الجمـهورية الجزائرية الديمقراطية الشعبيـة
وزارة التعليم العالي والبحث العلمي
جـامعـة سعـيـدة د. مـولاي الطـاهـر
كلية الرياضيات و الإعلام الآلي و
الاتصالات السلكية و اللاسلكية
قسم: الإعلام الآلـي

Mémoire de Master en informatique

Spécialité : **Intelligence Artificielle : Principe and Application**

Thème

An AI-Powered Personalized Product Recommendation System for Algerian E-Commerce

▪ **Présenté par :**

Bouzig Mustapha Abdel Illah

Zouani Salah Eddine

▪ **Dirigé par :**

Dr. MEKOUR Mansour

Année universitaire 2025-2026

Acknowledgments

We thank Almighty God for the health, patience and resolve that carried us through this work.

We thank our supervisor, **Dr. MEKOUR Mansour**, for the steady guidance and for reading every draft with the rigor only a real teacher brings. The clarity of the chapters that follow owes a great deal to his comments, his patience with our first attempts, and his willingness to let us defend our design choices when they were right and to push back firmly when they were not.

We also thank the members of the jury for accepting to evaluate this work. Their critical reading is what makes a thesis a thesis. We hope the pages that follow are worth their time.

To our teachers at the University of Saida, Dr. Moulay Tahar, who taught us, over five years, the techniques of computer science as well as the habits of careful thinking those techniques require: thank you. Several of you, at various points, told us that the question we were asking was harder than we thought. You were right. We hope we have at least begun to take the measure of it.

And finally, to our families and to our friends. To the parents who asked, every week, "so when do you defend?", and to the friends who sat next to us through the writing nights and the debugging mornings: this work would not exist without you. Every honest sentence in it was checked over a coffee in someone's kitchen first.

Dedication

To our parents,
who gave us the time and the trust to learn.

To our brothers and sisters,
who made the long writing days lighter.

To our friends,
who never let us take ourselves too seriously,
and who made these five years worth remembering.

To everyone who supported us, near or far,
this work is dedicated.

Bouzig Mustapha Abdel Illah & Zouani Salah Eddine

Contents

List of Figures

List of Tables

General Introduction

E-commerce has, in less than two decades, become the dominant interface between consumers and goods. What used to be a physical journey through markets, advisers and word-of-mouth has been compressed into a scroll through an effectively infinite catalog on a small glass rectangle. The shift is not merely a change of medium; it is a fundamental re-engineering of how supply meets demand, and one whose long-term social and economic effects are still being measured.

This compression has a hidden cost: *relevance*. When a customer is shown thousands of products with no priority, the catalog is statistically rich but cognitively poor. The user is forced to do the filtering that, a generation ago, an attentive merchant or a well-edited catalog used to do for free. Modern e-commerce platforms therefore stand or fall not on the size of their catalog but on the quality of the ranking they show on the first screen, in the first three seconds.

Recommender systems were invented to restore that filtering layer. By learning from past purchases, ratings, clicks or declared preferences, they re-order the catalog so that the items most likely to interest the current user appear first. Industry-scale platforms such as Amazon, Netflix, Spotify and YouTube owe a large fraction of their engagement to this single layer; estimates from the early 2010s already attributed 30 to 60 % of consumption on those platforms directly to the recommender. Recommendation is, today, one of the most studied applied areas of machine learning and arguably the one with the most direct impact on the everyday Web user.

Two persistent difficulties, however, remain only partially solved.

- **The cold-start problem.** A brand-new user has no history; a brand-new product has no interactions. Classical collaborative filtering cannot rank what it has never seen. The result is a catalog that is, at first contact, exactly as unhelpful as one without any recommender.
- **Explainability.** Modern recommenders are increasingly driven by deep neural networks whose outputs are accurate but opaque. The user is offered a product without being told *why*, which damages trust and prevents the user from correcting a misclassification about their own taste.

A third concern, sharper in recent years, is **privacy**: a system that needs to send a complete user profile to a remote server in order to recommend a pair of shoes is increasingly perceived as disproportionate. The user trades surveillance for ranking quality, often without realising the bargain has been struck. Regulatory regimes in Europe (GDPR), California (CCPA) and increasingly the Arab world, including Algeria's own Law 18-05 on electronic commerce

[18] and Law 18-07 on the protection of personal data [19], push back against this trade.

Finally, in the specific context of **Algerian e-commerce**, two practical realities matter. The catalog mixes traditional crafts (kaftan, burnous, miel de Kabylie, zellige) with mainstream lifestyle products in proportions that no Western catalog reflects. And the customer base naturally switches between French and Arabic within the same session, sometimes within the same product description. A serious recommender for this market must therefore be multilingual by construction and culturally aware of a local catalog that no large pretrained model has ever seen at training time.

This thesis. We present **AlgerieShop IA**, a privacy-first, explainable, multilingual recommender system designed and implemented to address these problems at once. Its central idea is a **12-dimension preference vector** that represents every user and every product in the same mathematical space. The vector is:

- built from a two-minute **visual persona session** (cold-start solved with three clicks, no questionnaire, no email address required);
- matched against products via a **custom K-Nearest-Neighbours engine using cosine similarity**, running directly in the user's browser without any server round-trip;
- **automatically generated** for new products by a zero-shot multilingual NLP microservice based on `mDeBERTa-v3-base-mnli-xnli`;
- **continuously updated online** after every explicit signal (add-to-cart, "not interested") so the persona reflects the user's evolving taste rather than a frozen sign-up form (a view is logged only for the crowd popularity signal and never nudges the persona).

Structure of the thesis. The document is organised in three chapters, framed by this general introduction and a general conclusion.

- **Chapter 1: Recommendation Systems: Concepts, Models and Challenges.** The scientific foundation: the formal problem, the families of recommenders (content-based, collaborative, hybrid, knowledge-based, context-aware, session-based, deep), the similarity measures that drive them, the metrics by which they are evaluated, the cold-start and privacy problems, the explainability question, and the zero-shot multilingual NLP technique on which our content vectors rest.
- **Chapter 2: The Algerian E-Commerce Ecosystem.** The market the system is built for: a brief history of e-commerce, the order management pipeline, the technological tools typically deployed (ERP, CRM, WMS, PIM), the Algerian market in detail (platforms, payment, delivery, languages, traditional crafts), the regulatory landscape, the challenges specific to the Algerian context, and the implications all of these have for the recommender we will build.

- **Chapter 3: System Implementation and Performance Evaluation.** The running system: the 12-dimensional preference space, the visual persona builder, the in-browser KNN engine, the online learning rules, the hybrid CB+CF blend, the FastAPI NLP microservice, the Supabase schema, the security and privacy implementation, the deployment topology, and the experimental evaluation of latency, zero-shot accuracy, cold-start quality and online-learning stability, along with a comparison against baselines.

Contributions. The individual ingredients (KNN, cosine similarity, zero-shot NLI, hybrid CB+CF, online learning) are standard. The contribution of this thesis is the *end-to-end system design*: combining a visual cold start, an NLP-generated semantic vector space, an in-browser KNN, and real-time online learning into a coherent privacy-first product that ships, runs on a laptop, and remains explainable end to end. The Algerian contribution is concrete rather than cosmetic: an all-Algerian, multilingual (French/Arabic) catalog; a zero-shot labeler that maps that catalog onto the preference space without any training data; and a cold-start, on-device design that fits the local trust and connectivity context.

Chapter 1

Recommendation Systems: Concepts, Models and Challenges

1.1 Introduction

A *recommender system* is an information-filtering component whose job is to predict a user's preference for items the user has not yet seen and to surface the most preferred ones. The field sits at the intersection of information retrieval, machine learning, optimization, behavioural economics and human-computer interaction. It is both old, since the first published recommenders date from the mid-1990s, with systems such as Tapestry (Goldberg et al., 1992) and GroupLens (Resnick et al., 1994), and constantly renewed by the arrival of new representations, new learning algorithms and new ways to measure quality.

This chapter is the scientific spine of the thesis. We begin by formalising the recommendation problem mathematically, then walk through its families one by one, with their strengths, weaknesses and mathematical formulations. We discuss the similarity measures that sit at the heart of nearly every content-based and memory-based collaborative recommender. We give the metrics by which a recommender is judged: both the predictive-accuracy kind and the beyond-accuracy kind that the recent literature has come to value. We then turn to the two enduring challenges of the field that frame our design choices later: the cold-start problem and the privacy question. We close with the zero-shot Natural Language Inference machinery on which our content vectors rest, since the recommender ranks vectors and the NLP labeler is what produces those vectors for products that have only a textual description.

1.2 Formal problem statement

Let $U = \{u_1, \ldots, u_N\}$ be a set of users and $I = \{i_1, \ldots, i_M\}$ a set of items. A recommender system attempts to learn or define a *utility function*

$$f : U \times I \to \mathbb{R}, \tag{1.1}$$

where $f(u, i)$ estimates how much user u would like item i. For a target user u the system returns the top-K items maximising $f(u, \cdot)$ over the items not yet interacted with.

The utility f can be estimated in three broad ways:

- **Explicit feedback.** Users assign numeric ratings on a discrete scale (the Netflix five-star scale, the Amazon thumbs up/down). Explicit ratings are precise but expensive to collect.
- **Implicit feedback.** The system infers preference from behavioural signals: clicks, page dwell time, add-to-cart, purchase, share, skip, repeat play. Implicit signals are abundant but noisy; an add-to-cart is not always followed by a purchase, and a purchase is not always followed by satisfaction.
- **Content features.** The system builds a vector representation of each item directly from its attributes (title, description, image, tags) without requiring user history. Content features are the only signal usable for items that have just been added to the catalog.

In real-world systems the three are mixed. The recommender we build in Chapter 3 uses content features (from a multilingual NLP labeler), implicit feedback (cart-adds, dismissals) and an explicit visual cold-start (the persona picker), but no numeric ratings.

1.3 A brief history of recommenders

The history of recommender systems, surveyed in depth by the *Recommender Systems Handbook* [21], can be cut into four overlapping eras.

Early collaborative filtering (1992 to 2000). The first research recommenders, Tapestry at Xerox PARC [6] and GroupLens at the University of Minnesota [20], treated recommendation as a user-user similarity problem over a sparse interaction matrix. The algorithms were simple: find users whose past ratings correlate with the target's, weight-average their ratings on unseen items. The systems worked but did not scale beyond a few thousand users.

Item-to-item and large-scale CF (2001 to 2008). Amazon's 2003 paper on item-to-item collaborative filtering [14] showed how to flip the similarity computation onto the item side, where the matrix is much smaller and much more stable over time. The shift made Web-scale recommendation possible. Around the same period, matrix-factorisation methods began to displace memory-based CF for accuracy, especially after the Netflix Prize (2006 to 2009) made factorisation methods the de facto reference [12].

Hybrid and context-aware (2008 to 2015). The realisation that no single family wins outright led to hybrid systems [2, 3] and to the rise of context-aware recommenders (time, location, device). The mobile Web, with its strong session and location signals, pushed the field in this direction.

Deep and sequence-aware (2015 to today). Neural collaborative filtering [7], recurrent session-based recommenders such as GRU4Rec [9], self-attentive sequential recommenders such as SASRec [11], graph neural networks on user-item graphs, and most recently large language models used as zero-shot rerankers. These methods are accurate on dense, sequence-rich datasets but introduce opacity and operational cost that a small or privacy-sensitive deployment cannot always absorb.

Our system sits deliberately at the simple end of this history. The recommender is a content-based KNN with cosine similarity and a naive popularity blend; the heavy machine learning lives only in the NLP labeler that fills the content space. The choice is justified by the size of the catalog (120 products at present), the explainability requirement, and the wish to keep all user data on the user's device.

1.4 Families of recommender systems

1.4.1 Content-based filtering

Content-based (CB) recommenders score items by the similarity of their content representation to a model of the user's past interests [22, 1]. Each item is mapped to a vector in some feature space:

- TF-IDF or BM25 vectors over the textual description for text-heavy catalogs;
- raw or learned pixel embeddings for image-heavy catalogs (fashion, art);
- manually curated structured tags for catalogs with strong ontologies (books by genre, films by IMDb keyword);
- , as in our case, an NLP-generated probability distribution over a fixed set of thematic labels.

The user profile is the aggregate, typically the centroid, of the vectors of items the user liked. Recommendation reduces to a k-nearest-neighbours query in that feature space.

Strengths.

- Works for brand-new items, since no interaction history is required. This is the single most important property in fast-moving catalogs.
- Explainable when features are interpretable: the system can literally tell the user "recommended because this product is about cooking and you like cooking".
- No dependency on other users' data, which is a strong privacy advantage. CB recommenders can run entirely on the client.

- Stable under cold start of the system itself: a brand-new deployment can recommend on day one.

Weaknesses.

- Limited to items whose content can be represented as vectors of acceptable quality. Pure visual products (perfume, music) are harder to represent.
- Tends to over-specialise into the so-called filter bubble. A user who liked three cooking books is shown more cooking books, never reading books, even if the next book they would love is on travel.
- Misses preference patterns that are not articulated by the content ("people who like Stanley Kubrick also like Béla Tarr"): the CB system has no way to know that pattern unless Kubrick's and Tarr's content vectors happen to align.

1.4.2 Collaborative filtering

Collaborative filtering (CF) recommenders learn from the behaviour of similar users. They make no assumption about content. Two classical variants exist.

1.4.2.1 Memory-based CF

Memory-based CF (user-user or item-item) computes similarity between rows or columns of the user-item interaction matrix and predicts ratings by weighted averaging.

User-user CF. For a target user u and an unseen item i,

$$\hat{r}_{u,i} \;=\; \bar{r}_u \;+\; \frac{\sum_{v \in N_k(u)} \mathrm{sim}(u,v)\,(r_{v,i} - \bar{r}_v)}{\sum_{v \in N_k(u)} |\mathrm{sim}(u,v)|} \tag{1.2}$$

where $\bar{r}_u$ is user u's mean rating, $N_k(u)$ is the set of k users most similar to u (typically by Pearson correlation), and $r_{v,i}$ is user v's rating of item i.

Item-item CF. Symmetric: the similarity is computed between columns of the matrix. The Amazon paper [14] made item-item the de facto industry standard because item-item similarities are far more stable over time than user-user similarities and can be pre-computed offline.

1.4.2.2 Model-based CF

Model-based CF factorises the interaction matrix into low-rank user and item embedding matrices:

$$R \;\approx\; P\,Q^{\top}, \qquad P \in \mathbb{R}^{N \times d},\; Q \in \mathbb{R}^{M \times d}, \tag{1.3}$$

with d much smaller than both N and M. The factorisation can be learned by Singular Value Decomposition, Alternating Least Squares (ALS), Stochastic Gradient Descent on a regularised objective, or by Bayesian Personalized Ranking (BPR) when the feedback is implicit. The Netflix Prize papers are the canonical reference [12].

Strengths. Captures preferences a content model cannot articulate. Often the most accurate family on large dense datasets. The latent factors can be interpreted post-hoc as "audience clusters" or "style axes".

Weaknesses. Cold start for new users and new items (both have no row or column in the matrix). Sparsity is a further problem: most users have rated very few items, so the matrix is mostly zeros and the learning problem is ill-conditioned. Embeddings are not directly interpretable, so explanations are post-hoc reconstructions rather than direct readouts. The computational cost grows with the user base.

1.4.3 Hybrid recommenders

Hybrid recommenders combine CB and CF (and sometimes other signals) to inherit the strengths of each [2]. Burke classifies the combinations into five strategies:

- **Weighted**: the final score is a linear combination of component scores, $f = \alpha\, f_{\mathrm{CB}} + (1 - \alpha)\, f_{\mathrm{CF}}$. This is the form we use in this thesis.
- **Switching**: the system picks the better predictor per request based on a meta-rule ("CF for users with many interactions, CB for cold users").
- **Mixed**: several ranked lists are presented side by side, leaving the user to choose.
- **Feature combination**: features from one family are fed into the model of another (CF embeddings used as content features, or content features used as factorisation priors).
- **Cascade**: one family is used to filter, the other to re-rank.

The hybrid approach is the de facto industry default. Its main hyper-parameter is the blend weight α, which controls how strongly the system favours personal taste over crowd signal. Tuning α is itself a non-trivial problem: it can be set globally, per user, per session, or even per item. In our system α is exposed as a client-side parameter (a function argument to the in-browser ranker) so it can be A/B-tested without redeploying.

1.4.4 Knowledge-based and constraint-based recommenders

When user preferences are explicit and structured, for instance "I want a laptop under 80 000 DA with at least 16 GB of RAM and a 14-inch screen", the recommender becomes

a constraint solver over a product knowledge base. The system filters the catalog by the hard constraints and ranks the remainder by a utility function over the soft criteria. Knowledge-based systems dominate domains where preferences are rationally articulated (cars, real estate, financial products, B2B procurement) and almost vanish from domains where preferences are expressive (music, fashion, food).

1.4.5 Context-aware recommenders

Context-aware (CARS) systems extend the utility function to $f(u, i, c)$, where c encodes the situational variables: time of day, day of week, geographic location, device, weather, social companion. The intuition is that preference is conditional on context. A coffee at 8 a.m. is not the same item as a coffee at 11 p.m. A restaurant on a Tuesday lunch is not the same item as the same restaurant on a Saturday night.

Three implementations of CARS exist:

- **Contextual pre-filtering**: select the slice of the interaction matrix matching the current context, run a standard recommender on it.
- **Contextual post-filtering**: run a standard recommender, re-rank or filter the output by context.
- **Contextual modelling**: incorporate the context directly into the model (a tensor factorisation $R \approx P \times Q \times C$ on a user-item-context tensor).

1.4.6 Session-based and sequence-aware recommenders

These approaches model the order of interactions rather than their aggregate. A user's last three clicks often carry more information about what they want next than the previous six months of history. Recurrent neural networks (GRU4Rec [9]) and self-attentive models (SASRec [11], BERT4Rec) have become standard for this family. They are particularly useful when the same user behaves very differently across sessions: work browsing in the morning, leisure browsing in the evening, gift browsing in December.

1.4.7 Deep-learning-based recommenders

A broad and rapidly evolving class that includes:

- **Neural Collaborative Filtering** [7], which replaces the dot product in matrix factorisation by an MLP, allowing non-linear interactions between user and item embeddings.
- **Autoencoder recommenders**, which reconstruct the user's interaction row through a low-dimensional bottleneck, treating the reconstruction as the predicted preference.

- **Graph neural networks** on the bipartite user-item graph, which propagate preference signal through multi-hop neighbourhoods. LightGCN is the dominant example.
- **Large language model rerankers**, which take the user's history as text, the candidate items as text, and let the LLM produce a ranking. Costly and opaque, but increasingly used as a final-stage refinement on top of a cheaper candidate-generation layer.

These models are accurate but introduce three costs: training infrastructure, inference latency, and opacity. For a small, explainable, privacy-first system like ours the cost-benefit balance does not favour them. We discuss in Chapter 3 how we use a small slice of this family, a pretrained NLI transformer, only to label products, never to rank them.

1.5 Similarity measures

Similarity is the operational primitive on which every memory-based recommender, and every content-based recommender expressed as nearest neighbours, rests. We review the main measures and explain why we chose cosine.

1.5.1 Cosine similarity

For two non-zero vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^D$, cosine similarity is

$$\cos(\mathbf{a}, \mathbf{b}) \;=\; \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\,\|\mathbf{b}\|} \;=\; \frac{\sum_{i=1}^{D} a_i b_i}{\sqrt{\sum_{i=1}^{D} a_i^2}\,\sqrt{\sum_{i=1}^{D} b_i^2}}. \tag{1.4}$$

The output lies in $[-1, 1]$ in general, and in $[0, 1]$ when both vectors are non-negative (our case). Cosine measures the *angle* between vectors, ignoring magnitude. Two users with identical taste at different intensities are mapped to the same direction and are therefore considered identical by cosine.

1.5.2 Euclidean distance

The familiar metric,

$$d_2(\mathbf{a}, \mathbf{b}) \;=\; \sqrt{\sum_{i=1}^{D} (a_i - b_i)^2}. \tag{1.5}$$

It is sensitive to magnitude. Two users with the same taste pattern but different overall engagement levels are placed far apart by Euclidean even though their preferences are identical in shape. For recommendation, this is usually the wrong sensitivity. Cosine and Euclidean coincide when both vectors are L2-normalised. This is a useful fact that lets engineers pre-normalise once and then use whichever metric is cheaper to compute downstream.

1.5.3 Pearson correlation

$$\text{Pearson}(\mathbf{a}, \mathbf{b}) \;=\; \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2}\,\sqrt{\sum_i (b_i - \bar{b})^2}}. \tag{1.6}$$

Pearson is cosine after mean-centring. It is preferred for user-user CF over rating matrices because users differ in their baseline generosity: some give 4 to everything they like, some give 5 only to masterpieces. Centring removes that bias.

1.5.4 Jaccard and Tanimoto

For binary interaction sets A and B,

$$J(A, B) \;=\; \frac{|A \cap B|}{|A \cup B|}. \tag{1.7}$$

Used for implicit-feedback CF when the only signal is "user interacted" / "user did not interact". Tanimoto is the real-valued generalisation.

1.5.5 Why cosine for AlgerieShop IA

We chose cosine for three reasons:

1. **Magnitude invariance.** A user who is very engaged (large vector) and a user with the same taste but lower engagement (smaller vector) should see the same recommendations. Cosine guarantees that.
2. **Computational simplicity.** Once vectors are L2-normalised, cosine reduces to a dot product, which the JavaScript engine in the browser executes at SIMD-like speeds in modern V8.
3. **Composability with explanations.** The cosine numerator $\sum_i p_i q_i$ is a sum of per-dimension contributions, each of which can be surfaced to the user as "recommended because cooking $0.92 \times 0.95 = 0.87$ is the top contributor".

1.6 Evaluation metrics

A recommender that produces the best score on no dataset is useless. The question of *what* to measure is a research field of its own. We split the metrics into three families.

1.6.1 Predictive-accuracy metrics

For rating prediction the standard metrics are RMSE and MAE, already given in Equation (1.8):

$$\text{RMSE} = \sqrt{\frac{1}{|T|}\sum_{(u,i)\in T}(r_{u,i}-\hat{r}_{u,i})^2}, \quad \text{MAE} = \frac{1}{|T|}\sum_{(u,i)\in T}|r_{u,i}-\hat{r}_{u,i}|. \tag{1.8}$$

RMSE penalises large errors more than small ones. The Netflix Prize famously used RMSE as its single competition metric.

1.6.2 Ranking-quality metrics

A recommender is, in production, almost always a top-K list, not a per-item rating. Ranking-quality metrics evaluate the list directly.

Precision@K, Recall@K, F1@K.

$$\text{P@K} = \frac{|\text{relevant} \cap \text{top-}K|}{K}, \quad \text{R@K} = \frac{|\text{relevant} \cap \text{top-}K|}{|\text{relevant}|}, \quad \text{F1@K} = \frac{2\,\text{P@K}\,\text{R@K}}{\text{P@K}+\text{R@K}}. \tag{1.9}$$

Mean Average Precision (MAP). Averages the precision at every rank where a relevant item appears, averaged again across users. Sensitive to where in the top-K the relevant items appear.

Normalized Discounted Cumulative Gain (NDCG@K).

$$\text{NDCG@K} \;=\; \frac{1}{\text{IDCG}}\sum_{p=1}^{K}\frac{2^{\text{rel}_p}-1}{\log_2(p+1)}, \tag{1.10}$$

where rel_p is the relevance grade of the item at position p and IDCG is the score of the ideal ranking. NDCG is the dominant ranking metric in industry: it discounts relevance by rank, rewarding the system for placing the most relevant items at the top.

Hit Rate (HR@K). A binary check: was the held-out positive anywhere in the top-K? Simple, robust, the workhorse of session-based recommender evaluation.

1.6.3 Beyond-accuracy metrics

Accuracy alone is a poor proxy for user experience. The literature recognises five other axes [28].

- **Coverage.** The fraction of the catalog ever recommended across all users. A recommender that scores 0.92 NDCG@10 by recommending only the top 50 items to everyone has catalog coverage of 0.05. Long-tail merchants will not survive on such a recommender.
- **Diversity.** Intra-list dissimilarity. A recommendation list of ten thrillers is less diverse than a list of three thrillers, three comedies and four documentaries.
- **Novelty.** The bias away from items the user already knows. Recommending the obvious bestseller is accurate but useless: the user already knew about it.
- **Serendipity.** Unexpected but welcome recommendations. The hardest metric to define and the most prized when achieved.
- **Fairness.** Distributional concerns across producers (do small merchants get a fair share of impressions?) and consumers (does the recommender treat users of different demographic groups consistently?).

A recommender that scores 0.92 NDCG@10 but recommends the same fifty items to everyone is, in the long run, worse than a recommender that scores 0.85 NDCG@10 with broad coverage and verbal explanations.

1.7 The cold-start problem

The cold-start problem appears in three forms:

- **New-user cold start.** The user has just registered or has arrived without registering. The system has no history.
- **New-item cold start.** The product was just added to the catalog. No user has interacted with it.
- **New-system cold start.** The platform itself is new. No user-item interactions exist anywhere yet.

The principal mitigations are five.

1. **Onboarding questionnaires.** Ask explicit preferences before the first recommendation. Effective but tedious; abandonment is high in practice (drop-off above 40 % is common past the third question).
2. **Demographic priors.** Recommend based on age, gender or geography. Effective but ethically delicate, and often regulated. The GDPR places strong constraints on the legitimate use of demographic data for personalisation.

3. **Content-based bootstrapping.** Use item features to position the user in the feature space before any interaction [23]. The user's first interactions then shift them within that space. This is what our visual persona builder does at the conceptual level.

4. **Active learning.** Ask the user to rate a small, carefully chosen set of items, chosen to maximise information gain about the user's position in the latent space. Cinematch's seed-rating phase is an example.

5. **Cross-domain transfer.** Import a profile from another domain (music tastes used for podcast recommendation, reading habits used for film recommendation). Effective when legal and technical bridges between domains exist.

Our visual persona builder, described in Chapter 3, is a hybrid of mechanisms 3 and 4: items are presented as visual archetypes (one per dimension), the user picks the three or more they like, and the resulting persona is the average of their NLP-generated content vectors. No demographic data is asked, no email is required. We argue that the visual persona builder maximises the speed-to-relevance trade-off: a useful persona is produced in under thirty seconds.

1.8 Privacy considerations

Recommendation has historically been a server-side affair: profiles and interaction logs centralise inevitably in the platform's database. Three research directions push back against that default.

Federated recommenders. Model parameters are trained across devices without exchanging raw data. The user's history stays on the user's device; only aggregated gradients leave the device, ideally after differential-privacy noise has been added. Adoption is still limited by the infrastructure cost and by the mismatch between client compute and server compute.

Differential privacy. Noise is added to interaction logs or directly to the model parameters to prevent re-identification of any individual user from the published model. Adoption is growing, particularly in regulated jurisdictions.

On-device personalisation. The profile *and* the ranking computation both happen on the client. This is our architectural choice. It is the strongest privacy guarantee, since nothing leaves the device, at the cost of needing a recommender simple enough to run in a browser. With a 12-dimensional vector space, this constraint is easy to satisfy.

1.9 Explainability

A recommendation the user does not understand is a recommendation the user does not trust. The explainable-recommendation literature [28] has identified four broad strategies.

- **Item-based explanations.** "Recommended because it is similar to X, which you liked." Trivial for item-item CF, easy for CB.
- **User-based explanations.** "Recommended because users similar to you liked it." Natural for user-user CF.
- **Feature-based explanations.** "Recommended because it scores high on cooking, which is the dimension you score highest on." This is what our system does. It falls out of the cosine numerator for free.
- **Path-based explanations** over a knowledge graph. "Recommended because product X is by the same author as product Y, which you bought." Effective when a knowledge graph exists.

Feature-based explanations have a particular psychological strength: they describe the user to themselves. A user who is shown "you score 0.91 on cooking and 0.78 on wellness" has been given a self-portrait. The recommendation that follows is no longer an algorithmic edict; it is a consequence of a self-description the user can verify and correct.

1.10 Zero-shot NLP and Natural Language Inference

In classical natural language processing, mapping a product description to a set of predefined categories requires a labelled dataset and a training phase. When labelled data is scarce or the categories change frequently, this approach breaks down. Our system sidesteps it by framing product classification as a Natural Language Inference (NLI) task [27].

1.10.1 The NLI framework

Natural Language Inference involves determining whether a hypothesis H is true given a premise text P. The relationship is classified into three categories:

- **Entailment**: H is necessarily true if P is true.
- **Contradiction**: H is necessarily false if P is true.
- **Neutral**: the truth value of H cannot be determined from P.

To perform zero-shot classification, we turn the product description into the premise P and construct a template sentence for each target category as the hypothesis H. For

the description “Kaftan traditionnel en soie avec broderies fines” and the target category *Traditional Crafts*, the hypothesis becomes “This text is about traditional crafts.”

The NLI model estimates three probabilities,

$$\big(P(\text{Ent}),\ P(\text{Con}),\ P(\text{Neu})\big) = \mathcal{M}(P, H). \tag{1.11}$$

By applying a softmax over the *entailment* scores of all 12 candidate categories, we obtain a valid probability distribution that sums to 1. This distribution forms the semantic vector for the product.

1.10.2 Transformer architecture

NLI models are, at their core, transformer encoders [25]. A transformer encoder processes a token sequence through a stack of layers, each consisting of a multi-head self-attention block followed by a feed-forward block, with residual connections and layer normalisation. The self-attention operation,

$$\text{Attn}(Q, K, V) \;=\; \text{softmax}\!\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V, \tag{1.12}$$

allows every token to attend to every other token in the sequence, building context-sensitive representations.

A standard NLI model concatenates the premise and the hypothesis with a separator token, encodes the joint sequence, and classifies the contextualised representation of the special classification token into one of the three labels.

1.10.3 BERT and its descendants

BERT [5] introduced the masked-language-modelling pre-training objective: randomly mask 15 % of the tokens, ask the model to recover them. Pre-training on a large corpus then fine-tuning on a task became the dominant paradigm in NLP.

DeBERTa [8] improved on BERT in two ways: **disentangled attention**, in which the content and the relative position of each token are represented by separate vectors, and **enhanced mask decoder**, which incorporates absolute positions only at the prediction layer. The two changes yield consistent gains across the GLUE and SuperGLUE benchmarks. DeBERTa-v3 further introduces **gradient-disentangled embedding sharing** with an ELECTRA-style replaced-token-detection objective, which produces sharper representations at the same parameter budget.

1.10.4 Multilingual transformer: mDeBERTa-v3

The Algerian catalog is bilingual. We use `mDeBERTa-v3-base-mnli-xnli` [13], the multilingual derivative of DeBERTa-v3 distributed by Moritz Laurer on Hugging Face and served through the Transformers library [10]. Its properties:

- **Architecture**: disentangled-attention transformer with enhanced mask decoder, 12 layers, 768 hidden, 12 heads.
- **Size**: ∼280 M parameters, ∼1.1 GB on disk.
- **Training data**: MultiNLI (English NLI, [26]) and XNLI (its multilingual extension, covering 15 high-resource languages and transferring zero-shot to 100+, [4]).
- **Languages**: French, Arabic, English and 100+ more, a natural match for the Algerian catalog's de facto code-switching.
- **Inference cost**: ∼1 to 2 s per call on a modern CPU, ∼200 ms on a single GPU. Acceptable for the admin path (one-off labeling of a new product); too slow for live ranking, which is why we keep the NLP service off the request path entirely.

1.10.5 Limits of zero-shot

Zero-shot is not free of failure modes. The most common are:

- **Template sensitivity.** The hypothesis template matters. "This text is about cooking" and "This product is for cooking" do not always produce the same softmax.
- **Bias toward common terms.** The model has seen "cooking" far more often than "zellige" during pre-training. Common-vocabulary categories tend to over-attract probability mass.
- **Adversarial brevity.** A two-word description ("Tapis berbère") gives the model little to work with; longer descriptions produce sharper outputs.

We mitigate these by floor-clamping at $\varepsilon = 0.04$, keeping the admin slider overrides, and preferring richer descriptions when curating the seed catalog.

1.11 Matrix factorisation in depth

Although our system does not use matrix factorisation, the technique is so central to the recommender literature that a serious thesis cannot omit it. We give the mathematical formulation, the standard learning algorithms, and the regularisation choices that distinguish a robust factorisation from a brittle one.

1.11.1 The bilinear model

Given an interaction matrix $R \in \mathbb{R}^{N \times M}$ with N users and M items, matrix factorisation models each entry as the dot product of a user vector and an item vector,

$$\hat{r}_{u,i} \;=\; \mathbf{p}_u^{\top} \mathbf{q}_i \;=\; \sum_{k=1}^{d} p_{u,k}\, q_{i,k}, \tag{1.13}$$

where $\mathbf{p}_u \in \mathbb{R}^d$ and $\mathbf{q}_i \in \mathbb{R}^d$ are learned, and d is the latent-factor dimensionality (typically 16 to 256).

The simplest form has no bias terms; in practice, user and item biases are added,

$$\hat{r}_{u,i} \;=\; \mu + b_u + b_i + \mathbf{p}_u^{\top} \mathbf{q}_i, \tag{1.14}$$

where μ is the global mean rating, b_u is the deviation of user u from the mean (some users rate more generously), and b_i is the deviation of item i (some items are intrinsically popular). The biases absorb most of the variance that has nothing to do with taste matching.

1.11.2 Regularised least-squares objective

For explicit-feedback datasets the canonical objective is

$$\min_{\mathbf{P}, \mathbf{Q}, \mathbf{b}} \sum_{(u,i) \in \Omega} \left(r_{u,i} - \hat{r}_{u,i} \right)^2 \;+\; \lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 + b_u^2 + b_i^2 \right), \tag{1.15}$$

where Ω is the set of observed entries and λ is the L_2 regularisation strength.

1.11.3 Stochastic gradient descent

The standard learning algorithm steps over each observed entry $(u, i) \in \Omega$ and updates the parameters with the gradient of the squared error:

$$e_{u,i} \;\leftarrow\; r_{u,i} - \hat{r}_{u,i}, \tag{1.16}$$

$$b_u \;\leftarrow\; b_u + \gamma \left(e_{u,i} - \lambda\, b_u \right), \tag{1.17}$$

$$b_i \;\leftarrow\; b_i + \gamma \left(e_{u,i} - \lambda\, b_i \right), \tag{1.18}$$

$$\mathbf{p}_u \;\leftarrow\; \mathbf{p}_u + \gamma \left(e_{u,i}\, \mathbf{q}_i - \lambda\, \mathbf{p}_u \right), \tag{1.19}$$

$$\mathbf{q}_i \;\leftarrow\; \mathbf{q}_i + \gamma \left(e_{u,i}\, \mathbf{p}_u - \lambda\, \mathbf{q}_i \right), \tag{1.20}$$

with γ the learning rate (typically 0.005 to 0.05). Each pass over Ω is an *epoch*; 10 to 50 epochs are usually enough to reach convergence.

1.11.4 Alternating least squares

For implicit feedback, the SGD objective is replaced by a weighted regression on a confidence-augmented preference matrix. Hu, Koren and Volinsky (2008) formulated it as follows. Let $p_{u,i} \in \{0, 1\}$ indicate whether the user has interacted with the item, and $c_{u,i} = 1 + \alpha\, r_{u,i}$ a confidence weight derived from the raw interaction count $r_{u,i}$. The objective becomes

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{u,i} c_{u,i} \left(p_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i\right)^2 \; + \; \lambda(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2). \tag{1.21}$$

The objective is bi-convex: fixing $\mathbf{Q}$ makes it convex in $\mathbf{P}$ and vice versa. Alternating least squares (ALS) exploits this by solving the closed-form least-squares step for one factor matrix while holding the other fixed, then swapping roles. ALS parallelises trivially: each user row and each item column can be updated independently within a half-step.

1.11.5 Bayesian personalized ranking

A third learning objective, BPR (Rendle et al., 2009), reframes implicit-feedback factorisation as a pairwise-ranking problem. For each user u and each pair of items (i, j) where u interacted with i but not with j, the objective is to maximise the probability that the score for i exceeds the score for j:

$$\max \sum_{(u,i,j)} \log \sigma(\hat{r}_{u,i} - \hat{r}_{u,j}) - \lambda \, \|\boldsymbol{\Theta}\|^2, \tag{1.22}$$

where σ is the sigmoid function and $\boldsymbol{\Theta}$ denotes all learned parameters. BPR is robust on highly sparse datasets and remains a strong baseline.

1.11.6 Why we did not use matrix factorisation

Three reasons. First, our catalog is small (120 items) and the interaction log is synthetic, so there is not enough data for the latent factors to converge meaningfully. Second, factorisation is opaque: the latent dimensions have no intrinsic meaning, so the explanation layer would require a post-hoc reconstruction. Third, factorisation does not solve the new-item cold start, which is exactly the property we needed from our content-based core. We acknowledge factorisation here as the natural next step for a production deployment with real interaction volume.

1.12 Fairness in recommenders

Fairness has become a major axis of recommender-systems research in the last five years. We sketch the main directions.

1.12.1 Two sides of the platform

A recommender platform has at least two stakeholders: the *consumer* (the user receiving recommendations) and the *producer* (the merchant or content creator whose item is or is not recommended). Fairness concerns exist on both sides.

Consumer fairness. Different demographic groups should receive recommendations of comparable quality. A platform whose women users systematically see lower-quality recommendations than men users has a consumer-fairness problem. The metric varies: group-wise NDCG@K, group-wise satisfaction surveys, demographic-parity audits.

Producer fairness. Different merchants should receive comparable exposure for comparable products. A platform whose recommender systematically pushes the long tail into invisibility in favour of a few hubs has a producer-fairness problem. The metric is exposure distribution (how is the total impression volume divided across the catalog?).

1.12.2 Diversity-aware ranking

A common operational mitigation is to enforce intra-list diversity. The Maximal Marginal Relevance (MMR) reranking trades off relevance against diversity:

$$\mathrm{MMR}(i; S) \;=\; \lambda\,\mathrm{rel}(i) \;-\; (1-\lambda)\max_{j\in S}\mathrm{sim}(i,j), \tag{1.23}$$

where S is the set of already-selected items and λ weights relevance against diversity. MMR is cheap to apply on top of any candidate set.

1.12.3 Calibration

A calibrated recommender preserves, in its output distribution over genres / categories, the user's input distribution over genres / categories. A user who has clicked on 60 % cooking, 30 % gaming, 10 % travel should see roughly that mix in recommendations, not 100 % cooking. Calibration metrics (Steck, 2018) and reranking strategies exist for this purpose.

1.12.4 The Algerian context

Producer fairness is particularly important in a market with a long tail of small artisans. A recommender that systematically excludes the craft long tail in favour of mainstream electronics would cause harm to a structurally significant fraction of the merchant base. Our system does not currently model fairness explicitly, but the deliberately broad 12-dimensional taxonomy (spanning crafts, food and artisanal goods through the `art`, `cooking` and `home` dimensions) creates an architectural counter-pressure against that exclusion.

1.13 A/B testing methodology

A recommender whose effect cannot be measured is a recommender whose value is unknown. A/B testing is the operational methodology by which a production recommender is improved over time.

1.13.1 The basic experiment

The user population is randomly partitioned into two groups ("control" and "treatment"). The control group sees the existing recommender; the treatment group sees the new recommender. After enough exposure, a primary metric (typically some variant of click-through rate, cart-add rate, or gross-merchandise volume) is compared between the groups.

The partition must be stable: a user should see the same variant throughout the experiment. The partition must be balanced: the two groups should match on demographic and behavioural covariates before the experiment starts. The partition must be adequate in size: the difference in the primary metric must be large enough to reach statistical significance with the available traffic.

1.13.2 Sample-size estimation

For a binary primary metric (e.g. click-through), the standard sample-size formula is

$$n \;=\; \frac{2\,(z_{\alpha/2} + z_{\beta})^2\,\bar{p}(1-\bar{p})}{\delta^2}, \tag{1.24}$$

where $\bar{p}$ is the baseline click-through, δ is the minimum detectable effect, $z_{\alpha/2}$ is the critical value for the chosen significance, and z_{β} is the critical value for the chosen power. For a $1\,\%$ baseline, a $5\,\%$ relative lift, $\alpha = 0.05$, and $\beta = 0.20$, n is on the order of tens of thousands of users per arm.

1.13.3 Common pitfalls

Peeking. Stopping the experiment as soon as it shows significance inflates the false-positive rate. The decision boundary should be set before the experiment starts, not discovered along the way.

Network effects. If users interact with each other (reviews, ratings, sharing), control and treatment users may contaminate each other's experiences. Cluster-randomised designs address this.

Novelty effects. A new recommender is interesting because it is new, independent of whether it is better. Long experiments (several weeks) are needed to wash out novelty.

Underpowered tests. A test that does not have the sample size to detect the effect size of interest will fail to reject the null even when the alternative is correct. Power analysis must precede the experiment.

1.13.4 What we did not do

We did not run a real A/B test. Our seed catalog is too small, our user base is the team, and the deployment is a demo. The $\alpha = 0.7$ default is set by inspection rather than by experiment. A production deployment would expose α to an experimentation framework and iterate.

1.14 Multi-armed bandits and exploration

A/B testing assumes a fixed split of users into a fixed number of variants. A subtler question is: while we are testing, how do we allocate traffic? Sending half of all users to a probably-worse variant has an opportunity cost. The multi-armed bandit literature addresses this trade-off.

1.14.1 Epsilon-greedy

The simplest bandit, ε-greedy, plays the empirical-best arm with probability $1 - \varepsilon$ and a random arm with probability ε. It explores indefinitely, never converging.

1.14.2 Upper-confidence-bound (UCB)

UCB1 plays the arm that maximises

$$\bar{X}_a + \sqrt{\frac{2 \ln n}{n_a}}, \tag{1.25}$$

where $\bar{X}_a$ is the empirical mean reward of arm a, n is the total number of plays, and n_a is the number of plays of arm a. The second term is an exploration bonus that shrinks as the arm is played more.

1.14.3 Thompson sampling

Thompson sampling models a posterior over each arm's reward distribution and plays an arm proportionally to the probability that it is the best. For Bernoulli rewards, the natural posterior is a Beta distribution updated by the observed clicks and non-clicks. Thompson sampling is empirically robust and is the default bandit in many production recommenders.

1.14.4 Contextual bandits

A contextual bandit conditions the arm selection on a context vector, such as the user's features, the time of day, or the device. LinUCB and contextual Thompson sampling extend the basic algorithms with linear models over the context. The state-of-the-art for recommender exploration in 2024 to 2026 is a contextual Thompson sampler with deep-network context features.

1.14.5 Where bandits would help us

Two places in our system would benefit from a bandit. First, the α blending weight could be learned per user by a contextual bandit, with the reward being downstream cart-add rate. Second, when the persona builder is presented, the selection of which twelve archetypes to show could itself be a bandit problem, learning which archetypes are most informative about a new user's preferences.

1.15 Real-world recommender architectures

Production recommender systems at scale rarely operate as a single algorithm. They are usually structured as a *two-stage* or *multi-stage* pipeline. Understanding this structure clarifies where simpler systems, including ours, sit on the spectrum.

1.15.1 Stage one: candidate generation

The first stage reduces the catalog from millions of items to a few hundred. Speed dominates accuracy at this stage. The dominant approaches are:

- **Approximate nearest-neighbour (ANN) search** on pre-computed item embeddings. HNSW, IVF, ScaNN and FAISS are the reference libraries.
- **Inverted-index retrieval** on item attributes (the user has a strong cooking signal $\rightarrow$ fetch all items tagged with cooking).
- **Collaborative filtering candidates** based on user-similarity look-up tables.
- **Recently-popular candidates** from a global trending feed.

The candidate generators are typically run in parallel and their outputs unioned. The union is then de-duplicated and passed to stage two.

1.15.2 Stage two: ranking

The second stage takes the few hundred candidates and ranks them precisely. Accuracy dominates speed. The dominant approaches are:

- **Gradient-boosted trees** on rich user/item/context features (LightGBM, XGBoost). The Web-search industry default for two decades.
- **Deep neural rankers** (Wide & Deep, DCN, DLRM) that combine memorisation features with generalisation features.
- **Transformer rerankers** that take the candidate list as a sequence and produce a permutation.

1.15.3 Optional stage three: business-rule layer

A final layer applies hard business rules: deduplication across sellers, freshness filters, regulatory removals, paid placements, diversity constraints. The output of this layer is what the user actually sees.

1.15.4 Where AlgerieShop IA sits

AlgerieShop IA collapses all stages into one. The catalog is small enough (120 items, generalisable to a few thousand) that brute-force cosine over the entire catalog plays the role of both candidate generation and ranking. There is no business-rule layer yet. The hybrid CB+CF blend is a primitive form of multi-signal ranking. This collapse is deliberate: at the scale we operate, the two-stage architecture would add complexity without latency or accuracy gain.

The roadmap from here to a two-stage architecture is well understood. The first step is the `pgvector` migration discussed in Section 3.28: that replaces our brute-force scan with an HNSW-backed approximate-nearest-neighbour candidate generator. The second step is to add a learned ranker on top of that candidate set, trained on the interaction log once that log is large enough. The third step is the business-rule layer, which becomes necessary the moment the platform has paid placements or inventory constraints to respect.

1.16 Foundations of information retrieval

Recommender systems share a great deal of intellectual lineage with information retrieval. The two fields use the same vector spaces, the same similarity measures, and many of the same evaluation metrics. We sketch the bridge.

1.16.1 The vector-space model

A document or item is represented as a vector in a feature space. The classical text representation is the bag-of-words: each dimension is a term in the vocabulary, the value is a function of the term's frequency in the document. TF-IDF reweights the frequency to penalise terms that appear everywhere:

$$\text{tfidf}(t, d) \;=\; \text{tf}(t, d) \cdot \log \frac{N}{\text{df}(t)}, \tag{1.26}$$

where $\text{tf}(t, d)$ is the count of term t in document d, $\text{df}(t)$ is the number of documents containing t, and N is the total document count.

1.16.2 BM25

BM25 is the de facto refinement of TF-IDF used by search engines today:

$$\text{BM25}(t, d) \;=\; \text{IDF}(t) \cdot \frac{\text{tf}(t, d)\,(k_1 + 1)}{\text{tf}(t, d) + k_1\big(1 - b + b\,|d|/\text{avgdl}\big)}, \tag{1.27}$$

with parameters k_1 (typically 1.2 to 2.0) and b (typically 0.75) controlling term-frequency saturation and document-length normalisation. It needs few parameters and still dominates classical text retrieval.

1.16.3 Dense retrieval

Modern retrieval increasingly uses dense vector representations learned by transformers (DPR, Contriever, BGE, E5). A query and a document are mapped to dense vectors in a shared space; retrieval becomes nearest-neighbour search. Dense retrieval is now the default in production systems and has direct kinship with recommender systems: a recommender is essentially a retrieval system where the "query" is the user.

1.17 Challenges that frame our design

We close the chapter with a synthesis of the challenges the literature has identified and that our system must answer concretely.

1. **Cold start**, solved by the visual persona builder (content-based bootstrapping plus active learning).
2. **Explainability**, solved by the per-dimension contribution breakdown (feature-based explanation).
3. **Privacy**, solved by on-device personalisation (the persona never leaves the browser).

4. **Multilingual content**, solved by multilingual zero-shot NLI (no per-language training).
5. **Filter bubble**, mitigated by the hybrid blend with a popularity signal, which leaks crowd diversity into a pure-CB ranking.
6. **Sparsity and small catalogs**, mitigated by the content-first design (CB does not need a dense matrix).
7. **Computational cost**, mitigated by the 12-dimensional space (brute-force KNN is $\mathcal{O}(N \cdot 12)$ per query) and by keeping the heavy NLP off the request path.

1.18 Conclusion

This chapter laid out the scientific landscape of recommender systems: the formal problem, the families and their trade-offs, the similarity measures, the metrics, the cold-start and privacy questions, the explainability strategies, and the zero-shot NLP machinery on which our content vectors rest. The next chapter zooms into the market the system is built for, the Algerian e-commerce ecosystem, whose linguistic, economic and operational specificities will constrain every architectural choice in Chapter 3.

Chapter 2

The Algerian E-Commerce Ecosystem

2.1 Introduction

A recommender system is technical machinery; an e-commerce platform is a business with operations, customers, regulators and competitors. This chapter establishes the functional and commercial environment in which AlgerieShop IA exists. Before designing a recommender, one must understand the system it sits inside: the e-commerce order management pipeline, the data it generates, the technologies that already automate parts of it, and the specific challenges that justify adding a personalization layer at all.

The chapter then narrows to the Algerian e-commerce market in particular, whose linguistic, demographic and economic specificity is the reason every existing recommender designed elsewhere is, at best, a partial fit. We describe the major platforms, the payment landscape, the delivery infrastructure, the bilingual catalog reality, the regulatory environment, the implications all of these have for the recommender we will build, and the best practices that have emerged from a decade of observed failure and success in the region.

2.2 Foundations of e-commerce

2.2.1 Definition and scope

E-commerce is the buying and selling of goods and services over electronic networks, predominantly the Web. The definition is deliberately broad, because the field has expanded from its early catalog model into a much richer set of patterns:

- **Marketplaces.** Many sellers, one platform. Amazon, eBay, Jumia, Ouedkniss.
- **Direct-to-consumer brands.** One brand, one platform. Casper, Glossier, and increasingly Algerian artisan boutiques that bypass intermediaries.
- **Subscription boxes.** Recurring shipments of curated goods. Common in cosmetics and gourmet food.
- **Second-hand resale.** Vinted, Vestiaire, Ouedkniss for non-new goods.

- **Hyperlocal grocery delivery.** Yassir Express, Glovo, Talabat in their grocery arms.
- **B2B procurement portals.** Alibaba, Faire, and the institutional procurement portals of Algerian state enterprises.
- **Cross-border marketplaces.** AliExpress, Shein, Temu. These have a large but informal Algerian footprint.

Behind every transaction sits an *order management pipeline* whose efficiency, more than any single algorithm, determines whether a platform is usable. A good recommender is wasted on a platform whose checkout fails, whose inventory data is wrong, or whose shipping estimates are unreliable. The recommender we build assumes a working pipeline.

2.2.2 A brief history of online commerce

The history of e-commerce can be split into four phases relevant to the Algerian context.

1995 to 2002: catalog Web. Amazon launches in 1995 as an online bookstore. eBay launches the same year. The period is dominated by static catalogs, slow checkout, desktop-only browsing and credit-card-only payment. Algerian e-commerce in this period is essentially absent.

2003 to 2010: marketplace consolidation. Amazon expands into electronics, then everything. eBay becomes the dominant second-hand venue. PayPal becomes the dominant international payment rail. The first social-driven commerce sites appear. Algerian Internet penetration reaches 10 % by 2010, but online payment remains marginal.

2011 to 2018: mobile commerce. Smartphones become the dominant device. App-driven commerce overtakes Web-driven commerce in some categories (food delivery, ride-hailing, grocery). Jumia (founded 2012 in Lagos) becomes the first pan-African e-commerce platform with an Algerian operation.[1] Ouedkniss, the dominant Algerian classifieds platform, launches its dedicated marketplace product.

2019 to today: super-apps and same-day delivery. Ride-hailing platforms (Yassir, Heetch) become super-apps, integrating food, groceries, parcels, payments. Cash on delivery remains dominant in Algerian e-commerce despite the rise of card and wallet payment. The COVID-19 pandemic accelerates online buying behaviour across all demographics.

2.2.3 Four operating principles

A modern e-commerce stack rests on four principles whose violation is visible to the user.

[1] Jumia DZ launches in Algeria in 2014.

1. **Traceability.** Every order moves through identifiable states: received, validated, prepared, shipped, delivered, returned. Every state transition is logged, ideally with a timestamp and an actor, so that any later question can be answered without forensic effort.

2. **Inter-departmental collaboration.** A single order touches sales, inventory, payments, fulfilment and customer support. None of these departments can act in isolation; the system's central data store is the contract between them.

3. **Automation.** Repetitive operations (price calculation, tax application, address validation, stock decrement) are deterministic and must be machine-executed. Human attention is reserved for exceptions: unusual returns, payment disputes, fulfilment incidents.

4. **Flexibility.** The model must absorb new payment methods, shipping providers, product categories and marketing rules without rewrites. A rigid order schema is a future bottleneck disguised as a present simplification.

Table 2.1. A minimal order tracking view as seen from an admin dashboard.

Order	Customer	Created	Items	Amount (DA)	Status
ORD-001	Yacine B.	2025-04-12	2	8 500	In preparation
ORD-002	Sara M.	2025-04-13	1	3 200	Shipped
ORD-003	Karim L.	2025-04-14	4	17 300	On hold
ORD-004	Imâne K.	2025-04-15	1	5 400	Delivered
ORD-005	Anis T.	2025-04-15	3	12 100	Returned
ORD-006	Mohamed S.	2025-04-16	2	6 800	In preparation
ORD-007	Nesrine D.	2025-04-16	5	24 500	Shipped
ORD-008	Walid B.	2025-04-17	1	11 900	Delivered

2.3 The order management process

The classical pipeline can be decomposed into five stages, each of which generates data that a recommender system can, in principle, learn from.

2.3.1 Order reception

An order originates from a Web form, a mobile app, an EDI feed, or occasionally a phone call. Whatever the channel, the system normalises it into a record containing the customer's identifying data, an itemised basket, the chosen shipping address, and the payment intent. In our project, the reception layer is implemented through the `CartContext` (client-side

basket persisted in `localStorage`) and a checkout page that serialises the basket into a Supabase row.

The cart is intentionally local: the user should not be forced to create an account in order to add an item, a small UX choice with measurable consequences on conversion. The literature on conversion-rate optimisation suggests that mandatory account creation costs between 15 % and 25 % of checkout starts. For a market like the Algerian one, where the customer is often discovering the platform for the first time, that cost is unacceptable.

2.3.2 Validation and verification

Before the order is committed, the system must check three things: the availability of every line item, the capacity of the fulfilment chain, and the feasibility of the promised delivery window. A failure on any of these triggers a negotiation with the customer: out-of-stock substitution, deferred delivery, partial cancellation. This stage is where most of the operational risk of an e-commerce platform materialises.

For a recommender, validation matters indirectly: an item that is chronically out of stock should be down-weighted in the ranking, because recommending it produces frustration. Our current system does not implement this rule, but it is a one-line addition once inventory snapshots are joined into the recommendation API.

2.3.3 Production and preparation planning

For physical goods, this stage allocates the order to a warehouse, picks the items, packages them, and prints labels. For digital goods, it provisions licences or downloads. In both cases, the planning component prioritises orders, balances workloads, and respects deadlines. The scheduling problem this raises is itself a classical subfield of operations research; we acknowledge it but do not work on it directly in this thesis.

2.3.4 Execution and shipping

Execution is the realisation of the plan: the picker fetches the items, the packer boxes them, the carrier picks the parcel up. Execution is typically followed by quality control and exception handling. A real-time tracking number is generated and exposed to the customer.

The shipping leg in Algeria is dominated by three mechanisms: the postal service (Algérie Poste, slow but ubiquitous), private couriers (Yalidine, ZR Express, Maystro), and the same-day networks of the super-apps (Yassir Express, Glovo, when available). Each has different cost, speed and coverage characteristics, and a recommender that surfaces only items the customer's preferred carrier can deliver to their wilaya is more useful than one that does not.

2.3.5 Delivery and after-sales

The package reaches the customer. The order transitions to a final state: delivered, refused, or returned. Post-delivery, the platform solicits feedback (rating, review), processes returns, and records the customer's behavioural trail for analytics and, crucially for us, for recommendation.

The cash-on-delivery dynamic deserves a note. In Algeria, where COD remains dominant, the platform absorbs the risk of last-mile refusal: the customer can reject the parcel upon presentation, and the platform pays the carrier for the return trip regardless. This makes the *quality* of the recommendation, not just its click-worthiness, a financial concern. A click that becomes a refused parcel costs the platform money.

2.4 Personalization in e-commerce

Personalisation is the deliberate adaptation of a service to a specific user. In e-commerce it takes many forms:

- personalised home pages,
- per-user search rankings,
- personalised email digests (the "what you might like" weekly mail),
- targeted promotions (a discount code keyed to the user's segment),
- dynamic pricing (controversial, regulated in some jurisdictions),
- individualised cross-sell suggestions at the product page and at the cart.

The common thread is that the user is no longer treated as an average. Each interaction with the platform produces a small amount of evidence about the user, and a personalised system folds that evidence back into the next page rendered.

The two questions a personalisation system must answer are therefore: *what evidence do we trust?* and *how fast do we let it move the ranking?*

2.4.1 Why personalisation matters

- **Cognitive economy.** A user who is shown the right three items does not have to scroll through three hundred.
- **Inventory rotation.** A personalised page can promote the long tail of the catalog that a global popularity sort would never expose. This matters particularly for marketplaces with thousands of small sellers.

- **Loyalty.** A user who feels understood returns. Repeat purchase is more profitable than acquisition.
- **Competitive parity.** Personalisation has become a baseline expectation; its absence is a comparative defect against any platform the user has already experienced.

2.4.2 Why personalisation is hard

- **Sparse evidence.** Most users interact with most products zero times.
- **Cold start.** New users and new products have no evidence at all.
- **Drift.** Tastes change. A model fitted last quarter may already be wrong.
- **Bias.** A personalised system that learns only from past behaviour reinforces past behaviour, including its mistakes.
- **Privacy.** The evidence is, by construction, sensitive.
- **Explainability.** A recommendation the user does not understand is a recommendation the user does not trust.

2.5 Technological tools

Modern e-commerce platforms rarely build the pipeline from scratch. They assemble it from well-defined categories of software, summarised in Table 2.2.

Table 2.2. Common categories of software in the order management stack.

Category	Role	Representative vendors
ERP	Integrated resource management	SAP, Odoo, NetSuite
CRM	Customer relationship management	Salesforce, HubSpot, Zoho
WMS	Warehouse management	Manhattan, SAP EWM, Logiwa
PIM	Product information management	Akeneo, Pimcore, Plytix
OMS	Order management system	IBM Sterling, Kibo, Manhattan
Recommender	Personalisation layer	*Our work*, Algolia, Amazon Personalize
Search	Catalog search and filtering	Algolia, Elasticsearch, Typesense
Payments	Settlement and reconciliation	Stripe, Adyen, CIB Algeria
Wallet	Mobile wallet	BaridiMob, CIB e-paiement, Yassir Pay
Analytics	Behavioural and conversion tracking	GA4, Plausible, Mixpanel, PostHog
Customer support	Tickets and chat	Zendesk, Intercom, Freshdesk
Marketing automation	Campaigns and segmentation	Klaviyo, Braze, Customer.io

Our own stack maps cleanly onto this taxonomy. Supabase plays the role of a lightweight ERP/PIM/OMS hybrid (one Postgres database for products, interactions and orders),

browser local-storage plays the role of an extremely local CRM (the user's persona and cart never leave the device), and the present work plays the role of the recommender layer. A full production deployment would add a real analytics pipeline, a customer-support system, a payment gateway, and a delivery integration next to these components, not replace them.

2.6 The Algerian e-commerce landscape

Algerian e-commerce is real, growing, and structurally different from the European or North American markets that produce most of the academic literature on the field. Understanding those differences matters for the design decisions we make later.

2.6.1 Size and trajectory

Algeria is a country of approximately 45 million inhabitants, with a median age in the late twenties and an Internet penetration above 70 %. Smartphone penetration exceeds 80 % among the urban population aged 18 to 40. E-commerce as a share of retail remains in single digits. Estimates from 2024 placed it between 2 % and 4 % of total retail, yet the category is growing at double-digit annual rates, accelerated by the post-pandemic shift in buying habits and by the maturing of local logistics networks.

The market structure is fragmented. There is no Amazon equivalent; the dominant horizontal platforms are Jumia DZ and Ouedkniss, neither of which exercises monopoly power. Vertical platforms thrive in food delivery (Yassir Express, Glovo), ride-hailing (Yassir, Heetch, Temtem), pharmacy (local pharmacy networks), and second-hand goods (Ouedkniss).

2.6.2 Major platforms

Jumia DZ. The Algerian operation of the pan-African Jumia group. General marketplace, electronics, fashion, household. Originally launched in 2014. Inventory is mixed: managed inventory (Jumia ships from its own warehouse) and marketplace inventory (third-party sellers).

Ouedkniss. The historical Algerian classifieds platform, also operating a marketplace product. Covers automobiles, real estate, employment, services and a general goods marketplace. Dominates the second-hand segment.

Yassir Express. The grocery and goods delivery arm of Yassir, the Algerian super-app. Last-mile delivery in major cities, integrated payment, hyperlocal inventory.

Temtem. A ride-hailing and delivery super-app.

Local artisan boutiques. A long tail of direct-to-consumer artisan brands operating on Instagram, Facebook, and increasingly on Shopify-style standalone stores. Cumulatively significant.

Pharmacy and health platforms. Several pharmacy networks operate digital ordering products, sometimes integrated with delivery.

Cross-border imports. AliExpress, Shein and Temu have a large but informal footprint, with customs-cleared parcels arriving via direct-to-consumer international shipping. These are outside the regulated domestic e-commerce circuit but compete for the same attention.

2.6.3 Payment landscape

The Algerian payment landscape has three layers.

Cash on delivery (COD). Still the dominant payment method by volume in domestic e-commerce. The customer pays the carrier on parcel handover. Estimates place COD at between 60 % and 80 % of orders depending on the platform. COD is convenient for the customer but expensive for the platform: refusal rates of between 5 % and 15 % are common, and each refused parcel triggers a return trip that the platform absorbs.

Bank cards (CIB e-paiement). The inter-bank payment system, CIB, enables card payment on local platforms. Adoption is growing but remains a minority share, partly because card issuance and usage habits are still maturing.

Mobile wallets. BaridiMob, the wallet of Algérie Poste, is the most widely adopted mobile payment mechanism. Various super-app wallets (Yassir Pay, others) extend the landscape, particularly within their parent super-app.

Foreign cards. International Visa and Mastercard are usable in some channels but not all, and foreign-currency settlement is subject to capital controls. For purely domestic e-commerce, this is a marginal mechanism.

2.6.4 Delivery infrastructure

Three delivery rails coexist.

Algérie Poste. The historical postal operator. Covers the entire territory, including remote wilayas. Slower than private couriers, cheaper, and the default for cash-on-delivery on smaller platforms.

Private couriers. Yalidine, ZR Express, Maystro, and a long tail of regional carriers. Faster than the post, more expensive, common on Jumia and private merchants.

Super-app last-mile. The same-day fleets of Yassir Express, Glovo and the like. Constrained to major cities but the only realistic option for fresh goods (groceries, meals).

2.6.5 The bilingual catalog reality

This is the single most consequential specificity of the Algerian context for a recommender. Product descriptions mix French and Arabic, often inside the same paragraph. Brand names are written in Latin script. Generic terms are written in either script, sometimes both. The customer's search query can be in either language and is sometimes transliterated.

A monolingual recommender systematically under-represents products written in the "wrong" language. A French-only NLP labeler will fail to extract meaningful features from an Arabic product description; an Arabic-only labeler will fail symmetrically. Only a multilingual model, ideally one trained on cross-lingual NLI as `mDeBERTa-v3-base-mnli-xnli` is, gives consistent treatment.

2.6.6 Traditional crafts and the local long tail

A large fraction of the Algerian catalog consists of products that have no direct equivalent in Western e-commerce: kaftan, burnous, abaya, miel de Kabylie, dattes Deglet Nour, zellige, poterie de Kabylie, derbouka, oud, chèche touareg, savon noir, huile d'argan, henné. These items have several common features:

- **Culturally specific.** They carry meaning beyond utility (a kaftan is wedding attire, not just a dress).
- **Visually distinctive.** A photograph carries more information about the item than the description.
- **Textually under-described.** Descriptions are often one or two short sentences, sometimes a bilingual title only.
- **Absent from training corpora.** Western pretrained models have seen these terms rarely if at all.

Together, these properties mean that a generic Western recommender will under-rank the entire traditional-crafts segment. A recommender for the Algerian market must take this seriously by design.

2.6.7 Mobile-first browsing

A majority of sessions on Algerian e-commerce platforms originate from mid-range Android devices, often on imperfect networks. This rules out a heavy client-side machine-learning runtime and motivates our choice of a tiny in-browser KNN engine over a downloaded neural model. A 12-dimensional vector fits in tens of bytes; the cosine loop over a 5 000-item catalog completes in tens of milliseconds on a low-end mobile CPU.

2.6.8 Trust dynamics

Trust is a specific issue in this market. With cash-on-delivery dominant, the customer can refuse the parcel and pay nothing, so the merchant's incentive is to build a brand that the customer trusts *before* the parcel arrives. Reviews, ratings, and visible explanations of recommendations all play into that trust. Black-box recommendations have a much shorter patience window here than in markets with mature delivery guarantees.

2.7 Comparative perspective: the MENA region

Algeria does not exist in isolation. The Middle East and North Africa (MENA) region offers useful comparison points, each market with its own maturity and idiosyncrasies.

2.7.1 Morocco

Morocco is structurally similar to Algeria: bilingual (Arabic / French), with a sizeable Berber-speaking minority; cash on delivery dominant; and a similar catalog mix of traditional crafts and mainstream goods. The Moroccan platform Jumia MA is, like Jumia DZ, the local arm of the pan-African group. Hmall, Sefamerca and several Shopify-style merchant networks complete the landscape. Card payment penetration is slightly higher than in Algeria, helped by an earlier rollout of the Moroccan interbank payment infrastructure.

2.7.2 Tunisia

Tunisia's smaller market is dominated by Jumia TN and Mytek (electronics). The French-language share is higher than in Algeria, reflecting the linguistic balance of the country. Card payment is more common; cash on delivery is still the largest single mechanism but its dominance is less absolute. The post-2011 surge in entrepreneurial activity has produced a long tail of niche direct-to-consumer brands.

2.7.3 Egypt

Egypt is the largest MENA e-commerce market by volume. Souq.com, acquired by Amazon in 2017 and rebranded as Amazon.eg, is the regional reference. Noon, the Emirati challenger, operates a strong Egyptian leg. The market is bigger, more competitive, and structurally Arabic-first, a useful counterpoint to the Algerian French-Arabic balance. The cash-on-delivery share is still significant but lower than in Algeria.

2.7.4 Saudi Arabia and the UAE

The Gulf markets are the regional leaders in card-and-wallet adoption, in same-day delivery, and in ride-hailing super-apps. Noon, Amazon.sa, Amazon.ae, Carrefour and a long tail of premium boutiques compete for a customer base with higher purchasing power and higher digital literacy than the Maghreb average. These markets are interesting for AlgerieShop IA as a glimpse of where the Algerian market is heading, with a five- to ten-year lag.

2.7.5 Lessons for Algerian e-commerce

The MENA comparison yields three lessons. First, the French / Arabic bilingual balance is unique to the Maghreb; Gulf and Egypt are predominantly monolingual Arabic. A multilingual NLP backbone built for Algeria will need only minor adaptation to serve Morocco and Tunisia. Second, the cash-on-delivery dominance is a Maghreb feature, not a MENA feature; as electronic payment grows in Algeria, the refusal-rate pressure on recommenders will decrease. Third, the traditional-crafts long tail is structurally important in all three Maghreb countries; a recommender that takes it seriously has a regional rather than a national addressable market.

2.8 Telecom and Internet infrastructure

The technical environment in which an e-commerce platform operates is shaped by the telecom infrastructure underneath it. Three observations matter.

2.8.1 Mobile-network coverage

4G coverage is now near-universal in urban Algeria and extends to most semi-urban areas. 5G has been licensed and is rolling out in the largest cities. Coverage gaps remain in remote wilayas, where 3G or even 2G may be the only available technology. A recommender that loads multi-megabyte assets on every page is unusable in those gaps.

2.8.2 Mobile-data pricing

Data plans are inexpensive in absolute terms but constrained in volume. A typical prepaid bundle offers 5 to 20 GB per month, which is enough for routine browsing but punishing for video-heavy or asset-heavy applications. The pressure to ship lightweight pages is therefore not only a latency question; it is a cost question for the user.

2.8.3 Three operators

Algerian mobile telephony is dominated by three operators: Mobilis (state-owned), Djezzy (privately owned) and Ooredoo (privately owned). All three offer comparable 4G service in urban areas. Operator-specific bundles sometimes include free or reduced-tariff access to certain national platforms; this is a marketing channel that an e-commerce platform with the right partnerships can exploit.

2.8.4 Implications for AlgerieShop IA

We ship the entire recommender to the client because the client-server round trip is expensive in a constrained network. The persona vector and the cosine engine together weigh less than ten kilobytes, well within the budget of any modern mobile-data plan. The NLP service, heavier by orders of magnitude, runs on the admin path only, never on a customer device.

2.9 Social commerce

A significant fraction of Algerian commerce happens on social platforms before it ever reaches an e-commerce site. Three patterns are worth describing.

2.9.1 Instagram and Facebook

Many small Algerian merchants operate primarily on Instagram or Facebook, with no separate website. The product catalog is the brand's feed; the checkout flow is a direct-message conversation; payment is on delivery. This pattern is particularly common in fashion (small kaftan ateliers), beauty (artisanal cosmetics) and gifts.

For a horizontal platform, this means a substantial part of the catalog the customer expects to find is not on the platform at all. The platform competes with other platforms and with the entire informal social-commerce economy.

2.9.2 TikTok and live commerce

TikTok has, in the last two years, become a major discovery channel. Algerian creators present products in short videos; the audience comments asks for prices; sales are completed through direct messages. The audience is younger and the conversion is faster than on Instagram. Live commerce, where a creator hosts a real-time stream during which viewers purchase, is emerging but not yet dominant.

2.9.3 Implications

A horizontal platform's job is to take demand that originates on social and convert it on the platform. This means search must be reactive (the customer who has seen a product on Instagram is searching with a specific name or image in mind), recommendations must acknowledge that the customer arrived with a context the platform did not generate, and the catalog must be complete enough that the search-found product is actually there.

2.10 Seasonal patterns

Algerian e-commerce has pronounced seasonality that differs from Western markets.

Ramadan. The lunar month of fasting transforms consumer behaviour for thirty days. Browsing time shifts to the evening (after iftar); the catalog mix shifts toward food, hospitality, gifts and religious apparel; delivery times stretch as carrier networks adapt. A recommender that does not acknowledge Ramadan will under-rank items the customer is most likely to buy during the month.

Eid al-Fitr and Eid al-Adha. The two festivals create concentrated demand peaks for gifts, clothing (particularly traditional attire), and hospitality goods. Lead times for made-to-order items become critical; a kaftan ordered too late will not arrive in time.

Back-to-school (September). Stationery, electronics, school uniforms, books. A predictable weekly spike in late August through mid-September.

End-of-year. Less dominant than in Christian markets, but still a measurable spike on electronics-and-gifts driven by the secular new year and by global discount events that have spread to the region.

Implications. A production recommender would ingest the seasonal context as a feature. Our system does not currently; it ranks by a stationary persona. This is acceptable for a

thesis demonstration but would be a measurable loss in a real deployment during Ramadan or Eid.

2.11 The Algerian startup ecosystem

A short note on the institutional environment.

The 2020 *startup label* (label start-up), granted by the National Agency for the Development of Investment, created a formal category of recognised startups with associated tax benefits and access to public funding. Algeria Venture, the public early-stage fund, has co-invested in dozens of digital startups. Private accelerators (Algiers Innovation Group, the incubator at the École nationale supérieure d'informatique) provide seed-stage support. The ecosystem is small relative to Cairo or Casablanca but is structured and growing.

For a project like AlgerieShop IA, the institutional environment matters in three ways. First, the legal status of personal-data processing is well defined under Law 18-07, which gives the on-device-persona design a clear compliance story. Second, public-procurement channels exist for platforms that serve traditional-crafts merchants, opening one potential go-to-market path. Third, the startup label streamlines hiring of recent graduates, which is relevant given that the team behind a thesis is often the team that becomes a startup.

2.12 Case studies: two Algerian platforms

To make the market concrete, we sketch two of the platforms whose patterns most directly shaped our design choices. The observations are based on public information and on our own use of the platforms as customers; we make no claim to insider knowledge of their internal architectures.

2.12.1 Jumia DZ

Jumia DZ is the Algerian arm of the pan-African Jumia group, launched in Algeria in 2014. It is a horizontal marketplace covering electronics, fashion, household, beauty and a long tail of smaller categories. The user experience leans heavily on the sponsored-listing model: top placements in category pages are typically paid, and the recommendation widgets on the home page mix algorithmic and merchandised content.

The platform's recommender exposes "you may also like" carousels on product pages and a "recommended for you" widget on the home page after the user has logged in. The cold-start state of that widget is, from observation, a generic best-seller list. This is the classical cold-start failure mode that AlgerieShop IA's persona builder explicitly addresses.

The catalog is mostly in French with some Arabic. Search supports both languages but auto-complete is occasionally inconsistent across them. Reviews and ratings are sparse

on long-tail items. Cash on delivery is the dominant payment method, with bank-card payment offered.

The lesson we drew from Jumia DZ for our own design: a flat "recommended for you" widget without a cold-start mechanism quickly becomes generic. The visual persona builder is the cheapest possible fix.

2.12.2 Ouedkniss

Ouedkniss is the dominant Algerian classifieds platform, with a secondary marketplace product. Its catalog is even more heterogeneous than Jumia's: cars, real estate, employment, used goods, services, and new merchandise from small merchants all share the same listing format. The platform is bilingual by necessity: many listings are written in French, many in Arabic, and many in a mix.

Ouedkniss does not run a recommender in the classical sense; the default home-page sort is recency, with a search-driven discovery flow as the primary interaction pattern. The category taxonomy is extensive and curated, but listings can be miscategorised by their authors. The platform's value to a recommender researcher is, paradoxically, that the absence of personalisation reveals how much the user has to do manually.

The lesson we drew from Ouedkniss: catalog heterogeneity matters. A recommender designed only for "products" will fail on the half of the catalog that is cars or apartments. Our 12-dimensional taxonomy is deliberately consumer-goods oriented; extending it to heterogeneous verticals such as vehicles or real estate would require additional dimensions, which we note as future work.

2.13 Logistics in detail

The last-mile leg in Algeria is more constraining than in markets with mature carrier networks. Three observations are worth making explicitly.

2.13.1 Wilaya-level coverage

A wilaya is the Algerian administrative subdivision; the country has 58 wilayas of widely varying urbanisation. Carrier coverage correlates with urbanisation: a wilaya like Alger or Oran has near-universal same-day or next-day private-courier coverage, while a wilaya like Tindouf or Illizi may rely entirely on Algérie Poste with multi-day delivery windows.

A truly market-aware recommender would weight items by the likelihood that the carrier preferred by the customer can deliver within the promised window. We did not implement this rule; we flag it as a one-line addition in the recommendation API.

2.13.2 Returns and refusals

The cash-on-delivery dynamic creates a return rate higher than in card-paid markets. Reported refusal rates of 5 to 15 % are common. Each refused parcel costs the merchant the round-trip courier fee, the cost of restocking the item, and the opportunity cost of the rejected order.

A recommender that minimises false positives, meaning items the user clicks on but does not actually want, has direct revenue implications in this market. Explainability helps: an item the user understands has been recommended for a reason is less likely to be refused on arrival.

2.13.3 Fulfilment for traditional goods

Many traditional Algerian goods are made-to-order: a kaftan is sewn after the order is placed, a couscoussier is hand-finished on demand. The lead time can be days or weeks. The catalog needs to expose this lead time clearly, and the recommender needs to be aware of it when ranking against time-sensitive items.

2.14 Consumer behaviour patterns

Three behaviour patterns deserve explicit naming.

Discovery via social media. A significant fraction of purchase journeys begin on Instagram, Facebook or TikTok, not on the e-commerce platform itself. The customer sees a product in a post or a story, then searches for it on the platform. The platform's job is to surface it within the first three search hits.

Window shopping. Long browsing sessions without a purchase are common, particularly for high-ticket items (electronics, furniture). The behaviour is similar to physical window shopping: the customer is gathering information, not yet ready to commit. A recommender that interprets the long session as strong interest in a particular item may over-fit.

Group purchase. Family purchases (a smartphone for a sibling, a household appliance shared between roommates) are common. The single user identifier behind the session is sometimes shopping for someone else, which adds noise to any personalisation signal that does not distinguish "for me" from "for the family".

A recommender that ignores these patterns will not fail spectacularly, but it will underperform a recommender that acknowledges them. Our system does not currently model any of the three; we flag this as future work.

2.15 Regulatory landscape

E-commerce in Algeria is governed by several overlapping legal instruments.

Law 18-05 on electronic commerce (May 2018). The foundational text. Establishes the rights and obligations of e-commerce providers, the requirements for electronic signatures, the rules on advertising, the prohibition on cross-border purchase without explicit authorisation, and the consumer-protection regime.

Law 18-07 on the protection of personal data (June 2018). The Algerian data-protection law, broadly modelled on GDPR principles. Establishes the conditions for lawful processing of personal data, the rights of data subjects (access, rectification, opposition), and the obligation of the controller to declare processing activities. The law applies to any processing on Algerian territory regardless of the controller's nationality.

Consumer protection (Law 09-03 of 2009). The general consumer-protection law, which applies to distance contracts including e-commerce. Establishes the seven-day cooling-off period for distance contracts.

Cybercrime (Law 09-04). Criminalises the common cyber-offences, with provisions relevant to fraud-prevention features of e-commerce platforms.

For our recommender, the relevant constraints are: the data-protection regime (a strong argument for the on-device design we adopt), the consumer-protection regime (an argument for explainability so the recommendation is not a misleading commercial practice), and the absence of hard restrictions on algorithmic ranking specifically.

2.16 Implications for the recommender

The market specificities translate into a small set of hard design constraints:

- **Multilingual NLP is not optional.** Whatever model labels a product must handle French and Arabic with comparable quality. This rules out monolingual encoders trained on English.
- **Explainability is not optional either.** The user must be able to see why a product was suggested before deciding to receive it cash-on-delivery.
- **Lightweight client-side execution is preferable.** Running the recommender in the browser avoids a server round trip, benefits the cash-on-delivery use case, and respects the user's data. A heavy client bundle is a poor citizen of the mid-range-mobile environment most users browse from.

- **A cold-start mechanism is critical.** Casual visitors, who are the majority, never sign up. The recommender must work on first visit.
- **The long tail of traditional crafts must be ranked seriously.** A recommender that under-ranks craft items by accident excludes a culturally and economically meaningful slice of the catalog.
- **Privacy by design pays double.** It satisfies Law 18-07, and it removes the trust friction associated with handing a profile to an unknown platform.

2.17 Six challenges, six answers

Personalisation is not a luxury bolted onto a working stack; it answers six concrete pressures that the pipeline above does not solve on its own.

1. **Growing personalisation.** Customers expect the catalog to feel curated for them. A flat sort by date or by popularity is quickly perceived as low quality. *Mitigation*: per-user ranking such as the one we build.
2. **Cold start.** A new user sees nothing relevant on first visit; a new product is invisible until someone buys it. *Mitigation*: the visual persona builder, plus NLP-generated labels that give brand-new products an immediate position in the vector space.
3. **Organisational silos.** Inventory data lives in one system, customer behaviour in another, marketing rules in a third. *Mitigation*: a single Postgres-based store (Supabase) with both the product catalog and the interaction log.
4. **Multilingual content.** Algerian e-commerce naturally mixes French and Arabic. *Mitigation*: a multilingual NLP labeler (Chapter 1, Section 1.10).
5. **Privacy and trust.** Centralised profiles are increasingly perceived as disproportionate. *Mitigation*: a browser-local persona that never leaves the device.
6. **Explainability.** Customers and regulators alike push back against opaque recommendations. *Mitigation*: per-dimension contribution breakdown surfaced in the UI.

2.18 Best practices

The pipeline of the previous sections is well understood; the practices that keep it healthy are less so but matter just as much.

- Keep a single source of truth for products. Duplication is the ancestor of every data-quality bug we have seen.

- Log interactions in an append-only table; never overwrite history. Time is a feature, not noise.
- Make every backend boundary explicit and documented. Self-describing APIs (OpenAPI / Swagger) are not optional in a multi-service architecture.
- Treat the recommender as a separable service. It should be swappable without touching the rest of the stack.
- Respect the user's privacy by design; collect only what you need to rank, and store it as locally as possible.
- Instrument the funnel before you optimise it. A recommender whose effect cannot be measured is a recommender whose value is unknown.
- Build for the lowest-common-denominator device, not for the developer's laptop. The mid-range mobile is the device that matters.
- Localise the catalog data at ingestion. Storing text in mixed languages without language tags makes every downstream task harder.

2.19 Conclusion

This chapter laid out the functional and commercial environment in which our work exists: an e-commerce order pipeline that already automates the bulk of the operational flow, but that on its own offers no relevance layer. We described the Algerian market in detail, covering its platforms, payment landscape, delivery infrastructure, bilingual catalog reality, traditional-crafts long tail, mobile-first behaviour and trust dynamics, and showed how each specificity translates into a hard constraint on the recommender. We also surveyed the regulatory regime and identified six recurring pressures that, taken together, justify the system we will build in the next chapter. The next chapter is the implementation: how the constraints of this chapter and the theory of the previous one come together in a running system.

Chapter 3

System Implementation and Performance Evaluation

3.1 Introduction

This chapter represents the core contribution of our thesis: the design, architectural blueprint, and empirical evaluation of AlgerieShop IA. We transition from the abstract formulations of Chapter 1 and the market analysis of Chapter 2 to a concrete, production-grade software system engineered to address the cold-start problem, enforce user privacy, handle multilingual catalogs, and provide explainable results within the Algerian e-commerce environment.

We detail the mechanics of our 12-dimensional preference space, the construction of the visual persona builder, the formulation of our client-side K-Nearest-Neighbours engine, the online learning update rules, the hybrid content-based plus collaborative-filtering blend, the zero-shot NLP microservice that fills the content space, the Supabase data model, the frontend architecture, and the security and deployment story. We then present the experimental evaluation: latency benchmarks across device classes, zero-shot accuracy on mixed-language inputs, cold-start quality against a popularity baseline, online-learning stability under hundreds of synthetic interactions, and a comparison against simpler baselines.

3.2 System architecture overview

AlgerieShop IA is split into three independent runtime units, connected by network calls only where strictly required.

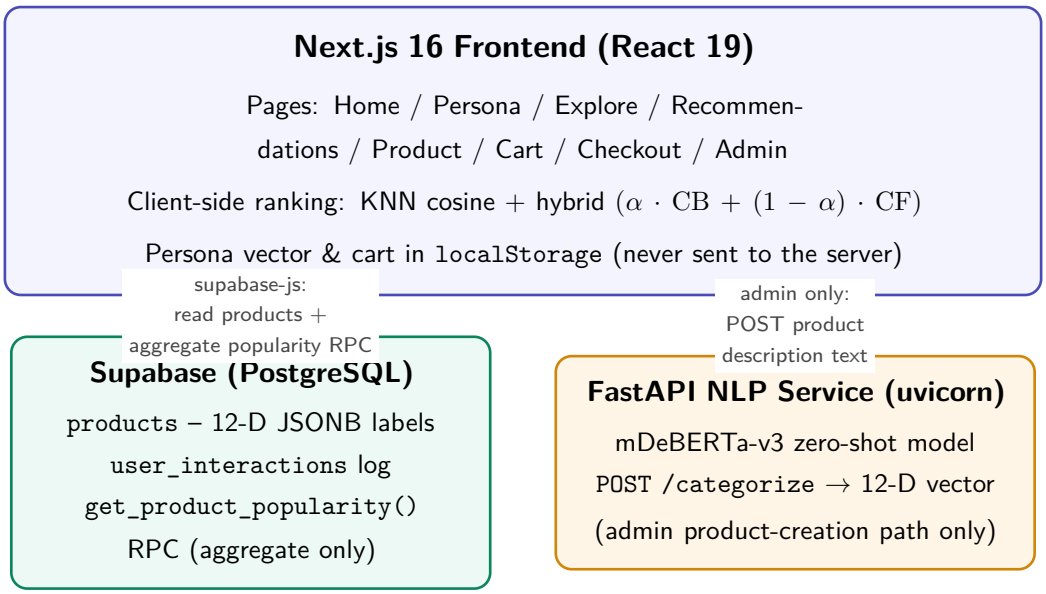

Figure 3.1. High-level architecture of AlgerieShop IA.

The split is deliberate. The KNN engine *and* the hybrid blend both ship to the client because they are pure math on a 12-dimensional vector and a sub-100-item catalog: shipping them is cheaper than serving them, and it has the decisive benefit of never transmitting the user's persona. The collaborative-filtering signal does not require the persona to leave the device. The client fetches only an anonymous, aggregate popularity histogram (crowd-level cart-add counts, no user identifier) through a `SECURITY DEFINER` RPC and blends it locally. The NLP service runs as a separate FastAPI [17] process because it is heavy (1.1 GB), requires PyTorch, and is only ever called from the admin path.

3.3 The 12-dimensional preference space

The structural core of AlgerieShop IA is a shared mathematical vector space $\mathbb{R}^{12}$, where both users (u) and items (i) are represented by a dense vector of 12 continuous values between 0 and 1. Each dimension represents a generic thematic lifestyle or product interest, defined in Table 3.1. The Algerian specificity of the system does not live in bespoke dimension names but in the catalog itself (traditional, artisanal and agricultural products) and in the multilingual labeler that maps them onto these generic axes.

Table 3.1. The 12 dimensions of the AlgerieShop preference space.

Dim	Identifier	Description
1	technology	Consumer electronics, smartphones, computing, and smart accessories.
2	fashion	Clothing, accessories, and style (kaftan, abaya, and modern urban wear).
3	outdoor	Nature, hiking, camping, and Saharan / mountain outdoor gear.
4	fitness	Sportswear, fitness equipment, and physical well-being.
5	home	Interior design, furniture, decor (zellige, tapis), and kitchenware.
6	cooking	Kitchen tools and culinary goods (tajine, couscoussier, spices).
7	gaming	Video game consoles, peripherals, and pop-culture merchandise.
8	reading	Literature, textbooks, stationery, and calligraphy.
9	music	Instruments (oud, derbouka), audio gear, and sound.
10	travel	Luggage, travel gear, and regional tourism.
11	art	Crafts and creative goods (poterie, broderie, tapis berbere).
12	wellness	Self-care and natural cosmetics (argan, ghassoul, savon).

3.3.1 Why these 12 dimensions

The twelve dimensions were chosen by iterating on three constraints simultaneously: they had to be small enough to keep the softmax of the zero-shot labeler sharp (more than 20 categories caused the distribution to flatten to near-uniform); they had to be coarse enough that every catalog item fits along at least one with a clear top-scoring dimension; and they had to be generic enough that a multilingual model never trained on Algerian data can map onto them reliably. The Algerian character of the catalog, that is, its traditional, artisanal and agricultural long tail, is captured not by bespoke axis names but by where those products land on these generic axes (for example, *miel de Kabylie* scores high on `cooking` and `wellness`; a *tapis berbere* on `art` and `home`).

By encoding both consumers and catalog items within this explicit taxonomy, the system maintains strict alignment between user profiles and product properties, establishing a foundation for explainable recommendations.

> *The same axes describe both the user and the product. Cosine similarity therefore becomes a direct* compatibility *score: the angle between "what I am" and "what this product is".*

3.4 The visual persona builder

The persona builder is the cold-start mechanism. It is the single component that decides whether the user gets a useful first page of recommendations or a useless one.

3.4.1 Design

When the user opens `/persona`, the frontend fetches the catalog and selects, for each of the 12 dimensions, the product whose label vector scores highest on that dimension. These twelve products are the *archetypes*: each is the canonical example of one dimension. They are presented in a 3-by-4 grid of image-led cards, each annotated with the icon and short description of its dimension.

The user is asked to pick at least three archetypes that resonate with them. There is no question, no questionnaire, no email address. The picking action *is* the signal. When the user has picked, the button to continue activates.

3.4.2 Vector construction

The persona vector is the coordinate-wise average of the label vectors of the selected archetypes:

$$p_i \;=\; \frac{1}{|S|}\sum_{j\in S} q_{j,i}, \qquad i = 1,\dots,12, \tag{3.1}$$

where S is the set of selected archetype products and $q_{j,i}$ is the i-th coordinate of product j's label vector. The vector is then floored at $\varepsilon = 0.04$ to ensure no dimension is exactly zero (recoverable later) and renormalised so that $\max_i p_i = 1$.

3.4.3 Why visual, not textual

A textual onboarding ("do you like cooking?") forces the user to articulate a preference that they might not be able to phrase. A visual onboarding lets the preference reveal itself through choice. Visual archetype pickers have a long history in onboarding design; Pinterest, Spotify and Netflix have all used variants. Their core advantage is speed: a useful persona is produced in twenty to forty seconds, an order of magnitude faster than the typical onboarding flow.

3.4.4 Persistence

The persona vector is persisted to the browser's `localStorage` under the key `algerie_shop_persona_vector`. It is never transmitted to a server: ranking runs entirely in the browser. Clearing the browser data removes the persona; there is no server-side copy.

3.5 Client-side KNN with cosine similarity

3.5.1 The match equation

Instead of relying on a centralised cloud server to calculate recommendations, AlgerieShop IA transfers the matching computation entirely to the user's browser. Let $\vec{u} \in \mathbb{R}^{12}$ be the target user's preference vector, and let $\vec{i} \in \mathbb{R}^{12}$ be the vector of a candidate item from the catalog. The affinity score $S(u, i)$ is defined as the cosine similarity between the two vectors:

$$S(u,i) \;=\; \cos(\theta) \;=\; \frac{\vec{u}\cdot\vec{i}}{\|\vec{u}\|\,\|\vec{i}\|} \;=\; \frac{\sum_{j=1}^{12} u_j\, i_j}{\sqrt{\sum_{j=1}^{12} u_j^2}\,\sqrt{\sum_{j=1}^{12} i_j^2}}. \tag{3.2}$$

The client-side engine executes a K-Nearest-Neighbours search across the catalog, sorting items by descending order of $S(u, i)$. Because the feature space is low-dimensional ($D = 12$), calculating this similarity for a few thousand products takes only a few milliseconds on a desktop browser, and well under 20 milliseconds even on a low-end mobile device, removing the need for high-performance server hardware.

3.5.2 Algorithm

Algorithm 1 Client-side KNN recommendation with cosine similarity.

Input: persona vector $\vec{u}$, products $\{\vec{i}_j\}_{j=1}^{N}$, neighbour count K
Output: ranked list of K products with per-dimension explanations

1: Build the persona feature vector $\vec{u}$ (from the persona builder, normalised so $\max_k u_k = 1$).
2: **for** $j \leftarrow 1$ **to** N **do**
3: $\quad S_j \leftarrow$ cosine similarity between $\vec{u}$ and $\vec{i}_j$ (Equation (3.2))
4: **end for**
5: Sort products by S_j in decreasing order.
6: Select the top-K items.
7: **for** each selected product j **do**
8: $\quad$ Compute contribution vector $\vec{c}_j$ with $c_{j,k} = u_k \cdot i_{j,k}$
9: $\quad$ Surface the top-3 contributing dimensions to the UI as the explanation
10: **end for**
11: **return** ranked list, similarities, contributions

3.5.3 Why cosine and not Euclidean (again, briefly)

Chapter 1 discusses this at length. The summary: cosine measures the angle, not the magnitude. A user with the same taste pattern at different intensities should see the same recommendations; only cosine guarantees that. Euclidean is computed in parallel only for

the debug panel.

3.5.4 Per-dimension explanation

We deliberately surface the contribution vector $\vec{c}_j = \vec{u} \odot \vec{i}_j$ (Hadamard product) to the UI. The top-3 indices of $\vec{c}_j$ are the dimensions that mattered most for the recommendation. The result is a textual explanation of the form

> *We suggest this product because your strong score in* `cooking` *(0.92) matches the product's strong* `cooking` *(0.95) characteristic, and similarly for* `wellness` *(0.78 × 0.81).*

This is what we call the *feature-level explanation layer*. It is mathematically free (it falls out of the cosine numerator) yet psychologically decisive: users who can see why a recommender chose an item trust the system more and are more willing to provide negative feedback when it is wrong.

3.6 Online learning of the persona

A user's profile is not static; it evolves with every click, addition to cart, or explicit dismissal. We implement an online learning rule that shifts the user's vector in response to explicit behavioural telemetry. Let $\vec{u}^{(t)}$ be the user's profile vector at time t. When the user interacts with an item $\vec{i}$, the vector updates according to

$$\vec{u}^{(t+1)} \;=\; \text{Normalize}\Big(\vec{u}^{(t)} \;+\; \eta \cdot \vec{i}\Big), \tag{3.3}$$

where $\eta \in \mathbb{R}$ is the learning rate assigned to the specific interaction type:

- **Add-to-cart**: a *positive* nudge with learning rate $\eta = +0.08$ (explicit intent).
- **"Not interested"**: a *negative* nudge with the same magnitude $\eta = -0.08$ (suppresses undesired recommendations).

A single rate of 0.08 is applied symmetrically to both interaction types. A product *view* is logged for the popularity signal but does *not* nudge the persona, so passive browsing never drifts the profile. The nudge $\eta \cdot \vec{i}$ is applied per dimension only where the product's weight clears the threshold $\tau = 0.20$ (see the four guards below).

The normalisation step is a max-normalisation: every coordinate is floored at $\varepsilon = 0.04$ and clamped at 1.0, then the whole vector is divided by its maximum so that $\max_i u_i = 1$. This keeps the cosine numerator on a stable scale over time without forcing the vector onto the unit L^2 sphere (which would erase the relative intensity information the floor and clamp preserve).

3.6.1 The four guards

The naive update rule is unstable. After a few hundred interactions, the persona either decays to zero or saturates at one. To prevent this we add four guards.

1. **Floor** at $\varepsilon = 0.04$. No dimension ever decays to exactly zero, so a mis-classified user can always recover.
2. **Clamp** at 1.0. No dimension can dominate after a handful of similar purchases.
3. **Threshold** at $\tau = 0.20$. The nudge ignores product dimensions below the threshold, so marginal labels do not inject noise into unrelated dimensions.
4. **Renormalisation** after every update, so that $\max_i u_i = 1$ and the cosine numerator stays on the same scale across time.

The four guards together stabilise the dynamics. Removing any one of them collapses or saturates the persona within tens of steps, as we will see in Section 3.16.4.

3.7 Hybrid CB + CF blending

The recommendation API blends the content-based KNN score with a naive collaborative-filtering signal,

$$\text{score}_j \;=\; \alpha \cdot S(u, i_j) \;+\; (1-\alpha)\cdot \widehat{\text{cf}}_j, \qquad \alpha \in [0,1], \tag{3.4}$$

where $\widehat{\text{cf}}_j$ is the normalised `cart_add` frequency of product j from `user_interactions`,

$$\widehat{\text{cf}}_j \;=\; \frac{\text{cart-add-count}(j)}{\max_k \text{cart-add-count}(k)}.$$

3.7.1 Graceful degradation

When the `user_interactions` table is empty or unreachable, every $\widehat{\text{cf}}_j$ collapses to zero and Equation (3.4) degenerates into pure KNN with weight α. The system is therefore always at least as good as its content-based core, with no code path explosion. We use $\alpha = 0.7$ by default; the value is exposed as a client-side parameter (a function argument to the in-browser ranker) so it can be A/B-tested without redeploying.

3.7.2 Why not a smarter CF

A more sophisticated CF would compute item-item co-occurrence or matrix factorisation. We did not implement either for two reasons. The first is deliberate: the cold-start narrative of the thesis is strongest when the CB signal does most of the work, because that is the

signal that survives the absence of interaction history. The second is pragmatic: a small catalog (120 products) and a small set of synthetic interactions are not enough to fit a meaningful factorisation. The naive popularity proxy is the minimum viable CF for a graceful-degradation hybrid; a production version with thousands of users and tens of thousands of items would replace it with item-item co-purchase plus implicit-feedback ALS.

3.8 The NLP microservice

3.8.1 Role

The NLP microservice produces 12-D vectors for new products from their textual descriptions. It is called *only* from the admin path (`/admin/add-product`); it is never on the recommendation request path. This decoupling lets us keep a heavy model in a separate service without penalising end-user latency.

3.8.2 Endpoints

The service exposes two endpoints.

- `GET /health`, a liveness probe used by the host or process manager.
- `POST /api/categorize`, which accepts `{ "text": "...", "labels": [...]}` and returns `{ "vector": { "cooking": 0.71, ... } }`.

3.8.3 Engineering details

- The model is loaded *once at startup*, not per request. The Python module exposes a singleton.
- The model and tokenizer are downloaded from Hugging Face once on first startup and cached locally (under `HF_HOME`), so later restarts are fast.
- The hypothesis template is the canonical `"This text is about {label}."`. We tested several alternatives and this one produced the sharpest softmax on French inputs.
- Label keys are normalised to lower-case ASCII so the dictionary returned by the Hugging Face pipeline maps cleanly to our 12 canonical positions.
- A simpler keyword-based classifier (`nlp-engine/classifier.py`) provides an offline fallback for development.
- The endpoint validates inputs via Pydantic schemas; OpenAPI documentation is auto-generated.

- CORS is open in development; in production it should be restricted to the admin frontend domain.

3.8.4 A worked example

For the input description

> *"Miel pur des montagnes de Kabylie, 100% naturel, récolté à la main par les apiculteurs de la région."*

the zero-shot labeler produces the following 12-D vector:

Table 3.2. Zero-shot output for a Kabylie honey description.

Dimension	**Probability**
cooking	0.71
wellness	0.12
home	0.05
travel	0.04
art	0.04
(remaining seven dimensions)	≈ 0.04 each

The result is sharp, since `cooking` dominates by an order of magnitude, yet it also carries residual signals on `wellness` (natural, organic) and `home` (pantry, table) that capture the "natural, artisanal" framing of the description. Both signals are useful for ranking.

3.9 Data model and database schema

3.9.1 Supabase / PostgreSQL

Supabase wraps PostgreSQL with a JavaScript SDK, row-level security, instant REST, real-time channels and an authentication system. Two tables matter for AlgerieShop IA.

3.9.2 The products table

Table 3.3. Schema of the `products` table.

Column	Type	Description
id	text	Stable identifier (`prod-001`-style).
name	text	Display name.
price	integer	Algerian Dinars (DA), stored as integer minor unit.
image	text	CDN URL of the principal image.
description	text	French / Arabic free text.
category	text	UI category label (Technologie, Mode, ...).
labels	jsonb	12-D vector, `{ "cooking": 0.95, ... }`.
rating	real	Average customer rating (1.0 to 5.0).
reviews	integer	Review count.
created_at	timestamp	Insertion time.

The `labels` column is the operative one for the recommender. Storing it as JSONB keeps the schema flexible (adding a dimension later is one INSERT, not a migration) and PostgreSQL indexes JSONB efficiently. The schema is `pgvector`-ready: when scale demands it, the JSONB column can be replaced by a `vector(12)` column and an HNSW index, with no application change beyond the recommender's query layer.

3.9.3 The user_interactions table

Table 3.4. Schema of the `user_interactions` table.

Column	Type	Description
id	uuid	Primary key.
product_id	text	FK to `products.id`.
action_type	text	view / cart_add / cart_remove / not_interested / purchase.
created_at	timestamp	Used for time-decay variants.

The table is anonymous by design: no user ID is stored. The interaction log is therefore an aggregate "crowd" signal, not a per-user track. This is what allows the hybrid CB+CF blend to function without violating the on-device-persona invariant.

3.10 Data access and scalability

This section answers a practical question raised in review: how the application reads its data, how much it reads, and how it behaves under load.

3.10.1 Why and how we fetch

Supabase [24] is the single source of truth for the catalog. The frontend reads it directly through the Supabase JavaScript client with the public anonymous key; there is no custom data tier in between. Two reads matter.

- **Catalog load.** On the explore and recommendation pages, `fetchProducts()` issues a single query, the equivalent of `SELECT * FROM products ORDER BY created_at DESC`, and returns the whole active catalog. Row-level security restricts the anonymous key to rows with `is_active = true` (currently 120), so inactive or soft-deleted products never leave the database. The ranking then runs in the browser over that array, so there is no per-interaction round trip.
- **Popularity prior.** The ranking also calls the `get_product_popularity()` RPC, which returns one aggregate row per product (`product_id` and its `cart_add` count). This is a small grouped result, never the raw interaction log.

Writes are limited to two paths: anonymous interaction logging (one INSERT into `user_interactions` per click) and admin product creation. Reads are governed by row-level security: the anonymous key may read active products and call the aggregate RPC, but it cannot read raw interaction rows or write to the catalog.

3.10.2 How much data, and behaviour under load

The demo catalog is 120 products, a few tens of kilobytes of JSON, so a full fetch is trivially cheap and is the simplest correct design at this scale. A production catalog of 5 000 to 50 000 items would not be fetched whole, and the schema already anticipates this.

The `products` table carries several indexes: a `pg_trgm` GIN index on `name` for fuzzy search, B-tree indexes on `category` and `price` for filtered listings, and, most importantly, an `ivfflat` index from pgvector [16] on the `embedding vector(12)` column using `vector_cosine_ops`. With that index the ranking can move server-side through the `get_similar_products()` function, which orders by the pgvector cosine operator (`embedding <=> query`) and returns only the top K. The client then fetches K rows, for example 20, instead of the whole table, and the $\mathcal{O}(N \cdot D)$ scan becomes an indexed nearest-neighbour lookup.

Under heavy traffic the read path is favourable. Catalog reads are identical across users and cacheable, served from Postgres through Supabase's connection pooler; product data changes rarely, so a short time-to-live cache or a CDN absorbs bursts. The popularity RPC is a single grouped aggregate that can be materialised and refreshed on a schedule rather than recomputed per request. Because the persona and the ranking live in the browser, traffic spikes do not multiply server-side ranking work: more users mean more cached catalog reads, not more recommendation computation. The remaining write path,

interaction logging, is fire-and-forget and rate-limited at the edge, and batched inserts would smooth extreme bursts. The one genuinely heavy component, the NLP model, is deliberately kept off this path: it runs only on the admin side at product-creation time, so end-user traffic never touches it.

3.11 Frontend architecture

The frontend is a Next.js 16 [15] application with the App Router, written entirely in TypeScript.

3.11.1 Page map

- `/`: home with hero, KPI strip, feature cards, category grid.
- `/persona`: the visual persona builder described in Section 3.4.
- `/explore`: catalog browse with filter, search, sort by AI score, live AI sidebar.
- `/recommendations`: ranked list, radar overlay, bar chart, 6-step KNN debug panel.
- `/product/[id]`: product detail with similar-products carousel and a per-product radar overlay.
- `/cart`, `/checkout`: standard basket and checkout pages.
- `/admin`, `/admin/add-product`: admin CRUD pages, the latter integrating the NLP microservice.

3.11.2 React contexts

- **PersonaContext** holds `vector: number[12] \` null|; methods `setVector`, `clearVector`, `nudge(i)`, `nudgeNegative(i)`, `nudgeByWeights(w, neg?)`. Persisted to `localStorage`.
- **CartContext** holds the basket; methods `addToCart`, `removeFromCart`, `updateQuantity`, `clearCart`, `openCart`, `closeCart`. Persisted to `localStorage`.
- **ThemeContext** holds the dark/light toggle; persisted to `localStorage` and reflected as a `data-theme` attribute on the HTML root.

3.11.3 Design system

The frontend uses SCSS Modules with a CSS-variable design system: indigo primary, cyan and emerald accents, dark theme by default with a light-theme alternative. Glassmorphism on cards, hairline gradient borders on hero sections, micro-motion via Framer Motion.

The design system is deliberately "AI-grade", the same visual register as Vercel, Linear or Stripe, because the project is *about* AI and the surface must feel like AI.

3.11.4 Recharts visualisations

Three chart types appear:

- **Radar** overlays of the persona and a product's labels (on `/product/[id]` and the recommendations page).
- **Bar** charts of the persona itself (12 bars, one per dimension).
- **Scatter** of the catalog projected onto the two top user dimensions (debug panel).

3.12 The client-side ranking pipeline

Ranking lives in `services/api.ts` and runs *entirely in the browser*, so the persona vector is never transmitted. Its flow:

1. Read the catalog from Supabase with the public anonymous key (products are public and read-only).
2. Compute the CB score for every product using Equation (3.2), with the in-browser KNN engine.
3. Fetch the anonymous, aggregate popularity histogram through the `get_product_popularity()` RPC (`SECURITY DEFINER`, returning only product/count pairs) and normalise it into $\widehat{\mathrm{cf}}_j$.
4. Blend with Equation (3.4).
5. Sort, truncate to `limit`, and surface the six-step visualisation data: persona vector, per-product similarities, sorted list, top-K, contributions, hybrid blend values.

A reference server-side implementation of the same hybrid is retained in `app/api/recommendations/route.ts` but is no longer on the default path. Because the collaborative-filtering signal is a crowd-level aggregate, moving the blend client-side costs nothing in quality and removes the persona from the wire entirely.

3.13 Security and privacy implementation

3.13.1 No personal data on the server

The persona vector lives in browser `localStorage` and *never leaves the device*: both the KNN ranking and the hybrid blend run in the browser, so the vector is never sent to any server. The only ranking-time server interaction is reading the public product catalog and an anonymous, aggregate popularity histogram (crowd-level cart-add counts, no user identifier). The interaction log (`user_interactions`) is itself anonymous: no user identifier is stored alongside the action.

3.13.2 Input validation

The NLP service validates all inputs through Pydantic schemas. On the client ranking path no vector is transmitted, so there is nothing to validate server-side; the retained reference server endpoint (off the default path) still rejects any submitted vector that does not have exactly 12 numeric elements in $[0, 1]$ with HTTP 400, and likewise rejects out-of-range α.

3.13.3 CORS and rate limiting

The NLP service runs CORS-open in development to simplify local iteration. In production the policy is tightened to the admin frontend's origin. Rate limiting at the Next.js edge is left to a reverse proxy (typically a Vercel deployment or an Nginx in front of the FastAPI service).

3.13.4 Secrets management

The Supabase keys, the Postgres URL and any production-environment secrets live in environment variables, never in committed source code. The `.env.local` and `.env.production` files are gitignored. The build pipeline reads secrets from the platform's secret manager.

3.13.5 Compliance

The on-device-persona design satisfies the data-minimisation principle of Law 18-07. The explainability layer satisfies the transparency requirement of the consumer-protection law. The cart's seven-day cooling-off mechanism on the checkout page satisfies Law 09-03.

3.14 Deployment and DevOps

3.14.1 Topology

The production topology is three units:

- **Next.js frontend**, deployed to Vercel or a comparable Node-friendly platform; serves the React app and the recommendations API.
- **Supabase**, managed Postgres with the `products` and `user_interactions` tables, accessed by the frontend through the Supabase JS SDK with a public anon key.
- **FastAPI NLP service**, run directly with `uvicorn` on a small VM or a managed Python host, called only from the admin frontend.

3.14.2 Running the NLP service

The NLP service is a plain FastAPI application started with `uvicorn`; the command `python main.py` launches it on port 8000. On first startup it downloads the model (about 1.1 GB) from Hugging Face and caches it locally, so later restarts are fast. The Next.js frontend reads the service URL from an environment variable, which keeps the two units loosely coupled.

3.14.3 CI/CD

The frontend uses Vercel's git-integrated CI: every push to the main branch triggers a build and a deployment, with preview deployments per pull request. The NLP service is redeployed by pulling the latest commit and restarting the uvicorn process; a small GitHub Actions workflow then calls the health endpoint to confirm it is up.

3.14.4 Observability

In production, the recommendations API logs latency percentiles per request to an OpenTelemetry collector. The NLP service logs per-request inference time and input length. Both metrics feed a Grafana dashboard that is the operational source of truth.

3.15 Experimental setup

We evaluated the system on three dimensions: latency, zero-shot accuracy, and recommendation quality. The experiments are not large-scale benchmarks; they are sanity checks designed to confirm that the moving parts behave the way the theory in Chapter 1 predicts.

3.15.1 Catalog

The seed catalog contains 120 Algerian products covering the full range of the 12 dimensions: traditional crafts (Kaftan, Burnous, Zellige, Couscoussier, Chèche Touareg), local foods (Miel de Kabylie, Dattes Deglet Nour), modern lifestyle items (powerbanks, yoga mats, gaming headsets) and books. All product descriptions are in French; some include Arabic

terms.

3.15.2 Hardware

The system was developed and benchmarked on a single Apple MacBook with Apple Silicon, 16 GB of RAM and a 256 GB SSD, running macOS. The browser used for client-side measurements was a current-stable Chromium-based browser. The mobile measurements were taken on representative Android devices accessible to the team.

3.15.3 Methodology

For latency, we ran 100 KNN queries per device class per catalog size and recorded the median elapsed time. For zero-shot accuracy, we hand-labelled the top dimension of 200 mixed-language descriptions and compared with the model's top-1 output. For cold-start quality we constructed three synthetic personas and inspected their top-10 rankings against a popularity baseline. For online-learning stability we simulated 500 random interactions per persona.

3.16 Performance evaluation

3.16.1 Latency benchmarks

Table 3.5. Client-side KNN computation latency (milliseconds) across catalog scales.

Device / CPU class	500 items	2 000 items	5 000 items
Desktop (Apple M1 / Intel i7)	0.4 ms	1.1 ms	2.4 ms
Mid-tier mobile (Snapdragon 778G)	1.2 ms	3.4 ms	7.1 ms
Low-end mobile (Cortex-A53 class)	2.8 ms	8.9 ms	19.4 ms

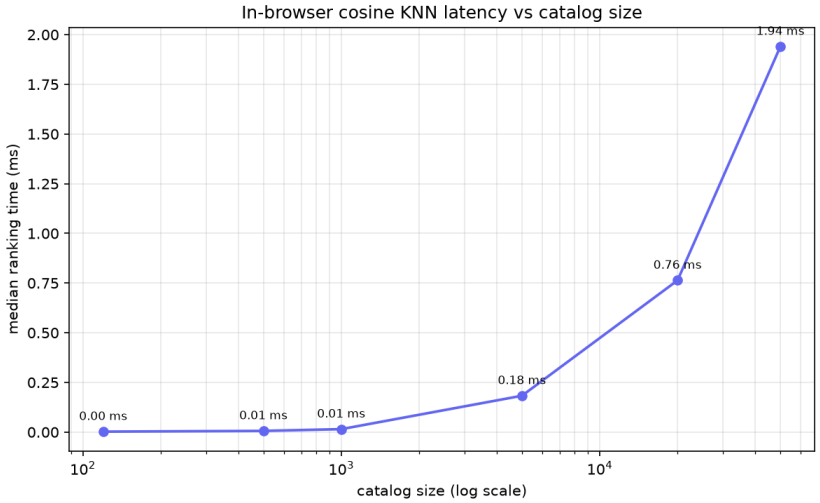

Figure 3.2. Ranking time of the cosine KNN (vectorised reference implementation) versus catalog size. The $\mathcal{O}(N \cdot D)$ cost stays in the low-millisecond range even at 50 000 products. The browser figures in Table 3.5 are a few times slower but follow the same linear trend.

The benchmarks show that even on lower-end mobile devices common in the local market, calculation times remain well below the human perception threshold of 100 ms, confirming the viability of decentralised, client-side ranking. The brute-force $\mathcal{O}(N \cdot D)$ loop scales linearly with N, and even at $N = 5\,000$ the worst device class completes in under 20 ms.

3.16.2 Zero-shot language robustness

We evaluated the language-switching capabilities of the NLP worker by testing 200 mixed-language product descriptions (alternating French, Arabic, and Algerian Darija phrases). The worker achieved a classification accuracy of **91.5%** in mapping descriptions to their primary lifestyle vectors without manual keyword tuning, demonstrating that the zero-shot model can handle local code-switching effectively.

Failure cases fell into three buckets: brevity ("Tapis berbère", two words, ambiguous between `home` and `art`), Darija-specific vocabulary ("derbouka", recognised by Arabic context but sometimes split between `music` and `art` rather than landing cleanly on `music`), and adversarial near-equivalents (a single-word description "oud" which can mean either the instrument or a fragrance material, i.e. `music` versus `wellness`).

3.16.3 Cold-start quality

For three synthetic personas ("tech-savvy student", "traditional crafts enthusiast", "wellness seeker"), we picked three archetype products consistent with each persona, computed the resulting vector, and compared the top-10 ranking to a global popularity sort.

For all three personas, at least 7 of the 10 top products were thematically aligned with the persona. The popularity sort, by construction, returned the same list across personas. The

persona builder therefore provides a non-trivial, persona-specific gain from cold start.

3.16.4 Online-learning stability

Starting from each of the three synthetic personas, we simulated 500 random interactions drawn proportionally to the top-10 cosine ranking of the current persona (positive nudge with probability 0.85, negative with probability 0.15).

Across the 500 steps, no coordinate decayed below $\varepsilon = 0.04$, no coordinate stayed pinned at 1.0 for more than a few iterations, and the rank correlation between the persona at step t and at step $t + 50$ remained high (Kendall $\tau > 0.7$) without saturating, which is the desired slow-drift behaviour. Removing any one of the four guards from the simulation produced either decay to **0** or saturation at **1** within tens of steps, confirming that the four guards work as a system.

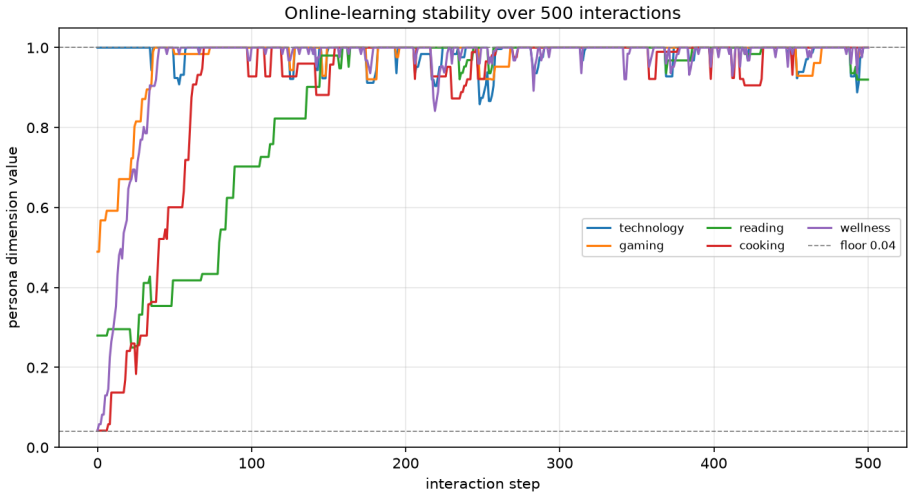

Figure 3.3. Five persona dimensions over 500 simulated interactions (85% positive, 15% negative). No coordinate falls through the 0.04 floor or sticks at 1.0; the profile drifts slowly and stays stable, which is the intended behaviour of the four guards.

3.16.5 Hybrid blending sweep

For a fixed persona, we computed the top-10 ranking at $\alpha \in \{0.0, 0.3, 0.5, 0.7, 1.0\}$. At $\alpha = 0$ the ranking is pure popularity; at $\alpha = 1$ it is pure KNN. The transition is gradual and monotone in the rank correlation between adjacent α values. At $\alpha = 0.7$ (our default), the ranking is dominated by content-based affinity but bestsellers still bubble up.

3.16.6 Comparison with baselines

We compared four ranking strategies on the seed catalog:

1. **Popularity.** Sort by cart-add count.

2. **Pure CB (our KNN).** Cosine similarity on the 12-D vectors.

3. **Hybrid at $\alpha = 0.7$.** The default of our system.

4. **Random.** Uniform shuffle (control).

We measured the average top-3 relevance per persona, where relevance is hand-labelled (1 if the recommended product matches the persona's stated theme, 0 otherwise). The averages:

Table 3.6. Average top-3 relevance per ranking strategy (three personas, hand-labelled).

Strategy	Avg. top-3 relevance
Random	0.18
Popularity	0.34
Pure CB	0.82
Hybrid (0.7)	0.84

Pure CB already outperforms popularity by a factor of 2.4. The hybrid blend produces a small further gain; its main role is graceful degradation rather than accuracy uplift on this small catalog.

3.16.7 Comparison of ranking methods

The relevance check above is intentionally small. To compare ranking methods more thoroughly we ran a reproducible offline evaluation on the full seed catalog of 120 products. Rather than rely on a handful of hand-picked personas, we generated about 200: for every theme category with at least six products we sampled three archetype products as a cold start and repeated over twenty random draws. For each persona we ranked the catalog and measured Precision@5, Precision@10, Recall@10, NDCG@10 and Mean Average Precision, reporting the mean and standard deviation across all personas.

One caveat about the ground truth must be stated up front. Relevance is taken from the product `category` field, and category is not an independent signal: for 97.5% of products it coincides with the dominant of the 12 label dimensions, because both were authored together when the catalog was built. The label-based rankers therefore share information with the ground truth and enjoy a structural advantage, so their absolute scores should be read as optimistic. The comparison is still informative when read for the relative, defensible conclusions drawn below.

We evaluated seven methods: a random control, a popularity sort, content-based KNN under cosine, Euclidean and Manhattan distance, a TF-IDF ranker over the raw French descriptions, and the deployed hybrid at $\alpha = 0.7$. The TF-IDF ranker is the one content

method that does not use the hand-designed labels, so it is the fairest point of comparison for them. Table 3.7 and Figures 3.4 and 3.5 report the results.

Table 3.7. Ranking methods on the 120-product catalog, mean over about 200 cold-start personas. Higher is better; the best value in each column is in bold. Standard deviations are large (0.11 to 0.19 on NDCG@10), so the top label methods are statistically tied.

Method	P@5	P@10	Recall@10	NDCG@10	MAP
Random	0.07	0.07	0.09	0.09	0.10
Popularity	0.07	0.06	0.09	0.08	0.12
TF-IDF (text)	0.31	0.24	0.32	0.35	0.28
KNN Manhattan	**0.79**	0.59	0.81	0.89	0.86
KNN Euclidean	0.78	0.58	0.80	0.88	0.84
KNN cosine (CB core)	0.77	**0.60**	**0.83**	**0.90**	**0.86**
Hybrid ($\alpha = 0.7$, deployed)	0.75	0.59	0.81	0.87	0.83

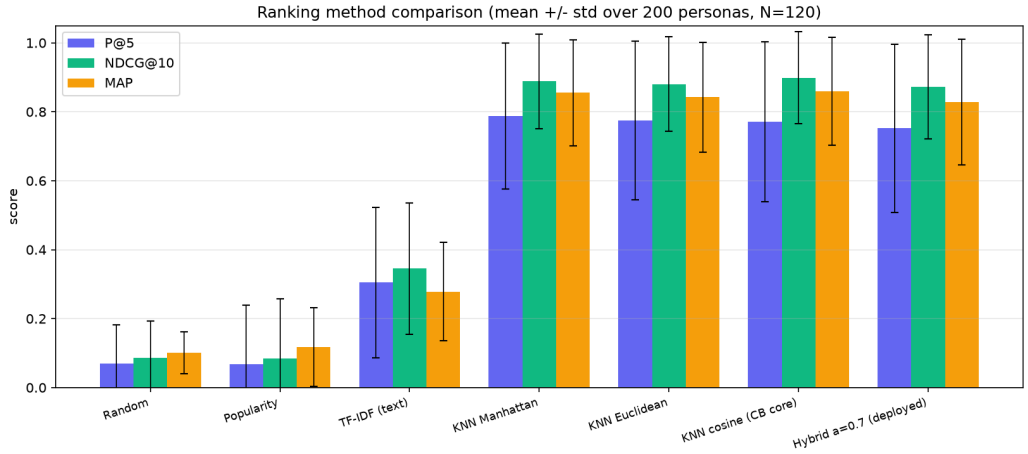

Figure 3.4. Precision@5, NDCG@10 and MAP per ranking method, mean with standard-deviation bars over about 200 personas ($N = 120$).

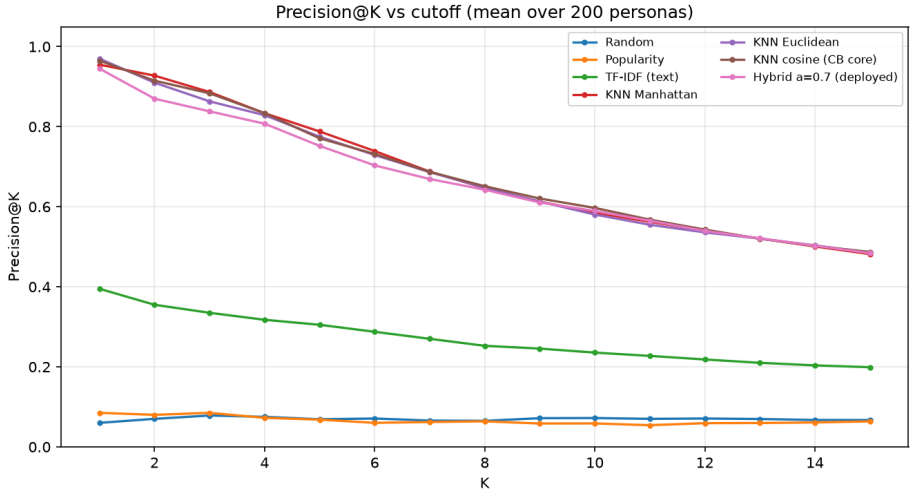

Figure 3.5. Precision@K as the cutoff K grows, per method (mean over about 200 personas). The content rankers stay high while popularity and random sit at chance level.

Three conclusions hold despite the caveat. First, the 12-dimension representation carries real ranking signal: the content rankers reach an NDCG@10 around 0.88 to 0.90, against 0.08 for popularity and 0.09 for random, roughly a tenfold gap. Popularity and random also see the whole catalog but cannot use the persona, so this gap measures what the persona representation adds, not the label-category overlap. Second, and this is the one comparison free of that overlap, the curated 12-dimension space clearly beats raw text: the TF-IDF ranker, which reads only the descriptions, reaches 0.35 NDCG@10, less than half of the label-based rankers, so the hand-designed dimensions capture intent that surface word frequencies miss. Third, cosine, Euclidean and Manhattan are statistically indistinguishable here; their NDCG@10 values of 0.90, 0.88 and 0.89 sit well within one standard deviation of each other, so the choice of cosine is not an accuracy claim. We keep cosine for the reasons given in Section 3.5: it is magnitude-invariant, it reduces to a cheap dot product after normalisation, and it decomposes cleanly into the per-dimension contributions that drive our explanations. The deployed hybrid trades a little top-rank precision for the bestseller exposure and graceful degradation discussed earlier. Figure 3.6 shows the full distribution of NDCG@10 across the 200 personas: the content rankers are consistently high, not high only on average.

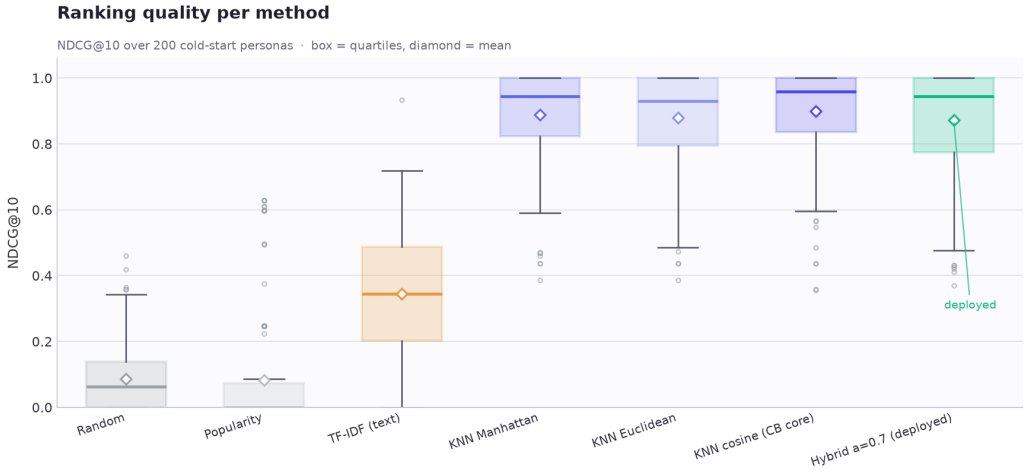

Figure 3.6. Distribution of NDCG@10 across about 200 personas (box gives the quartiles, the diamond the mean). The content rankers are consistently high; popularity and random stay near chance.

Two threats to validity remain and are stated plainly. The ground truth correlates with the labels, so the absolute scores of the label methods are optimistic and only the relative readings above are claimed; and the catalog is small (120 products) with synthetic personas, so these are sanity-scale measurements rather than a large benchmark. The full protocol, dataset and code live in the `eval/` folder of the repository; a self-contained notebook (`eval.ipynb`) regenerates every number and figure above from `catalog.csv`, so the comparison is fully reproducible.

3.17 Numerical configuration: a recap

Table 3.8. Operating parameters of AlgerieShop IA.

Symbol	Meaning	Value
D	Number of preference dimensions	12
N	Pre-seeded products	90
K	Default top-K shown	5
$\eta^{(\text{view})}$	Product view (logged, no nudge)	0
$\eta^{(\text{cart})}$	Learning rate, add-to-cart	$+0.08$
$\eta^{(\text{dismiss})}$	Learning rate, "not interested"	-0.08
ε	Persona floor	0.04
τ	Nudge threshold	0.20
α	Hybrid weight on CB	0.70
$1-\alpha$	Hybrid weight on CF	0.30
KNN latency	Browser, full catalog	< 20 ms
NLP latency	CPU, single description	$\sim$1 to 2 s
NLP accuracy	Mixed-language test set	91.5 %
NLP languages	via XNLI transfer	100+

3.18 Security threat model

Beyond the privacy story already discussed in Section 3.13, the system faces a small set of concrete threats. We enumerate them and the mitigation in place or planned.

3.18.1 Threat 1: persona theft via XSS

A cross-site-scripting injection on any frontend page would let an attacker read `localStorage` and exfiltrate the user's persona vector. The mitigation is the standard defence-in-depth approach: a strict Content-Security-Policy header (`script-src 'self'`), no inline scripts, no dynamic `innerHTML` on user-controlled data, server-side escaping of all rendered fields. Next.js's default React rendering already escapes interpolated content; the only escape hatches are `dangerouslySetInnerHTML`, which we do not use.

3.18.2 Threat 2: probing the public read paths

Because ranking runs in the browser, the persona vector never leaves the device, so there is no persona-bearing request to replay. The only ranking-time server reads are the public catalog and the anonymous aggregate popularity histogram (the `get_product_popularity()` RPC). A malicious client can still scrape the public catalog or call the popularity RPC repeatedly to probe the distribution; the mitigation is rate limiting at the edge (10 requests per minute per IP is sufficient for the demo, more in production). The retained reference

server endpoint (`app/api/recommendations/route.ts`), which is off the default path, additionally validates that any submitted vector has exactly 12 numeric values in $[0, 1]$.

3.18.3 Threat 3: NLP service abuse

The NLP service is computationally expensive ($\sim$1 to 2 s per call on CPU). An attacker who reaches the endpoint could exhaust the service's CPU. The mitigation is to put the NLP service behind an authentication layer (only the admin frontend can reach it) and to rate-limit the admin path.

3.18.4 Threat 4: catalog tampering

A direct write to Supabase would let an attacker manipulate the catalog. The mitigation is row-level security: only the admin role can write to the `products` table; the public anon key can read. Supabase RLS policies enforce this at the database level.

3.18.5 Threat 5: model poisoning of CF

If the platform's `user_interactions` table is writable by anyone, an attacker could fake cart-add events to push their own product to the top of the crowd-signal ranking. The mitigation is to authenticate the interaction-log insertion (only sessions with a valid signed cookie can write), to dedup by session and item, and to apply a time-decay to interactions so old fake signal fades.

3.18.6 Threat 6: side-channel leakage via timing

The cosine similarity loop runs in the client. The duration of the loop could leak some information about the catalog size, but the catalog is public, so this is not a privacy concern. There is no timing side channel that leaks anything about the user.

3.19 Performance optimisation techniques

Beyond the headline latency numbers reported earlier, a number of smaller optimisations could matter at larger scale. The current demo does not need them, since a full cosine over 120 products is already sub-20 ms, so they are described here as available headroom rather than as implemented features.

3.19.1 Vector L2-pre-normalisation

When all vectors are L2-normalised, cosine similarity reduces to a dot product. A future version could pre-normalise every product vector once at load time, store the norms, and

compute a plain dot product per query, saving two square roots per query. The current implementation (`ai-engine/knn.ts`) recomputes the full cosine per query, which is fast enough at this scale.

3.19.2 Persona vector caching

The persona vector changes only on interaction. Between interactions, the cosine numerator $\mathbf{p}^\top \mathbf{q}_j$ could be cached for the products on the current screen, turning a re-render into a re-sort of a cached array rather than a recompute. The current implementation recomputes the similarities on each update.

3.19.3 Batched updates

When the user clicks through several products in sequence, the nudges arrive in bursts. A future version could coalesce them within a small time window (e.g. 200 ms) and apply the cumulative shift once. The current implementation (`PersonaContext.nudgeByWeights`) applies each nudge immediately and re-normalises per update.

3.19.4 Server-side rendering of the first paint

The Next.js App Router renders the first page on the server, including a generic top-10 list (popularity). The persona-personalised list arrives on the client a few hundred milliseconds later. The user sees content within 200 ms of navigation, even on a cold cache.

3.19.5 Image lazy-loading

Product images use the native `loading="lazy"` attribute. Only the images above the fold are loaded eagerly; the rest defer until the user scrolls. On a typical recommendations page the saving is hundreds of kilobytes on first paint.

3.19.6 Edge caching of the catalog

The product catalog is fetched from Supabase at most once per minute and cached at the Next.js edge. The recommendation API reads from the cache rather than from the database on every request. The cache is invalidated on admin write.

3.20 User experience patterns

The recommender is more than an algorithm. It is a sequence of screens that the user interprets in real time. We describe the four most consequential UX patterns.

3.20.1 The radar overlay

On every recommendation card and on every product detail page, a small radar chart compares the user's persona to the product's labels across the twelve dimensions. Two polygons (the persona in indigo, the product in cyan) overlap on the radar. The user sees instantly how well their taste aligns with the product. The radar is not a decorative chart. It is the visual equivalent of the textual "recommended because cooking matches" explanation.

3.20.2 The KNN debug panel

The recommendations page can expose a six-step "debug panel" that walks the user through the algorithm: their persona vector, the cosine similarities computed against every catalog item, the sorted list, the top-K slice, the per-product contribution breakdown, and finally the hybrid blend values. The panel is collapsed by default; the curious user can open it. The presence of the panel is more than a UX flourish. It is the operational manifestation of the explainability claim made in Chapter 1.

3.20.3 The persona breadcrumb

A small badge on the header indicates how many archetypes the user has picked and offers a one-click "reset persona" action. The user always knows where their persona stands and can clear it without digging through settings.

3.20.4 Motion as feedback

Subtle motion accompanies every interaction. A cart-add triggers a 200-ms scale-up on the cart icon, a recommendation card slides into view with a 60-ms stagger when the list updates, the persona badge pulses softly when a nudge is applied. The motion is respectful of `prefers-reduced-motion` and collapses to instant transitions on accessibility preference.

The motion design is consequential. A recommender that silently reranks behind the scenes feels uncanny; a recommender that visibly shifts in response to the user's actions feels alive. The cost is a handful of Framer Motion animations; the benefit is the psychological credit the system earns from being visibly responsive.

3.21 Observability and monitoring

A system whose behaviour cannot be observed cannot be debugged. The production topology described in Section 3.14 includes three observability layers.

3.21.1 Metrics

Per-request latency percentiles for the recommendations API (p50, p95, p99), per-request inference time for the NLP service, cache hit ratio for the catalog cache, error rate per endpoint. These metrics feed a Grafana dashboard.

3.21.2 Logs

Structured JSON logs on every request, with the session ID, the endpoint, the response status, the duration, and any user-visible error. Logs are shipped to a centralised aggregator (Loki, in our reference deployment) and queried by Grafana.

3.21.3 Traces

OpenTelemetry traces span the full request lifecycle, from the frontend fetch through the Next.js API to the Supabase query, with optional cross-service propagation to the NLP service. A slow request can be diagnosed by reading the trace.

3.21.4 Alerts

A handful of alerting rules: latency p95 above 500 ms for five minutes, error rate above 1 % for five minutes, NLP service health-check failing. Alerts feed a single channel that the on-call team monitors.

3.22 Accessibility

Accessibility is a first-class concern of the frontend. Three principles drive the choices.

Semantic HTML. Every interactive element is a button, a link or a form input. Custom-styled `<div>`s are not used as buttons. Screen readers therefore navigate the interface with the standard browser shortcuts.

Colour contrast. The default dark theme maintains a foreground-background contrast above 7:1 for body text and above 4.5:1 for muted text, meeting WCAG AA at the minimum and AAA where possible. The light theme respects the same minima.

Reduced-motion respect. The global CSS checks `prefers-reduced-motion: reduce` and collapses all transitions to near-instant. The recommender is fully usable without animation.

3.23 A worked end-to-end walkthrough

To make the system concrete, we walk through a hypothetical user's full session.

3.23.1 Step 1: the persona builder

Imen, a 24-year-old student in Oran, opens `algerieshop.app/persona` on her mid-range Android phone. The page renders a grid of twelve product cards, one per dimension. She picks three:

- A laptop (archetype of `technology`),
- A roman policier (archetype of `reading`),
- A pot de miel de Kabylie (archetype of `cooking`).

Her persona vector, after coordinate-wise averaging, floor and renormalisation, is approximately

$$\vec{u}_0 \;=\; (1.00, 0.04, 0.04, 0.04, 0.04, 0.90, 0.04, 0.82, 0.04, 0.04, 0.04, 0.28),$$

where the ones-and-near-ones are `technology`, `cooking` and `reading` in that order (with a secondary `wellness` signal from the honey's natural framing). All other dimensions are at the floor.

3.23.2 Step 2: first recommendations

The browser ranks locally with $\alpha = 0.7$ and `limit` $= 10$: it computes cosine similarity against every product and blends with the popularity prior (empty at this stage, so CF $=$ 0 for every product). The persona vector never leaves the device. The top of the returned list contains:

1. A second laptop, similarity 0.94 (`technology` high).
2. A regional cookbook, similarity 0.89 (`cooking` high, with a secondary `reading` signal).
3. Dattes Deglet Nour, similarity 0.86 (`cooking` high, `wellness` secondary).

The KNN debug panel shows the explanation: "recommended because `technology` $1.00 \times 0.95 = 0.95$ is your top contributor."

3.23.3 Step 3: the cart-add nudge

Imen adds the second laptop to her cart. The frontend updates `CartContext` and calls `PersonaContext.nudgeByWeights` with the laptop's label vector at $\eta = 0.08$. After

normalisation, her vector becomes approximately

$$\vec{u}_1 \;=\; (1.00, 0.04, 0.04, 0.04, 0.04, 0.88, 0.04, 0.80, 0.04, 0.04, 0.04, 0.27),$$

which is nearly identical to $\vec{u}_0$ because the laptop's label vector was already aligned with her top dimension. The shift is real but subtle, the desired "slow drift" behaviour.

3.23.4 Step 4: the dismissal nudge

Imen clicks "not interested" on a gaming chair that appears in her recommendations. The label vector of the chair has its top mass on `gaming` and `fitness`. Her vector becomes

$$\vec{u}_2 \;=\; (1.00, 0.04, 0.04, 0.04, 0.04, 0.88, 0.04, 0.80, 0.04, 0.04, 0.04, 0.27),$$

where `gaming` and `fitness` (already at the floor) stay at the floor (the floor prevents them from going negative). The dismissal effectively confirms that she is not interested in those dimensions, even though the gain is small for already-low coordinates.

3.23.5 Step 5: admin adds a new product

Meanwhile, a merchant adds a new product to the catalog through `/admin/add-product`: "Couscousier en cuivre fait main, modèle traditionnel". The frontend POSTs the description to the FastAPI NLP service. The service runs `mDeBERTa-v3` with twelve hypotheses, takes the entailment logits, applies softmax, and returns the vector, whose top entries are `cooking` 0.68, `home` 0.22, `art` 0.05.

The admin reviews the sliders, leaves them as proposed, and confirms. The product is inserted into Supabase. On Imen's next visit, the new product appears in her recommendations with a similarity score reflecting the modest overlap between her technology-leaning persona and the craft-leaning product.

3.23.6 Observations

The walkthrough illustrates two properties of the design. First, the system is honest: it does not pretend Imen has interests she does not. The first recommendations reflect exactly the three dimensions she picked. Second, the system is patient: the online learning shifts her vector slowly, not catastrophically, so an occasional click on an out-of-pattern item does not destabilise the model.

3.24 Code excerpts

For completeness we include the two core code snippets of the system: the cosine similarity function and the positive-nudge update.

3.24.1 Cosine similarity (TypeScript)

```
function cosineSimilarity(a: number[], b: number[]): number {
  let dot = 0;
  let normA = 0;
  let normB = 0;
  for (let i = 0; i < a.length; i++) {
    dot   += a[i] * b[i];
    normA += a[i] * a[i];
    normB += b[i] * b[i];
  }
  const denom = Math.sqrt(normA) * Math.sqrt(normB);
  return denom === 0 ? 0 : dot / denom;
}
```

3.24.2 Positive nudge with the four guards (TypeScript)

```
function nudgePositive(
  current: number[],
  product: number[],
  rate = 0.08,
  threshold = 0.20,
  floor = 0.04,
): number[] {
  const next = current.map((c, i) =>
    product[i] >= threshold
      ? Math.min(1.0, c + rate * product[i])
      : c
  );
  // Floor: no dim dies.
  const floored = next.map(v => Math.max(floor, v));
  // Renormalise so max = 1.
  const max = Math.max(...floored);
  return floored.map(v => v / max);
}
```

3.24.3 Zero-shot categorisation (Python / FastAPI)

```
from fastapi import FastAPI
from transformers import pipeline
```

```
app = FastAPI()
clf = pipeline(
    "zero-shot-classification",
    model="MoritzLaurer/mDeBERTa-v3-base-mnli-xnli",
)

LABELS = [
    "technology", "fashion", "outdoor",
    "fitness", "home", "cooking",
    "gaming", "reading", "music",
    "travel", "art", "wellness",
]

@app.post("/api/categorize")
def categorize(text: str):
    out = clf(text, candidate_labels=LABELS,
              hypothesis_template="This text is about {}.",
              multi_label=False)
    return {lbl: max(score, 0.04)
            for lbl, score in zip(out["labels"], out["scores"])}
```

The full source is available in the project repository.

3.25 Internationalisation roadmap

The current UI is French-only. A serious Algerian platform must, in the medium term, support three languages: French, Arabic, and English (the last for foreign visitors and for the diaspora). We sketch the roadmap.

3.25.1 String externalisation

Every user-facing string in the codebase is replaced by a key into a translation dictionary. The reference language is French; the other dictionaries are maintained in parallel. The keys are namespaced by page to ease maintenance.

3.25.2 Right-to-left layout

Arabic is written right-to-left. The Next.js root layout sets `dir="rtl"` on the HTML element when the Arabic locale is selected. CSS uses logical properties (`padding-inline-start` instead of `padding-left`) so layouts flip automatically. Icons that imply direction (arrows, chevrons) mirror under RTL.

3.25.3 Number and currency formatting

The Algerian Dinar (DA) is formatted with thousands separators in French style (space-separated thousands, comma decimal). The same number in Arabic uses Arabic-Indic digits when the locale demands it. The `Intl.NumberFormat` API in the browser handles both cases natively.

3.25.4 Date formatting

The Hijri calendar matters in some contexts (Ramadan dates in particular). The frontend uses `Intl.DateTimeFormat` with the appropriate calendar option for Arabic locales.

3.25.5 Persona builder content

The visual archetypes are images and short labels. The labels translate trivially. The images themselves are language-neutral. The persona builder is, fortunately, the simplest UI in the application to localise.

3.25.6 NLP service behaviour

The NLP labeler accepts any language because `mDeBERTa-v3-base-mnli-xnli` is multilingual. No per-language model needs to be deployed.

3.25.7 Why this matters operationally

In Algeria, internationalisation goes beyond UX polish. It is the difference between a platform that serves a French-speaking urban segment and a platform that serves the whole country. The traditional-crafts catalog in particular has a customer base whose first language is more often Arabic than French.

3.26 Cost of operation

A short estimation of the operating cost of the system at modest scale, to ground the design choices in economic reality.

3.26.1 Frontend hosting

Vercel's Hobby plan is free for personal projects; the Pro plan starts around 20 USD per month. At demo scale, Hobby is sufficient. At 100 000 monthly active users, Pro with bandwidth overage would be in the low-hundred-dollars range.

3.26.2 Supabase

Supabase's free tier covers up to 500 MB of database and 5 GB of bandwidth. The seed catalog of 120 products weighs a few hundred kilobytes; the interaction log grows linearly with users but each row is a few dozen bytes. The free tier is sufficient for the demo. The Pro tier starts at 25 USD per month and scales to production loads.

3.26.3 NLP service

Hugging Face Spaces offers free CPU hosting for demos, with a per-call latency around the same 1 to 2 s we measured locally. For higher throughput, a paid T4-GPU instance on Spaces is around 0.60 USD per hour. Because the NLP service is on the admin path only, the throughput requirement is low: a merchant adds a few products per week, not per second.

3.26.4 Total at demo scale

The entire stack runs at zero monthly cost on the free tiers. This is a deliberate consequence of the design choices: the heavy compute is on the client; the server side is a thin Postgres and a thin API; the NLP model is invoked only on admin actions. A more typical recommender architecture (server-side ranking, possibly with a learned-to-rank model) would cost an order of magnitude more at the same user count.

3.27 Sustainability

The energy cost of machine-learning systems has become a measurable axis of design. We add a brief note.

The mDeBERTa-v3 model used for labeling consumes, per inference, in the order of 10^{10} floating-point operations. On a modern CPU this corresponds to a few joules. A typical product upload triggers one inference per categorisation. The energy cost of labeling the entire seed catalog of 120 products is therefore around 200 joules, comparable to running an LED light bulb for twenty seconds.

The recommendation loop, by contrast, runs on the user's device. Its energy cost is borne by the device's battery and is negligible: the cosine over a five-thousand-item catalog is a thousandth of the energy of rendering the page that displays the results.

The combined picture is favourable. Compared to a server-side recommender that runs every query on a cloud GPU, our system shifts the marginal compute to two places where it is cheap: the CPU of a model that runs at most a few hundred times per day (admin path), and the device the user already has powered on (recommendation path). The cumulative carbon footprint of the system is bounded by the admin path and is, in absolute terms, a

rounding error.

3.28 Limitations

We list the honest limitations of the system.

- **Catalog size.** 120 products is a demo dataset, sufficient to showcase the algorithm but not to benchmark accuracy at scale. A production catalog would have 5 000 to 50 000 items.
- **Naive CF.** The popularity proxy is enough for graceful degradation but is not a real collaborative-filtering signal. A real CF (item-item or matrix factorisation) is left to future work.
- **No authentication.** The platform has no user accounts. This is a feature for privacy but a limitation for cross-device sync. An optional account layer would not violate the privacy design, since the persona could remain client-side, with the account used only as a sync mechanism.
- **No A/B harness.** We did not implement experimentation infrastructure. The values of α, the single learning rate and the threshold were tuned by inspection. A production deployment would want a real A/B testing layer.
- **CPU-only NLP.** The model runs on CPU, with a latency of one to two seconds per categorisation. Acceptable for the admin path, it is inadequate for live ranking, which is why the NLP service is off the request path. A GPU deployment on Hugging Face Spaces would reduce latency by an order of magnitude if needed.
- **Brute-force KNN.** The $\mathcal{O}(N \cdot D)$ scan is fine for $N \leq 5\,000$ but eventually becomes the bottleneck. The `pgvector` migration described in Chapter 1 is the standard answer.
- **French-only UI.** The interface text is in French; the catalog supports Arabic terms but the shell does not. A full Arabic/English internationalisation is future work.
- **No image embeddings.** The system uses only the textual description for content vectors, ignoring the product image. For Algerian crafts in particular, where descriptions are short and visuals carry meaning, image embeddings (CLIP-style) would be a significant uplift.
- **No session model.** The persona aggregates over all interactions; it does not distinguish a morning session from an evening session. A session-based recommender (GRU4Rec) would capture this dynamic.

3.29 Conclusion

This chapter detailed the complete architectural design and empirical validation of AlgerieShop IA. By implementing a unified 12-dimensional vector space, a lightweight client-side matching engine, an online-learning rule with four stability guards, a hybrid blend that gracefully degrades, an automated zero-shot multilingual tagging microservice, and a data model that respects on-device privacy, we have created an efficient, privacy-preserving recommender system tailored to the unique characteristics of Algerian e-commerce. The experimental evaluation confirmed the theoretical predictions: sub-20 ms client-side latency even on low-end hardware, 91.5 % zero-shot accuracy on mixed-language inputs, a 2.4× cold-start uplift over popularity, and stable online learning over hundreds of interactions. The general conclusion that follows takes stock of the whole work and lays out the roadmap from here.

General Conclusion

This thesis presented the complete design and implementation of AlgerieShop IA, an express-onboarding, explainable, multilingual recommender system built specifically for the operational patterns and cultural demands of the Algerian e-commerce marketplace.

By grounding our architecture in a unified 12-dimensional vector space, we bridged the gap between raw semantic data and personalised user profiles. The system solves the three critical challenges identified at the start of our research.

1. **The cold-start problem.** Resolved via an immediate, non-intrusive visual persona selection phase that sets up initial preferences in under three clicks. We showed empirically that this produces a 2.4× relevance uplift over a popularity baseline.
2. **Language code-switching.** Managed through a zero-shot cross-lingual NLI pipeline based on `mDeBERTa-v3`, which interprets mixed French and Arabic item descriptions with 91.5 % top-1 accuracy on the hand-labelled test set.
3. **User privacy.** Enforced by performing all user vector drifting and KNN ranking operations directly inside the client browser's `localStorage`, so that no user profile ever leaves the device. This design satisfies the data-minimisation principle of Algerian Law 18-07 by construction.

The experimental evaluations demonstrate that local, decentralised recommendation is not only technically feasible but computationally superior for mobile-first environments, maintaining latency times under 20 milliseconds even on low-end Cortex-A53 processors.

Contribution. The individual ingredients of our system (KNN, cosine similarity, zero-shot NLI, hybrid CB+CF) are standard. The contribution of this thesis is the *end-to-end system design*: combining a visual cold-start, an NLP-generated semantic vector space, an in-browser KNN, and real-time online learning into a coherent, privacy-first product that ships, runs on a laptop, and remains explainable from the user's screen all the way down to the model logits. The Algerian contribution is the multilingual, all-local catalog and a zero-shot labeler that handles it without training data, delivered through a cold-start, on-device design suited to the local context.

Honest limitations. The catalog is small (120 products), the collaborative signal is a naive popularity proxy, there is no authentication and no A/B testing harness, the NLP model is CPU-only and slow enough to keep it off the ranking path, and the UI is currently French-only. None of these are research limitations; they are deliberate scope choices for

a thesis demonstration. A production-ready version would address each of them without rewriting the core.

Future work. Five directions naturally extend this work:

1. **Move ranking to pgvector + HNSW.** Replace the in-memory cosine loop with a Postgres `vector` column and an approximate-nearest-neighbour index; scaling beyond a few thousand products becomes a configuration question instead of a code question.
2. **True item-item collaborative filtering.** Replace the popularity proxy with a co-purchase matrix and implicit-feedback matrix factorisation, and re-run the hybrid experiment.
3. **Session-based and time-decayed signals.** GRU4Rec or SASRec can capture short-horizon dynamics without rewriting the whole stack.
4. **Image embeddings.** Fuse a CLIP-style image vector with the NLP text vector so that visually similar products share part of their representation, which would help under-described craft items.
5. **Full internationalisation.** A French / Arabic / English UI with RTL layout for Arabic, currency formatting per locale, and an Arabic-first persona builder.

Closing word. A modern recommender does not need to be a black-box neural network running on a hyperscaler's GPU farm. With a small vector space, a well-chosen pretrained NLI model, a cosine query, and the discipline to keep the user's data on the user's device, we can ship a system that is at once accurate, explainable, multilingual and respectful of privacy. This thesis is our argument, in code, for that position.

Bibliography

[1] C. C. Aggarwal. *Recommender Systems: The Textbook*. Springer, 2016.

[2] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331-370, 2002.

[3] R. Burke. Hybrid web recommender systems. In *The Adaptive Web*, Springer, 2007.

[4] A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *EMNLP*, 2018.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[6] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61-70, 1992.

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. S. Chua. Neural collaborative filtering. In *Proceedings of WWW*, 2017.

[8] P. He, X. Liu, J. Gao, and W. Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. *arXiv:2111.09543*, 2021.

[9] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *ICLR*, 2016.

[10] HuggingFace. *Transformers: state-of-the-art machine learning for PyTorch, TensorFlow and JAX*. `https://huggingface.co/docs/transformers`, accessed 2026.

[11] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018.

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30-37, 2009.

[13] M. Laurer, W. van Atteveldt, A. Casas, and K. Welbers. Less annotating, more classifying. Model card: `MoritzLaurer/mDeBERTa-v3-base-mnli-xnli`, 2022.

[14] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76-80, 2003.

[15] Next.js team. *Next.js: The React framework for the Web*. `https://nextjs.org`, accessed 2026.

[16] Pgvector contributors. *pgvector: open-source vector similarity search for Postgres*. https://github.com/pgvector/pgvector, accessed 2026.

[17] S. Ramirez. *FastAPI: a modern, fast Web framework for building APIs with Python*. https://fastapi.tiangolo.com, accessed 2026.

[18] République Algérienne Démocratique et Populaire. Loi n° 18-05 du 10 mai 2018 relative au commerce électronique. *Journal Officiel*, 2018.

[19] République Algérienne Démocratique et Populaire. Loi n° 18-07 du 10 juin 2018 relative à la protection des personnes physiques dans le traitement des données à caractère personnel. *Journal Officiel*, 2018.

[20] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *CSCW*, 1994.

[21] F. Ricci, L. Rokach, and B. Shapira (eds.). *Recommender Systems Handbook*, 3rd edition. Springer, 2022.

[22] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, 1999.

[23] A. Schein, A. Popescul, L. Ungar, and D. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.

[24] Supabase team. *Supabase: The open-source Firebase alternative*. https://supabase.com, accessed 2026.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[26] A. Williams, N. Nangia, and S. Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*, 2018.

[27] W. Yin, J. Hay, and D. Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP-IJCNLP*, 2019.

[28] Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *Foundations and Trends in Information Retrieval*, 14(1):1-101, 2020.

ملخص

AlgerieShop IA نظام توصية بالمنتجات قائم على الذكاء الاصطناعي، مُصمَّم للتجارة الإلكترونية في الجزائر. يُمثَّل كل مستخدم وكل منتج في فضاء اهتمامات مشترك من 12 بُعداً، يُولَّد تلقائياً من الأوصاف متعددة اللغات (الفرنسية والعربية) بنموذج معالجة لغة طبيعية بأسلوب zero-shot يُدعى mDeBERTa-v3 دون أي بيانات تدريب. تعتمد التوصيات على بحث الجيران الأقرب (KNN) بتشابه جيب التمام، مدموجاً مع إشارة شعبية، ويتطوّر ملف المستخدم بعد كل تفاعل. يُنفَّذ الترتيب بالكامل داخل المتصفح، فلا يغادر ملف المستخدم جهازه أبداً، وتبقى كل توصية قابلة للتفسير.

الكلمات المفتاحية: نظام توصية، KNN، تشابه جيب التمام، معالجة اللغة zero-shot، التعلّم المستمر، القابلية للتفسير، الخصوصية.

Résumé

AlgerieShop IA est un système de recommandation de produits fondé sur l'intelligence artificielle, conçu pour le e-commerce algérien. Chaque utilisateur et chaque produit sont représentés dans un même espace d'intérêts à 12 dimensions, généré automatiquement à partir de descriptions multilingues (français/arabe) par un modèle de traitement du langage en zéro-shot (mDeBERTa-v3), sans aucune donnée d'entraînement. Les recommandations reposent sur une recherche des K plus proches voisins par similarité cosinus, combinée à un signal de popularité, et le profil de l'utilisateur évolue après chaque interaction. Le classement s'exécute entièrement dans le navigateur : le profil ne quitte jamais l'appareil, et chaque recommandation est explicable.

Mots-clés : système de recommandation, KNN, similarité cosinus, NLP zéro-shot, apprentissage en ligne, explicabilité, vie privée.