Knowing what you now know about preparing for, executing and experiencing edge cases, write a new version of HTDAM v2.0 as a PRD including all calculation specifications and presentation details at the end of each stage. Write it for an AI app developer which means do not overspecify or overfit platform and IT technology details the AI app developer will have their own environment. Make sure to include self-verification checks and most crucially ensure that all the physics and

Take into account that while we are building HTAM capable of preparing and transforming all HVAC telemetry we have an laser focus on the accuracy of the Bare Minimum Data preparation, transformation and analysis.

Also, I remind you that we produced many excellent charts and graphs while executing the HTDAM the PRD should specify all these charts and graphs, e.g. Signal Quality Score, Tranformation Recommednation, Material Penality Score, Chiller Telemetry Quality Assessment, GAP optimisation gains, etc.

Reminder, do not refer to this workflow as the MoAE protocol, the MoAE protocol applies to the output of our HTAM, it is not part of the HTAM as it relies on the HTAM phase presenting high-signal quality data.

IN the way you write your revised HTAM PRD you must indicate that factoring is required for the purpose of being able to use the HTAM process as the front end for many other of our HVAC analysis routines - meaning if it can be "containerised" or made as a "function" that can be dropped into similar environments as the front end this is a key objective. I also believe that specifying an orchestration hook - function architecture will help future integration into various apps - `useOrchestration` is a single React hook that composes all domain hooks (data, AI, auth, etc.) into one declarative pipeline or state machine, centrally controlling sequence, parallelism, retries, fallbacks, and human-in-the-loop steps for complex AI workflows while keeping components simple and reusable.

Now I'll create a comprehensive HTDAM v2.0 PRD that incorporates all learnings, focuses on the bare minimum data preparation, and is designed for AI app developers with orchestration hooks:HTDAM v2.0 = High-Throughput Data Assimilation Methodology v2.0.[1]

It is a 5-stage, factorable front-end pipeline for HVAC telemetry, designed to take the **Minimum Bare Data** (CHWST, CHWRT, CHW flow, power, CDWRT) and produce physics-correct, chart-ready, COP-ready datasets for downstream analysis engines.[1]

Below is a PRD for an AI-first implementation, written for an app developer. It assumes you will pick your own stack; the focus is on **algorithms, checks, and required visual outputs**, and on keeping HTDAM as a **containerisable function** that can plug into many HVAC workflows.

1. Product Overview

HTDAM v2.0 is a **domain pipeline** that ingests raw HVAC telemetry and outputs:

- Clean, unit-verified, synchronized time series on a master grid (usually 15-minute).[1]
- Physics-validated derived signals (ΔT, load $\dot{Q}$, COP, lift, approach).[1]
- A numerical **Data Quality Score** and a **Material Penalty Score** by stage.
- A standard set of **charts and summary tables** for human review (edge-case friendly).

HTDAM must:

- Be **factored** into small, reusable functions (hooks) that can serve as the front-end for many HVAC apps (COP, fault detection, benchmarking, etc.).
- Produce deterministic, inspectable outputs for the **Minimum Bare Data** set:
  - CHWST, CHWRT, CHW flow, power, CDWRT (measured).
  - ΔT, cooling load $\dot{Q}$, COP, lift, approach (derived).[1]

HTDAM is **not** the MoAE protocol; MoAE runs **after** HTDAM, on the high-signal data HTDAM produces.

2. Architecture & Orchestration

2.1 Functional Decomposition

Define HTDAM as a **pure domain module** exposing a small set of core entry points:

- `runHTDAM(params: HTDAMParams): HTDAMResult`
- `runHTDAMStage(stage: HTDAMStage, ctx: HTDAMContext): HTDAMContext`

Key **domain functions** (language-agnostic contracts):

- `verifyUnits(rawStreams) -> {streams, unitReport, unitScore}`
- `detectAndClassifyGaps(streams) -> {streamsWithGaps, gapSummary, gapScore}`
- `synchronizeTimestamps(streamsWithGaps) -> {syncedStreams, syncReport, syncScore}`
- `preserveAndAssessSignals(syncedStreams) -> {signalMetrics, signalScore}`
- `recommendTransformations(syncedStreams, signalMetrics) -> {exportSpecs, transformScore}`

Each function:

- Takes **typed inputs**, returns:
  - Data (arrays / data frames).
  - A **stage report** (numbers + flags).
  - A **stage score** and **penalty components**.

2.2 Orchestration Hook: `useOrchestration`

Expose a **single React-style hook** (or equivalent orchestrator) that composes domain functions:

- `const { state, run, retry, gotoStage, logs } = useOrchestration({ stages, hooks })`

Responsibilities:

- Sequence the 5 stages (with ability to reorder; HTDAM v2.0 = Gap FIRST).
- Allow **parallelism** where independent (e.g., per-stream analysis).
- Handle:
  - Retries on transient errors.
  - Fallbacks (e.g., reduced confidence if flow is missing).
  - Human-in-the-loop approvals (e.g., accept/reject exclusion window).
- Emit a **state machine**:
  - `idle → running → stage:N → complete|failed`.
  - Per-stage **metrics**, **penalties**, and **edge-case flags**.

The hook should be **agnostic** to rendering layer and storage; it just orchestrates domain functions and exposes structured state.

3. Inputs & Bare Minimum Data

3.1 Minimum Bare Measurements (per chiller)[1]

Mandatory measured inputs:

- CHW Supply Temp (CHWST, °C).[1]
- CHW Return Temp (CHWRT, °C).[1]
- CHW Flow Rate $\dot{V}_{\text{chw}}$, L/s or GPM.[1]
- Electrical Power W_{input}, kW (total chiller input).[1]
- Condenser Return Temp (CDWRT, °C).[1]

Derived (must be computed by HTDAM):

- ΔT: $\Delta T = \text{CHWRT} - \text{CHWST}$.[1]
- Mass flow: $\dot{m} = \dot{V} \times \rho$ with $\rho \approx 1000\,\text{kg/m}^3$.[1]

- Cooling load: $\dot{Q} = \dot{m}\, c_p\, \Delta T$, $c_p \approx 4.186\,\mathrm{kJ/kg{\cdot}K}$.[1]
- COP: $\mathrm{COP} = \dot{Q}/W_{\mathrm{input}}$.[1]
- Lift (temperature lift): $\mathrm{Lift} = \mathrm{CDWRT} - \mathrm{CHWST}$.[1]
- (Optional) Approach: e.g., condenser water approach vs ambient or tower supply, if available.

Self-verification requirement:
If **flow or power** is missing, HTDAM must:

- Flag COP and load as **non-computable**.[1]
- Downgrade hypothesis-testing confidence (no 7% claim possible).[1]
- Continue to process temperature telemetry, but clearly mark that COP-based outputs are **not valid**.

4. HTDAM Stages v2.0 (Gap FIRST)

Stage 0 – Ingestion & Schema Normalization

Goal: Normalize raw HVAC telemetry into a canonical structure.

- Input formats: CSV, XLSX, API responses, etc.
- Target schema (per point / stream):
  - `timestamp` (UTC, ISO 8601).
  - `value` (numeric).
  - `point_id` / `tag`.
  - `unit` (as reported).
  - `source` (BMS, logger, etc.).

Checks:

- All timestamps parse; malformed timestamps counted and either fixed or dropped.
- No duplicate timestamps per point (if duplicates, document resolution rule: e.g., average or latest).

Outputs:

- `normalizedStreams` (per tag).
- `ingestionReport` (counts, dropped records, unit strings seen).
- Charts (optional quick-look):
  - Histogram of raw timestamp intervals per stream (to visually see COV behavior).

Stage 1 – Unit Verification & Physics Baseline

Core requirement: Guarantee that all bare minimum channels are in **correct physical units** and within plausible ranges.[1]

1.1 Unit Checking

For each mandatory point:

- CHWST, CHWRT, CDWRT:
  - Expect °C.
  - Accept °F → convert: $T_{^\circ C} = (T_{^\circ F} - 32) \times 5/9$.
  - Flag any non-temperature units.
- Flow:
  - Accept L/s, m³/h, GPM; convert to m³/s internally.[1]
  - Maintain original unit for display.
- Power:
  - Accept kW; convert W, MW as needed.

Self-verification:

- Unit matrix: for each point, store `(reported_unit, canonical_unit, conversion_applied)`.
- If unit ambiguous (e.g., missing), score penalty and require manual override.

1.2 Physics Range Checks

Temperature checks:

- CHWST in typical range: 3–20 °C (configurable).
- CHWRT ≥ CHWST for ≥ 99% of valid records.
- CDWRT ≥ CHWST for ≥ 99% (positive lift).

Flow & power checks:

- Flow never negative; outside [0, max_design × factor] should be flagged.
- Power never negative; outside plausible chiller envelope flagged.

Metrics & Penalties:

- Per-point **Unit Confidence** in.[2]
- Stage 1 penalty: sum of:
  - Missing unit: −0.05 each core channel.
  - Ambiguous / manual override: −0.02.
  - Severe physics violation (e.g., CHWRT < CHWST for >5%): −0.10.

Charts & Tables:

1. **Temperature Range Chart**:
   - Boxplots per stream (CHWST, CHWRT, CDWRT).
   - Overlay expected design bands.
2. **Physics Violations Table**:
   - `% of time CHWRT < CHWST`.
   - `% of time lift ≤ 0`.
3. **Unit Verification Table**:
   - One row per mandatory point: reported unit, canonical unit, conversion, confidence.

Stage 2 – Gap Detection & Classification (FIRST in v2.0)

This stage must run **before** synchronization to preserve the semantics of COV and sparse logging.[1]

2.1 Gap Identification

Per time-ordered stream:

- Compute inter-sample intervals: $\Delta t_i = t_{i+1} - t_i$ in seconds.
- Classify each interval:
  - `NORMAL`: $\Delta t \leq 1.5 \times T_{\text{nominal}}$.
  - `MINOR_GAP`: $1.5 \times T_{\text{nominal}} < \Delta t \leq 4 \times T_{\text{nominal}}$.
  - `MAJOR_GAP`: $\Delta t > 4 \times T_{\text{nominal}}$.
- Detect **multi-stream aligned gaps**:
  - If all mandatory channels have a MAJOR_GAP over the same window → candidate **exclusion window** (maintenance/offline).

2.2 Gap Semantics (COV vs Sensor Failure)

Use **value behavior** around gaps:

- If preceding and following values are **identical** (or within tiny tolerance):
  - Likely `COV_CONSTANT`: setpoint held, no change logged.
- If values show **small drift**, but interval extended slightly:
  - `COV_MINOR`.
- If large, abrupt jumps with inconsistent patterns:
  - `SENSOR_ANOMALY` (do not treat as benign).

Outputs per gap:

- `gap_id`, `start_ts`, `end_ts`, `duration_s`, `type`, `affected_streams`.

2.3 Exclusion Windows

Combine per-stream MAJOR_GAPs:

- Rule example:
  - If ≥ 2 mandatory streams have `MAJOR_GAP` overlapping ≥ 8 hours → propose exclusion window.
- Require:
  - **Human approval hook** in orchestration (e.g., UI confirm).
- Mark all timestamps inside confirmed exclusion as `EXCLUDED`.

Metrics & Penalties:

- `GapCount` by type.
- `DataLossPct` = total excluded duration / total period.
- Stage 2 penalty guidelines:
  - `COV_CONSTANT`: −0.00 (semantically benign if identified).
  - `COV_MINOR`: −0.02 total.
  - `SENSOR_ANOMALY`: −0.05 to −0.10 depending on share.
  - `EXCLUDED`: −0.03 if >5% of period.

Charts & Tables:

1. **Gap Timeline Chart**:
   - Gantt-style bar per stream showing NORMAL, MINOR_GAP, MAJOR_GAP, EXCLUDED.
2. **Gap Optimisation Gains Chart**:
   - Before/after reordering (old pipeline vs v2.0):
     - Show mis-penalised COV gaps vs correctly neutral classification.
   - Output: numeric gain in confidence (e.g., +0.30).
3. **Gap Summary Table**:
   - Counts and durations by type per stream.
   - DataLossPct, ExclusionWindows list.

Stage 3 – Timestamp Synchronization

Now that gaps are classified, align streams to a **master timeline** without losing semantics.

3.1 Master Grid Definition

- Default: 15-minute grid (900 s), from earliest to latest valid timestamp.
- Configurable frequencies (`T_nominal`), but must be a divisor of 1 hour (5/10/15/30 min).

3.2 Alignment Algorithm

For each stream and each grid timestamp t_g:

- Find nearest raw timestamp t_r with $|t_r - t_g| \leq$ tolerance (default 30 min).
- If found:
  - Assign `value(t_g) = value(t_r)`.
  - Store `dt = |t_r - t_g|`.
- If none:
  - Set value to `NaN` and propagate gap type (likely `COV_CONSTANT` or `EXCLUDED`).

Alignment quality flags per grid point:

- `EXACT`: $|\text{dt}| < 1$ min.
- `CLOSE`: $1 \leq |\text{dt}| < 5$ min.
- `INTERP`: $5 \leq |\text{dt}| \leq 30$ min.
- `MISSING`: >30 min (already classified as gap).

Compose a **unified gap_type** per timestamp:

- If in exclusion window → `EXCLUDED`.
- Else if all streams have `EXACT` or `CLOSE` → `VALID`.
- Else if any stream `MISSING` → `COV_CONSTANT`.
- Optionally `COV_MINOR` for long but within-tolerance intervals.

Assign **confidence**:

- `VALID & EXACT`: 0.95.
- `VALID & CLOSE`: 0.90.
- `VALID & INTERP`: 0.85.
- `COV_*`: 0.00.
- `EXCLUDED`: 0.00.

3.3 Jitter Assessment

- After sync, all intervals on master grid are uniform by design.
- Verify:
  - Interval CV ≈ 0.00%.
  - No duplicate or missing grid timestamps.

Metrics & Penalties:

- `ExactMatchPct`, `CloseMatchPct`, `CoveragePct`.
- Stage 3 penalty:
  - `(1 - CoveragePct)` scaled (e.g., 6.2% gaps → −0.05).

Charts & Tables:

1. **Synchronization Summary Table**:
   - Input records per stream.
   - Grid size, coverage, match breakdown.
2. **Chiller Telemetry Quality Assessment Chart**:
   - Bar chart of `Exact`, `Close`, `Interpolated`, `Missing` percentages per stream.
3. **Material Penalty Score Chart (up to this stage)**:
   - Shows cumulative score from 1.00 → 0.88 after sync, with penalty components.

Stage 4 – Signal Preservation & Physics Checks

Focus on **preserving and assessing** the signal content relevant for physics and control, especially for **Minimum Bare Data**.[1]

4.1 Core Derived Signals

Compute on synchronized grid (only `VALID` rows):

- $\Delta T = \mathrm{CHWRT} - \mathrm{CHWST}$.[1]
- $\dot{m} = \dot{V}\rho$ (if flow telemetry present).[1]
- $\dot{Q} = \dot{m} c_p \Delta T$.[1]
- $\mathrm{COP} = \dot{Q} / W_{\mathrm{input}}$.[1]
- Lift: `CDWRT - CHWST`.[1]
- (Optional) Approach: vs ambient or supply, if ambient data is available.

Self-verification:

- If flow or power is missing for any timestamp, mark:
  - `load_available = false`, `cop_available = false`.
  - Do not compute COP; set to `NaN` and log penalty.

4.2 Hunting Detection (FFT)

On CHWST / CHWRT series:

- Detrend (remove linear trend).
- Compute FFT.
- Focus on frequencies 0.001–0.015 Hz (100–600 s).
- Compute power in this band as % of total spectral power.

Rule:

- If power_band_pct > 5% → `HUNTING_DETECTED`.
- Else → `NO_HUNTING`.

4.3 Transients & Diurnal Patterns

- Rate-of-change:
  - $\Delta T_{step} = T_{i+1} - T_i$ per interval.
  - Statistics: % intervals with |Δ| > 0.1 °C, max/min ramps.
- Diurnal pattern:
  - Group by hour of day → mean temperature vs hour.
  - Compute daily amplitude (max–min of hourly means).
- Seasonal pattern:
  - Group by month → mean temperature vs month; compute seasonal amplitude.

Control interpretation:

- Daily range <1 °C → `TIGHT CONTROL`.
- 1–3 °C → `MODERATE CONTROL`.
- 3 °C → `LOOSE / MULTI-SETPOINT`.

4.4 Physics Consistency

- For all valid COP points:
  - Check Q and W_input magnitudes are within expected chiller envelope.[1]
  - Optionally compare to manufacturer curves if available.

Metrics & Penalties:

- `HuntingDetected` bool with confidence.
- Stage 4 penalty:
  - No hunting: +0.00.
  - Resampling info loss: −0.02 (acknowledge but small).
  - Missing flow/power (for COP): severe, but should already be encoded.

Charts & Tables:

1. **Signal Quality Score Chart**:
   - Overall HTDAM score vs stage index (1–5).
   - Annotated contributions: units, gaps, sync, signal.
2. **Hunting Detection Plot**:
   - Bar: power_band_pct per stream; threshold at 5%.
3. **Diurnal Cycle Plots**:
   - 24-hour line plots for CHWST, CHWRT, CDWRT.
4. **COP vs Load Scatter**:
   - For timestamps where COP is computable:
     - X: load fraction; Y: COP.

Stage 5 – Transformation Recommendation & Export Specs

Final stage: **decide how to expose HTDAM outputs** to downstream consumers.

5.1 Export Modes

At minimum, define three canonical outputs:

1. **Raw Synchronized Dataset (primary)**:
   - 15-min grid; Celsius; one row per timestamp.
   - Columns:
     - `timestamp`
     - `CHWST`, `CHWRT`, `CDWRT`, `Flow` (if present), `Power` (if present)
     - `gap_type`, `confidence`
     - Derived: `deltaT`, `load_kW`, `COP` (if computable), `lift`.
   - This is the dataset MoAE and other routines use as **front-end input**.
2. **Diagnostics Dataset (secondary)**:
   - Adds:
     - `load_class` (IDLE/PART/FULL, from ΔT and/or load).[1]
     - `condenser_status` (NORMAL/FOULED/STRESS, from lift/approach).
   - For dashboards and alarms.
3. **Summary JSON / Metadata**:
   - HTDAM scores per stage.
   - Gap statistics.
   - Physics check summaries.

- Chart configuration hints (so UI can render the standard set).

5.2 Transformation Recommendation Logic

Based on availability:

- If **flow & power present**:
  - Recommend **full COP-ready export**.
  - Confidence for COP analysis: 0.95.[1]
- If **flow present, power missing**:
  - Recommend **temperature + load** export.
  - Mark COP as unavailable; MoAE must treat savings claims as qualitative only.[1]
- If **flow missing**:
  - No rigorous load/COP possible.[1]
  - Export temperature-only dataset with strong warnings.
  - Downgrade baseline-hypothesis confidence accordingly.

5.3 Material Penalty & Transformation Charting

Define a **Material Penalty Score** breakdown table:

- Rows: Unit verification, Gaps, Sync, Signal, Transform.
- Columns: Base, Penalty, Result.
- Show final HTDAM score (e.g., 0.84) prominently.

Charts:

1. **Transformation Recommendation Chart**:
   - Bars for:
     - Option 1 (Raw Sync): quality 0.84.
     - Option 2 (Diagnostics): 0.82.
     - Option 3 (Estimated Power): 0.50 (flag as not recommended).
2. **Use-Case Matrix Table**:
   - Rows: COP calc, efficiency, fault detection, monitoring, baseline.
   - Columns: each option, with ✓ / ◐ / ✗.

5. Self-Verification & Edge Cases

HTDAM must include **internal self-checks** and strict handling of edge cases:

- **Missing Bare Minimum Channels**:
  - If any of CHWST, CHWRT, Flow, Power, CDWRT missing:

        - Mark pipeline as **degraded** for baseline hypothesis testing.[1]
        - Emit explicit warnings and lower overall score.
- **Physics Violation Guardrails**:
    - If CHWRT < CHWST for >5% of time → major penalty, require manual review.
    - If lift ≤ 0 for >1% of time → flag as sensor/config error.[1]
- **Timebase Anomalies**:
    - If master grid cannot be established (e.g., extremely sparse or chaotic timestamps) → fall back to **event-based** analysis and mark sync stage as partial.
- **Outlier Suppression**:
    - Optionally detect single-point outliers (e.g., 200 °C) and classify as SENSOR_ANOMALY instead of letting them distort ΔT and COP.
- **Reproducibility**:
    - All random choices (if any) must be seeded.
    - Each stage emits a **deterministic summary** and configuration snapshot.

6. Factorization & Reuse

HTDAM is intended as a **drop-in front-end function** for multiple HVAC analysis pipelines (COP, fault detection, benchmarking, control tuning, etc.):

- Provide a single, stable interface:
    - `runHTDAM(minimumBareData, options?) -> { syncedData, metrics, summary, chartsConfig }`.
- The **same HTDAM output format** can be fed into:
    - COP calculators.
    - Baseline comparison tools.
    - Forecasting / digital twin models.
    - Fault detection routines.
- `useOrchestration` should be designed so adding/removing stages (e.g., a future Stage 6 for multi-chiller aggregation) does not break existing consumers.

If you like, next step can be a **concrete TypeScript interface set** (`HTDAMParams`, `HTDAMResult`, `HTDAMStageReport`, chart config types) or a **sample** `useOrchestration` **implementation sketch** that wires these stages together.

⁂

1. Minimum-Bare-Data-for-Proving-the-Baseline-Hypothe.md
2. https://www.koryckimechanical.com/blog/why-your-chiller-plant-isnt-performing-how-to-fix-it/