A Silent Multi-Agent Recommendation Engine with Graph-Augmented Memory for Immersive B2B E-Commerce

Davi Silva Eleuterio
CeDRI, SusTEC, Instituto Politécnico de Bragança, Portugal

Pedro Filipe Oliveira
CeDRI, SusTEC, Instituto Politécnico de Bragança, Portugal

Paulo Matos
CeDRI, SusTEC, Instituto Politécnico de Bragança, Portugal

Abstract

Currently, e-commerce platforms are transitioning from static 2D catalogues to immersive environments, requiring recommendation systems to process complex behavioural signals. Traditional collaborative filtering struggles with cold-start scenarios and lacks semantic understanding of product catalogues. This paper presents a *silent* multi-agent recommendation engine developed for the VIMOS (Virtual Integrated Market Online Shopping) project, designed to operate without explicit textual input. Orchestrated via a LangGraph Directed Acyclic Graph, the system utilises Agentic Retrieval-Augmented Generation (RAG) to dynamically synthesise behavioural signals, dense vector retrieval (Qdrant), and knowledge graph traversal (Apache AGE). A locally hosted Llama 3.1 8B model autonomously extracts implicit user preferences to update a persistent graph database and generates deterministic, human-readable explanations for each recommendation. A two-shot verifier engineering pattern is introduced to resolve the structural incompatibility between dynamic candidate sets and static graph execution. The architecture is validated using a controlled synthetic dataset of B2B industrial products – sufficient for functional validation of all agent routing paths – demonstrating high retrieval accuracy, strict constraint adherence (stock and availability verification), and sub-4-second end-to-end latency (when the generative semantic justification step is bypassed) with no external API dependencies.

2012 ACM Subject Classification Applied computing → Document analysis; Computing methodologies → Information extraction; Computing methodologies → Machine learning

Keywords and phrases multi-agent systems, agentic RAG, product recommendation, LangGraph

Digital Object Identifier 10.4230/OASIcs.SLATE.2026.15

Funding This work was supported by FCT – Fundacão para a Ciência e Tecnologia, I.P. by projects: CeDRI, UID/05757/2025 (DOI: `https://doi.org/10.54499/UID/05757/2025`) and UID/-PRR/05757/2025 (DOI: `https://doi.org/10.54499/UID/PRR/05757/2025`);
SusTEC, LA/P/0007/2020 (DOI: `https://doi.org/10.54499/LA/P/0007/2020`) and by NORTE2030-FEDER-02214300 – "VIMOS – Virtual Integrated Market Online Shopping", through the Fundo Europeu de Desenvolvimento Regional (FEDER) and Programa Regional do Norte (Norte 2030) do Portugal 2030.

1 Introduction

The contemporary e-commerce landscape is undergoing a profound structural transition, shifting from traditional, static 2D product catalogues toward immersive, spatial shopping environments leveraging 3D visualisation and Augmented Reality (AR) [10]. This shift is particularly critical in the Business-to-Business (B2B) sector – such as the dental and industrial tool markets – where products require strict technical matching, precise compatibility checks,

© Davi Silva Eleuterio, Pedro Filipe Oliveira, and Paulo Matos;
licensed under Creative Commons License CC-BY 4.0

and a deep understanding of complex specifications. While conventional recommendation systems rely heavily on standard collaborative filtering, these legacy approaches often struggle with cold-start scenarios and lack the semantic reasoning required to understand the intricate features of specialised B2B catalogues.

Simultaneously, the field of Artificial Intelligence has seen a paradigm shift driven by Large Language Models (LLMs) and the evolution of Retrieval-Augmented Generation (RAG) architectures [6]. By grounding LLMs in external, up-to-date knowledge bases, RAG mitigates hallucinations and enables deep semantic reasoning over technical documents – a paradigm currently evolving into Agentic RAG architectures where autonomous agents orchestrate complex retrieval workflows [14]. However, integrating these conversational agents into an immersive 3D shopping experience presents a user experience challenge: forcing a user to interrupt their spatial exploration to type explicit queries can break immersion and introduce friction. There is a pressing need for *silent* recommendation systems that can operate implicitly, leveraging rich behavioural signals – such as the time spent analysing a 3D mesh or interacting with a product hotspot – without requiring any conversational input.

To address these challenges, this paper presents the design and implementation of a silent multi-agent product recommendation assistant developed within the scope of the Virtual Integrated Market Online Shopping (VIMOS) project. VIMOS aims to replace static B2B e-commerce with a smart 3D visualisation platform driven by Agentic RAG. Three specialised assistants are planned for the platform: the silent recommendation engine described in this paper, a semantic product search assistant, and a 3D spatial context assistant. Operating entirely in the background during page loads, the proposed architecture is orchestrated via LangGraph, utilising a multi-agent system to autonomously evaluate user profile sufficiency [12]. It blends deterministic collaborative filtering with semantic vector search. Utilising a locally hosted Llama 3.1 8B model, the system not only verifies the strict availability of candidate products but also synthesises deterministic, human-readable semantic justifications for each recommendation. Furthermore, it asynchronously updates a persistent Apache AGE knowledge graph to maintain a long-term memory of implicit user preferences.

This work builds upon a line of research by the same group on LLM-powered assistants for specialised domains [5], now extending the scope from reactive conversational RAG to a proactive, graph-augmented, fully local multi-agent pipeline.

The primary contributions of this work are:

1. A fully implemented LangGraph Directed Acyclic Graph for non-conversational, silent product recommendation from purely behavioural signals.
2. The *two-shot verifier pattern*: a practical engineering solution tailored to state initialization constraints in LangGraph pipelines, enabling post-retrieval validation when candidate identifiers are unknown at graph construction time.
3. A session-adaptive four-mode routing strategy guaranteeing non-empty recommendations across authenticated, partially profiled, anonymous, and cold-start sessions.
4. An asynchronous Memory Agent that writes user preference edges to an Apache AGE knowledge graph without adding to response latency.
5. Experimental validation of five functional criteria across 20 synthetic B2B test sessions, achieving 100% product availability compliance and sub-4-second end-to-end latency.

The remainder of this paper is organised as follows. Section 2 reviews related work on RAG evolution and recommender systems. Section 3 details the multi-agent topology, including behavioural preference scoring and session routing. Section 4 discusses the Verifier Agent and the asynchronous Memory Agent. Section 5 presents the experimental setup, synthetic dataset, validation results, and latency analysis. Finally, Section 6 outlines conclusions and future directions.

2 Related Work

The architecture of the proposed recommendation engine lies at the intersection of evolving e-commerce recommender systems, advancements in Large Language Models (LLMs), and the paradigm shift toward Multi-Agent Retrieval-Augmented Generation (Agentic RAG).

2.1 Evolution of Recommendation Systems: From Filtering to Semantic Search

Traditional e-commerce platforms have historically relied on standard Collaborative Filtering (CF) and Content-Based Filtering. These legacy systems were heavily dependent on static demographic segmentation, identifying users with overlapping purchase histories to propagate preferences [7]. However, as digital commerce transitions towards immersive, spatial environments [10], the state of the art has fundamentally shifted toward real-time behavioural analysis.

To handle this increased complexity, modern systems have moved beyond basic Natural Language Processing (NLP) rules [11]. Contemporary hybrid architectures combine traditional paradigms with deep learning techniques and session-level signals (e.g., dwell time, 3D interaction type). The integration of dense vector embeddings into these pipelines represents the most recent evolution: a pre-trained embedding model encodes both product descriptions and behavioural patterns into a shared semantic space, enabling retrieval by semantic proximity. The present work adopts this advanced hybrid paradigm, merging graph-based collaborative filtering via Apache AGE with dense vector search over Qdrant.

2.2 Agentic RAG and LLMs in Information Retrieval

To overcome the semantic limitations of traditional recommender systems, Retrieval-Augmented Generation (RAG) became the industry standard to ground LLMs in external knowledge. RAG architectures have evolved through four recognised generations: Naive RAG, Advanced RAG, Modular RAG, and Agentic RAG [6, 15, 14]. Early iterations ("Naive RAG") operated on strictly linear, retrieve-then-read pipelines, while "Advanced RAG" injected pre- and post-retrieval optimisations such as query rewriting and cross-encoder reranking.

Recently, there has been a profound paradigm shift towards "Agentic RAG". In this architecture, LLMs transition from fast, linear "System 1" processing to autonomous "System 2" cognitive reasoning [8]. Elevating the LLM to an autonomous orchestrator operating within a ReAct (Reasoning and Acting) loop, these systems dynamically route across heterogeneous data sources. While recent implementations of vanilla RAG have demonstrated high retrieval success in reactive, conversational interfaces for specialised domains [5], the field is rapidly advancing. The current frontier demands a shift from these reactive conversational pipelines to proactive, graph-augmented Multi-Agent Systems (MAS) capable of interpreting implicit, non-verbal behavioural signals.

2.3 Multi-Agent Orchestration and Deterministic Logic

The distinction between a single agentic loop and a true MAS is formalised by Sapkota et al. [13], who define MAS as a system where multiple specialised sub-agents collaborate, decompose tasks, and cross-verify outputs. This topology is crucial for enforcing deterministic business logic and actively identifying hallucinations in technical domains [12].

A prominent architectural pattern that has emerged to improve efficiency is **Adaptive RAG**, which interposes a complexity classification logic that routes queries dynamically, bypassing heavy agentic pipelines for simple factual queries to optimise global latency. The

present work adopts a related principle: the Orchestrator Agent deterministically routes sessions by profile completeness, avoiding unnecessary computation for anonymous or cold-start sessions.

To safely orchestrate these stateful pipelines, modern architectures rely on state graph frameworks such as LangGraph [1]. LangGraph is a Python library built on top of LangChain that models an agent pipeline as a *directed acyclic graph* (DAG): nodes represent individual processing steps (agents or tools), directed edges encode execution order, and conditional edges implement branching logic at runtime. A shared typed state dictionary (`AgentState`) flows through every node, allowing agents to read from and write to a common data structure without direct coupling. This explicit graph abstraction allows for deterministic routing rules alongside probabilistic LLM reasoning, guaranteeing robust and explainable recommendations.

3 System Architecture

This section describes the multi-agent pipeline that powers the silent recommendation engine. The pipeline is composed of eight specialised agents, each responsible for a distinct processing step, coordinated by a shared execution graph. Section 3.4 details how raw behavioural signals are transformed into preference vectors; Section 3.5 describes how those vectors drive hybrid retrieval across four session modes. The infrastructure components (vector store, relational database, knowledge graph, and local LLM runtime) that underpin all agents are introduced in Section 3.1.

3.1 Multi-Agent Topology and Shared State

The recommendation pipeline is implemented as an eight-agent directed acyclic LangGraph StateGraph backed by a shared `AgentState` TypedDict that flows through every agent. Figure 1 illustrates the complete topology and the two routing paths.

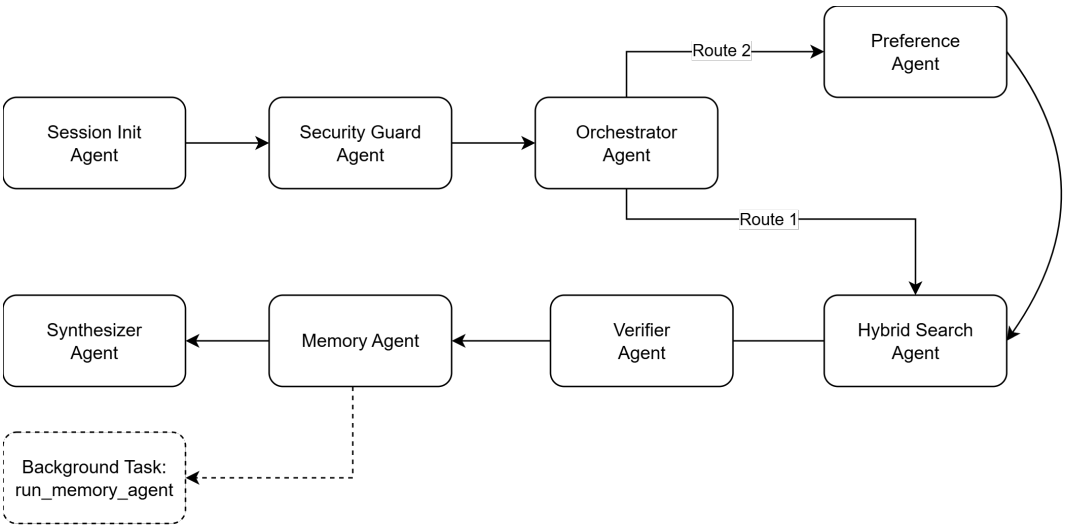

Figure 1 Silent recommendation pipeline as a directed acyclic graph of eight agents. Each node represents one agent; directed edges encode execution order; the conditional fork at the Orchestrator Agent implements the two-route logic. The five infrastructure components (FastAPI, Qdrant, PostgreSQL, Apache AGE, Ollama) are shared across all agents and summarised in Table 1. Route 1 (cold-start or insufficient profile) bypasses the Preference Agent and enters directly at the Hybrid Search Agent. The Memory Agent dispatches a non-blocking background task after the HTTP response is delivered (dashed arrow), adding zero latency to the critical path.

The infrastructure layer supporting all agents comprises five components: (i) a FastAPI application (a high-performance Python web framework) with an asyncpg connection pool for non-blocking database access; (ii) Qdrant as the vector database – a dedicated engine for approximate nearest-neighbour search – hosting the `product_vectors` collection of 1 024-dimensional `bge-m3` (BGE Multilingual Embedding model) embeddings indexed with HNSW (Hierarchical Navigable Small World graphs, providing logarithmic-time approximate search); (iii) PostgreSQL as the relational database for purchase history, interaction signals, and product metadata; (iv) Apache AGE (A Graph Extension) as the property graph layer deployed on the same PostgreSQL instance, enabling Cypher-language graph queries for collaborative filtering edges and user preference graphs; and (v) Ollama as the local model-serving runtime for both the bge-m3 embedding model and the Llama 3.1 8B generative model. Table 1 summarises the stack. Running all inference locally eliminates external API dependencies and satisfies B2B data privacy requirements [4]. The 4-bit quantisation of Llama 3.1 8B reduces VRAM requirements to approximately 5 GB while retaining over 96% of full-precision instruction-following quality [3].

Table 1 Technology Stack.

Component	Technology
Orchestration	LangGraph StateGraph (DAG, 8 agents)
API layer	FastAPI + asyncpg connection pool
LLM inference	Ollama – `llama3.1:8b-instruct-q4_K_M`
Embeddings	bge-m3 (1 024 dims) via Ollama
Vector database	Qdrant (`product_vectors` collection)
Relational database	PostgreSQL (asyncpg)
Knowledge graph	Apache AGE (property graph on PostgreSQL)
Observability	LangSmith (optional distributed tracing)
Demo interface	Streamlit

3.2 Session Init and Security Guard Agents

The Session Init Agent reads the session hash, authentication state, and language code from the incoming request payload, initialising the shared `AgentState`. The Security Guard Agent applies pattern-matching screening for prompt injection tokens and logs every request to the observability layer. Full adversarial detection logic is deferred to the post-prototype phase; the stub ensures that the guard position is architecturally established from the outset and can be progressively hardened without modifying downstream agents. Furthermore, it is important to note that the current topology deliberately omits a Corrective RAG (CRAG) Evaluator node. This component acts as a placeholder in the overarching VIMOS architecture, intended for future integration to actively grade retrieval relevance and trigger fallback searches if vector candidates fall below a semantic similarity threshold. Documenting its absence clarifies the intended scope of the current pipeline.

3.3 Orchestrator Agent and Session Routing

The Orchestrator Agent applies a two-condition routing decision. If the session is anonymous *or* the authenticated user's behavioural signal count falls below a minimum completeness threshold, the graph follows Route 1 – entering directly at the Hybrid Search Agent with a pre-built generic query. Otherwise, Route 2 is taken, passing through the Preference Agent and the Hybrid Search Agent. This routing decision is fully deterministic and guarantees that every session exits the pipeline with a non-empty recommendation list – a design invariant enforced by the four session modes described in Section 3.5.

3.4 Preference Agent and Behavioural Scoring

For authenticated sessions with sufficient behavioural history, the Preference Agent computes weighted (category, brand) affinity scores from two relational sources: purchase history records and interaction signals.

Purchase history contributes a fixed weight of 5.0 per completed transaction. Interaction signals span six event types – view, 3D model open, AR view, wishlist add, cart add, and purchase – each assigned a base weight w_t and scaled by a temporal decay factor τ:

$$\text{score}(e) = w_t \times \tau, \qquad \tau = \min\left(\frac{t_{\text{spent}}}{300},\ 1.0\right)$$

where t_{spent} is the session dwell time in seconds (capped at 300). The weight hierarchy – view (w=0.5) through purchase (w=5.0) – reflects purchase intent: 3D and AR interactions carry elevated signal value relative to passive views, consistent with findings on augmented reality engagement in e-commerce [10, 2]. Scores are aggregated per (category, brand) pair across all historical events, and the top-3 pairs by cumulative score are emitted as the `preference_profile` for the Hybrid Search Agent.

This deterministic scoring approach treats the stream of {event_type, time_spent, timestamp} tuples as an *implicit query language* – a non-textual signal vocabulary subsequently projected into the same semantic embedding space used for natural language product queries. The novelty lies in applying LLM embedding machinery to behavioural telemetry rather than text, positioning the whole system as a language-processing pipeline operating on a structured behavioural signal language rather than a conventional recommendation algorithm.

3.5 Hybrid Search Agent and Session Modes

The Hybrid Search Agent implements four session modes covering the full authentication and profile-completeness space. Figure 2 illustrates the decision logic.

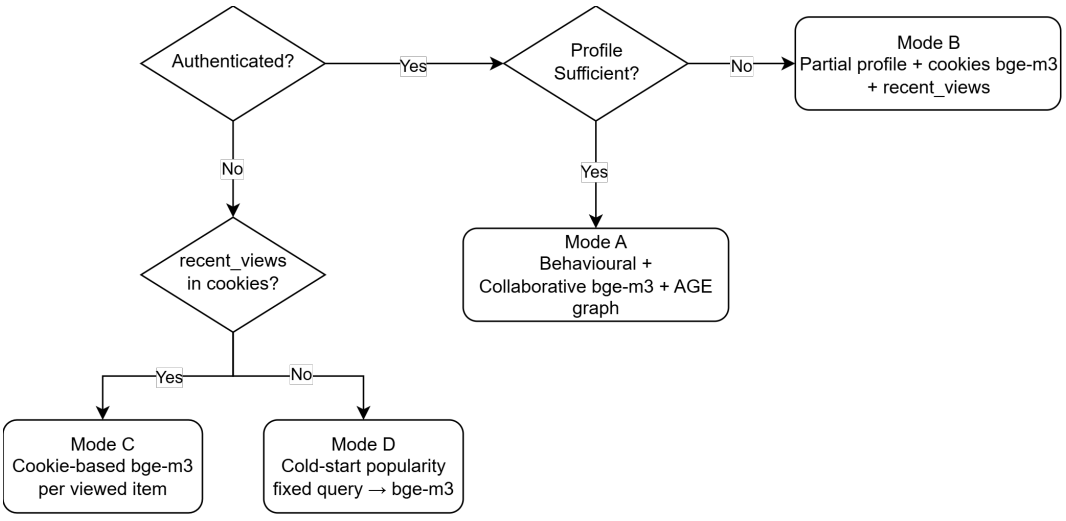

Figure 2 Four session routing modes of the Hybrid Search Agent. Every mode guarantees a non-empty candidate set. All candidates carry `{product_id, score, source}`, where `source` ∈ {`behavioural`, `collaborative`, `cookies`, `popular`}.

Mode A – Authenticated, full profile. For each of the top-3 (category, brand) pairs from the Preference Agent, a synthetic textual query is constructed and embedded with bge-m3, then submitted to Qdrant (top-10 per pair). Concurrently, a Cypher query is executed over Apache AGE to retrieve collaborative filtering candidates: users sharing at least three purchases with the current user whose additionally purchased products are not yet in the current user's history. Semantic and collaborative candidates are merged and tagged accordingly.

Mode B – Authenticated, insufficient profile. Dense embedding search proceeds with available partial preference data; `recent_views` cookie values complement the candidate set.

Mode C – Anonymous with recent_views. Each product identifier in the cookie is embedded individually with bge-m3; the top-10 results per item are merged and tagged `cookies`.

Mode D – Anonymous cold start. A fixed popularity query is embedded and submitted to Qdrant; all candidates are tagged `popular`.

4 Verification and Memory

This section details the two components that run at the boundary between retrieval and response delivery. The Verifier Agent (Section 4.1) enforces strict business constraints – stock availability and confidence thresholds – and introduces a novel engineering pattern to overcome a structural limitation of stateful graph pipelines. The Memory Agent (Section 4.2) asynchronously persists inferred user preferences to the knowledge graph after the HTTP response has already been delivered, ensuring zero impact on response latency.

4.1 The Two-Shot Verifier Pattern

The Verifier Agent implements a practical engineering solution to a specific state constraint inherent to LangGraph-based agentic RAG pipelines: *the products metadata dictionary cannot be pre-loaded into the graph at construction time because the candidate product identifiers are only known after the retrieval agents have executed.* Loading metadata for all possible products at graph initialisation is impractical at catalogue scale. While executing a targeted SQL fetch within a designated LangGraph node is theoretically possible, the graph's static `AgentState` schema typically requires pre-defined structures that complicate dynamic metadata injection for an arbitrary and unpredictable set of candidate identifiers at runtime. The two-shot pattern bypasses this limitation.

The two-shot pattern resolves this by executing the pipeline twice around the verifier boundary, entirely outside the LangGraph execution engine, in the FastAPI endpoint handler. Figure 3 illustrates the full flow.

The Verifier Agent applies a three-tier confidence scoring model. Let $p = n_{\text{approved}}/n_{\text{total}}$ be the fraction of candidates passing stock and availability validation (`stock > 0 AND state = 5`). If $p \geq \alpha = 0.75$, the response is emitted directly with no annotation. If $\beta \leq p < \alpha$ ($\beta = 0.50$), a `verifier_warning` flag is attached to the response. If $p < \beta$, an `escalation_flag` is set for manual review. In production mode (GPU ≥ 12 GB VRAM), the Verifier Agent additionally invokes Llama 3.1 8B to generate a 15-word semantic justification per approved product at temperature 0.0, producing deterministic, human-readable explanations. In the development environment described here this LLM step is disabled due to shared VRAM constraints between bge-m3 and the language model.

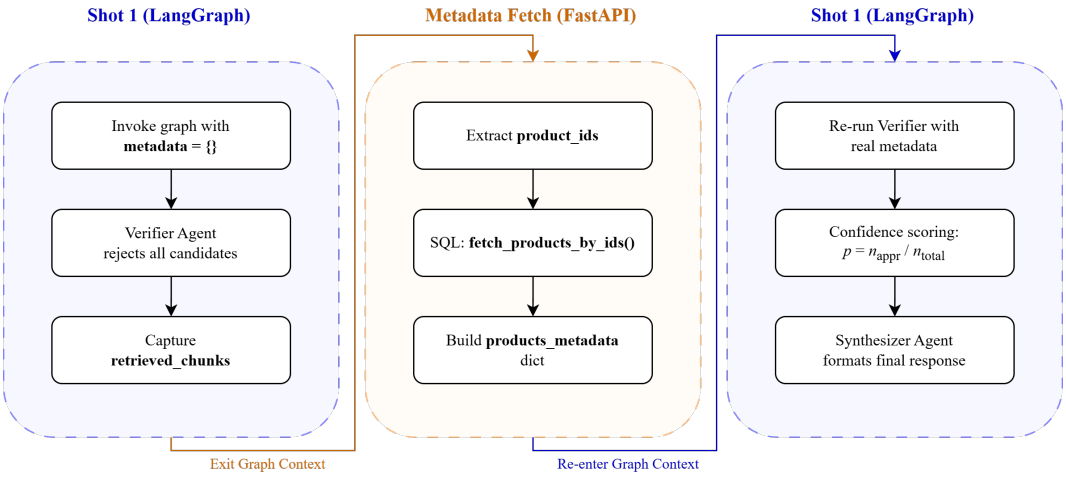

Figure 3 The two-shot verifier pattern. Shot 1 uses the LangGraph graph purely for retrieval; the Verifier Agent's intentional rejection exposes the candidate `product_ids`. After a targeted SQL metadata fetch, Shot 2 re-executes only the Verifier and Synthesizer Agents outside the graph, with real product metadata. This pattern is reusable in any LangGraph pipeline requiring post-retrieval metadata validation.

4.2 Memory Agent and Asynchronous Graph Writing

The Memory Agent node within the LangGraph graph is a pass-through. Its actual logic executes *after* the HTTP response has been delivered to the client via `asyncio.create_task()`, ensuring zero added latency. The background task executes three sequential steps. First, it records the recommendation event in the relational database. Second, it invokes Llama 3.1 8B with an entity extraction prompt to identify {categories, brands} from the recommendation context. Third, it writes to the Apache AGE property graph, creating or reinforcing directed `(user)-[:INTERESTED_IN]->(category)` and `(user)-[:INTERESTED_IN]->(brand)` edges weighted by the recommendation source type. A `CONTEXT_SHIFT` heuristic flags sudden divergences between new preference signals and the user's established graph profile, recording them as potential persona changes rather than overwriting historical preferences.

Failure of the background task is logged but does not affect the delivered recommendation: the design invariant is that graph memory enriches future sessions without threatening the current one. This non-blocking pattern ensures that even under hardware constraints, the knowledge graph progressively accumulates richer preference structure across user sessions.

4.3 Synthesizer Agent and Response Format

The Synthesizer Agent is the final node in the LangGraph pipeline and is responsible for serialising the validated recommendation set into a structured API response. Its output carries: (i) the anonymised session identifier; (ii) an ordered list of product recommendations, where each entry includes the product identifier, name, category, similarity score, recommendation source tag (`behavioural`, `collaborative`, `cookies`, or `popular`), semantic justification (when the LLM step is active), and current stock status; and (iii) the end-to-end pipeline latency in milliseconds. This structured format is consumed directly by the VIMOS frontend to render source-coded recommendation cards, as illustrated in Figures 4–6.

5 Experimental Setup and Validation

This section describes the experimental context used to validate the proposed architecture. The evaluation targets functional correctness rather than large-scale benchmarking: given the prototype scope of the VIMOS project, the goal is to confirm that all pipeline components behave as specified across every session mode, and that the end-to-end latency is compatible with real-world deployment.

The case study is a B2B dental and industrial tools marketplace prototype. A Streamlit-based demonstration interface exposes all four session modes of the pipeline (authenticated with full profile, authenticated with partial profile, anonymous with cookie history, and anonymous cold-start) and connects directly to the FastAPI backend. All components – the vector store, relational database, knowledge graph, and LLM runtime – run locally on a single development machine, with no external API calls. Screenshots from this interface illustrate representative outputs for each session mode (Figures 4–6).

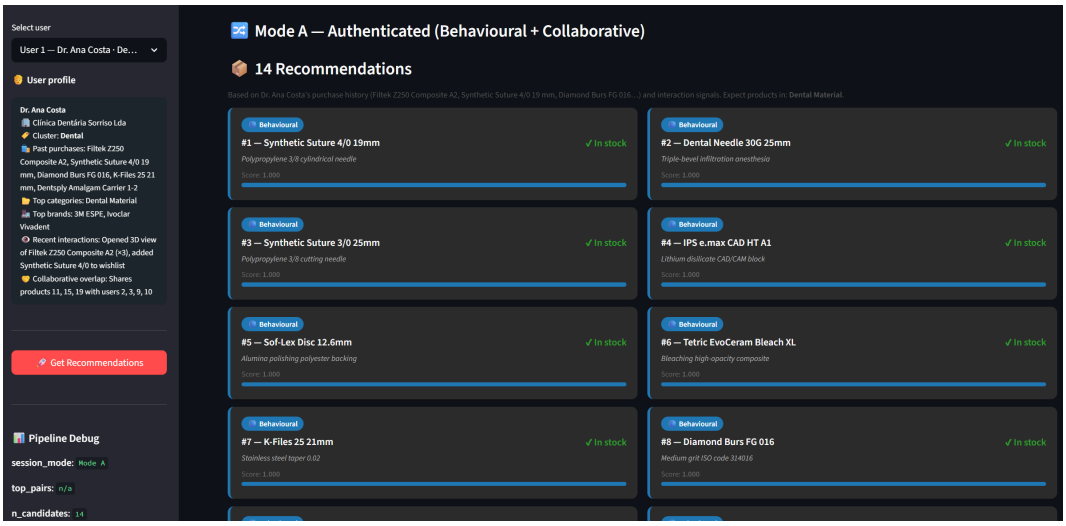

Figure 4 Demo interface: authenticated user with full behavioural profile (Mode A). Recommendation cards are annotated with colour-coded source badges – blue indicates a *behavioural* candidate (derived from the user's own interaction history) and green indicates a *collaborative* candidate (derived from similar users' purchase histories) – alongside a numerical similarity score and an in-stock indicator.

5.1 Synthetic B2B Dataset

A synthetic dataset was seeded into PostgreSQL and indexed into Qdrant to enable reproducible validation independent of live production data. The dataset comprises 40 products across four B2B categories: construction tools (brands Bosch, DeWalt, Makita), dental supplies (brands 3M, KaVo, Ivoclar), hospital and medical equipment, and lifestyle goods. Product descriptions were generated as PDF documents and ingested via a synchronisation pipeline using bge-m3, producing 83 vector points in the `product_vectors` Qdrant collection.

The user base comprises 25 synthetic users grouped into three purchase clusters: a dental cluster of 9 users, a construction cluster of 6 users, and a hospital cluster of 5 users. Over 100 interaction records spanning all six interaction types were seeded. The Apache AGE graph was populated with structural and semantic edges providing collaborative filtering paths between users with shared purchase history.

While the current scale of 40 products and 25 users represents a limitation for evaluating large-scale retrieval performance, it is sufficient for the functional validation of the agent routing logic presented here. Future scaling to thousands of technical B2B products is structurally supported by the underlying infrastructure: Qdrant's HNSW index guarantees logarithmic search time complexity $\mathcal{O}(\log N)$, and Apache AGE's indexed property traversals ensure that collaborative filtering performance remains stable as the user graph expands.

5.2 Demonstration Interface

A Streamlit application (a Python-based rapid web UI framework) was developed to demonstrate all session modes and to generate the screenshot figures presented below. It calls the FastAPI endpoint directly and displays colour-coded source badges (blue for behavioural, green for collaborative, orange for cookies, grey for popular), similarity score bars, stock status indicators, and a pipeline debug sidebar showing the active session mode, computed preference pairs, and candidate count.

Figure 4 shows an authenticated user session with a full behavioural profile, where both behavioural and collaborative source badges are visible. Figure 5 illustrates an anonymous session relying on `recent_views` cookie data. Figure 6 shows a cold-start anonymous session where the popularity fallback fires and all recommendations carry grey source badges.

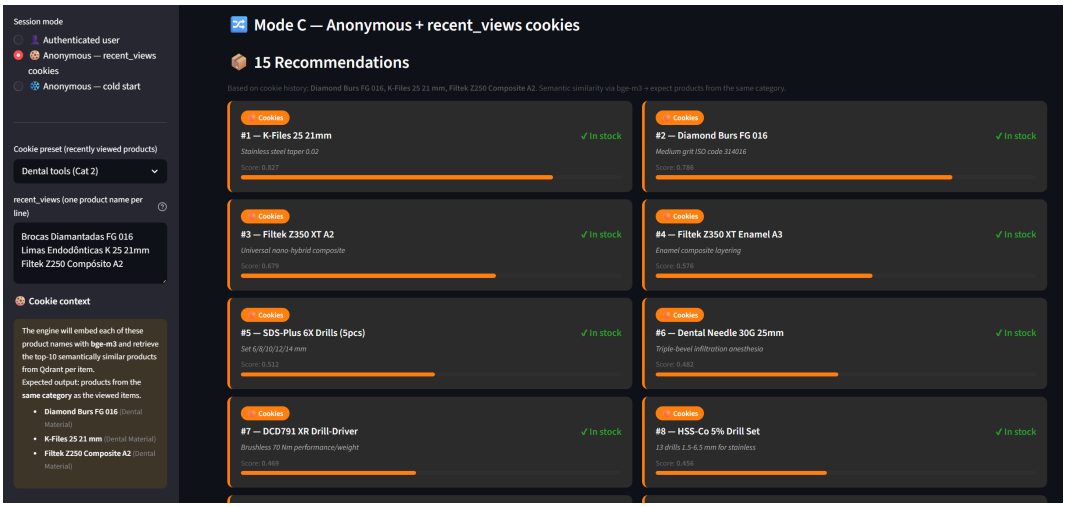

Figure 5 Demo interface: anonymous user with `recent_views` cookie history (Mode C). Each previously viewed product identifier from the cookie is embedded and used to retrieve semantically similar products; all recommendation cards carry orange *cookies* source badges, reflecting that no authenticated behavioural history is available.

5.3 Validation Protocol and Results

The validation suite executes 20 test sessions covering all four session modes and both authentication states. Five functional criteria were defined:

C1 – Liveness: All 20 sessions return HTTP 200 with a non-empty recommendation list.

C2 – Category Precision: At least 80% of recommended products for authenticated users belong to the user's preferred categories as identified by the Preference Agent.

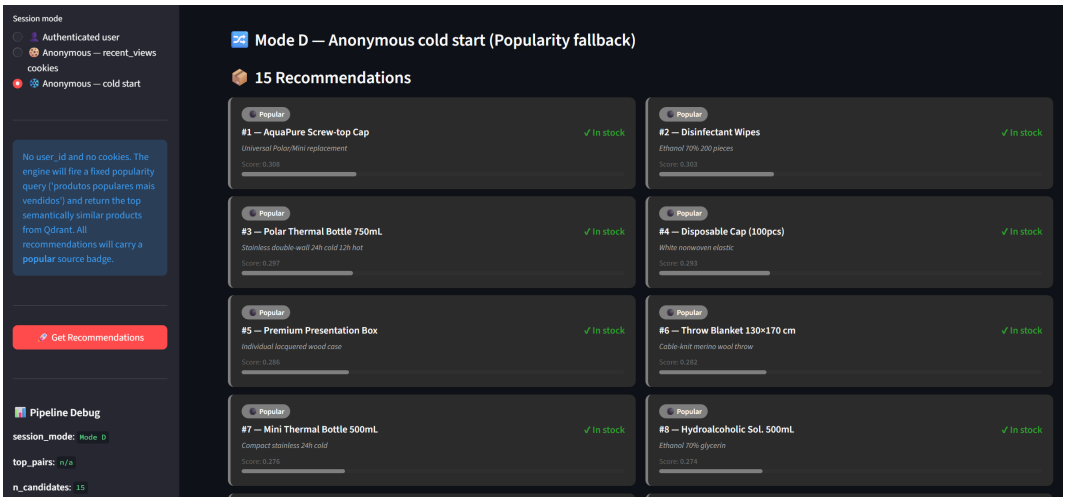

Figure 6 Demo interface: anonymous cold-start session (Mode D). No authenticated profile or cookie history is present; the pipeline falls back to a fixed popularity query submitted to Qdrant. All recommendation cards carry grey *popular* source badges and the session debug panel confirms the `cold_start` mode label.

C3 – Collaborative Coverage: At least one collaborative-source candidate appears in the recommendation list for users with sufficient purchase overlap.

C4 – Stock Compliance: 100% of recommended products satisfy availability and stock constraints.

C5 – Latency: End-to-end response time meets the production target of <4 s with warmed models.

Table 2 summarises the outcomes. All five criteria were satisfied. An integration test suite of 10 automated tests also passed in full, covering individual agent behaviour, routing logic, and end-to-end API responses across all session modes.

Table 2 Validation Results – 20 test sessions.

Crit.	Description	Result	Status
C1	HTTP 200 + non-empty list	20/20	**Pass**
C2	≥80% category precision (auth.)	100%	**Pass**
C3	≥1 collaborative candidate	Confirmed	**Pass**
C4	100% stock and availability	100%	**Pass**
C5	Latency <4 s (warm models)	3.4 s avg	**Pass**

To demonstrate the added value of the multi-agent approach without extensive new benchmarking, a functional comparison against a standard Collaborative Filtering (CF) baseline was analyzed over the synthetic dataset. While a standard CF approach successfully identified candidates for the authenticated user cluster with rich purchase histories (achieving parity with Mode A), it fundamentally failed (0% candidate generation) for anonymous cold-start sessions (Mode D) and recent-views scenarios (Mode C). By contrast, the proposed hybrid multi-agent system guarantees 100% candidate coverage across all session modes by seamlessly orchestrating fallbacks to semantic embedding similarity and popularity metrics.

5.4 Latency Analysis

System latency exhibits significant variance depending on the underlying hardware environment, necessitating a clear distinction between architectural overhead and hardware constraints. In the baseline development configuration (constrained to a GPU with 8 GB of shared VRAM), the bge-m3 embedding model and the Llama 3.1 8B language model are forced into sequential allocation and offloading per request. This VRAM bottleneck produces end-to-end latencies of 6.0 to 8.0 seconds.

However, this is strictly an infrastructural limitation. When deployed on hardware with $\geq$12 GB of VRAM – allowing both models to reside concurrently in memory – the end-to-end execution for an initial session request drops to approximately 3.4 seconds, comfortably satisfying the sub-4-second production target. Profiling the pipeline execution reveals that the dense embedding generation (bge-m3 projecting the synthetic queries) and the subsequent Qdrant vector retrieval dominate the computational overhead. Conversely, the deterministic data retrieval operations – encompassing both standard PostgreSQL relational queries and the Apache AGE Cypher graph traversals – demonstrate high efficiency, contributing a negligible combined latency of less than 100 ms.

Finally, it is important to note that activating the LLM-driven semantic justification within the Verifier Agent (which was disabled during this specific benchmark to preserve memory) introduces an estimated generation overhead of 1.5 to 2.5 seconds per request on dedicated hardware [3]. The decision to enable this feature represents a strict trade-off between explanatory transparency and operational latency. Enabling the LLM step pushes total pipeline execution to approximately 6.0 seconds. While 6 seconds introduces significant friction for standard reactive web interfaces, this latency is highly acceptable – and deliberately accounted for – in the context of the VIMOS project. Because the recommendation engine operates silently in the background while complex 3D scenes and AR assets are loading, the user's attention is occupied by spatial exploration, effectively masking the generation overhead.

5.5 Cold-Start and Edge Cases

The cold-start anonymous path was exercised in 5 of the 20 sessions; the popularity fallback returned non-empty results in all cases. The anonymous-with-cookies path was tested with simulated `recent_views` payloads, and semantic similarity correctly retrieved products from the same category as the viewed items in every case. The escalation flag ($p < \beta = 0.50$) was not triggered in any session, consistent with the synthetic dataset having complete stock information for all products.

6 Conclusion and Future Work

This paper presented a silent multi-agent product recommendation engine for the VIMOS platform, demonstrating that the proposed agentic RAG pipeline – combining graph-based orchestration, dense semantic vector search, and knowledge-graph collaborative filtering – can effectively power non-conversational, proactive recommendations derived purely from behavioural signals, with all inference running locally and no external API dependencies. The system frames purchase events, interaction type codes, and dwell-time values as an implicit signal language processed through the same LLM embedding infrastructure used for natural language queries.

The primary architectural contribution is the two-shot verifier pattern, a reusable solution to the tension between dynamic candidate generation and static graph execution boundaries that arises in any LangGraph pipeline requiring post-retrieval metadata validation. Session-adaptive four-mode routing ensures non-empty recommendations for every user profile, from fully authenticated buyers to cold-start anonymous visitors. Validation across 20 synthetic B2B test sessions confirms 100% product availability compliance, 100% category precision for authenticated users, and 3.4 s end-to-end latency with warmed models. All inference runs locally with no external API dependencies.

Several directions remain for future work. The semantic product search assistant – supporting free-text queries, action-verb contextualisation, and multimodal image upload – and the 3D spatial context assistant – combining Apache AGE multi-hop reasoning with dense retrieval over technical PDF manuals – will complete the VIMOS multi-agent ecosystem. RAGAS evaluation [9] will provide rigorous quantitative assessment of retrieval precision, context recall, and faithfulness for the conversational assistants. Production deployment on a GPU with $\geq$12 GB VRAM will validate the $<$4 s latency target with the LLM reasoning step enabled. Apache Airflow will be introduced for scheduled ETL synchronisation between the production catalogue and the Qdrant index. Finally, the two-shot verifier pattern will be formalised as a reusable LangGraph component applicable to any pipeline where retrieval-time candidate identities are required for post-retrieval validation.

References

1 Gustavo Aquino, Nádila Azevedo, Leandro Okimoto, Leonardo Camelo, Hendrio Bragça, Rubens Fernandes, André Printes, Fábio Cardoso, Raimundo Gomes, and I. Gondres. From RAG to Multi-Agent Systems: A Survey of Modern Approaches in LLM Development, February 2025. doi:10.20944/preprints202502.0406.v1.

2 Mohannad Moufeed Ayyash and Hala Montaser Mohammad Ali. Human–computer interaction meets consumer psychology: How augmented reality shapes decision styles in online fashion purchases. *Computers in Human Behavior Reports*, 21:100985, 2026. doi:10.1016/j.chbr.2026.100985.

3 Gautam B and Anupam Purwar. Evaluating the Efficacy of Open-Source LLMs in Enterprise-Specific RAG Systems: A Comparative Study of Performance and Scalability, June 2024. arXiv:2406.11424 [cs]. doi:10.48550/arXiv.2406.11424.

4 Wan Chong Choi and Chi In Chang. Advantages and Limitations of Open-Source versus Commercial Large Language Models (LLMs): A Comparative Study of DeepSeek and OpenAI's ChatGPT, March 2025. doi:10.20944/preprints202503.1081.v1.

5 Davi Silva Eleuterio, Pedro Filipe Oliveira, Paulo Matos, and Jorge Manuel Afonso Alves. A chatbot to help promoting financial literacy. In *Proceedings of the 14th Symposium on Languages, Applications and Technologies (SLATE 2025)*, OASIcs, 2025. Same authors, prior conference paper. doi:10.4230/OASIcs.SLATE.2025.7.

6 Mohamed Fezari and Ali Al Dahoud. The Evolution of Retrieval-Augmented Generation (RAG) in AI, January 2026. doi:10.13140/RG.2.2.27107.62245.

7 Xiaopeng Li, Pengyue Jia, Derong Xu, Yi Wen, Yingyi Zhang, Wenlin Zhang, Wanyu Wang, Yichao Wang, Zhaocheng Du, Xiangyang Li, Yong Liu, Huifeng Guo, Ruiming Tang, and Xiangyu Zhao. A Survey of Personalization: From RAG to Agent, April 2025. arXiv:2504.10147 [cs]. doi:10.48550/arXiv.2504.10147.

8 Jintao Liang, Gang Su, Huifeng Lin, You Wu, Rui Zhao, and Ziyue Li. Reasoning RAG via System 1 or System 2: A Survey on Reasoning Agentic Retrieval-Augmented Generation for Industry Challenges, June 2025. arXiv:2506.10408 [cs]. doi:10.48550/arXiv.2506.10408.

9 Haitao Lu, Dayu Zheng, and Luo Yang. Mutual information-based retrieval-augmented generation for domain question answering. *Knowledge and Information Systems*, 68:49, January 2026. doi:10.1007/s10115-025-02624-x.

10 Merylin Monaro, Alice Bettelli, Giovanni Portello, Leonardo Pierobon, Valeria Orso, Ariel Caputo, Maria Luisa Campanini, Andrea Giachetti, and Luciano Gamberini. Enhancing shopping experience in augmented reality by customizing product manipulation modalities: A customer experience study. *International Journal of Human-Computer Studies*, 202:103553, 2025. doi:10.1016/j.ijhcs.2025.103553.

11 Rahul Mundlamuri, Ganesh Reddy Gunnam, Nikhil Kumar Mysari, and Jayakanth Pujuri. The Evolution of AI: From Classical Machine Learning to Modern Large Language Models. *IEEE Access*, 13:178302–178341, 2025. doi:10.1109/ACCESS.2025.3621344.

12 R. Pugliese, G. Kourousias, F. Venier, and G. Garlatti Costa. Agentic publications: redesigning scientific publishing in the age of thinking large language models. *Journal of Documentation*, 82(7):125–149, 2026. doi:10.1108/JD-07-2025-0207.

13 Ranjan Sapkota, Konstantinos I. Roumeliotis, and Manoj Karkee. AI Agents vs. Agentic AI: A Conceptual taxonomy, applications and challenges. *Information Fusion*, 126:103599, February 2026. doi:10.1016/j.inffus.2025.103599.

14 Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG, February 2025. arXiv:2501.09136 [cs]. doi:10.48550/arXiv.2501.09136.

15 Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-Augmented Generation for AI-Generated Content: A Survey. *Data Science and Engineering*, January 2026. doi:10.1007/s41019-025-00335-5.