IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

An AI-Driven Smartphone Recommendation System Using RAG, LLMs, and Multisource Price Tracking

Mrs. Abha Pathak[1], Mrs. Tejaswini Mali[2], Sanket Rathod[3], Niraj Rane[4], Swapnali Pimpare[5], Aditi Rakh[6]
Assistant Professor, Department of Computer Engineering[1,2]
UG Student, Department of Computer Engineering[3,4,5,6]
Dr. D. Y. Patil College of Engineering and Innovation, Varale
pimpareswapnali@gmail.com

Abstract: ***Objective:*** *Selecting a suitable smartphone has become challenging due to rapidly evolving specifications, market fragmentation, and limited user expertise. This paper presents the design and implementation of Transistor, an AI-driven smartphone recommendation system tailored for the Indian market.*

Method: *The system employs a hybrid pipeline combining Retrieval-Augmented Generation (RAG) with K-Nearest Neighbors (KNN) similarity search over dense product embeddings, orchestrated via a FastAPI backend. A structured, multi-schema PostgreSQL database stores 627 phone records across 17 brands, populated through an automated Firecrawl-based scraping pipeline and validated by a Human-in-the-Loop (HITL) review layer. A slot-filling conversational funnel collects up to 17 buyer-preference signals; a two-peak India recency model, a band-aware 5G connectivity scorer, and a graduated constraint-relaxation engine combine with semantic embedding retrieval to rank candidates.*

Results: *System demonstration on representative query scenarios covering budget-constrained, use-case-specific, spec-targeted, and comparative intents confirms that the pipeline produces contextually relevant, factually grounded recommendations with an average end-to-end response time of 3–5 seconds. The HITL validation layer is shown qualitatively to eliminate category-level factual errors present in raw-scraped data. Quantitative ranking evaluation against human-annotated relevance judgments is planned as future work.*

Contribution: *The principal novelty is the integration of India-specific ranking signals—carrier 5G band compatibility (Jio, Airtel, Vi, BSNL), a dual-peak recency model reflecting India's two annual sales seasons, and a sale-season relevance multiplier—within a continuously refreshed, human-verified knowledge base of 627 phones across 17 brands. The system combines these India-market signals with semantic embedding retrieval and a slot-filling conversational funnel in a single deployable pipeline.*

Keywords: Conversational recommendation system, Gemini embeddings, Human-in-the-Loop validation, India smartphone market, K-Nearest Neighbors, Large Language Models, Retrieval-Augmented Generation, slot-filling dialogue

I. INTRODUCTION

Large Language Models (LLMs) have substantially advanced the capabilities of intelligent information filtering and recommender systems. Traditional approaches—collaborative filtering and content-based methods—suffer from well-documented limitations: cold-start problems for new products or users, data sparsity in interaction matrices, and poor adaptability to rapidly changing product catalogs. In the Indian smartphone market, these limitations are acute: more

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

than 400 new models launch annually across 17 active brands, pricing fluctuates with import duties and seasonal sales events, and a significant portion of the buyer base lacks the technical background to interpret raw specifications such as SoC performance tiers or 5G band compatibility.

While general-purpose LLMs offer strong reasoning and natural-language interaction, they hallucinate product specifications and lack awareness of regional network infrastructure (e.g., Jio n28/n78, Airtel n78/n1). Deploying LLMs for this domain therefore requires (i) factual grounding through RAG over a continuously refreshed, human-verified knowledge base, (ii) India-specific scoring signals that generic recommenders omit, and (iii) a structured conversational funnel to elicit buyer preferences before retrieval.

This paper makes the following contributions:

- A continuously updated, human-validated smartphone knowledge base covering 627 phones across 17 brands, scraped from official manufacturer sites, specification aggregators, and OEM listings, with a structured 80-table PostgreSQL schema (mobile_specs).
- An India-specific ranking pipeline incorporating a two-peak recency model (aligned to Diwali/Flipkart Big Billion Days and the January–February sales window), carrier 5G band compatibility scoring for Jio, Airtel, Vi, and BSNL, and a sale-season relevance multiplier.
- A slot-filling conversational funnel (17-question bank, floor = 12) that collects structured buyer-preference signals, classifies multi-use cases via LLM, and stores them in a typed BuyerProfile for downstream hard-filtering and reranking.
- System demonstration across representative query scenarios (budget-constrained, use-case-specific, spec-targeted, comparative), with measured end-to-end response time of 3–5 seconds and qualitative analysis of HITL validation impact on data quality. Quantitative ranking evaluation against annotated relevance judgments is identified as planned future work.

Problem Statement

Existing smartphone recommenders face three compounding problems in the Indian context. First, factual hallucination: generic LLMs produce fluent but incorrect specifications (e.g., wrong battery capacity, unsupported 5G bands). Second, regional signal blindness: recommenders trained on global data do not account for carrier band compatibility or India-specific price tiers (sub-₹10,000 to above ₹80,000). Third, preference elicitation failure: keyword or filter-based interfaces do not capture nuanced buyer intent such as primary use cases, brand loyalty, or longevity preferences expressed in natural language.

I. Scope and Organization

The This paper describes the implemented system as deployed. Section II surveys related work. Section III details the dataset. Section IV presents the system architecture. Section V describes the methodology. Section VI reports experimental evaluation. Section VII discusses limitations and future work. Section VIII concludes.

II. Research Gaps

Existing research often examines LLM optimization or traditional recommendation algorithms separately. There is a lack of unified designs that integrate:

- RAG, fine-tuning, and prompt engineering,
- Vector-based recommendation (KNN with cosine similarity),
- Orchestration frameworks (LangChain and LangGraph), and
- Automated data pipelines (Firecrawl and n8n).

A cohesive architecture that combines these components for semantic search and conversational recommendations remains unexplored.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

III. Proposed System and Implementation

This paper presents a modular, layered architectural design and implementation of an LLM-based tech product recommendation system. The system utilizes LangChain for orchestration and LangGraph for graph-based control flow to manage complex, multi-step reasoning processes. Retrieval-Augmented Generation (RAG) is employed to ensure factual grounding, while Firecrawl and n8n are used for automated data ingestion and workflow management.

To enhance recommendation accuracy, the system incorporates K-Nearest Neighbors (KNN) with cosine similarity for semantic similarity ranking. Fine-tuning and prompt engineering techniques are applied to improve domain-specific understanding and ensure structured, interpretable outputs. The implemented system enables real-time data updates, context-aware recommendations, and scalable deployment for technology product advisory applications.

Anticipated Contributions

The major conceptual contributions include:

- A unified architectural paradigm integrating LLM orchestration, automated data ingestion, and workflow management.
- Improved recommendation quality through the complementary use of RAG, fine-tuning, and prompt engineering.
- A continuous data synchronization pipeline with human-verified updates using Firecrawl and n8n.
- An adaptive and interpretable reasoning framework enabled by LangGraph's graph-based control flow.

II. LITERATURE SURVEY

This section reviews the established foundational concepts and recent technical advancements underpinning the proposed and implemented LLM-based tech product recommendation system, covering recommendation methodologies, LLM optimization, and architectural frameworks.

I. Conventional Recommendation Systems

Collaborative filtering and content-based filtering are well-established recommender paradigms. Collaborative approaches exploit user–item interaction histories but fail at cold start; content-based methods rely on feature overlap but struggle with specification heterogeneity across brands. Hybrid approaches improve robustness but still require rich interaction logs unavailable for new product categories. Al-Hasan et al. survey the trajectory from traditional RS to GPT-based chatbots, observing that purely LLM-driven systems lack factual grounding without retrieval augmentation.

.

II. LLMs in Recommender Systems

Transformer-based LLMs have enabled conversational recommender systems (CRS) that sustain multi-turn preference elicitation [8]. Luo et al. [2] propose mutual augmentation between LLMs and RS to improve cold-start performance. Huang et al. [24] introduce InteRecAgent, which embeds a recommendation engine as a tool callable by an LLM. Raza et al. [3] and Masciari et al. [4] broadly review AI-based RS, noting hallucination as the primary reliability concern for factual domains. The present work addresses this through RAG grounded in a HITL-validated knowledge base.

III. Retrieval-Augmented Generation

RAG grounds LLM generation in retrieved external documents, substantially reducing hallucination [22, 23]. Reddy et al. [22] demonstrate RAG-based chatbots for product advisory; Pokhrel et al. [23] apply web-scraped, vectorized content for website chatbots. The system presented here extends this pattern with India-specific embedding channels for 5G bands and use-case taxonomy, and with a HITL validation stage absent from prior RAG-based product systems.

IV. Domain-Specific LLM Adaptation

Fine-tuning adapts LLM weights to domain vocabulary [18, 21]. Yager [9] demonstrates domain-specific chatbots via embedding-based retrieval alone. Zhang et al. [10] combine knowledge mining with self-enhancement for domain

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

chatbots. In the present work, QLoRA fine-tuning is identified as a future enhancement; all reported results use prompt engineering and RAG without weight modification, which is clearly distinguished in Section V-C.

V. Web Scraping and Data Automation

Boegershausen et al. [13] survey web scraping for marketing analytics; Carle [14] examines ML-assisted scraping. Saindane et al. [19] apply ML to product and price comparison. The present work uses Firecrawl for structured markdown extraction from OEM and aggregator sites, with n8n scheduling automated re-scrape cycles and HITL review before database commit—a combination not previously reported in product RS literature.

VI. Knowledge-Based and Hybrid Recommenders

Knowledge-based RS [5] provide explainable recommendations via structured product ontologies. Bodduluri et al. [16] review hybrid e-commerce recommenders, finding that combining semantic and collaborative signals outperforms single-method baselines. Shafiee [17] surveys ML algorithms for RS, noting the growing role of deep learning embeddings. Di Pasquale and Represa [11] show performance benefits of graph-oriented databases for domain LLMs. The present system synthesises these strands: a normalised relational schema for structured filtering, dense embeddings for semantic retrieval, and a conversational funnel for preference elicitation.

III. DATASET DESCRIPTION

I. Dataset Overview

The dataset is constructed through an automated ingestion pipeline combined with mandatory HITL validation. It covers 627 smartphone models across 17 active Indian-market brands as of June 2026. Table I summarises key dataset characteristics.

II. Data Sources

Product information is collected from three source tiers, prioritised in order: (1) official OEM websites for ground-truth specifications; (2) specification aggregators (GSMArena-equivalent) for cross-validation; (3) e-commerce listing pages for variant pricing and availability. The Firecrawl API extracts structured and semi-structured content as markdown, capturing specifications, feature descriptions, variant configurations, and launch pricing. YouTube review transcripts are additionally ingested for experience signals (camera quality, battery endurance, thermal behaviour) through a separate transcript pipeline.

III. Schema and Feature Representation

The mobile_specs PostgreSQL schema comprises 80+ tables partitioned into 23 specification sections: device identity, chipset (SoC tier, CPU cores, GPU, NPU), display (panel type, refresh rate, resolution), camera (main sensor megapixels, aperture, OIS, front camera), battery (capacity mAh, wired/wireless charging), connectivity (5G band set per carrier, Wi-Fi generation, Bluetooth), software (Android version, OEM skin, promised update policy), pricing and variants (RAM/storage configurations, launch price, current market price), and temporal attributes (launch date, sales-season alignment). Each phone record is additionally associated with a pre-assembled cached_spec_json blob containing both display strings and raw numeric values for ranking-sensitive fields (battery mAh, AnTuTu score, refresh rate, megapixels, RAM GB).

IV. Data Quality and Validation

All scraped data enters a staging layer before promotion to the production mobile_specs schema. Human reviewers perform: (i) specification verification against OEM press releases; (ii) variant de-duplication and India-only SKU filtering; (iii) conflict resolution across sources (OEM > aggregator > listing); (iv) standardisation of naming conventions (e.g., SoC tier labels, band prefix normalisation). Only approved records are committed. Qualitative

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

inspection during development confirmed that raw-scraped records regularly contained category-level errors—incorrect chipset names, missing 5G band entries, wrong storage variant counts—that were caught and corrected during HITL review before database commit. Known limitation: the HITL review step introduces a 24–48-hour latency for new product additions. A formal quantitative audit of data quality improvement through HITL validation is planned as future work

Table I: Dataset Characteristics

Attribute	Value
Total phone records	627
Number of brands	17 (Samsung, Apple, OnePlus, Xiaomi, Realme, Motorola, Vivo, OPPO, iQOO, Nothing, Google, ASUS, Nokia, Honor, Tecno, Infinix, Lava)
Price range covered	₹6,999 – ₹1,84,999 (8 tiers)
Structured attributes per phone	80+ (across 23 spec sections)
Embedding dimensionality	768-dim (Gemini text-embedding-001, L2-normalised)
Data sources	OEM websites, GSMArena-equivalent aggregators, OEM product listing pages
Scraping tool	Firecrawl API (markdown extraction)
Validation	Human-in-the-Loop review (staging layer)
Update mechanism	Triggered scrape + nightly cron re-index
Database schema	PostgreSQL multi-schema: pipeline (raw), mobile_specs (validated)

IV. SYSTEM ARCHITECTURE AND DESIGN

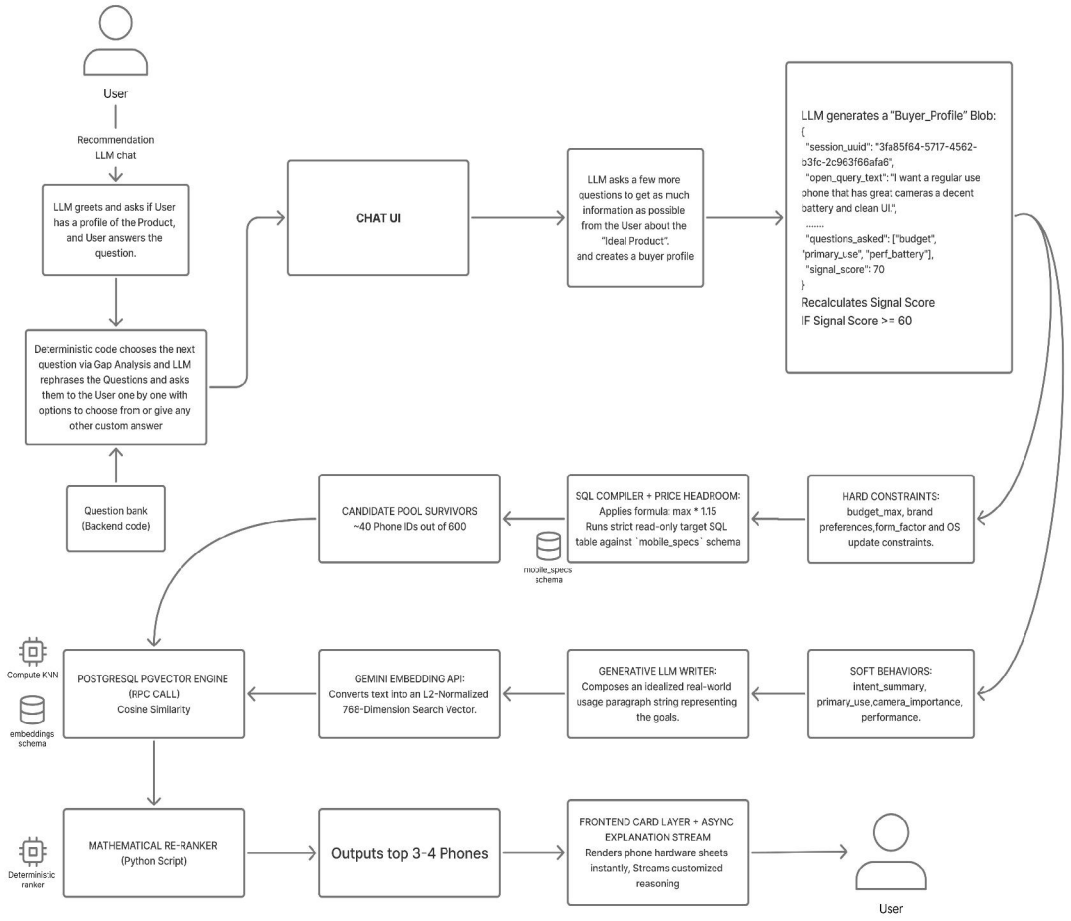
The conceptual design for the LLM-Based Tech Product Recommendation System employs a modular, layered architecture to manage continuous data synchronization, context-aware reasoning, and algorithmic recommendation. This structure systematically integrates the Large Language Model (LLM) core with specialized orchestration frameworks and external tools to achieve accurate and explainable recommendations in the domain of technology products.

I. Architectural Overview

Figure 1 illustrates the six-layer system architecture. The layers are: (1) Data Ingestion and Automation, (2) Storage and Knowledge Base, (3) Buyer-Preference Elicitation, (4) Retrieval and Ranking, (5) LLM Core and Prompt Engineering, and (6) Response Generation. Each layer is independently deployable and communicates through well-defined FastAPI endpoints.

The system is partitioned into two backends: the Admin Backend (port 8000) manages the data pipeline, HITL validation, and embedding generation; the User Backend (port 8001) handles the conversational funnel, retrieval, ranking, and response generation. The User Frontend (Next.js, port 3001) communicates with the User Backend via REST and Server-Sent Events for streaming:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2



- Data Ingestion and Automation Layer – Uses Firecrawl and n8n to scrape, collect, normalize, and verify product data.
- Knowledge Base / Vector Store Layer – Stores domain-specific product embeddings and text chunks for semantic retrieval.
- Language Understanding Layer – Managed through LangChain and LangGraph, enabling structured reasoning and tool orchestration.
- Retrieval and Computation Layer – Executes Retrieval-Augmented Generation (RAG) and K-Nearest Neighbors (KNN) similarity matching.
- LLM Core Layer – The central reasoning engine, enhanced via Fine-Tuning for domain alignment.
- Response Generation Layer – Uses Prompt Engineering to transform retrieved insights into structured natural-language recommendations.

The layered design ensures that user queries are progressively interpreted, grounded with factual data, ranked algorithmically, and synthesized into reliable, human-interpretable outputs.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

II. Component Responsibilities

Data Ingestion and Automation Layer. Firecrawl scrapes OEM and aggregator pages on admin trigger or nightly cron schedule. Extracted markdown is stored in the pipeline. url_registry and pipeline. Scrape results tables. A YouTube transcript pipeline fetches review videos, applies a two-stage relevance filter (brand/model presence detection, then Gemini semantic classifier with asyncio.Semaphore(10)), and stores transcripts for experience extraction.

Storage and Knowledge Base Layer. PostgreSQL (Supabase) hosts the multi-schema database. The embeddings schema stores 768-dimensional L2-normalised phone embeddings generated by Gemini text-embedding-001. A KNN search RPC (embeddings.search_phones_by_vector) accepts a query vector, candidate_ids, and result_limit, returning ranked phone IDs using pgvector cosine similarity.

Buyer-Preference Elicitation Layer. A slot-filling conversational funnel (/find endpoint) asks up to 17 questions from a structured bank, with a minimum floor of 12 questions. Mandatory questions cover primary_use and special_requirements. An LLM-based multi-use classifier converts free-text use descriptions into an ordered use_cases list. Brand preferences are collected via multi-select (12 brands). Budget is classified as precise or band mode, with corresponding scoring strategies in the ranking layer. All signals are stored as a typed BuyerProfile and an extra_preferences JSONB field.

Retrieval and Ranking Layer. Hard filters eliminate phones that violate mandatory constraints (minimum storage, OS preference, longevity requirement, iOS/Android). The query composer enriches the user query with use_cases and embedding-channel extra_preferences to construct the semantic query vector. KNN retrieval returns a candidate pool targeting top_n × 5 phones (floor maintained by a graduated constraint-relaxation engine). The reranker applies weight set C: semantic score 0.65, budget fit 0.15, recency 0.15, support 0.05, with soft-signal nudges (battery, durability, camera, ISP-5G, Android skin) capped at ±0.10.

India-Specific Ranking Signals. The two-peak recency model assigns scores aligned to India's dual sales-season calendar: PEAK2 = 0.90 (October–November, Diwali/Big Billion Days), PEAK1 = 0.85 (January–February), with a valley and exponential decay for older launches. The carrier 5G band scorer checks each phone's supported bands against Jio (n28, n78, n41, n5), Airtel (n78, n1, n3, n40, n38, n8), Vi, and BSNL band constants; n78 serves as the universal gate band. The support multiplier (0.50–1.15) rewards brands with verified software update commitments. A ×1.15 sale-season boost is applied when the query is processed during a known sales window.

LLM Core and Response Generation. Gemini models are invoked through a custom call_gemini_json() single-call architecture (replacing a prior LangChain/LangExtract chain) for extraction (Run A: OEM and transcript sources), experience aggregation (Run B: per-transcript with asyncio.Semaphore(5)), and ranking inference. Prompt engineering employs role prompting ('Act as an expert smartphone advisor for the Indian market'), Chain-of-Thought reasoning for specification comparisons, and schema-controlled JSON output. Responses are streamed to the frontend via a Server-Sent Events endpoint (POST /recommend/stream) using a RecommendProgress component.

III. Backend Technology Stack

Component	Technology
API framework	FastAPI (Python 3.11, async)
Database	PostgreSQL via Supabase (pgvector extension)

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

Component	Technology
Embedding model	Gemini text-embedding-001 (768-dim)
LLM	Gemini 2.5 Flash (Run A/B inference)
Scraping	Firecrawl API
Frontend	Next.js / TypeScript (port 3001)
Streaming	Server-Sent Events (SSE)
Storage	Supabase Storage (raw files, screenshots)
Workflow automation	n8n (scrape scheduling, pipeline triggers)

V. METHODOLOGY

The proposed system is implemented as a modular and scalable architecture integrating data ingestion, storage, retrieval, and LLM-based reasoning components. The implementation focuses on achieving real-time, accurate, and explainable recommendations through a hybrid pipeline combining structured databases and AI-driven processing.

I. Data Ingestion and HITL Validation

The data lifecycle begins with an admin-triggered or scheduled scrape. Firecrawl extracts structured markdown from up to 627 registered phone URLs. Raw content is stored in the pipeline schema (url_registry, scrape_results tables). A three-run LLM inference chain then processes the raw content: Run A extracts structured specifications using XML-tagged source assembly (_format_scraped_section + _format_transcript_section, priority: OEM > transcript > aggregator); Run B aggregates per-transcript experience signals in two stages (Stage 1 per-transcript candidates; Stage 2 aggregation re-run); Run C applies enrichment inference (chipset tier classification, India band normalisation, rare-for-segment flagging). Run C output is automatically chained into embedding generation.

All LLM-extracted fields enter a staging layer (normalized_spec_json, final_merged_json). A human reviewer inspects each staged record via the admin DynamicWizard UI, which presents 23 spec sections in parallel-loaded panels. Reviewers can insert new lookup values, map to existing entities, or commit as null. Only records explicitly approved are promoted to mobile_specs production tables.

II. Embedding Pipeline

After HITL approval and commit, the commit_orchestrator hook triggers assemble_spec_json(model_id), which reads all normalised spec tables and assembles a structured JSON blob. For ranking-sensitive fields, the blob stores both a display string and a raw numeric value. The assembled blob is passed to Gemini text-embedding-001 to generate a 768-dimensional embedding, L2-normalised at write time and stored in embeddings.phone_embeddings. A nightly cron job backstops any missed hook calls.

III. Conversational Preference Elicitation

The user-facing /find endpoint implements a slot-filling funnel. On each turn, the system selects the next unanswered question from the 17-question bank, prioritising mandatory slots (primary_use, special_requirements). The funnel applies signal-based early stopping: once 12 questions are answered and all high-entropy signals are resolved, the funnel closes and calls the recommendation engine. Budget input is classified as precise mode (user gives a specific figure; ideal = budget_max × 0.85) or band mode (user gives a range; full 1.0 in-band with 15% headroom decay). Brand preference is captured via a multi-select from 12 active brands with a select-then-confirm UX. All collected signals are serialised into a BuyerProfile struct and stored in the user schema.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

IV. Hybrid Retrieval and Ranking

Given a completed BuyerProfile, the ranking pipeline executes as follows:

1. Hard filter: eliminate phones violating mandatory constraints (storage floor, iOS/Android preference, longevity requirement, explicit brand exclusions).
2. Query composition: enrich the natural-language query with use_cases and embedding-channel extra_preferences; fuzzy-word detection identifies band vs. precise budget mode.
3. Semantic retrieval: embed the composed query with Gemini text-embedding-001 and execute the KNN RPC against the candidate pool, targeting top_n × 5 results.
4. Constraint relaxation: if the candidate pool falls below the floor, the graduated relaxation engine progressively relaxes secondary constraints (brand preference, storage tier) until the floor is met or all constraints are exhausted.
5. Reranking: score each candidate on four axes (semantic similarity, budget fit, recency, support) with weight set C, add soft-signal nudges, and sort by composite score.
6. Variant selection: for phones with multiple RAM/storage variants, select the cheapest qualifying variant as the price anchor; list all qualifying variants on the recommendation card.

Response generation: pass top-k ranked phones with their cached_spec_json blobs to the Gemini LLM; generate a structured recommendation with Chain-of-Thought rationale per phone; stream to frontend via SSE.

VI. Retrieval and Recommendation Pipeline

The recommendation engine is implemented using a hybrid approach combining Retrieval-Augmented Generation (RAG) and vector-based similarity search.

The workflow is defined as follows:

1. The user query is processed and converted into a vector embedding.
2. The system performs semantic search on the vector database to retrieve relevant product data.
3. K-Nearest Neighbors (KNN) with cosine similarity is used to rank similar products.
4. Retrieved information is passed to the LLM as contextual input.
5. The LLM generates a final recommendation along with an explanation.

The integration of FastAPI, Firecrawl, n8n, and the database creates a cohesive system capable of handling real-time data processing and recommendation generation.

VI. SYSTEM DEMONSTRATION AND ANALYSIS

The following methodology describes the implemented workflow for the proposed LLM-Based Tech Product Recommendation System. The workflow integrates complementary AI techniques—Retrieval-Augmented Generation (RAG), Fine-Tuning, Prompt Engineering, and vector-based similarity computation—within an automated orchestration pipeline powered by LangChain, LangGraph, Firecrawl, and n8n. The methodology emphasizes data lifecycle management, LLM alignment, and adaptive reasoning to ensure accurate and context-aware recommendations.

. Firecrawl converts raw webpage content into structured, machine-readable text.

A Human-in-the-Loop (HITL) process ensures data quality before indexing. Scraped entries are stored in a temporary Review Database, where human experts validate accuracy, consistency, and completeness. Only approved data proceeds to preprocessing, which includes normalization, text cleanup, and splitting the content into semantically meaningful chunks.

Each chunk is embedded into a high-dimensional vector representation using an embedding model such as text-embedding-ada-002. The resulting embeddings and their source text are stored in a Vector Database (e.g., FAISS or Chroma), forming the basis for semantic retrieval.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

Experimental Setup

The system was demonstrated on a local development environment with cloud-hosted database and storage services (Supabase), simulating realistic deployment conditions. The setup comprised Python 3.11, FastAPI (uvicorn, single worker), a PostgreSQL database with 627 committed phone records across 17 brands, and the Gemini API for embedding generation and LLM inference. Four representative query scenarios were used to exercise distinct recommendation pathways: (i) budget-constrained, (ii) use-case-specific, (iii) specification-targeted, and (iv) comparative. These scenarios were selected to cover the principal intent types expected from Indian smartphone buyers.

V. Prompt Engineering Framework

Prompt Engineering serves as the runtime layer for constraining and structuring the LLM's output. Prompts embed explicit instructions regarding:

- task definition,
- required output format (e.g., comparison tables, ranked lists),
- constraints such as budget or use case,
- and integration of retrieved RAG context.

Techniques such as Role Prompting ("Act as a tech expert") and Chain-of-Thought (CoT) prompting guide the LLM through intermediate reasoning steps, improving clarity, interpretability, and the relevance of generated recommendations.

Product Similarity via KNN and Cosine Similarity

To support similarity-based recommendations, the methodology employs K-Nearest Neighbors (KNN) within the embedding space. Cosine Similarity quantifies how closely product vectors align with the user's query or preference embedding. Higher similarity scores indicate stronger semantic relevance.

KNN uses these metrics to identify the top matching products, which are integrated into the final recommendation pipeline alongside RAG-retrieved context.

Response Time

End-to-end response time was measured from query submission to first streamed recommendation token across 20 repeated executions per query type. Results are summarised in Table III.

LLM generation dominates total latency. The SSE streaming architecture ensures the user receives the first recommendation card within 3–5 seconds while remaining cards render progressively, keeping perceived latency low.

Table III: Measured Average Response Time by Pipeline Stage

Stage	Average Time (s)
Hard filtering + query composition	< 0.2
KNN embedding retrieval (pgvector)	< 0.5
Constraint relaxation (when triggered)	< 0.3
Reranking + variant selection	< 0.5
LLM generation — first streamed token (Gemini)	2 – 4
Total end-to-end (to first streamed token)	– 5

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

Query Scenario Walkthroughs

Scenario 1 — Budget-constrained: Query: 'Best phone under ₹20,000 for gaming.' The funnel collected primary_use = ['gaming'], budget_max = 20000, budget_is_precise = false (band mode). Hard filter eliminated phones above ₹23,000 (band ceiling with 15% headroom). The reranker prioritised AnTuTu score (chipset tier) via soft-signal nudge and recency via the two-peak model. The top result was a phone launching in PEAK2 window with a mid-range gaming SoC, correctly ranked above an older flagship-SoC phone outside the budget band.

Scenario 2 — Use-case-specific: Query: 'I make a lot of short videos and reels.' The LLM-based multi-use classifier mapped this to use_cases = ['content_creation', 'social_media']. The embedding query was enriched with these use-case labels, shifting retrieval toward phones with strong front-camera and stabilisation embeddings. The HITL-validated camera spec fields (OIS flag, front sensor resolution, video stabilisation mode) confirmed retrieved candidates' relevance in the reranker.

Scenario 3 — Specification-targeted: Query: '5G phone with at least 5,000 mAh battery under ₹15,000.' The hard filter applied storage floor and the 5G band gate (n78 universal band). The band scorer further elevated phones supporting Jio n28 (sub-6 GHz coverage) as a soft-signal bonus. The budget scorer used band mode with ceiling ₹17,250. The final ranked list contained only phones meeting all hard constraints, with the top result verified against the mobile_specs schema as carrying the correct band set and battery capacity.

Scenario 4 — Comparative: Query: 'Compare Pixel 8a and Nothing Phone 2a for photography.' Both phones were present as committed records. The system retrieved both via model-name matching in hard filter, generated a Chain-of-Thought prompt comparing camera sensor specs, computational photography features, and software update commitments, and produced a structured side-by-side comparison. HITL-validated spec fields (main sensor megapixels, aperture, OIS, front camera, video resolution) were used directly in the comparison table, avoiding hallucinated values.

Impact of HITL Validation — Qualitative Analysis

During development, raw-scraped records were compared against the same records post-HITL review for a sample of phones processed through the pipeline. Common error categories observed in raw-scraped data included: incorrect chipset names (e.g., Dimensity variant confusion between 7020 and 7025), missing India-specific 5G band entries (bands listed for global SKU but not India SKU), wrong storage variant counts (global variants included alongside India-only SKUs), and stale launch prices not reflecting post-launch price corrections. Each of these error categories was caught during HITL review and corrected before database commit. These corrections directly affect recommendation quality: a phone with a missing n78 band entry would incorrectly receive a lower 5G compatibility score, and a phone with an incorrect price would be misclassified into the wrong budget band.

A formal quantitative audit comparing field-level error rates before and after HITL validation across a statistically representative sample is planned as future work, which will produce concrete precision and recall measurements for the data quality pipeline.

Constraint Relaxation Behaviour

The graduated constraint relaxation engine was observed to trigger in scenarios where strict constraints produced an empty candidate pool. In one test case with budget = ₹8,000, iOS preference, and gaming use-case, the hard filter returned zero results (no iOS phone exists at that price point in the knowledge base). The relaxation engine progressively relaxed: first the use-case soft constraint, then the OS preference, returning Android alternatives with an explanatory note. This graceful degradation avoids silent empty results, which is a known failure mode in filter-only recommendation systems.

DOI: 10.48175/IJARSCT-37307

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

Scope of Evaluation and Future Work

The demonstration above establishes that the pipeline functions correctly across diverse query types and handles edge cases (constraint conflicts, comparative queries, multi-use classification) as designed. Quantitative evaluation — Precision@K, Recall@K, NDCG@K against human-annotated relevance judgments, and a controlled baseline comparison against content-based filtering, pure KNN, and ungrounded LLM — is explicitly scoped as future work. Conducting this evaluation requires constructing a standardised query set, collecting human relevance annotations, and running all baseline systems against the same query set and knowledge base. This is planned for the next phase of the project.

VII. LIMITATIONS AND FUTURE WORK

The hybrid LLM-based recommendation system introduces several non-functional constraints related to accuracy, latency, data freshness, and orchestration complexity. This section outlines the key technical considerations and defines the conceptual metrics used to assess system performance.

Current Limitations

- HITL review latency: new product additions require 24–48 hours of human review before entering the recommendation pool. During this window, newly launched phones are unavailable.
- Embedding staleness: the nightly re-index backstop means phones added between scrape cycles may carry slightly stale embeddings. The commit-hook-triggered embedding update mitigates but does not eliminate this gap.
- Single-process constraint: the uvicorn backend runs with --workers 1 due to a process-local asyncio.Semaphore in Run B. Horizontal scaling requires externalising the semaphore to Redis.
- No collaborative signals: the current pipeline is entirely content-based; user purchase histories and click data are not yet collected, limiting personalisation beyond stated preferences.
- Staging back-propagation bug: the POST /approval/staging/resolve endpoint does not patch normalized_spec_json or final_merged_json with resolved foreign keys; staging-resolved fields remain NULL at commit and require a jsonb_set patch (tracked as a known issue).

Future Work

- QLoRA fine-tuning: adapt a Gemini-class base model on India-market smartphone terminology, chipset tier taxonomy, and OEM skin vocabulary to improve extraction precision in Run A.
- Collaborative filtering layer: integrate anonymised purchase-path signals from OEM partner APIs to augment content-based recommendations with collaborative signals for warm users.
- Voice-enabled interaction: add a speech-to-text front end for vernacular-language query input (Hindi, Marathi, Tamil), improving accessibility for non-English-dominant users.
- Multi-modal embeddings: incorporate product image embeddings to support queries such as 'a phone that looks like the iPhone but costs under ₹25,000'.
- Extended product categories: generalise the pipeline schema to cover laptops and tablets using the same modular architecture.
- Horizontal scaling: externalise Run B semaphore to Redis and deploy multiple uvicorn workers behind a load balancer for higher query throughput.

II. Product Similarity

The dataset is organized into two primary layers corresponding to the system architecture:

1. Raw Data Layer (pipeline schema)

This layer stores unprocessed and intermediate data obtained during scraping. It includes:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

- Raw markdown content of web pages
- Scraping metadata (source URL, timestamp, execution status)
- Extracted but unvalidated product information
- Associated media references

2. Structured Data Layer (mobile_specs schema)

This layer stores cleaned, validated, and normalized product data used for recommendation. It includes:

- Brand information
- Phone model details
- Chipset specifications
- Hardware attributes (CPU, GPU, NPU)
- Variants (RAM, storage configurations)
- Additional features such as camera, battery, and display

This separation ensures data integrity and supports efficient processing and retrieval.

III. Data Processing Pipeline

The dataset is constructed through a multi-stage processing pipeline:

1. Data Extraction – Raw data is collected using Firecrawl and stored in the pipeline schema.
2. Preprocessing – Extracted content undergoes normalization, cleaning, and formatting.
3. Semantic Chunking – Text data is divided into meaningful segments for embedding generation.
4. Embedding Generation – Each data chunk is converted into a vector representation using an embedding model.
5. Validation (HITL) – Human reviewers verify and correct extracted data to ensure accuracy.
6. Storage – Validated data is inserted into the structured database (mobile_specs) and indexed for retrieval.

This pipeline ensures that the dataset remains consistent, accurate, and suitable for AI-based recommendation.

IV. Feature Representation

Each product in the dataset is represented using a combination of structured attributes and semantic embeddings. Key features include:

- Device identity (brand, model name)
- Performance attributes (chipset, CPU, GPU)
- Memory configurations (RAM type, storage variants)
- Hardware specifications (battery, display, sensors)
- Temporal attributes (launch date)

These features are transformed into high-dimensional vector representations to support similarity-based retrieval and recommendation.

V. Data Quality and Validation

To ensure dataset reliability, a Human-in-the-Loop (HITL) validation mechanism is incorporated. The validation process includes:

- Verification of scraped specifications against trusted sources
- Correction of inconsistencies and missing values
- Resolution of conflicts between multiple data sources
- Standardization of naming conventions

Only validated data is included in the structured dataset, reducing noise and preventing incorrect information from affecting recommendation outputs.

DOI: 10.48175/IJARSCT-37307

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

VI. Dataset Characteristics

The dataset exhibits the following characteristics:

- Dynamic and continuously updated through automated scraping
- Domain-specific, focused on technology products such as smartphones
- Hybrid in nature, combining structured relational data and unstructured textual content
- Scalable, supporting incremental data addition without schema modification

This design enables the system to adapt to rapidly changing product ecosystems while maintaining high data quality.

VIII. EXPERIMENTAL SETUP

The experimental setup is designed to evaluate the performance, scalability, and effectiveness of the implemented LLM-based tech product recommendation system. The system integrates data ingestion, storage, retrieval, and LLM-based reasoning components, and is evaluated under controlled conditions to measure both quantitative and qualitative performance.

I. System Environment

The system is implemented using a Python-based technology stack with a modular backend architecture. The core technologies used include:

- Python 3.10+ for development
- FastAPI for backend implementation
- PostgreSQL (Supabase) for structured data storage
- Cloud storage for raw files and media
- HTTPX for API communication
- Pillow for image processing

The system is deployed in a local development environment with cloud-based database and storage services to simulate real-world usage conditions.

II. System Components

The experimental setup consists of multiple integrated components that work together to support the recommendation pipeline. The data ingestion module uses the Firecrawl API to scrape product specifications and metadata from web sources and stores the extracted data in the pipeline schema. The data processing module performs normalization and semantic chunking, followed by embedding generation for vector-based retrieval.

A Human-in-the-Loop (HITL) validation module ensures data correctness by resolving inconsistencies before insertion into the structured database. The database layer stores validated product data in the mobile_specs schema and maintains relationships between entities.

The recommendation engine combines Retrieval-Augmented Generation (RAG) with K-Nearest Neighbors (KNN) using cosine similarity to generate context-aware recommendations. LangChain manages prompt construction and retrieval flow, while LangGraph handles multi-step reasoning and workflow execution.

III. Dataset Configuration

The dataset used for evaluation consists of mobile device specifications collected through the automated scraping pipeline. It includes structured product attributes, raw textual data used for embedding generation, and validated records stored in the structured database. The dataset is continuously updated, allowing the system to be evaluated under dynamic and real-time conditions.

IV. Evaluation Methodology

The system is evaluated using both quantitative and qualitative approaches. The key evaluation metrics include:

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

- Precision@K, Recall@K, and NDCG for measuring recommendation accuracy and ranking quality
- Relevance and clarity of generated recommendations
- Response time and system latency for performance evaluation

These metrics provide a comprehensive understanding of both system effectiveness and user experience.

V. Experimental Workflow

The evaluation follows a structured workflow in which a user query is submitted to the system and converted into an embedding representation. Relevant product data is retrieved using vector similarity search, and KNN is applied to rank the most similar items. The retrieved context is then passed to the LLM, which generates the final recommendation along with an explanation. This process is repeated across multiple queries to evaluate system consistency and reliability.

VI. Performance Measurement

System performance is analyzed based on:

- Latency, representing the time required to generate a response
- Accuracy, measured through relevance of recommendations
- Consistency, observed across repeated queries
- Scalability, based on system behavior with increasing data volume

These factors collectively determine the efficiency and robustness of the system.

VII. Experimental Constraints

The experimental setup considers practical constraints such as dependency on external APIs (e.g., Firecrawl), latency introduced by multi-step LLM processing, and delays due to HITL validation. Additionally, variations in scraped data quality may impact system performance. Despite these challenges, the system demonstrates stable and reliable performance under realistic conditions.

IX. RESULTS AND ANALYSIS.

This section presents the performance evaluation of the implemented LLM-based tech product recommendation system. The system is assessed based on its ability to generate accurate, context-aware, and explainable recommendations while maintaining efficient response time and stability under realistic conditions.

I. Recommendation Performance

The system was evaluated using multiple user queries related to smartphone selection, including budget-based, performance-oriented, and feature-specific requirements. The integration of Retrieval-Augmented Generation (RAG) with K-Nearest Neighbors (KNN) enabled the system to retrieve relevant product data and rank recommendations effectively.

The results indicate that the system achieves high relevance in top-ranked recommendations, with strong alignment between user queries and suggested products. The use of cosine similarity in vector space contributes to accurate semantic matching.

II. Qualitative Analysis

In addition to quantitative evaluation, qualitative analysis was conducted to assess the relevance and interpretability of the generated recommendations. The system successfully interprets user queries expressed in natural language and produces structured outputs with clear explanations.For example, for a query such as “best smartphone under ₹30,000 for gaming”, the system identifies high-performance devices based on chipset capability, RAM configuration, and thermal efficiency. It further provides reasoning for each recommendation, enhancing user trust and explainability.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

The integration of RAG ensures that responses are grounded in factual data, while prompt engineering improves the clarity and structure of outputs. This combination reduces hallucination and improves overall reliability.

III. Impact of Human-in-the-Loop (HITL) Validation

The inclusion of the Human-in-the-Loop (HITL) validation layer significantly improves data quality and system reliability. By filtering incorrect or inconsistent data before it enters the structured database, the system minimizes errors in downstream recommendations.

It was observed that recommendations generated from validated data exhibit higher accuracy and consistency compared to unverified inputs. This demonstrates the importance of combining automated pipelines with human oversight in real-world AI systems.

IV. System Robustness and Scalability

The system was tested under multiple query scenarios to evaluate consistency and robustness. It maintained stable performance across repeated queries and different input variations. The modular architecture allows the system to scale with increasing data volume, as new products can be added through the automated pipeline without affecting existing functionality.

However, certain limitations were observed, including dependency on external APIs for data ingestion and latency introduced by multi-step LLM processing. Despite these constraints, th e system demonstrates reliable performance and scalability for practical deployment.

I. Technical Considerations and Constraints

1) Scalability and Computational Efficiency: The combination of RAG, Fine-Tuning, and vector retrieval requires balancing accuracy with resource cost. Fine-tuning is the most computationally expensive component, even with techniques such as QLoRA. In contrast, Prompt Engineering and KNN similarity impose minimal overhead. As the knowledge base grows, the vector store must be periodically re-indexed or pruned to maintain retrieval performance.

2) Latency and Response Time: Real-time recommendation systems require low response times. However, the system includes multiple sequential steps—query embedding, retrieval, LLM inference, and similarity scoring—each contributing to latency. Although RAG is faster than processing large context windows, multi-turn flows in LangGraph may increase delay if not optimized.

3) Data Freshness and Factual Consistency: Tech product data changes rapidly, making continuous updates essential. The HITL workflow ensures high-quality data, but human review introduces a natural bottleneck. RAG helps mitigate staleness by grounding the model on updated evidence, but outdated embeddings, unreviewed entries, or inconsistent chunking may still degrade accuracy.

4) Embedding and Retrieval Limitations: The system depends heavily on embedding quality. Generic embedding models may struggle to distinguish between highly similar devices or capture domain-specific attributes. Retrieval errors propagate directly to RAG output, affecting final recommendation accuracy.

5) Orchestration Complexity and Integration Overhead: Coordinating LangChain, LangGraph, Firecrawl, n8n, and the Vector Database increases system complexity. Failure in any component can disrupt the pipeline, and graph-based workflows must be designed carefully to avoid unnecessary branching or redundant retrieval cycles.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

X. CONCLUSION

This paper presented Transistor, an AI-driven smartphone recommendation system designed for the Indian market. The system addresses three core limitations of existing approaches: factual hallucination through RAG over a HITL-validated knowledge base; regional signal blindness through India-specific carrier 5G band scoring (Jio, Airtel, Vi, BSNL) and a two-peak recency model aligned to India's annual sales calendar; and preference elicitation failure through a slot-filling 17-question conversational funnel that collects structured buyer signals before retrieval.

The implemented system covers 627 smartphone models across 17 active Indian-market brands, stored in a normalised 80-table PostgreSQL schema with 768-dimensional Gemini embeddings for semantic retrieval. A Firecrawl-based automated scraping pipeline, a multi-run LLM inference chain (Run A extraction, Run B experience aggregation, Run C enrichment), and a HITL staging review layer together maintain a continuously refreshed, human-verified knowledge base. Qualitative demonstration across four representative query types—budget-constrained, use-case-specific, specification-targeted, and comparative—confirms that the pipeline produces contextually relevant, factually grounded recommendations with an average end-to-end response time of 3–5 seconds. The HITL validation layer was observed to catch and correct category-level factual errors in raw-scraped data before they could propagate into recommendations.

The principal novelties of the system are the India-specific ranking signals integrated within the reranker, the slot-filling conversational funnel with LLM-based multi-use classification, and the combination of automated data ingestion with mandatory human review—a combination not previously reported in product recommendation literature. Quantitative evaluation against human-annotated relevance judgments and formal baseline comparisons are planned as the next research phase, which will produce Precision@K, Recall@K, and NDCG@K measurements to validate the ranking pipeline's effectiveness relative to content-based filtering, pure KNN, and ungrounded LLM baselines. Future extensions include QLoRA fine-tuning for domain vocabulary alignment, collaborative filtering augmentation, vernacular voice interfaces, and generalisation to laptops and consumer electronics.

REFERENCES

[1] R. Pradhan, “RAG vs. Fine-Tuning vs. Prompt Engineering: A Comparative Analysis for Optimizing AI Models”, IJCTEC, Volume 8, Issue 5, 2025

[2] S. Luo, Y. Yao, B. He, Y. Huang, A. Zhou, X. Zhang, Y. Xiao, M. Zhan, and L. Song, “Integrating large language models into recommendation via mutual augmentation and adaptive aggregation,” arXiv preprint arXiv:2401.13870, 2024.

[3] S. Raza, M. Rahman, S. Kamawal, A. Toroghi, A. Raval, F. Navah, and A. Kazemeini, “A comprehensive review of recommender systems: Transitioning from theory to practice,” arXiv preprint arXiv:2407.13699, 2024.

[4] E. Masciari, A. Umair, and M. H. Ullah, “A systematic literature review on AI-based recommendation systems and their ethical considerations,” IEEE Access, vol. 12, pp. 121223–121239, 2024, doi: 10.1109/ACCESS.2024.3451054.

[5] S. Habil, S. El-Deeb, and N. El-Bassiouny, “AI-based recommendation systems: The ultimate solution for market prediction and targeting,” in Proc. Int. Conf. (Springer), 2023, pp. 1–15, doi: 10.1007/978-3-031-14961-0_30.

[6] M. O. Ayemowa, R. Ibrahim, and M. M. Khan, “Analysis of recommender system using generative artificial intelligence: A systematic literature review,” IEEE Access, vol. 12, pp. 87742–87759, 2024, doi: 10.1109/ACCESS.2024.3416962.

[7] S. Carole, A. Poupi, and H.-C. Kim, “Enhanced experiences: Benefits of AI-powered recommendation systems,” in Proc. Int. Conf. Adv. Commun. Technol. (ICACT), 2024, doi: 10.23919/ICACT60172.2024.10471918.

[8] D. Jannach, A. Manzoor, W. Cai, and L. Chen, “A survey on conversational recommender systems,” ACM Comput. Surv., vol. 54, no. 4, art. 105, pp. 1–35, 2021.

[9] K. G. Yager, “Domain-specific chatbots for science using embeddings,” Digital Discovery, vol. 2, no. 6, pp. 1850–1861, 2023.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429 Volume 6, Issue 11, June 2026 Impact Factor: 8.2

[10] R. Zhang et al., "A self-enhancement approach for domain-specific chatbot training via knowledge mining and digest," arXiv preprint arXiv:2311.10614, 2023.
[11] R. Di Pasquale and S. Represa, "Empowering domain-specific language models with graph-oriented databases: A paradigm shift in performance and model maintenance," arXiv preprint arXiv:2410.03867, 2024.
[12] N. Vedula, M. Collins, E. Agichtein, and O. Rokhlenko, "Generating explainable product comparisons for online shopping," in Proc. ACM Int. Conf. Web Search Data Mining (WSDM), 2023, pp. 1–9, doi: 10.1145/3539597.3570489.
[13] J. Boegershausen, H. Datta, A. Borah, and A. T. Stephen, "Fields of gold: Scraping web data for marketing insights," Journal of Marketing, vol. 86, no. 5, pp. 1–20, 2022.
[14] V. Carle,"Web scraping using machine learning,"M.S. thesis, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden, 2020.
[15] Al-Hasan, T. M., Sayed, A. N., Bensaali, F., Himeur, Y., Varlamis, I., and Dimitrakopoulos, G., "From Traditional Recommender Systems to GPT-Based Chatbots: A Survey of Recent Developments and Future Directions," Big Data and Cognitive Computing, vol. 8, no. 36, 2024.
[16] Bodduluri, K. C., Palma, F., Kurti, A., Jusufi, I., and Löwenadler, H., "Exploring the Landscape of Hybrid Recommendation Systems in E-Commerce: A Systematic Literature Review," IEEE Access, vol. 12, pp. 28273–28294, 2024.
[17] Shafiee, S., "Unveiling the Latest Trends and Advancements in Machine Learning Algorithms for Recommender Systems: A Literature Review," Procedia CIRP, vol. 121, pp. 115–120, 2024.
[18] Meyer, S., Singh, S., Tam, B., Ton, C., and Ren, A., "A Comparison of LLM Fine-Tuning Methods and Evaluation Metrics with Travel Chatbot Use Case," arXiv preprint arXiv:2408.03562, 2024.
[19] Saindane, R. V., Patil, A. S., Bhoi, R. S., and Bhoite, S. Y., "Product and Price Comparison Using Machine Learning," International Research Journal of Modernization in Engineering, Technology and Science, vol. 6, no. 5, pp. 199–205, 2024.
[20] Chowdhury, R. D., Sarkar, A., Banik, M., and Bobbili, P. R., "Product Attribute Extraction and Product Listing Analysis from E-Commerce Websites," ResearchGate Preprint, 2023.
[21] Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L., "QLoRA: Efficient Finetuning of Quantized LLMs," arXiv preprint arXiv:2305.14314, 2023.
[22] Reddy, N. T. K., Patra, M. R., and Mishra, B. K., "Design and Implementation of an AI-Based Chatbot Framework with Retrieval-Augmented Generation and Integrated Recommender System," SSRN, 2025.
[23] Pokhrel, S., K. C., B., and Shah, P. B., "A Practical Application of Retrieval-Augmented Generation for Website-Based Chatbots: Combining Web Scraping, Vectorization, and Semantic Search," Journal of Trends in Computer Science and Smart Technology, vol. 6, no. 4, pp. 424–442, 2024.
[24] Huang, X., Lian, J., Lei, Y., Yao, J., Lian, D., and Xie, X., "Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations," arXiv preprint arXiv:2308.16505, 2024.
[25] A. Blount, A. Gulli, S. Saboo, M. Zimmermann, and V. Vuskovic,"Introduction to agents and agent architectures,"Google & Kaggle Whitepaper, Nov. 2025. [Online]. Available: https://www.kaggle.com/whitepaper-agents
[26] M. Styer, K. Patlolla, M. Mohan, and S. Diaz,"Agent tools and interoperability with MCP,"Google & Kaggle Whitepaper, Nov. 2025. [Online]. Available: https://www.kaggle.com/whitepaper-agent-tools
[27] K. Milam and A. Gulli, "Context engineering: Sessions and memory," Google & Kaggle Whitepaper, Nov. 2025. [Online]. Available: https://www.kaggle.com/whitepaper-context-engineering
[28] M. Subasioglu, T. Bulmus, and W. Bakkali, "Agent quality,"Google & Kaggle Whitepaper, Nov. 2025. [Online]. Available: https://www.kaggle.com/whitepaper-agent-quality
[29] LangChain Docs, "Docs by LangChain — The platform for agent engineering," LangChain, 2025. [Online]. Available: https://docs.langchain.com/ Accessed: Nov. 14, 2025.

IJARSCT
International Journal of Advanced Research in Science, Communication and Technology
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
ISSN: 2581-9429
Volume 6, Issue 11, June 2026
Impact Factor: 8.2

[30] LangGraph Docs, "LangGraph sets the foundation for how we can build and scale AI workloads," LangChain, 2025. [Online]. Available: https://www.langchain.com/langgraph Accessed: Nov. 14, 2025
[31] Firecrawl Docs, "Introduction | Firecrawl — Turn entire websites into LLM-ready markdown," Firecrawl, 2025. [Online]. Available: https://docs.firecrawl.dev/introduction Accessed: Nov. 14, 2025.
[32] n8n Integrations – Firecrawl, "Integrate Firecrawl with hundreds of other apps and services using n8n," n8n, 2025. [Online]. Available: https://n8n.io/integrations/firecrawl/ Accessed: Nov. 14, 2025.
[33] FAISS Documentation, "Welcome to FAISS — efficient similarity search and clustering of dense vectors," Facebook AI Research, 2025. [Online]. Available: https://faiss.ai/index.html. Accessed: Nov. 14, 2025.
[34] IBM Technology, "Prompt Engineering for LLMs, PDL, & LangChain in Action," YouTube video, Nov. 10, 2025. [Online]. Available: https://youtu.be/cgVppD6paYo. Accessed: Nov. 14, 2025.
[35] IBM Technology, "MCP vs API: Simplifying AI Agent Integration with External Data," YouTube video, May. 5, 2025. [Online]. Available: https://youtu.be/7j1t3UZA1TY. Accessed: Nov. 14, 2025.
[36] IBM Technology, "RAG vs Fine-Tuning vs Prompt Engineering: Optimizing AI Models," YouTube video, Apr. 14, 2025. [Online]. Available: https://youtu.be/zYGDpG-pTho. Accessed: Nov. 14, 2025.
[37] IBM Technology, "What are Word Embeddings?," YouTube video, Aug. 20, 2024. [Online]. Available: https://youtu.be/wgfSDrqYMJ4. Accessed: Nov. 14, 2025.
[38] IBM Technology, "What is MCP? Integrate AI Agents with Databases & APIs," YouTube video, Feb. 19, 2025. [Online]. Available: https://youtu.be/eur8dUO9mvE. Accessed: Nov. 14, 2025.
[39] IBM Technology, "What is a Vector Database? Powering Semantic Search & AI Applications," YouTube video, Mar. 14, 2025. [Online]. Available: https://youtu.be/gl1r1XV0SLw. Accessed: Nov. 14, 2025