International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 12-No.2 (2026) pp. 399-404
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

A Practitioner's Runbook for Building Scalable Recommendation Systems in Large-Scale E-Commerce Platforms

Jay Bankimchandra Desai*

San Jose State University, USA
* **Corresponding Author Email:** jaydesai.reach@gmail.com- **ORCID:** 0000-0002-5247-7643

Article Info:

DOI: 10.22399/ijcesen.5231
Received : 25 March 2026
Revised : 05 May 2026
Accepted : 10 May 2026

Keywords

E-Commerce Platforms,
Practitioner,
Recommendation Systems,
Runbook,
Scalable

Abstract:

Personalized recommendation systems sit at the heart of large-scale e-commerce platforms, directly shaping how users discover products, engage with content, and return over time. Delivering high-quality recommendations is not a secondary concern; it is a primary driver of retention, satisfaction, and sustained platform growth. Building these systems at scale demands careful coordination of several engineering components, from capturing recent user behavior and determining interest categories to constructing features for AI/ML inference pipelines and closing the loop through user interaction feedback integration, all while operating under strict latency and accuracy requirements. The strength of a recommendation system depends as much on how its components are architecturally organized as on the sophistication of the models running inside it. This article covers the end-to-end design of recommendation pipelines across four areas. First, context building draws on user interactions and trending signals to assemble the inputs the algorithm needs. Second, the online inferencing algorithm handles low-latency candidate retrieval, AI/ML model ranking, and post-processing feedback that feeds offline training. Third, scaling and reusability strategies allow generic pipelines to serve multiple pages, surfaces, and category-specific experiences through caching, unified interfaces, and deduplication. Fourth, broader implications are examined, including how well-designed systems build user trust, reduce operational overhead, and support healthy ecosystem diversity. Recommendation systems function as strategic infrastructure rather than isolated technical components. The architectural decisions made during their design shape day-to-day performance as well as the long-term resilience and competitive position of the platform.

1. Introduction

Personalized recommendation modules are a core requirement for large-scale e-commerce platforms, with direct consequences for user retention, engagement, and revenue growth. Relevant recommendations shape the user journey at every stage, starting from the first product a user discovers through browsing, purchase, and the likelihood of returning. Platforms that consistently serve poor or generic suggestions push users toward alternatives, and the cost of that attrition compounds over time [3]. Getting recommendations right at production scale is harder than it looks. The pipeline behind a single recommendation involves analyzing what a user has recently browsed, inferring current interests, constructing the features and embeddings that feed AI/ML inference models, running those models, and then evaluating the results after the fact. Each of these steps runs under strict latency budgets, and a failure at any point degrades the quality of what the user sees. A retrieval layer that is fast but feeds poor candidates into ranking produces bad results just as reliably as a ranking model that is accurate but receives its inputs too slowly to meet response time requirements [4].

Across a large platform, the same recommendation infrastructure needs to serve multiple surfaces simultaneously, including the homepage, product detail pages, and category views, without producing inconsistent or contradictory outputs. Caching intermediate outputs like category rankings lets different modules share computed context rather than rebuilding it independently, and deduplication layers catch repetitions before they reach the user [6]. These are not optional refinements; they are the mechanisms that make platform-wide

personalization coherent rather than fragmented. The deeper engineering challenge is that recommendation pipelines sit at the intersection of large-scale AI/ML inferencing and distributed systems design. Serving personalized results at internet scale requires both model quality and infrastructure discipline, and neither compensates for deficiencies in the other. Platforms that treat recommendation quality as purely a modeling problem, without investing in the pipeline architecture that supports it, find that performance holds at low traffic and breaks under the conditions where it matters most [2].

2. Building Context for the Algorithm

2.1 Recently Visited Items, Historical Clicks, and Transactional Events

The foundation of any online recommendation inference system is user behavior data, which provides the contextual basis for AI/ML models to infer user intent. Within e-commerce platforms, this context is typically derived from recent user interactions such as items visited, products clicked historically, and transactional records including purchases. During an online inference request, the system generally queries multiple data sources to construct this context. Recent interactions are indicative of short-term intent, whereas historical clicks and purchases reflect longer-term preference signals. It is essential to balance these signals; for instance, items that a user has recently purchased should ordinarily be excluded from recommendations unless historical patterns demonstrate a tendency toward repeat purchasing. Furthermore, the algorithm must avoid reliance on stale data, as outdated signals can substantially diminish recommendation relevance. This stage requires non-trivial pattern analysis and frequently constitutes the backbone of a scalable and effective recommendation system.

2.2 Current Most-Watched or Trending Items

Alongside personalized signals, effective recommendation systems often include trending or popular items to reflect current interests on the platform. This requires integration with offline or near-real-time systems that continuously process interaction events from all users. These systems examine aggregate behavior such as clicks, views, and purchases to identify items or categories that are trending. The online inference algorithm can then query this trend layer and combine trending signals with personalized context. This blended approach balances personalization with discovery, keeping users engaged with both familiar content and new items. Incorporating trending signals strengthens relevance and supports sustained engagement in dynamic environments.

2.3 Determining Recommendation Categories

The system reviews user behavior and trending patterns to establish which item categories should be prioritized in recommendations. Because user interactions often span multiple categories, and trending signals may introduce additional candidates, the data needs to be normalized and aggregated into a coherent representation of user interest. One method of dealing with category signals is to develop a weighted and directed acyclic graph, where each node represents a category and the weight of the nodes represents the level of interest of the user in the category. This graph-based approach is considered the core mechanism for category determination, as it captures both frequency and recency of user interactions. Using the graph, it is possible to get a list of categories in order of user interest. The edges represent the relationship, and navigation is done at nodes with the highest weight. Node weights are determined by factors such as how often and how recently users interact with a category. Using the graph, it is possible to get a list of categories in order of user interest. Another approach is to feed raw category signals into an AI/ML model trained on historical user behavior to predict the most likely next category of interest. Overall, this stage of category determination also supports multi-round recommendations, where successive modules on a page or similar modules across different pages target different interest levels or categories in sequence.

3. Online Inferencing Algorithm

3.1 Title Generation and Low-Latency Item Retrieval

After the target categories are determined, the system proceeds to build the recommendation modules. Each module includes a descriptive title that conveys the nature of its suggestions, with titles often generated by downstream services to ensure semantic relevance to the chosen categories. At the heart of each module is item retrieval. The system fetches candidate items from the identified categories in an efficient manner, integrating with services that monitor top-performing or highly engaged products. These candidates then serve as the input for the AI/ML-based ranking stage. Efficient retrieval helps achieve low latency and

meet the stringent latency requirements of real-time inference.

3.2 Complex AI/ML Model Integration

While retrieval ensures category alignment, relevance requires further modeling of user behavior and engagement. Advanced AI/ML systems rank candidate items according to forecasted engagement, using behavioral data and item metadata to predict the likelihood of interaction between a user and each candidate. The goals of optimization are platform- and use-case-specific, such as click-through rate, add-to-cart rate, retention in the long term, or a mix of both with priority based on business concerns. Reinforcement learning strategies dynamically trade off short-term action with long-term gratification, changing recommendations based on current feedback instead of a fixed model [11]. Practically, the results of the various models are often pooled together to give strong rankings, which consider various factors in user behavior. Each of the click prediction models and purchase prediction models, for example, might provide a weighted signal to the end ranking decision. Platforms should trade off model complexity with inference speed and use lightweight models to execute real-time ranking and serving and train and evaluate more complex models offline [8]. This distinction further guarantees that accuracy is not compromised on behalf of responsiveness or vice versa.

3.3 Post-Processing and Offline Model Training

Internet inference is just one of the stages of the recommendation cycle. The analysis of user engagement with recommended objects and transferring such feedback to the offline training pipelines is also crucial in maintaining the quality of models in line with the expected behavior. Once interactions have been recommended, the system logs the interaction events asynchronously to not introduce latency to the serving path. The positive interaction strengthens the prediction and validates the predictive signs, and negative interaction informs the models to down-weight or shun some categories of output [12].The feedback is gathered in batch pipelines on a daily basis or on shorter periods based on the amount of interactions, retraining the models to capture new preferences and changing catalog behavior. Experimentation can also be facilitated by offline training, where new model structures or sets of features can be tested by controlled tests without interruption to live inference serving. This process of online serving and offline retraining is repeated to guarantee flexibility with time. Scalable recommendation systems are characterized by continuous feedback loops that allow them to adapt to shifts in user behavior and catalog behavior instead of deteriorating with changes in conditions [3].

4. Scaling and Reusability

4.1 Same Algorithm Across Multiple Pages and Recommendation Modules

The initial module on a page can be focused on the strongest interest category of the user; the following modules on the page can tell the user about the secondary interests, making sure the user is not shown only a narrow range. Due to the combination of various data sources and the implementation of sophisticated AI/ML models in this pipeline, re-extraction of the complete pipeline per module would be both computationally expensive and operationally challenging to support at scale. One common optimization is to store intermediate results, e.g., category rankings and candidate item lists, in distributed in-memory stores. Later modules can reuse this cached context and merely rerun the final ranking stage with module-specific parameters. Benchmarking evidence in microservice environments has shown that careful allocation of workloads across storage layers directly affects the latency and throughput profile of multiple consumer pipelines, which often orchestrate downstream processes in the recommendation workflow [8]. Close attention to the value of cache time-to-live values is needed so that the stale recommendations are not served and the maximum performance benefit of caching is achieved [7]. The approach to reuse minimizes the latency, the infrastructure expenses, and the uniformity of modules that might be found on the same page or on various surfaces of the platform [6].

4.2 Unified Interface for Recommendation Output

In case multiple pages are served by a single recommendation algorithm by various microservices, it is necessary to present a single output interface to ensure uniformity and to minimize the engineering cost. Downstream services can make a single query to a recommendation service and get a coherent, structured reply to their query without having to bother with the internal machinery of the pipeline. This prevents the need to repeat the integration logic in different page-specific microservices and

makes continued maintenance easier, as well as ensures that the behavior of recommendations is not different among the various surfaces that request it. A single interface also allows better scalability, in that new modules or services can be added without the need to add individual pipelines or even individual integration effort on behalf of the new consumer. Platforms attain consistency of personalization in addition to lowering the engineering burden that would otherwise be created by fragmented pipeline design by locating the result of recommendations in a single layer of service [2].

4.3 Scaling Across Multiple Categories

The recommendation algorithm can be repurposed for the category-related experiences without the need to implement it for each vertical separately. For example, the same pipeline can be shared by different page entities, with configuration limiting recommendations to the categories relevant to each context instead of altering the code. The reuse reduces engineering overhead, centralizes the development of algorithms, and can enable the system to scale effectively across specializations of experiences that have a specific target audience segment. In category-specific scaling, personalization will be applicable to a particular context but will also use the same architecture as is used to generate recommendations on a platform-wide scale. It is also modular and allows the team to add a product domain or audience segment onto the pipeline without having to re-architect the system itself [5]. This reuse and specialization balance is a realistic need of platforms at internet scale, where the cost of having different pipelines per category would soon become unsustainable.

4.4 Deduplication of Interests

It is also important to avoid repetition of similar or very similar products displayed in more than one module or page to ensure good quality of recommendations and end-user trust. By giving repeated suggestions, the system encourages users to believe that the system does not listen to what has already been experienced, and this reduces the level of trust in the platform's personalization features. A unified deduplication service is capable of monitoring newly delivered recommendations within the platform and preventing overlap items before they are delivered to the user to allow wider item exposure and enhance the general exploration [6]. Diverse products are also promoted by deduplication because the users are not forced to stick to the same products in different modules or sessions, as this fosters their curiosity to use new products. Through a log of consumed products and implementation of filtering protocols at the delivery point, platforms are able to produce suggestions that are new and differentiate over time. This process enhances trust among users and facilitates continuous usage, as it gives diversity to customized outputs, especially to those users who frequent the site and would have otherwise seen the same content over and over.

5. Broader Implications

When suggestions arrive in a way that is timely, varied, and relevant, users sense that the platform recognizes and responds to their interests. A system that is not designed effectively and results in stale or repetitive recommendations leads to a degradation in user trust over time, which ultimately impacts the engagement levels that personalization is meant to improve [6]. At the system level, an effectively designed recommendation system reduces the complexity in operations. This is especially important from an engineering standpoint. This enables the system to maintain personalization behavior in all modules and services without having to implement personalization on each surface. The general idea here is that a recommendation system is not a technical feature but a strategic feature whose implementation and design are critical to building trust, efficiency, and competitiveness. A platform that focuses on architectures that are scalable, reusable, and adaptable is better positioned to create consistent and engaging user experiences while growing in a dynamic and competitive environment. The integration of caching strategies, deduplication strategies, and category scaling strategies enables the adaptation of the recommendation systems without compromising the responsiveness and consistency of the systems [7].The inclusion of diversity strategies enables the promotion of popular items and niche items alike, thus preventing the concentration of the recommendation strategies on a selected group of items that have a higher level of engagement from the users. This enables the sustainable growth of the platform by encouraging the exploration of different items and the diversification of the interests of the users beyond the scope that the concentrated recommendation strategies would otherwise provide [5]. The feedback loops enable the adaptation of the systems by continually incorporating the feedback from the users into the offline training process, thus ensuring that the systems do not deteriorate with the passage of time and the changes in the behavior of the users [3].

Table 1. *Role of Recommendation Pipelines in E-Commerce [3, 4]*

Key Focus	Mechanisms / Strategies	Outcomes / Implications
Importance of personalization	AI/ML inference, distributed systems, caching	User retention, engagement, revenue growth
Consistency across platform	Shared caching, deduplication layers	Prevents fragmented personalization

Table 2. *Context Construction for Personalization [1, 6]*

Subsection	Key Focus	Mechanisms / Strategies	Outcomes / Implications
Recently Visited Items	Behavioral signals	Low-latency logs, distributed access	Real-time personalization under load
Trending Items	Collective signals	Event-driven pipelines, trend inference	Discovery value, relevance during shifts
Recommendation Categories	Category targeting	Aggregation, cross-category affinities, historical models	Comprehensive recommendation experience

Table 3. *Online Inferencing and Feedback Loops [6, 9]*

Subsection	Key Focus	Mechanisms / Strategies	Outcomes / Implications
Title Generation & Retrieval	Latency control	Distributed caching, optimized indexing	Fast candidate retrieval
Complex AI/ML Integration	Engagement prediction	Reinforcement learning: lightweight vs complex models	Balance of accuracy and responsiveness
Post-Processing & Offline Training	Feedback incorporation	Asynchronous logging, batch retraining	Continuous adaptation, model flexibility

Table 4. *Scaling and Reusability [5, 6]*

Focus Area	Strategy	Outcome
Reuse	Cached results, microservice reuse	Lower latency, reduced cost
Interface	Single output layer	Consistent personalization
Categories	Config-based reuse	Flexible scaling
Deduplication	Filtering protocols	Trust, diversity

6. Conclusions

As discussed in this article, building a recommendation system for a high-scale website requires more than just powerful AI/ML algorithms; rather, there is a need for a perfect blend of data pipelines, inference logic, caching mechanisms, and feedback integration throughout all stages of the system. The design of each module needs to be focused on scalability and flexibility rather than considering them as afterthoughts. Moreover, the use of universal design principles and interfaces, along with mechanisms for eliminating redundancy, helps in ensuring that the system for building a recommendation system works efficiently and coherently across all modules, pages, and even various category-specific experiences. The use of offline retraining for continuous learning also helps in enhancing flexibility rather than becoming progressively less accurate as the situation demands. A well-designed system for building a recommendation system is one that ensures a perfect blend of personalization and discovery, efficiency and responsiveness, and innovation and reliability. Such a system can provide a large-scale website with a chance to deliver a high-quality experience that becomes more efficient as the situation demands rather than becoming progressively less efficient as the situation demands.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

[1] Sungju Lee and Taikyeong Jeong, "Large-Scale Distributed System and Design Methodology for Real-Time Cluster Services and Environments," Electronics, vol. 11, no. 23, Dec. 2022. https://www.mdpi.com/2079-9292/11/23/4037

[2] Maarten van Steen, Guillaume Pierre, and Spyros Voulgaris, "Challenges in Very Large Distributed Systems," Journal of Internet Services and Applications, vol. 3, pp. 59–66, Nov. 2011. https://link.springer.com/article/10.1007/s13174-011-0043-x

[3] Praveen Kumar Donta et al., "Exploring the Potential of Distributed Computing Continuum Systems," Computers, vol. 12, no. 10, Oct. 2023. https://www.mdpi.com/2073-431X/12/10/198

[4] Elias Del-Pozo-Puñal et al., "Hierarchical and Distributed Data Storage for Computing Continuum," Future Generation Computer Systems, vol. 174, Jun. 2026. https://www.sciencedirect.com/science/article/pii/S0167739X25002262

[5] Asif Mehmood, Mohammad Arif, and Faisal Mehmood, "Towards a Unified Digital Ecosystem: The Role of Platform Technology Convergence," Electronics, vol. 14, no. 24, Dec. 2025. https://www.mdpi.com/2079-9292/14/24/4787

[6] Irene Kilanioti et al., "Towards Efficient and Scalable Data-Intensive Content Delivery: State-of-the-Art, Issues and Challenges," in High-Performance Modelling and Simulation for Big Data Applications, Springer, pp. 88–137, Mar. 2019. https://link.springer.com/chapter/10.1007/978-3-030-16272-6_4

[7] Cornelia A. Győrödi et al., "Performance Analysis of NoSQL and Relational Databases with CouchDB and MySQL for Applications' Data Storage," Applied Sciences, vol. 10, no. 23, Nov. 2020. https://www.mdpi.com/2076-3417/10/23/8524

[8] Nenad Pantelic et al., "Benchmarking SQL and NoSQL Persistence in Microservices Under Variable Workloads," Future Internet, vol. 18, no. 1, Jan. 2026. https://www.mdpi.com/1999-5903/18/1/53

[9] Maryam Abbasi et al., "Revisiting Database Indexing for Parallel and Accelerated Computing: A Comprehensive Study and Novel Approaches," Information, vol. 15, no. 8, Jul. 2024. https://www.mdpi.com/2078-2489/15/8/429

[10] Maryam Abbasi et al., "Optimizing Database Performance in Complex Event Processing through Indexing Strategies," Data, vol. 9, no. 8, Jul. 2024. https://www.mdpi.com/2306-5729/9/8/93

[11] Najla Sassi and Wassim Jaziri, "Efficient AI-Driven Query Optimization in Large-Scale Databases: A Reinforcement Learning and Graph-Based Approach," Mathematics, vol. 13, no. 11, May 2025. https://www.mdpi.com/2227-7390/13/11/1700

[12] Paraskevas Koukaras, "Data Integration and Storage Strategies in Heterogeneous Analytical Systems: Architectures, Methods, and Interoperability Challenges," Information, vol. 16, no. 11, Oct. 2025. https://www.mdpi.com/2078-2489/16/11/932