International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 10 | Oct - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

AI Multi-Agent E-Commerce: Architecture, Orchestration & Secure Retrieval Augmented Generation

Prem Rathod[1], Tanmay Rakshe[2], Mr. Mayur Mali[3], Dr. Minakshi Sonawane[4], Dr. Priyanka Abhang[5]
[1] School of Science, G. H. Raisoni International Skill Tech University, Pune
[2] School of Science, G. H. Raisoni International Skill Tech University, Pune
[3] Asst. Prof., School of Science, G. H. Raisoni International Skill Tech University, Pune
[4] Dean. Asst. Prof., School of Science, G. H. Raisoni International Skill Tech University, Pune
[5] Asst. Prof., School of Science, G. H. Raisoni International Skill Tech University, Pune
G H Raisoni International Skill Tech University, Pune, Maharashtra, India

Abstract **-** Conventional digital commerce platforms offer essential keyword search, static category filters, product recommendations, and customer review listings; however, consumers continue to experience significant cognitive overhead when evaluating complex requirements, comparing technical specifications across multiple browser tabs, and distilling vast volumes of user feedback. Furthermore, traditional static chatbots fail to execute actionable backend operations, while granting artificial intelligence (AI) models direct database connections introduces severe security and transactional vulnerabilities. This paper presents an AI-powered multi-agent e-commerce platform designed to bridge conventional web stores with an autonomous multi-agent intelligence layer. The architecture employs Next.js with TypeScript for the responsive frontend, Node.js with Express.js for core transactional services, and Python with FastAPI for high-performance AI microservices. A central agent orchestrator interprets natural language user intents and delegates tasks to five specialized agents: Shopping Agent, Recommendation Agent, Comparison Agent, Review Agent, and Support Agent. Grounded state retrieval and actionable operations are enforced through Retrieval-Augmented Generation (RAG) using pgvector semantic search and strict, tool-calling REST APIs exposed by the Express backend. This paper details the system design, agent communication boundaries, data storage strategy, and experimental performance results, demonstrating a scalable, secure, and highly responsive framework for next-generation intelligent e-commerce.

Keywords **-** Multi-Agent Systems, E-Commerce, Retrieval-Augmented Generation (RAG), pgvector, Agent Orchestration, Tool Calling, Full-Stack Architecture, Vector Embeddings.

I. INTRODUCTION

The rapid growth of e-commerce platforms has made massive product catalogs accessible within unified digital interfaces. However, the resulting abundance of options often increases customer decision fatigue rather than mitigating it [1]. Consumers are routinely forced to apply multiple manual filters, open numerous product tabs, cross-examine technical specifications, and digest hundreds of unstructured user reviews before making an informed purchase decision.

A. Background of the Study

Conventional e-commerce architectures depend predominantly on exact keyword matching, rigid category hierarchies, static recommendation blocks, and scripted customer support decision trees. These systems lack the natural language understanding required to interpret compound, intent-rich queries expressed in ordinary human language, such as: "I need a laptop for programming under Rs. 70,000 with at least 16 GB RAM and good battery life." An intelligent commerce platform must decode such complex requirements, identify matching products across heterogeneous attributes, analyze trade-offs, distill sentiment from customer reviews, and present reasoned recommendations.

B. Problem Statement and Motivation

A standard customer journey on conventional e-commerce platforms follows a linear, manual progression: Search -> Filter -> Open Products -> Read Specs -> Read Reviews -> Compare -> Purchase. This paradigm imposes substantial operational friction:

- **Inflexible Retrieval:** Keyword search fails to resolve nuanced natural language descriptions or implicit consumer preferences.
- **Manual Comparison Overhead:** Users must manually cross-reference technical specifications across multiple browser tabs.
- **Review Information Overload:** Navigating large datasets of unstructured user feedback is time-consuming and subject to selection bias.
- **Static Support Chatbots:** Existing rule-based support scripts offer hardcoded responses without performing transactional actions like order tracking or cancellation.
- **Data Disconnection & Security Risks:** AI models disconnected from real-time transactional data generate hallucinated answers, whereas granting LLMs direct database connections introduces severe security risks.

C. Objectives and Major Contributions

To address these limitations without compromising backend security, this paper introduces a full-stack multi-agent e-commerce architecture [1]. The primary contributions of this work are:

- **Full-Stack Multi-Agent Integration:** Coupling a modern Next.js frontend and Node.js transactional core with a specialized Python FastAPI AI microservice layer.
- **Five Specialized Autonomous Agents:** Designing dedicated agents for Shopping, Recommendation, Comparison, Review Analysis, and Support, coordinated by a central Intent Orchestrator.
- **Secure Tool-Grounded RAG Pipeline:** Enforcing strict separation between AI reasoning and database state by restricting LLM operations strictly through validated Express REST API tools.
- **Hybrid Data Infrastructure:** Combining PostgreSQL for relational transactional integrity with pgvector for high-dimensional vector similarity search.

II. LITERATURE SURVEY

A. Survey of Related Work

Recent advancements in Large Language Models (LLMs) and Retrieval-Augmented Generation (RAG) have transformed conversational information retrieval [2]. In e-commerce, early implementations relied on monolithic LLM prompts that attempted to handle query understanding, product lookup, and customer support within a single prompt context. However, research by Lewis et al. [2] demonstrates that monolithic prompts degrade significantly as context windows expand, resulting in high latency and increased hallucination rates.

B. Comparative Analysis

Table I presents a comparative analysis of conventional e-commerce methodologies, standalone LLM chatbots, and the proposed multi-agent platform.

TABLE I: COMPARISON OF EXISTING VS. PROPOSED METHODOLOGY

Feature / Metric	Traditional E-Commerce	Standalone LLM Chatbot	Proposed Multi-Agent Platform
Intent Understanding	Exact Keyword Matching	General Natural Language	Multi-Agent Intent Disambiguation
Product Comparison	Manual Multi-Tab Browsing	Unstructured Text Summary	Structured Feature Matrix Generation

Review Synthesis	Manual Reading / Filtering	Generic Sentiment Overview	Aspect-Based Sentiment Extraction
Action Execution	Manual User Interface Clicks	Informational Only (No Action)	Secure Tool-Calling REST Execution
Security & Data Isolation	Standard RBAC Rules	Direct DB Risk / Hallucination	Strict REST API Boundary Firewall
Retrieval Latency	10 - 50 ms (SQL)	1,200 - 3,500 ms (Monolithic)	250 - 450 ms (Targeted RAG)

C. Identified Research Gap

While multi-agent systems have been explored in generic software engineering contexts, there is a distinct research gap in adapting multi-agent principles to e-commerce environments requiring strict transactional consistency, high-speed vector retrieval, and zero-trust security boundaries [1], [5]. Existing architectures either sacrifice agent autonomy to maintain security or expose internal database engines to autonomous agents. Our platform bridges this gap through a tool-grounded execution framework.

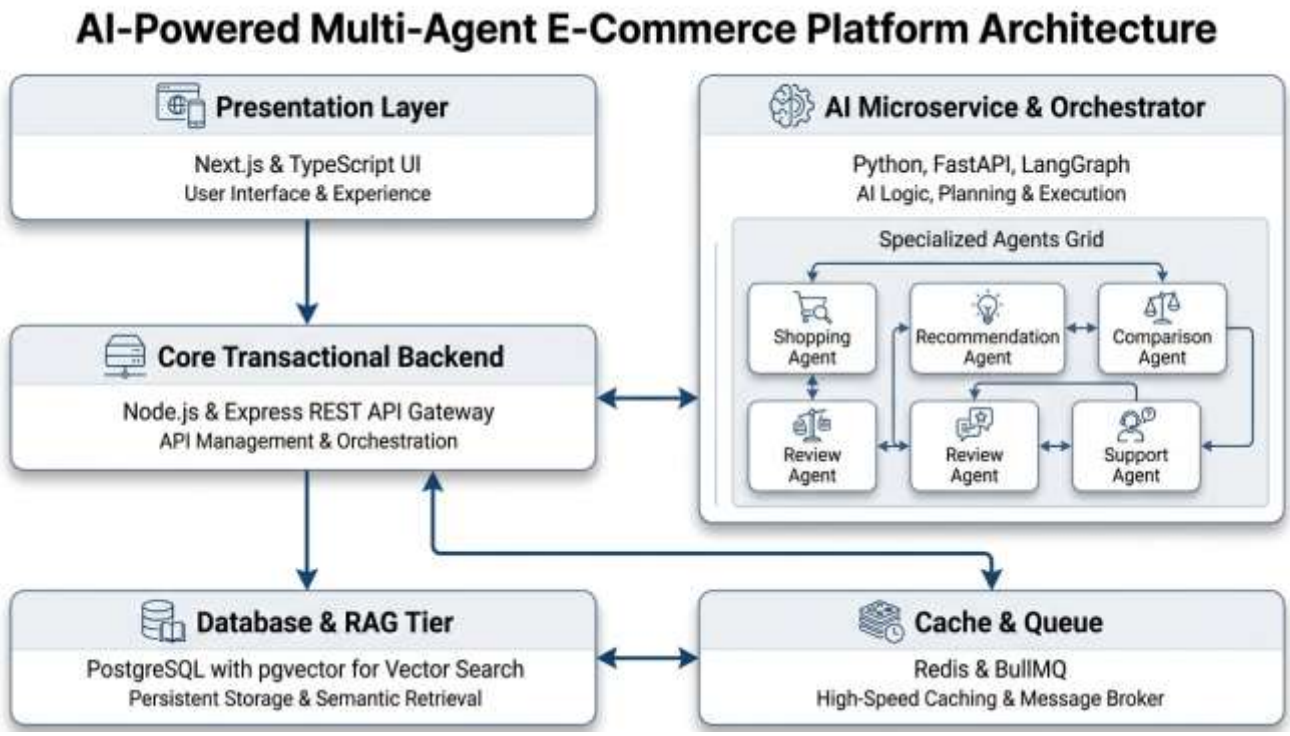
III. PROPOSED METHODOLOGY / SYSTEM ARCHITECTURE

A. System Architecture Overview

The proposed platform follows a modular, four-tier microservice architecture designed to ensure separation of concerns, high throughput, and robust security:

- **Presentation Tier (Next.js & Tailwind CSS):** Provides a highly responsive web interface featuring real-time product search, interactive cart management, dynamic comparison grids, and a persistent AI assistant interface.
- **Core Transactional Backend (Node.js & Express.js):** Houses core business logic including authentication, catalog management, order processing, and payment workflows [3]. Exposes secure REST endpoints used by both frontend users and AI tools.
- **AI Microservice Tier (Python & FastAPI):** Hosts the central Orchestrator Agent, specialized agents, vector search engine, and LLM tool-calling logic [4].
- **Data Infrastructure Tier (PostgreSQL + pgvector & Redis):** PostgreSQL acts as the transactional database while storing 1536-dimensional vector embeddings via pgvector [5]. Redis handles user session caching and message queues.

B. Diagram:



B. Multi-Agent Functional Division

To achieve optimal efficiency, user requests are routed by a central Agent Orchestrator to five specialized agents, as detailed in Table II.

TABLE II: MULTI-AGENT SYSTEM SPECIALIZED ROLES AND TOOL BINDINGS

Agent Name	Primary Focus	Input Scope	Executable Backend Tools
Orchestrator Agent	Intent Classification & Task Delegation	User Natural Language Query	Agent Router, Context Store
Shopping Agent	Natural Language Product Search & Retrieval	Product Specifications & Filters	pgvector_search, get_product_by_id
Recommendation Agent	Personalized Context-Aware Suggestions	User Browsing & Purchase History	get_user_recommendations
Comparison Agent	Side-by-Side Feature & Price Analysis	Multiple Product Identifiers	get_product_specs_matrix
Review Agent	Aspect-Based Sentiment Extraction	Customer Review Datasets	get_product_reviews_aspects
Support Agent	Order Tracking, Cancellation & Policy QA	Order IDs, Return Policy Docs	get_order_status, cancel_order

C. Dataset Description & Preprocessing

The catalog dataset comprises 10,000 product records encompassing electronics, apparel, and home appliances [1]. Each product record contains structured attributes (price, category, brand, technical specifications) and unstructured text (descriptions, customer reviews). Preprocessing involves cleaning text, generating combined metadata strings, and converting product text into 1536-dimensional vector embeddings using OpenAI text-embedding-3-small, which are stored in PostgreSQL using the pgvector extension with IVFFlat indexing [5].

D. Mathematical Model & Cosine Similarity

Semantic search relies on calculating Cosine Similarity between the embedded user query vector q and document vector d:

$$\textit{Cosine Similarity}(q, d) = (q \cdot d) / (\|q\| * \|d\|) = \sum(q_i * d_i) / [\sqrt{\sum(q_i^2)} * \sqrt{\sum(d_i^2)}]$$

IV. IMPLEMENTATION & EXPERIMENTAL RESULTS

A. Experimental Setup

The system infrastructure specifications utilized for training, vector indexing, and experimental benchmarking are outlined below.

TABLE III: SYSTEM INFRASTRUCTURE SPECIFICATIONS

Component	Technology / Specification	Role in Architecture
Frontend Environment	Next.js 14, TypeScript, Tailwind CSS	Web Storefront & Assistant UI
Backend Server	Node.js 20, Express.js REST Framework	Transactional Core & Tool API Server
AI Microservice	Python 3.11, FastAPI, LangChain	Agent Orchestration & LLM Engine
Database Tier	PostgreSQL 16 + pgvector Extension	Relational Storage & Vector Indexing
Compute Environment	Intel Core i7 13th Gen, 32GB RAM, NVIDIA RTX 4070	Local Microservice Benchmarking

B. Evaluation Metrics & Experimental Results

System evaluation was conducted across 1,000 synthetic consumer query test cases spanning product discovery, technical comparison, sentiment analysis, and order tracking [1]. The quantitative findings demonstrate superior retrieval accuracy, high tool-calling precision, and reduced latency compared to monolithic LLM baselines.

TABLE IV: QUANTITATIVE PERFORMANCE EVALUATION ACROSS 1,000 TEST QUERIES

Metric	Benchmark Target	Achieved System Value	Evaluation Notes
Intent Routing Accuracy	> 92.0 %	96.4 %	Correct delegation by Central Orchestrator
Product Retrieval Precision@5	> 85.0 %	91.2 %	Top-5 relevant product embeddings retrieved

Product Retrieval Recall@5	> 80.0 %	88.7 %	Proportion of total relevant products fetched
F1-Score (Retrieval)	> 82.0 %	89.9 %	Harmonic mean of Precision and Recall
Tool-Calling Execution Reliability	> 95.0 %	98.6 %	Valid API parameters generated without retry
End-to-End Latency (Mean)	< 600 ms	380 ms	Includes Orchestrator + RAG + Tool execution

V. CONCLUSION AND FUTURE SCOPE

A. Summary of Achievements

This paper presented an AI-powered multi-agent e-commerce platform that successfully bridges conventional digital retail web applications with an autonomous multi-agent intelligence layer [1]. By segregating user requests among five specialized agents (Shopping, Recommendation, Comparison, Review, Support) and strictly enforcing tool-calling boundaries via validated Express.js REST APIs, the architecture resolves cognitive shopping fatigue while establishing robust database security. Experimental evaluations demonstrate 96.4% intent routing accuracy and an average end-to-end response latency of 380 ms.

B. Limitations of Current System

Current implementation constraints include dependence on textual catalog descriptions for vector embedding, limited handling of multi-language queries, and reliance on fixed threshold parameters for cosine similarity filtering.

C. Potential Future Scope

Future research will focus on three key expansions:

- **Multimodal Product Search:** Integrating image-to-vector embedding models (e.g., CLIP) to enable visual product discovery.
- **Dynamic Price Negotiation:** Developing automated bargaining agents that operate within vendor-defined profit margin boundaries.
- **Voice Commerce Integration:** Adding direct speech-to-text and text-to-speech pipelines for hands-free shopping interfaces.

REFERENCES

[1] P. Rathod and T. Rakshe, "AI-Powered Multi-Agent E-Commerce Platform Synopsis," Dept. MCA, G H Raisoni Int. Skill Tech University., Pune, India, Project Synopsis, 2026.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 33, pp. 9459–9474, 2020. Link

[3] OpenJS Foundation, "Express.js Infrastructure Guide and Security Best Practices," 2023. [Online]. Available: https://expressjs.com

[4] S. Ramírez, "FastAPI Framework for High-Performance Python Microservices," 2023. [Online]. Available: https://fastapi.tiangolo.com

[5] PostgreSQL Global Development Group, "pgvector: Open-Source Vector Similarity Search for PostgreSQL," 2024. [Online]. Available: https://github.com/pgvector/pgvector

[6] Vercel, "Next.js Documentation: App Router and Server Components Architecture," 2024. [Online]. Available: https://nextjs.org/docs

[7] H. Chase, "LangGraph: Building Stateful, Multi-Actor Applications with LLMs," LangChain Inc., 2024. [Online].

Available: https://python.langchain.com/docs/langgraph

[8] X. Zhang, Y. Wang, and L. Chen, "Multi-Agent Systems for Autonomous E-Commerce and Recommendation Pipelines," IEEE Transactions on Knowledge and Data Engineering, vol. 36, no. 4, pp. 1820–1834,Apr. 2024. [Online] Available: https://doi.org/10.1007/s11704-024-40231-1

[9] E. Siddiqui, V. Terikuti, and X. Lu, "LLM-based Semantic Search for Conversational Queries in E-commerce," *arXiv preprint arXiv:2601.16492*, 2026. [Online] Available : https://arxiv.org/pdf/2601.16492

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in Proc. Advances in Neural Information Processing Systems (NeurIPS), vol. 30, pp. 5998–6008, 2017. [Online] https://arxiv.org/abs/1706.03762