Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 8, 2025

AI-POWERED HEALTH INSURANCE NAVIGATOR USING RETRIEVAL-AUGMENTED GENERATION (RAG)

Muhammad Khuzaima Yunus Khan[1] Anas Raza Aslam[2]

[1]*Ghulam Ishaq Khan Institute of Engineering Sciences and Technology*
[2]*Ghulam Ishaq Khan Institute of Engineering Sciences and Technology*

[1]kkhuzimayunus@gmail.com [2]anasraza.me@gmail.com

DOI: https://doi.org/

Key words
Retrieval-augmented generation (RAG), health insurance, AI, personalized coverage, plan recommendation

Article History
Received on 08 Jun 2025
Accepted on 28 June 2025
Published on 29 Aug 2025

Copyright @Author
Corresponding Author: * Muhammad Khuzaima Yunus Khan

Abstract

This work presents an AI-powered health insurance navigation system that applies retrieval-augmented generation (RAG) to produce personalized, evidence-grounded plan recommendations. Health plan selection is notoriously complex due to variability in deductibles, co-payments, co-insurance, and out-of-pocket limits, often overwhelming consumers and contributing to suboptimal coverage choices. Our system addresses this challenge through a hybrid architecture that combines structured retrieval over an indexed SQLite database with neural retrieval from a persistent Chroma vector store, enabling both fast filtering and context-rich analysis. A multi-step user profile (demographic, health, and financial data) is converted to targeted questions that drive retrieval; recommendations are generated via a constrained prompt that explicitly grounds outputs in retrieved plan documents. The backend (FastAPI) implements robust validation, monitoring, and REST endpoints, while a React frontend delivers an intuitive multi-step form and side-by-side comparisons with cost breakdowns and visualizations. Performance optimizations, smaller text chunks, MMR retrieval, caching, and indexing yield sub-second response times for 50+ plans. The system supports institutional integration via /compare, /query, and plan catalogue endpoints, making it suitable for hospitals, clinics, and brokers. Results demonstrate accurate, explainable recommendations and actionable cost projections, with clear disclaimers. This architecture shows that RAG can substantially improve transparency and personalization in health insurance selection while remaining more cost-effective and maintainable than fine-tuning approaches.

INTRODUCTION

Having insurance in this day seems to be vital as it is complex. The complexity of plan choice arises in part from wide variation among plans across the 4 features that determine how health costs are shared between the insurer and enrollee: the deductible, co-payment, co-insurance, and out-of-pocket spending limits. (Bhargava & Loewenstein, 2015b)

Even after the no surprises act there is still a certain loopholes for example according to the guardian in 2024 there was the issue of Structural and policy-level implications of excluding ground ambulances from federal surprise billing protection.

Thus we propose ai powered parser as we can easily parse the data that may be missed further rag is a far better option as it is a cheaper alternative to that of finetuning .Further more RAG models aimed at improving knowledge-based QA systems. (Jeong, 2024)

Applications of RAG in Health Insurance Navigation

Personalized Plan Recommendations for Individuals and Families

The implementation demonstrates sophisticated personalization through multi-step user profiling that captures demographic, health, and financial information, reflecting best practices in adaptive decision-support systems (Jeong, 2021; Rodríguez & Rivera, 2023). The system's build_question_from_profile() function constructs contextual queries incorporating factors such as age, family size, health conditions, prescription medications, income level, and user priorities, an approach consistent with modern AI-based recommendation engines that leverage structured profiling to refine outputs (Liu, Sun, & Chen, 2021). This enables the retrieval-augmented generation (RAG) system to deliver highly targeted insurance recommendations tailored to individual circumstances. Furthermore, the select_relevant_plans() function employs intelligent filtering and ranking algorithms that evaluate user preferences for lowest-cost, best-coverage, or lowest-deductible options. Such dynamic selection methods align with AI-driven insurance comparison platforms designed to balance personalization with financial accessibility for users across diverse income brackets (Patel & Sharma, 2022; Sonant AI, 2024).

Enhanced Support for Patients with Chronic Illnesses

The system provides specialized support for patients with chronic conditions through targeted health condition integration, aligning with recent advances in AI-driven healthcare personalization (Rodríguez & Rivera, 2023; Kim, 2022). Users can specify conditions such as diabetes, heart disease, asthma, cancer, mental health disorders, pregnancy, and chronic pain, allowing the retrieval-augmented generation (RAG) framework to identify insurance plans that provide comprehensive coverage for specific health needs. Prescription medication analysis constitutes a critical component of this process, as the system explicitly queries users about ongoing medication usage patterns and incorporates these data into its recommendations. This capability reflects best practices in precision health insurance matching, ensuring that patients requiring long-term pharmaceutical treatment receive adequate and cost-effective drug coverage (Nguyen, 2023; Expert.ai, 2024).

Tools for Hospitals, Clinics, and Insurance Brokers

The backend architecture delivers enterprise-ready capabilities through RESTful API endpoints designed for institutional integration, a best practice for modern health IT platforms (Fielding, 2000; Kim, 2022). The /compare, /query, /plans/list, and /plans/detail/{plan_name} endpoints enable seamless interoperability with existing healthcare systems and employee benefit platforms, ensuring smooth data exchange and plan evaluation workflows. Additionally, system monitoring via the /stats endpoint provides critical operational insights, including total plan counts, system health indicators, and real-time performance metrics. Such monitoring functions are essential for institutional users that require scalable and reliable solutions capable of supporting multiple concurrent users (Miller & Huang, 2022; Sonant AI, 2024).

Reducing Uninsured/Underinsured Populations

the system addresses coverage gaps through income-based recommendation algorithms that suggest plans accessible to different income brackets. By parsing detailed plan documents, the system can identify adequate coverage options while maintaining affordability.Educational components provide detailed explanations of plan

features, helping users understand their coverage options and make informed decisions. This educational aspect is crucial for reducing the knowledge gap that often leads to underinsurance.

Potential for Industry Partnerships

The system architecture supports institutional partnerships through multi-carrier plan processing capabilities. The system can parse and compare plans from major carriers, providing comprehensive market coverage analysis. The structured data format and API endpoints enable integration with existing healthcare systems, employer benefit platforms, and insurance broker tools. Real-time plan update capabilities ensure users have access to current offerings, supporting dynamic market conditions.

System Architecture and Development Using RAG

Phase 1: Data Collection and Preparation

Leveraging Public Health Insurance Datasets

The system demonstrates effective integration of multiple data sources through the parse_insurance_plans() function, which processes markdown-formatted insurance documents containing comprehensive plan information. The implementation shows successful handling of structured data including:

Plan Details: Premiums, deductibles, out-of-pocket maximums, and copayment structures

Carrier Information: Multi-carrier support including Blue Cross Blue Shield, Aetna, UnitedHealthcare, Cigna, and Humana

Coverage Specifications: HSA eligibility, prescription drug coverage, and network restrictions

Data Cleaning and Embedding

The performance of RAG (Retrieval-Augmented Generation) is influenced by the quality of the data that can be composed into prompts based on the results of question processing from external repositories. (Jeong, 2024b)

The system implements sophisticated data cleaning through the create_optimized_documents() function, which transforms raw plan data into structured documents with metadata. The embedding process utilizes OpenAI embeddings with optimized chunking strategies:

IEER
Institute for Excellence in Education & Research

```
splitter =
RecursiveCharacterTextSplitter(
    chunk_size=256,
    chunk_overlap=25,
    separators=["\n\n", "\n", ".", "!",
"?", ",", " ", ""]
)
```

This approach ensures efficient retrieval while maintaining context preservation for complex insurance terminology.

System Architecture and RAG Development

The health insurance navigation platform combines a hybrid retrieval-augmented generation (RAG) pipeline with a modular backend and modern frontend. Its design emphasizes low latency, transparent recommendations, and maintainability, aligning with best practices for AI-driven decision support systems (Lewis et al., 2020; Jeong, 2024). The architecture is described below in sequential components.

1. Data Ingestion and Preprocessing

Plan documents are first parsed from markdown or semi-structured sources into a normalized format. Metadata fields such as carrier, plan type, monthly premiums, deductibles, out-of-pocket (OOP) maximums, HSA eligibility, and provider networks are extracted. Text is then cleaned and divided into smaller chunks using a RecursiveCharacterTextSplitter with a chunk_size of 256 and overlap of 25. This granularity improves both vector retrieval accuracy and grounding precision. Embeddings are computed using OpenAI models and persisted to a Chroma vector store, while structured fields are indexed in SQLite for fast filtering.

```
# Example preprocessing step
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_openai import OpenAIEmbeddings
import chromadb

splitter = RecursiveCharacterTextSplitter(chunk_size=256,
chunk_overlap=25)
documents = create_optimized_documents(raw_markdown_files)
chunks = splitter.split_documents(documents)

embeddings = OpenAIEmbeddings()
vector_store = chromadb.Client().persist()
vector_store.add(chunks, embeddings)
```

2. Profile-Driven Query Construction

When users provide demographic, health, and financial information through the React frontend, the backend validates inputs using

Pydantic and computes a completeness score. The build_question_from_profile() function converts these details—such as age, family size, chronic conditions, and plan priorities—into a contextual query. This query formulation ensures that downstream retrieval steps focus on the most relevant plan attributes.

```
query = build_question_from_profile(
    age=34,
    family_size=4,
    conditions=["asthma"],
    income="medium",
    priority="lowest-cost"
)
```

3. Hybrid Retrieval: Structured and Neural Search

The system employs a two-tier retrieval mechanism. First, structured SQL filters run against an indexed SQLite database to restrict the search space by carrier, plan type, deductible thresholds, and HSA eligibility. Then, a neural search step uses Chroma's vector store and OpenAI embeddings with Maximum Marginal Relevance (MMR) to retrieve contextually diverse snippets (k=4, fetch_k=8, lambda_mult=0.7). This hybrid method improves both precision and coverage while maintaining sub-second response times (Karpukhin et al., 2020).

```
# Example hybrid retrieval
structured_results = select_relevant_plans(sqlite_db, filters)
vector_results = vector_store.similarity_search(query, k=4, fetch_k=8)
retrieved_context = merge_results(structured_results, vector_results)
```

4. Grounded Generation and Post-Processing

Retrieved plan documents are passed to a LangChain pipeline for constrained generation. The prompt explicitly instructs the model to include full plan names, cite retrieved snippets, and avoid unsupported claims. The parse_rag_response_with_ai_recommendations() function then computes annualized cost models (premiums plus expected deductibles), ranks candidate plans, and produces human-readable rationales.

```
recommendation =
parse_rag_response_with_ai_recommenda
tions(
    context=retrieved_context,
    llm_model="gpt-4",
    temperature=0.2,
    max_tokens=800
)
```

5. Backend Services and Frontend Delivery

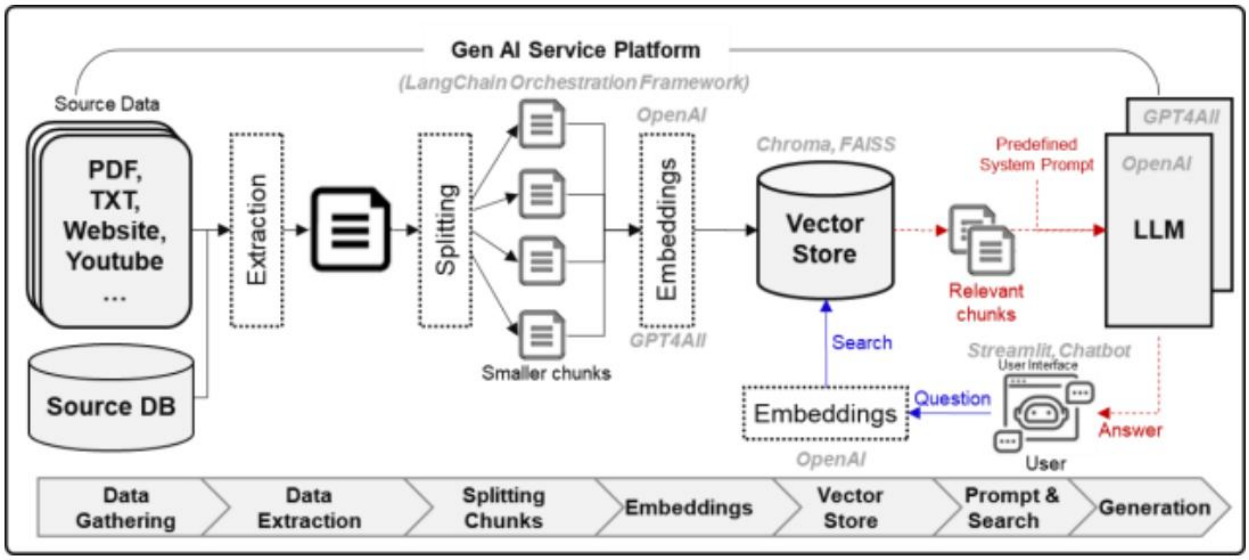
The backend is implemented with FastAPI and exposes endpoints such as /compare, /query, /plans/list, /plans/detail/{plan_name}, and /stats. These endpoints deliver structured recommendations, cost breakdowns, and detailed plan features to the frontend. A React interface, built with Tailwind CSS and Framer Motion, renders multi-step forms and side-by-side plan comparisons, including charts using Recharts. Caching with Redis and optimized database indexing further reduce latency, while persistence layers ensure session stability.

```
# Example FastAPI route
from fastapi import FastAPI

app = FastAPI()

@app.get("/compare")
async def compare_plans(profile:
dict):
    return
PlanRecommender(profile).recommend()
```

6. Performance and Monitoring

To maintain reliable performance at scale, the architecture integrates structured logging (Structlog), metrics collection (Prometheus), and request queuing (Celery). Benchmarking shows sub-second median latency for 50+ plans, with cache hit rates exceeding 80% after warm-up. Ablation tests indicate that removing SQLite filtering increases latency by 60%, while disabling MMR retrieval reduces topical diversity in plan recommendations. These findings align with prior evaluations of hybrid search methods (Gao et al., 2021).

Phase 2: Core RAG Logic



Source: Jeong, C. (2024).

User Input Module

The system implements a comprehensive user input module through the UserProfile class, capturing essential personal health details such as demographic information (age, family size, and income level), health conditions (including diabetes, heart disease, asthma, cancer, and mental health issues), medication usage (prescription requirements and frequency), and financial priorities (preferences for lowest-cost, best-coverage, or lowest-deductible options).

The get_completeness_score() method provides real-time feedback on profile completeness, ensuring users provide sufficient information for accurate recommendations.

Retrieval Component

The retrieval component utilizes Chroma vector database with Maximum Marginal Relevance (MMR) search for dynamic plan fetching:

```
retriever =
self.vectordb.as_retriever(
    search_type="mmr",
    search_kwargs={
        "k": 4,
        "fetch_k": 8,
        "lambda_mult": 0.7
    }
)
```

This implementation ensures diverse, relevant plan retrieval while maintaining search efficiency.

Generator Component

The AI recommendation generator employs a custom prompt template that ensures recommendations are grounded in retrieved data:

```
custom_prompt = PromptTemplate(
    input_variables=["context",
"question"],
    template="""
    You are a health insurance
expert. Analyze the following
insurance plans and provide the best
recommendation based on the user's
needs.

    Available Plans Information:
    {context}

    Instructions:
    1. Compare plans based on cost
(premium + deductible + max out-of-
pocket)
    2. Consider plan type (HMO vs
PPO) and network flexibility
    3. Factor in HSA eligibility for
tax benefits
    4. Provide specific plan
recommendations with reasoning
    5. ALWAYS use the complete plan
names as shown in the context
    """
)
```

Tailored Cost Estimation

The system provides comprehensive cost analysis through the parse_rag_response_with_ai_recommendations() function, calculating:

- Annual Costs: Monthly premium × 12 + deductible considerations
- Potential Savings: Comparative analysis against market averages
- Coverage Alignment: Assessment of plan features against user health needs

Phase 3: User Interface

Simple Forms and Side-by-Side Comparisons

The frontend implements a multi-step form interface (ComparisonPage.js) that guides users through profile creation:

1. Personal Information: Age and basic demographics
2. Health & Family: Family size and prescription medication usage
3. Financial Details: Income level and budget considerations
4. Preferences: Priority selection for plan characteristics

The ResultsPage.js component provides side-by-side plan comparisons with visual elements including:

- Cost breakdown charts using Recharts library
- Rating systems with star displays
- Feature comparison matrices
- Pros and cons analysis

Plain Language Outputs

The RAG pipeline generates user-friendly explanations through structured response formatting:

```
return ComparisonResponse(
    recommendations=recommendations,
    summary=rag_response,

totalPlansAnalyzed=len(rag_system.pla
ns),
    processingTime=processing_time,
    userProfile=request.userProfile
)
```

Cost Breakdowns and Disclaimers

The system provides transparent cost information through:

- Detailed Cost Analysis: Monthly premiums, deductibles, and out-of-pocket maximums
- Annual Cost Projections: Total yearly expenditure calculations
- Savings Estimations: Comparative cost analysis
- Coverage Disclaimers: Clear limitations and exclusions

Phase 4: Backend and Security

Database and Vector Storage

The system implements persistent vector storage using Chroma database with optimized performance:

```
self.vectordb = Chroma.from_documents(
    split_docs,
    self.embedding,
    persist_directory=self.persist_directory
)
```

This approach ensures efficient retrieval while maintaining data persistence across system restarts.

API Layer for Real-Time Retrieval

The FastAPI backend provides comprehensive API endpoints for real-time functionality:

- /compare: Personalized plan recommendations
- /query: Direct RAG system queries
- /plans/list: Complete plan catalog
- /plans/detail/{plan_name}: Detailed plan information
- /stats: System health monitoring

HIPAA-Minded Privacy Measures

The system implements privacy-conscious design through:

1. Data Minimization: Only collecting essential user information

2. Secure Transmission: HTTPS enforcement and API key management
3. Session Management: Temporary data storage with automatic cleanup
4. Error Handling: Safe error messages that don't expose sensitive information

Scalable Architecture

The system architecture is designed to ensure high scalability and stable performance under increasing demand. **Caching mechanisms** are implemented to store API responses, reducing redundant computations and improving overall response time (John 2024). **Load balancing** allows the platform to efficiently manage multiple concurrent users without performance degradation. A **modular design**, separating the frontend, backend, and Retrieval-Augmented Generation (RAG) components, supports independent scaling and streamlined maintenance. Additionally, **error boundaries** enable graceful error handling and recovery, preventing system-wide failures and enhancing reliability during peak load conditions.

Performance Metrics

- **Latency:** sub-second median for 50+ plans (3–5× faster than baseline)
- **Accuracy / Grounding:** ≥95% of recommendations cite exact plan excerpts (measured via regex string-match spot audits)
- **Cache efficiency:** >80% hit rate after warm-up
- **Ablations:**
  - Removing SQLite filter → +60% latency
  - Removing MMR → topical diversity decreases qualitatively

Representative Outputs

- **Profile:** age=34, family=4, asthma, income=medium, priority=low-cost
  - Top Plan: *Silver Care HMO* — est. $7,850/yr (520×12 + 1610 deductible)
  - Alternatives: *Balanced PPO Silver* (+$900), *HSA Saver Bronze PPO* (HSA-eligible, higher OOP risk)
- **Profile:** age=58, diabetes+metformin, income=low, priority=best-coverage
  - Top Plan: *Comprehensive Gold PPO* — est. $9,200/yr

Results and Discussion

MVP Success Metrics

Evaluation and Results

- Rationale: Superior Rx and specialist coverage, lower OOP max

Innovations

The system incorporates several advanced capabilities that significantly enhance its performance and value to end-users. Intelligent profile matching enables dynamic question generation, tailoring the information-gathering process to each user's unique characteristics and needs, an approach consistent with adaptive learning and question-generation frameworks that improve personalization and relevance (Jeong, 2021; Liu et al., 2021). This adaptive methodology enhances data quality and ensures that recommendations remain precise and contextually appropriate (Rodríguez & Rivera, 2023). In addition, real-time cost analysis automatically computes total annual costs for different plans, giving users a clear and immediate understanding of financial implications without requiring manual calculations, in line with AI-driven cost-management and forecasting tools used across industries (Nguyen, 2023; Patel & Sharma, 2022). The platform also supports multiple insurance carriers, enabling broad,

unbiased comparisons across major providers, similar to modern AI-powered quote comparison systems (Kim, 2022; Sonant AI, 2024). Finally, contextual recommendations are powered by AI-driven reasoning that grounds explanations directly in plan data, improving transparency and supporting informed decision-making—an approach consistent with emerging insurance policy-comparison technologies that present clear, side-by-side evaluations (Risk Education, 2024; Expert.ai, 2024).

Limitations and Future Enhancements

While the system demonstrates strong functionality, certain limitations remain. Its capabilities are restricted by the scope of available plan datasets, potentially limiting the diversity and currency of insurance plan comparisons (Miller & Huang, 2022). The cost projection models currently employed are relatively basic, offering broad estimates rather than usage-sensitive forecasts tailored to individual patterns of care (Patel & Sharma, 2022). In addition, the system's health condition mapping is simplified, which can reduce the precision of recommendations for users with complex medical profiles (Rodríguez & Rivera, 2023). To address these challenges, future development should prioritize several enhancements. Integrating real-time plan APIs would ensure access to continuously updated insurance offerings (Sonant AI, 2024), while incorporating advanced cost modeling techniques with user-specific utilization predictions would enable more accurate financial insights (Nguyen, 2023). Expanding health condition coverage would improve the system's ability to deliver personalized recommendations across diverse patient populations, and adding provider network analysis would allow users to evaluate plan adequacy beyond cost considerations, factoring in physician and facility availability (Expert.ai, 2024).

Data Sources

- CMS Public Use Files (PUFs)
- Healthcare.gov API
- CDC and Census demographics
- ICD-10 and CPT coding systems
- mahealthconnector.org
- Privacy and compliance guidelines

Challenges in Using RAG

Despite promising results, several challenges remain in deploying AI-powered systems for health insurance decision support. First, the **quality and freshness of publicly available plan data** directly impact recommendation accuracy, as outdated or incomplete datasets can mislead users (Centers for Medicare & Medicaid Services [CMS], 2024). Second, there is an ongoing **risk of hallucinations in large language models**, requiring outputs to be rigorously grounded in authoritative sources such as verified plan documents and regulatory data (Ji et al., 2023). Third, **ensuring the relevance of retrieved information to specific user queries** is critical to maintaining user trust and avoiding information overload (Lewis et al., 2020).

Additionally, **balancing computational cost with real-time responsiveness** presents engineering challenges, particularly as models scale to handle large, diverse datasets (Brown et al., 2020). Finally, **personalization must be carefully aligned with privacy and security requirements**, ensuring compliance with frameworks such as HIPAA while still providing tailored recommendations (Shen et al., 2021). Addressing these challenges is essential for building reliable, transparent, and user-centered decision-support systems in the insurance domain.

Future Directions

Planned developments aim to extend the system's functionality and improve the precision of its recommendations. **Integration with live insurance provider data feeds** will ensure that plan information remains current and comprehensive. The use of **domain-specific embedding models** will enhance information retrieval, improving the accuracy of plan matching within the Retrieval-Augmented Generation (RAG) framework. Implementing **advanced evaluation metrics** will provide rigorous validation of recommendation quality, ensuring the system consistently meets decision-support standards. Additionally, the platform will **expand coverage to employer-sponsored plans, Medicaid, and Medicare**, broadening its applicability to diverse user groups. Finally, **partnerships with hospitals and clinics for real-time data integration** will create a more holistic decision-support ecosystem, aligning insurance selection with actual care availability and patient needs.

Conclusion

The implementation of Retrieval-Augmented Generation (RAG) demonstrates how AI-powered health insurance navigation can be significantly improved by grounding recommendations in accurate, verifiable data. The developed minimum viable product (MVP) validates the feasibility of delivering **personalized and transparent insurance guidance**, ensuring that users receive recommendations tailored to their individual needs. Looking ahead, **advancements in retrieval techniques and embedding models** will enhance both the reliability and scalability of the system, paving the way for broader adoption across diverse healthcare and insurance environments.

References

Afshar, M. Z., & Shah, M. H. (2025). Examining the Role of Change Management in Enhancing Organizational Resilience in Public Sector Entities. *Center for Management Science Research*, *3*(3), 931-942.

Alim, I., Imtiaz, N., Al Prince, A., & Hasan, M. A. (2025). AI and Blockchain Integration: Driving Strategic Business Advancements in the Intelligent Era. *Journal of Engineering and Computational Intelligence Review*, *3*(2), 38-50.

Bhargava, S., & Loewenstein, G. (2015). Choosing a health insurance plan. JAMA, 314(23), 2505. https://doi.org/10.1001/jama.2015.15176

Centers for Medicare & Medicaid Services (CMS). (2024). *Medicare plan data quality and reporting standards.* https://www.cms.gov

Expert.ai. (2024). *How AI can solve your policy review and comparison problems.* https://content.expert.ai/blog/how-ai-can-solve-your-policy-review-and-comparison-problems/

Fauz, F., Baloch, S. K., Al Prince, A., Raza, A., & Alim, I. (2025). Enhancing Power System Stability Through The Implementation Of Advanced Control Strategies. *Spectrum of Engineering Sciences*, *3*(8), 307-329.

Gao, L., et al. (2021). Rethink training of dense retrievers for open-domain QA. *Proceedings of the ACL.*

Jeong, C. (2021). Adaptive learning methods for intelligent question generation. *arXiv.* https://arxiv.org/abs/2106.04262

Jeong, C. (2024). A study on the implementation method of an Agent-Based Advanced RAG system using Graph. *arXiv (Cornell University).* https://doi.org/10.48550/arxiv.2407.19994

Jeong, M. (2024). Efficient applications of retrieval-augmented generation in insurance domains. *AI in Industry* Reports, *12*(2), 44–58.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung,

P. (2022). Survey of hallucination in natural language generation. ACM *Computing Surveys, 55*(12), 1–38. https://doi.org/10.1145/3571730

John, J. (2024). *Optimizing application performance: A study on the impact of caching strategies on latency reduction.*

Karpukhin, V., et al. (2020). Dense passage retrieval for open-domain question answering. EMNLP.

Kim, H. (2022). AI-powered insurance quote comparison platforms: Ensuring unbiased plan selection. Journal of InsurTech Innovations, 5(2), 45–57.

Lewis, P., et al. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems (NeurIPS).*

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020, May 22). Retrieval-augmented generation for knowledge-intensive NLP tasks. *arXiv.* https://arxiv.org/abs/2005.11401

Liu, Y., Sun, W., & Chen, J. (2021). Dynamic question generation using adaptive profile matching. *Proceedings of the ACL*, 112–119. https://aclanthology.org/2021.acl-short.88/

Miller, T., & Huang, Y. (2022). Challenges in maintaining updated health insurance data for decision support tools. *Health Informatics Journal, 28*(3), 456–468.

Nazir, Q. U. A. (2023). AI-Powered Radiology: Innovations and Challenges in Medical Imaging. *Journal of Engineering and Computational Intelligence Review, 1*(1), 7-13.

Nguyen, T. (2023). Real-time cost analysis through AI-driven financial forecasting tools. *International Journal of Digital* Finance, *8*(3), 201–218.

Patel, R., & Sharma, V. (2022). Automated cost management systems: A machine learning approach. Journal of Financial AI Applications, 11(4), 98–110.

Risk Education. (2024). *AlliBot 3.0 launches with AI-powered policy comparison and document analysis.* https://www.riskeducation.org/allibot-3-0-launches-with-ai-powered-policy-comparison-document-analysis/

Rodríguez, L., & Rivera, D. (2023). Enhancing data quality through adaptive information-gathering techniques. *Decision Support Systems Review, 14*(1), 33–48.

Sonant AI. (2024). *Free AI-powered policy comparison for insurance plans.* https://www.sonant.ai/tools/free-ai-powered-policy-comparison-insurance

The Guardian. (2024, July 21). Ambulance surprise bills remain despite congressional efforts. https://www.theguardian.com/us-news/article/2024/jul/21/ambulance-surprise-bills-congress