters for efficiency.
6. **Real-time updates using WebSocket**
    - Changes to device health/status are pushed via WebSocket

Device Health Monitoring System (Azure):

Health Monitoring Workflow

1. Device sends telemetry every few seconds/minutes
    - Metrics include CPU usage, memory, battery level, signal strength via MQTT.
2. Azure IoT Hub receives and routes telemetry
    - Routes data to Stream Analytics or Azure Functions via Event Hub-compatible endpoints.
3. Health evaluator processes telemetry

    - Function logic compares metrics against thresholds and classifies device:
        - Healthy: All metrics within range.
        - Warning: One or more borderline.
        - Critical: Severe metric failure (e.g., battery < 10%).
4. Health status is updated in DB + Device Twin
    - Result is stored in Cosmos DB and/or set as a reported property in the device's twin.
5. Real-time updates pushed to dashboard using WebSocket
    - WebSocket connection pushes health changes live to the dashboard UI.
6. Fallback: polling via API

<u>OTA Updates</u>:

OTA Functional Flow:

1. Firmware Upload Interface

- Hosted in your existing App Service (React + API).
- File upload backed by Azure Blob Storage.
- Metadata (version, description, hash, upload time) stored in Azure SQL.

2. Device/Group Selection UI

- Admin chooses:
    - Individual devices.
    - A device group (stored in DB or as a tag/twin property).
- Selection saved to an update job table in Azure SQL.

3. Triggering the OTA Update

- Azure Function:
    - Validates if the device firmware version is outdated.

- Sends a Direct Method or Cloud-to-Device message to target device via IoT Hub:

4. Device-Side Logic

- Device:
  - Validates firmware (version & checksum).
  - Downloads from Blob URL using HTTPS.
  - Applies firmware and reboots.
  - Reports update status: Queued, In Progress, Completed, Failed.

5. Monitoring and Status Dashboard

- Device update status is sent via MQTT telemetry or Device Twin update.
- Status stored in Azure SQL.
- Displayed in dashboard (React UI): searchable, sortable by device/group.

Architecture for OTA:

Admin UI - Upload Firmware

Azure App Service (UI/API)

Azure Blob Storage ← Firmware binary (.bin, .hex, etc.)

Azure SQL Database (Firmware metadata table)

Admin selects devices/groups for update via UI

Azure Function: OTA Orchestrator

Azure IoT Hub Direct Method→ Sends "startUpdate" method to device

Device Downloads Firmware via |

Secure Blob URL (SAS token)

Device reports status via MQTT

Azure IoT Hub receives status

Azure Function / Stream Analytics
Updates Azure SQL with update state

RBAC:

User logs in → Azure AD B2C → receives JWT token with role claims →
Token is used in API calls → Azure APIM or Backend validates role before action

RBAC + Reporting Integration

- **Report Download API**: Only Admin and Viewer can call it.
- **Firmware Update API**: Only Admin allowed.
- **Device Logs View API**: Admin + Technician allowed.
- **Audit Logs**: Each action (even denied ones) logged for accountability.

Azure WorkFlow:

1. Devices are shipped with A unique identity. Ex: (TPM or X.509 certificate)
   - Devices identities are preloaded into Azure Device Provisioning Service (DPS). Either individually or Enrolment group

2. Automatic verification and secure connection
   - Connects securely to Azure DPS over MQTT or HTTPS
   - DPS verifies

3. Device onboarding into Azure IoT Hub
    - Device uses credentials from DPS to connect to Azure IoT Hub
    - Device sends meta data (device type, firmware version, etc.) and initial telemetry
    - Azure Functions or Logic apps to store metadata (in cosmos DB or SQL)

4. Device configuration and OTA setup
    - Device twin is updated with desired properties
    - Device syncs configuration and check for OTA update commands

5. Scale and Monitoring
    - Devices continuously send the data to IoT Hub.
    - Scaled via: IoT Hub auto-scaling
    - Message routing to Blob or Cosmos DB

Architecture Diagram:

Manufacturing System or

Device Vendor Database

Device Provisioning Portal | ◄——— Admin/Technician

(React + API App Service) |

Azure API Management (APIM)

Azure Functions (Verify +

generate device credentials)

Azure DPS (Device Provisioning Service | ◄——► | Azure SQL / Cosmos DB), (Device metadata, logs)

Azure IoT Hub

Device Twin / Identity

Device connects over MQTT/HTTPS

<u>WorkFlow</u>:

Azure Device Onboarding Workflow (Step-by-Step)

1. Device Verification Initiation

- A device boots up and displays a unique code or serial number.
- A technician/user visits the Provisioning Portal (React UI hosted on App Service).
- Enters the device's code or serial number.

2. API Request & Verification

- The frontend calls a secure API (via Azure API Management).
- The backend (Azure Function) checks:
    - Serial number / MAC against internal DB or manufacturing record.
    - That this device hasn't already been provisioned.
    - Optional: Validates a device-side certificate.

3. Azure DPS Enrollment

- Upon success:
    - Azure Function registers the device with Azure DPS (Device Provisioning Service).
    - Generates a device-specific key or certificate (X.509 or symmetric).
    - Stores metadata in Azure SQL or Cosmos DB.

4. Device Connects Automatically

- Device connects to Azure DPS using credentials.
- DPS assigns it to a target IoT Hub.

- Device receives its assigned IoT Hub hostname, security credentials, and provisioning status.

5. Device Lifecycle Management

- Device Twin is created in IoT Hub.
- OTA, telemetry routing, and monitoring now become active.