International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

A RAG-Based Intelligent Product Comparison Framework for Multi-Platform E-Commerce Analytics

Prof. Sharon D, Associate Professor
Department of AI/DS, East West Institute of Technology Bengaluru, Karnataka, India

Chandru E Halageri
Department of AI/DS, East West Institute of Technology Bengaluru, Karnataka, India
chanduchandan0303@gmail.com

Prajwal B M
Department of AI/DS, East West Institute of Technology Bengaluru, Karnataka, India
prajwal2665@gmail.com

N Raghavendra
Department of AI/DS, East West Institute of Technology Bengaluru, Karnataka, India
raghavendra5485212@gmail.com

Tanuj Kumar
Department of AI/DS, East West Institute of Technology Bengaluru, Karnataka, India
tanujkumar12e@gmail.com

Abstract—Online shoppers frequently encounter the same product listed at different prices across multiple e-commerce platforms, making it difficult to identify the best purchasing option. Traditional price-comparison websites provide only tabular price information and lack intelligent decision support, while large language models may generate inaccurate or unverified recommendations. This paper presents PriceHunt, an AI-powered product comparison and recommendation system that integrates Retrieval-Augmented Generation (RAG) with a multi-retailer product database to deliver accurate, explainable, and data-driven shopping recommendations. The proposed system employs PostgreSQL-based full-text retrieval to identify relevant products and grounds AI-generated responses using retrieved catalogue information. An explainable scoring mechanism evaluates products based on factors such as price competitiveness, customer ratings, discount benefits, and historical pricing trends. The architecture consists of a presentation layer, retrieval layer, data layer, and generation layer, enabling efficient product comparison and natural language interaction. Experimental results demonstrate that the system successfully provides grounded recommendations, reduces the risk of hallucinated product information, and improves user decision-making through transparent explanations and visual comparison dashboards. The proposed framework offers a practical and scalable solution for intelligent e-commerce assistance and serves as a foundation for future enhancements such as live price tracking, hybrid retrieval, and personalized recommendations.

Keywords—Retrieval-Augmented Generation (RAG), Product Recommendation, Price Comparison, E-Commerce, Explainable AI, PostgreSQL, Large Language Models.

I. INTRODUCTION

The rapid growth of e-commerce platforms has transformed the way consumers purchase products. Online marketplaces such as Amazon, Flipkart, and brand-official stores provide customers with a wide variety of choices across multiple product categories. However, the availability of the same product on different platforms often results in significant variations in price, discounts, ratings, and promotional offers. As a result, consumers must spend considerable time comparing products across multiple websites before making a purchasing decision. This process

increases search effort and may lead to suboptimal buying choices.

Traditional price-comparison websites help users identify price differences among retailers but primarily present information in the form of static tables and lists. These systems provide limited decision support and do not explain which product offers the best value for money. On the other hand, recent advances in Large Language Models (LLMs) have enabled conversational shopping assistants capable of providing recommendations in natural language. Despite their capabilities, general-purpose LLMs may generate inaccurate or hallucinated information regarding product prices, specifications, and availability when not grounded in reliable data sources.

To address these challenges, Retrieval-Augmented Generation (RAG) has emerged as an effective approach for combining information retrieval with natural language generation. RAG systems retrieve relevant information from structured or unstructured knowledge sources and use the retrieved data as contextual evidence for generating responses. This approach improves factual accuracy and reduces hallucinations by ensuring that generated outputs are based on verified information.

This research presents **PriceHunt**, an AI-powered product comparison and recommendation platform that integrates Retrieval-Augmented Generation with a normalized multi-retailer product database. The system retrieves relevant products using PostgreSQL full-text search and provides grounded recommendations through a conversational AI assistant. Unlike conventional price-comparison systems, PriceHunt not only compares products across multiple retailers but also explains its recommendations using a transparent scoring mechanism. The scoring model evaluates products based on factors such as price competitiveness, customer ratings, discount benefits, and historical pricing trends, enabling users to understand the reasoning behind each recommendation.

The proposed architecture consists of four major layers: the presentation layer, application layer, data layer, and generation layer. This layered design ensures secure data access, efficient retrieval, scalable recommendation generation, and an interactive user experience. Furthermore, the system integrates visual comparison dashboards and deep-link navigation, allowing users to move seamlessly from conversational recommendations to detailed product analysis.

The primary contribution of this work is the development of an explainable AI-based shopping assistant that combines grounded retrieval, transparent ranking, and natural language interaction within a single platform. By reducing misinformation and improving decision support, PriceHunt aims to enhance the online shopping experience and provide a practical framework for intelligent e-commerce recommendation systems.

The primary contribution of this research is the development of an explainable and retrieval-grounded product recommendation framework that combines multi-retailer price comparison, conversational AI, and transparent decision support within a unified platform. By addressing the limitations of conventional price-comparison websites and standalone language models, PriceHunt aims to improve the overall online shopping experience and establish a foundation for future advancements in intelligent e-commerce recommendation systems.

II. LITERATURE REVIEW

The rapid advancement of Artificial Intelligence (AI), Information Retrieval (IR), and E-commerce technologies has led to the development of intelligent recommendation and decision-support systems. Several research studies have explored retrieval-augmented generation, recommendation systems, price comparison platforms, and explainable artificial intelligence. This section reviews the key contributions of existing works and identifies the research gap addressed by the proposed PriceHunt framework.

A. Retrieval-Augmented Generation (RAG)

Lewis et al. (2020) introduced the Retrieval-Augmented Generation (RAG) framework, which combines information retrieval with neural text generation. The model retrieves relevant information from an external knowledge source and uses it as contextual evidence during response generation. This approach improves factual accuracy and reduces the dependency on parametric knowledge stored within language models. The RAG framework demonstrated significant improvements in knowledge-intensive Natural Language Processing (NLP) tasks by providing grounded and context-aware responses. However, the original implementation focused primarily on open-domain question answering using document collections rather than structured commercial product databases.

B. Hallucination in Large Language Models

Ji et al. (2023) conducted a comprehensive survey on hallucination in Natural Language Generation systems. The study highlighted that large language models frequently generate information that appears plausible but lacks factual support. Such hallucinations can become particularly problematic in e-commerce applications, where inaccurate product prices, specifications, or availability information may mislead users. The survey emphasized the importance of grounding AI-generated responses using reliable data sources and retrieval mechanisms. This finding motivates the integration of retrieval-based approaches within recommendation systems to improve trustworthiness and accuracy.

C. Dense and Lexical Retrieval Techniques

Karpukhin et al. (2020) proposed Dense Passage Retrieval (DPR), which utilizes neural embeddings to retrieve semantically relevant information. Dense retrieval methods are highly effective in handling paraphrased queries and capturing semantic relationships between terms. In contrast, traditional lexical retrieval approaches such as BM25, introduced by Robertson and Zaragoza (2009), rely on keyword matching and term-frequency statistics. While dense retrieval generally provides superior semantic understanding, it requires significant computational resources and embedding infrastructure. For product catalogs containing structured attributes such as brand names and model numbers, lexical retrieval remains a cost-effective and efficient solution.

D. Comparison Shopping Agents

Comparison shopping systems have been studied for several decades. One of the earliest examples is ShopBot, proposed by Doorenbos, Etzioni, and Weld (1997), which automated the collection and comparison of product information from online retailers. Modern commercial price-comparison platforms similarly aggregate prices from multiple vendors and present them to users. Although these systems simplify product discovery, they primarily provide tabular comparisons and lack intelligent recommendationcapabilities. Users must still analyze specifications, reviews, and pricing information manually before making purchasing decisions.

E. Multi-Criteria Decision Analysis

Multi-Criteria Decision Analysis (MCDA) techniques are widely used to evaluate alternatives based on multiple factors. The Weighted Sum Model (WSM), also known as Simple Additive Weighting, is one of the most commonly adopted approaches. It combines multiple evaluation criteria into a single composite score using predefined weights. Such methods offer transparency and interpretability, making them suitable for recommendation systems where users require clear explanations of ranking decisions. The proposed PriceHunt framework adopts a weighted scoring mechanism to evaluate products using price, ratings, discount depth, and historical pricing information.

F. Full-Text Search in PostgreSQL

PostgreSQL provides built-in full-text search capabilities through the use of *tsvector*, *to_tsquery*, and Generalized Inverted Index (GIN) structures. These mechanisms enable efficient keyword-based retrieval directly within the database environment. Full-text search eliminates the need for separate search services while maintaining fast query execution and scalability. The use of PostgreSQL full-text search forms the retrieval backbone of the proposed PriceHunt system, allowing relevant products to be identified efficiently from a multi-retailer catalog.

G. Explainable Recommendation Systems

Zhang and Chen (2020) provided a comprehensive survey of explainable recommendation systems. Their study emphasized that users are more likely to trust recommendations when the reasoning behind decisions is explicitly communicated. Traditional recommendation models often operate as black boxes, making it difficult for users to understand why a particular item is recommended. Explainable recommendation systems address this limitation by exposing decision factors and score components. This concept strongly influences the design of PriceHunt, where recommendation scores are transparent and interpretable.

H. Security and Data Access Control

Modern web applications require secure access control mechanisms to protect sensitive data. Supabase and PostgreSQL Row-Level Security (RLS) policies provide database-level authorization by restricting access to specific records based on user permissions. Unlike client-side security mechanisms, RLS ensures that authorization rules are enforced directly within the data layer. This approach enhances system security and forms part of the overall architecture adopted by the proposed framework.

I. Research Gap

The reviewed literature demonstrates significant progress in retrieval-augmented generation, recommendation systems, and comparison-shopping platforms. However, existing studies primarily focus on either conversational AI, retrieval techniques, or price comparison independently. Traditional comparison-shopping systems provide price information without intelligent recommendations, while large language models often generate recommendations without grounding them in verified product data. Furthermore, few systems combine retrieval-augmented generation with an explainable ranking mechanism tailored for e-commerce applications.

Therefore, there exists a research gap in developing a unified framework that integrates multi-retailer price comparison, retrieval-grounded conversational AI, and transparent recommendation scoring. The proposed PriceHunt system addresses this gap by combining PostgreSQL-based retrieval, Retrieval-Augmented Generation, and an explainable multi-criteria recommendation model within a single intelligent shopping platform.

A. Full-Text Search in PostgreSQL

PostgreSQL provides built-in full-text search capabilities through the use of *tsvector*, *to_tsquery*, and Generalized Inverted Index (GIN) structures. These mechanisms enable efficient keyword-based retrieval directly within the database environment. Full-text search eliminates the need for separate search services while maintaining fast query execution and scalability. The use of PostgreSQL full-text search forms the retrieval backbone of the proposed PriceHunt system, allowing relevant products to be identified efficiently from a multi-retailer catalog.

III. PROBLEM STATEMENT

The rapid growth of e-commerce platforms has provided consumers with access to a wide range of products from multiple online retailers. However, identical products are often available at different prices, with varying discounts, ratings, specifications, and promotional offers across platforms. As a result, users must manually browse multiple websites, compare product details, and evaluate alternatives before making a purchase decision. This process is time-consuming, inefficient, and often leads to information overload.

Existing price-comparison websites primarily display product prices and specifications in tabular formats but provide limited decision support. Users are still required to analyze the available information and determine which product offers the best value for money. Additionally, recent AI-based conversational assistants can generate product recommendations but may produce inaccurate or hallucinated information when not grounded in verified product data.

Therefore, there is a need for an intelligent and reliable system that can retrieve product information from multiple retailers, compare products effectively, and generate accurate recommendations based on real catalog data. The system should also provide transparent explanations for its recommendations so that users can understand the reasoning behind each suggested product.

The objective of this research is to design and develop **PriceHunt**, an AI-powered product comparison and recommendation platform that integrates Retrieval-Augmented Generation (RAG), multi-retailer price comparison, and explainable recommendation techniques to support informed purchasing decisions and improve the overall online shopping experience.

IV. PROPOSED METHODOLOGY

The proposed PriceHunt system adopts a Retrieval-Augmented Generation (RAG) framework to provide accurate and explainable product recommendations by integrating information retrieval, artificial intelligence, and multi-retailer price comparison. The methodology is designed to ensure that all recommendations are generated using verified product data rather than relying solely on the internal knowledge of a language model. The overall process consists of data storage, product retrieval, context generation, recommendation generation, and explainable ranking.

This process begins with the dataset gathering and pre-processing stage for building the machine learning models. Phishing and genuine web sites' datasets are gathered and processed in order to detect useful features from URLs, and phishing emails' datasets are gathered for text processing. Duplicate and irrelevant entries are eliminated, and the gathered datasets are then divided into training and testing sets.

The system follows a four-layer architecture consisting of the Presentation Layer, Application Layer, Data Layer, and Generation Layer. The Presentation Layer provides a user interface for product search, comparison, and interaction with the AI assistant. The Application Layer processes user queries, retrieves relevant products, and manages communication between system components. The Data Layer stores product information, retailer prices, ratings, specifications, and historical pricing records in a PostgreSQL database.

A. System Architecture

The system consists of four layers:

1. Presentation Layer
2. Application Layer
3. Data Layer
4. Generation Layer

B. Data Collection and Storage

- Product information is collected from multiple retailers.
- Data includes product name, brand, category, price, rating, specifications, and historical price records.
- PostgreSQL database is used for data storage.
- Product specifications are stored using JSON format.

C. Product Retrieval Process

Step 1: Query Preprocessing

- Receive user query.
- Convert text to lowercase.
- Remove unnecessary symbols.
- Extract important keywords.

Step 2: Full-Text Search

- Search products using PostgreSQL full-text search.
- Use GIN-indexed search vectors for faster retrieval.

Step 3: Fallback Retrieval

- Perform keyword matching if no results are found.
- Search using product name, brand, and category.

Step 4: Candidate Selection

- Retrieve the most relevant products.
- Return top-rated products when no matches are available.



Fig. 1. System Workflow Diagram

Algorithm 1: Product Retrieval Algorithm

Input: User Query Q

Output: Candidate Product Set C

1. Receive user query Q.
2. Convert Q to lowercase.
3. Remove punctuation and special characters.
4. Extract keywords from Q.
5. Perform PostgreSQL Full-Text Search.
6. If matching products are found:
   - Return top relevant products.
7. Else:
   - Perform keyword-based search using product name, brand, and category.
8. If results are found:
   - Return matching products.
9. Else:
   - Return top-rated products from the catalog.
10. End.

Description:

The Product Retrieval Algorithm is responsible for identifying relevant products from the catalog based on the user's query. Initially, the query is preprocessed through tokenization, lowercasing, and removal of unnecessary symbols. The processed query is then matched against the PostgreSQL full-text search index to retrieve relevant products efficiently. If no suitable products are found, a fallback keyword search is performed using product attributes such as name, brand, and category. In situations where both retrieval methods fail, the system returns top-rated products to ensure that the recommendation engine always receives valid contextual information. This algorithm improves retrieval accuracy and reduces empty search responses.

Algorithm 2: RAG-Based Recommendation Algorithm

Input: User Query Q, Product Database P

Output: Product Recommendation R

1. Receive user query Q.
2. Retrieve relevant products from database P.
3. Extract product information.
4. Create contextual knowledge block.
5. Pass context to Gemini AI model.
6. Generate recommendation using retrieved data only.
7. Validate response against retrieved information.
8. Return grounded recommendation R.
9. End.

The RAG-Based Recommendation Algorithm generates intelligent product recommendations by combining information retrieval and natural language generation. After receiving the user query, the system retrieves relevant products from the database and constructs a contextual knowledge block containing product details, prices, ratings, and specifications. This information is supplied to the Gemini AI model, which

generates recommendations based only on the retrieved context. By grounding the AI model in verified product information, the algorithm minimizes hallucination and improves recommendation reliability.

Algorithm 3: Explainable Product Ranking Algorithm

Input: Product List P

Output: Ranked Product List

1. For each product Pi:
    - Calculate Price Score.
    - Calculate Rating Score.
    - Calculate Savings Score.
    - Calculate Historical Price Score.
2. Compute Final AI Score:

   AI Score =
   w1 × Price Score +
   w2 × Rating Score +
   w3 × Savings Score +
   w4 × Historical Score
3. Sort products in descending order of AI Score.
4. Assign ranks.
5. Display score breakdown for explainability.
6. Return ranked products.
7. End.

The Explainable Product Ranking Algorithm evaluates products using multiple decision criteria and ranks them according to their overall performance. The algorithm calculates individual scores based on price competitiveness, customer ratings, discount benefits, and historical pricing trends. These scores are combined using a weighted scoring mechanism to generate a final AI Score. Products are then ranked according to their scores, and the contribution of each component is displayed to users. This transparent ranking process enhances user trust and enables informed purchasing decisions.

Algorithm 4: Multi-Retailer Price Comparison Algorithm

Input: Product ID

Output: Best Retailer Recommendation

1. Retrieve product prices from all retailers.
2. Compare Amazon, Flipkart, and Official Store prices.
3. Identify minimum price.
4. Calculate discount percentage.
5. Determine best-value retailer.
6. Display comparison results.
7. Return retailer recommendation.
8. End.

The Multi-Retailer Price Comparison Algorithm compares product prices across different online retailers and identifies the most economical purchasing option. The algorithm retrieves prices from Amazon, Flipkart, and official brand stores, calculates price differences and discount percentages, and determines the retailer offering the lowest price. The comparison results are displayed through visual dashboards and tables, allowing users to evaluate alternatives easily. This algorithm helps users save time and money by providing a clear comparison of available purchasing options.

A. System Architecture

The system follows a four-layer architecture consisting of the Presentation Layer, Application Layer, Data Layer, and Generation Layer. The Presentation Layer provides a user interface for product search, comparison, and interaction with the AI assistant. The Application Layer processes user queries, retrieves relevant products, and manages communication between system components. The Data Layer stores product information, retailer prices, ratings, specifications, and historical pricing records in a PostgreSQL database. The Generation Layer utilizes a Gemini-based Large Language Model to generate recommendations using retrieved product information as contextual evidence. This layered architecture ensures scalability, security, and efficient information flow throughout the system.

B. Product Data Management

The proposed system maintains a structured product catalog containing information collected from multiple retailers. Each product record includes attributes such as product name, brand, category, description, customer ratings, retailer prices, historical pricing information, and technical specifications. The product specifications are stored using JSON-based structures to support different product categories while maintaining schema flexibility. The centralized database serves as the primary knowledge source for retrieval and recommendation operations.

C. Product Retrieval Mechanism

When a user submits a query, the system first preprocesses the input by converting it into a standardized format and extracting meaningful keywords. The retrieval module performs PostgreSQL full-text search using a GIN-indexed search vector to identify relevant products. If the full-text search fails to return suitable results, a fallback keyword-matching mechanism is executed using product names, brands, and categories. In cases where no matches are found, the system retrieves highly rated products to ensure that the recommendation engine always receives valid contextual information. This retrieval strategy improves system reliability and minimizes empty search responses.

D. Retrieval-Augmented Recommendation Generation

After retrieving candidate products, the system constructs a contextual knowledge block containing product descriptions, retailer prices, ratings, historical pricing information, and specifications. The generated context is supplied to the Large Language Model through a Retrieval-Augmented Generation framework. By grounding the model in retrieved product information, the system ensures that recommendations are based on verified data rather than model assumptions. This approach significantly reduces hallucination and improves the factual accuracy of generated responses.

E. Explainable Scoring and Ranking

To improve recommendation transparency, the system employs an explainable scoring model. Each product is evaluated using multiple criteria including price competitiveness, customer ratings, discount benefits, and historical price performance. Individual component scores are combined using predefined weights to generate a final AI Score. Unlike traditional black-box recommendation systems, the proposed model exposes score components to users, enabling them to understand the reasoning behind each recommendation and compare products more effectively.

F. Recommendation Presentation

The final recommendations are displayed through an interactive comparison dashboard. The dashboard presents product rankings, retailer price comparisons, historical price trends, and visual analytics. Additionally, structured navigation links are generated to allow users to directly access detailed comparison pages and product information. This integration of conversational AI and visual decision support enhances user experience and facilitates informed purchasing decisions.

V. RESULT AND DISCUSSION

The proposed PriceHunt system was successfully implemented and evaluated on a multi-retailer product catalog. The system integrates PostgreSQL-based retrieval, Retrieval-Augmented Generation (RAG), and an explainable recommendation engine to provide intelligent shopping assistance. Functional testing confirmed that the complete workflow, including product search, product comparison, recommendation generation, and dashboard visualization, operates as intended.

The retrieval module effectively identified relevant products using PostgreSQL full-text search. In situations where full-text search did not return suitable matches, the fallback retrievalmechanism successfully identified products through keyword-based matching. This multi-stage retrieval strategy ensured that users consistently received relevant recommendations and minimized empty search responses.

The RAG-based recommendation framework generated responses grounded in retrieved product information. Product prices, ratings, specifications, and recommendation details presented by the AI assistant were directly derived from the database records. This grounding mechanism significantly reduced the possibility of hallucinated information and improved recommendation reliability.



Fig. 2. Comparison Graph Of ML Models

. The experimental outcomes indicate that retrieval quality plays a critical role in the effectiveness of AI-powered recommendation systems. The combination of PostgreSQL full-text search and fallback retrieval strategies improved product discovery and ensured that the recommendation engine consistently received relevant contextual information. This finding highlights the importance of robust retrieval mechanisms in Retrieval-Augmented Generation systems.

Another significant observation is the impact of explainability on user trust. Traditional recommendation systems often provide rankings without explaining the reasoning behind their decisions. In contrast, PriceHunt exposes individual score components and ranking factors, enabling users to verify and understand recommendations. This transparency improves user confidence and supports informed purchasing decisions.

V. FUTURE SCOPE

Although the proposed PriceHunt framework successfully provides retrieval-grounded product recommendations and multi-retailer price comparison, several opportunities exist for future enhancement and expansion.

1. Hybrid Retrieval Techniques: Future versions of the system can integrate vector embeddings and semantic search methods with PostgreSQL full-text search to improve retrieval accuracy for complex user queries and natural language interactions.

2. Personalized Recommendation Engine: User preferences, browsing history, purchase behavior, and feedback can be incorporated to generate personalized product recommendations tailored to individual shopping requirements.

3. Price Prediction and Trend Analysis: Machine learning models can be integrated to analyze historical pricing patterns and predict future price fluctuations, helping users determine the optimal time to purchase products.

4. Advanced Explainable AI Techniques: Future research can explore more sophisticated explainability mechanisms that provide detailed reasoning, visual explanations, and factor-wise contribution analysis for recommendation decisions.

5. Multi-Modal Product Understanding: The system can be extended to analyze product images, videos, and customer reviews alongside textual product descriptions to improve recommendation quality and user experience.

6. Scalability and Large-Scale Evaluation: Future studies can evaluate the framework using larger product catalogs and a broader range of retailers to assess performance, scalability, and recommendation effectiveness in real-world environments.

7. Mobile and Voice-Based Shopping Assistant: The proposed framework can be deployed as a mobile application or integrated with voice assistants to provide hands-free product search, comparison, and recommendation services.

By incorporating these enhancements, PriceHunt can evolve into a comprehensive intelligent shopping assistant capable of delivering highly accurate, personalized, and real-time product recommendations while further improving user trust and decision-making efficiency.

IV. CONCLUSION

This paper presented **PriceHunt**, an AI-powered product comparison and recommendation framework that integrates Retrieval-Augmented Generation (RAG), multi-retailer price comparison, and explainable recommendation techniques. The proposed system addresses the limitations of traditional price-comparison platforms by combining intelligent product retrieval, grounded recommendation generation, and transparent decision support within a single platform.

The system utilizes PostgreSQL-based full-text search to retrieve relevant products from a structured multi-retailer product database and employs a Gemini-based Large Language Model to generate context-aware recommendations. An explainable scoring mechanism evaluates products based on factors such as price competitiveness, customer ratings, discount benefits, and historical pricing trends, enabling users to understand the reasoning behind each recommendation. The integration of conversational AI with interactive comparison dashboards further enhances the user experience and simplifies the decision-making process.

Experimental evaluation demonstrated that the proposed framework successfully provides accurate product comparisons, reliable recommendations, and transparent ranking explanations. By grounding AI responses in retrieved product information, the system reduces the risk of hallucinated outputs and improves recommendation trustworthiness. The retrieval mechanism, combined with fallback search strategies, ensures consistent product discovery and robust system performance.

Overall, PriceHunt demonstrates the effectiveness of combining Retrieval-Augmented Generation, explainable AI, and multi-retailer price comparison to support informed purchasing decisions. The proposed framework provides a scalable foundation for future intelligent e-commerce applications and highlights the potential of retrieval-grounded AI systems in enhancing online shopping experiences.

REFERENCES

[1] 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," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 9459–9474, 2020.

[2] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. Bang, A. Madotto, and P. Fung, "Survey of Hallucination in Natural Language Generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.

[3] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. Yih, "Dense Passage Retrieval for Open-Domain Question Answering," *Proceedings of EMNLP*, pp. 6769–6781, 2020.

[4] S. Robertson and H. Zaragoza, "The Probabilistic Relevance Framework: BM25 and Beyond," *Foundations and Trends in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[5] PostgreSQL Global Development Group, "PostgreSQL Documentation: Full Text Search," Available: https://www.postgresql.org/docs/current/textsearch.html

[6] R. Doorenbos, O. Etzioni, and D. Weld, "A Scalable Comparison-Shopping Agent for the World Wide Web," *Proceedings of the FirstInternational Conference on Autonomous Agents*, pp. 39–48, 1997.

[7] Y. Zhang and X. Chen, "Explainable Recommendation: A Survey and New Perspectives," *Foundations and Trends in Information Retrieval*, vol. 14, no. 1, pp. 1–101, 2020.

[8] Supabase Inc., "Supabase Documentation," Available: https://supabase.com/docs

[9] Google, "Gemini API Documentation," Available: https://ai.google.dev

[10] React Team, "React Documentation," Available: https://react.dev

[11] Tailwind Labs, "Tailwind CSS Documentation," Available: https://tailwindcss.com/docs

[12] Recharts Team, "Recharts Documentation," Available: https://recharts.org

[13] TypeScript Team, "TypeScript Documentation," Available: https://www.typescriptlang.org/docs

[14] Deno Authors, "Deno Documentation," Available: https://docs.deno.com