Agentic AI Framework for Building Management Systems: Towards Intelligent and Autonomous Building Operations

Dissertation Mid-Semester Review – Progress & Demonstration

Prithviraj Acharya
BITS ID: 2023DC04009

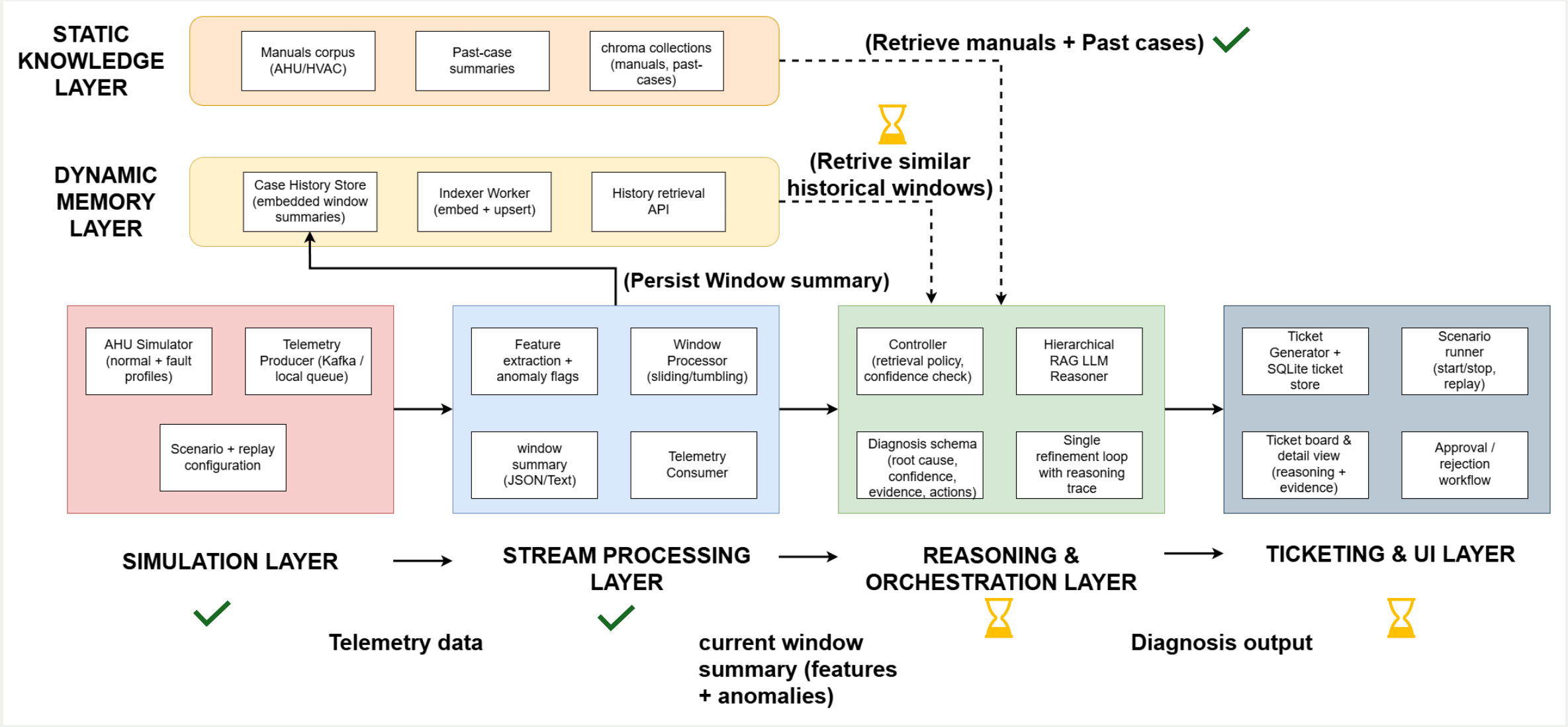
Dissertation Work carried out at: Honeywell Technology Solutions Lab Pvt. Ltd Bangalore, India
Submitted in partial fulfillment of Data Science & Engineering degree program
BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE

Why HVAC Fault Diagnosis Remains Hard

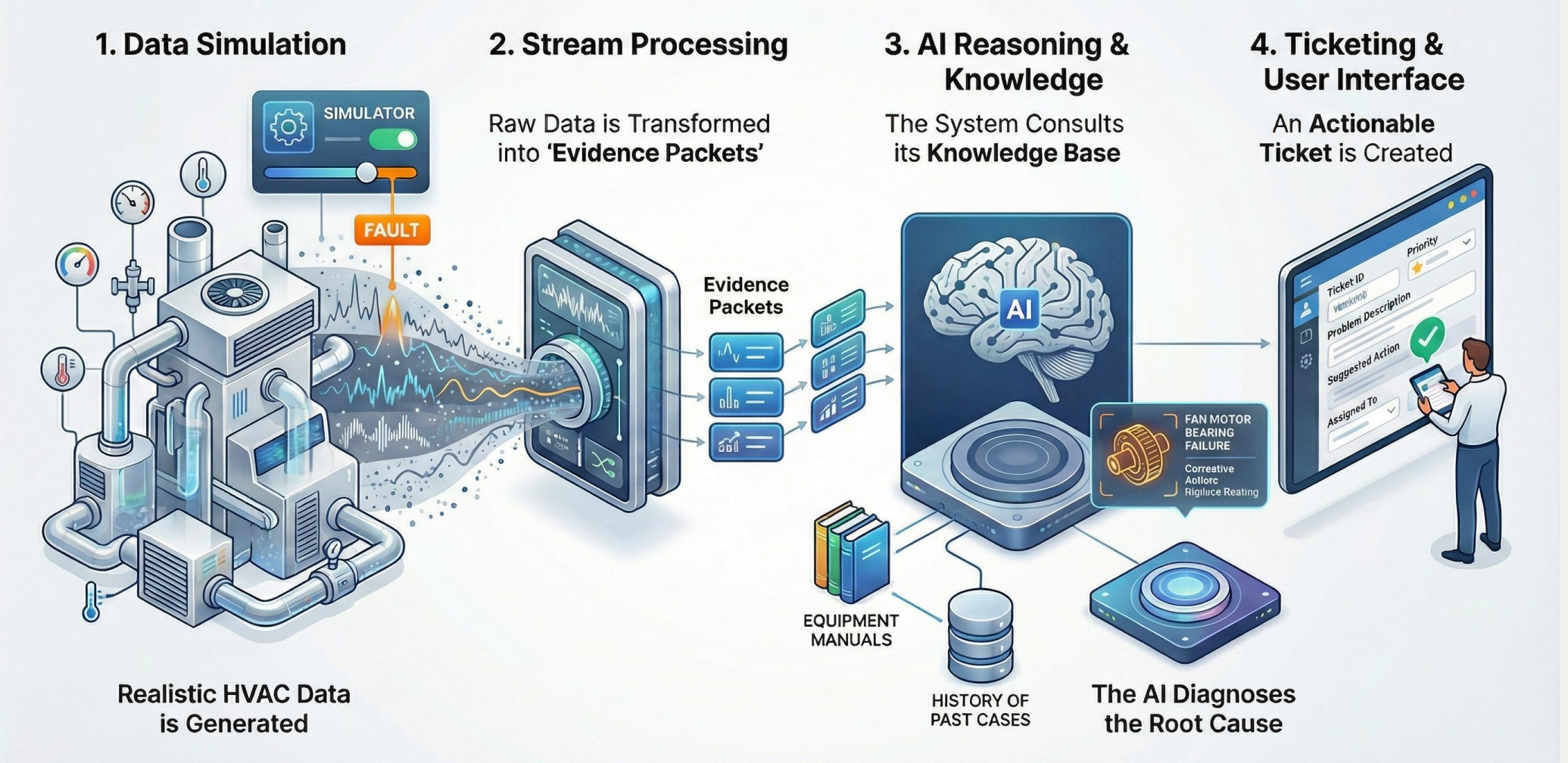
- Modern Building Management Systems generate **large volumes of HVAC telemetry.**
- Faults manifest as **temporal, multi-signal behavior patterns**, not single anomalies.
- Existing BMS diagnostics rely on **fixed rules and manual expert interpretation.**
- Current approaches detect faults but offer **limited root-cause explanation.**
- This leads to **reactive maintenance, longer downtime, and operational inefficiency.**

The Blueprint for Intelligent Diagnosis

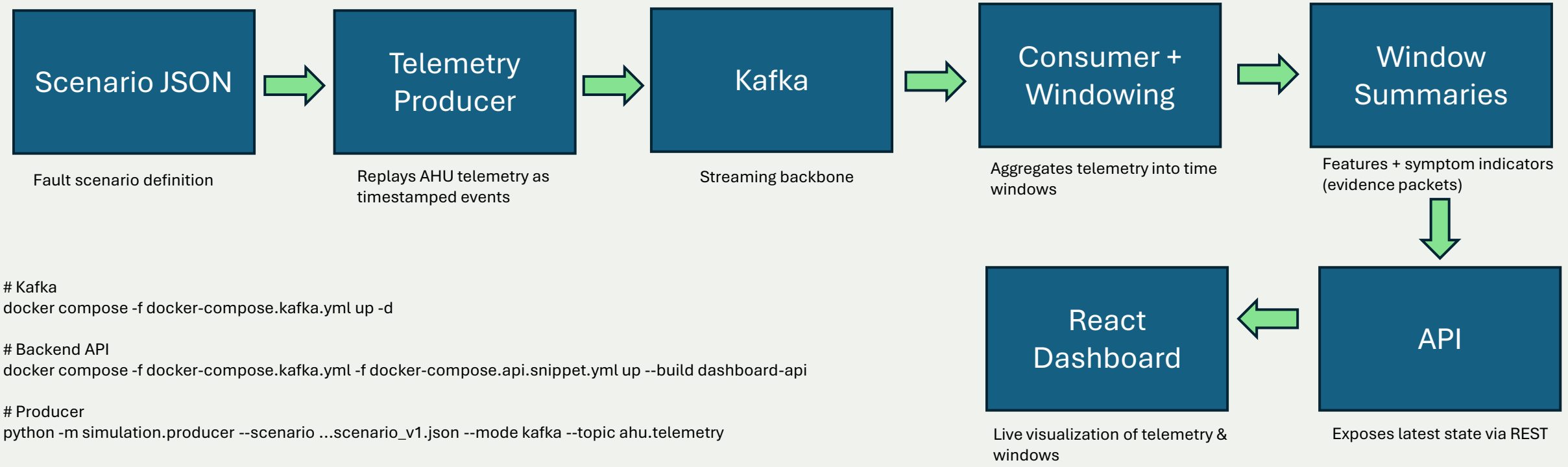
This layered, modular architecture maps the journey of data from raw telemetry to actionable insight. Each layer performs a distinct role, ensuring a clear separation of responsibilities from data generation to final diagnosis.



From **Sensor Noise** to **Smart Diagnostics**: An AI Framework for HVAC Faults



End-to-End Data Flow (Till Mid-Sem Submission)



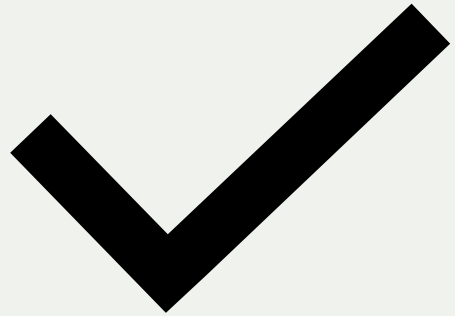

```
# Kafka
docker compose -f docker-compose.kafka.yml up -d

# Backend API
docker compose -f docker-compose.kafka.yml -f docker-compose.api.snippet.yml up --build dashboard-api

# Producer
python -m simulation.producer --scenario ...scenario_v1.json --mode kafka --topic ahu.telemetry

# Consumer (window summaries)
python -m telemetry_pipeline.run_pipeline --mode kafka --topic ahu.telemetry --summary-topic
window_summaries
```

No fault labels exist in telemetry - Faults emerge only as **window-level behavioral patterns.**

What Is Already Implemented

Implemented Components:

- **Deterministic AHU Telemetry Simulator**
- **Kafka-based Producer and Consumer Pipeline**
- **Sliding Windowing with Feature Extraction**
- **Rule-Based Symptom Detection at Window Level**
- **FastAPI Backend for State Exposure**
- **React (Vite) Frontend for Live Monitoring**
- **Static Knowledge Layer**

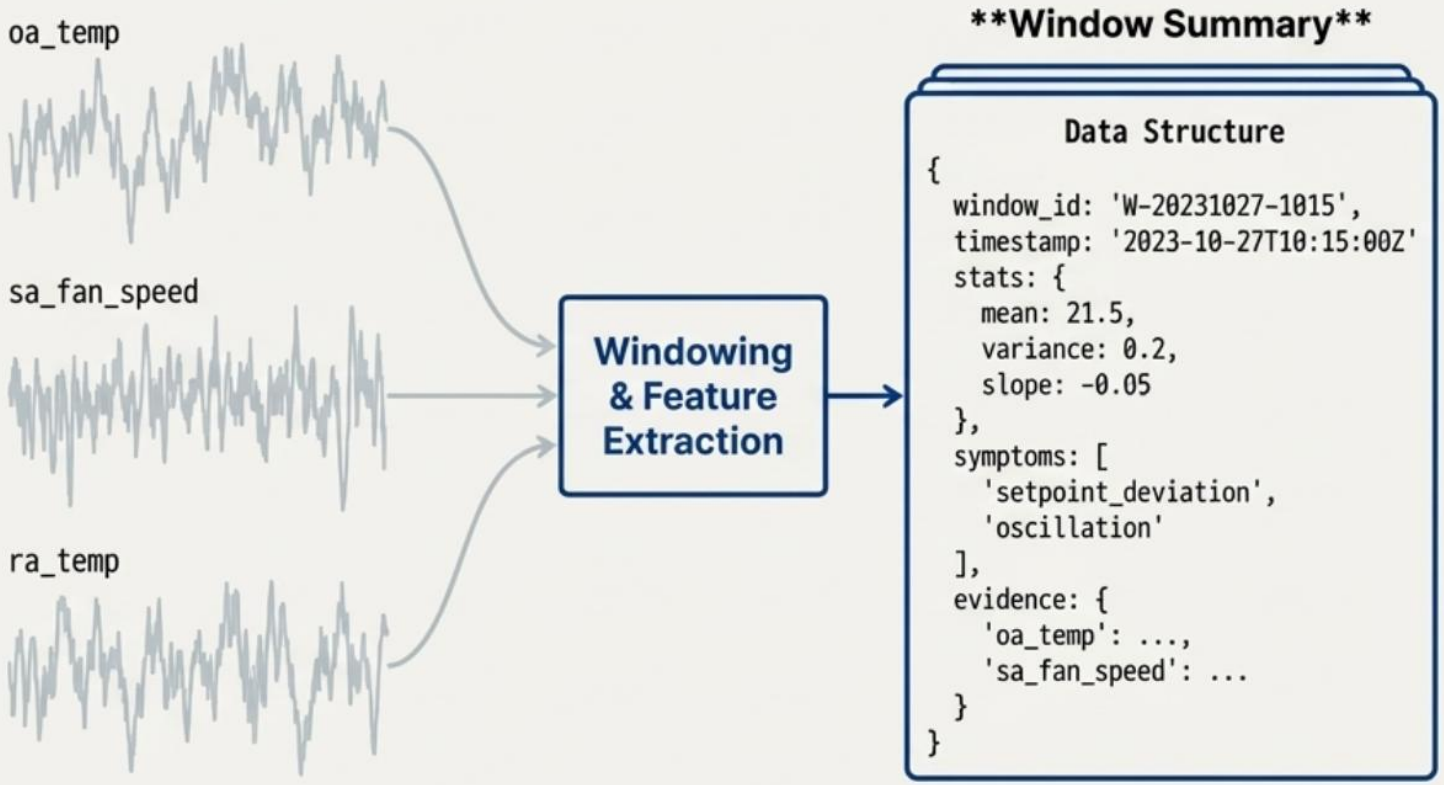


All listed components will be demonstrated live during the mid-semester demo.

Windowing: The Core Design Decision

Why Windowing?

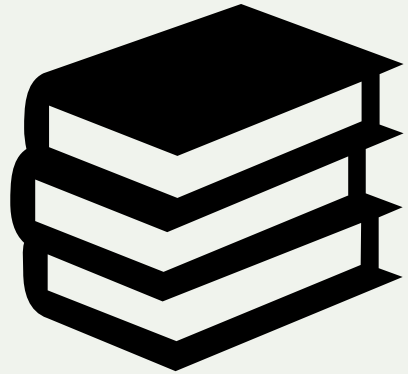
- Raw telemetry is noisy and ambiguous
- Faults are temporal patterns, not point anomalies
- Window summaries to convert time-series data into act as **evidence packets**



Symptom detection is deterministic. Diagnosis is reasoning-driven.

Static Knowledge Layer (Foundation for RAG)

- Builds two grounded knowledge collections: manuals (chunked PDF docs) + past cases (historical incidents)
- Uses fully local, deterministic embeddings (feature hashing) for repeatable retrieval (no external APIs)
- Retrieves Top-K evidence with provenance (doc/case IDs + chunk/page metadata) and similarity ranking



Sample run: python -m static_layer.smoke_test --query "economizer ineffective oa damper stuck" --top-k 3
Output shows: Top-K hits from **both** collections + provenance + distance/score (retrieval quality sanity check).

Static knowledge retrieval is validated independently before enabling reasoning.

Live Demonstration Sequence

- **Start a fault scenario**

Deterministic AHU telemetry replay from scenario configuration

- **Observe telemetry evolve in real time**

Sensor streams updating continuously in the dashboard

- **Watch window summaries form**

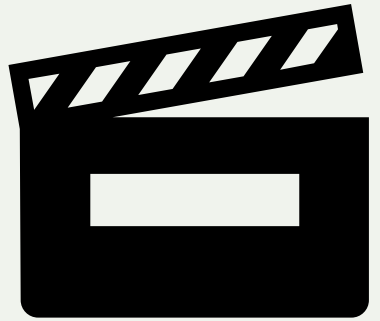
Telemetry aggregated into windows with extracted features

- **See symptoms emerge from patterns**

Rule-based symptom flags triggered at window level

- **View system state in the UI**

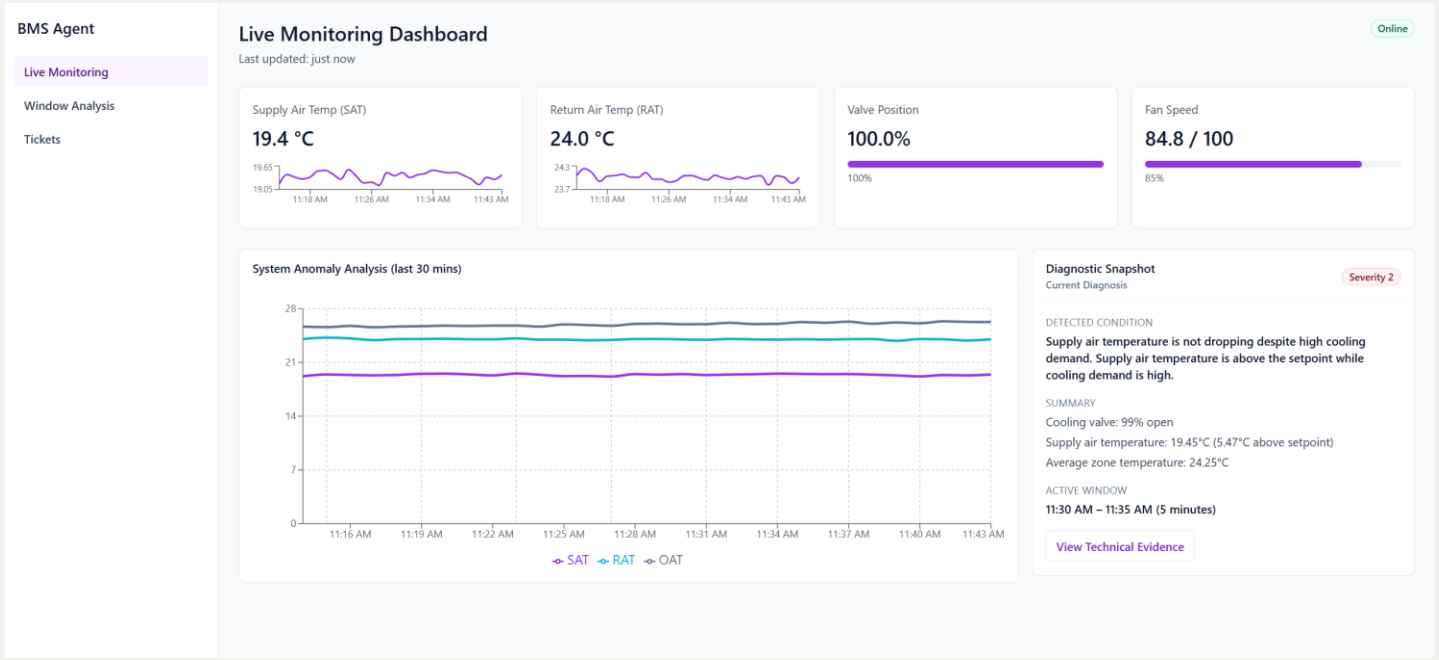
Live telemetry, current window summaries, and detected symptoms
(no diagnosis or automation yet)



- **Validate retrieved reference material**

Manual sections and past cases retrieved via static-layer smoke test
(shown via console output, not UI)

Why the UI Matters? (Even Before Full Automation)



Purpose of the UI

- **Transparency**

Makes intermediate system state visible (windows, features, symptoms)

- **Human Trust**

Operators can see *why* a condition was flagged before automation

- **Debugging & Validation**

Enables inspection of data, thresholds, and patterns before agentic reasoning

Technology

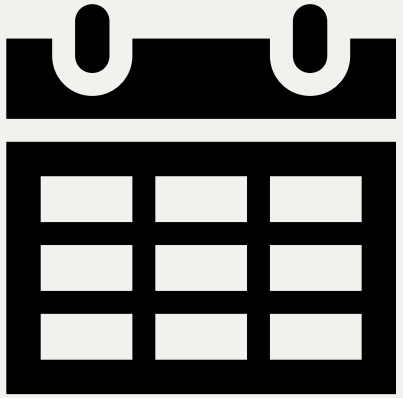
- **Frontend:** Vite + React
- **Backend:** FastAPI
- **Data Access:** Stateless REST polling (no Kafka in browser)

Planned Reasoning and Workflow Components

(Status at Mid-Semester Review)

Controller & Retrieval Routing *(Planned)*

- Triggers diagnostic reasoning from window-level evidence
- Routes retrieval across manuals, past cases, and history



Hierarchical RAG Reasoning *(Planned)*

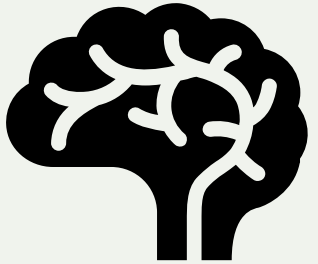
- Performs multi-stage, evidence-grounded diagnosis
- Outputs root cause, confidence, and cited evidence

Ticketing & Review Workflow *(Planned)*

- Persists diagnoses as structured maintenance tickets
- Enables human review and auditability

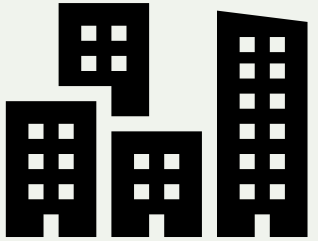
Looking Ahead!

- **Agentic Diagnostic Reasoning**
  Autonomously reasons over window-level evidence
  Combines telemetry with manuals and past cases



- **Explainable AI Decisions**
  Produces root cause, confidence, and justification
  Exposes reasoning and evidence for inspection

- **Human-in-the-Loop Validation**
  Supports review, approval, and traceability
  Enables systematic evaluation across fault scenarios



(**Outcome:** End-to-end, explainable Agentic AI diagnostics demonstrated in the final viva.).

Honeywell

THANK YOU

This mid-semester deliverable proves the system can reliably convert HVAC telemetry into explainable diagnostic evidence — the foundation required for agentic reasoning.