Building an AI Wardrobe Intelligence System

From clothing images to personalised outfit recommendation: how Wearsper turns a photograph of a garment into structured data a recommendation system can reason over.

PHOTO **Teslim Bello** · Founder & AI/Product Developer, Wearsper · September 11, 2026

Teslim holds an MSc in Data Science (Distinction, University of East London) and a BSc in Mathematics and Statistics, and built Wearsper's backend, mobile application, and AI pipeline independently.

1. Overview

A photograph of a shirt is, to a computer, an unstructured grid of pixel values. A person looking at the same photograph immediately understands it as a garment with a category, a colour, a level of formality, and a set of other garments it would or wouldn't pair well with. Wearsper's core technical problem is closing that gap: turning a wardrobe of ordinary photographs into a structured representation that a recommendation system can actually reason over, and using that representation to suggest outfits from clothes the person already owns.

This is a case study of the pipeline as it's actually built today, not a proposal or a roadmap presented as finished work. Sections are written to distinguish clearly between what is implemented and what is a planned future direction, and development-time observations are presented as exactly that, not as formal benchmarks.

2. The problem

The simplest version of a digital wardrobe app is a photo album: a folder of pictures the user manually sorted into categories. This is a reasonable starting point, but it caps out quickly, because a folder of images has no way to answer the questions an outfit recommendation actually depends on:

- What category is this item — a shirt, a jacket, a pair of trousers?
- What colour is it, and does that colour combination work with another item?
- Is it formal, casual, or something in between?
- Does the person already own something that would clash or compete with it?
- Is it appropriate for the current weather or occasion?

None of these questions can be answered from a raw image file. They require the image to first be converted into attributes — discrete, structured facts about the garment — before any reasoning about outfits can happen at all. This is the reason Wearsper's architecture treats image understanding as a distinct, upstream stage from recommendation, rather than asking a single model to go directly from "here are some photos" to "here's an outfit."

3. From images to structured wardrobe intelligence

When a user photographs a clothing item, the image moves through a sequence of distinct stages before it becomes a usable wardrobe entry:

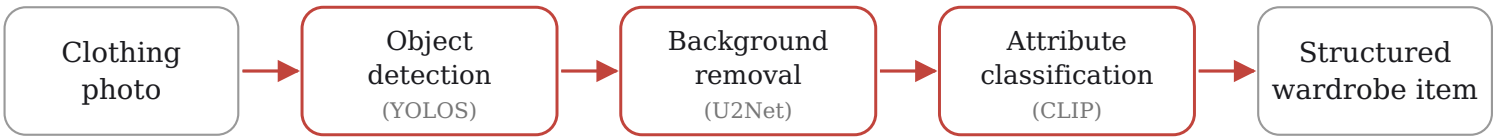

Figure 2 — Image-to-wardrobe pipeline. A photographed garment passes through detection, segmentation, and attribute classification before it's stored as a structured wardrobe entry.

The pipeline uses a locally-run object detection model to first localise the actual garment within the photo — separating it from background clutter, other objects in frame, or the person wearing it. A separate background-removal step produces a clean cutout of the item, which is what the user sees represented in their digital wardrobe. A third stage, built on a CLIP-based vision-language model, classifies visual attributes such as colour and style category.

These are three separate, specialised models rather than one general-purpose model asked to do everything, because each stage solves a genuinely different problem: finding where the garment is, isolating what it looks like visually, and describing what kind of thing it is. Chaining specialised stages together produced more reliable, more debuggable results during development than a single larger model attempting the same task end-to-end.

> **On confidence and uncertainty.** Not every photo is classified with equal confidence. Wearsper supports two classification modes: an automatic mode that applies the model's best guess directly, and a manual review mode that surfaces lower-confidence items for the user to confirm or correct before they're treated as final. This exists because a wrong automatic guess in a wardrobe app has a real cost — it directly corrupts the exact data a later recommendation depends on.

4. Representing the wardrobe as structured data

Once a garment has been through the pipeline above, it isn't stored as "an image with a caption." It's stored as a record with distinct fields — category, colour, style attributes — alongside the processed image itself. A simplified version of one wardrobe item's structure:

```
{ "category": "tops", "garment_type": "shirt", "colour": "navy", "style": "smart-casual", "classification_status": "done" }
```

This is a simplified illustration of the shape of the data, not a literal database export. The important property it demonstrates is that a garment's category, colour, and style are each independently addressable fields, not text buried inside a single description.

5. The recommendation layer

Once a wardrobe is represented as structured data, generating an outfit recommendation becomes a reasoning problem over that structured data, rather than a raw image-understanding problem. Wearsper's recommendation layer takes the user's structured wardrobe, along with context such as an occasion the user describes or the current weather, and uses an AI reasoning step (currently Google's Gemini) to propose a coherent outfit from the items that already exist in the wardrobe.

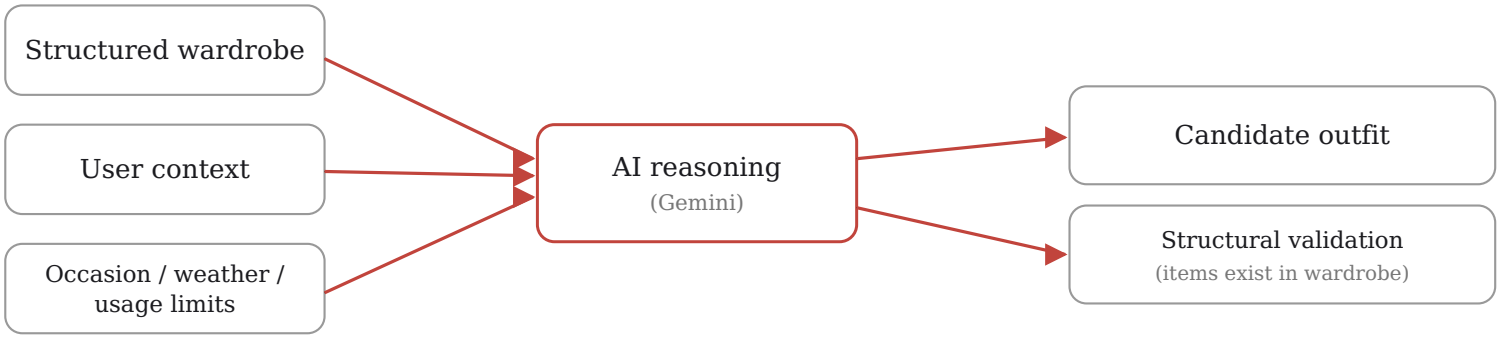

Figure 3 — Recommendation pipeline. Structured wardrobe data and user context are passed into the reasoning step; its output is then checked against the wardrobe before being shown to the user.

The language model is not treated as the source of truth for what's in the wardrobe. The computer vision stage already produced the factual record of what the user owns. The reasoning layer operates on top of that existing structured record — it doesn't re-derive it, and its output is checked against it.

6. Why this architecture

Separating "what does the user own" from "what should the user wear" into two distinct stages was a deliberate trade-off. The advantages observed during development:

- **Contained failure modes.** A reasoning mistake stays a recommendation problem, not a data-integrity problem.
- **Debuggability.** It's possible to tell whether a bad recommendation came from wrong wardrobe data or faulty reasoning over correct data.
- **Cost and latency control.** Classification happens once at upload time and is cached as structured data.

The trade-offs are real: more moving parts, and a genuine dependency on the classification stage being accurate — a misclassified item propagates its error into every future recommendation involving it, silently, until corrected.

7. Engineering challenges

Inconsistent classification confidence

Real user-submitted photos vary enormously in lighting, framing, and background clutter compared to clean product photography, which directly motivated the dual automatic/manual classification mode.

Concurrency in the image-processing pipeline

Running multiple garment uploads concurrently surfaced a real issue: the background-removal library's underlying numerical routines were not safe to run from multiple threads at once, crashing the worker process. The fix serialised just that specific operation, preserving concurrent throughput elsewhere.

Resource contention under load

Running multiple ML worker processes on one host caused each to compete for every CPU core simultaneously, producing a severe, measured slowdown until explicit thread-count limits were set per process.

Third-party integration correctness

A real production bug traced usage-limit resets failing after a subscription tier upgrade to the backend reading field names that didn't match a billing provider's actual, documented response schema.

8. Performance and observations

Formal benchmarking, with a controlled methodology and representative dataset, remains future work. What follows are observed request timings from development, not a performance claim.

Stage (recommendation request)	Observed duration
Wardrobe fetch	~605 ms
Request processing before model call	~1.1 s
Gemini reasoning request	~14.1 s
Total observed	~15.1 s

The large majority of end-to-end latency sits in the language model reasoning call itself, not in Wearsper's own data-fetching or processing logic — a useful, honest data point for prioritising future latency work.

9. Product architecture

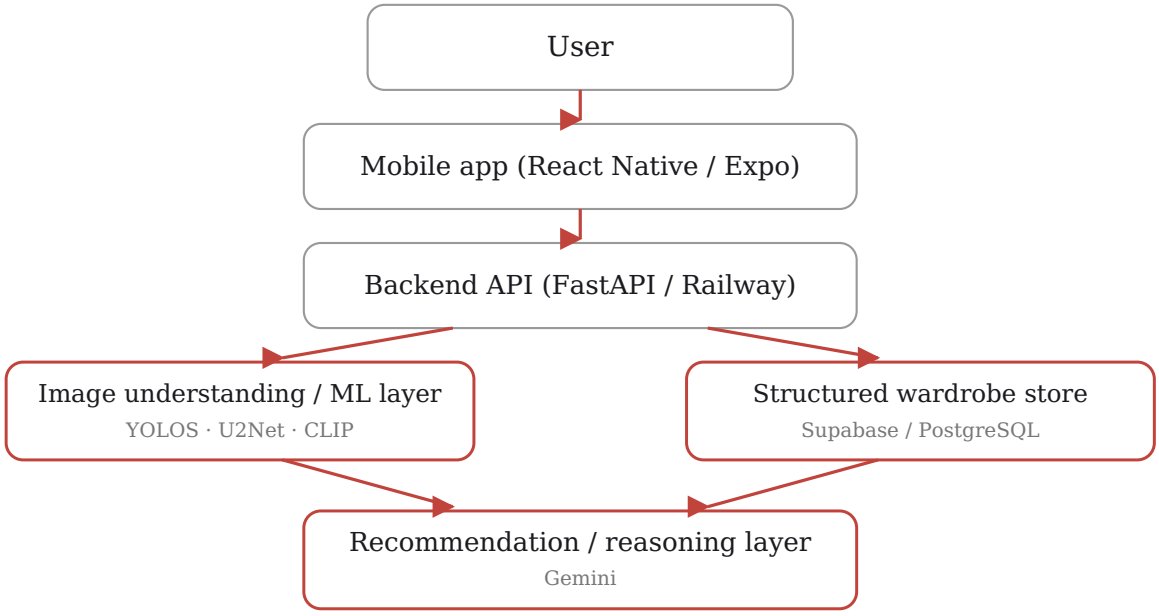

Figure 1 — System architecture. The mobile app talks only to Wearsper's own backend; the backend coordinates between the ML layer, the structured data store, and the reasoning layer.

10. What makes wardrobe intelligence different

This is not a generative fashion tool that produces new clothing designs or dresses a person in garments they don't own. The system is constrained to reason over a specific, real, individually-owned inventory — making better use of clothes that already exist, which is a meaningfully more constrained problem than open-ended fashion generation, since every recommendation has to remain grounded in a real, verifiable set of physical items.

11. Current limitations

- **Classification is not perfect**, particularly on ambiguous garments, unusual lighting, or overlapping items in a single photo.
- **Colour and style are inherently somewhat subjective** — a model's judgement won't always match an individual user's own categorisation.
- **Outfit compatibility reasoning is still relatively coarse**, without yet incorporating a learned, personalised model of an individual user's own taste.
- **No large-scale evaluation exists yet** — observations here come from development and early real-world usage, not a structured study.
- **Latency is dominated by the external reasoning call**, a dependency outside Wearsper's own direct control.

12. Future research and development

Implemented Weather-aware recommendations, using current conditions as context for the reasoning layer.

Planned **Wardrobe gap analysis** — identifying categories missing from a wardrobe, not only recommending from what exists.

Planned **Learned personal preference modelling**, improving from real accept/reject feedback over time.

Planned **Embedding-based wardrobe representation**, supplementing categorical attributes with learned visual embeddings.

Planned **Formal recommendation evaluation**, replacing development-time observations with a structured methodology.

Planned **Latency optimisation of the reasoning call**, the current dominant cost in end-to-end recommendation time.

13. Conclusion

The underlying problem this system addresses — turning unstructured personal visual data into a structured representation that a reasoning system can act on reliably — is broader than clothing. The decision that has mattered most is keeping the factual record (what a person actually owns, per computer vision) architecturally separate from the reasoning applied on top of it (what they should wear, per an AI model). That separation keeps failures contained and debuggable, and is the design principle most worth carrying forward as the reasoning layer continues to improve.