FLUVENTA · STAKEHOLDER EDITION

How It Works
Plain-Language Guide

A visual, nontechnical explanation of how Fluventa learns normal pipeline behavior, notices change, handles unreliable sensor data, and keeps people in control.

For

Operators, managers, evaluators, clients, and interview panels

Covers

The complete journey from sensor reading to human decision

Uses

Real screens and the actual implemented operating rules

Avoids

Code, formulas, and unexplained machine-learning terminology

CODE-VERIFIED PROTOTYPE · 08 AUGUST 2026

START HERE

The whole idea in one page

Fluventa is an early-warning assistant for one monitored pipeline checkpoint.

It watches four sensor signals together: pressure, temperature, flow, and vibration. Instead of waiting for a person to notice every unusual combination, it continuously asks whether the latest one-minute pattern still looks like approved healthy operation.

WATCH	COMPARE	ASSIST
Collect one reading per second and preserve the original evidence.	Compare the latest 60-second pattern with learned healthy behavior.	Show evidence, open a record when rules are met, and let a person decide.

What the system can say

“This behavior is different from what the approved healthy examples taught me, and this is the evidence that contributed most.”

What the system cannot say

It cannot independently prove that there is a leak, blockage, mechanical fault, or safety emergency. Those are human interpretations that need operational context and, eventually, field validation.

The key promise Fluventa helps people notice and investigate change earlier. It does not replace an operator, engineer, protection system, or emergency procedure.

Inside this guide

Part	Plain-language question
1–4	What does it watch, and how does a reading become useful evidence?
5–9	What happens during a sudden event, recovery, slow drift, or bad sensor data?
10–13	What does the operator see, and where do people remain in control?
14–17	Why was it designed this way, how should it be demonstrated, and what should stakeholders ask?

1

The problem Fluventa is trying to solve

A pipeline checkpoint produces more information than a person can comfortably compare every second.

Pressure

How strongly the fluid is pushing at the checkpoint.

Temperature

The thermal condition of the fluid at that moment.

Flow

How much material is moving through the checkpoint.

Vibration

How much mechanical movement the sensor is measuring.

Each signal can look ordinary on its own while the combined pattern is unusual. Fluventa therefore looks at the four signals together and over time, rather than treating each number as an isolated alarm.

Who uses it

Person	What Fluventa helps them do
Operator	See current conditions, review evidence, acknowledge a record, and add operational context.
Engineer	Review patterns, investigate causes, set controlled monitoring values, and evaluate model candidates.
Manager or evaluator	Understand when the system speaks, when it stays silent, and which decisions remain human.

A useful analogy Think of an experienced listener who learns the normal sound of a machine. Fluventa learns a numerical version of that normal rhythm across four sensors.

2

A reading’s journey

The system separates evidence, comparison, alerting, and human interpretation.

1. Sensors report
Pressure, temperature, flow, and vibration arrive once per second.

↓

2. Data quality is checked
The system marks missing, late, frozen, repeated, or impossible values.

↓

3. A one-minute picture is formed
The newest reading joins the previous 59 readings.

↓

4. Healthy similarity is measured
The learned healthy pattern is used as a reference, not as a fault dictionary.

↓

5. Operating rules interpret the evidence
Fast rules watch sudden change; a separate slow path watches sustained drift.

↓

6. People receive a traceable record
The screen shows what changed, when it changed, and what action a person recorded.

Why these steps are separate A strong comparison score is evidence of unusual behavior. It is not automatically a diagnosis. Keeping diagnosis separate prevents the model from claiming more than it knows.

3

What “learning healthy” really means

Fluventa is taught from approved normal examples because real fault examples are often rare, incomplete, or disputed.

During training, the system sees many overlapping one-minute examples of healthy operation. It learns how the four signals usually move together. When shown a new minute, it tries to reproduce that familiar pattern. A small difference means “this resembles what I learned.” A large difference means “this deserves attention.”

TEACH	CHECK	FREEZE
Use only data that people have approved as healthy.	Confirm the model behaves sensibly on a later, separate healthy period.	Save the model and its matching scale and thresholds as one controlled package.

Why not teach it every known fault?

- Some faults may never have occurred at the monitored checkpoint.
- Historical labels may be incomplete or based on hindsight.
- A new kind of unusual behavior can still differ from healthy operation even if nobody has named it before.

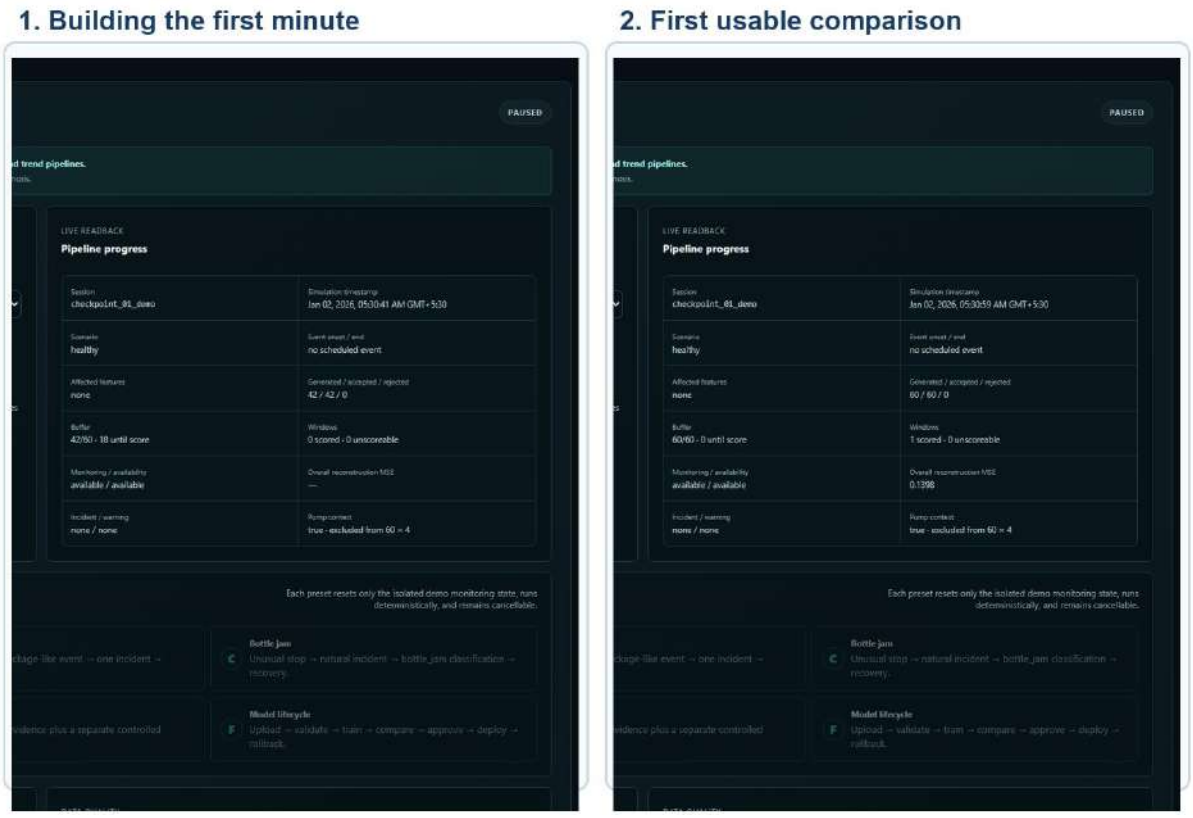
Important consequence The model detects unfamiliar behavior. It does not know the business name or physical cause until a qualified person investigates and classifies the event.

4

Why the first score takes one minute

Fluventa needs context, not just a single number.

The application receives one reading each second. A complete comparison requires 60 readings, so the first 59 seconds are a deliberate “building context” period. The first usable score appears only after the 60th valid reading completes the one-minute picture.



The real application first shows a growing buffer, then produces the first usable comparison when 60 readings are available.

Condition	What a stakeholder should expect
Seconds 1–59	Live readings are visible, but there is not yet enough history for a model score.
At reading 60	The first complete 60-second comparison can be made.
Every valid reading after that	The one-minute window moves forward by one second and a fresh comparison can be produced.
Data becomes unscoreable	The screen explains limited monitoring rather than inventing a result.

This is expected behavior “No score yet” during startup is not a failure. It protects the system from making a comparison without enough context.

5

Two clocks watch two kinds of change

A sudden event and a slow deterioration should not be treated as the same operational story.

Fast watch: sudden change	Slow watch: sustained change
Looks at each usable one-minute comparison as it arrives.	Summarizes the evidence into five-minute medians.
One score above the opening level can start an incident.	Reviews a six-hour view every 30 minutes.
Repeated abnormal scores update the same incident.	Five of the latest six valid reviews must support the trend before warning.
Designed for immediate operational attention.	Designed as a lower-priority prompt for planned investigation.

The same model evidence feeds both views, but the operating rules deliberately ask different questions:

Fast question

“Did the latest behavior change sharply enough to deserve attention now?”

Slow question

“Has a smaller change persisted often enough that we should investigate it?”

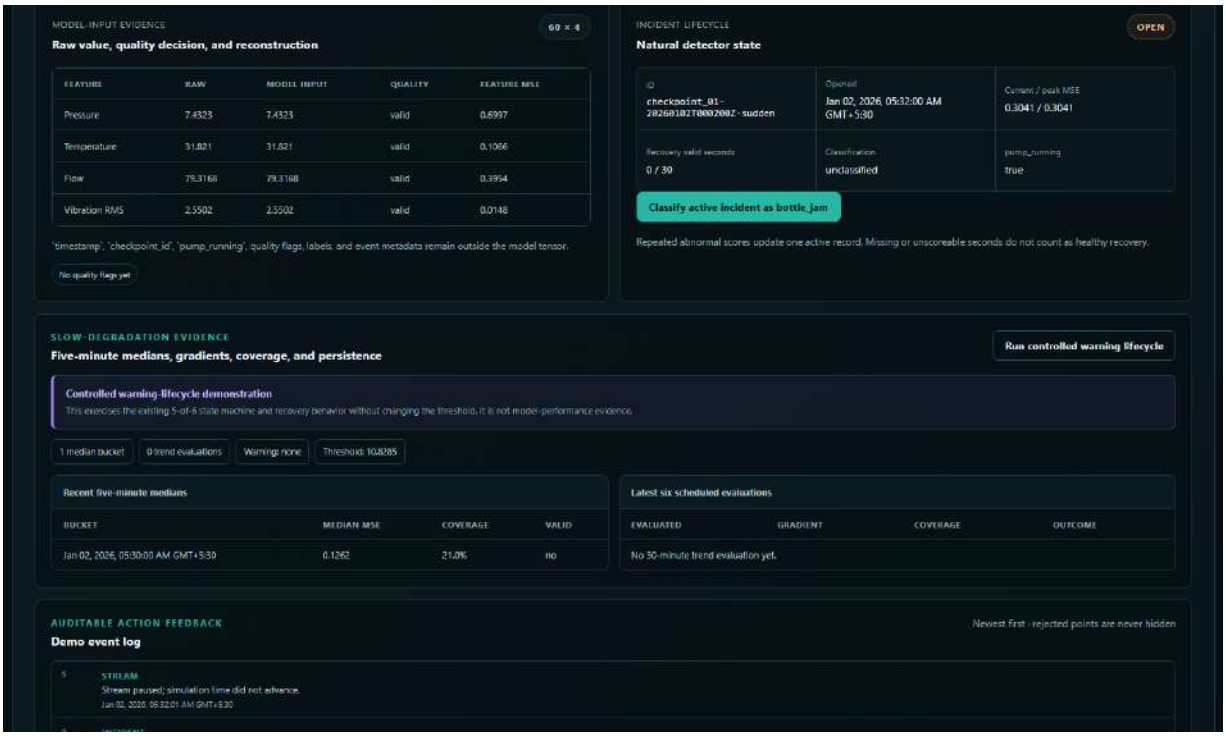
Why this matters One compromise rule would either react too slowly to sudden change or too noisily to small drift. Two time scales make the behavior easier to explain and govern.

6

What happens during a sudden event

The system opens one incident record and keeps adding evidence to it while the condition continues.

OPEN	UPDATE	CLASSIFY
A valid score crosses the sudden-event opening level.	More abnormal evidence updates the same incident instead of creating duplicates.	An operator may acknowledge it, choose a suspected category, and add a note.



A real demo state showing the model-input evidence beside the single active incident record and its operator action.

What the screen is communicating

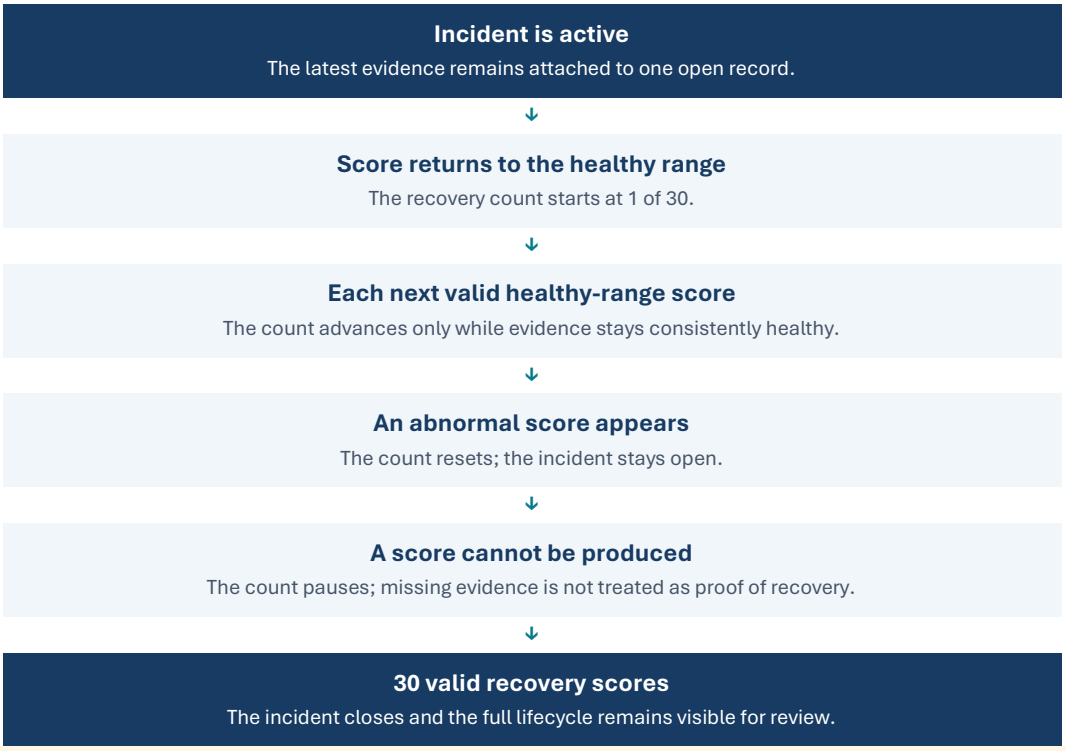
- The event was opened by the natural detector rule, not by a manual “create incident” button.
- The score shows how unusual the full one-minute pattern is.
- Per-sensor contributions show which signals influenced that score most.
- A classification such as “bottle jam” is a person’s interpretation, not the model’s diagnosis.

7

How recovery works

An incident does not disappear as soon as one reading looks better.

Fluventa uses a separate, lower recovery level. Once an incident is active, the score must remain below that recovery level for 30 consecutive valid comparisons before the incident closes.

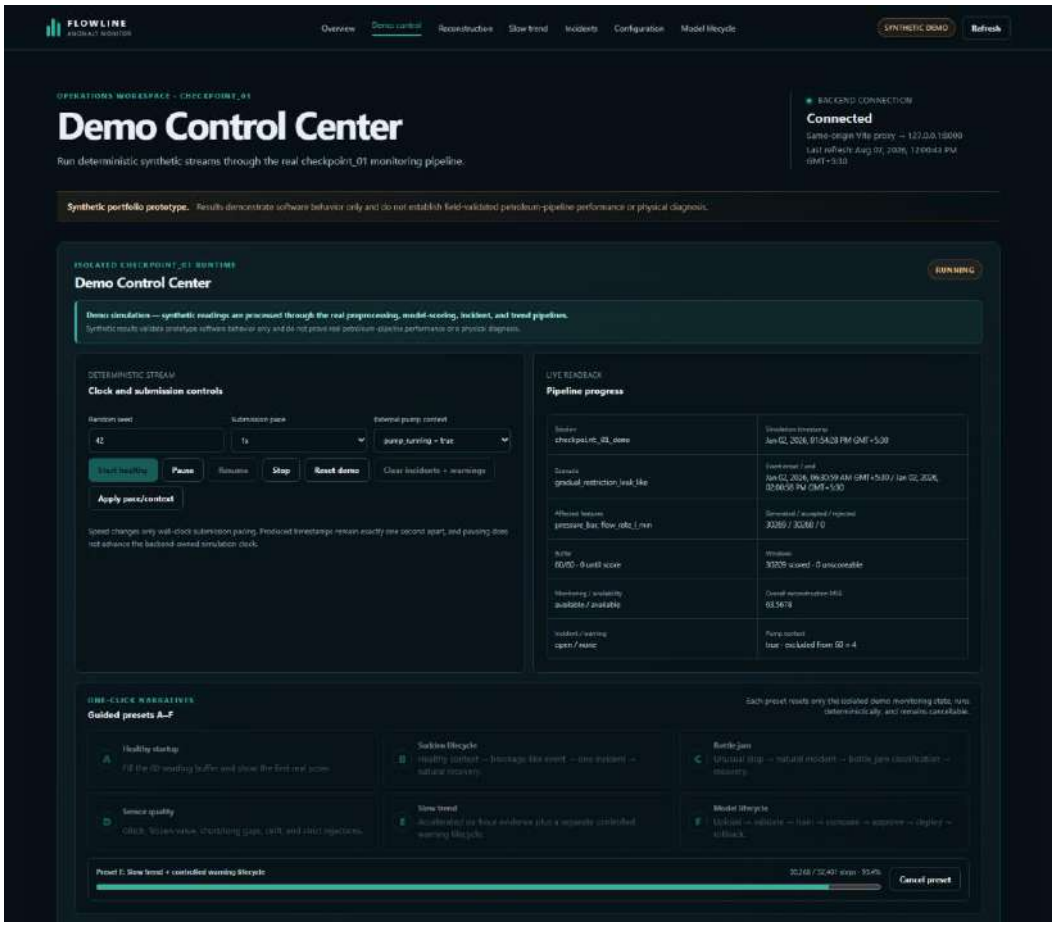


Why 30 consecutive results? It prevents a brief calm moment from closing an event that may still be developing. The exact setting is provisional in this prototype and would require site validation.

8

How slow deterioration is noticed

Small changes are summarized and judged for persistence before a warning is opened.



The real controlled slow-trend demonstration: five-minute evidence is collected before the six-hour trend view can make a persistence decision.

Every usable model score
Evidence arrives from the same healthy-pattern comparison.

↓

Five-minute summary
A median reduces the influence of one isolated peak.

↓

Six-hour view
The system examines the broader direction, not a single moment.

↓

Review every 30 minutes
Each scheduled review records whether the trend evidence is strong enough.

↓

Five of six valid reviews agree
A separate low-priority warning is opened for human investigation.

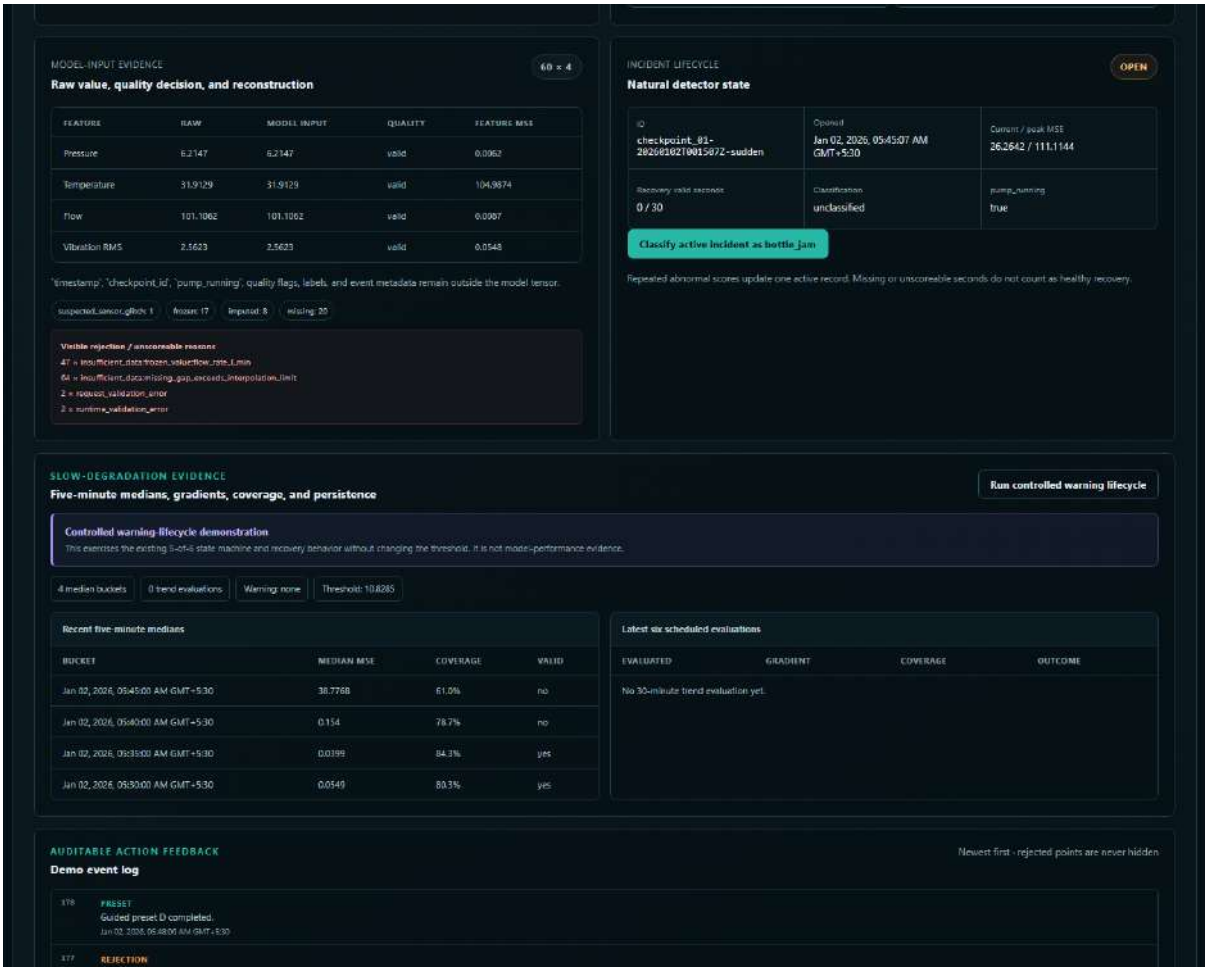
When evidence is incomplete If the six-hour view has poor data coverage, Fluventa records limited monitoring. It does not turn missing history into a confident trend claim.

9

What happens when sensor data is unreliable

Data quality is treated as evidence, not as an inconvenience to hide.

Data condition	Implemented response	Reason
Gap of 1–3 seconds	May be filled only in the processed model stream.	Preserves a short window without rewriting the original evidence.
Longer gap	The affected window is not scored.	A guess would be more misleading than an honest absence of evidence.
One isolated spike on one sensor	May be cleaned in the model input and clearly labelled.	Reduces sensitivity to a single-sample glitch.
Several sensors change, or change persists	The change reaches the model.	It may be a real operating event rather than sensor noise.
Frozen or invalid value	Quality status is shown and scoring can be suppressed.	A plausible-looking number can still be untrustworthy.



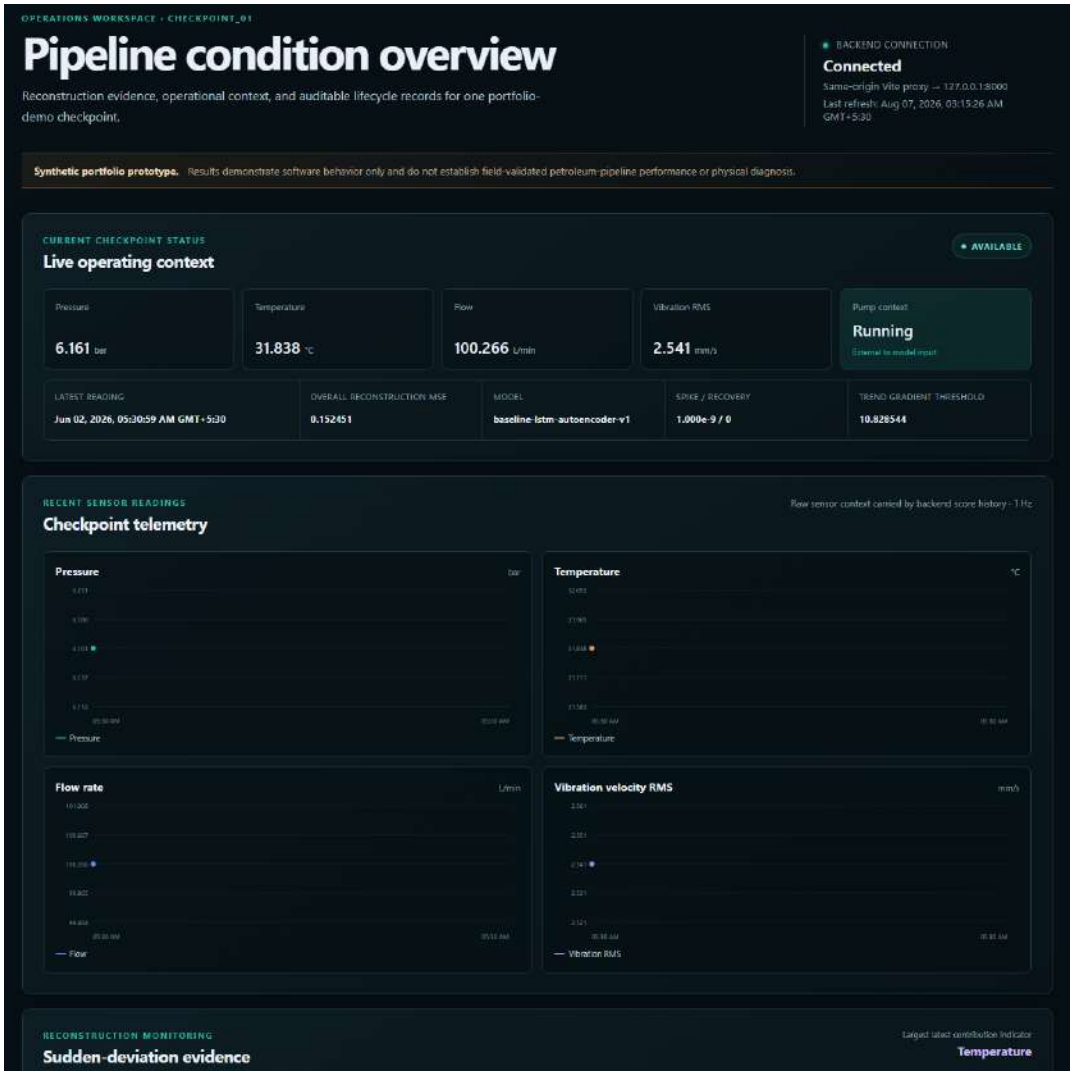
The real sensor-quality demonstration: raw values, processed model input, quality decisions, rejected events, and coverage evidence remain visible.

Two records are kept Raw readings preserve what arrived. The processed stream shows exactly what the model received. This makes any small-gap filling or spike cleaning reviewable.

10

What an operator sees

The overview brings live context, model evidence, trend evidence, and incident history into one place.



Top of the real overview: connection status, current checkpoint context, latest readings, and live telemetry.

Live context

The latest sensor values and whether the pump is reported as running.

Model evidence

The overall difference from learned healthy behavior and the contribution from each sensor.

Lifecycle state

Whether an incident or slow warning is open, recovering, acknowledged, classified, or closed.

Audit trail

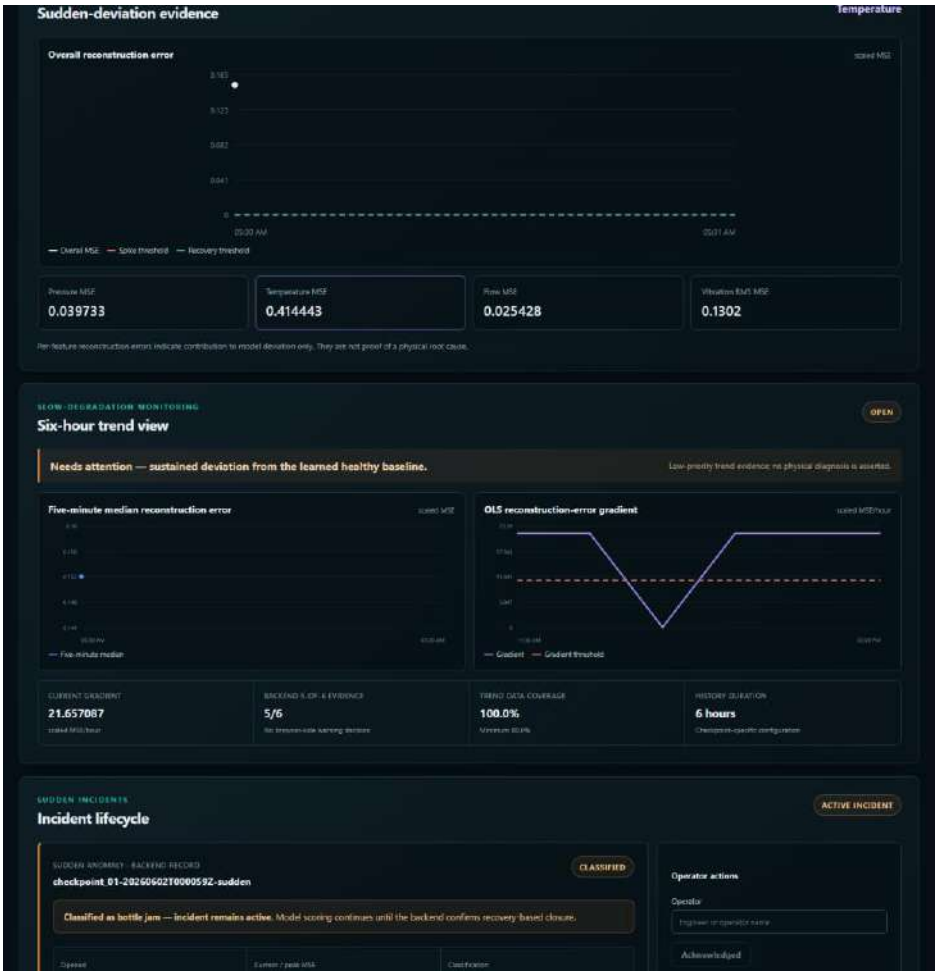
Who recorded a decision, what they selected, and which note they added.

Pump status is context The current prototype displays pump-running state to help interpretation, but it does not automatically silence an alert. That avoids hiding unusual behavior merely because one status flag looks normal.

11

How to read the evidence without overclaiming

The dashboard separates “how unusual?” from “what caused it?”



The real overview shows sudden evidence and the slower trend view as separate monitoring stories.

What you see	Safe interpretation	Unsafe interpretation
Overall score	The four-sensor pattern differs from learned healthy examples.	The system has proved a specific physical fault.
Largest sensor contribution	That sensor most influenced this model comparison.	That sensor or component is definitely the root cause.
Repeated high evidence	The unusual pattern is persisting.	Severity or safety impact is known without investigation.
Operator classification	A person has recorded an operational interpretation.	The neural network automatically diagnosed the event.

Good stakeholder language Say “unusual compared with the learned healthy baseline” and “supports investigation.” Avoid “the AI detected a leak” unless separate validated systems and evidence prove that claim.

12

People remain in control

Fluventa automates comparison and record-keeping; people own operational meaning and model approval.

System responsibility	Human responsibility
Collect and preserve readings.	Confirm sensors and operating context are trustworthy.
Apply consistent quality rules.	Investigate conditions the software cannot observe.
Compare behavior with the approved healthy baseline.	Decide whether the evidence represents a fault, maintenance need, or benign change.
Open or update records when explicit rules are met.	Acknowledge, classify, comment, and follow site procedures.
Present candidate-model evidence.	Approve deployment or choose rollback.

A typical operating response

1. REVIEW
Check the live values, quality labels, score, and sensor contributions.

2. VERIFY
Compare with field conditions, maintenance activity, and other protection systems.

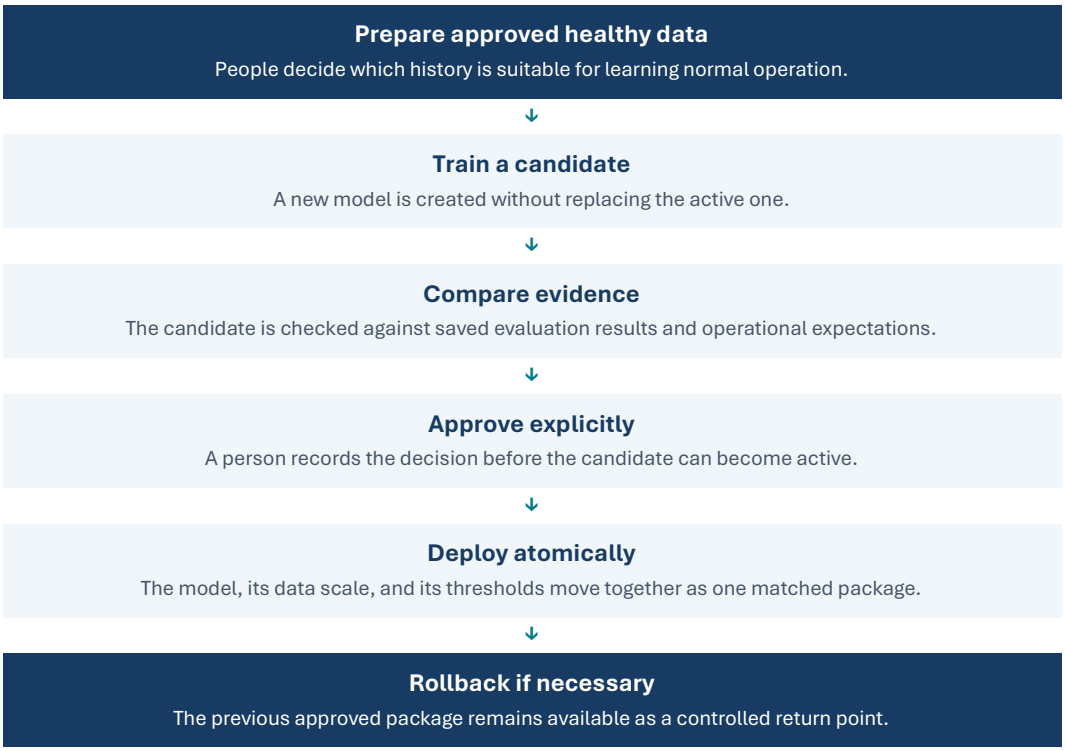
3. RECORD
Acknowledge the record, select a classification if justified, and add context.

No automatic shutdown The prototype does not control valves, pumps, or emergency equipment. It supports observation and investigation only.

13

How the model is updated safely

A new model is treated as a candidate until a person reviews and approves it.



Why the whole package moves together

A model interprets numbers using the scale and limits that were created with it. Mixing one model with another model's scale or thresholds could produce convincing-looking but invalid results. Fluventa therefore treats them as one versioned unit.

Plain-language governance rule Training may be automated; approval is not. The active model changes only through a deliberate, recorded human action.

14

Why it was designed this way

The major choices make the behavior more trustworthy and easier to explain.

Design choice	Why this approach	What it prevents
Learn approved healthy operation	Reliable fault examples are limited, but healthy operation can be reviewed.	Pretending the model knows every possible fault.
Use one-minute, four-sensor patterns	Context and relationships matter more than one isolated value.	Overreacting to a single ordinary fluctuation.
Separate sudden and slow paths	Fast events and gradual change need different evidence clocks.	One rule being simultaneously noisy and slow.
Use a lower recovery level	Opening and closing should not bounce around the same boundary.	Rapid open-close-open “chatter.”
Keep raw and processed data	Any cleaning decision must be visible later.	Silent rewriting of operational evidence.
Require human model approval	A technically valid candidate may still be operationally unsuitable.	Unreviewed model changes reaching users.

Design principle When the evidence is incomplete, Fluventa prefers an explicit “limited monitoring” state over a confident guess.

15

What this prototype does—and does not—prove

The software behavior is real; the field claims are intentionally limited.

It demonstrates

A working end-to-end architecture, real model inference, data-quality decisions, incident lifecycles, slow-trend logic, dashboard views, and model governance.

It does not establish

Field-validated petroleum-pipeline performance, regulatory compliance, physical root-cause diagnosis, or safe control-system integration.

Its data

The included walkthrough uses deterministic synthetic readings so the same story can be repeated and inspected.

Its operating mode

A local Windows application for one portfolio checkpoint, bound to the local computer rather than exposed as a public service.

Before production use

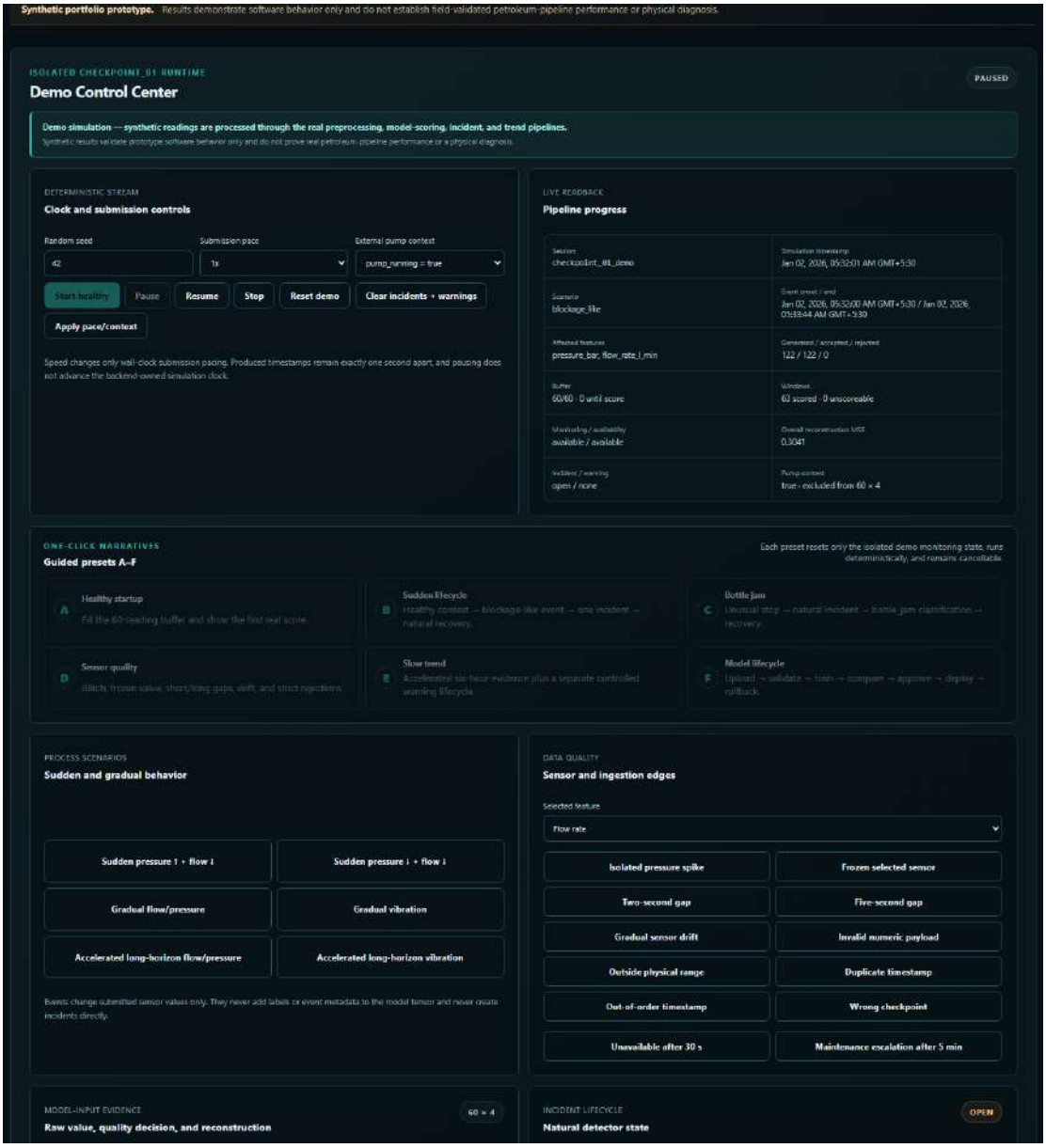
- Validate sensors, thresholds, and recovery behavior with site engineers and real historical conditions.
- Define approved operating envelopes for startup, shutdown, maintenance, and different flow regimes.
- Measure false alerts, missed events, detection time, and data-quality coverage over representative periods.
- Complete security, access control, audit retention, reliability, failover, and regulatory reviews.
- Integrate only through a formally approved operational and safety process.

Honest product statement Fluventa is an explainable portfolio prototype for anomaly-monitoring workflows. It is not a certified safety or automatic fault-diagnosis system.

16

A guided demonstration anyone can follow

The Demo Control Center uses repeatable scenarios so each behavior can be shown without hidden manual record creation.



The real Demo Control Center: controlled stream settings, guided presets, process scenarios, data-quality scenarios, and live feedback.

Six guided stories

Preset	Story to demonstrate	What to point out
A — Healthy startup	Build the 60-reading buffer and show the first score.	No score before enough context is expected.
B — Sudden lifecycle	Move from healthy operation to an event, then recovery.	One incident opens naturally and closes only after confirmed recovery.
C — Bottle jam	Run an unusual pattern and record a classification.	The person supplies “bottle jam”; the model supplies anomaly evidence.
D — Sensor quality	Show a glitch, frozen value, and short or long gaps.	Raw data and model input stay distinguishable; unscoreable windows are honest.
E — Slow trend	Accelerate the history needed for a slow-warning lifecycle.	Five-minute summaries and repeated evaluations remain separate from sudden incidents.
F — Model lifecycle	Train, compare, approve, deploy, and roll back.	Every active-model change is deliberate and recorded.

17

Three-minute presentation script

A concise way to explain Fluventa to a nontechnical audience.

Time	Suggested wording
0:00–0:30 — The need	“A checkpoint produces four readings every second. A person can view them, but it is hard to compare their combined pattern continuously. Fluventa acts as an early-warning assistant.”
0:30–1:00 — What it learns	“We train it only on approved healthy operation. It learns the normal one-minute relationship between pressure, temperature, flow, and vibration.”
1:00–1:30 — What a score means	“Every second, after the first minute, it compares the latest one-minute pattern with that healthy reference. A larger difference means unusual behavior—not a diagnosis.”
1:30–2:00 — Two operating paths	“A fast path can open one incident for sudden change. A slow path summarizes evidence and requires repeated support before opening a lower-priority warning.”
2:00–2:30 — Trust and data quality	“Short glitches may be cleaned only in the processed stream, while raw evidence is preserved. Long gaps suppress scoring, and missing evidence never proves recovery.”
2:30–3:00 — Human control	“Operators investigate and classify events. Engineers approve model changes. The prototype does not diagnose faults or control equipment; it makes monitoring evidence clearer and more traceable.”

Best live-demo ending Open the incident record, point to the score and sensor contributions, then show the human classification and the 30-step recovery counter. That single sequence communicates evidence, restraint, and accountability.

FAQ

Questions stakeholders are likely to ask

Short answers that preserve the project's actual boundaries.

Is this predicting a specific failure?

No. It detects behavior that differs from the learned healthy baseline. A person investigates the physical cause.

Why wait 60 seconds?

The model compares complete one-minute patterns. Before 60 readings, it does not have the required context.

Does one high score create many incidents?

No. One active sudden event is updated with continuing evidence rather than duplicated.

Can a single good reading close an incident?

No. Closure requires 30 consecutive valid results in the recovery range.

What if data is missing during recovery?

Recovery pauses. Absence of evidence is not counted as healthy evidence.

Why show which sensor contributed most?

It helps an investigator decide where to look first, but it is not proof of root cause.

Why is slow change handled separately?

A gradual trend needs longer, repeated evidence; treating it like a sudden event would create noise.

Can the application change the pipeline?

No. This prototype monitors and records; it does not control valves, pumps, or safety equipment.

Can the model update itself?

It can create a candidate through training, but a person must approve deployment. Rollback remains available.

Is it ready for a live petroleum pipeline?

No. It requires real-site validation, security and reliability work, and formal operational approval.

TAKEAWAY

The one sentence to remember

Fluventa compares the latest pipeline behavior with approved healthy behavior, turns persistent unusual evidence into traceable records, and leaves diagnosis and action with people.

EVIDENCE	RESTRAINT	ACCOUNTABILITY
Four sensor signals, their quality, and the one-minute pattern.	No score without enough context; no recovery without valid evidence; no automatic diagnosis.	Human classification, approval, notes, deployment, and rollback remain traceable.

Mini glossary

Term	Plain meaning
Healthy baseline	Approved examples of normal operation used as the reference.
Model score	A measure of how different the latest one-minute pattern is from that reference.
Threshold	A controlled decision level used by the operating rules.
Incident	A traceable record for sudden unusual behavior.
Slow warning	A separate lower-priority record for sustained trend evidence.
Data coverage	How much of the expected sensor history is present and usable.
Candidate model	A newly trained version awaiting human review.
Rollback	Returning to the previous approved model package.

Related technical document For implementation details, architecture decisions, edge conditions, file maps, and interview preparation, see “Fluventa Project & ML Architecture Interview Handbook.”

FLUVENTA ML · CHECKPOINT_01

Project & ML Architecture Interview Handbook

How the healthy-only LSTM autoencoder, sensor-quality pipeline, sudden incident path, slow-trend path, API, dashboard, persistence, and model lifecycle work—plus why each decision was made and what happens under edge conditions.

Document basis	Scope
Implementation	30 Python modules, 19 frontend source files, configuration and tests
Model artifact	baseline-lstm-autoencoder-v1; 11,280 trainable parameters
Runtime	1 Hz, 60 × 4 windows, one checkpoint, CPU inference
Evidence	Saved manifests, scaler, threshold files, and synthetic evaluation reports

CODE-VERIFIED EDITION · 08 AUGUST 2026

How to use this handbook

Read Chapters 1–7 first if you need to explain the ML system in an interview. Chapters 8–14 connect it to the API, UI, operations, security, and scale. The appendices provide a file map, a generated callable inventory, and 105 interview questions. Statements labeled ‘implemented’ come from code or artifacts; ‘rationale’ is an architectural explanation; ‘recommendation’ describes future production work.

Critical honesty rule This repository is a synthetic portfolio prototype. Its behavior is real software behavior, but its thresholds and performance are not field-validated petroleum-pipeline evidence.

Contents

1. Executive overview: what Fluventa actually does

Fluventa is a single-checkpoint pipeline-monitoring prototype. It learns what an approved healthy 60-second multivariate sensor sequence looks like. At runtime it tries to reconstruct each incoming 60-second window. A large mismatch means the current sequence does not resemble the healthy behavior learned during training. The system then interprets that mismatch through two separate rule-based paths: a fast sudden-incident path and a slower sustained-trend path.

The most important distinction The neural network produces reconstruction evidence; it does not classify 'blockage,' 'leak,' or 'mechanical failure.' Threshold and persistence state machines decide when to open operational records, while an engineer supplies interpretation.

Problem, users, and boundaries

Question	Project-specific answer
What problem?	Detect deviations from a checkpoint's learned healthy pressure, temperature, flow, and vibration behavior.
Primary users	Operators who monitor conditions and engineers who review evidence, configure thresholds, and manage models.
What is automated?	Quality processing, reconstruction scoring, threshold comparison, persistence, recovery, and local state persistence.
What remains human?	Healthy-data approval, model approval/deployment, incident interpretation, and physical diagnosis.
What is excluded?	Automatic root-cause diagnosis, safety shutdown, automatic retraining, field-reliability claims, and production identity controls.

One-minute explanation

Every second, the backend accepts one checkpoint_01 reading. It preserves the raw values, performs deterministic sensor-quality checks, creates a chronological 60 × 4 window, scales it with checkpoint-specific healthy medians and interquartile ranges, and runs a small LSTM autoencoder. The mean squared reconstruction error becomes an anomaly score. One valid score strictly above the spike threshold opens a single sudden incident. Independently, valid scores are summarized into five-minute medians; every 30 minutes a six-hour ordinary-least-squares slope is calculated, and five of the latest six valid slopes above threshold open a low-priority needs-attention warning. All state is persisted locally and presented by a React dashboard.

Fixed contract at a glance

Contract	Implemented value	Why it matters
Checkpoint scope	One separately trained model per checkpoint	Prevents different physical baselines from being mixed.
Sampling	1 reading/second	Makes each 60-row window exactly one minute.
Model features	Pressure, temperature, flow, vibration RMS	Continuous process signals suitable for reconstruction.
External context	pump_running	Visible to operators but excluded from the tensor and suppression logic.
Window	60 samples, stride 1	One-minute context and one new score per second after warm-up.
Score	Overall MSE + four per-feature MSE values	Supports one decision score and contribution indicators.

Contract	Implemented value	Why it matters
Sudden path	First valid score > spike; 30 valid scores < recovery	Fast opening with hysteretic closure.
Slow path	5-min median; 6-h OLS; 5 of 6	Suppresses brief spikes and requires persistence.

Evidence in this repository: *README.md; pipeline_anomaly_detection_plan/00_locked_decisions_and_status.md; src/pipeline_anomaly_detection/application_service.py*

Mentor checkpoint

Prompt	Takeaway
What you should remember	The model learns healthy sequence reconstruction; state machines convert error into operational records.
What interviewers care about	Explain the boundary between ML evidence, rule-based decisions, and human diagnosis.
Simple analogy	A skilled copyist can reproduce familiar handwriting; reconstruction mistakes reveal unfamiliar writing, but do not identify who wrote it.
Common confusion	Calling this a supervised fault classifier. It has no fault class output.
Quick recap	The model learns healthy sequence reconstruction; state machines convert error into operational records.

2. End-to-end architecture

Runtime inference path

Why the layers are separated

Layer	Responsibility	Input → output	Reason for separation
API schemas/routes	Transport validation and HTTP status	JSON → domain record	No ML logic is duplicated in controllers.
Application service	Order operations and guard shared state	Raw record → response	One orchestration point for API, demo, and restart reconciliation.
Preprocessor	Preserve raw evidence and produce model-safe values	Raw timeline → processed timeline	Data quality is deterministic and auditable.
Scaler/windowing	Apply the training coordinate system and sequence contract	Processed rows → 60 × 4 tensor	Prevents feature-scale domination and boundary leakage.
Model	Reconstruct learned healthy patterns	60 × 4 → 60 × 4	Keeps evidence generation independent from alert policy.
Sudden detector	Immediate incident lifecycle	Per-second MSE → incident state	Fast events need different timing from slow drift.
Trend detector	Persistent low-priority warning	Error history → slope → warning	Long-horizon evidence should ignore one-second spikes.
Stores	Restart-safe evidence and audit	Domain state → JSON	Portfolio simplicity without introducing a database.
Frontend	Presentation and engineer actions	Typed API → charts/forms	The browser never recreates decision formulas.

Training and deployment path

Deployment unit

The deployable unit is not just model_weights.pt. It is a compatible bundle: model weights, exact scaler, ordered feature list, window shape, sudden thresholds, slow-trend configuration, version metadata, and reload fixture. This prevents a mathematically valid model from being paired with the wrong coordinate system or decision policy.

Evidence in this repository: *src/pipeline_anomaly_detection/api.py; application_service.py:128; model_lifecycle.py; docs/model_lifecycle.md*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Transport, data quality, ML inference, decision policy, persistence, and presentation are distinct layers.
What interviewers care about	Show exactly where ML stops and deterministic domain logic begins.
Simple analogy	A laboratory instrument measures; a protocol decides when to raise an alarm; a clinician interprets it.
Common confusion	Describing FastAPI or React as part of the neural architecture.
Quick recap	Transport, data quality, ML inference, decision policy, persistence, and presentation are distinct layers.

3. ML architecture: how the LSTM autoencoder works

Why an autoencoder

An autoencoder receives an input and is trained to reproduce that same input. Fluventa trains only on approved healthy windows. The encoder compresses a one-minute sequence into a small latent representation; the decoder tries to rebuild the complete sequence. Healthy patterns should be easier to reconstruct than patterns not represented during training. The reconstruction error is therefore a deviation score—not a fault probability.

Design rationale (inferred from the implementation) Healthy data is available while reliable labeled examples of every real failure mode are not. A healthy-only reconstruction model avoids pretending the training set covers all faults. This is a sensible prototype choice, but it can also flag harmless operating changes and cannot name the cause.

Tensor and feature contract

Axis	Size	Meaning
Batch	B	Number of windows evaluated together.
Time	60	One sample per second for the most recent minute.
Features	4	Pressure, temperature, flow, vibration RMS—in this exact order.
Output	B × 60 × 4	Reconstruction with the same shape as the input.

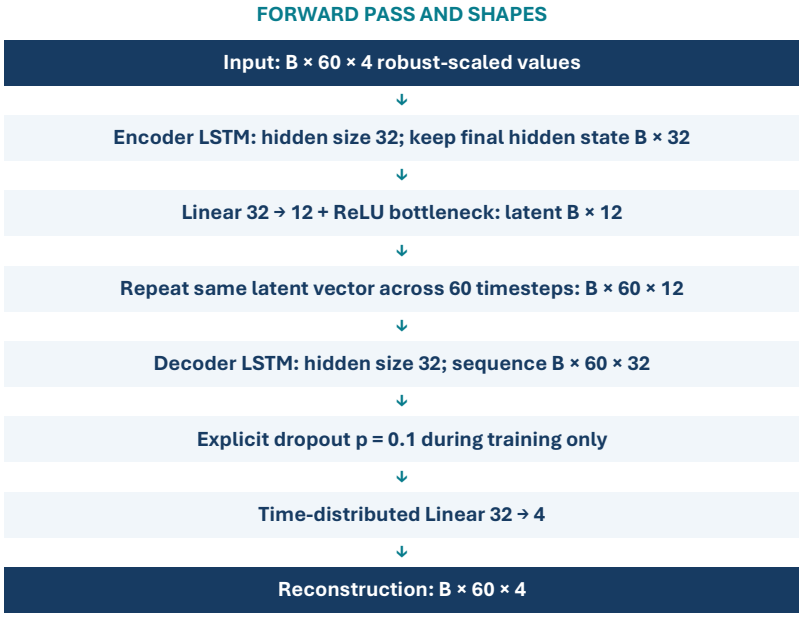
Robust scaling before the model

Each feature is transformed with $z_r = (x - \text{median}) / \text{IQR}$, where IQR is the 75th percentile minus the 25th percentile of approved healthy training rows. Robust scaling reduces the influence of extreme values and puts differently sized units into a comparable coordinate system. The scaler is fitted separately per checkpoint and only on the training split.

Feature	Healthy median	Healthy IQR	Provenance
pressure_bar	6.058976	0.274688	training only
temperature_c	31.786276	0.118225	training only
flow_rate_l_min	97.884875	6.162872	training only
vibration_velocity_rms_mm_s	2.460667	0.197141	training only

Why model and scaler are inseparable If the scaler changes, the same physical reading maps to a different numerical coordinate. The saved neural weights would then interpret the input incorrectly even though the tensor shape still matches.

Layer-by-layer architecture



What the LSTM contributes

A Long Short-Term Memory (LSTM) unit is a recurrent neural network cell with gates that control what to retain, add, and reveal. At each second, it combines the current four-feature vector with its previous state. This gives the encoder access to ordering and temporal relationships—such as pressure and flow moving together—not just a bag of values. The final hidden state summarizes the complete minute.

Internally, sigmoid gates produce values between 0 and 1. The forget gate controls retained memory; the input gate controls new candidate information; the output gate controls exposed state. This helps gradients travel across a 60-step sequence more reliably than a simple recurrent neural network, although it does not guarantee long-horizon understanding beyond the one-minute input.

Why repeat one latent vector

The bottleneck output is unsqueezed and expanded across 60 positions. The decoder therefore receives the same compressed summary at every step and must generate the full sequence from that summary. This deliberately forces information through the 12-value bottleneck. A skip connection would make reconstruction easier but could also let the network copy abnormal input too faithfully, weakening anomaly contrast.

Parameter count

Block	Trainable parameters	Explanation
Encoder LSTM (4 → 32)	4,864	Four gates, input/recurrent weights, and two bias vectors.
Linear bottleneck (32 → 12)	396	384 weights + 12 biases.
Decoder LSTM (12 → 32)	5,888	Four gates over 12 input channels and 32 hidden channels.
Output Linear (32 → 4)	132	128 weights + 4 biases.
Total	11,280	Small enough for fast CPU inference in this prototype.

Exact forward-pass code

src/pipeline_anomaly_detection/model.py:82

```
_, (hidden, _) = self.encoder(inputs)
latent = self.bottleneck(hidden[-1])
repeated = latent.unsqueeze(1).expand(-1, self.window_size, -1)
decoded, _ = self.decoder(repeated)
decoded = self.decoder_output_dropout(decoded)
return self.output_layer(decoded)
```

Reconstruction score

For feature f, per-feature MSE is the mean of $(x_t,f - \hat{x}_t,f)^2$ over the 60 timesteps. Overall MSE is the equal mean of the four per-feature MSE values. Because all values are robust-scaled first, no physical unit directly dominates merely because its numeric range is larger.

src/pipeline_anomaly_detection/model.py:104

```
squared_error = (inputs - reconstructions).pow(2)
per_feature = squared_error.mean(dim=1)   # B × 4
overall = per_feature.mean(dim=1)        # B
```

Why these choices—and credible alternatives

Decision	Why this project uses it	Tradeoff / alternative
Healthy-only autoencoder	Does not require labeled examples of every fault.	May detect benign novelty; supervised classification gives labels but requires trustworthy fault data.
LSTM	Models ordered temporal relationships across one minute.	1D CNN/TCN can be faster and parallel; Transformer is likely excessive for 60 steps and this dataset.
One layer	Small, understandable baseline with 11,280 parameters.	Deeper networks add capacity and overfitting risk; complexity must earn measurable improvement.
12-unit bottleneck	Forces compression while retaining enough capacity for four coupled signals.	Experimental value; too small underfits, too large may reconstruct anomalies.
Equal feature weighting	Simple, auditable, and matches locked specification.	Domain-weighted loss could prioritize safety-critical signals but requires justified weights.
MSE	Direct reconstruction objective; heavily penalizes large scaled errors.	MAE/Huber is more robust but could reduce sensitivity to sharp deviations.
Explicit decoder dropout	Regularizes one-layer model; LSTM internal dropout would be ineffective with one layer.	Recurrent dropout or weight decay are alternatives requiring comparison.
Checkpoint-specific model	Avoids mixing different sensor baselines and operating geometry.	A global conditional model reduces model count but needs checkpoint embeddings and broader validation.

Evidence in this repository: *src/pipeline_anomaly_detection/model.py; scaling.py; data/models/checkpoint_01/baseline_lstm_autoencoder_v1/manifest.json*

Mentor checkpoint

Prompt	Takeaway
What you should remember	The 11,280-parameter LSTM compresses 60 × 4 healthy behavior into 12 latent values and reconstructs it; MSE measures unfamiliarity.
What interviewers care about	Defend the choice using label scarcity, temporal coupling, simplicity, and explicit limitations—not 'LSTMs are powerful.'
Simple analogy	Compress a one-minute movie into a short note, then recreate every frame; unusual footage is harder to recreate.
Common confusion	Per-feature error is not a probability or root-cause attribution.

Prompt	Takeaway
Quick recap	The 11,280-parameter LSTM compresses 60 × 4 healthy behavior into 12 latent values and reconstructs it; MSE measures unfamiliarity.

4. Training pipeline and model selection

Temporal data boundary

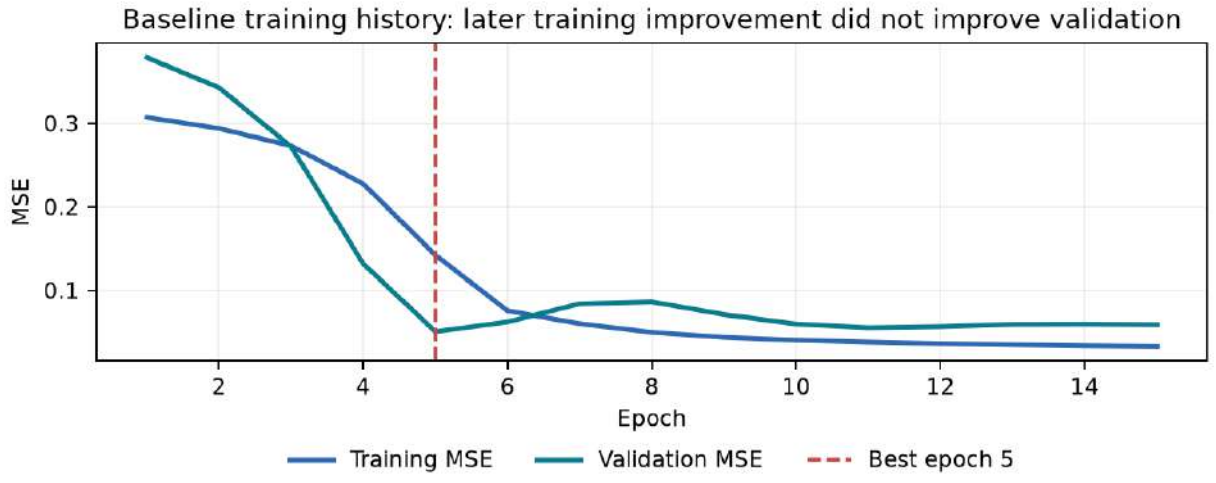
The checked-in baseline uses 1,200 deterministic healthy seconds: the first 900 seconds are training and the following 300 are validation. The raw split happens before preprocessing and window creation. Each side is processed independently, so no 60-second window crosses the boundary. This is crucial because adjacent stride-one windows overlap by 59 samples.

Interval	Raw rows	Windows	Use
Training 00:00:00–00:14:59	900	841	Scaler fit and gradient updates
Validation 00:15:00–00:19:59	300	241	Early stopping and model selection only

Leakage warning 841 overlapping windows are not 841 independent examples. The temporal boundary protects validation from direct overlap, but the entire dataset is still a small synthetic pipeline-verification run.

Training loop

1. Seed Python and PyTorch, use deterministic algorithms, and restrict CPU threads to one for the artifact run.
2. Convert training and validation windows to float32 tensors with exact shape N × 60 × 4.
3. Create the LSTM autoencoder; optimize MSE with Adam at learning rate 0.001.
4. Shuffle training windows with a seeded generator; never shuffle validation windows.
5. For each batch: zero gradients, reconstruct, compute MSE, backpropagate, clip global gradient norm to 1.0, and update weights.
6. Run validation in evaluation/inference mode so dropout is disabled.
7. Save every strictly lower validation-loss state. Separately, reset early-stopping patience only when improvement exceeds 1e-5.
8. Stop after 10 epochs without meaningful improvement; restore the absolute best saved state.
9. Recompute training and validation loss in evaluation mode and package the exact scaler and reload fixture.



Actual loss history from the saved baseline manifest. Best validation weights came from epoch 5; training stopped after epoch 15.

Why the best epoch and early-stop reference are separate

Any lower validation loss becomes the saved best state, even if the improvement is smaller than min_delta. However, only an improvement larger than min_delta resets patience. This preserves the numerically best candidate while still terminating when progress is too small to justify more epochs.

Saved baseline evidence

Metric	Value	Correct interpretation
Best epoch	5	Lowest healthy validation MSE in this run.
Epochs completed	15	Early stopping ended a 100-epoch maximum.
Restored train MSE	0.095486	Evaluation-mode reconstruction of training windows.
Restored validation MSE	0.050640	Healthy validation reconstruction only.
Validation p95	0.088748	Descriptive healthy distribution, not the runtime spike threshold.
CPU inference	0.276 ms/window	Artifact-run timing; not an end-to-end latency SLA.

What the negative validation–training gap means

The restored validation MSE is lower than restored training MSE, giving a negative gap. That does not prove the model generalizes perfectly. The later validation period may simply be easier, and the temporal process distribution can change. The stronger overfitting evidence is the trajectory: after epoch 5, training loss continued to fall while validation loss worsened, so later weights were rejected.

Training complexity

For N windows, T=60 timesteps, hidden size H, and E epochs, recurrent training is roughly O(E·N·T·H·(H+F)) plus decoder cost, with memory dominated by activations needed for backpropagation. Runtime inference is much smaller because it evaluates one or a small batch of windows and stores no gradients.

Evidence in this repository: *src/pipeline_anomaly_detection/training.py:82; training.py:250; docs/baseline_model.md; data/reports/checkpoint_01_baseline_evaluation.json*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Split raw time first, fit the scaler only on training, train deterministically, and restore the lowest validation-loss state.
What interviewers care about	Discuss leakage, window dependence, early stopping, and what the metrics do not prove.
Simple analogy	Practice on earlier days, audition on a later day, and never let audition material influence the measuring ruler.
Common confusion	Treating overlapping window count as independent sample size or validation MSE as anomaly accuracy.
Quick recap	Split raw time first, fit the scaler only on training, train deterministically, and restore the lowest validation-loss state.

5. Preprocessing and sensor-quality architecture

The preprocessor protects the ML boundary. It never edits a RawSensorRecord. Instead, it creates a processed timeline containing replacement values, quality flags, reasons, availability state, and auditable QualityDecision records. The raw evidence remains available for investigation.

Exact processing order

1. Quarantine wrong checkpoints, duplicate timestamps, disallowed out-of-order records, and non-integral sampling intervals.
2. Sort accepted readings and expand absent whole seconds into processed rows whose raw reference is null.
3. Mark missing, nonfinite, physical-range-invalid, and frozen values.
4. Apply a two-sided one-sample glitch rule using previous and next valid values.
5. Linearly interpolate missing runs of one to three seconds when both endpoints exist.
6. Compute checkpoint availability and transition flags.
7. Create split-safe scaled windows; reject any window containing incomplete processed values.

Isolated glitch logic

A candidate feature must jump beyond its configured absolute or relative threshold, and the next valid value must return close to the previous value. If fewer than two features support the same transient, the candidate is treated as a suspected single-sensor glitch and replaced in the processed stream with the midpoint of its neighbors. If two or more features change together, the event is preserved because correlated process motion is more likely than a lone sensor artifact.

src/pipeline_anomaly_detection/preprocessing.py:193

```
is_isolated_glitch = (
    returns_to_baseline
    and supporting_count < multi_feature_transient_min_features
)
replacement = (previous_value + next_value) / 2.0
```

Missing-data behavior

Condition	Processed behavior	ML effect
1–3 missing seconds, two-sided context	Linear interpolation; flag imputed; raw stays null	Windows remain scoreable.
>3 missing seconds	Values remain null; reason records limit violation	Every containing 60-second window is rejected.
Gap at batch edge	No two-sided context; no interpolation	Containing windows are rejected.
30 seconds without fully valid raw row	Availability becomes unavailable	State changes; scores may already be unavailable depending on window completeness.
300 seconds without fully valid raw row	Maintenance escalation	Operator-facing quality state; not an anomaly diagnosis.
Next fully valid raw reading	Counter resets and availability returns available	Does not retroactively invent missing raw data.

Frozen values

A feature held at exactly the same value for 30 contiguous seconds is marked frozen from the threshold-reaching row onward; its processed value becomes null. This blocks model scoring for windows containing the frozen row. Exact equality is a deterministic demo rule and must be revisited for real sensors with quantization and noise.

Physical ranges versus normal operating ranges

Normal ranges are chart context. Physical ranges are broad validity guards. A finite API value outside the physical range is accepted as raw evidence, then marked invalid by preprocessing instead of being rejected or clipped at the transport layer. This keeps one authoritative quality implementation and preserves what the sensor actually sent.

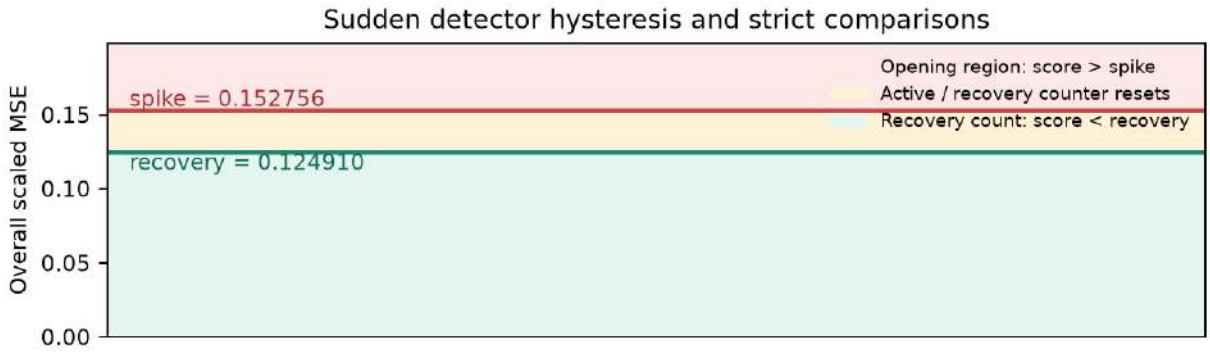
Evidence in this repository: *src/pipeline_anomaly_detection/preprocessing.py; windowing.py; config/healthy_generation.json; docs/preprocessing.md*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Raw evidence is immutable; processed values are explicitly flagged, and untrustworthy windows are suppressed rather than filled with zero.
What interviewers care about	Explain why a one-sensor spike can be cleaned while a multi-sensor transient is preserved.
Simple analogy	Keep the original lab sample sealed; perform corrections on a labeled working copy.
Common confusion	Saying the preprocessor 'fixes the raw data' or that missing scores equal healthy zero error.
Quick recap	Raw evidence is immutable; processed values are explicitly flagged, and untrustworthy windows are suppressed rather than filled with zero.

6. Sudden-anomaly path and incident state machine

The sudden path consumes each valid overall reconstruction MSE. It is intentionally simple: the first valid score strictly greater than the checkpoint spike threshold opens an incident immediately. While active, later scores update the same incident. Recovery requires 30 consecutive valid scores strictly below a lower recovery threshold.



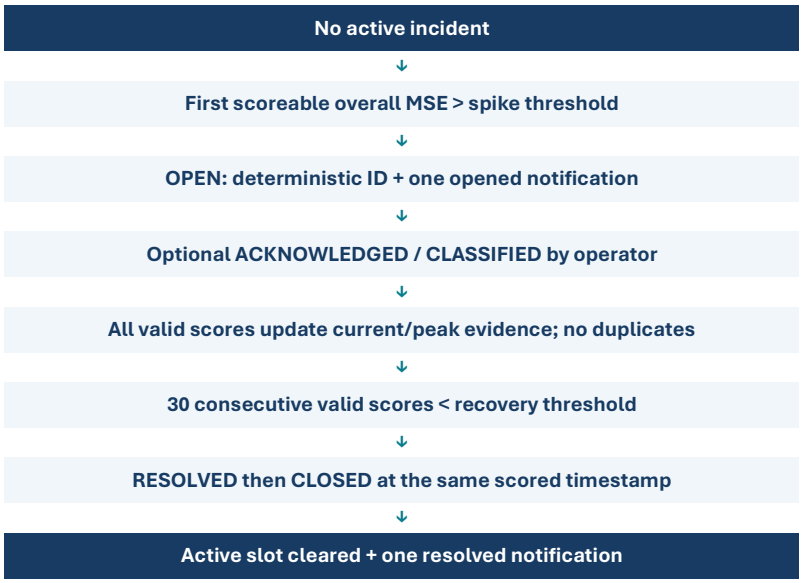
The two thresholds create hysteresis: opening and closing use different regions, preventing rapid on/off oscillation near one boundary.

Current provisional values

Setting	Value	Derivation	Status
Spike	0.152756208	p99.5 of 361 held-out healthy scores	Synthetic provisional
Recovery	0.124910451	Independent held-out healthy p95	Synthetic provisional
Recovery duration	30 valid seconds	Locked lifecycle rule	Implemented

State transitions

SUDDEN INCIDENT LIFECYCLE



Boundary conditions

Input condition	Exact result	Reason
score > spike, no active incident	Open incident	Strict first-crossing rule.
score = spike	Do not open	Opening condition is strictly greater-than.
more abnormal scores while active	Update same record	Avoid one incident per second.
score < recovery	Increment recovery counter	Valid evidence of normalization.
score = recovery	Reset counter to zero	Recovery condition is strictly less-than.
score > recovery	Reset counter to zero	Not continuously recovered.
unscoreable second	Counter unchanged	Missing evidence is not healthy evidence.
operator classifies bottle_jam	Interpretation changes; scoring continues	Classification cannot erase model evidence.
30th qualifying score	Resolve and close; notify once	Confirmed hysteretic recovery.

Why immediate opening but delayed closing

A sudden safety-relevant deviation should be visible quickly, so there is no opening persistence requirement. Closing is more conservative: a lower threshold plus 30 consecutive valid seconds prevents a noisy score from declaring recovery too early. This asymmetry is the operational meaning of hysteresis.

Why classification does not suppress scoring

Bottle jams, maintenance, and pump stops can produce unusual process behavior. Version 1 retains the incident and score stream even after classification, because an operator label describes context; it does not prove the system has returned to its learned healthy state. Automatic suppression would require a validated operating-state model that is intentionally out of scope.

Evidence in this repository: *src/pipeline_anomaly_detection/incidents.py:454; incidents.py:561; incidents.py:615; docs/sudden_detection.md*

Mentor checkpoint

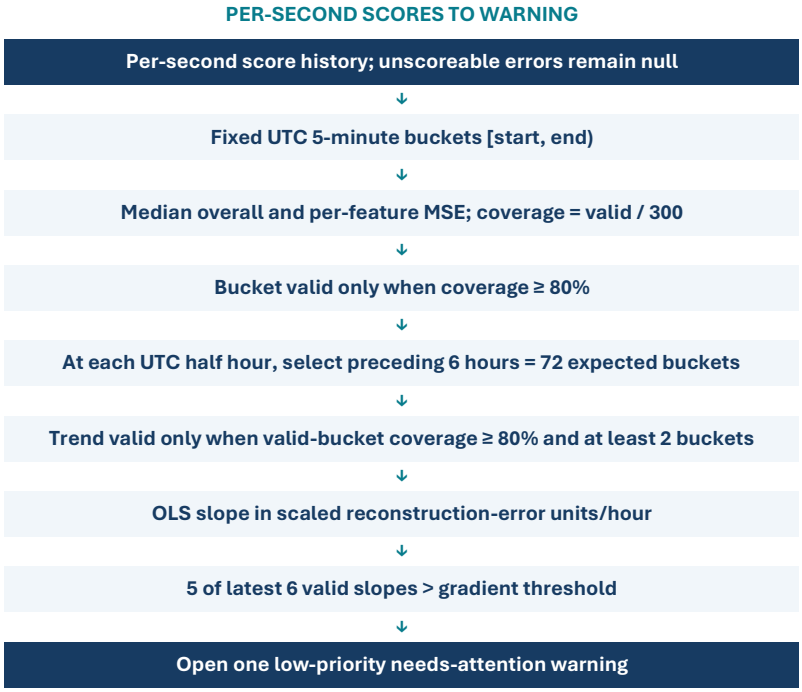
Prompt	Takeaway
What you should remember	Open fast on one strict crossing; close slowly after 30 valid strict-below scores; never count unavailable evidence as recovery.
What interviewers care about	State exact inequalities and explain deduplication, hysteresis, and operator-classification semantics.

Prompt	Takeaway
Simple analogy	A smoke alarm rings immediately but requires a sustained clear reading before declaring the room safe.
Common confusion	Saying a bottle_jam label closes or suppresses the incident.
Quick recap	Open fast on one strict crossing; close slowly after 30 valid strict-below scores; never count unavailable evidence as recovery.

7. Slow-degradation path

The slow path looks for a sustained rise in reconstruction error, not a single high point. It aggregates per-second evidence into robust five-minute medians, calculates a six-hour linear slope every 30 minutes, checks data coverage, and requires persistence across recent valid evaluations.

Mathematical pipeline



Why median, slope, and persistence

Mechanism	Purpose	Tradeoff
5-minute median	Suppresses brief spikes in the slow series.	Can delay and soften a genuine fast change; the sudden path covers that case.
6-hour OLS slope	Summarizes direction and rate across a configurable horizon.	Assumes one linear trend; nonlinear or cyclic behavior may be poorly summarized.
30-minute schedule	Limits noise and compute while providing repeated evidence.	A warning cannot react faster than the schedule/persistence allows.
5 of 6	Allows one dip while demanding sustained evidence.	Earliest opening needs six valid evaluations after full history exists.
Coverage gates	Prevents missing data from becoming a false trend.	Monitoring becomes limited instead of guessing.

Current configuration

Setting	Value	Operational consequence
Aggregation	300 s median	240 valid scores are insufficient? No: ≥80% requires at least 240/300; equality is valid.
Horizon	6 h	72 expected five-minute buckets.
Trend coverage	80%	At least 58 valid buckets because counts are integral and 57/72 = 79.17%.
Evaluation	Every 1,800 s	UTC half-hour alignment is enforced.
Opening	5 of latest 6 valid	Insufficient evaluations are excluded from the valid sequence.
Gradient threshold	10.828544 / h	Strict > comparison; synthetic provisional.
Recovery	3 valid evaluations ≤ recovery	At least 90 minutes of scheduled valid recovery evidence.

Insufficient-data behavior

If coverage is too low, the evaluation is stored as insufficient_trend_data with null gradient and no threshold decision. It does not enter the latest-six valid sequence, cannot open a warning, cannot advance the recovery counter, and cannot resolve an active warning. An active warning moves to monitoring_limited until a valid evaluation returns.

Slow warning lifecycle

Condition	Result
Fewer than six valid evaluations	Cannot open, even if all available values exceed threshold.
Exactly 5 of latest 6 are strictly above threshold	Open one warning.
Gradient equals opening threshold	Does not count as exceeded.
Repeated qualifying evaluations	Update the same active warning.
Insufficient data while active	Mark monitoring_limited; freeze recovery progress.
Valid gradient ≤ recovery threshold	Increment recovery evaluation counter.
Three consecutive valid recovery evaluations	Mark resolved and clear active slot.
Resolved warning	Not automatically closed; reviewed closure is a separate operator action.

Honest prototype result The two long-horizon injected scenarios exceeded the provisional threshold in only four evaluations, so the locked 5-of-6 rule opened no warning. The project reports this instead of tuning on the test events.

Evidence in this repository: *src/pipeline_anomaly_detection/trend_monitoring.py:314; trend_monitoring.py:395; trend_warnings.py:306; docs/slow_trend_monitoring.md*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Slow monitoring is median → coverage → six-hour OLS → 5-of-6 persistence; insufficient evidence never becomes zero.
What interviewers care about	Explain why fast and slow paths coexist and quantify earliest warning timing.

Prompt	Takeaway
Simple analogy	One rain drop triggers a fast alert; climate change needs repeated summaries across a long horizon.
Common confusion	Calling the slope a corrosion probability or filling missing errors with zero.
Quick recap	Slow monitoring is median → coverage → six-hour OLS → 5-of-6 persistence; insufficient evidence never becomes zero.

8. Complete runtime request walkthrough

Example: the 60th valid API reading arrives at 00:00:59Z after 59 buffered readings.

1. SensorReadingRequest requires exactly the documented JSON fields. Pydantic rejects strings/Booleans as sensor numbers, non-UTC or fractional-second timestamps, and non-Boolean pump state.
2. api_runtime_routes.ingest_reading converts the validated request into an immutable RawSensorRecord; no score or threshold logic exists in the route.
3. CheckpointApplicationService acquires an RLock, checks checkpoint_01, whole-second timestamp, finite features, and Boolean pump context.
4. JsonRuntimeBufferStore assigns a monotonic source_sequence, requires strictly increasing time, persists atomically, and retains at most the latest 60 raw readings.
5. SuddenRuntimeScorer reruns preprocessing for the current buffer. At 60 readings, create_sliding_windows can produce its first candidate ending at 00:00:59Z.
6. The preprocessor preserves raw values, applies quality rules, and yields processed features plus availability. Any long gap, invalid, or frozen row makes the candidate unscoreable.
7. RobustScaler transforms the four features in locked order using the bundled checkpoint medians/IQRs.
8. The model receives shape 1 × 60 × 4, reconstructs under torch.inference_mode(), and reconstruction_errors returns one overall plus four feature MSEs.
9. SuddenIncidentDetector appends the RuntimeScore. If score > spike and no active incident exists, it opens one. Otherwise it updates/recovery-checks the active record.
10. Only when the score timestamp is exactly aligned to the configured half-hour does the application rebuild five-minute buckets, evaluate the six-hour trend, and update the warning detector.
11. The response reports buffering/scored/unscoreable state, the score, active incident/warning, any scheduled trend evaluation, and notification events.
12. React polling later retrieves the same persisted state. It formats evidence and draws lines; it does not recalculate detector decisions.

What happens during the first 59 readings

The API accepts and persists each reading, but the response runtime_state is buffering and score is null because no complete 60-second context exists. There is no padding with zeros, duplication of early samples, or shortened window. At exactly 60 chronological processed rows, the first window becomes eligible.

Failure propagation

Failure boundary	Response/effect	Why
Request schema	HTTP 422 request_validation_error	Reject malformed transport before domain construction.
Runtime ordering/checkpoint	HTTP 422 runtime_validation_error	Preserve a strict checkpoint timeline.
Quality issue after acceptance	Accepted but score may be unscoreable	Raw physical evidence is valid transport even if unsafe for ML.
Historical incident admin action	HTTP 409 lifecycle_conflict	Only active records may change.

Failure boundary	Response/effect	Why
Missing record	HTTP 404	Differentiate absent resources from state conflict.
Model/config mismatch on startup	Fail initialization	Never score with an incompatible bundle.

Evidence in this repository: *src/pipeline_anomaly_detection/api_runtime_routes.py; api_schemas.py; application_service.py:204; sudden_runtime.py:109*

Mentor checkpoint

Prompt	Takeaway
What you should remember	One route creates a raw record; the application service orchestrates the existing domain modules under a lock.
What interviewers care about	Trace an exact request and name validation, buffering, preprocessing, model, detectors, persistence, and response.
Simple analogy	A parcel passes reception, inventory, inspection, analysis, policy, and reporting—each desk has one job.
Common confusion	Saying the API route or frontend computes the anomaly decision.
Quick recap	One route creates a raw record; the application service orchestrates the existing domain modules under a lock.

9. Operating-condition decision matrix

Condition	System decision	Detailed meaning
0–59 accepted readings	Buffer only	No score, incident, or trend evidence; readings_until_full_buffer counts down.
60 chronological scoreable readings	First score	One 60 × 4 window ends at the newest reading.
New reading after warm-up	Stride-one score	Oldest raw row leaves bounded buffer; newest window overlaps prior by 59 rows.
Wrong checkpoint	Reject	Portfolio runtime supports checkpoint_01 only.
Duplicate timestamp	Reject/quarantine	Timeline must be unique.
Out-of-order timestamp	Reject/quarantine	Tolerance is zero in current config.
Non-integral interval	Quarantine	Model contract assumes whole-second sampling.
Timestamp gap	Accept later reading and expand gap	Missingness is quality evidence, not an ordering error.
1–3 second gap with endpoints	Interpolate processed values	Raw remains missing; flags record imputation; window can score.
4+ second gap	Leave null and reject windows	Do not invent too much data.
Nonfinite API value	Reject at API	JSON/runtime numeric contract requires finite values.
Finite physical-range violation	Accept raw; mark processed invalid	Preserve evidence and keep one quality implementation.
30 repeated identical feature values	Mark frozen	Processed value becomes null; affected windows rejected.
One-sample single-feature jump then return	Suspected glitch + midpoint	Sensor-artifact pattern; raw preserved.
Same-time two-feature transient	Preserve	Correlated process event should reach the model.

Condition	System decision	Detailed meaning
Persistent two-sample deviation	Preserve	Not an isolated glitch.
pump_running=false	Store/display; still score	Pump context is outside model and suppression in v1.
Overall MSE > spike	Open/update incident	Strict comparison.
Overall MSE = spike	No opening	Equality does not cross.
Active incident + unscoreable second	Recovery pauses	No evidence is not healthy evidence.
Active incident + score = recovery	Recovery resets	Strictly below is required.
30th valid score below recovery	Resolve + close	One resolution notification; active slot cleared.
5-min bucket has 240/300 scores	Valid	Coverage equals 80%.
5-min bucket has 239/300 scores	Invalid	Coverage below 80%.
6-h trend has 57/72 valid buckets	Insufficient	79.17% < 80%; at least 58 required.
Gradient = opening threshold	Not exceeded	Opening uses strict >.
5 of latest 6 valid exceed	Open warning	One dip is permitted.
Insufficient evaluation while warning active	monitoring_limited	Does not count for opening/recovery.
3 valid gradients ≤ recovery	Resolve warning	Warning is not automatically archived/closed.
Operator classification	Add interpretation only	No evidence deletion, suppression, or forced closure.
Restart after buffer write before score write	Reconcile buffer	Backfills missing score deterministically.
Interrupted training job	training_failed on restart	No silent resume or automatic retraining.
Activation failure	Restore prior scorer/configs	Old active pointer remains valid.

Interview technique When asked ‘what happens if…’, answer in this order: acceptance at the API, raw preservation, processed quality state, scoreability, sudden recovery effect, slow-trend coverage effect, and what the operator sees.

Mentor checkpoint

Prompt	Takeaway
What you should remember	Conditions are explicit and deterministic; missing evidence pauses decisions instead of being interpreted as normal.
What interviewers care about	Senior interviewers test boundary inequalities, invalid data, startup, restart, and partial-failure behavior.
Simple analogy	Traffic rules define what happens at green, red, yellow, and a broken signal; the edge cases are the system.
Common confusion	Answering only the happy path.
Quick recap	Conditions are explicit and deterministic; missing evidence pauses decisions instead of being interpreted as normal.

10. Model lifecycle, persistence, and rollback

Manual lifecycle

1. Upload immutable UTF-8 CSV content with operator and reason.

2. Validate schema, checkpoint, UTC ordering, approximate 1 Hz coverage, values, missing percentages, and minimum duration.
3. Require explicit healthy_data_confirmed=true before training.
4. Serialize training in one background ThreadPoolExecutor worker while the active model keeps scoring.
5. Persist epoch progress, candidate metadata, scaler, thresholds, trend config, and reload verification.
6. Engineer approves or rejects with a reason; approval alone never changes runtime.
7. Deployment loads and validates the full candidate bundle, writes compatible configs, swaps all in-memory objects under the service lock, then commits the active pointer.
8. Rollback runs the same activation path for a previously deployed compatible version; it does not retrain.

Atomicity model

Persistence uses write-to-temporary-file followed by atomic replacement, with a small retry for Windows sharing violations. Activation is copy-on-write: prepare and validate a complete replacement, persist its configurations, swap scorer and detectors together under the application lock, and write the one active-version pointer last. On failure, restore the prior files and objects.

Artifact	Mutable?	Why
Original raw upload	No	Audit and reproducibility.
Candidate bundle	No after creation	Weights/scaler/config provenance stays coherent.
Runtime threshold copy	Yes, audited	Operator tuning without mutating protected source artifacts.
Incident/warning stores	Yes, append/update lifecycle	Operational state must survive restart.
Active-version pointer	Yes, atomic	Single source of deployment truth.

Why local JSON instead of a database

For one checkpoint, one local process, and a portfolio demo, JSON keeps the system portable, inspectable, and dependency-light. The tradeoff is limited concurrency, query performance, retention, and multi-user audit guarantees. A production multi-checkpoint deployment would likely move histories and lifecycle state to transactional storage while keeping the same domain contracts.

Evidence in this repository: *src/pipeline_anomaly_detection/model_lifecycle.py; atomic_io.py; application_service.py:805; docs/model_lifecycle.md*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Training never mutates the active model; deployment swaps a verified model+scaler+policy bundle atomically and rollback uses the same path.
What interviewers care about	Explain failure atomicity, active service continuity, and why approval is separate from deployment.
Simple analogy	Build a replacement engine beside the running one, inspect it, then swap the complete assembly—not individual parts.
Common confusion	Saying approval automatically deploys or rollback retrains.
Quick recap	Training never mutates the active model; deployment swaps a verified model+scaler+policy bundle atomically and rollback uses the same path.

11. Backend, frontend, and local application

Backend

FastAPI routes are divided into read, runtime ingestion, administration, and isolated demo modules. Pydantic supplies strict request models and OpenAPI. Routes call CheckpointApplicationService; they do not own preprocessing, model, threshold, or lifecycle formulas. The service uses an RLock to serialize state-changing operations in the single-process runtime.

Frontend

React 19 and TypeScript render seven hash-routed pages. A shared hook fetches fast operational data every five seconds, slow histories every 30 seconds, demo status every second, and stable metadata once. API calls use Promise.all where requests are independent. Existing data remains visible if a later refresh fails.

Page	Shows	Does not decide
Overview	Readings, normal ranges, pump context, monitoring state	Whether a value is anomalous.
Reconstruction	Overall/per-feature MSE and threshold lines	Incident opening or diagnosis.
Slow trend	Medians, gradients, coverage, backend 5-of-6 evidence	Browser-side warning state.
Incidents	Active/history, acknowledge/classify actions	Recovery or evidence deletion.
Configuration	Audited threshold and coverage forms	Final authority; backend validates relationships.
Model lifecycle	Upload, training, review, deploy, rollback	Automatic activation.
Demo control	Deterministic synthetic scenarios through real services	A second ML implementation.

One-click Windows architecture

The PyInstaller onedir build bundles CPU PyTorch, starts Uvicorn on 127.0.0.1, serves the production React build from the same FastAPI process, opens Edge in app mode, and stores mutable state under %LOCALAPPDATA%/Fluventa. Loopback-only binding reduces exposure of the unauthenticated portfolio API, but it is not a substitute for production identity and authorization.

Why hash routing

Routes such as #/slow-trend keep page selection after the # symbol, which the browser handles without asking the static server for a separate path. That avoids server-side single-page-application rewrite rules and another router dependency in the portable build.

Evidence in this repository: *src/pipeline_anomaly_detection/api.py; frontend/src/App.tsx; frontend/src/hooks/useDashboardData.ts; docs/windows_local_app.md*

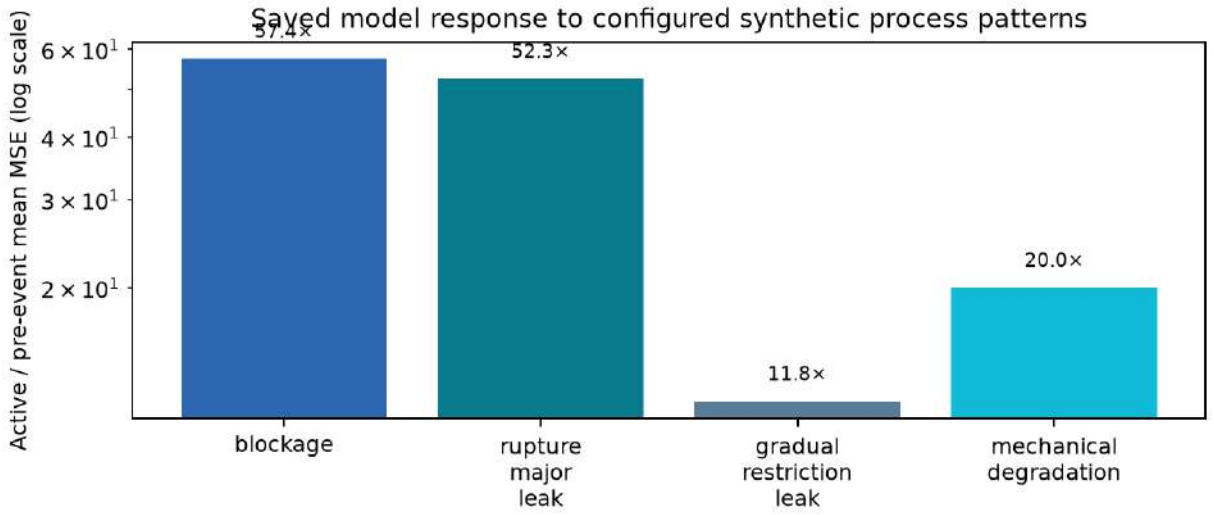
Mentor checkpoint

Prompt	Takeaway
What you should remember	The backend owns all decisions; React presents typed evidence and actions; the Windows app packages both into one loopback process.
What interviewers care about	Explain route/service separation and why no business rule is duplicated in the browser.
Simple analogy	The dashboard is an instrument panel, not the engine control algorithm.
Common confusion	Equating loopback binding with full authentication or saying charts generate alerts.

Prompt	Takeaway
Quick recap	The backend owns all decisions; React presents typed evidence and actions; the Windows app packages both into one loopback process.

12. Evaluation evidence and its limits

Offline evaluation loads the saved model and bundled scaler without retraining or refitting. It hashes protected artifacts before and after. Each scenario starts from a separate held-out healthy copy, injects configured patterns, runs the real preprocessing/window/model path, and exports per-window overall and per-feature errors.



Ratios compare active-event mean MSE with each scenario's pre-event mean. The log scale prevents the larger ratios from hiding smaller—but still increased—responses.

Configured process scenarios

Synthetic pattern	Affected features	Mean ratio	Peak MSE	Observed shape
blockage_like	pressure_bar, flow_rate_l_min	57.4×	8.602	0 s
rupture_major_leak_like	pressure_bar, flow_rate_l_min	52.3×	8.245	0 s
gradual_restriction_leak_like	pressure_bar, flow_rate_l_min	11.8×	4.054	rising slope
mechanical_degradation_like	vibration_velocity_rms_mm_s	20.0×	6.832	rising slope

Sensor-quality probes

Scenario	Expected/observed behavior
Isolated pressure spike	One suspected glitch; processed midpoint; no rejected windows.
45-second frozen flow	Frozen flags after configured duration; affected windows rejected.
2-second missing gap	Four features imputed for two rows; windows remain scoreable.
5-second missing gap	64 windows rejected; errors remain null, not zero.
Slow +3 °C drift	Persistent change reaches model; temperature error dominates.

Sudden lifecycle results

Scenario	Delay (s)	Peak MSE	Event incidents	Recovered
blockage_like	0.0	8.602	1	True
rupture_major_leak_like	0.0	8.245	1	True

Known false incident The provisional p99.5 spike threshold produced one incident during the complete held-out healthy replay. This is recorded evidence that the threshold is not production-ready.

What can and cannot be claimed

Supported claim	Unsupported claim
The code detects its configured synthetic sudden patterns immediately in this run.	The system detects real leaks with zero delay.
The saved model's error rises for the configured process probes.	Per-feature error proves physical root cause.
Persistence/recovery/restart logic works in deterministic tests.	The provisional thresholds have acceptable field false-alarm rate.
Protected artifacts remain unchanged during evaluation.	Synthetic validation equals production generalization.
The slow detector state machine opens in controlled replay.	The long-horizon injected scenarios passed the operational 5-of-6 rule—they did not.

Evidence in this repository: *docs/synthetic_evaluation.md; data/reports/checkpoint_01_synthetic_offline_evaluation.json; checkpoint_01_sudden_incident_evaluation.json; checkpoint_01_slow_trend_evaluation.json*

Mentor checkpoint

Prompt	Takeaway
What you should remember	Evaluation proves code-path behavior against configured synthetic probes, not real-world accuracy or diagnosis.
What interviewers care about	A strong candidate reports the false incident and failed 5-of-6 scenario instead of hiding them.
Simple analogy	A flight simulator can verify control software paths without certifying real aircraft safety.
Common confusion	Quoting synthetic ratios as precision, recall, reliability, or field performance.
Quick recap	Evaluation proves code-path behavior against configured synthetic probes, not real-world accuracy or diagnosis.

13. Design decisions and tradeoffs

Decision	Why	Cost	Alternative
One model per checkpoint	Prevents baseline mixing; simpler provenance	More models to train/store	Global conditional model with checkpoint embedding
pump_running external	Preserves context without teaching automatic suppression	Model may flag legitimate pump transitions	Operating-state-specific models after validation
60-second window	One minute of temporal context; low latency	Cannot directly learn six-hour behavior	Multi-resolution model or longer TCN

Decision	Why	Cost	Alternative
Stride one	Score every second	59/60 overlap and extra compute	Larger stride at cost of response time
Robust scaling	Median/IQR resist outliers and normalize units	Fails on near-zero IQR; not adaptive	Standard scaling, quantile transform, adaptive scaler with governance
Healthy-only MSE	Works without labeled faults; transparent score	Novel but benign states cause alerts	One-class models or supervised/multitask system
Immediate sudden opening	Fast visibility	Sensitive to single model-score excursions	k-of-n opening persistence
Lower recovery + 30 seconds	Hysteresis and stable closure	Delayed recovery	EWMA or probabilistic state model
Median slow buckets	Robust to short spikes	Ignores spike magnitude	Trimmed mean/quantile/EWMA
OLS slope	Simple, explainable trend units	Linear assumption; outlier sensitivity after aggregation	Theil–Sen, Mann–Kendall, change-point detection
5-of-6	Persistent but tolerates one dip	Minimum response delay	CUSUM, Bayesian persistence, configurable k-of-n
Atomic JSON	Portable and auditable for one local process	Weak for scale/concurrent queries	SQLite/PostgreSQL/event store
FastAPI/Pydantic	Typed validation and OpenAPI	Production auth still absent	Other web frameworks with equivalent contract
Native SVG charts	Small dependency surface and accessible labels	Less feature-rich than chart libraries	Recharts/ECharts/Plotly
Manual retraining/deploy	Engineer governance and rollback	Slower operational iteration	Automated pipeline with approval gates
PyInstaller onedir	Faster startup for large CPU PyTorch runtime	Whole folder must stay together	Installer, container, or one-file with extraction cost

How to defend a design decision

Use this structure: constraint → decision → mechanism → evidence → tradeoff → trigger to revisit. Example: reliable fault labels are scarce, so the project uses a healthy-only autoencoder. It reconstructs 60-second scaled windows and exposes MSE. Synthetic probes show response, but one healthy false incident proves threshold work remains. Revisit the approach when real labeled faults, multiple operating modes, or unacceptable false alarms become available.

Mentor checkpoint

Prompt	Takeaway
What you should remember	Every choice answers a constraint and carries a measurable cost; alternatives become appropriate when constraints change.
What interviewers care about	Tradeoff maturity matters more than defending every original choice as perfect.
Simple analogy	Choose the smallest tool that solves today's measured problem and know the signal that justifies replacement.
Common confusion	Answering ‘because it is industry standard’ without project-specific constraints.
Quick recap	Every choice answers a constraint and carries a measurable cost; alternatives become appropriate when constraints change.

14. Performance, scalability, security, and production gaps

Current performance

- The saved artifact reports about 0.276 ms of CPU model inference per validation window; preprocessing, JSON persistence, API overhead, and UI polling are outside that number.
- The model has 11,280 trainable parameters, so neural inference is not the obvious single-checkpoint bottleneck.
- The application currently reprocesses the full 60-reading buffer for each ingestion. This is bounded and simple, but a production stream could maintain incremental quality/window state.
- Five-minute buckets are derived from persisted per-second history; long histories in JSON will eventually dominate memory, parsing, and rewrite time.
- Demo long-horizon mode batches persistence writes because rewriting growing JSON per second was expensive; normal runtime retains safer per-update persistence.

Scale progression

Scale	Likely first constraint	Reasonable next move
1 checkpoint / demo	JSON rewrite and local process lifecycle	Current design is acceptable; monitor latency and file growth.
10–100 checkpoints	Per-reading reprocessing, model residency, serialized lock	Partition workers by checkpoint; incremental buffers; SQLite/Postgres; batch inference.
1,000 checkpoints	Ingestion throughput, scheduling, history volume	Message broker, horizontally scaled stateless API, model-worker pool, time-series storage.
10,000+ checkpoints	Model fleet/version management and storage I/O	Shard by checkpoint, centralized registry, object storage, autoscaling, observability.
1 million devices	Network, cardinality, deployment economics	Edge aggregation/scoring, regional ingestion, compact models, fleet governance; redesign required.

Security posture

Implemented protection	Gap before production
Strict typed requests, exact fields, finite numbers, UTC whole seconds	Authentication, authorization, device identity, replay protection
Read/runtime/admin routes separated in code	Enforced RBAC at gateway/service; separation alone is not authorization
Loopback-only packaged server	TLS and network controls for any remote deployment
20 MiB lifecycle upload cap and immutable source	Malware/content scanning, quotas, retention, tenant isolation
No weight-download route; safe metadata only	Artifact encryption/signing and supply-chain verification
Atomic local persistence and audit reasons	Tamper-evident centralized audit, backups, disaster recovery
No dynamic SQL	Database security becomes relevant when production storage is added
React escapes rendered text by default	CSP, dependency scanning, secure headers, CSRF strategy if cookie auth is introduced

Observability and debugging

Symptom	Check first	Likely cause
No score	buffered_reading_count and rejection reason	Warm-up below 60 or unscoreable window.
Too many sudden incidents	Threshold audit, healthy score distribution, model/scaler versions	Provisional threshold, drift, mismatch, or new operating mode.
Incident never closes	Recovery counter and scoreability	Scores equal/above recovery or missing evidence pauses counter.
No slow warning	Bucket/trend coverage and latest-six valid outcomes	Insufficient history/coverage or only four exceedances.
Unexpected dominant feature	Raw/processed values, scaler IQR, per-feature MSE	Quality issue or scale-sensitive reconstruction—not proof of cause.
Blank dashboard	API response schema and browser error boundary	Frontend/backend version mismatch or missing required status field.
Activation rejected	Package, scaler, feature order, window, config provenance	Incomplete or incompatible bundle.

Evidence in this repository: *docs/implementation_decisions.md; docs/backend_api.md; docs/windows_local_app.md; src/pipeline_anomaly_detection/windows_app.py*

Mentor checkpoint

Prompt	Takeaway
What you should remember	The small model is fast; persistence, history growth, checkpoint orchestration, and governance become the larger scaling problems.
What interviewers care about	Do not claim production security because routes are separate or the demo binds to loopback.
Simple analogy	A fast sensor is only one component; storage, traffic control, audit, and fleet operations determine system scale.
Common confusion	Jumping straight to microservices without naming the measured bottleneck.
Quick recap	The small model is fast; persistence, history growth, checkpoint orchestration, and governance become the larger scaling problems.

15. Folder and file map

Top-level folders

Path	Purpose	If removed
config/	Versioned healthy generation, preprocessing/model defaults, synthetic scenarios	The project loses reproducible configuration and scenario definitions.
data/artifacts/	Standalone robust scaler	Demo preprocessing/scaling examples and provenance break.
data/models/	Saved active baseline package	Runtime cannot load the model/scaler.
data/configuration/	Provisional sudden and trend source configs	Runtime cannot initialize detector policy.
data/evaluation/	Held-out/scenario datasets and reconstruction histories	Synthetic evidence is no longer reproducible/auditable.

Path	Purpose	If removed
data/reports/	Baseline, synthetic, sudden, and trend reports	Saved quantitative evidence disappears.
data/runtime/	Mutable buffer/incidents/warnings/lifecycle/logs	Runtime restarts clean and loses operational history; protected artifacts remain.
docs/	Behavioral and operational documentation/screenshots	Design intent and limitations become harder to audit.
examples/	Runnable training/evaluation/backend/demo entry points	Core library remains but guided execution paths disappear.
frontend/	React/TypeScript dashboard	Backend/ML continue, but no visual operator workspace.
packaging/	PyInstaller spec, build script, hooks, icon	No portable one-click Windows distribution.
pipeline_anomaly_detection_plan/	Locked requirements and phased source of truth	Implementation loses its requirements baseline.
src/pipeline_anomaly_detection/	Python implementation	Core system is removed.
tests/	Python unit/integration verification	Regression evidence and executable contracts disappear.

Python source modules

Module	Responsibility
__init__.py	Public package exports.
api.py	FastAPI application factory, error envelopes, route registration, optional static dashboard.
api_admin_routes.py	Administrative HTTP endpoints for incidents, warnings, configuration, and model lifecycle.
api_demo_routes.py	HTTP surface for isolated synthetic demo orchestration.
api_read_routes.py	Read-only status, history, incident, warning, configuration, and model metadata endpoints.
api_runtime_routes.py	Validated per-reading ingestion endpoint.
api_schemas.py	Strict Pydantic request/response contracts and validators.
application_service.py	Thread-safe orchestration of buffer, scorer, detectors, history, configuration, and lifecycle.
atomic_io.py	Atomic file replacement with bounded Windows sharing-violation retry.
config.py	Typed project, sensor, preprocessing, model, and generation configuration parsing.
demo_control.py	Cancellable deterministic demo stream and preset orchestration through real services.
incidents.py	Sudden score/incident models, JSON store, opening/deduplication/recovery state machine.
model.py	LSTM autoencoder and exact overall/per-feature reconstruction error.
model_lifecycle.py	Healthy upload, validation, background training, candidate review, activation registry.
model_package.py	Immutable package creation/loading and reload fixture verification.
preprocessing.py	Raw-preserving quality checks, glitch handling, interpolation, and availability state.
runtime_store.py	Bounded persistent raw-reading buffer.
scaling.py	Checkpoint-specific healthy-training-only robust scaler.

Module	Responsibility
schemas.py	Immutable domain records and enums for raw, processed, and evaluation data.
slow_trend_evaluation.py	Offline long-horizon synthetic trend evaluation and report export.
sudden_runtime.py	Saved-package runtime scoring and sudden lifecycle evaluation.
synthetic.py	Deterministic healthy data generation and CSV/metadata output.
synthetic_evaluation.py	Immutable synthetic injection, offline scoring, summaries, and artifact hashing.
thresholds.py	Sudden threshold provenance, validation, audit, and provisional derivation.
training.py	Temporal preparation, deterministic training, early stopping, and reconstruction evaluation.
trend_config.py	Slow-trend configuration, audit, validation, and provisional gradient derivation.
trend_monitoring.py	Five-minute medians, coverage, OLS gradients, and schedule evaluation.
trend_warnings.py	Persistent 5-of-6 warning and three-evaluation recovery state machine.
windowing.py	Strict 60 × 4 chronological scaled window construction and rejection.
windows_app.py	Portable local app paths, loopback server, Edge window, smoke test, and diagnostics.

Frontend source files

File	Responsibility
App.tsx	Shared shell, hash routing, demo/normal evidence selection, page composition.
api/client.ts	Typed fetch client, error normalization, endpoint methods, OpenAPI enum loading.
api/types.ts	TypeScript API/domain contracts and feature presentation metadata.
components/AppErrorBoundary.tsx	Recoverable application-level rendering error state.
components/ConfigurationPanel.tsx	Sudden/trend configuration forms and client relationship checks.
components/DemoControlPanel.tsx	Stream, scenarios, presets, evidence, and event-log controls.
components/IncidentPanel.tsx	Sudden incident evidence and operator actions.
components/LineChart.tsx	Dependency-light accessible SVG line chart.
components/ModelLifecyclePanel.tsx	Upload, validation, training, candidate review, deploy, rollback UI.
components/OperatorActions.tsx	Reusable acknowledgement/classification controls.
components/ReconstructionPanel.tsx	Overall/per-feature MSE and sudden threshold visualization.
components/SensorCharts.tsx	Recent sensor series with configured minimum Y domains.
components/SlowTrendPanel.tsx	Median, gradient, persistence, coverage, and warning visualization.
components/StatusOverview.tsx	Checkpoint status cards and operating context.
components/WarningPanel.tsx	Slow warning evidence and operator actions.
format.ts	Presentation-only number, time, label, status, and dominant-feature helpers.
hooks/useDashboardData.ts	Shared polling, parallel refresh, stale-data retention, and loading/error state.
main.tsx	React root bootstrap and error-boundary mount.
styles.css	Responsive dashboard visual system.

Configuration and artifact files that matter most

File	What it proves
config/healthy_generation.json	Locked features/window plus sensor quality, model, and generation settings.
data/models/.../manifest.json	Exact architecture, training provenance, best epoch, errors, timing, and reload fixture.
data/models/.../model_weights.pt	Selected PyTorch state dictionary.
data/models/.../scaler.json	Scaler bundled with the model package.
data/configuration/checkpoint_01_sudden_thresholds.json	Spike/recovery values and audit history.
data/configuration/checkpoint_01_slow_trend.json	Aggregation, horizon, coverage, thresholds, and persistence.
data/reports/checkpoint_01_*	Actual saved evaluation evidence and limitations.

Mentor checkpoint

Prompt	Takeaway
What you should remember	Source, immutable model artifacts, runtime state, evaluation evidence, UI, packaging, and requirements are deliberately separate.
What interviewers care about	Be able to point from a behavior to its implementation, configuration, test, and saved evidence.
Simple analogy	Blueprints, machinery, calibration certificates, logs, and control panels belong in different drawers.
Common confusion	Treating data/runtime as training data or configuration as neural weights.
Quick recap	Source, immutable model artifacts, runtime state, evaluation evidence, UI, packaging, and requirements are deliberately separate.

16. Complete Python callable inventory

This appendix is generated directly from the current Python abstract syntax trees. It lists all 435 module functions and class methods, including private helpers. ‘Direct calls’ and ‘referenced by’ are conservative name-based static indicators; dynamic dispatch can make the runtime graph broader. Complexity excludes delegated neural, filesystem, or network work unless stated.

api.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
create_app **L29**	Create an API with injectable services and optional bundled dashboard.	(service: CheckpointApplicationService | None=None, *, runtime_root: str | Path | None=None, demo_controller: DemoCont...) → FastAPI	O(1) excluding delegated I/O/model work	CheckpointApplicationService, DemoController, FastAPI, JSONResponse, Path, StaticFiles, ValueError, errors...	windows_app.py:create_local_application

api_admin_routes.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_service **L26**	Callable for get service in api_admin_routes.py.	(request: Request) → CheckpointApplicationService	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
acknowledge_incident **L34**	Callable for acknowledge incident in api_admin_routes.py.	(incident_id: str, request: OperatorActionRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	acknowledge_incident, post	api_admin_routes.py:acknowledge_incident
classify_incident **L43**	Callable for classify incident in api_admin_routes.py.	(incident_id: str, request: ClassificationRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	classify_incident, post	api_admin_routes.py:classify_incident, demo_control.py:DemoController.classify_bottle_jam
acknowledge_warning **L56**	Callable for acknowledge warning in api_admin_routes.py.	(warning_id: str, request: OperatorActionRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	acknowledge_warning, post	api_admin_routes.py:acknowledge_warning
classify_warning **L65**	Callable for classify warning in api_admin_routes.py.	(warning_id: str, request: ClassificationRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	classify_warning, post	api_admin_routes.py:classify_warning
update_sudden_configuration **L78**	Callable for update sudden configuration in api_admin_routes.py.	(request: SuddenThresholdUpdateRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	patch, update_sudden_thresholds	entry point / dynamic / unreferenced statically
update_trend_configuration **L91**	Callable for update trend configuration in api_admin_routes.py.	(request: TrendConfigurationUpdateRequest, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	patch, update_trend_configuration	api_admin_routes.py:update_trend_configuration
upload_healthy_dataset **L107**	Callable for upload healthy dataset in api_admin_routes.py.	(request: HealthyDatasetUploadRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, upload_healthy_dataset	api_admin_routes.py:upload_healthy_dataset, demo_control.py:DemoController._run_model_lifecycle_demo
validate_healthy_dataset **L119**	Callable for validate healthy dataset in api_admin_routes.py.	(dataset_id: str, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, validate_healthy_dataset	api_admin_routes.py:validate_healthy_dataset, demo_control.py:DemoController._run_model_lifecycle_demo
start_model_training **L124**	Callable for start model training in api_admin_routes.py.	(request: ModelTrainingRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	float, post, start_model_training	api_admin_routes.py:start_model_training, demo_control.py:DemoController._run_model_lifecycle_demo

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_model_training_status L138	Callable for get model training status in api_admin_routes.py.	(job_id: str, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, model_training_status	entry point / dynamic / unreferenced statically
get_model_training_progress L143	Callable for get model training progress in api_admin_routes.py.	(job_id: str, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, model_training_status	entry point / dynamic / unreferenced statically
list_model_versions L154	Callable for list model versions in api_admin_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, list_model_versions	api_admin_routes.py:list_model_versions, demo_control.py:DemoController._run_model_lifecycle_demo
get_model_candidate L159	Callable for get model candidate in api_admin_routes.py.	(candidate_id: str, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, get_model_candidate	api_admin_routes.py:get_model_candidate
compare_model_candidate L164	Callable for compare model candidate in api_admin_routes.py.	(candidate_id: str, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	compare_model_candidate, get	api_admin_routes.py:compare_model_candidate, demo_control.py:DemoController._run_model_lifecycle_demo
approve_model_candidate L169	Callable for approve model candidate in api_admin_routes.py.	(candidate_id: str, request: ModelLifecycleActionRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, review_model_candidate	entry point / dynamic / unreferenced statically
reject_model_candidate L181	Callable for reject model candidate in api_admin_routes.py.	(candidate_id: str, request: ModelLifecycleActionRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, review_model_candidate	entry point / dynamic / unreferenced statically
deploy_model_candidate L193	Callable for deploy model candidate in api_admin_routes.py.	(candidate_id: str, request: ModelLifecycleActionRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	deploy_model_candidate, post	api_admin_routes.py:deploy_model_candidate, demo_control.py:DemoController._run_model_lifecycle_demo
rollback_model_version L202	Callable for rollback model version in api_admin_routes.py.	(version_id: str, request: ModelLifecycleActionRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, rollback_model_version	api_admin_routes.py:rollback_model_version, demo_control.py:DemoController._run_model_lifecycle_demo

api_demo_routes.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_demo_controller L22	Callable for get demo controller in api_demo_routes.py.	(request: Request) → DemoController	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
demo_status L30	Callable for demo status in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, status	entry point / dynamic / unreferenced statically
reset_demo L35	Callable for reset demo in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, reset	entry point / dynamic / unreferenced statically
clear_demo_records L40	Callable for clear demo records in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	clear_records, post	entry point / dynamic / unreferenced statically
start_demo_stream L45	Callable for start demo stream in api_demo_routes.py.	(body: DemoStreamStartRequest, controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, start	entry point / dynamic / unreferenced statically
pause_demo_stream L54	Callable for pause demo stream in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	pause, post	entry point / dynamic / unreferenced statically
resume_demo_stream L59	Callable for resume demo stream in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, resume	entry point / dynamic / unreferenced statically

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
stop_demo_stream L64	Callable for stop demo stream in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, stop	entry point / dynamic / unreferenced statically
step_demo_stream L69	Callable for step demo stream in api_demo_routes.py.	(body: DemoStepRequest, controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, step	entry point / dynamic / unreferenced statically
update_demo_settings L76	Callable for update demo settings in api_demo_routes.py.	(body: DemoSettingsRequest, controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	patch, settings	entry point / dynamic / unreferenced statically
trigger_demo_scenario L85	Callable for trigger demo scenario in api_demo_routes.py.	(scenario_id: str, body: DemoScenarioRequest, controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, trigger_scenario	entry point / dynamic / unreferenced statically
cancel_demo_preset L96	Callable for cancel demo preset in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	cancel_preset, post	entry point / dynamic / unreferenced statically
run_demo_preset L101	Callable for run demo preset in api_demo_routes.py.	(preset_id: str, controller: Controller, body: DemoPresetRequest | None=None) → dict[str, Any]	O(1) excluding delegated I/O/model work	post, run_preset	entry point / dynamic / unreferenced statically
run_controlled_warning_lifecycle L114	Callable for run controlled warning lifecycle in api_demo_routes.py.	(controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	controlled_warning_lifecycle, post	entry point / dynamic / unreferenced statically
classify_demo_bottle_jam L119	Callable for classify demo bottle jam in api_demo_routes.py.	(incident_id: str, controller: Controller) → dict[str, Any]	O(1) excluding delegated I/O/model work	classify_bottle_jam, post	entry point / dynamic / unreferenced statically

api_read_routes.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_service L24	Callable for get service in api_read_routes.py.	(request: Request) → CheckpointApplicationService	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
checkpoint_status L32	Return current sensor, scoring, incident, warning, and version status.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	checkpoint_status, get	api_read_routes.py:checkpoint_status, demo_control.py:DemoController.status
reconstruction_history L39	Return a filtered per-second, five-minute, or gradient series.	(service: Service, series: Literal['per_second', 'five_minute', 'gradient']=Query(default='per_second'), start: dateti…) → dict[str, Any]	O(1) excluding delegated I/O/model work	Query, get, reconstruction_history	api_read_routes.py:reconstruction_history, demo_control.py:DemoController.status
list_incidents L56	Callable for list incidents in api_read_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, len, list_incidents	api_read_routes.py:list_incidents, demo_control.py:DemoController.status
active_incident L62	Callable for active incident in api_read_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	active_incident, get	api_read_routes.py:active_incident, demo_control.py:DemoController._process_point
get_incident L67	Callable for get incident in api_read_routes.py.	(incident_id: str, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	get, get_incident	api_read_routes.py:get_incident
list_warnings L72	Callable for list warnings in api_read_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, len, list_warnings	api_read_routes.py:list_warnings, demo_control.py:DemoController.status
active_warning L78	Callable for active warning in api_read_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	active_warning, get	api_read_routes.py:active_warning

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_warning **L83**	Callable for get warning in api_read_routes.py.	(warning_id: str, service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	get, get_warning	api_read_routes.py:get_warning
sudden_configuration **L88**	Callable for sudden configuration in api_read_routes.py.	(service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	get, get_sudden_thresholds	entry point / dynamic / unreferenced statically
trend_configuration **L93**	Callable for trend configuration in api_read_routes.py.	(service: Service) → dict[str, object]	O(1) excluding delegated I/O/model work	get, get_trend_configuration	entry point / dynamic / unreferenced statically
model_information **L98**	Callable for model information in api_read_routes.py.	(service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, model_information	api_read_routes.py:model_information

api_runtime_routes.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
get_service **L17**	Callable for get service in api_runtime_routes.py.	(request: Request) → CheckpointApplicationService	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ingest_reading **L25**	Validate and ingest one raw checkpoint_01 sensor reading.	(request: SensorReadingRequest, service: Service) → dict[str, Any]	O(1) excluding delegated I/O/model work	RawSensorRecord, ingest_reading, post	api_runtime_routes.py:ingest_reading, demo_control.py:DemoController._process_point

api_schemas.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_validate_utc_second **L25**	Internal helper for validate utc second in api_schemas.py.	(value: datetime) → datetime	O(1) excluding delegated I/O/model work	ValueError, timedelta, utcoffset	entry point / dynamic / unreferenced statically
SensorReadingRequest.require_json_number **L52**	Callable for require json number in api_schemas.py.	(cls, value: Any) → Any	O(1) excluding delegated I/O/model work	ValueError, field_validator, isinstance	entry point / dynamic / unreferenced statically
SuddenThresholdUpdateRequest.require_update **L76**	Callable for require update in api_schemas.py.	(self) → 'SuddenThresholdUpdateRequest'	O(1) excluding delegated I/O/model work	ValueError, model_validator	entry point / dynamic / unreferenced statically
SuddenThresholdUpdateRequest.require_json_number_or_null **L83**	Callable for require json number or null in api_schemas.py.	(cls, value: Any) → Any	O(1) excluding delegated I/O/model work	ValueError, field_validator, isinstance	entry point / dynamic / unreferenced statically
TrendConfigurationUpdateRequest.require_update **L104**	Callable for require update in api_schemas.py.	(self) → 'TrendConfigurationUpdateRequest'	O(N) typical	ValueError, all, model_validator	entry point / dynamic / unreferenced statically
TrendConfigurationUpdateRequest.require_json_number_or_null **L125**	Callable for require json number or null in api_schemas.py.	(cls, value: Any) → Any	O(1) excluding delegated I/O/model work	ValueError, field_validator, isinstance	entry point / dynamic / unreferenced statically
DemoSettingsRequest.require_demo_setting **L166**	Callable for require demo setting in api_schemas.py.	(self) → 'DemoSettingsRequest'	O(1) excluding delegated I/O/model work	ValueError, model_validator	entry point / dynamic / unreferenced statically

application_service.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
ApplicationServiceError.__init__ **L47**	Initializes the object and validates/loads required collaborators.	(self, message: str) → None	O(1) excluding delegated I/O/model work	__init__, super	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
ServicePaths.for_project **L77**	Callable for for project in application_service.py.	(cls, project_root: str | Path, *, runtime_root: str | Path | None=None) → 'ServicePaths'	O(1) excluding delegated I/O/model work	Path, cls, resolve	api.py:create_app, windows_app.py:create_local_application
ServicePaths.runtime_sudden_thresholds **L103**	Callable for runtime sudden thresholds in application_service.py.	(self) → Path	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ServicePaths.runtime_trend_config **L107**	Callable for runtime trend config in application_service.py.	(self) → Path	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ServicePaths.runtime_buffer **L111**	Callable for runtime buffer in application_service.py.	(self) → Path	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ServicePaths.incident_store **L115**	Callable for incident store in application_service.py.	(self) → Path	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ServicePaths.warning_store **L119**	Callable for warning store in application_service.py.	(self) → Path	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
CheckpointApplicationService.__init__ **L128**	Initializes the object and validates/loads required collaborators.	(self, paths: ServicePaths) → None	O(1) excluding delegated I/O/model work	JsonIncidentStore, JsonRuntimeBufferStore, JsonTrendWarningStore, ModelLifecycleManager, RLock, SlowTrendWarningDetector, SuddenIncidentDetector, SuddenRuntimeScorer…	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
CheckpointApplicationService._initialize_runtime_configurations **L171**	Internal helper for initialize runtime configurations in application_service.py.	(self) → None	O(1) excluding delegated I/O/model work	exists, load, save	application_service.py:CheckpointApplicationService.__init__
CheckpointApplicationService._validate_package_compatibility **L181**	Internal helper for validate package compatibility in application_service.py.	(self) → None	O(N) typical	ValueError, str	application_service.py:CheckpointApplicationService.__init__
CheckpointApplicationService.ingest_reading **L204**	Accept one raw reading and orchestrate the existing runtime paths.	(self, record: RawSensorRecord) → dict[str, Any]	O(N) typical	RuntimeValidationError, _active_incident_dict, _active_warning_dict, _latest_preprocessing_state, _score_buffer_and_process_new_results, _score_dict, _validate_runtime_record, append…	api_runtime_routes.py:ingest_reading, demo_control.py:DemoController._process_point
CheckpointApplicationService._reconcile_persisted_buffer **L262**	Backfill a buffer saved immediately before an interrupted score write.	(self) → None	O(1) excluding delegated I/O/model work	_score_buffer_and_process_new_results	application_service.py:CheckpointApplicationService.__init__
CheckpointApplicationService._score_buffer_and_process_new_results **L268**	Internal helper for score buffer and process new results in application_service.py.	(self) → tuple[list[RuntimeScore], list[dict[str, object]], TrendEvaluation | None]	O(N·M) / data-dependent loops	_evaluate_slow_trend_if_scheduled, extend, process_score, score_records, to_dict	application_service.py:CheckpointApplicationService._reconcile_persisted_buffer, application_service.py:CheckpointApplicationService.ingest_reading

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
CheckpointApplicationService._validate_runtime_record L292	Internal helper for validate runtime record in application_service.py.	(self, record: RawSensorRecord) → None	O(N) typical	RuntimeValidationError, getattr, isfinite, isinstance	application_service.py:CheckpointApplicationService.ingest_reading
CheckpointApplicationService._latest_preprocessing_state L308	Internal helper for latest preprocessing state in application_service.py.	(self) → str	O(1) excluding delegated I/O/model work	SensorPreprocessor, process	application_service.py:CheckpointApplicationService.checkpoint_status, application_service.py:CheckpointApplicationService.ingest_reading
CheckpointApplicationService._evaluate_slow_trend_if_scheduled L318	Internal helper for evaluate slow trend if scheduled in application_service.py.	(self, score: RuntimeScore) → TrendEvaluation | None	O(1) excluding delegated I/O/model work	_reconstruction_history_records, aggregate_five_minute_medians, evaluate_trend_at, int, process_evaluation, timestamp	application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results
CheckpointApplicationService._reconstruction_history_records L337	Internal helper for reconstruction history records in application_service.py.	(self) → tuple[ReconstructionHistoryRecord, ...]	O(N) typical	from_runtime_score, str, tuple	application_service.py:CheckpointApplicationService._evaluate_slow_trend_if_scheduled, application_service.py:CheckpointApplicationService.reconstruction_history
CheckpointApplicationService.checkpoint_status L348	Callable for checkpoint status in application_service.py.	(self) → dict[str, Any]	O(N) typical	_active_incident_dict, _active_warning_dict, _latest_preprocessing_state, checkpoint, format_utc, len, model_features	api_read_routes.py:checkpoint_status, demo_control.py:DemoController.status
CheckpointApplicationService._score_dict L417	Internal helper for score dict in application_service.py.	(self, score: RuntimeScore) → dict[str, Any]	O(1) excluding delegated I/O/model work	to_dict	application_service.py:CheckpointApplicationService.ingest_reading, application_service.py:CheckpointApplicationService.reconstruction_history
CheckpointApplicationService.reconstruction_history L423	Callable for reconstruction history in application_service.py.	(self, series: str, *, start: datetime | None=None, end: datetime | None=None, limit: int=1000) → dict[str, Any]	O(N·M) / data-dependent loops	RuntimeValidationError, _reconstruction_history_records, _require_utc, _score_dict, aggregate_five_minute_medians, format_utc, len, to_dict	api_read_routes.py:reconstruction_history, demo_control.py:DemoController.status
CheckpointApplicationService.list_incidents L475	Callable for list incidents in application_service.py.	(self) → list[dict[str, object]]	O(N log N) typical	sorted, to_dict, values	api_read_routes.py:list_incidents, demo_control.py:DemoController.status
CheckpointApplicationService.get_incident L484	Callable for get incident in application_service.py.	(self, incident_id: str) → dict[str, object]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, get, to_dict	api_read_routes.py:get_incident
CheckpointApplicationService.active_incident L490	Callable for active incident in application_service.py.	(self) → dict[str, object] | None	O(1) excluding delegated I/O/model work	_active_incident_dict	api_read_routes.py:active_incident, demo_control.py:DemoController._process_point
CheckpointApplicationService._active_incident_dict L493	Internal helper for active incident dict in application_service.py.	(self) → dict[str, object] | None	O(1) excluding delegated I/O/model work	to_dict	application_service.py:CheckpointApplicationService.active_incident, application_service.py:CheckpointApplicationService.checkpoint_status, application_service.py:CheckpointApplicationService.ingest_reading
CheckpointApplicationService.list_warnings L497	Callable for list warnings in application_service.py.	(self) → list[dict[str, object]]	O(N log N) typical	sorted, to_dict, values	api_read_routes.py:list_warnings, demo_control.py:DemoController.status
CheckpointApplicationService.get_warning L506	Callable for get warning in application_service.py.	(self, warning_id: str) → dict[str, object]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, get, to_dict	api_read_routes.py:get_warning
CheckpointApplicationService.active_warning L512	Callable for active warning in application_service.py.	(self) → dict[str, object] | None	O(1) excluding delegated I/O/model work	_active_warning_dict	api_read_routes.py:active_warning
CheckpointApplicationService._active_warning_dict L515	Internal helper for active warning dict in application_service.py.	(self) → dict[str, object] | None	O(1) excluding delegated I/O/model work	to_dict	application_service.py:CheckpointApplicationService.active_warning, application_service.py:CheckpointApplicationService.checkpoint_status, application_service.py:CheckpointApplicationService.ingest_reading

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
CheckpointApplicationService.acknowledge_incident L519	Callable for acknowledge incident in application_service.py.	(self, incident_id: str, *, timestamp: datetime, operator: str) → dict[str, object]	O(1) excluding delegated I/O/model work	_require_active_incident, _require_utc, acknowledge_active, to_dict	api_admin_routes.py:acknowledge_incident
CheckpointApplicationService.classify_incident L529	Callable for classify incident in application_service.py.	(self, incident_id: str, classification: OperatorClassification, *, note: str, timestamp: datetime, operator: str) → dict[str, object]	O(1) excluding delegated I/O/model work	_require_active_incident, _require_utc, classify_active, to_dict	api_admin_routes.py:classify_incident, demo_control.py:DemoController.classify_bottle_jam
CheckpointApplicationService.acknowledge_warning L548	Callable for acknowledge warning in application_service.py.	(self, warning_id: str, *, timestamp: datetime, operator: str) → dict[str, object]	O(1) excluding delegated I/O/model work	_require_active_warning, _require_utc, acknowledge_active, to_dict	api_admin_routes.py:acknowledge_warning
CheckpointApplicationService.classify_warning L558	Callable for classify warning in application_service.py.	(self, warning_id: str, classification: OperatorClassification, *, note: str, timestamp: datetime, operator: str) → dict[str, object]	O(1) excluding delegated I/O/model work	_require_active_warning, _require_utc, mark_active_under_review, to_dict	api_admin_routes.py:classify_warning
CheckpointApplicationService._require_active_incident L577	Internal helper for require active incident in application_service.py.	(self, incident_id: str) → SuddenIncident	O(1) excluding delegated I/O/model work	LifecycleConflictError, ResourceNotFoundError	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.classify_incident
CheckpointApplicationService._require_active_warning L587	Internal helper for require active warning in application_service.py.	(self, warning_id: str) → SlowTrendWarning	O(1) excluding delegated I/O/model work	LifecycleConflictError, ResourceNotFoundError	application_service.py:CheckpointApplicationService.acknowledge_warning, application_service.py:CheckpointApplicationService.classify_warning
CheckpointApplicationService.get_sudden_thresholds L597	Callable for get sudden thresholds in application_service.py.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	to_dict	api_read_routes.py:sudden_configuration, demo_control.py:DemoController.status
CheckpointApplicationService.update_sudden_thresholds L600	Callable for update sudden thresholds in application_service.py.	(self, *, spike_threshold: float | None, recovery_threshold: float | None, timestamp: datetime, operator: str, reason:...) → dict[str, object]	O(1) excluding delegated I/O/model work	RuntimeValidationError, SuddenIncidentDetector, _require_utc, record_active_configuration, save, str, to_dict, with_updates	api_admin_routes.py:update_sudden_configuration
CheckpointApplicationService.get_trend_configuration L629	Callable for get trend configuration in application_service.py.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	to_dict	api_read_routes.py:trend_configuration, demo_control.py:DemoController.status
CheckpointApplicationService.update_trend_configuration L632	Callable for update trend configuration in application_service.py.	(self, *, gradient_threshold: float | None, recovery_threshold: float | None, trend_history_hours: float | None, bucke...) → dict[str, object]	O(1) excluding delegated I/O/model work	RuntimeValidationError, SlowTrendWarningDetector, _require_utc, record_active_configuration, save, str, to_dict, with_updates	api_admin_routes.py:update_trend_configuration
CheckpointApplicationService.model_information L667	Callable for model information in application_service.py.	(self) → dict[str, Any]	O(N) typical	active_version, to_dict	api_read_routes.py:model_information
CheckpointApplicationService.upload_healthy_dataset L701	Callable for upload healthy dataset in application_service.py.	(self, *, filename: str, csv_text: str, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	RuntimeValidationError, str, upload_dataset	api_admin_routes.py:upload_healthy_dataset, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.validate_healthy_dataset L714	Callable for validate healthy dataset in application_service.py.	(self, dataset_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, RuntimeValidationError, str, validate_dataset	api_admin_routes.py:validate_healthy_dataset, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.start_model_training L722	Callable for start model training in application_service.py.	(self, dataset_id: str, *, healthy_data_confirmed: bool, operator: str, reason: str, hyperparameters: dict[str, Any], ...) → dict[str, Any]	O(1) excluding delegated I/O/model work	LifecycleConflictError, start_training, str	api_admin_routes.py:start_model_training, demo_control.py:DemoController._run_model_lifecycle_demo

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
CheckpointApplicationService.model_training_status L744	Callable for model training status in application_service.py.	(self, job_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, str, training_job	api_admin_routes.py:get_model_training_progress, api_admin_routes.py:get_model_training_status, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.list_model_versions L750	Callable for list model versions in application_service.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	list_versions	api_admin_routes.py:list_model_versions, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.get_model_candidate L753	Callable for get model candidate in application_service.py.	(self, candidate_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, candidate, str	api_admin_routes.py:get_model_candidate
CheckpointApplicationService.compare_model_candidate L759	Callable for compare model candidate in application_service.py.	(self, candidate_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ResourceNotFoundError, compare_candidate, str	api_admin_routes.py:compare_model_candidate, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.review_model_candidate L765	Callable for review model candidate in application_service.py.	(self, candidate_id: str, *, approved: bool, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	LifecycleConflictError, action, str	api_admin_routes.py:approve_model_candidate, api_admin_routes.py:reject_model_candidate, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.deploy_model_candidate L783	Callable for deploy model candidate in application_service.py.	(self, candidate_id: str, *, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	_activate_model_version	api_admin_routes.py:deploy_model_candidate, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService.rollback_model_version L794	Callable for rollback model version in application_service.py.	(self, version_id: str, *, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	_activate_model_version	api_admin_routes.py:rollback_model_version, demo_control.py:DemoController._run_model_lifecycle_demo
CheckpointApplicationService._activate_model_version L805	Internal helper for activate model version in application_service.py.	(self, version_id: str, *, operator: str, reason: str, action: str, require_approved: bool) → dict[str, Any]	O(1) excluding delegated I/O/model work	LifecycleConflictError, RuntimeValidationError, SlowTrendWarningDetector, SuddenIncidentDetector, SuddenRuntimeScorer, commit_activation, deployment_bundle, save…	application_service.py:CheckpointApplicationService.deploy_model_candidate, application_service.py:CheckpointApplicationService.rollback_model_version
CheckpointApplicationService.shutdown L871	Callable for shutdown in application_service.py.	(self) → None	O(1) excluding delegated I/O/model work	shutdown	api.py:create_app, application_service.py:CheckpointApplicationService.shutdown, demo_control.py:DemoController.shutdown, model_lifecycle.py:ModelLifecycleManager.shutdown, windows_app.py:main
CheckpointApplicationService.reset_monitoring_runtime L874	Reset buffer and detector stores without changing configuration/artifacts.	(self) → None	O(1) excluding delegated I/O/model work	SlowTrendWarningDetector, SuddenIncidentDetector, reset	demo_control.py:DemoController.clear_records, demo_control.py:DemoController.reset
CheckpointApplicationService.process_trend_evaluation L893	Application hook for scheduled evaluators and deterministic tests.	(self, evaluation: TrendEvaluation) → dict[str, object] | None	O(1) excluding delegated I/O/model work	process_evaluation, to_dict	entry point / dynamic / unreferenced statically
CheckpointApplicationService._require_utc L903	Internal helper for require utc in application_service.py.	(value: datetime, name: str) → None	O(1) excluding delegated I/O/model work	RuntimeValidationError, timedelta, utcoffset	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.acknowledge_warning, application_service.py:CheckpointApplicationService.classify_incident, application_service.py:CheckpointApplicationService.classify_warning, application_service.py:CheckpointApplicationService.reconstruction_history…

atomic_io.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
replace_with_retry **L9**	Replace atomically, retrying transient Windows reader-sharing failures.	(temporary: Path, destination: Path) → None	O(N) typical	range, replace, sleep	demo_control.py:_atomic_json, incidents.py:JsonIncidentStore.save, model_lifecycle.py:_atomic_json, runtime_store.py:JsonRuntimeBufferStore.save, slow_trend_evaluation.py:_json_dump…

config.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_require_number **L17**	Internal helper for require number in config.py.	(mapping: Mapping[str, Any], key: str) → float	O(1) excluding delegated I/O/model work	ConfigurationError, float, get, isinstance	config.py:ModelConfig.from_dict, config.py:PreprocessingConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:SensorQualityConfig.from_dict, synthetic_evaluation.py:ScenarioConfig._validate
SensorQualityConfig.from_dict **L33**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SensorQualityConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, _require_number, cls	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
SensorGenerationConfig.from_dict **L70**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SensorGenerationConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, _require_number, cls, float, from_dict, get, isinstance	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
PreprocessingConfig.from_dict **L119**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'PreprocessingConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, _require_number, cls, int, len, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
ModelConfig.from_dict **L190**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'ModelConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, _require_number, cls, int, isinstance, lower, min, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
DemoModelDataConfig.from_dict **L250**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'DemoModelDataConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, all, cls, int, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
PumpScheduleConfig.from_dict **L291**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'PumpScheduleConfig'	O(1) excluding delegated I/O/model work	ConfigurationError, cls, get, int, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
PumpScheduleConfig.running_at **L309**	Callable for running at in config.py.	(self, sequence: int) → bool	O(1) excluding delegated I/O/model work	—	synthetic.py:_checkpoint_records
CheckpointGenerationConfig.from_dict **L331**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'CheckpointGenerationConfig'	O(N) typical	ConfigurationError, cls, from_dict, get, int, isinstance, items, keys…	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
ProjectConfig.from_dict **L384**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'ProjectConfig'	O(N·M) / data-dependent loops	ConfigurationError, cls, from_dict, int, len, set, str, tuple	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
ProjectConfig.checkpoint **L425**	Callable for checkpoint in config.py.	(self, checkpoint_id: str) → CheckpointGenerationConfig	O(N) typical	KeyError	application_service.py:CheckpointApplicationService.checkpoint_status, model_lifecycle.py:ModelLifecycleManager.validate_dataset, preprocessing.py:SensorPreprocessor.__init__, synthetic.py:generate_healthy_records
load_config **L432**	Callable for load config in config.py.	(path: str | Path) → ProjectConfig	O(1) excluding delegated I/O/model work	ConfigurationError, Path, from_dict, isinstance, load, open	application_service.py:CheckpointApplicationService.__init__, synthetic.py:main

demo_control.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_atomic_json **L77**	Internal helper for atomic json in demo_control.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, with_name, write	demo_control.py:DemoController._load_or_initialize, demo_control.py:DemoController._save, model_lifecycle.py:ModelLifecycleManager._load_or_initialize, model_lifecycle.py:ModelLifecycleManager._save, model_lifecycle.py:ModelLifecycleManager.commit_activation
_record_from_request **L86**	Internal helper for record from request in demo_control.py.	(request: SensorReadingRequest) → RawSensorRecord	O(1) excluding delegated I/O/model work	RawSensorRecord	demo_control.py:DemoController._process_point
DemoController.__init__ **L102**	Initializes the object and validates/loads required collaborators.	(self, service: CheckpointApplicationService, *, state_path: str | Path, simulation_start: datetime | None=None) → None	O(N·M) / data-dependent loops	Event, Path, RLock, _load_or_initialize, datetime, deque, format_utc, len…	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
DemoController._new_state **L145**	Internal helper for new state in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	—	demo_control.py:DemoController._load_or_initialize, demo_control.py:DemoController.reset
DemoController._load_or_initialize **L184**	Internal helper for load or initialize in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	DemoControlError, ValueError, _append_event_to, _atomic_json, _new_state, exists, get, load…	demo_control.py:DemoController.__init__, model_lifecycle.py:ModelLifecycleManager.__init__
DemoController._append_event_to **L215**	Internal helper for append event to in demo_control.py.	(state: dict[str, Any], level: str, category: str, message: str, **details: Any) → None	O(1) excluding delegated I/O/model work	append, get, int, setdefault	demo_control.py:DemoController._capture_scores_and_notifications, demo_control.py:DemoController._event, demo_control.py:DemoController._load_or_initialize, demo_control.py:DemoController._process_point, demo_control.py:DemoController._record_rejection…
DemoController._event **L231**	Internal helper for event in demo_control.py.	(self, level: str, category: str, message: str, **details: Any) → None	O(1) excluding delegated I/O/model work	_append_event_to	demo_control.py:DemoController._capture_quality, demo_control.py:DemoController._process_point, demo_control.py:DemoController._run_model_lifecycle_demo, demo_control.py:DemoController.classify_bottle_jam
DemoController._save **L235**	Persists current state through the module's storage boundary.	(self) → None	O(1) excluding delegated I/O/model work	_atomic_json	demo_control.py:DemoController._process_point, demo_control.py:DemoController._run, demo_control.py:DemoController.cancel_preset, demo_control.py:DemoController.classify_bottle_jam, demo_control.py:DemoController.clear_records…
DemoController._ensure_source **L239**	Internal helper for ensure source in demo_control.py.	(self, needed: int) → None	O(1) excluding delegated I/O/model work	generate_healthy_records, int, len, max, replace, tuple	demo_control.py:DemoController._take_healthy
DemoController._take_healthy **L256**	Internal helper for take healthy in demo_control.py.	(self, count: int) → tuple[RawSensorRecord, …]	O(1) excluding delegated I/O/model work	_ensure_source, int	demo_control.py:DemoController._healthy_points, demo_control.py:DemoController._payload_point, demo_control.py:DemoController._scenario_points, demo_control.py:DemoController.trigger_scenario

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
DemoController._healthy_points **L263**	Internal helper for healthy points in demo_control.py.	(self, count: int) → list[DemoPoint]	O(N) typical	DemoPoint, _take_healthy	demo_control.py:DemoController._run, demo_control.py:DemoController.run_preset, demo_control.py:DemoController.step, demo_control.py:DemoController.trigger_scenario
DemoController._scenario_points **L269**	Internal helper for scenario points in demo_control.py.	(self, scenario: ScenarioConfig, *, minimum_length: int | None=None, pump_stopped_during_active: bool=False) → list[DemoPoint]	O(N) typical	DemoPoint, _take_healthy, append, get, inject_scenario, int, max, replace…	demo_control.py:DemoController.run_preset, demo_control.py:DemoController.trigger_scenario
DemoController._payload_point **L315**	Internal helper for payload point in demo_control.py.	(self, scenario_id: str) → DemoPoint	O(1) excluding delegated I/O/model work	DemoControlError, DemoPoint, _take_healthy, as_dict, format_utc, pop, timedelta	demo_control.py:DemoController._process_point, demo_control.py:DemoController.trigger_scenario
DemoController.start **L338**	Callable for start in demo_control.py.	(self, *, seed: int | None=None, speed: str | None=None, pump_running: bool | None=None) → dict[str, Any]	O(1) excluding delegated I/O/model work	DemoControlError, _append_event_to, _apply_settings, _save, _spawn_worker, clear, format_utc, is_alive…	api_demo_routes.py:start_demo_stream, demo_control.py:DemoController._spawn_worker, windows_app.py:main, windows_app.py:run_server_smoke_test
DemoController._spawn_worker **L370**	Internal helper for spawn worker in demo_control.py.	(self) → None	O(1) excluding delegated I/O/model work	Thread, start	demo_control.py:DemoController.run_preset, demo_control.py:DemoController.start, demo_control.py:DemoController.trigger_scenario
DemoController.pause **L378**	Callable for pause in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	DemoControlError, _append_event_to, _save, clear, status	api_demo_routes.py:pause_demo_stream
DemoController.resume **L390**	Callable for resume in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	DemoControlError, _append_event_to, _save, set, status	api_demo_routes.py:resume_demo_stream
DemoController.stop **L402**	Callable for stop in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	_append_event_to, _save, current_thread, is_alive, join, set, status	api_demo_routes.py:stop_demo_stream
DemoController.reset **L421**	Returns isolated mutable state to its initial value.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	_append_event_to, _new_state, _save, clear, get, is_alive, join, reset_monitoring_runtime…	api_demo_routes.py:reset_demo, application_service.py:CheckpointApplicationService.reset_monitoring_runtime, demo_control.py:DemoController.run_preset, slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation…
DemoController.clear_records **L451**	Callable for clear records in demo_control.py.	(self) → dict[str, Any]	O(N) typical	_append_event_to, _save, clear, reset_monitoring_runtime, status	api_demo_routes.py:clear_demo_records
DemoController._apply_settings **L474**	Internal helper for apply settings in demo_control.py.	(self, *, seed: int | None, speed: str | None, pump_running: bool | None) → None	O(1) excluding delegated I/O/model work	DemoControlError, bool, int	demo_control.py:DemoController.run_preset, demo_control.py:DemoController.settings, demo_control.py:DemoController.start
DemoController.settings **L488**	Callable for settings in demo_control.py.	(self, *, seed: int | None=None, speed: str | None=None, pump_running: bool | None=None) → dict[str, Any]	O(1) excluding delegated I/O/model work	_append_event_to, _apply_settings, _save, status	api_demo_routes.py:update_demo_settings
DemoController.trigger_scenario **L506**	Callable for trigger scenario in demo_control.py.	(self, scenario_id: str, *, feature: str | None=None, duration_seconds: int | None=None) → dict[str, Any]	O(N) typical	DemoControlError, DemoPoint, _append_event_to, _healthy_points, _payload_point, _save, _scenario_points, _spawn_worker…	api_demo_routes.py:trigger_demo_scenario
DemoController.run_preset **L595**	Callable for run preset in demo_control.py.	(self, preset_id: str, *, speed: str | None=None, pump_running: bool | None=None) → dict[str, Any]	O(N) typical	DemoControlError, DemoPoint, _append_event_to, _apply_settings, _healthy_points, _save, _scenario_points, _spawn_worker…	api_demo_routes.py:run_demo_preset
DemoController.cancel_preset **L698**	Callable for cancel preset in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	DemoControlError, _append_event_to, _save, clear, set, status, update	api_demo_routes.py:cancel_demo_preset

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
DemoController.step **L718**	Callable for step in demo_control.py.	(self, count: int=1) → dict[str, Any]	O(N) typical	DemoControlError, _healthy_points, _process_point, _save, format_utc, is_alive, popleft, range…	api_demo_routes.py:step_demo_stream, training.py:train_autoencoder
DemoController.classify_bottle_jam **L734**	Callable for classify bottle jam in demo_control.py.	(self, incident_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	_event, _save, classify_incident, get, parse_utc_datetime	api_demo_routes.py:classify_demo_bottle_jam, demo_control.py:DemoController._process_point
DemoController.controlled_warning_lifecycle **L761**	Callable for controlled warning lifecycle in demo_control.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	_append_event_to, _controlled_lifecycle_demo, _save	api_demo_routes.py:run_controlled_warning_lifecycle, demo_control.py:DemoController._process_point
DemoController._run **L778**	Internal helper for run in demo_control.py.	(self) → None	O(N) typical	_append_event_to, _healthy_points, _process_point, _save, clear, flush, get, is_set…	entry point / dynamic / unreferenced statically
DemoController._process_point **L839**	Internal helper for process point in demo_control.py.	(self, point: DemoPoint) → None	O(N) typical	_append_event_to, _capture_quality, _capture_scores_and_notifications, _event, _increment_preset_progress, _payload_point, _record_from_request, _record_rejection…	demo_control.py:DemoController._run, demo_control.py:DemoController.step
DemoController._record_rejection **L979**	Internal helper for record rejection in demo_control.py.	(self, point: DemoPoint, reason: str, *, reasons: list[str]) → None	O(1) excluding delegated I/O/model work	Counter, _append_event_to, dict, format_utc	demo_control.py:DemoController._process_point
DemoController._capture_quality **L1006**	Internal helper for capture quality in demo_control.py.	(self) → None	O(N·M) / data-dependent loops	Counter, SensorPreprocessor, _event, add, dict, format_utc, items, model_features…	demo_control.py:DemoController._process_point
DemoController._capture_scores_and_notifications **L1054**	Internal helper for capture scores and notifications in demo_control.py.	(self, response: Mapping[str, Any]) → None	O(N·M) / data-dependent loops	Counter, _append_event_to, add, append, dict, format_utc, get, len…	demo_control.py:DemoController._process_point
DemoController._increment_preset_progress **L1103**	Internal helper for increment preset progress in demo_control.py.	(self) → None	O(1) excluding delegated I/O/model work	—	demo_control.py:DemoController._process_point
DemoController._run_model_lifecycle_demo **L1108**	Internal helper for run model lifecycle demo in demo_control.py.	(self) → None	O(N) typical	DemoControlError, _event, compare_model_candidate, deploy_model_candidate, get, list_model_versions, model_training_status, next…	demo_control.py:DemoController._process_point
DemoController.status **L1191**	Callable for status in demo_control.py.	(self) → dict[str, Any]	O(N log N) typical	checkpoint_status, dumps, get_sudden_thresholds, get_trend_configuration, len, list, list_incidents, list_warnings…	api_demo_routes.py:demo_status, demo_control.py:DemoController.cancel_preset, demo_control.py:DemoController.clear_records, demo_control.py:DemoController.pause, demo_control.py:DemoController.reset…
DemoController.shutdown **L1240**	Callable for shutdown in demo_control.py.	(self) → None	O(1) excluding delegated I/O/model work	is_alive, join, set, shutdown	api.py:create_app, application_service.py:CheckpointApplicationService.shutdown, demo_control.py:DemoController.shutdown, model_lifecycle.py:ModelLifecycleManager.shutdown, windows_app.py:main

incidents.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_validate_feature_mapping **L47**	Internal helper for validate feature mapping in incidents.py.	(values: Mapping[str, float | None], *, allow_null: bool) → dict[str, float | None]	O(N) typical	ValueError, float, isfinite, keys, tuple	incidents.py:RuntimeScore.__post_init__

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_contribution_shares **L67**	Internal helper for contribution shares in incidents.py.	(errors: Mapping[str, float]) → dict[str, float]	O(N) typical	sum, values	incidents.py:SuddenIncidentDetector._open, incidents.py:SuddenIncidentDetector._update
RuntimeScore.__post_init__ **L86**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _validate_feature_mapping, any, isfinite, timedelta, utcoffset, values	entry point / dynamic / unreferenced statically
RuntimeScore.to_dict **L103**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	format_utc	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
RuntimeScore.from_dict **L117**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'RuntimeScore'	O(N) typical	bool, cls, float, get, parse_utc_datetime, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
NotificationEvent.to_dict **L161**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, str]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
NotificationEvent.from_dict **L172**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'NotificationEvent'	O(1) excluding delegated I/O/model work	NotificationEventType, cls, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
SuddenIncident.to_dict **L220**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
SuddenIncident.from_dict **L267**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SuddenIncident'	O(N·M) / data-dependent loops	IncidentStatus, OperatorClassification, cls, float, get, int, list, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
IncidentStoreState.to_dict **L339**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N log N) typical	sorted, to_dict	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
IncidentStoreState.from_dict **L355**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'IncidentStoreState'	O(N·M) / data-dependent loops	ValueError, cls, from_dict, get, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
JsonIncidentStore.__init__ **L377**	Initializes the object and validates/loads required collaborators.	(self, path: str | Path, *, checkpoint_id: str) → None	O(1) excluding delegated I/O/model work	IncidentStoreState, Path, ValueError, _load, exists	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
JsonIncidentStore._load **L386**	Loads and validates persisted local state.	(self) → IncidentStoreState	O(1) excluding delegated I/O/model work	ValueError, from_dict, load, open	incidents.py:JsonIncidentStore.__init__, runtime_store.py:JsonRuntimeBufferStore.__init__, trend_warnings.py:JsonTrendWarningStore.__init__
JsonIncidentStore.save **L394**	Persists the artifact or state document.	(self, *, force: bool=False) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, to_dict, with_name, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration…
JsonIncidentStore.set_deferred_save_interval **L409**	Batch persistence writes without changing in-memory detector updates.	(self, score_writes: int=1) → None	O(1) excluding delegated I/O/model work	ValueError, isinstance, save	demo_control.py:DemoController._run

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
JsonIncidentStore.flush **L423**	Callable for flush in incidents.py.	(self) → None	O(1) excluding delegated I/O/model work	save	demo_control.py:DemoController._run
JsonIncidentStore.reset **L426**	Returns isolated mutable state to its initial value.	(self) → None	O(1) excluding delegated I/O/model work	IncidentStoreState, save	api_demo_routes.py:reset_demo, application_service.py:CheckpointApplicationService.reset_monitoring_runtime, demo_control.py:DemoController.run_preset, slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation...
JsonIncidentStore.active_incident **L431**	Callable for active incident in incidents.py.	(self) → SuddenIncident | None	O(1) excluding delegated I/O/model work	—	api_read_routes.py:active_incident, demo_control.py:DemoController._process_point
SuddenIncidentDetector.__init__ **L440**	Initializes the object and validates/loads required collaborators.	(self, thresholds: SuddenThresholdConfig, store: JsonIncidentStore) → None	O(1) excluding delegated I/O/model work	ValueError	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
SuddenIncidentDetector.active_incident **L451**	Callable for active incident in incidents.py.	(self) → SuddenIncident | None	O(1) excluding delegated I/O/model work	—	api_read_routes.py:active_incident, demo_control.py:DemoController._process_point
SuddenIncidentDetector.process_score **L454**	Callable for process score in incidents.py.	(self, score: RuntimeScore) → tuple[NotificationEvent, ...]	O(1) excluding delegated I/O/model work	ValueError, _open, _update, append, save, tuple	application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, incidents.py:SuddenIncidentDetector.replay, sudden_runtime.py:_evaluate_replay, sudden_runtime.py:run_controlled_lifecycle_demo
SuddenIncidentDetector.replay **L479**	Processes chronological evidence again to verify deterministic state.	(self, scores: Iterable[RuntimeScore]) → IncidentStoreState	O(N) typical	process_score	slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:_replay_to_fresh_store
SuddenIncidentDetector.acknowledge_active **L484**	Callable for acknowledge active in incidents.py.	(self, *, timestamp: datetime, operator: str) → SuddenIncident	O(1) excluding delegated I/O/model work	ValueError, _require_active, append, format_utc, save	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.acknowledge_warning
SuddenIncidentDetector.classify_active **L507**	Callable for classify active in incidents.py.	(self, classification: OperatorClassification, *, note: str, timestamp: datetime, operator: str) → SuddenIncident	O(1) excluding delegated I/O/model work	ValueError, _require_active, append, format_utc, save	application_service.py:CheckpointApplicationService.classify_incident, sudden_runtime.py:run_controlled_lifecycle_demo
SuddenIncidentDetector._require_active **L555**	Internal helper for require active in incidents.py.	(self) → SuddenIncident	O(1) excluding delegated I/O/model work	ValueError	incidents.py:SuddenIncidentDetector.acknowledge_active, incidents.py:SuddenIncidentDetector.classify_active, trend_warnings.py:SlowTrendWarningDetector.acknowledge_active, trend_warnings.py:SlowTrendWarningDetector.mark_active_under_review
SuddenIncidentDetector._open **L561**	Internal helper for open in incidents.py.	(self, score: RuntimeScore) → tuple[SuddenIncident, NotificationEvent]	O(N) typical	NotificationEvent, SuddenIncident, _contribution_shares, append, astimezone, dict, float, format_utc...	incidents.py:SuddenIncidentDetector.process_score, trend_warnings.py:SlowTrendWarningDetector.process_evaluation
SuddenIncidentDetector._update **L615**	Internal helper for update in incidents.py.	(self, incident: SuddenIncident, score: RuntimeScore) → NotificationEvent | None	O(N) typical	_close_after_recovery, _contribution_shares, append, dict, float, format_utc, max	incidents.py:SuddenIncidentDetector.process_score, trend_warnings.py:SlowTrendWarningDetector.process_evaluation
SuddenIncidentDetector._close_after_recovery **L654**	Internal helper for close after recovery in incidents.py.	(self, incident: SuddenIncident, timestamp: str) → NotificationEvent	O(N) typical	NotificationEvent, any, append	incidents.py:SuddenIncidentDetector._update

model.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
LSTMAutoencoder.__init__ **L18**	Initializes the object and validates/loads required collaborators.	(self, *, window_size: int, input_size: int, encoder_hidden_size: int, bottleneck_size: int, decoder_hidden_size: int,...) → None	O(1) excluding delegated I/O/model work	Dropout, LSTM, Linear, ReLU, Sequential, __init__, super	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
LSTMAutoencoder.from_config **L56**	Callable for from config in model.py.	(cls, config: ModelConfig, *, window_size: int) → 'LSTMAutoencoder'	O(1) excluding delegated I/O/model work	cls, len	training.py:train_autoencoder
LSTMAutoencoder.from_architecture **L69**	Callable for from architecture in model.py.	(cls, architecture: Mapping[str, object]) → 'LSTMAutoencoder'	O(1) excluding delegated I/O/model work	cls, float, int	model_package.py:load_model_package
LSTMAutoencoder.architecture **L83**	Callable for architecture in model.py.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	model_package.py:save_model_package
LSTMAutoencoder.forward **L103**	Callable for forward in model.py.	(self, inputs: Tensor) → Tensor	O(B·T·H·(F+H)) recurrent inference	ValueError, bottleneck, decoder, decoder_output_dropout, encoder, expand, output_layer, tuple...	entry point / dynamic / unreferenced statically
reconstruction_errors **L125**	Return exact per-window overall and per-feature robust-scaled MSE.	(inputs: Tensor, reconstructions: Tensor) → ReconstructionErrors	O(1) excluding delegated I/O/model work	ReconstructionErrors, ValueError, len, mean, pow	model_package.py:save_model_package, model_package.py:verify_model_package, sudden_runtime.py:SuddenRuntimeScorer._reconstruction_scores, synthetic_evaluation.py:_score_windows, training.py:evaluate_reconstruction

model_lifecycle.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_now **L53**	Internal helper for now in model_lifecycle.py.	() → str	O(1) excluding delegated I/O/model work	format_utc, now	model_lifecycle.py:ModelLifecycleManager._audit, model_lifecycle.py:ModelLifecycleManager._load_or_initialize, model_lifecycle.py:ModelLifecycleManager._review, model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager.commit_activation...
_atomic_json **L57**	Internal helper for atomic json in model_lifecycle.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, with_name, write	demo_control.py:DemoController._load_or_initialize, demo_control.py:DemoController._save, model_lifecycle.py:ModelLifecycleManager._load_or_initialize, model_lifecycle.py:ModelLifecycleManager._save, model_lifecycle.py:ModelLifecycleManager.commit_activation
_identifier **L66**	Internal helper for identifier in model_lifecycle.py.	(prefix: str) → str	O(1) excluding delegated I/O/model work	now, strftime, uuid4	model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager._training_config, model_lifecycle.py:ModelLifecycleManager.start_training, model_lifecycle.py:ModelLifecycleManager.upload_dataset
_parse_timestamp **L71**	Internal helper for parse timestamp in model_lifecycle.py.	(value: str) → datetime	O(1) excluding delegated I/O/model work	ValueError, astimezone, endswith, fromisoformat, strip, utcoffset	model_lifecycle.py:parse_healthy_csv
_parse_sensor **L83**	Internal helper for parse sensor in model_lifecycle.py.	(value: str, feature: str) → float | None	O(1) excluding delegated I/O/model work	ValueError, float, isfinite, strip	model_lifecycle.py:parse_healthy_csv
_parse_bool **L92**	Internal helper for parse bool in model_lifecycle.py.	(value: str) → bool	O(1) excluding delegated I/O/model work	ValueError, lower, strip	model_lifecycle.py:parse_healthy_csv
parse_healthy_csv **L101**	Parse the upload contract; extra columns stay outside the model tensor.	(csv_text: str) → tuple[tuple[RawSensorRecord, ...], tuple[str, ...]]	O(N) typical	DictReader, ModelLifecycleError, RawSensorRecord, StringIO, _parse_bool, _parse_sensor, _parse_timestamp, append...	model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager.validate_dataset

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
ModelLifecycleManager.__init__ **L144**	Initializes the object and validates/loads required collaborators.	(self, project: ProjectConfig, *, runtime_root: Path, baseline_package: Path, runtime_sudden_config: Path, runtime_tre…) → None	O(1) excluding delegated I/O/model work	RLock, ThreadPoolExecutor, _load_or_initialize, mkdir, resolve	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
ModelLifecycleManager._load_or_initialize **L167**	Internal helper for load or initialize in model_lifecycle.py.	(self) → dict[str, Any]	O(N) typical	ModelLifecycleError, _atomic_json, _now, exists, get, load, load_model_package, mkdir…	demo_control.py:DemoController.__init__, model_lifecycle.py:ModelLifecycleManager.__init__
ModelLifecycleManager.shutdown **L233**	Callable for shutdown in model_lifecycle.py.	(self, *, wait: bool=True) → None	O(1) excluding delegated I/O/model work	shutdown	api.py:create_app, application_service.py:CheckpointApplicationService.shutdown, demo_control.py:DemoController.shutdown, model_lifecycle.py:ModelLifecycleManager.shutdown, windows_app.py:main
ModelLifecycleManager._save **L236**	Persists current state through the module's storage boundary.	(self) → None	O(1) excluding delegated I/O/model work	_atomic_json	demo_control.py:DemoController._process_point, demo_control.py:DemoController._run, demo_control.py:DemoController.cancel_preset, demo_control.py:DemoController.classify_bottle_jam, demo_control.py:DemoController.clear_records…
ModelLifecycleManager._audit **L239**	Internal helper for audit in model_lifecycle.py.	(self, action: str, *, operator: str, reason: str, **references: object) → None	O(1) excluding delegated I/O/model work	_now, append	model_lifecycle.py:ModelLifecycleManager._review, model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager.start_training, model_lifecycle.py:ModelLifecycleManager.upload_dataset, model_lifecycle.py:ModelLifecycleManager.validate_dataset
ModelLifecycleManager.upload_dataset **L257**	Callable for upload dataset in model_lifecycle.py.	(self, *, filename: str, csv_text: str, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, Path, _audit, _identifier, _now, _safe_dataset, _save, encode…	application_service.py:CheckpointApplicationService.upload_healthy_dataset
ModelLifecycleManager.validate_dataset **L300**	Callable for validate dataset in model_lifecycle.py.	(self, dataset_id: str) → dict[str, Any]	O(N·M) / data-dependent loops	ModelLifecycleError, Path, _audit, _now, _safe_dataset, _save, any, append…	application_service.py:CheckpointApplicationService.validate_healthy_dataset
ModelLifecycleManager.start_training **L412**	Callable for start training in model_lifecycle.py.	(self, dataset_id: str, *, healthy_data_confirmed: bool, operator: str, reason: str, hyperparameters: Mapping[str, Any…) → dict[str, Any]	O(N) typical	ModelLifecycleError, _audit, _identifier, _now, _safe_job, _save, _training_config, any…	application_service.py:CheckpointApplicationService.start_model_training
ModelLifecycleManager._training_config **L478**	Internal helper for training config in model_lifecycle.py.	(self, overrides: Mapping[str, Any]) → ModelConfig	O(N log N) typical	ModelLifecycleError, _identifier, asdict, from_dict, join, replace, set, sorted	model_lifecycle.py:ModelLifecycleManager.start_training
ModelLifecycleManager._run_training **L504**	Internal helper for run training in model_lifecycle.py.	(self, job_id: str, model_config: ModelConfig) → None	O(N) typical	ModelLifecycleError, Path, _active_version_internal, _audit, _identifier, _now, _save, derive_provisional_thresholds…	entry point / dynamic / unreferenced statically
ModelLifecycleManager.training_job **L708**	Callable for training job in model_lifecycle.py.	(self, job_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, _safe_job, get	application_service.py:CheckpointApplicationService.model_training_status, model_lifecycle.py:ModelLifecycleManager.wait_for_job
ModelLifecycleManager.wait_for_job **L715**	Callable for wait for job in model_lifecycle.py.	(self, job_id: str, timeout: float=120.0) → dict[str, Any]	O(1) excluding delegated I/O/model work	get, result, training_job	entry point / dynamic / unreferenced statically
ModelLifecycleManager.list_versions **L721**	Callable for list versions in model_lifecycle.py.	(self) → dict[str, Any]	O(N) typical	_safe_version, get, len, list, sort, str, values	application_service.py:CheckpointApplicationService.list_model_versions
ModelLifecycleManager.candidate **L733**	Callable for candidate in model_lifecycle.py.	(self, candidate_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, _safe_candidate, get	application_service.py:CheckpointApplicationService.get_model_candidate
ModelLifecycleManager.compare_candidate **L740**	Callable for compare candidate in model_lifecycle.py.	(self, candidate_id: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, _active_version_internal, _safe_candidate, get, load, load_model_package, to_dict	application_service.py:CheckpointApplicationService.compare_model_candidate

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
ModelLifecycleManager.approve L783	Callable for approve in model_lifecycle.py.	(self, candidate_id: str, *, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	_review	entry point / dynamic / unreferenced statically
ModelLifecycleManager.reject L786	Callable for reject in model_lifecycle.py.	(self, candidate_id: str, *, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	_review	entry point / dynamic / unreferenced statically
ModelLifecycleManager._review L789	Internal helper for review in model_lifecycle.py.	(self, candidate_id: str, status: str, *, operator: str, reason: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, _audit, _now, _safe_candidate, _save, get, strip, update	model_lifecycle.py:ModelLifecycleManager.approve, model_lifecycle.py:ModelLifecycleManager.reject
ModelLifecycleManager.deployment_bundle L820	Callable for deployment bundle in model_lifecycle.py.	(self, version_id: str, *, require_approved: bool) → tuple[Path, SuddenThresholdConfig, SlowTrendConfig]	O(N) typical	ModelLifecycleError, Path, any, get, load, load_model_package, str, verify_model_package	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version
ModelLifecycleManager.commit_activation L855	Callable for commit activation in model_lifecycle.py.	(self, version_id: str, *, operator: str, reason: str, action: str) → dict[str, Any]	O(1) excluding delegated I/O/model work	ModelLifecycleError, _atomic_json, _now, append, deepcopy, get, update	application_service.py:CheckpointApplicationService._activate_model_version
ModelLifecycleManager.active_version L912	Callable for active version in model_lifecycle.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	_active_version_internal, _safe_version	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService.model_information
ModelLifecycleManager._active_version_internal L916	Internal helper for active version internal in model_lifecycle.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	—	model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager.active_version, model_lifecycle.py:ModelLifecycleManager.compare_candidate, model_lifecycle.py:ModelLifecycleManager.record_active_configuration
ModelLifecycleManager.record_active_configuration L919	Refresh rollback snapshots after an audited runtime configuration edit.	(self) → None	O(1) excluding delegated I/O/model work	_active_version_internal, _save, load, save	application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration
ModelLifecycleManager._safe_dataset L930	Internal helper for safe dataset in model_lifecycle.py.	(record: Mapping[str, Any]) → dict[str, Any]	O(N) typical	items	model_lifecycle.py:ModelLifecycleManager.upload_dataset, model_lifecycle.py:ModelLifecycleManager.validate_dataset
ModelLifecycleManager._safe_job L934	Internal helper for safe job in model_lifecycle.py.	(record: Mapping[str, Any]) → dict[str, Any]	O(1) excluding delegated I/O/model work	dict	model_lifecycle.py:ModelLifecycleManager.start_training, model_lifecycle.py:ModelLifecycleManager.training_job
ModelLifecycleManager._safe_candidate L938	Internal helper for safe candidate in model_lifecycle.py.	(record: Mapping[str, Any]) → dict[str, Any]	O(N) typical	items	model_lifecycle.py:ModelLifecycleManager._review, model_lifecycle.py:ModelLifecycleManager.candidate, model_lifecycle.py:ModelLifecycleManager.compare_candidate
ModelLifecycleManager._safe_version L943	Internal helper for safe version in model_lifecycle.py.	(record: Mapping[str, Any]) → dict[str, Any]	O(N) typical	items	model_lifecycle.py:ModelLifecycleManager.active_version, model_lifecycle.py:ModelLifecycleManager.list_versions

model_package.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_json_dump L40	Internal helper for json dump in model_package.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, write	model_package.py:save_model_package, model_package.py:write_evaluation_report, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios, synthetic_evaluation.py:_write_held_out_metadata...
save_model_package L47	Callable for save model package in model_package.py.	(package_path: str | Path, *, project: ProjectConfig, model_config: ModelConfig, data: PreparedModelData, training: Tr...) → dict[str, Any]	O(N) typical	Path, _json_dump, architecture, asdict, enumerate, eval, float, format_utc...	model_lifecycle.py:ModelLifecycleManager._run_training

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
load_model_package **L204**	Callable for load model package in model_package.py.	(package_path: str | Path, *, device: str='cpu') → LoadedModelPackage	O(1) excluding delegated I/O/model work	LoadedModelPackage, Path, ValueError, eval, from_architecture, load, load_state_dict, open…	model_lifecycle.py:ModelLifecycleManager._load_or_initialize, model_lifecycle.py:ModelLifecycleManager.compare_candidate, model_lifecycle.py:ModelLifecycleManager.deployment_bundle, model_package.py:verify_model_package, slow_trend_evaluation.py:run_slow_trend_evaluation…
verify_model_package **L248**	Callable for verify model package in model_package.py.	(package_path: str | Path) → dict[str, object]	O(N) typical	abs, all, allclose, enumerate, float, inference_mode, load_model_package, math_isclose…	model_lifecycle.py:ModelLifecycleManager._run_training, model_lifecycle.py:ModelLifecycleManager.deployment_bundle
math_isclose **L294**	Callable for math isclose in model_package.py.	(left: float, right: float, *, atol: float, rtol: float) → bool	O(1) excluding delegated I/O/model work	abs	model_package.py:verify_model_package
write_evaluation_report **L298**	Callable for write evaluation report in model_package.py.	(path: str | Path, *, project: ProjectConfig, manifest: dict[str, Any], evaluation: ReconstructionEvaluation) → None	O(N) typical	Path, _json_dump, asdict, list	entry point / dynamic / unreferenced statically

preprocessing.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_finite_number **L45**	Internal helper for finite number in preprocessing.py.	(value: object) → bool	O(1) excluding delegated I/O/model work	float, isfinite, isinstance	preprocessing.py:SensorPreprocessor._apply_glitch_rule, preprocessing.py:SensorPreprocessor._initial_row
SensorPreprocessor.__init__ **L56**	Initializes the object and validates/loads required collaborators.	(self, project: ProjectConfig, checkpoint_id: str) → None	O(1) excluding delegated I/O/model work	checkpoint	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
SensorPreprocessor.process **L61**	Callable for process in preprocessing.py.	(self, raw_records: Iterable[RawSensorRecord], *, split_id: str) → PreprocessingBatch	O(N·F) typical; includes ordering	PreprocessingBatch, ValueError, _apply_glitch_rule, _detect_frozen_values, _expand_to_one_hz, _finalize, _initial_row, _interpolate_short_missing_runs…	application_service.py:CheckpointApplicationService._latest_preprocessing_state, demo_control.py:DemoController._capture_quality, sudden_runtime.py:SuddenRuntimeScorer.score_records, synthetic_evaluation.py:run_synthetic_offline_evaluation, training.py:prepare_temporal_model_data…
SensorPreprocessor._validate_and_order **L79**	Internal helper for validate and order in preprocessing.py.	(self, records: list[RawSensorRecord]) → tuple[list[RawSensorRecord], list[QuarantinedRecord]]	O(N) typical	QuarantinedRecord, add, append, max, set, sort, timedelta	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._expand_to_one_hz **L105**	Internal helper for expand to one hz in preprocessing.py.	(self, records: list[RawSensorRecord]) → tuple[list[tuple[datetime, RawSensorRecord | None]], list[QuarantinedRecord]]	O(N) typical	QuarantinedRecord, append, int, is_integer, range, timedelta, total_seconds	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._initial_row **L126**	Internal helper for initial row in preprocessing.py.	(self, timestamp: datetime, raw: RawSensorRecord | None) → _WorkingRow	O(N) typical	_WorkingRow, _finite_number, add, float, getattr, set	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._detect_frozen_values **L158**	Internal helper for detect frozen values in preprocessing.py.	(self, rows: list[_WorkingRow]) → None	O(N·M) / data-dependent loops	add, timedelta	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._apply_glitch_rule **L188**	Internal helper for apply glitch rule in preprocessing.py.	(self, rows: list[_WorkingRow]) → None	O(N·M) / data-dependent loops	QualityDecision, _finite_number, abs, append, discard, float, getattr, items…	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._interpolate_short_missing_runs **L273**	Internal helper for interpolate short missing runs in preprocessing.py.	(self, rows: list[_WorkingRow]) → None	O(N·M) / data-dependent loops	QualityDecision, add, append, enumerate, len, range	preprocessing.py:SensorPreprocessor.process
SensorPreprocessor._finalize **L324**	Internal helper for finalize in preprocessing.py.	(self, rows: list[_WorkingRow], split_id: str) → list[ProcessedInferenceRecord]	O(N log N) typical	ProcessedInferenceRecord, all, append, join, sorted, tuple	preprocessing.py:SensorPreprocessor.process

runtime_store.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_record_from_dict **L18**	Internal helper for record from dict in runtime_store.py.	(raw: Mapping[str, Any]) → RawSensorRecord	O(1) excluding delegated I/O/model work	RawSensorRecord, float, get, parse_utc_datetime, str	runtime_store.py:RuntimeBufferState.from_dict
RuntimeBufferState.__post_init__ **L40**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, any, format_utc, len, zip	entry point / dynamic / unreferenced statically
RuntimeBufferState.to_dict **L59**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N) typical	as_dict	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
RuntimeBufferState.from_dict **L69**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'RuntimeBufferState'	O(N) typical	ValueError, _record_from_dict, cls, get, int, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
JsonRuntimeBufferStore.__init__ **L83**	Initializes the object and validates/loads required collaborators.	(self, path: str | Path, *, checkpoint_id: str, max_readings: int=60) → None	O(1) excluding delegated I/O/model work	Path, RuntimeBufferState, ValueError, _load, exists	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
JsonRuntimeBufferStore._load **L101**	Loads and validates persisted local state.	(self) → RuntimeBufferState	O(1) excluding delegated I/O/model work	ValueError, from_dict, load, open	incidents.py:JsonIncidentStore.__init__, runtime_store.py:JsonRuntimeBufferStore.__init__, trend_warnings.py:JsonTrendWarningStore.__init__
JsonRuntimeBufferStore.save **L108**	Persists the artifact or state document.	(self, *, force: bool=False) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, to_dict, with_name, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration…
JsonRuntimeBufferStore.set_deferred_save_interval **L123**	Callable for set deferred save interval in runtime_store.py.	(self, reading_writes: int=1) → None	O(1) excluding delegated I/O/model work	ValueError, isinstance, save	demo_control.py:DemoController._run
JsonRuntimeBufferStore.flush **L135**	Callable for flush in runtime_store.py.	(self) → None	O(1) excluding delegated I/O/model work	save	demo_control.py:DemoController._run
JsonRuntimeBufferStore.append **L138**	Callable for append in runtime_store.py.	(self, record: RawSensorRecord) → None	O(1) excluding delegated I/O/model work	ValueError, append, format_utc, is_integer, save, total_seconds	application_service.py:CheckpointApplicationService.ingest_reading, demo_control.py:DemoController._append_event_to, demo_control.py:DemoController._capture_scores_and_notifications, demo_control.py:DemoController._scenario_points, demo_control.py:DemoController.run_preset…
JsonRuntimeBufferStore.reset **L155**	Clear only the bounded runtime buffer and persist the empty state.	(self) → None	O(1) excluding delegated I/O/model work	RuntimeBufferState, save	api_demo_routes.py:reset_demo, application_service.py:CheckpointApplicationService.reset_monitoring_runtime, demo_control.py:DemoController.run_preset, slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation…

scaling.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_quantile **L19**	Internal helper for quantile in scaling.py.	(values: Sequence[float], probability: float) → float	O(N log N)	ScalerFitError, ceil, floor, len, sorted	scaling.py:RobustScaler.fit, thresholds.py:derive_provisional_thresholds, trend_config.py:derive_provisional_trend_config
RobustScaler.fit **L49**	Callable for fit in scaling.py.	(cls, records: Iterable[ProcessedInferenceRecord], *, checkpoint_id: str, settings: PreprocessingConfig, approved_heal...) → 'RobustScaler'	O(N log N) typical	ScalerFitError, _quantile, any, append, cls, format_utc, isfinite, items...	training.py:prepare_temporal_model_data, training.py:prepare_uploaded_model_data
RobustScaler.transform **L105**	Callable for transform in scaling.py.	(self, values: Mapping[str, float | None]) → tuple[float, ...]	O(N) typical	ValueError, append, isfinite, keys, tuple, zip	windowing.py:create_sliding_windows
RobustScaler.inverse_transform **L118**	Callable for inverse transform in scaling.py.	(self, values: Sequence[float]) → tuple[float, ...]	O(N) typical	ValueError, len, tuple, zip	entry point / dynamic / unreferenced statically
RobustScaler.to_dict **L126**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	dict, list, zip	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident...
RobustScaler.save **L142**	Persists the artifact or state document.	(self, path: str | Path) → None	O(1) excluding delegated I/O/model work	Path, dump, mkdir, open, to_dict, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration...
RobustScaler.load **L150**	Loads and validates a persisted artifact.	(cls, path: str | Path) → 'RobustScaler'	O(N·M) / data-dependent loops	Path, ScalerFitError, any, cls, float, int, isinstance, load...	application_service.py:CheckpointApplicationService._initialize_runtime_configurations, config.py:load_config, demo_control.py:DemoController._load_or_initialize, incidents.py:JsonIncidentStore._load, model_lifecycle.py:ModelLifecycleManager._load_or_initialize...

schemas.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_require_utc **L88**	Internal helper for require utc in schemas.py.	(value: datetime, field_name: str) → None	O(1) excluding delegated I/O/model work	ValueError, isinstance, timedelta, utcoffset	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.acknowledge_warning, application_service.py:CheckpointApplicationService.classify_incident, application_service.py:CheckpointApplicationService.classify_warning, application_service.py:CheckpointApplicationService.reconstruction_history...
_require_numeric_optional **L95**	Internal helper for require numeric optional in schemas.py.	(value: float | None, field_name: str) → None	O(1) excluding delegated I/O/model work	ValueError, isinstance	schemas.py:RawSensorRecord.__post_init__, schemas.py:_require_finite_optional
_require_finite_optional **L102**	Internal helper for require finite optional in schemas.py.	(value: float | None, field_name: str) → None	O(1) excluding delegated I/O/model work	ValueError, _require_numeric_optional, isfinite	schemas.py:ProcessedInferenceRecord.__post_init__
format_utc **L108**	Return a stable ISO-8601 UTC representation ending in ``Z``.	(value: datetime) → str	O(1) excluding delegated I/O/model work	_require_utc, astimezone, isoformat, replace	application_service.py:CheckpointApplicationService.checkpoint_status, application_service.py:CheckpointApplicationService.ingest_reading, application_service.py:CheckpointApplicationService.reconstruction_history, demo_control.py:DemoController.__init__, demo_control.py:DemoController._capture_quality...
RawSensorRecord.__post_init__ **L132**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _require_numeric_optional, _require_utc, getattr, isinstance, strip	entry point / dynamic / unreferenced statically

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
RawSensorRecord.model_features **L148**	Callable for model features in schemas.py.	(self) → dict[str, float | None]	O(N) typical	getattr	application_service.py:CheckpointApplicationService.checkpoint_status, demo_control.py:DemoController._capture_quality
RawSensorRecord.as_dict **L151**	Callable for as dict in schemas.py.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	format_utc	demo_control.py:DemoController._payload_point, demo_control.py:DemoController._process_point, runtime_store.py:RuntimeBufferState.to_dict, synthetic.py:write_csv
QualityDecision.__post_init__ **L183**	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	ValueError	entry point / dynamic / unreferenced statically
ProcessedInferenceRecord.__post_init__ **L213**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _require_finite_optional, _require_utc, fields, getattr, isinstance, startswith, type	entry point / dynamic / unreferenced statically
ProcessedInferenceRecord.processed_features **L234**	Callable for processed features in schemas.py.	(self) → dict[str, float | None]	O(1) excluding delegated I/O/model work	—	demo_control.py:DemoController._capture_quality, scaling.py:RobustScaler.fit, windowing.py:create_sliding_windows
ProcessedInferenceRecord.quality_flags **L244**	Callable for quality flags in schemas.py.	(self) → dict[str, DataQualityFlag]	O(1) excluding delegated I/O/model work	—	demo_control.py:DemoController._capture_quality, synthetic_evaluation.py:_quality_counts, synthetic_evaluation.py:_window_quality_context, windowing.py:create_sliding_windows
EvaluationLabels.__post_init__ **L266**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _require_utc, any	entry point / dynamic / unreferenced statically

slow_trend_evaluation.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
SlowTrendEvaluationConfig.from_dict **L62**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SlowTrendEvaluationConfig'	O(N) typical	Path, ValueError, _validate, cls, float, from_dict, int, parse_utc_datetime...	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict...
SlowTrendEvaluationConfig._validate **L96**	Internal helper for validate in slow_trend_evaluation.py.	(self) → None	O(N) typical	ValueError, any, int, len, total_seconds	slow_trend_evaluation.py:SlowTrendEvaluationConfig.from_dict, synthetic_evaluation.py:ScenarioConfig.from_dict, synthetic_evaluation.py:SyntheticEvaluationConfig.from_dict
load_slow_trend_evaluation_config **L124**	Callable for load slow trend evaluation config in slow_trend_evaluation.py.	(path: str | Path) → SlowTrendEvaluationConfig	O(1) excluding delegated I/O/model work	Path, ValueError, from_dict, isinstance, load, open	demo_control.py:DemoController.__init__
_resolve **L137**	Internal helper for resolve in slow_trend_evaluation.py.	(path: Path, root: Path) → Path	O(1) excluding delegated I/O/model work	is_absolute	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_protected_hashes, synthetic_evaluation.py:run_synthetic_offline_evaluation
_reference **L141**	Internal helper for reference in slow_trend_evaluation.py.	(path: Path, root: Path) → str	O(1) excluding delegated I/O/model work	relative_to, replace, resolve, str	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_write_held_out_metadata, synthetic_evaluation.py:_write_scenario_metadata, synthetic_evaluation.py:run_synthetic_offline_evaluation
_sha256 **L148**	Internal helper for sha256 in slow_trend_evaluation.py.	(path: Path) → str	O(N) typical	hexdigest, iter, open, read, sha256, update, upper	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_protected_hashes, synthetic_evaluation.py:_write_held_out_metadata, synthetic_evaluation.py:_write_scenario_metadata, synthetic_evaluation.py:run_synthetic_offline_evaluation

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_json_dump **L156**	Internal helper for json dump in slow_trend_evaluation.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, with_name, write	model_package.py:save_model_package, model_package.py:write_evaluation_report, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios, synthetic_evaluation.py:_write_held_out_metadata…
_history_from_scores **L165**	Internal helper for history from scores in slow_trend_evaluation.py.	(scores: Sequence[Any], *, model_version: str, start: datetime, end: datetime) → tuple[ReconstructionHistoryRecord, …]	O(N) typical	from_runtime_score, tuple	slow_trend_evaluation.py:run_slow_trend_evaluation
_write_history_csv **L181**	Internal helper for write history csv in slow_trend_evaluation.py.	(path: Path, history: Sequence[ReconstructionHistoryRecord]) → None	O(N·M) / data-dependent loops	DictWriter, format_utc, mkdir, open, writeheader, writerow	slow_trend_evaluation.py:run_slow_trend_evaluation
_write_bucket_csv **L218**	Internal helper for write bucket csv in slow_trend_evaluation.py.	(path: Path, buckets: Sequence[TrendBucket]) → None	O(N·M) / data-dependent loops	DictWriter, format_utc, mkdir, open, writeheader, writerow	slow_trend_evaluation.py:run_slow_trend_evaluation
_write_evaluation_csv **L263**	Internal helper for write evaluation csv in slow_trend_evaluation.py.	(path: Path, evaluations: Sequence[TrendEvaluation]) → None	O(N·M) / data-dependent loops	DictWriter, format_utc, mkdir, open, writeheader, writerow	slow_trend_evaluation.py:run_slow_trend_evaluation
_controlled_evaluation **L320**	Internal helper for controlled evaluation in slow_trend_evaluation.py.	(config: SlowTrendConfig, evaluated_at: datetime, gradient: float | None) → TrendEvaluation	O(N) typical	TrendEvaluation, enumerate, strftime, timedelta	slow_trend_evaluation.py:_controlled_lifecycle_demo
_controlled_lifecycle_demo **L370**	Internal helper for controlled lifecycle demo in slow_trend_evaluation.py.	(config: SlowTrendConfig, store_path: Path) → dict[str, Any]	O(N·M) / data-dependent loops	JsonTrendWarningStore, Path, SlowTrendWarningDetector, TemporaryDirectory, _controlled_evaluation, append, bool, enumerate…	demo_control.py:DemoController.controlled_warning_lifecycle, slow_trend_evaluation.py:run_slow_trend_evaluation
_missing_data_demo **L447**	Internal helper for missing data demo in slow_trend_evaluation.py.	(healthy_history: Sequence[ReconstructionHistoryRecord], config: SlowTrendConfig, evaluated_at: datetime) → dict[str, Any]	O(N) typical	aggregate_five_minute_medians, evaluate_trend_at, replace, timedelta, to_dict, tuple	slow_trend_evaluation.py:run_slow_trend_evaluation
_scenario_summary **L480**	Internal helper for scenario summary in slow_trend_evaluation.py.	(*, scenario: ScenarioConfig, event: Mapping[str, Any], buckets: Sequence[TrendBucket], evaluations: Sequence[TrendEva…) → dict[str, Any]	O(N log N) typical	append, bool, dict, float, format_utc, get, items, latest_valid_outcomes…	slow_trend_evaluation.py:run_slow_trend_evaluation
run_slow_trend_evaluation **L614**	Score long held-out gradual scenarios without retraining or scaler fitting.	(project: ProjectConfig, evaluation_config: SlowTrendEvaluationConfig, *, project_root: str | Path='.') → dict[str, Any]	O(N·M) / data-dependent loops	JsonTrendWarningStore, Path, RuntimeError, SlowTrendWarningDetector, SuddenRuntimeScorer, ValueError, _controlled_lifecycle_demo, _history_from_scores…	entry point / dynamic / unreferenced statically

sudden_runtime.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_parse_optional_float **L39**	Internal helper for parse optional float in sudden_runtime.py.	(value: str) → float | None	O(1) excluding delegated I/O/model work	float	sudden_runtime.py:read_raw_csv
_parse_optional_bool **L43**	Internal helper for parse optional bool in sudden_runtime.py.	(value: str) → bool | None	O(1) excluding delegated I/O/model work	ValueError, lower, strip	sudden_runtime.py:read_raw_csv

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
read_raw_csv **L54**	Callable for read raw csv in sudden_runtime.py.	(path: str | Path) → tuple[RawSensorRecord, ...]	O(N) typical	DictReader, Path, RawSensorRecord, _parse_optional_bool, _parse_optional_float, append, int, isdigit...	sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios
SuddenRuntimeScorer.__init__ **L93**	Initializes the object and validates/loads required collaborators.	(self, project: ProjectConfig, package_path: str | Path, *, checkpoint_id: str='checkpoint_01') → None	O(1) excluding delegated I/O/model work	ValueError, load_model_package, set_num_threads	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
SuddenRuntimeScorer.score_records **L109**	Callable for score records in sudden_runtime.py.	(self, raw_records: Iterable[RawSensorRecord]) → RuntimeScoringBatch	O(N·W·F + windows·model)	Counter, RuntimeScore, RuntimeScoringBatch, SensorPreprocessor, _reconstruction_scores, append, create_sliding_windows, dict...	application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios
SuddenRuntimeScorer._reconstruction_scores **L180**	Internal helper for reconstruction scores in sudden_runtime.py.	(self, windows: Sequence[Any]) → dict[datetime, dict[str, float]]	O(windows·model)	ValueError, append, cat, enumerate, eval, float, inference_mode, len...	sudden_runtime.py:SuddenRuntimeScorer.score_records
_json_dump **L212**	Internal helper for json dump in sudden_runtime.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, write	model_package.py:save_model_package, model_package.py:write_evaluation_report, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios, synthetic_evaluation.py:_write_held_out_metadata...
_score **L219**	Internal helper for score in sudden_runtime.py.	(timestamp: datetime, overall: float | None, *, scoreable: bool=True, monitoring_state: str='available') → RuntimeScore	O(N) typical	RuntimeScore	sudden_runtime.py:run_controlled_lifecycle_demo
_replay_to_fresh_store **L248**	Internal helper for replay to fresh store in sudden_runtime.py.	(scores: Sequence[RuntimeScore], thresholds: SuddenThresholdConfig, path: Path) → dict[str, object]	O(1) excluding delegated I/O/model work	JsonIncidentStore, SuddenIncidentDetector, replay, reset, to_dict	sudden_runtime.py:run_controlled_lifecycle_demo
run_controlled_lifecycle_demo **L260**	Exercise classification, restart, recovery pause/reset, and closure.	(thresholds: SuddenThresholdConfig, *, store_path: str | Path) → dict[str, object]	O(N·M) / data-dependent loops	JsonIncidentStore, Path, SuddenIncidentDetector, TemporaryDirectory, _replay_to_fresh_store, _score, append, classify_active...	sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios
_evaluate_replay **L378**	Internal helper for evaluate replay in sudden_runtime.py.	(scores: Sequence[RuntimeScore], thresholds: SuddenThresholdConfig, *, event: Mapping[str, Any] | None) → dict[str, Any]	O(N·M) / data-dependent loops	JsonIncidentStore, Path, SuddenIncidentDetector, TemporaryDirectory, format_utc, len, list, parse_utc_datetime...	sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios
derive_thresholds_and_evaluate_sudden_scenarios **L479**	Callable for derive thresholds and evaluate sudden scenarios in sudden_runtime.py.	(project: ProjectConfig, *, package_path: str | Path, held_out_healthy_path: str | Path, blockage_dataset_path: str | ...) → dict[str, Any]	O(N·M) / data-dependent loops	Path, SuddenRuntimeScorer, _evaluate_replay, _json_dump, append, derive_provisional_thresholds, float, len...	entry point / dynamic / unreferenced statically

synthetic.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
parse_utc_datetime **L26**	Callable for parse utc datetime in synthetic.py.	(value: str) → datetime	O(1) excluding delegated I/O/model work	ValueError, astimezone, fromisoformat, replace, strip, timedelta, utcoffset	demo_control.py:DemoController._process_point, demo_control.py:DemoController.classify_bottle_jam, incidents.py:RuntimeScore.from_dict, runtime_store.py:_record_from_dict, slow_trend_evaluation.py:SlowTrendEvaluationConfig.from_dict...
_clamp **L37**	Internal helper for clamp in synthetic.py.	(value: float, minimum: float, maximum: float) → float	O(1) excluding delegated I/O/model work	max, min	synthetic.py:_checkpoint_records
_checkpoint_records **L41**	Internal helper for checkpoint records in synthetic.py.	(project: ProjectConfig, checkpoint: CheckpointGenerationConfig, start_time: datetime, duration_seconds: int) → Iterator[RawSensorRecord]	O(N) typical	Random, RawSensorRecord, _clamp, gauss, max, min, range, round...	synthetic.py:generate_healthy_records
generate_healthy_records **L132**	Yield anomaly-free raw records, grouped by checkpoint.	(project: ProjectConfig, start_time: datetime, duration_seconds: int | None=None, checkpoint_ids: Sequence[str] | None...) → Iterator[RawSensorRecord]	O(N) typical	ValueError, _checkpoint_records, checkpoint, isinstance, len, list, set, timedelta...	demo_control.py:DemoController._ensure_source, slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic.py:generate_dataset, synthetic_evaluation.py:run_synthetic_offline_evaluation, training.py:prepare_temporal_model_data
write_csv **L163**	Callable for write csv in synthetic.py.	(records: Iterable[RawSensorRecord], path: str | Path) → int	O(N) typical	DictWriter, Path, as_dict, mkdir, open, writeheader, writerow	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic.py:generate_dataset, synthetic_evaluation.py:run_synthetic_offline_evaluation, training.py:write_demo_dataset
write_metadata_report **L176**	Callable for write metadata report in synthetic.py.	(path: str | Path, *, project: ProjectConfig, start_time: datetime, duration_seconds: int, checkpoint_ids: Sequence[st...) → None	O(1) excluding delegated I/O/model work	Path, dump, format_utc, len, list, mkdir, open, write	synthetic.py:generate_dataset
generate_dataset **L215**	Callable for generate dataset in synthetic.py.	(project: ProjectConfig, *, start_time: datetime, duration_seconds: int, checkpoint_ids: Sequence[str], output_path: s...) → int	O(1) excluding delegated I/O/model work	generate_healthy_records, write_csv, write_metadata_report	synthetic.py:main
build_parser **L242**	Callable for build parser in synthetic.py.	() → argparse.ArgumentParser	O(1) excluding delegated I/O/model work	ArgumentParser, Path, add_argument	synthetic.py:main, windows_app.py:main
main **L262**	Callable for main in synthetic.py.	(argv: Sequence[str] | None=None) → int	O(N) typical	build_parser, dumps, generate_dataset, load_config, parse_args, parse_utc_datetime, print, str	entry point / dynamic / unreferenced statically

synthetic_evaluation.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_require_mapping **L86**	Internal helper for require mapping in synthetic_evaluation.py.	(raw: Mapping[str, Any], key: str) → Mapping[str, Any]	O(1) excluding delegated I/O/model work	SyntheticEvaluationConfigurationError, get, isinstance	synthetic_evaluation.py:ScenarioConfig.from_dict, synthetic_evaluation.py:SyntheticEvaluationConfig.from_dict
_require_integer **L93**	Internal helper for require integer in synthetic_evaluation.py.	(raw: Mapping[str, Any], key: str) → int	O(1) excluding delegated I/O/model work	SyntheticEvaluationConfigurationError, get, isinstance	synthetic_evaluation.py:HeldOutEvaluationConfig.from_dict, synthetic_evaluation.py:ScenarioConfig.from_dict, synthetic_evaluation.py:SyntheticEvaluationConfig.from_dict
_require_number **L100**	Internal helper for require number in synthetic_evaluation.py.	(raw: Mapping[str, Any], key: str) → float	O(1) excluding delegated I/O/model work	SyntheticEvaluationConfigurationError, float, get, isinstance	config.py:ModelConfig.from_dict, config.py:PreprocessingConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:SensorQualityConfig.from_dict, synthetic_evaluation.py:ScenarioConfig._validate

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
HeldOutEvaluationConfig.from_dict **L115**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'HeldOutEvaluationConfig'	O(1) excluding delegated I/O/model work	Path, SyntheticEvaluationConfigurationError, _require_integer, cls, parse_utc_datetime, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
ScenarioConfig.total_event_seconds **L147**	Callable for total event seconds in synthetic_evaluation.py.	(self) → int	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
ScenarioConfig.from_dict **L151**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'ScenarioConfig'	O(N) typical	SyntheticEvaluationConfigurationError, _require_integer, _require_mapping, _validate, cls, dict, get, str…	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
ScenarioConfig._validate **L176**	Internal helper for validate in synthetic_evaluation.py.	(self) → None	O(N·M) / data-dependent loops	SyntheticEvaluationConfigurationError, _require_number, any, get, isinstance, keys, len, set…	slow_trend_evaluation.py:SlowTrendEvaluationConfig.from_dict, synthetic_evaluation.py:ScenarioConfig.from_dict, synthetic_evaluation.py:SyntheticEvaluationConfig.from_dict
SyntheticEvaluationConfig.from_dict **L260**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SyntheticEvaluationConfig'	O(N) typical	Path, SyntheticEvaluationConfigurationError, _require_integer, _require_mapping, _validate, cls, from_dict, str…	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
SyntheticEvaluationConfig._validate **L285**	Internal helper for validate in synthetic_evaluation.py.	(self) → None	O(N·M) / data-dependent loops	SyntheticEvaluationConfigurationError, len, set	slow_trend_evaluation.py:SlowTrendEvaluationConfig.from_dict, synthetic_evaluation.py:ScenarioConfig.from_dict, synthetic_evaluation.py:SyntheticEvaluationConfig.from_dict
SyntheticEvent.to_dict **L353**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, Any]	O(1) excluding delegated I/O/model work	format_utc, list	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
load_synthetic_evaluation_config **L375**	Callable for load synthetic evaluation config in synthetic_evaluation.py.	(path: str | Path) → SyntheticEvaluationConfig	O(1) excluding delegated I/O/model work	Path, SyntheticEvaluationConfigurationError, from_dict, isinstance, load, open	demo_control.py:DemoController.__init__
_event_for **L393**	Internal helper for event for in synthetic_evaluation.py.	(healthy_start: datetime, checkpoint_id: str, scenario: ScenarioConfig, seed: int) → SyntheticEvent	O(1) excluding delegated I/O/model work	SyntheticEvent, dict, timedelta, update	synthetic_evaluation.py:inject_scenario
_profile_gain **L429**	Internal helper for profile gain in synthetic_evaluation.py.	(relative_second: int, scenario: ScenarioConfig) → float	O(1) excluding delegated I/O/model work	max	synthetic_evaluation.py:inject_scenario
inject_scenario **L443**	Return a new scenario dataset without mutating the healthy source records.	(healthy_records: Sequence[RawSensorRecord], scenario: ScenarioConfig, *, seed: int) → InjectedScenarioDataset	O(N·M) / data-dependent loops	InjectedScenarioDataset, ValueError, _event_for, _profile_gain, any, append, enumerate, float…	demo_control.py:DemoController._scenario_points, slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:run_synthetic_offline_evaluation
_json_dump **L519**	Internal helper for json dump in synthetic_evaluation.py.	(path: Path, value: object) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, write	model_package.py:save_model_package, model_package.py:write_evaluation_report, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios, synthetic_evaluation.py:_write_held_out_metadata…
_sha256 **L526**	Internal helper for sha256 in synthetic_evaluation.py.	(path: Path) → str	O(N) typical	hexdigest, iter, open, read, sha256, update	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_protected_hashes, synthetic_evaluation.py:_write_held_out_metadata, synthetic_evaluation.py:_write_scenario_metadata, synthetic_evaluation.py:run_synthetic_offline_evaluation
_resolve **L534**	Internal helper for resolve in synthetic_evaluation.py.	(path: Path, project_root: Path) → Path	O(1) excluding delegated I/O/model work	is_absolute	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_protected_hashes, synthetic_evaluation.py:run_synthetic_offline_evaluation

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_reference **L538**	Internal helper for reference in synthetic_evaluation.py.	(path: Path, project_root: Path) → str	O(1) excluding delegated I/O/model work	relative_to, str	slow_trend_evaluation.py:run_slow_trend_evaluation, synthetic_evaluation.py:_write_held_out_metadata, synthetic_evaluation.py:_write_scenario_metadata, synthetic_evaluation.py:run_synthetic_offline_evaluation
_protected_hashes **L545**	Internal helper for protected hashes in synthetic_evaluation.py.	(loaded: LoadedModelPackage, project_root: Path) → dict[str, str]	O(1) excluding delegated I/O/model work	Path, _resolve, _sha256, exists, str	synthetic_evaluation.py:run_synthetic_offline_evaluation
_distribution **L562**	Internal helper for distribution in synthetic_evaluation.py.	(values: Sequence[float]) → dict[str, float | int] | None	O(N log N) typical	ceil, floor, fmean, len, max, median, min, pstdev…	synthetic_evaluation.py:_summarize_scenario, training.py:evaluate_reconstruction
_score_windows **L583**	Internal helper for score windows in synthetic_evaluation.py.	(loaded: LoadedModelPackage, windows: Sequence[ModelWindow], *, batch_size: int) → dict[datetime, dict[str, float]]	O(N·M) / data-dependent loops	ValueError, append, cat, enumerate, eval, float, inference_mode, len…	synthetic_evaluation.py:run_synthetic_offline_evaluation
_quality_counts **L614**	Internal helper for quality counts in synthetic_evaluation.py.	(batch: PreprocessingBatch) → dict[str, dict[str, int]]	O(N log N) typical	Counter, dict, items, quality_flags, sorted	synthetic_evaluation.py:_summarize_scenario
_window_quality_context **L627**	Internal helper for window quality context in synthetic_evaluation.py.	(processed: PreprocessingBatch, end_timestamp: datetime) → tuple[bool | None, bool, int, int]	O(N·M) / data-dependent loops	enumerate, len, next, quality_flags, set, sum, values	synthetic_evaluation.py:_build_history
_event_relation **L646**	Internal helper for event relation in synthetic_evaluation.py.	(start: datetime, end: datetime, event: SyntheticEvent) → tuple[str, bool, bool, int | None]	O(1) excluding delegated I/O/model work	int, total_seconds	synthetic_evaluation.py:_build_history
_build_history **L666**	Internal helper for build history in synthetic_evaluation.py.	(processed: PreprocessingBatch, window_batch: WindowBatch, scores: Mapping[datetime, Mapping[str, float]], event: Synt…) → list[dict[str, Any]]	O(N log N) typical	_event_relation, _window_quality_context, append, dumps, format_utc, get, join, set…	synthetic_evaluation.py:run_synthetic_offline_evaluation
_write_history **L740**	Internal helper for write history in synthetic_evaluation.py.	(path: Path, history: Sequence[Mapping[str, Any]]) → None	O(N) typical	DictWriter, get, mkdir, open, writeheader, writerow	synthetic_evaluation.py:run_synthetic_offline_evaluation
_ols_slope **L749**	Internal helper for ols slope in synthetic_evaluation.py.	(xs: Sequence[float], ys: Sequence[float]) → float | None	O(N) typical	fmean, len, sum, zip	synthetic_evaluation.py:_summarize_scenario
_summarize_scenario **L763**	Internal helper for summarize scenario in synthetic_evaluation.py.	(scenario: ScenarioConfig, event: SyntheticEvent, processed: PreprocessingBatch, window_batch: WindowBatch, history: S…) → dict[str, Any]	O(N log N) typical	Counter, _distribution, _ols_slope, _quality_counts, bool, dict, float, fmean…	synthetic_evaluation.py:run_synthetic_offline_evaluation
_write_held_out_metadata **L926**	Internal helper for write held out metadata in synthetic_evaluation.py.	(path: Path, *, project: ProjectConfig, config: SyntheticEvaluationConfig, dataset_path: Path, records: Sequence[RawSe…) → dict[str, Any]	O(1) excluding delegated I/O/model work	_json_dump, _reference, _sha256, format_utc, len, list	synthetic_evaluation.py:run_synthetic_offline_evaluation
_write_scenario_metadata **L962**	Internal helper for write scenario metadata in synthetic_evaluation.py.	(path: Path, *, config: SyntheticEvaluationConfig, event: SyntheticEvent, dataset_path: Path, healthy_path: Path, heal…) → None	O(1) excluding delegated I/O/model work	_json_dump, _reference, _sha256, list, to_dict	synthetic_evaluation.py:run_synthetic_offline_evaluation
run_synthetic_offline_evaluation **L1010**	Generate and score all configured scenarios using the saved package only.	(project: ProjectConfig, config: SyntheticEvaluationConfig, *, project_root: str | Path='.') → dict[str, Any]	O(N·M) / data-dependent loops	Path, RuntimeError, SensorPreprocessor, ValueError, _build_history, _json_dump, _protected_hashes, _reference…	entry point / dynamic / unreferenced statically

thresholds.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_quantile **L24**	Internal helper for quantile in thresholds.py.	(values: Sequence[float], probability: float) → float	O(N log N)	ThresholdConfigurationError, any, ceil, floor, isfinite, len, sorted	scaling.py:RobustScaler.fit, thresholds.py:derive_provisional_thresholds, trend_config.py:derive_provisional_trend_config
ThresholdSetting.__post_init__ **L50**	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	ThresholdConfigurationError, all, isfinite	entry point / dynamic / unreferenced statically
ThresholdSetting.to_dict **L58**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
ThresholdSetting.from_dict **L69**	Validates and reconstructs the domain object from persisted data.	(cls, raw: dict[str, Any]) → 'ThresholdSetting'	O(1) excluding delegated I/O/model work	ThresholdConfigurationError, cls, float, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
ThresholdAuditEntry.to_dict **L92**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
ThresholdAuditEntry.from_dict **L103**	Validates and reconstructs the domain object from persisted data.	(cls, raw: dict[str, Any]) → 'ThresholdAuditEntry'	O(1) excluding delegated I/O/model work	ThresholdConfigurationError, cls, float, get, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
SuddenThresholdConfig.__post_init__ **L130**	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	ThresholdConfigurationError, all	entry point / dynamic / unreferenced statically
SuddenThresholdConfig.to_dict **L153**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N) typical	to_dict	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
SuddenThresholdConfig.save **L172**	Persists the artifact or state document.	(self, path: str | Path) → None	O(1) excluding delegated I/O/model work	Path, dump, mkdir, open, replace_with_retry, to_dict, with_name, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration…
SuddenThresholdConfig.load **L182**	Loads and validates a persisted artifact.	(cls, path: str | Path) → 'SuddenThresholdConfig'	O(N) typical	Path, ThresholdConfigurationError, cls, from_dict, int, isinstance, load, open…	application_service.py:CheckpointApplicationService._initialize_runtime_configurations, config.py:load_config, demo_control.py:DemoController._load_or_initialize, incidents.py:JsonIncidentStore._load, model_lifecycle.py:ModelLifecycleManager._load_or_initialize…
SuddenThresholdConfig.with_updates **L212**	Callable for with updates in thresholds.py.	(self, *, spike_threshold: float | None=None, recovery_threshold: float | None=None, updated_at: datetime, updated_by:…) → 'SuddenThresholdConfig'	O(1) excluding delegated I/O/model work	ThresholdAuditEntry, ThresholdConfigurationError, ThresholdSetting, append, float, format_utc, list, replace…	application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration, trend_config.py:SlowTrendConfig.with_threshold_updates
derive_provisional_thresholds **L281**	Derive spike p99.5 and an independent recovery p95 from healthy errors.	(healthy_overall_errors: Sequence[float], *, checkpoint_id: str, model_version: str, scaler_version: str, configuratio…) → SuddenThresholdConfig	O(N) typical	SuddenThresholdConfig, ThresholdAuditEntry, ThresholdConfigurationError, ThresholdSetting, _quantile, float, format_utc, tuple	model_lifecycle.py:ModelLifecycleManager._run_training, sudden_runtime.py:derive_thresholds_and_evaluate_sudden_scenarios

training.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
configure_determinism L75	Callable for configure determinism in training.py.	(seed: int, cpu_threads: int) → None	O(1) excluding delegated I/O/model work	manual_seed, seed, set_num_threads, use_deterministic_algorithms	training.py:train_autoencoder
prepare_temporal_model_data L82	Callable for prepare temporal model data in training.py.	(project: ProjectConfig, *, checkpoint_id: str, start_time: datetime, training_duration_seconds: int, validation_durat…) → PreparedModelData	O(N) typical	PreparedModelData, SensorPreprocessor, ValueError, any, create_sliding_windows, fit, generate_healthy_records, process…	entry point / dynamic / unreferenced statically
prepare_uploaded_model_data L152	Prepare an operator-approved healthy upload without crossing its temporal split.	(project: ProjectConfig, records: Sequence[RawSensorRecord], *, checkpoint_id: str, validation_fraction: float, datase…) → PreparedModelData	O(N) typical	PreparedModelData, SensorPreprocessor, ValueError, create_sliding_windows, fit, int, len, process…	model_lifecycle.py:ModelLifecycleManager._run_training
windows_to_tensor L223	Callable for windows to tensor in training.py.	(windows: Sequence[ModelWindow]) → Tensor	O(N) typical	ValueError, len, tensor, tuple	model_package.py:save_model_package, training.py:evaluate_reconstruction, training.py:train_autoencoder
_mean_batch_loss L233	Internal helper for mean batch loss in training.py.	(model: LSTMAutoencoder, values: Tensor, *, batch_size: int, criterion: nn.Module) → float	O(N) typical	criterion, eval, float, inference_mode, len, model, range	training.py:train_autoencoder
train_autoencoder L250	Callable for train autoencoder in training.py.	(training_windows: Sequence[ModelWindow], validation_windows: Sequence[ModelWindow], *, model_config: ModelConfig, win…) → TrainingResult	O(E·N·T·H·(F+H))	Adam, Generator, MSELoss, RuntimeError, TrainingResult, ValueError, _mean_batch_loss, any…	model_lifecycle.py:ModelLifecycleManager._run_training
_distribution L359	Internal helper for distribution in training.py.	(values: Sequence[float]) → dict[str, float | int]	O(N log N) typical	ceil, floor, fmean, len, max, median, min, pstdev…	synthetic_evaluation.py:_summarize_scenario, training.py:evaluate_reconstruction
evaluate_reconstruction L378	Callable for evaluate reconstruction in training.py.	(model: LSTMAutoencoder, windows: Sequence[ModelWindow], *, batch_size: int) → ReconstructionEvaluation	O(N·M) / data-dependent loops	ReconstructionEvaluation, ReconstructionRecord, _distribution, append, cat, enumerate, eval, float…	model_lifecycle.py:ModelLifecycleManager._run_training
write_demo_dataset L430	Callable for write demo dataset in training.py.	(data: PreparedModelData, *, dataset_path: str | Path, metadata_path: str | Path, project: ProjectConfig) → None	O(1) excluding delegated I/O/model work	Path, dump, format_utc, len, list, mkdir, open, write…	entry point / dynamic / unreferenced statically

trend_config.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_quantile L24	Internal helper for quantile in trend_config.py.	(values: Sequence[float], probability: float) → float	O(N log N)	TrendConfigurationError, any, ceil, float, floor, isfinite, len, sorted	scaling.py:RobustScaler.fit, thresholds.py:derive_provisional_thresholds, trend_config.py:derive_provisional_trend_config
TrendThresholdSetting.__post_init__ L49	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	TrendConfigurationError, all, isfinite	entry point / dynamic / unreferenced statically
TrendThresholdSetting.to_dict L67	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
TrendThresholdSetting.from_dict **L79**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'TrendThresholdSetting'	O(1) excluding delegated I/O/model work	TrendConfigurationError, cls, float, isinstance, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
TrendAuditEntry.__post_init__ **L105**	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	TrendConfigurationError, isfinite	entry point / dynamic / unreferenced statically
TrendAuditEntry.to_dict **L111**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
TrendAuditEntry.from_dict **L122**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'TrendAuditEntry'	O(1) excluding delegated I/O/model work	TrendConfigurationError, cls, float, get, isinstance, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
SlowTrendConfig.__post_init__ **L159**	Validates dataclass invariants immediately after construction.	(self) → None	O(1) excluding delegated I/O/model work	TrendConfigurationError, all, is_integer	entry point / dynamic / unreferenced statically
SlowTrendConfig.expected_bucket_count **L200**	Callable for expected bucket count in trend_config.py.	(self) → int	O(1) excluding delegated I/O/model work	int	entry point / dynamic / unreferenced statically
SlowTrendConfig.to_dict **L203**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N) typical	to_dict	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
SlowTrendConfig.save **L235**	Persists the artifact or state document.	(self, path: str | Path) → None	O(1) excluding delegated I/O/model work	Path, dump, mkdir, open, replace_with_retry, to_dict, with_name, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration…
SlowTrendConfig.load **L245**	Loads and validates a persisted artifact.	(cls, path: str | Path) → 'SlowTrendConfig'	O(N) typical	Path, TrendConfigurationError, cls, float, from_dict, int, isinstance, load…	application_service.py:CheckpointApplicationService._initialize_runtime_configurations, config.py:load_config, demo_control.py:DemoController._load_or_initialize, incidents.py:JsonIncidentStore._load, model_lifecycle.py:ModelLifecycleManager._load_or_initialize…
SlowTrendConfig.with_threshold_updates **L294**	Callable for with threshold updates in trend_config.py.	(self, *, gradient_threshold: float | None=None, recovery_threshold: float | None=None, updated_at: datetime, updated_…) → 'SlowTrendConfig'	O(1) excluding delegated I/O/model work	with_updates	entry point / dynamic / unreferenced statically
SlowTrendConfig.with_updates **L311**	Callable for with updates in trend_config.py.	(self, *, gradient_threshold: float | None=None, recovery_threshold: float | None=None, trend_history_hours: float | N…) → 'SlowTrendConfig'	O(N) typical	TrendAuditEntry, TrendConfigurationError, TrendThresholdSetting, all, append, float, format_utc, items…	application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration, trend_config.py:SlowTrendConfig.with_threshold_updates
derive_provisional_trend_config **L403**	Create a synthetic provisional p99.5 healthy-gradient candidate.	(healthy_gradients: Sequence[float], *, checkpoint_id: str, model_version: str, scaler_version: str, source_configurat…) → SlowTrendConfig	O(1) excluding delegated I/O/model work	SlowTrendConfig, TrendAuditEntry, TrendThresholdSetting, _quantile, format_utc, max	slow_trend_evaluation.py:run_slow_trend_evaluation

trend_monitoring.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
_require_utc **L26**	Internal helper for require utc in trend_monitoring.py.	(value: datetime, name: str) → None	O(1) excluding delegated I/O/model work	ValueError, timedelta, utcoffset	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.acknowledge_warning, application_service.py:CheckpointApplicationService.classify_incident, application_service.py:CheckpointApplicationService.classify_warning, application_service.py:CheckpointApplicationService.reconstruction_history…
_floor_timestamp **L31**	Internal helper for floor timestamp in trend_monitoring.py.	(value: datetime, seconds: int) → datetime	O(1) excluding delegated I/O/model work	_require_utc, fromtimestamp, int, timestamp	trend_monitoring.py:_ceil_timestamp, trend_monitoring.py:aggregate_five_minute_medians
_ceil_timestamp **L39**	Internal helper for ceil timestamp in trend_monitoring.py.	(value: datetime, seconds: int) → datetime	O(1) excluding delegated I/O/model work	_floor_timestamp, timedelta	trend_monitoring.py:evaluate_trend_schedule
ols_slope_hours **L44**	Return OLS slope where x is measured in elapsed hours.	(xs: Sequence[float], ys: Sequence[float]) → float	O(N)	ValueError, any, isfinite, len, sum, zip	trend_monitoring.py:evaluate_trend_at
ReconstructionHistoryRecord.__post_init__ **L75**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _require_utc, any, isfinite, tuple, values	entry point / dynamic / unreferenced statically
ReconstructionHistoryRecord.from_runtime_score **L89**	Callable for from runtime score in trend_monitoring.py.	(cls, score: RuntimeScore, *, model_version: str) → 'ReconstructionHistoryRecord'	O(1) excluding delegated I/O/model work	cls, dict	application_service.py:CheckpointApplicationService._reconstruction_history_records, slow_trend_evaluation.py:_history_from_scores
ReconstructionHistoryRecord.to_dict **L104**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	format_utc	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
ReconstructionHistoryRecord.from_dict **L118**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'ReconstructionHistoryRecord'	O(N) typical	bool, cls, float, get, parse_utc_datetime, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
TrendBucket.__post_init__ **L159**	Validates dataclass invariants immediately after construction.	(self) → None	O(N) typical	ValueError, _require_utc, any, isclose, isfinite, tuple, values	entry point / dynamic / unreferenced statically
TrendBucket.to_dict **L181**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	format_utc	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
TrendEvaluation.is_valid **L224**	Callable for is valid in trend_monitoring.py.	(self) → bool	O(1) excluding delegated I/O/model work	—	entry point / dynamic / unreferenced statically
TrendEvaluation.to_dict **L227**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N log N) typical	float, format_utc, items, sorted	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident…
TrendEvaluation.from_dict **L264**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'TrendEvaluation'	O(N) typical	TrendEvaluationStatus, cls, float, get, int, parse_utc_datetime, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict…
aggregate_five_minute_medians **L314**	Aggregate valid scores into fixed UTC-aligned five-minute buckets.	(history: Iterable[ReconstructionHistoryRecord], config: SlowTrendConfig) → tuple[TrendBucket, …]	O(N log N) worst-case medians	TrendBucket, ValueError, _floor_timestamp, all, any, append, float, get…	application_service.py:CheckpointApplicationService._evaluate_slow_trend_if_scheduled, application_service.py:CheckpointApplicationService.reconstruction_history, slow_trend_evaluation.py:_missing_data_demo, slow_trend_evaluation.py:run_slow_trend_evaluation
evaluate_trend_at **L395**	Evaluate one overlapping OLS history at a UTC half-hour boundary.	(buckets: Iterable[TrendBucket], evaluated_at: datetime, config: SlowTrendConfig) → TrendEvaluation	O(B·F); B=expected buckets	TrendEvaluation, ValueError, _require_utc, any, astimezone, dict, float, int…	application_service.py:CheckpointApplicationService._evaluate_slow_trend_if_scheduled, slow_trend_evaluation.py:_missing_data_demo, trend_monitoring.py:evaluate_trend_schedule

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
evaluate_trend_schedule **L487**	Evaluate every aligned half hour once a nominal full history is available.	(buckets: Iterable[TrendBucket], config: SlowTrendConfig) → tuple[TrendEvaluation, ...]	O(N) typical	_ceil_timestamp, append, evaluate_trend_at, timedelta, tuple	slow_trend_evaluation.py:run_slow_trend_evaluation

trend_warnings.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
SlowTrendWarning.to_dict **L83**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(1) excluding delegated I/O/model work	—	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident...
SlowTrendWarning.from_dict **L137**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'SlowTrendWarning'	O(N) typical	OperatorClassification, TrendMonitoringState, TrendWarningStatus, cls, float, get, int, list...	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict...
TrendWarningStoreState.latest_valid_outcomes **L201**	Callable for latest valid outcomes in trend_warnings.py.	(self, count: int=6) → list[dict[str, object]]	O(N) typical	bool, format_utc	slow_trend_evaluation.py:_scenario_summary, trend_warnings.py:SlowTrendWarningDetector._open, trend_warnings.py:SlowTrendWarningDetector._opening_rule_met, trend_warnings.py:SlowTrendWarningDetector._update, trend_warnings.py:TrendWarningStoreState.to_dict
TrendWarningStoreState.to_dict **L217**	Serializes the domain object into a JSON-safe dictionary.	(self) → dict[str, object]	O(N log N) typical	latest_valid_outcomes, sorted, to_dict	application_service.py:CheckpointApplicationService._active_incident_dict, application_service.py:CheckpointApplicationService._active_warning_dict, application_service.py:CheckpointApplicationService._score_buffer_and_process_new_results, application_service.py:CheckpointApplicationService._score_dict, application_service.py:CheckpointApplicationService.acknowledge_incident...
TrendWarningStoreState.from_dict **L233**	Validates and reconstructs the domain object from persisted data.	(cls, raw: Mapping[str, Any]) → 'TrendWarningStoreState'	O(N·M) / data-dependent loops	ValueError, cls, from_dict, get, str	config.py:CheckpointGenerationConfig.from_dict, config.py:ProjectConfig.from_dict, config.py:SensorGenerationConfig.from_dict, config.py:load_config, incidents.py:IncidentStoreState.from_dict...
JsonTrendWarningStore.__init__ **L256**	Initializes the object and validates/loads required collaborators.	(self, path: str | Path, *, checkpoint_id: str) → None	O(1) excluding delegated I/O/model work	Path, TrendWarningStoreState, ValueError, _load, exists	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
JsonTrendWarningStore._load **L267**	Loads and validates persisted local state.	(self) → TrendWarningStoreState	O(1) excluding delegated I/O/model work	ValueError, from_dict, load, open	incidents.py:JsonIncidentStore.__init__, runtime_store.py:JsonRuntimeBufferStore.__init__, trend_warnings.py:JsonTrendWarningStore.__init__
JsonTrendWarningStore.save **L274**	Persists the artifact or state document.	(self) → None	O(1) excluding delegated I/O/model work	dump, mkdir, open, replace_with_retry, to_dict, with_name, write	application_service.py:CheckpointApplicationService.__init__, application_service.py:CheckpointApplicationService._activate_model_version, application_service.py:CheckpointApplicationService._initialize_runtime_configurations, application_service.py:CheckpointApplicationService.update_sudden_thresholds, application_service.py:CheckpointApplicationService.update_trend_configuration...
JsonTrendWarningStore.reset **L282**	Returns isolated mutable state to its initial value.	(self) → None	O(1) excluding delegated I/O/model work	TrendWarningStoreState, save	api_demo_routes.py:reset_demo, application_service.py:CheckpointApplicationService.reset_monitoring_runtime, demo_control.py:DemoController.run_preset, slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation...
JsonTrendWarningStore.active_warning **L287**	Callable for active warning in trend_warnings.py.	(self) → SlowTrendWarning | None	O(1) excluding delegated I/O/model work	—	api_read_routes.py:active_warning

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
SlowTrendWarningDetector.__init__ **L296**	Initializes the object and validates/loads required collaborators.	(self, config: SlowTrendConfig, store: JsonTrendWarningStore) → None	O(1) excluding delegated I/O/model work	ValueError	application_service.py:ApplicationServiceError.__init__, model.py:LSTMAutoencoder.__init__
SlowTrendWarningDetector.active_warning **L303**	Callable for active warning in trend_warnings.py.	(self) → SlowTrendWarning | None	O(1) excluding delegated I/O/model work	—	api_read_routes.py:active_warning
SlowTrendWarningDetector.process_evaluation **L306**	Callable for process evaluation in trend_warnings.py.	(self, evaluation: TrendEvaluation) → SlowTrendWarning | None	O(1) excluding delegated I/O/model work	ValueError, _open, _opening_rule_met, _update, append, save	application_service.py:CheckpointApplicationService._evaluate_slow_trend_if_scheduled, application_service.py:CheckpointApplicationService.process_trend_evaluation, slow_trend_evaluation.py:_controlled_lifecycle_demo, trend_warnings.py:SlowTrendWarningDetector.replay
SlowTrendWarningDetector.replay **L333**	Processes chronological evidence again to verify deterministic state.	(self, evaluations: Iterable[TrendEvaluation]) → TrendWarningStoreState	O(N) typical	process_evaluation	slow_trend_evaluation.py:_controlled_lifecycle_demo, slow_trend_evaluation.py:run_slow_trend_evaluation, sudden_runtime.py:_replay_to_fresh_store
SlowTrendWarningDetector.acknowledge_active **L338**	Callable for acknowledge active in trend_warnings.py.	(self, *, timestamp: datetime, operator: str) → SlowTrendWarning	O(1) excluding delegated I/O/model work	ValueError, _require_active, append, format_utc, save	application_service.py:CheckpointApplicationService.acknowledge_incident, application_service.py:CheckpointApplicationService.acknowledge_warning
SlowTrendWarningDetector.mark_active_under_review **L361**	Callable for mark active under review in trend_warnings.py.	(self, classification: OperatorClassification, *, note: str, timestamp: datetime, operator: str) → SlowTrendWarning	O(1) excluding delegated I/O/model work	ValueError, _require_active, append, format_utc, save	application_service.py:CheckpointApplicationService.classify_warning
SlowTrendWarningDetector.close_resolved **L393**	Callable for close resolved in trend_warnings.py.	(self, *, timestamp: datetime, operator: str, reason: str) → SlowTrendWarning	O(N) typical	ValueError, append, format_utc, max, save, values	entry point / dynamic / unreferenced statically
SlowTrendWarningDetector._require_active **L423**	Internal helper for require active in trend_warnings.py.	(self) → SlowTrendWarning	O(1) excluding delegated I/O/model work	ValueError	incidents.py:SuddenIncidentDetector.acknowledge_active, incidents.py:SuddenIncidentDetector.classify_active, trend_warnings.py:SlowTrendWarningDetector.acknowledge_active, trend_warnings.py:SlowTrendWarningDetector.mark_active_under_review
SlowTrendWarningDetector._opening_rule_met **L429**	Internal helper for opening rule met in trend_warnings.py.	(self) → bool	O(N) typical	bool, latest_valid_outcomes, len, sum	trend_warnings.py:SlowTrendWarningDetector.process_evaluation
SlowTrendWarningDetector._open **L437**	Internal helper for open in trend_warnings.py.	(self, evaluation: TrendEvaluation) → SlowTrendWarning	O(N) typical	SlowTrendWarning, ValueError, astimezone, float, format_utc, latest_valid_outcomes, strftime	incidents.py:SuddenIncidentDetector.process_score, trend_warnings.py:SlowTrendWarningDetector.process_evaluation
SlowTrendWarningDetector._update **L496**	Internal helper for update in trend_warnings.py.	(self, warning: SlowTrendWarning, evaluation: TrendEvaluation) → None	O(N) typical	append, float, format_utc, latest_valid_outcomes	incidents.py:SuddenIncidentDetector.process_score, trend_warnings.py:SlowTrendWarningDetector.process_evaluation

windowing.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
ModelWindow.shape **L26**	Callable for shape in windowing.py.	(self) → tuple[int, int]	O(1) excluding delegated I/O/model work	len	entry point / dynamic / unreferenced statically
create_sliding_windows **L44**	Callable for create sliding windows in windowing.py.	(records: Iterable[ProcessedInferenceRecord], *, scaler: RobustScaler, settings: PreprocessingConfig) → WindowBatch	O(N·W·F); W=60	ModelWindow, WindowBatch, WindowRejection, any, append, join, len, processed_features…	sudden_runtime.py:SuddenRuntimeScorer.score_records, synthetic_evaluation.py:run_synthetic_offline_evaluation, training.py:prepare_temporal_model_data, training.py:prepare_uploaded_model_data

windows_app.py

Callable	Purpose	Parameters → return	Complexity	Direct calls	Referenced by
resource_root L45	Resolve source checkout or PyInstaller's extracted resource directory.	(*, environ: Mapping[str, str] | None=None, frozen_root: str | Path | None=None) → Path	O(1) excluding delegated I/O/model work	Path, expanduser, get, getattr, hasattr, resolve	windows_app.py:resolve_paths
local_data_root L65	Return a writable location that remains separate from packaged files.	(*, environ: Mapping[str, str] | None=None) → Path	O(1) excluding delegated I/O/model work	Path, expanduser, get, home, resolve	windows_app.py:resolve_paths
resolve_paths L82	Callable for resolve paths in windows_app.py.	(*, configured_resource_root: str | Path | None=None, configured_data_root: str | Path | None=None) → WindowsAppPaths	O(1) excluding delegated I/O/model work	Path, WindowsAppPaths, local_data_root, resolve, resource_root	windows_app.py:main
required_resources L107	Callable for required resources in windows_app.py.	(root: Path) → tuple[Path, ...]	O(1) excluding delegated I/O/model work	—	windows_app.py:verify_resources
verify_resources L137	Callable for verify resources in windows_app.py.	(paths: WindowsAppPaths) → None	O(N) typical	FileNotFoundError, is_file, join, required_resources	windows_app.py:create_local_application
configure_logging L147	Callable for configure logging in windows_app.py.	(path: Path) → logging.Logger	O(N) typical	Formatter, RotatingFileHandler, addHandler, clear, getLogger, mkdir, setFormatter, setLevel	windows_app.py:main
available_port L171	Find a loopback port without exposing the application to the network.	(host: str, preferred: int, *, attempts: int=20) → int	O(N) typical	OSError, bind, range, socket	windows_app.py:main, windows_app.py:run_server_smoke_test
edge_candidates L186	Callable for edge candidates in windows_app.py.	(environ: Mapping[str, str] | None=None) → tuple[Path, ...]	O(N) typical	Path, append, get, tuple, which	windows_app.py:find_edge
find_edge L201	Callable for find edge in windows_app.py.	(environ: Mapping[str, str] | None=None) → Path | None	O(N) typical	edge_candidates, is_file, next	windows_app.py:launch_app_window
wait_until_healthy L205	Callable for wait until healthy in windows_app.py.	(url: str, timeout_seconds: float) → None	O(N) typical	TimeoutError, monotonic, sleep, urlopen	windows_app.py:_open_when_ready, windows_app.py:run_server_smoke_test
show_message L221	Give windowed builds a visible diagnostic without requiring a terminal.	(title: str, message: str, *, error: bool=False) → None	O(1) excluding delegated I/O/model work	MessageBoxW, print	windows_app.py:_open_when_ready, windows_app.py:launch_app_window, windows_app.py:main
launch_app_window L233	Open a standalone Edge window, falling back to the default browser.	(url: str, paths: WindowsAppPaths, logger: logging.Logger) → None	O(1) excluding delegated I/O/model work	Popen, find_edge, getattr, info, mkdir, open, show_message, str...	windows_app.py:_open_when_ready
build_parser L262	Callable for build parser in windows_app.py.	() → argparse.ArgumentParser	O(1) excluding delegated I/O/model work	ArgumentParser, add_argument	synthetic.py:main, windows_app.py:main
build_server_config L276	Create a server config that is safe in a console-free executable.	(app, host: str, port: int) → uvicorn.Config	O(1) excluding delegated I/O/model work	Config	windows_app.py:main, windows_app.py:run_server_smoke_test
run_server_smoke_test L289	Start the bundled HTTP server and verify its API and dashboard shell.	(app, host: str, preferred_port: int, logger: logging.Logger) → None	O(1) excluding delegated I/O/model work	RuntimeError, Server, Thread, TimeoutError, available_port, build_server_config, decode, info...	windows_app.py:main
_open_when_ready L320	Internal helper for open when ready in windows_app.py.	(server: uvicorn.Server, base_url: str, paths: WindowsAppPaths, logger: logging.Logger) → None	O(1) excluding delegated I/O/model work	exception, launch_app_window, show_message, wait_until_healthy	entry point / dynamic / unreferenced statically
create_local_application L340	Build the existing application service with isolated writable state.	(paths: WindowsAppPaths) → unspecified	O(1) excluding delegated I/O/model work	CheckpointApplicationService, create_app, for_project, mkdir, verify_resources	windows_app.py:main
main L355	Callable for main in windows_app.py.	(argv: Sequence[str] | None=None) → int	O(1) excluding delegated I/O/model work	Server, Thread, available_port, build_parser, build_server_config, configure_logging, create_local_application, exception...	entry point / dynamic / unreferenced statically

How to use the inventory For an interview, master public service/model/detector functions first. Private serialization and report helpers are included for completeness, but their shared pattern is validation → transformation → atomic persistence.

17. Interview preparation: 118 questions with answers

Practice aloud. Lead with the one-sentence answer, then add the mechanism and limitation. Do not memorize jargon; trace the real code path and state the exact comparison or data boundary when relevant.

Architecture

Q1. What does Fluventa detect?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Deviation from a checkpoint-specific learned healthy baseline; it does not output a physical fault class.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q2. Where does the ML architecture end?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At overall and per-feature reconstruction MSE. Incident and warning decisions are deterministic state machines.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q3. Why split API routes from application services?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Routes own HTTP concerns; the service owns orchestration and reuses one domain implementation across API, demo, and restart.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q4. Why one model per checkpoint?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Each location has a different healthy distribution; local models avoid mixing baselines and simplify scaler/threshold provenance.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q5. What is the deployed unit?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A verified bundle of weights, exact scaler, feature/window contract, sudden config, trend config, metadata, and reload fixture.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q6. Why have sudden and slow paths?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
One reacts to an immediate high score; the other requires sustained directional evidence and suppresses brief spikes.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q7. What role does the application service play?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It serializes ingestion, persists the buffer, invokes scoring/detectors, schedules trend evaluation, and returns coherent state.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q8. Why is the demo runtime isolated?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Synthetic scenarios must not contaminate normal operational state, configuration copies, incidents, warnings, or lifecycle records.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q9. How does restart recovery work?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Atomic stores reload; the saved buffer is rescored to backfill any result lost between buffer and score writes.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q10. Why does the frontend not calculate alert state?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
One authoritative backend prevents formula drift and keeps audit/restart behavior consistent.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q11. What is the source of truth for locked behavior?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The locked decisions/plan plus implemented code and versioned configuration; runtime values remain audited per checkpoint.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q12. Why is JSON acceptable here?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The scope is one local process/checkpoint; JSON is portable and inspectable, though not suitable for large concurrent history.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q13. What would you replace first at larger scale?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Growing JSON history and single-process checkpoint orchestration—not necessarily the small neural network.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q14. How are responsibilities made testable?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Services and controllers accept injectable collaborators/runtime roots, while domain functions are deterministic and file stores are isolated.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

Q15. What is the most important invariant?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Model, scaler, ordered features, window contract, thresholds, and trend config must be provenance-compatible.	Tests whether you understand boundaries, ownership, and invariants rather than naming technologies.	A buzzword-only answer or claiming every layer performs anomaly detection.	Which invariant would fail if two layers were combined?

ML

Q16. Why use a healthy-only autoencoder?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Reliable healthy data exists while complete labeled fault coverage does not; reconstruction provides open-set deviation evidence.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q17. Why an LSTM instead of a dense network?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The LSTM consumes ordered seconds and can learn temporal coupling; a dense model would flatten or ignore sequence structure.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q18. What is the exact input shape?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Batch × 60 × 4 in pressure, temperature, flow, vibration RMS order after robust scaling.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q19. What does the encoder output?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The final hidden state of shape batch × 32 summarizing the one-minute sequence.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q20. What does the bottleneck do?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A Linear 32→12 plus ReLU constrains the representation and prevents trivial high-capacity copying.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q21. Why repeat the latent vector 60 times?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The decoder needs one input per output step; repetition forces reconstruction from the compressed whole-window summary.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q22. Why is LSTM internal dropout zero?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
PyTorch applies it only between stacked recurrent layers; both LSTMs have one layer.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q23. Where is dropout applied?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Explicitly to decoder sequence outputs before the final linear layer, active only during training.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q24. How many trainable parameters?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
11,280: 4,864 encoder, 396 bottleneck, 5,888 decoder, and 132 output.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q25. What is per-feature MSE?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
For one window and feature, the average squared scaled reconstruction difference across 60 timesteps.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q26. How is overall MSE calculated?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The equal mean of the four per-feature MSE values.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q27. Why equal feature weighting?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It is simple and auditable after robust scaling; domain weights would require explicit safety justification.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q28. Why MSE rather than MAE?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
MSE matches the locked design and penalizes large scaled reconstruction misses more strongly; MAE would be more robust.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q29. What does a high MSE mean?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The sequence is hard for this healthy-trained model to reconstruct; it does not identify the cause or probability.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q30. Can the model detect an unseen fault?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It may flag its novelty, but detection depends on representation and thresholds; it cannot name an unseen fault.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q31. Why robust scaling?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Median/IQR reduce outlier influence and put four physical units on comparable scales.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q32. Why fit the scaler on training only?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Using validation or runtime data leaks future distribution information into model coordinates.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q33. What if a feature IQR is near zero?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Scaler fitting fails explicitly because dividing by an almost-zero IQR would amplify noise.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q34. Why keep pump_running out of the tensor?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It is external operating context in v1; including it would change the learned contract and could teach unjustified suppression.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q35. Would a 1D CNN be reasonable?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Yes; it may parallelize better and learn local motifs. It should replace the LSTM only after controlled comparison.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q36. Would a Transformer be better?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Not automatically. Sixty timesteps and limited data make its added capacity/complexity hard to justify.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q37. How could too-large a bottleneck hurt?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The model may copy unusual input too well, reducing anomaly-error contrast.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q38. How could too-small a bottleneck hurt?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It may underfit healthy variation and raise false alarms.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q39. What is the inference complexity?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Approximately O(B·T·H·(F+H)) for the recurrent blocks, with T=60 and small fixed dimensions here.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q40. Why CPU only?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The small model meets demo latency without GPU deployment complexity and supports deterministic local packaging.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Q41. What evidence verifies package reload?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A saved validation input, expected reconstruction/errors, and numeric tolerances are checked in a fresh load.	Tests whether you can explain the model mathematically and justify it under the project's data constraints.	Calling the model a supervised fault classifier or treating MSE as a diagnosis probability.	What experiment would justify changing this model choice?

Training

Q42. How is temporal leakage prevented?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Raw time is split first; each split is preprocessed/windowed separately; no window crosses the boundary.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q43. Why are 841 training windows not independent?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Stride-one 60-second windows share 59 samples with neighbors.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q44. Why shuffle training windows?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It improves minibatch mixing while the sequence inside each window remains chronological.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q45. Why not shuffle validation?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Validation measures the fixed temporal holdout and does not need gradient batches randomized.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q46. What optimizer/settings are used?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Adam, learning rate 0.001, batch 128, maximum 100 epochs, gradient clip 1.0, seed 42.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q47. Why gradient clipping?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Recurrent gradients can become large; global norm clipping bounds unstable updates.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q48. How does early stopping work?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Patience resets only for >1e-5 improvement; training stops after 10 non-meaningful epochs.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q49. Why save every strictly best state?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A tiny genuine improvement may not justify more training but should still be retained as the numeric best.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q50. What epoch was selected?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Epoch 5, the lowest healthy validation loss; training ended after 15 epochs.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q51. Does negative validation-minus-training MSE prove no overfitting?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
No. The later interval may be easier; the later divergence of train/validation trajectories is more informative.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q52. Why require explicit healthy confirmation?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Contaminated anomalies in the training baseline can teach the autoencoder to reconstruct faults as normal.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q53. Why manual retraining?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
An engineer must validate healthy provenance and review candidate evidence; automatic adaptation could normalize degradation.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q54. What happens while a candidate trains?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The active scorer continues unchanged; training uses a serialized background worker.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q55. Why approval separate from deployment?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Review and operational activation are distinct governance decisions and must be auditable.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Q56. What is not done for each candidate?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The full synthetic anomaly suite is not rerun in this scoped phase, so lower healthy MSE is not operational superiority.	Tests leakage control, reproducibility, model selection, and interpretation of evidence.	Ignoring temporal leakage/overlap or presenting validation reconstruction as anomaly accuracy.	How would you detect leakage or drift in a larger dataset?

Data quality

Q57. What happens before 60 readings?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The runtime buffers and reports no score; it does not pad or shorten the window.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q58. How are duplicate timestamps handled?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
They are rejected/quarantined to preserve a unique one-Hz checkpoint timeline.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q59. Are timestamp gaps rejected?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
No; the later reading is accepted and missing seconds are expanded as explicit quality evidence.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q60. How are 1–3 second gaps handled?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Processed values are linearly interpolated only with valid endpoints; raw records remain null and flags say imputed.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q61. How are gaps longer than 3 seconds handled?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Values remain null and every containing window is rejected as insufficient data.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q62. Why not replace missing errors with zero?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Zero means perfect reconstruction; missing means no evidence. Conflating them would create false recovery/trends.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q63. What makes an isolated glitch?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A one-sample absolute/relative jump that returns near baseline and lacks support from a second feature.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q64. Why preserve multi-feature transients?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Correlated sensor motion may reflect a real process event and should reach the model.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q65. How are frozen values detected?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Exact repeated valid values for 30 contiguous seconds trigger a frozen flag and null processed value.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q66. What happens to a physical-range violation?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The finite raw value is preserved; preprocessing marks it invalid and suppresses affected windows.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q67. When does availability become unavailable?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At the 30th consecutive second without a fully valid raw row.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q68. When is maintenance escalation shown?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At the 300th consecutive second without a fully valid raw row.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q69. Does imputation count as fully valid raw input?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
No. It may keep a window scoreable, but availability's fully-valid-raw counter still reflects the quality event.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q70. Why preserve raw and processed streams?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Investigators need original evidence, while the model needs explicit controlled transformations.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Q71. What metadata stays outside the tensor?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Timestamp, checkpoint, split, source sequence, pump state, quality counts, and evaluation labels.	Tests whether your ML system behaves safely when real sensor data is incomplete or misleading.	Assuming missing/invalid data becomes zero or that processed replacements overwrite raw evidence.	How does this condition affect both sudden recovery and slow-trend coverage?

Sudden

Q72. What opens a sudden incident?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The first scoreable overall MSE strictly greater than spike when no incident is active.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q73. What if MSE equals spike?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It does not open because the comparison is strict greater-than.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q74. Why no opening persistence?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The sudden path prioritizes immediate visibility; stability is handled on recovery.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q75. How are duplicates prevented?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
An active incident ID occupies the checkpoint slot; later scores update it instead of opening new records.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q76. What increments recovery?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
A scoreable overall MSE strictly below the recovery threshold.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q77. What if MSE equals recovery?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The counter resets because recovery uses strict less-than.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q78. What does an unscoreable second do to recovery?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It leaves the counter unchanged and may mark monitoring unavailable.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q79. Why use a lower recovery threshold?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Hysteresis prevents a score oscillating near the opening threshold from flapping incident state.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q80. What does classification change?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Operator interpretation and audit fields only; not score history, thresholds, or recovery.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Q81. How many notifications are emitted?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
One opened and one resolved event per incident; no per-second duplicates or invented escalation.	Tests precise state-machine reasoning and boundary conditions.	Using ≥/≤ incorrectly, counting unavailable seconds as recovery, or opening duplicate incidents.	What happens at exact equality and during an unscoreable second?

Trend

Q82. How is a five-minute bucket aligned?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Fixed UTC half-open interval [start, start+5 minutes), with 300 expected one-second scores.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q83. Why use a median?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It resists a brief high score dominating slow-degradation evidence.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q84. When is a bucket valid?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Coverage is at least 80%; 240/300 is valid and 239/300 is not.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q85. How is the slope calculated?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Ordinary least squares over valid bucket medians, with x as elapsed hours from the first valid bucket.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q86. What are slope units?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Scaled reconstruction-error units per hour.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q87. When is a trend evaluation scheduled?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At exact UTC half-hour boundaries over the preceding configurable six-hour window.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q88. How many valid buckets are needed at 80% of 72?

Ideal answer	Why asked	Common wrong answer	Likely follow-up

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At least 58; 57/72 is only 79.17%.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q89. What opens a warning?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
At least five of the latest six valid gradients are strictly above threshold.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q90. Does insufficient data enter the latest six?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
No; it cannot open, recover, or resolve and marks an active warning monitoring_limited.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q91. How does warning recovery work?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Three consecutive valid scheduled gradients at or below recovery mark it resolved.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q92. Why is resolved not automatically closed?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Recovery and reviewed archival are separate for slow operational records.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

Q93. What did long-horizon scenarios prove?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
They exercised aggregation and slope response but each had only four threshold exceedances, so no operational warning opened.	Tests aggregation, statistics, coverage, persistence, and latency reasoning.	Filling missing gradients with zero or forgetting coverage and 5-of-6 persistence.	What is the earliest possible valid/opening time under the default schedule?

API/UI

Q94. Why strict Pydantic models?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
They reject extra fields, wrong numeric types, non-UTC timestamps, and malformed actions before domain logic.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q95. Why allow finite out-of-range readings through the API?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The raw reading is evidence; the authoritative preprocessor decides physical validity.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q96. What status codes distinguish errors?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
422 validation, 404 missing resource, and 409 invalid lifecycle transition.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q97. How does frontend polling differ by data?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Fast state every five seconds, slow histories every 30 seconds, demo every second, stable metadata once.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q98. Why Promise.all?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Independent reads complete concurrently and reduce dashboard refresh latency.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q99. Why retain stale data after refresh failure?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Operators keep the last known evidence while seeing an explicit connection error.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q100. Why load classifications from OpenAPI?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Visible choices stay aligned with the backend enum while TypeScript keeps compile-time checking.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q101. Why native SVG charts?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The required simple line/threshold views stay lightweight and accessible without another chart dependency.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q102. Why loopback-only in the Windows app?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The portfolio API has no production identity system, so local binding reduces accidental network exposure.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Q103. Does loopback equal secure production deployment?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
No; remote use still needs TLS, authentication, authorization, rate limits, audit export, and hardening.	Tests separation of transport/presentation from domain decisions.	Saying the route or browser recomputes ML/lifecycle decisions.	Where is the authoritative validation or state transition implemented?

Evaluation

Q104. Why hash artifacts before and after evaluation?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
It proves the offline evaluation did not retrain, refit, or mutate protected inputs.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Q105. What does 0-second synthetic visibility mean?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The first post-onset score crossed the configured comparison/threshold in that synthetic timeline; not guaranteed field latency.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Q106. Why report one healthy false incident?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Honest calibration evidence shows p99.5 provisional threshold is not production-ready.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Q107. Why are per-feature errors indicators only?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Reconstruction coupling can move errors in unaffected features; they are not causal probabilities.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Q108. What metrics are missing for production?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Field-labeled precision/recall, false alarms per time, detection delay distributions, drift and operating-mode analysis.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Q109. How would you choose a production threshold?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Use independent representative healthy and labeled event periods, cost-sensitive validation, stability checks, and domain approval.	Tests scientific honesty and whether you distinguish prototype evidence from production claims.	Presenting configured synthetic results as field reliability or causal diagnosis.	What additional field evidence is required before deployment?

Security/scale

Q110. What production auth model would you add?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Authenticate devices/operators and enforce separate read, ingestion, and administrative roles at gateway and service layers.	Tests whether you can evolve the prototype without hand-waving.	Claiming loopback/separate routes equal production security or proposing microservices without a bottleneck.	What metric or threat would trigger that change?

Q111. How would you prevent replayed sensor requests?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Device identity, signed requests or mTLS, nonce/sequence/timestamp checks, and ingestion audit.	Tests whether you can evolve the prototype without hand-waving.	Claiming loopback/separate routes equal production security or proposing microservices without a bottleneck.	What metric or threat would trigger that change?

Q112. How would you scale history storage?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Move per-second/bucket/evaluation records to partitioned time-series or relational storage with retention policies.	Tests whether you can evolve the prototype without hand-waving.	Claiming loopback/separate routes equal production security or proposing microservices without a bottleneck.	What metric or threat would trigger that change?

Q113. How would you scale inference?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Partition by checkpoint, keep each model/scaler cached with provenance, batch compatible windows, and isolate worker failures.	Tests whether you can evolve the prototype without hand-waving.	Claiming loopback/separate routes equal production security or proposing microservices without a bottleneck.	What metric or threat would trigger that change?

Behavioral

Q114. What design choice would you revisit first?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Threshold calibration and operating-state handling once real data exists; they dominate false-alert usefulness more than adding layers.	Tests ownership, judgment, communication, and willingness to discuss limitations.	Pretending every choice is perfect or hiding negative evidence.	What did you learn and what would you change next?

Q115. Describe an honest limitation you kept.

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The project records a healthy false incident and slow scenarios that fail 5-of-6 rather than tuning on evaluation data.	Tests ownership, judgment, communication, and willingness to discuss limitations.	Pretending every choice is perfect or hiding negative evidence.	What did you learn and what would you change next?

Q116. What are you most proud of?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Clear separation of immutable evidence, ML scoring, deterministic lifecycle, audit, and manual model governance.	Tests ownership, judgment, communication, and willingness to discuss limitations.	Pretending every choice is perfect or hiding negative evidence.	What did you learn and what would you change next?

Q117. What would enterprise readiness require?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
Field validation, identity/RBAC, durable scalable storage, observability, model registry/signing, HA, and incident-response procedures.	Tests ownership, judgment, communication, and willingness to discuss limitations.	Pretending every choice is perfect or hiding negative evidence.	What did you learn and what would you change next?

Q118. How would you explain this to an operator?

Ideal answer	Why asked	Common wrong answer	Likely follow-up
The system compares the latest minute with learned healthy behavior, shows which signals contributed, and asks for investigation—not automatic diagnosis.	Tests ownership, judgment, communication, and willingness to discuss limitations.	Pretending every choice is perfect or hiding negative evidence.	What did you learn and what would you change next?

18. Deep-why summary and final mentor notes

Why was each core component created?

Component	Deep why
SensorPreprocessor	A neural model must never be the first component to discover malformed, missing, frozen, or implausible sensor input.
RobustScaler	The model needs stable checkpoint-specific coordinates and equal opportunity for differently scaled features to matter.
LSTM autoencoder	Learn temporal healthy structure without pretending the project has labeled coverage of all real failures.
Per-feature MSE	Give engineers interpretable contribution evidence while retaining one overall decision score.
Sudden detector	Convert one-second evidence into a deduplicated, recoverable operational lifecycle.
Slow detector	Separate sustained drift from transient spikes and refuse to guess when coverage is weak.

Component	Deep why
Operator classification	Attach real-world meaning without rewriting or deleting machine evidence.
Model lifecycle	Prevent training, approval, deployment, and rollback from becoming one unsafe automatic action.
Application service	Guarantee all entry points execute the same ordered domain behavior.
Dashboard	Make evidence and governance understandable without moving authority into presentation code.

The answer you should give when asked ‘Why this way?’

Suggested answer We had reliable healthy sequences, limited trustworthy fault labels, and a need for explainable checkpoint-specific monitoring on a local CPU. So we used a small healthy-only LSTM autoencoder to generate reconstruction evidence, kept data-quality and alert policy deterministic, and required human interpretation and manual model governance. The design is intentionally conservative: raw evidence is immutable, missing scores never mean healthy, state is deduplicated and restart-safe, and synthetic results are not presented as field accuracy. The tradeoff is that benign operating changes can alert and the current thresholds need real-world calibration.

Five sentences to remember

1. The model reconstructs healthy 60 × 4 sequences; it does not classify faults.
2. The exact checkpoint scaler is part of the model, because it defines the model's coordinate system.
3. Sudden incidents open on one strict crossing but close only after 30 valid strict-below scores.
4. Slow warnings need five-minute medians, six-hour coverage, OLS slope, and 5-of-6 persistence.
5. Every synthetic result is software evidence, not field reliability or physical diagnosis.

Final interview mindset A senior explanation does not make the project sound flawless. It makes the constraints, mechanisms, evidence, tradeoffs, and next decisions unmistakably clear.

Portfolio demonstration

Fluventa ML

Demo Testing & Verification Guide

Pipeline anomaly-detection prototype · checkpoint_01

Version 1.0
Prepared 7 August 2026

Manual demo cases, expected evidence, automated checks, and troubleshooting

Document purpose

This guide explains how to start, exercise, and verify the Fluventa portfolio application. It covers normal telemetry, reconstruction behavior, sudden process events, gradual trends, sensor-quality faults, incident lifecycle, retraining controls, and failure recovery.

> **Scope:** the system uses synthetic data and a small pipeline-verification model. Passing these cases validates prototype software behavior only. It does not establish anomaly-detection accuracy, safety integrity, or performance on a real petroleum pipeline.

Recommended test order

1. Start the backend and frontend.
2. Run the automated test suites.
3. Reset the demo.
4. Run Guided Presets A through F in order.
5. Run the individual process and sensor-quality cases as needed.
6. Record observations in the execution sheet at the end.

Contents

1. System under test

Item	Expected contract
Checkpoint	`checkpoint_01`; one demo model. Production scaling repeats the same package per checkpoint.
Sampling	One reading per second (1 Hz).
Model input/output	`60 × 4` windows, stride 1 second.
Fixed feature order	`pressure_bar`, `temperature_c`, `flow_rate_l_min`, `vibration_velocity_rms_mm_s`.
External context	`pump_running` is retained as metadata and never enters the tensor.
Time	`timestamp` is the only time column.
Scaling	Checkpoint-specific robust scaler fitted on healthy training data and reused for scoring.
Sudden-event threshold	Overall reconstruction MSE `0.15275620818138122`.
Recovery threshold	MSE `0.12491045147180557` for 30 consecutive valid scored seconds.
Trend configuration	5-minute medians; 6-hour horizon; evaluate every 30 minutes; warning threshold `10.828543538404618`; 5 of 6 warning votes and 3 recovery votes.
Availability	Unavailable after 30 seconds without a valid reading; maintenance escalation after 300 seconds.

Thresholds are prototype values already contained in the current runtime configuration. This guide verifies that the configured logic is used; it does not claim those thresholds are suitable for field deployment.

2. Starting the application

Open two PowerShell windows. Run commands from the repository root, not from `D:\Project flow`.

```
cd "D:\Project flow\pipeline_anomaly_detection_plan_refined"
python examples\run_backend.py
```

Expected backend address: `http://127.0.0.1:8000`.

```
cd "D:\Project flow\pipeline_anomaly_detection_plan_refined\frontend"
npm run dev
```

Open the URL printed by Vite, normally `http://localhost:5173`.

> **Important:** an `ENOENT ... package.json` error means `npm run dev` was launched from the wrong folder. Change to the `frontend` directory first.

Startup smoke check

1. The backend terminal stays running without a traceback.
2. The frontend displays navigation and page content, not a blank dark screen.
3. Browser developer tools show no uncaught JavaScript error.
4. Click **Reset Demo**, then run **Preset A — Healthy baseline**.
5. Recent readings appear and begin updating.

3. Standard verification workflow

Use this sequence for every manual case so results from a prior case cannot contaminate the next one.

1. Open **Demo Control**.
2. Click **Stop** if a stream is active.
3. Click **Reset Demo** and confirm counters return to their initial state.
4. Select the intended preset or scenario.
5. Record the displayed seed and parameters.
6. Start the case. Do not refresh while the action is starting.
7. Observe Demo Control, then verify the corresponding Overview, Reconstruction, Slow Trend, and Incidents pages.
8. Stop any individual scenario that continues healthy streaming after its event.
9. Save screenshots or export results where required.

Common evidence to capture

- Scenario name, seed, start/end time, and parameters.
- Reading, scored-window, rejected-window, incident, and trend-state counts.
- Overall and per-feature reconstruction-error graphs.
- Incident opened/closed timestamps and classification context.
- Data-quality flags and rejection reason where applicable.

4. Guided demo presets

PRE-A — Healthy baseline

Purpose: prove that telemetry reaches the UI and produces correctly shaped model windows without intentionally injected faults.

1. Reset the demo.
2. Run **Preset A — Healthy baseline**.
3. Watch the reading counter until at least 90 readings are generated.
4. Open Overview and Reconstruction.

Expected: the first score becomes possible when reading 60 completes the first 60-second window. Subsequent readings add one window each because stride is one. Sensor charts are stable around their healthy baselines and use sensible engineering ranges. No synthetic event label is present.

Pass if: 90 readings are present, scored windows exist, shapes are 60 × 4 internally, all four feature charts update, and no new event-driven incident is created.

PRE-B — Sudden blockage and recovery

Purpose: exercise a pressure increase with flow reduction, reconstruction rise, incident opening, recovery, and incident closure.

Reference parameters: seed 42; event starts after 120 seconds; active for 90 seconds; recovery transition 15 seconds; pressure +1.2 bar; flow −22 L/min.

1. Reset, then run Preset B.
2. On Overview, verify pressure rises while flow falls.
3. On Reconstruction, observe overall error and pressure/flow contributions.
4. On Incidents, wait for an incident to open, then for recovery to close it.

Expected reference run: incident ID similar to `checkpoint_01-20260102T000200Z-sudden`; peak overall MSE approximately 8.47; incident opens near simulated 00:02:00 and closes near 00:05:06 after the recovery rule is satisfied.

Pass if: the event is visible after onset, pressure and flow dominate the error evidence, exactly one sudden incident lifecycle is shown, and closure occurs only after 30 consecutive scores below the recovery threshold.

PRE-C — Bottle jam / pump context

Purpose: prove that `pump_running` influences incident interpretation but never enters the 60 × 4 model tensor.

1. Reset and run Preset C.
2. Observe `pump_running=false` in context metadata.
3. Inspect the incident detail/classification.
4. Confirm scoring still exposes only the four sensor features.

Expected: pump context can classify an otherwise similar deviation as operational/bottle-jam context. It does not by itself close an incident; closure still requires valid recovery evidence.

Pass if: pump state is visible in metadata/interpretation and absent from model feature tensors and per-feature reconstruction output.

PRE-D — Sensor-quality suite

Purpose: run isolated spike, frozen value, short/long gaps, drift, and invalid input as data-quality cases rather than process anomaly labels.

1. Reset and run Preset D.
2. Inspect the scenario results and validation/rejection counters.
3. Verify the individual expectations in Section 6.

Pass if: raw evidence is retained; repair occurs only in the model-input stream; short gaps are imputed; long gaps are rejected; and faults remain distinguishable from process events.

PRE-E — Long-horizon gradual trend

Purpose: exercise the separate slow-trend pathway over a compressed 9-hour (32,400-second) synthetic run.

1. Reset and run Preset E.
2. Open Slow Trend and observe 5-minute median aggregation and 30-minute evaluations.
3. Verify warning votes accumulate independently from sudden incidents.
4. Observe the controlled recovery lifecycle.

Expected: gradual flow/pressure behavior produces a rising trend score and a warning only after configured voting. Recovery requires three recovery votes. A trend warning is not a sudden-anomaly incident.

Pass if: the warning and recovery lifecycle completes and no sudden-alert state machine is substituted for the trend logic.

PRE-F — Retraining workflow simulation

Purpose: verify the isolated portfolio workflow for upload, validation, a short two-epoch candidate run, comparison, approval/deployment, and rollback.

1. Reset and run Preset F.
2. Review validation evidence before training begins.
3. Verify the candidate remains separate from the active model.
4. Exercise approval/deployment, then rollback.

Expected: explicit status transitions are recorded and the active version changes only at deployment. Rollback restores the previous candidate. This is a demo workflow, not an automated production approval system.

Pass if: each stage is auditable, validation gates are respected, and rollback restores the earlier active package.

5. Individual process-anomaly scenarios

Individual process scenarios normally continue with healthy readings after the injected event. Click **Stop** after sufficient recovery evidence has been observed.

ID / scenario	Injection and expected telemetry	Verification
PA-01 Blockage-like	Sudden pressure increase (+1.2 bar) and flow reduction (−22 L/min).	Overall MSE rises after onset; pressure and flow are the leading contributors; a sudden incident may open; after recovery it closes only after the configured 30-score rule.
PA-02 Rupture / major-leak-like	Sudden pressure reduction (−1.4 bar) and flow reduction (−28 L/min).	Both pressure and flow move downward. Reconstruction rises after onset. Feature attribution must not claim vibration is the injected cause.
PA-03 Developing restriction / leak	Gradual flow reduction (up to −24 L/min) with slowly changing pressure (up to +0.7 bar); approximately 180 seconds active and 30 seconds recovery.	Error should generally rise as the deviation grows. Judge the pattern, not strict monotonicity at every second. Verify the gradual-event metadata.
PA-04 Mechanical degradation	Gradual vibration RMS increase (up to +1.2 mm/s); approximately 180 seconds active and 30 seconds recovery.	Vibration feature MSE should become the primary injected contribution and overall error should form a rising pattern.
PA-05 Long-horizon flow/pressure	Extended gradual process deviation for trend evaluation.	Verify 5-minute aggregation and trend voting; do not interpret a single raw score as a slow-trend warning.
PA-06 Long-horizon vibration	Extended gradual vibration increase.	Trend evidence should be vibration-led and remain separate from sudden incident logic.

> **Evidence rule:** labels and event metadata accompany exported evaluation/history records but are not model features. The tensor must contain only scaled sensor values in the fixed four-feature order.

How to judge “visible quickly” for sudden events

1. Record the event start timestamp.
2. Find the first post-onset window whose overall MSE clearly departs from the immediately preceding healthy range.
3. Calculate *visibility delay = first visible window end − event start*.
4. Remember that each 60-second window progressively contains more anomalous seconds after onset. Do not mistake window overlap for independent confirmation.

How to judge a gradual pattern

Compare pre-event, early-event, and late-event median errors. A pass requires the late-event median to be higher and the relevant feature to explain the injection. Small reversals caused by temporal noise are acceptable; strict point-by-point monotonicity is not required.

6. Sensor-quality scenarios

DQ-01 — Isolated one-sample spike

Action: inject a single pressure sample approximately +3 bar while neighboring samples return to the prior range.

Expected: original raw value retained; `suspected_sensor_glitch=true`; only the model-input copy is interpolated. It is treated as sensor quality, not proof of a process incident.

DQ-02 — Persistent real deviation

Action: inject a multi-reading process change rather than a one-sample spike.

Expected: it is not removed by the isolated-glitch rule. Multiple persistent readings remain available to reconstruction scoring.

DQ-03 — Frozen sensor

Action: hold one sensor at exactly the same value for about 45 seconds.

Expected: after the configured 30-second frozen interval, affected processed values become unavailable/flagged while original raw values remain. Windows that require invalid values are unscoreable.

DQ-04 — Short missing gap

Action: omit two consecutive seconds.

Expected: the raw dataset remains unchanged and has no fabricated raw rows. The model-input stream expands to 1 Hz and linearly interpolates the gap. Imputation flags are present; for a two-row, four-feature gap, at least eight feature values are imputed.

DQ-05 — Long missing gap

Action: omit five consecutive seconds.

Expected: no 60-second window containing that gap is scored. The rejection reason is `missing_gap_exceeds_interpolation_limit` (or the current documented equivalent). Later windows become scoreable only after the gap falls outside the complete 60-second window.

DQ-06 — Gradual sensor drift

Action: inject a slowly accumulating sensor drift.

Expected: the persistent drift is not classified as an isolated spike and is not interpolated away. It remains available for evaluation and stays identified as a sensor-fault scenario in metadata.

DQ-07 — Invalid numeric input

Action: submit a non-numeric or non-finite sensor value.

Expected: the API/schema rejects it with a clear validation response; the backend continues running.

DQ-08 — Physically impossible value

Action: submit pressure −1 bar through the accepted numeric interface.

Expected: preprocessing retains the raw value, flags it outside the physical range, and excludes it from the valid model-input value. It must not silently replace it with zero.

DQ-09 — Duplicate timestamp

Action: submit a second reading for an existing timestamp/checkpoint.

Expected: deterministic validation rejection/quarantine; no duplicate scored window is produced.

DQ-10 — Ordering/checkpoint validation

Action: submit an out-of-order reading or an unsupported checkpoint ID.

Expected: rejection with a clear reason. Existing telemetry and incidents remain intact.

7. Availability and maintenance states

ID	Procedure	Expected result
AV-01 Monitoring unavailable	After a valid reading, advance simulated time for 31 seconds without another valid reading.	Checkpoint monitoring changes to **unavailable** after 30 seconds. This is a data-quality/availability state, not an anomaly alert.
AV-02 Maintenance escalation	Continue the no-valid-reading interval to 301 seconds.	A maintenance-escalation state is produced after 300 seconds. It remains distinct from sudden and trend incidents.

Recovery check: submit a valid in-order reading after the outage. Verify availability recovers according to runtime logic without rewriting the historical gap.

8. Page-by-page UI verification

ID / page	What to verify	Pass condition
UI-01 Navigation	Open every nav link.	Each link has its own route/page; the app does not place all sections on one long page.
UI-02 Overview	Run Preset A and watch Recent sensor readings.	Telemetry refreshes roughly once per second. Pressure, temperature, flow, and vibration have readable labels, units, timestamps, and non-misleading engineering scales.
UI-03 Chart safety	Refresh during and after a scenario; inspect console.	No `Cannot read properties of undefined` error; incomplete records are handled without blanking the app.
UI-04 Reconstruction	Run Presets A and B.	Overall and per-feature MSE update, window-ending timestamps align, and the graph retains completed-run evidence until Reset.
UI-05 Slow Trend	Run Preset E.	Trend statistics, votes, warning status, and recovery update. They are not labeled as sudden incidents.
UI-06 Incidents	Run Preset B and navigate away/back.	Opened/closed incidents update, survive navigation, and remain visible after the preset completes until Reset.
UI-07 Demo Control	Start, stop, continue, and reset scenarios.	Only one active stream is represented; statuses and counters agree with backend evidence.
UI-08 Responsive/empty/error	Test narrow window, initial empty state, and backend temporarily stopped.	No overlap or hidden controls; helpful empty/error state appears; normal content returns after backend recovery.

> Overview, Reconstruction, Slow Trend, and Incidents consume the same demo evidence and poll it periodically. A page can briefly lag by one polling interval, but it must converge without a manual refresh.

9. Control and persistence checks

ID	Test	Expected
CTL-01	Start a preset twice rapidly.	No duplicate uncontrolled run; the API returns a deterministic accepted/conflict state.
CTL-02	Stop an individual scenario.	New synthetic readings cease and current evidence remains visible.
CTL-03	Navigate between pages during a run.	The backend run continues; UI catches up on return.
CTL-04	Refresh the browser.	Current/completed evidence reloads from backend state; no blank-screen crash.
CTL-05	Restart backend with ordinary persisted state.	Startup reconciles the bounded buffer without a MemoryError.
CTL-06	Reset Demo.	Demo readings, scores, trend evidence, and incidents clear as documented; packaged model/scaler artifacts remain.
CTL-07	Compare event export with model inputs.	Event labels/metadata are in evaluation/history output only and never appear among tensor features.

10. Automated verification

Stop neither server unless a port conflict occurs; tests can be run in a third PowerShell window from the repository root.

Python suite

```
cd "D:\Project flow\pipeline_anomaly_detection_plan_refined"
python -m pytest
```

Current expected baseline: 130 tests pass.

Frontend unit suite

```
cd "D:\Project flow\pipeline_anomaly_detection_plan_refined\frontend"
npm test -- --run
```

Current expected baseline: 22 tests pass.

Production frontend build

```
cd "D:\Project flow\pipeline_anomaly_detection_plan_refined\frontend"
npm run build
```

Expected: TypeScript compilation and Vite production build complete with exit code 0.

> Counts are a recorded baseline, not a rule that forbids additional tests. A future run may legitimately contain more tests, but it must contain no failures.

Automated/manual coverage map

Area	Automated evidence	Manual evidence
Data/schema/preprocessing	Python unit tests	DQ cases and raw/processed comparison
Model reconstruction/runtime	Python model/runtime tests	PRE-A/B and Reconstruction page
Incidents and trends	State-machine/service tests	PRE-B/E and Incidents/Slow Trend
Frontend rendering	Frontend unit tests + build	UI-01 through UI-08
End-to-end behavior	API/demo-control tests	Guided presets A–F

11. Troubleshooting

Symptom	Likely cause	Action
`npm ENOENT ... D:\Projectflow\package.json`	Wrong working directory.	Run npm commands from `...\pipeline_anomaly_detection_plan_refined\frontend`.
Dark/blank frontend; content flashes then disappears	Runtime JavaScript exception or incompatible API record.	Open browser Console, record the first error, hard refresh once, verify both servers use current code, then run frontend tests.
`Cannot read ... pressure_bar`	Chart received an incomplete/undefined reading.	Use current frontend build, reset demo state, reload. If it repeats, capture the API payload and console stack.
Recent sensor readings empty	No active/completed telemetry, backend unavailable, or polling error.	Run Preset A, confirm backend terminal and browser Network responses, then wait one polling interval.
Reconstruction/Trend/Incidents do not update	Preset not far enough through simulated data, stale frontend, or API request failed.	Check Demo Control counters, Network responses, and page after one polling interval. Completed evidence should remain until Reset.
Backend `MemoryError` while expanding timestamps	Persisted readings contain an extreme timestamp separation.	Stop the backend, preserve the traceback, use the documented demo reset/clean-state procedure, and restart current code. Do not manually edit packaged model artifacts.
Port already in use	An older backend/frontend is still running.	Close the older terminal/process or use the configured alternate port and matching frontend API URL.

Failure report checklist

- Test case ID and exact time.
- Git/version identifier if available.
- Preset/scenario, seed, and parameters.
- Backend traceback and first browser-console error.
- Relevant API response/body.
- Screenshot before reset.
- Whether the failure reproduces from a clean Reset Demo state.

12. Test execution record

Field	Entry
Tester	
Date/time	
Version/commit	
Browser and OS	
Python / Node versions	

Case group	Pass / Fail	Evidence or issue reference
Startup smoke check		
PRE-A Healthy		
PRE-B Blockage/recovery		
PRE-C Pump context		
PRE-D Sensor quality		
PRE-E Slow trend		
PRE-F Retraining workflow		
PA-01 to PA-06		
DQ-01 to DQ-10		
AV-01 to AV-02		
UI-01 to UI-08		
CTL-01 to CTL-07		
Python tests		
Frontend tests/build		

Final result

Overall: ☐ Pass ☐ Conditional pass ☐ Fail

Open issues / limitations:

Tester signature: _____ Date: _____

Interpretation reminder

The 15-minute training and 5-minute validation slices used for the small baseline are pipeline-verification data. Their overlapping sliding windows are highly dependent observations. Validation can be easier than training, and explicit

training-mode regularization can differ from evaluation behavior. The relevant early-stopping evidence is that later validation loss deteriorated while training loss continued falling, after which the best validation epoch was restored.

> **Do not present this demo as field validation.** Synthetic scenario visibility and reconstruction behavior demonstrate software pathways and prototype plausibility only. Real deployment requires representative operational data, checkpoint-specific packages, independent evaluation, safety review, monitoring, governance, and controlled approval.