Correction-conditioned media walls for personal taste alignment

Alan N. Pham[1,*]

[1]AO Labs

[*]Correspondence: aolabs.io

Personal recommendation systems usually learn taste from ratings, clicks, watch time, purchases, or dense behavioral logs. Yum explores a different unit of personalization: the user's correction. A correction such as "do not show this model, color, pose, ingredient, or source pattern again" is treated as a reusable test that updates a deployed media wall through hard rejects, compact preference memory, source de-duplication, visual-group spacing, and optional model-assisted candidate ranking. The novelty is not another image feed, but a correction-conditioned control layer for personal AI alignment: free-form dissatisfaction is converted into inspectable feed constraints without requiring a large surveillance history. We describe a live static-first website, yum.aolabs.io, that mixes food, a preference-shaped modern-sedan lane anchored by an Audi A3 build, and K-pop imagery while preserving category balance and cross-device preference state. Screenshots of the deployed surface are included as artifact evidence. The system is presented as a design study for preference-aware personal media, not as a population-scale recommendation benchmark.

1 Introduction

Recommender systems have long framed personalization as a prediction problem: infer what an individual will rate, click, purchase, or consume next (*1–4*). That framing works when the objective is stable and the signal is abundant. It is weaker when the target is a private aesthetic standard that changes through direct correction: "not this color," "not this kind of image," "not this person," "not this texture," "not this crop," or "never cluster several similar items again."

Yum is built around that second setting. It is a personal media wall whose useful behavior is defined less by aggregate engagement than by whether it stops repeating the user's rejected mistakes. The core design question is therefore not "how can the system maximize interaction?" but "how can a single user's corrections become durable operational constraints on a live visual artifact?"

This question matters beyond one website. Human-AI systems increasingly depend on natural-language feedback, preference learning, and alignment from human judgments (*5–7*). However, most deployed personal interfaces still leave users with brittle local settings, opaque ranking models, or short-lived dislike buttons. Yum treats correction as a first-class artifact. A correction is translated into source keys, semantic rejects, group-spacing penalties, category quotas, and preference samples that are small enough to inspect and persistent enough to matter across sessions.

2 Novelty

2.1 From implicit feedback to explicit correction

Implicit-feedback recommenders infer taste from behavior that was not necessarily intended as feedback (*3*, *4*). A scroll, linger, or click can be ambiguous. Yum instead elevates direct correction. The user may reject an exact item, but the implementation must infer the reusable failure mode: the source may be low-quality, the category may be overrepresented, the subject may be repeated, the image may violate an aesthetic rule, or a cluster may be visually monotonous.

The proposed novelty is *correction-conditioned curation*: a deployed feed architecture in which free-form human correction is compiled into a transparent constraint stack. The stack has four properties.

1. It is **persistent**: a correction can affect a later page load and a different device through remote preference state.
2. It is **inspectable**: hidden keys, hidden samples, kept samples, blocked terms, and group identifiers are ordinary data rather than an opaque behavioral archive.
3. It is **compositional**: hard rejects, category balance, source rotation, model ranking, and layout placement each address a different failure mode.
4. It is **artifact-facing**: the final test is not an offline score but whether the public wall stops showing the rejected pattern.

2.2 Why this is different from a dislike button

A dislike button removes an item. A correction-conditioned system must generalize without over-generalizing. If the user rejects a blurred K-pop image, the system should prefer higher-resolution images without eliminating the subject. If the user identifies a specific configured vehicle as especially desirable, the system should keep that build as a high-confidence anchor without collapsing the entire car lane into one vehicle. If several images from one source or one car family appear together, the system should treat the cluster as a layout and queueing failure rather than a single bad tile.

This makes the central object of study a feedback-to-artifact loop. The user does not need to become a taxonomy editor. The system must convert correction into durable tests and then verify the deployed surface against those tests.

3 System design

Yum is a static-first site with optional dynamic services. The public wall is served at yum.aolabs.io. The frontend contains known-good seed media, source metadata, category labels, captions, source URLs, shape hints, focus hints, and optional person or car-group identifiers. In the current deployed build, the car lane restores the prior modern-sedan pool while keeping Audi Code A0J42547, an A3 TFSI quattro Premium Plus S tronic in Manhattan Gray metallic with Black optic package and Parchment Beige–Steel Gray interior, as a high-confidence local anchor. Vehicle candidates now pass a composition gate that rejects side-only profiles, cabin and detail crops, wheel or trim fragments, missing local derivatives, known source identifiers whose actual image contradicts its exterior caption, and margin-prone frames that would render as a small car inside a dark tile. K-pop candidates pass a body-and-styling gate: a tile must show exposed shoulders, exposed midriff or navel, both, or a clearly adult-era playful/confident pose, while coat, jacket, blazer, sweater, long-sleeve, turtleneck, stage, concert, heavy-makeup, and minor-era sources are rejected before display. Food candidates pass a composition gate that now rejects vegetable spreads, lunch boxes, packaged meals, visible people, family meals, children, dining-room scenes, home or office food snapshots, people-at-table scenes, and weak dining-documentation frames. The static K-pop pool was expanded with source-reviewed, body-visible Kpopping frames, and known repeated or covered-shoulder sources are excluded before the person-balanced wall is built. A small Node service provides preference persistence, model-assisted candidate ranking, and K-pop candidate retrieval. The static wall remains useful if the dynamic service is slow or unavailable.

3.1 Taste state

Preference state is intentionally compact. It contains hidden source keys, hidden samples, kept samples, a version number, and timestamps. Hidden keys remove exact items. Hidden samples provide stronger negative examples. Kept samples provide weak positive context from nearby non-hidden material. This is a privacy-leaning alternative to large behavioral histories: the state is small, inspectable, and biased toward explicit correction rather than passive surveillance.

3.2 Candidate control

Candidate generation and candidate control are separated. Candidate generation finds possible images from static pools and remote sources. Candidate control decides what is allowed to reach the wall. Control includes blocked terms, source de-duplication, category balancing, recent-person spacing, visual-group spacing, car-group spacing, quality thresholds, composition gates, natural-aspect tile sizing, and optional language-model ranking. For the present vehicle preference, the car lane allows modern compact and executive sedans such as the Audi A3/A4/A5/S3/RS3, BMW 2-series Gran Coupe and 3-series, and Mercedes CLA/C-Class while rejecting SUVs, fire trucks, old cars, disliked color/model clusters, adjacent non-sedan body styles, boring side-profile product renders, crop-heavy fragments where the full car is not visible, and source frames whose dimensions would create visible letterboxing. The same curator gate is applied to local seed items, queued refills, remote candidates, and stale queue entries so old fallback pools cannot bypass newer taste constraints. If a category stalls, local same-category cycling extends the wall without allowing one reliable category to overrun the 1:1:1 mix.

This separation is important because visually bad outcomes often arise after a candidate has passed a relevance check. Four similar cars in one view, three portraits of the same person, or repeated low-resolution source patterns are layout-level failures. They must be handled by queue and placement logic, not only by candidate retrieval.

4 Deployed artifact

The deployed wall is shown in Fig. 1 and Fig. 2. The screenshots show the actual interaction surface: a paper link, the wall itself, mixed subject categories, and a responsive masonry layout.

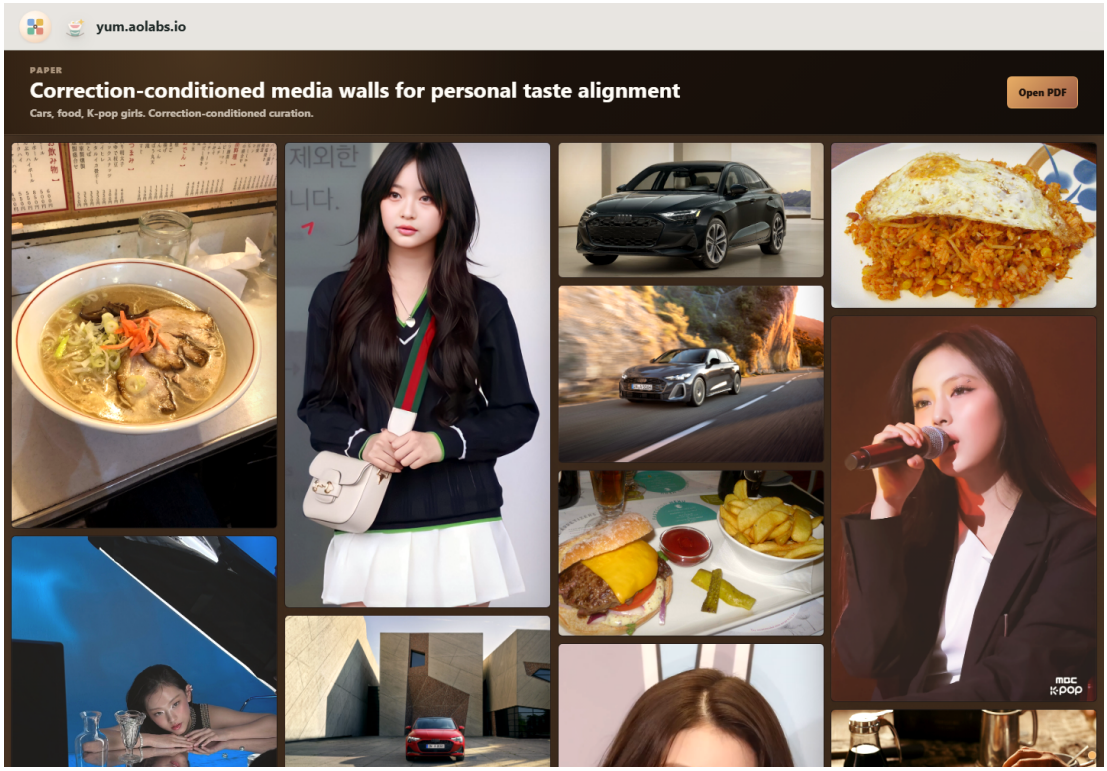

Figure 1. Desktop view of the Yum wall after the vehicle lane added frame-aware rendering. Car tiles use the loaded image's natural aspect ratio and cover the tile without artificial dark margins, while the candidate gate still favors whole-car or three-quarter exterior frames and preserves the food/car/K-pop category cadence.

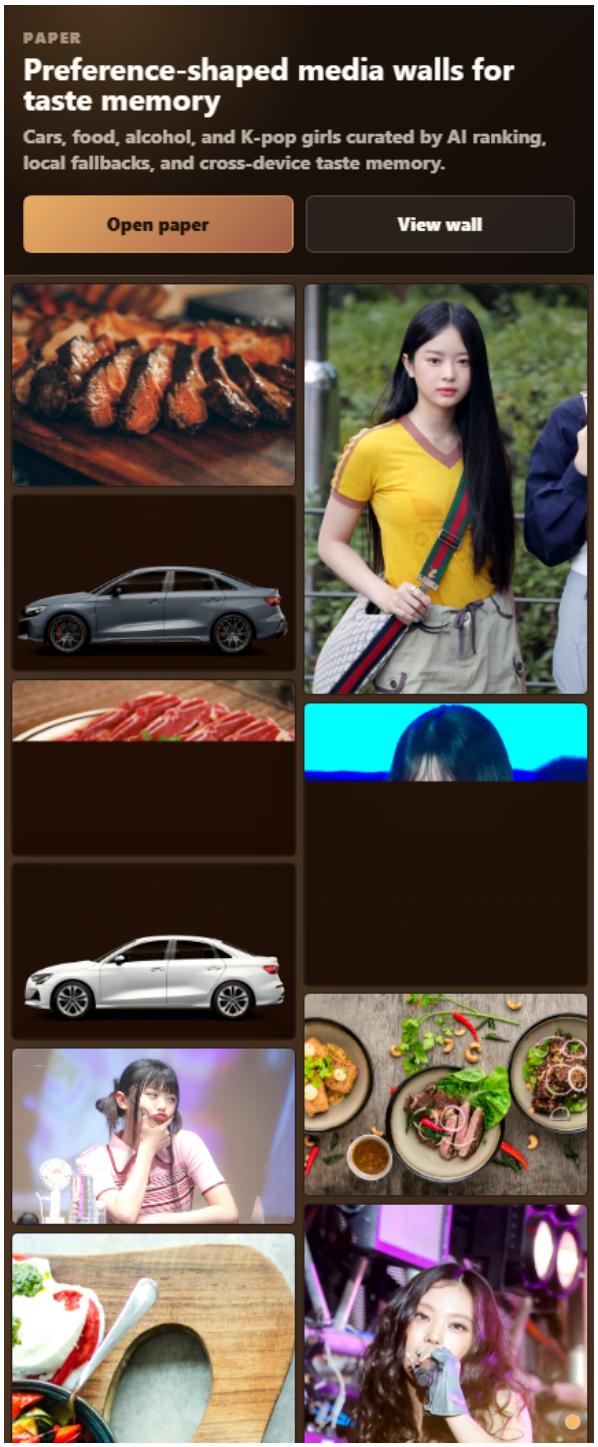

Figure 2. Narrow viewport view of the deployed wall. The same feed contract must survive a different column count, where clustering is more noticeable and lazy-loading failures are easier to see.

5 Relationship to prior work

Yum builds on recommender-system work but shifts the evaluation target. Classic surveys and hybrid recommender methods emphasize prediction, filtering, and combining sources of evidence (*2*, *8*). Implicit-feedback methods formalize preference from observed behavior (*3*, *4*). Visual recommendation work extends this to appearance-sensitive domains (*9*). Human-centered recommender evaluation argues that perceived quality, variety, effort, and privacy affect system experience, not just predictive accuracy (*10*).

The correction-conditioned wall fits between these traditions and human-AI interaction. Human preference learning and instruction-following models demonstrate that human judgments can train or steer model behavior (*5*, *6*). Human-AI guidelines emphasize intelligibility, controllability, and graceful handling of user feedback (*7*). Vision-language models show that natural language can name visual concepts and support flexible image understanding (*11*). Yum applies these ideas at the product-surface scale: the user provides corrections in ordinary language, and the live artifact is updated through explicit feed constraints.

6 Implications

The strongest implication is that small personal systems can be aligned through correction memory without becoming black-box surveillance systems. A personal media wall does not need to store every scroll event to improve. It can store what the user explicitly rejected, what remained nearby, and how those examples map to transparent constraints. This produces a more repairable system: when the wall is wrong, the correction can become a test.

The second implication is that personalization should be evaluated at the artifact boundary. A recommender may rank candidates correctly in isolation while still producing a bad page because similar items cluster, a category disappears, or a visual pattern repeats. Yum therefore treats the rendered wall as the unit of verification.

7 Limitations

This manuscript reports a deployed design study, not a controlled user study. The current evidence is artifact-level: source code, deployed behavior, screenshots, and live verification. No population-scale

preference model, benchmark dataset, or statistical generalization claim is made. The system also inherits the limitations of third-party image sources, including uneven metadata quality, unavailable images, source licensing constraints, and inconsistent visual resolution.

The next useful step is not to add more opaque intelligence. It is to measure correction-to-fix latency: after a user gives a rejection pattern, how quickly does the live wall stop violating it, and how often does the generalization accidentally suppress material the user would have liked?

8 Conclusion

Yum argues for correction-conditioned curation as a practical unit of personal AI alignment. A user should not need to repeatedly reject the same failure mode. The system should convert correction into persistent, inspectable constraints and verify the live artifact against those constraints. For personal visual media, this may be a more useful target than engagement maximization: the wall becomes better when it remembers what the user meant, not merely what the user clicked.

References

1. P. Resnick, H. R. Varian, Recommender systems. *Communications of the ACM* **40**, 56–58 (1997).
2. G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* **17**, 734–749 (2005).
3. Y. Hu, Y. Koren, C. Volinsky, presented at the 2008 Eighth IEEE International Conference on Data Mining, pp. 263–272.
4. S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, presented at the Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461.
5. P. F. Christiano *et al.*, presented at the Advances in Neural Information Processing Systems, vol. 30.
6. L. Ouyang *et al.*, presented at the Advances in Neural Information Processing Systems, vol. 35, pp. 27730–27744.
7. S. Amershi *et al.*, presented at the Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1–13.

8. R. Burke, Hybrid recommender systems: survey and experiments. *User Modeling and User-Adapted Interaction* **12**, 331–370 (2002).

9. R. He, J. McAuley, presented at the Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 144–150.

10. B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, C. Newell, Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* **22**, 441–504 (2012).

11. A. Radford *et al.*, presented at the Proceedings of the 38th International Conference on Machine Learning, pp. 8748–8763.

Methods

Artifact capture

Desktop and narrow-viewport screenshots were captured from the deployed public URL using Playwright. The current desktop figure was captured on May 17, 2026 and stored as `assets/yum-wall-desktop-20260517-car-frame.png`; the narrow-viewport figure was captured on May 7, 2026 and stored as `assets/yum-wall-mobile-20260507.png`.

Static surface and media metadata

The public surface is generated from a static bundle containing the HTML shell, CSS, JavaScript wall renderer, icon assets, preview images, paper files, and figure assets. Media candidates are represented as structured entries with image source, source URL, caption, category, tile shape, focus hint, and optional person or visual-group metadata.

Preference storage

Preference state is stored as a compact object containing hidden keys, hidden samples, kept samples, a version number, and update metadata. The browser keeps a local mirror for immediate interaction. The backend stores the same preference shape so state can be restored on another device when the remote service is reachable.

Candidate ranking and layout

Candidate ranking is static-first. Known-good seed media is available before any remote service responds. Optional curation requests can pass candidate metadata and preference samples to the configured model. Blocked terms, source keys, category balance, visual groups, person groups, car groups, quality thresholds, and recent placement history are then applied before display. The renderer maintains a background buffer using complete category sets, replaces failed images from the same category, and refills stalled categories from same-category local pools. Tiles are not made visible until their image has loaded, so failed URLs cannot leave visible blank card shells at the bottom of the wall. Quality and composition control happens before display through source, caption, group, URL, Commons category metadata, and image-information gates; the load path no longer rejects already loaded images in a way that can cascade into heavy re-layout. Hidden pending images are still fetched eagerly; otherwise the browser can stall because lazy images inside non-displayed tiles may never load. Failed images are removed and replaced in place without rebuilding the whole masonry wall, and slow images are not rejected on a short timer. Incremental placement uses stored column height scores rather than repeated live height measurements, so adding more tiles does not force layout reads across the whole wall. The refill trigger counts live loaded tiles and a bounded active pending buffer, not only intended records, so rejected images cannot make the wall believe it is full, pending images cannot flood the browser before paint, scroll/sentinel loading is gated by the same pending-state check, and column balancing does not initiate extra background loads. The car refill pool uses the curated modern-sedan set, including the A3 build derivatives, rather than unrestricted vehicle search. The K-pop refill prompt now treats playful or confident adult-era posing as an allowable positive signal but still rejects outerwear-heavy styling. Online food search is stricter: Commons titles, categories, and image metadata are screened for people, family, child, dining-room, and home-meal documentation terms, and external food candidates are accepted only when the model-assisted curator approves them after the metadata gate. If that service is unavailable, the wall cycles same-category curated food seeds instead of letting weak online search results through. This keeps the wall from terminating into blank space without allowing one reliable category to overrun the others.

Production synchronization

The deployed system has two independently updated surfaces: the GitHub Pages static bundle at yum.aolabs.io and the Railway Node service that serves preference and candidate APIs. A correction is not considered shipped until both surfaces are current. The May 19, 2026 production check found a split state: the static bundle already served the v41 correction gates, but the Railway `/api/kpop-candidates` endpoint still returned stale Hanni Fashion Week, Hanni 2022, Haerin Fashion Week, and Wonyoung 220513 candidate sources. The Node service was redeployed to Railway as deployment `b705079c-e3a7-4b57-8a45-1ba2a4a1d3f4`. Direct API queries for Hanni, Haerin, and Wonyoung then returned zero hits for those rejected source families, while the live JavaScript bundle still contained the outerwear gate and refill heartbeat and omitted the old Fashion Week and Ningning static seeds.

Verification

The manuscript PDF was generated from this LaTeX source using `pdflatex` and `biber`. The deployed site was verified by checking that the live paper URL returned a PDF, that the live page served the current JavaScript bundle, that the bundle contained the current correction-conditioned curation rules, and that the Railway candidate API no longer returned the source families removed by the correction gates.

Data availability

No new research dataset was generated. The media metadata required to inspect the public wall is included in the deployed JavaScript bundle and in the Yum application source. The current car lane combines local image derivatives extracted from a user-supplied Audi configuration artifact for Audi Code A0J42547 with restored third-party modern-sedan sources. Third-party media remains subject to the licensing and availability terms of the original sources linked from the wall.

Code availability

The application code and this LaTeX source are available in the Yum application repository at github.com/nalalalan/yum-app. The deployed LaTeX source is served at yum.aolabs.io/yum-preference-shaped-media-wall.tex.

Acknowledgements

No external acknowledgements are declared.

Funding statement

No external funding is declared.

Author contributions

A.N.P. conceived the system, implemented the wall and backend, designed the curation and preference-memory behavior, prepared the manuscript, and maintains the public deployment.

Competing interests

A.N.P. created and operates AO Labs and Yum.

Additional information

Supplementary information is not included. Correspondence and requests for materials should be directed to Alan N. Pham through aolabs.io.