Explanation of the new point-by-point files (IMPORTANT)

Hi,

This update introduces a point-by-point evaluation model.

The goal is to make the output explicit, actionable, and fair, regardless of report type or structure.

Different report types have different point structures.

Some points (e.g. 7.2.1, 7.3.1) only exist if the building actually has multiple bathrooms.

Therefore we cannot rely on a fixed list of points.

Instead, the system must first detect the points that actually exist in the uploaded report, and then generate output only for those points.

Core idea (very important)

The system must work in two distinct steps:

STEP 1 — Detect report points (document-specific)

For every uploaded PDF, the system must first build a list of all points that actually exist in that document.

This is done via:

detected_points.json

- One entry per detected report point
- Generated dynamically from the PDF
- Independent of report type
- Frozen/cached before scoring begins

Each detected point includes:

- point_id (e.g. 4.2, 7.2.1, 10.4)
- title
- page range
- optional TG
- a stable span_hash

This file becomes the ground truth for everything that follows.

If a point does not exist in detected_points.json, it must NOT appear anywhere in the output.

STEP 2 — Evaluate and generate point-by-point output

After detected_points.json is created and frozen:

1. Run all rules (ARKAT, language, TG consistency, etc.)
2. All findings MUST reference a point_id
3. Generate the final result output

Required behavior:

The final output must list EVERY detected point, even if:

- the point is perfectly OK
- there is no deduction
- there is no suggestion

This is done via:

points_overview[] (in feedback v1.1)

Each point must have:

- point_id
- title
- TG
- status:
  - ok
  - improve

    - deduction
    - blocking

a short, user-friendly summary

Example logic:

If no findings → status: ok

If only minor guidance → status: improve

If deductions → status: deduction

If 96% gate triggered → status: blocking

This ensures the user can see:

- exactly which points are good
- exactly which points need changes
- exactly where to edit in the report

Why this is required

Without this structure, the system produces generic feedback, such as:

> "Several ARKAT sections contain long sentences…"

This is technically correct, but useless to the user, because they don't know:

- where the problem is
- which point to edit
- what is already good

The point-by-point model solves this.

Important rules you must enforce in backend

Please ensure the following are hard rules, not suggestions:

1. detected_points.json is generated FIRST and cached by document_hash
2. All findings MUST include point_id (no exceptions)
3. points_overview[] MUST include all detected points
4. Aggregated messages (e.g. "several points…") may exist only as secondary summaries, never instead of point-level output

What you are implementing now

You are not changing scoring logic.

You are not adding new rules.

You are adding:

- **a document-specific point detection layer**
- **a mandatory point-by-point presentation layer**

The JSON files you received define:

- the schema for detected points
- the schema for final output
- examples of correct payloads

Once this is implemented, the system will:

- work with any report structure
- handle missing/conditional points correctly
- give users precise, actionable feedback
- eliminate confusion and frustration