INFORMATICS INSTITUTE OF TECHNOLOGY

INFORMATICS INSTITUTE OF TECHNOLOGY

In Collaboration with

UNIVERSITY OF WESTMINSTER

A Cost-Quantified, Hybrid-Retrieval RAG Agent for Pharmaceutical Q&A

A Project Proposal Requirement Specification by

Aron Fernando

Supervised by

Mr. Janith Prabhanuka

November 2025

ABSTRACT

Problem: Massive amounts of data, both structured and unstructured text, are produced by the pharmaceutical industry. The large number of documents and high operating costs of powerful Large Language Models (LLMs) make it difficult to extract precise and trackable insights from these data lakes. A major setback in pharmaceutical research and development is caused by the fact that current data retrieval systems are frequently expensive and lack the complex multimodal fusion capabilities needed to combine information spread across various data formats.

Methodology: This project develops a cost-quantified, hybrid-retrieval Retrieval-Augmented Generation (RAG) agent. The methodology is based on a two-tier model cascade that optimizes cost by routing queries to either a local, open-source LLM for simple tasks or a paid API for complex synthesis. The core innovation is a hybrid ingestion pipeline that parses documents, storing unstructured text in a Vector Database and structured tables in a SQL Database. The local LLM acts as an intelligent query classifier, routing requests to the appropriate data source (text, table, or both) to assemble the context. The economic benefit is measured using a novel metric, the Cost-Efficiency Ratio (CER).

Subject Descriptors: Information systems -> Information retrieval -> Retrieval models and ranking Computing methodologies -> Artificial intelligence -> Natural language processing -> Language models

Keywords: Retrieval-Augmented Generation (RAG), Hybrid Retrieval, Model Cascade, Cost-Efficiency, Multimodal Data

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Definitions

- **AI (Artificial Intelligence):** The field of computer science that is dedicated to creating systems that can perform tasks that generally require human intelligence.
- **API (Application Programming Interface):** A set of rules and protocols that allows different software applications to communicate with each other. In this project it refers to the paid service used for the Tier-2 LLM.
- **CER (Cost-Efficiency Ratio):** A novel metric proposed in this project to measure the operational cost of the RAG system, allowing for a benchmark against the baseline.
- **ETL (Extract, Transform, Load):** A data pipeline process. In this project it refers to **Extracting** text and tables from PDFs, **Transforming** them into clean formats, and **Loading** them into the Vector and SQL databases.
- **F1-Score:** A standard metric in machine learning used to measure a model's accuracy. It balances Precision and Recall and in this project it is used to evaluate both the Tier-1 Classifier and the accuracy of the final answer.
- **FDA (Food and Drug Administration):** The U.S. government agency responsible for regulating public health, including pharmaceutical drugs. Its DailyMed database is a key data source.
- **LLM (Large Language Model):** A type of AI model (e.g., GPT-4) trained on vast amounts of text data, capable of understanding and generating natural language.

- **MoSCoW:** A prioritization technique used in project management and software development, It stands for **M**ust Have, **S**hould Have, **C**ould Have, **W**on't Have.
- **NLP (Natural Language Processing):** A subfield of AI focused on the interaction between computers and human language which includes tasks like Text-to-SQL and semantic search.
- **PMC-OA (PubMed Central Open Access Subset):** A free digital archive of full-text biomedical and life sciences journal articles which is used as the primary data source for this project.
- **Q&A (Question & Answering):** The task in this project, of a user asking a natural language question and the RAG agent providing a fact-based, synthesized answer.
- **RAG (Retrieval-Augmented Generation):** The core AI architecture of the project. It involves **Retrieving** relevant information from a knowledge base (the databases) before **Generating** an answer, in order to improve factual accuracy.
- **SQL (Structured Query Language):** The standard programming language used to manage and query data in a relational (structured) database.
- **SRS (Software Requirement Specification):** A formal document (Chapter 4) that describes the functional and non-functional requirements of a software system.
- **UML (Unified Modeling Language):** A standardized modeling language used to visualize the design of a software system, such as in Use Case Diagrams or Context Diagrams.
- **Vector Database:** A specialized database designed to store and query high-dimensional vector embeddings, enabling fast and efficient "semantic" or "conceptual" search.
- **XML (Extensible Markup Language):** A markup language used to store and transport data in a structured format, common in the PMC-OA and DailyMed datasets.

Chapter 01: Introduction

1.1 Chapter Overview

This chapter lays the basement for our project, which aims to build a **cost-quantified, hybrid-retrieval RAG agent** for the pharmaceutical industry. We start by providing background on the core problem: the complexity of handling and integrating large amounts of complex **text and tabular data** in drug research. From there, we present a formal problem statement. The chapter then explains why this research is important, summarizes related work, and points out the specific gap in current technology that our project addresses. We conclude by outlining the project's contributions, key research questions, and the overall goals that will guide our work.

1.2 Problem Background

The pharmaceutical industry creates massive amounts of data in a number of formats (Wang and Krishnan, 2021). With healthcare data growing rapidly, there's a huge opportunity to speed up drug discovery and cut the high costs of research & development (Deloitte, 2022; Wouters, McKee and Luyten, 2020). However, this potential is often lost because the data is disconnected and hard to access.

Traditional keyword search fails when it comes to complex scientific documents (Hersh, 2017). Newer AI technologies like Retrieval-Augmented Generation (RAG) are better because they can understand natural language and provide fact-based answers (Lewis et al., 2020).

However, using this AI in the pharmaceutical world faces two key problems. First, it's very costly for large-scale use (Yue et al., 2023). Secondly, these systems struggle to connect information when it's split between text and tables in the same document, which is a challenge they weren't designed for (Cheng et al., 2022). This creates a need for a solution that is intelligent, affordable, and can handle mixed data formats.

1.3 Problem Definition

The core issue is the absence of a practical and affordable solution for pharmaceutical researchers to pull precise, fact-based information from their vast and mixed data sources. Today's RAG tools fail to solve the three unique challenges in this field:

1. **High Operational Costs:** The dependence on 100% API-based LLMs makes them financially unfeasible at scale.
2. **Factual Inaccuracy:** Current systems treat all data as unstructured text. They "chunk" and vectorize structured tables, destroying their tabular integrity. This leads to a retrieval of jumbled or incomplete context which causes the LLM to generate factually incorrect or incomplete answers when faced with a multimodal query.
3. **Lack of Efficiency Metrics:** There is no standard framework to measure the cost-benefit of these systems which makes it hard to justify their deployment.

This failure means pharmaceutical companies cannot leverage their data effectively and it leads to slower research and missed opportunities. Our project aims to solve this by developing a **hybrid-retrieval system** that processes text and tables using an optimal storage and retrieval strategy for each (a Vector DB for text, a SQL DB for tables) which ensures both cost-efficiency and factual accuracy.

1.3.1 Problem Statement

The pharmaceutical industry lacks a scalable and cost-effective data retrieval solution which is of accurately synthesizing insights from heterogeneous text and tabular data while providing trackable, evidence-backed answers.

1.4 Research Motivation

This research aims to showcase the value hidden in pharmaceutical data by handling the main barriers: cost and complexity. Our project will build a scalable and cost-effective AI agent that allows researchers to ask complex questions and receive quick, synthesized, evidence-based answers. If this knowledge discovery is sped up it has the potential to dramatically reduce drug development timelines, reduce R&D expenses, and subsequently accelerate the delivery of new medicines to patients.

1.5 Existing Work

Citation	Summary	Contribution	Limitation
(Chen et al., 2022)	Developed a generic RAG	Demonstrated the effectiveness of RAG	**Text-Only Architecture:** Lacks any mechanism for

	system for enterprise Q&A using a single, high-performance LLM.	for improving answer accuracy over standard LLMs in a corporate setting.	ingesting or retrieving data from **structured tables**, causing it to fail on multimodal queries. Does not address cost.
(Gupta and Li, 2023)	Created a multimodal model capable of parsing and querying financial reports containing both text and tables.	Advanced the field of table understanding and demonstrated techniques for fusing numerical data with surrounding text.	**Lacks Hybrid-Retrieval:** Relies on a single, complex model for fusion rather than a robust, hybrid system (SQL + Vector DB). **Lacks Cost-Quantification:** Not optimized for cost or built with a model cascade.
(Rodriguez et al., 2023)	Proposed an LLM routing system (cascade) to direct simple user queries to smaller, cheaper models, saving costs.	Proved the principle of LLM cascades as an effective strategy for reducing operational costs in AI systems.	**Unimodal Focus:** The cascade logic is for simple text queries, not for complex **hybrid-retrieval** (text/table) orchestration. Does not solve the multimodal data problem.
(Lee et al., 2020)	Introduced BioBERT, a language model pre-trained on a large-scale biomedical text corpus (e.g., PubMed abstracts).	Established the significant performance benefits of domain-specific pre-training for language models in the biomedical and pharmaceutical fields.	**Not a RAG System:** It is an encoder model, not a full, generative RAG agent. It is a potential *component* (for embedding text) but not a solution in itself.
(Asai et al., 2023)	Proposed Self-RAG, a framework where an LLM	Introduced an advanced RAG technique that improves factual	**Ignores Retrieval Source:** Focuses on *answer* quality, not the underlying *retrieval*

	learns to reflect on its own generated text to improve factual accuracy.	accuracy by making the retrieval and generation process more adaptive.	problem from hybrid sources. **Ignores Cost:** Adds computational overhead and does not address cost-efficiency.

Table 1: Existing Work

1.6 Research Gap

As summarized in the previous section the existing research reveals a clear and critical research gap. The literature is fragmented into three separate areas, with no single project unifying them:

1. **Text-Only RAG:** Systems like standard RAG (Chen et al., 2022) or Self-RAG (Asai et al., 2023) focus exclusively on unstructured text. They have no mechanism for accurately querying structured data which causes them to fail on multimodal questions.

2. **Cost-Only Optimization:** Systems like LLM cascades (Rodriguez et al., 2023) are a powerful cost-saving strategy but they have only been applied and evaluated on simple text-only tasks. They do not address the challenge of retrieving multimodal data.

3. **Multimodal Fusion (non-RAG):** Work on table-text fusion (Gupta and Li, 2023) often involves complex and singular models that are not designed as scalable retrieval systems and are mostly not optimized for cost.

Therefore, the specific unaddressed research gap is the **lack of a unified system that integrates a cost-saving model cascade with a hybrid-retrieval** for complex, multimodal document analysis.

1.7 Contribution to Body of Knowledge

This project will make a dual contribution to both the problem domain (pharmaceutical research) and the research domain (applied AI systems).

- Problem Domain Contribution (AI in Pharmaceutical Research): This project will deliver a strong and replicable architectural blueprint for a hybrid-retrieval RAG agent. This provides a practical path for pharmaceutical organizations to build cost-effective and accurate tools that can query both their unstructured text archives and their highly

structured tabular data via a single interface. This directly enhances research by unlocking disconnected data.

- Research Domain Contribution (Applied AI & LLM Systems): This work will objectively assess and report on the following key areas:

    1. The introduction and empirical validation of the Cost-Efficiency Ratio (CER) as a primary metric for benchmarking the financial performance of RAG systems.
    2. The design of a hybrid-retrieval orchestration layer showcases how a model cascade can be used not only for cost-saving but also for intelligently routing queries to different optimized data stores.
    3. A publishable benchmark comparing a "vector-only" RAG system against this novel "hybrid-retrieval" system, quantitatively proving the model's accuracy on text+tables questions.

1.8 Research Challenges

The successful execution of this project involves overcoming several key technical challenges.

Accurate Query Classification: The entire system depends on the Tier-1 (local) LLM's ability to accurately classify the user's intent as "text-only," "table-only," or "hybrid." An error here will lead to incomplete context and an incorrect answer.

- **Robust Table Ingestion (ETL):** PDF table extraction is extremely difficult. A significant challenge will be building a durable data pipeline that can parse structurally diverse tables from research papers and load them into a clean, query-able SQL database schema.

- **Hybrid Context Synthesis:** The Tier-2 (paid) LLM must be effectively prompted to synthesize a single answer from two distinct and differently formatted sources: a "semantic" text chunk from the Vector DB and a "factual" row of data from the SQL DB.

- **Balancing Cost vs. Accuracy:** A core challenge remains the central trade-off of the project: ensuring the cost-saving measures (using the cheaper local LLM) do not excessively degrade the final answer quality.

1.9 Research Questions

This research aims to answer the following questions:

1. How can a **two-tier model cascade** be successfully used to not only reduce API costs but also to act as an **intelligent query classifier** for a hybrid-retrieval RAG system?

2. How does a **hybrid-retrieval architecture** compare to a standard "vector-only" RAG baseline in terms of answer accuracy (F1-score) and reliability on multimodal (text/table) queries?

3. What is the **Cost-Efficiency Ratio (CER)** of this hybrid system and how can this metric be used to provide an accurate and quantitative analysis of the economic trade-offs between API cost their compute cost and answer quality?

1.10 Research Aim

The aim of this research is to **design, develop, and evaluate** a scalable and cost-quantified, **hybrid-retrieval** RAG agent that provides an efficient and accurate data lake solution for pharmaceutical research.

1.11 Research Objectives

To achieve the research aim, the following objectives have been defined:

Category	Objective ID	Research Objective Description	LOs Mapped	RQ Mapped
Literature Review	R01	To pilot an in-depth literature survey on RAG architectures, **hybrid-retrieval techniques**, and LLM cost-optimization strategies.	L01, L04	RQ1, RQ2

Requirement Elicitation	R02	To identify the key functional and non-functional requirements for a **hybrid (text/table) information retrieval** system in the pharmaceutical domain.	L03	RQ1, RQ2
System Design	R03	To design a **hybrid-retrieval architecture** that (a) ingests and stores text in a **Vector Database** and tables in a **SQL Database**, and (b) uses a **two-tier model cascade** as the query-classifying orchestrator.	L01	RQ1, RQ2
Implementation	R04	To implement a proof-of-concept prototype that integrates the **SQL DB** and **Vector DB** with the **model cascade**, enabling it to retrieve and synthesize hybrid (text/table) context.	L01, L02	RQ1, RQ2
Testing & Evaluation	R05	To quantitatively evaluate the prototype's performance based on answer quality (F1-score) and cost (CER) against a **"vector-only" RAG baseline**.	L02, L04	RQ2, RQ3
Documentation	R06	To document the project's methodology, findings, and contributions in a final thesis.	L04	RQ1, RQ2, RQ3

Table 2: Research Objectives

1.11 Chapter Summary

This chapter identified a critical problem in pharmaceutical research: data is both expensive to query and difficult to retrieve accurately due to its multimodal nature. Current RAG systems fail by being too expensive or by breaking tabular data.

To solve this we propose a hybrid-retrieval RAG agent using a Vector Database for text and a SQL Database for tables. This system is orchestrated by a two-tier model cascade for cost-efficiency and will be evaluated using a novel Cost-Efficiency Ratio (CER). The chapter has defined a clear aim, research questions, and objectives, setting the stage for the literature review.

Chapter 02: Literature Review

2.1 Chapter Overview

This chapter provides a comprehensive review of the existing literature relevant to the development of a cost-quantified, **hybrid-retrieval** RAG agent. It begins by defining the **problem domain** then into detailing the specific challenges of querying mixed text and tabular data in pharmaceutical research. The chapter then critically evaluates existing work in RAG systems, hybrid-retrieval architectures and LLM cost-optimization. Following this, it discusses dataset selection and the benchmarking metrics necessary to evaluate and validate the system. The chapter concludes with a summary of the key findings and the specific research gap that this project addresses.

2.2 Problem Domain

The landscape of pharmaceutical research is a data-driven endeavour, characterized by massive, complex, and heterogeneous datasets. This section provides an overview of this domain, the technological paradigms emerging to address its challenges, and the specific issues that motivate the present research.

2.2.1 Overview of Data Management in the Pharmaceutical Industry

A major challenge in the pharmaceutical world is that data is stored in a variety of formats. Information is spread across lab results in tables, academic articles, and clinical study reports. Previously pulling together and making sense of this disconnected data was a difficult and error-prone manual task performed by highly skilled specialists. This bottleneck makes research more expensive and slows down the pace of innovation.

2.2.2 The Emergence of Retrieval-Augmented Generation (RAG)

The Large Language Models (LLMs) have changed how we find information. They understand the meaning behind questions and can provide detailed answers. A newer approach called Retrieval-Augmented Generation (RAG) improves LLMs by first finding relevant facts from a privately held database and then using those facts to generate an answer. This makes the answers more accurate and traceable, which is essential for fields like pharmaceutical research.

However, using this technology in the drug industry faces three major challenges:

- **High Cost:** Using the best LLMs is expensive, making large-scale projects difficult to afford.
- **Complex Data:** Important information is often split between text and tables in the same document. Standard systems struggle to connect this mixed data, leading to wrong answers.
- **Scale:** The massive amount of data requires systems that are efficient and can grow without slowing down.

2.2.3 Challenges in Pharmaceutical Data Retrieval

Applying RAG to the pharmaceutical domain is hindered by three key challenges:

1. **High Operational Cost:** Relying on paid LLM APIs for all queries is financially unviable at scale.
2. **The "Lost-in-Vectorization" Problem:** Standard RAG systems treat all data as text. When they "chunk" and *vectorize* structured tables, they destroy their row-column integrity. This retrieves garbled, out-of-context fragments, leading to factually incorrect answers for multimodal queries.
3. **Inefficient Querying:** A vector-only database is not optimized for the precise, structured lookups that tables require.

2.2.4 Proposed Architectural Solution: The Hybrid-Retrieval RAG Agent

This project proposes a **Hybrid-Retrieval RAG Agent** to solve these challenges. The architecture intelligently segregates data:

1. **Unstructured Text** is stored in a **Vector Database** for semantic search.
2. **Structured Tables** are stored in a **SQL Database** to preserve their integrity for precise, factual queries.

This dual-storage system is orchestrated by a **two-tier model cascade**. This "brain" classifies incoming queries, routes them to the correct data source (Vector DB, SQL DB, or both), and intelligently allocates tasks to either a cheap local LLM or a powerful paid API, ensuring both accuracy and cost-efficiency.

2.3 Existing Work

This section provides a critical review of 20 key research papers and conference proceedings. The literature is analyzed to connect each source to its potential role in the project's pipeline, identify its limitations, and provide a critical review of its relevance to the proposed **Hybrid-Retrieval (Vector DB + SQL DB)** architecture.

Citation & Link	**Summary & Contribution**	**Methodology / Approach**	**Pipeline Role**	**Limitations**	**Critical review & Relevance**
(Lewis et al., 2020) Link	**Summary:** The seminal paper that introduced Retrieval-Augmented Generation (RAG). **Contribution:** Established the RAG framework, proving that grounding LLMs with external knowledge improves factual accuracy and reduces hallucinations.	A pre-trained dense passage retriever (DPR) finds relevant text passages, which are then fed as context to a pre-trained generative model (BART).	**Core Framework**	**Text-Only:** The original framework is designed exclusively for unstructured text and has no mechanism for handling structured data like tables.	**Critically Relevant.** This paper provides the foundational RAG concept that this project is built upon. Our project extends this by replacing its simple text retriever with a far more complex hybrid-retrieval orchestrator.
(Vaswani et al., 2017)	**Summary:** The "Attention Is All You Need" paper, which	An encoder-decoder model that relies	**Underpinning LLM Architecture**	**Not a RAG System:** This is a foundatio	**Foundational.** Understanding this paper is

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
Link	introduced the Transformer architecture. **Contribution:** The self-attention mechanism is the foundation of all modern LLMs, enabling the parallel scaling that made models like GPT possible.	entirely on self-attention and positional encodings to process sequences.		nal architecture, not a practical retrieval system. It defines the models, not the system that orchestrates them.	essential for understanding *why* LLMs work, which informs our model cascade (Tier-1 vs. Tier-2) and embedding model selections.
(Chen et al., 2022) Link	**Summary:** Developed a generic RAG system for enterprise Q&A using a single, high-performance LLM. **Contribution:** Demonstrated the effectiveness of RAG for improving answer accuracy in a	A standard RAG pipeline implementation using a state-of-the-art proprietary LLM, deployed to answer questions from internal company documents.	**Baseline System**	**Lacks Hybrid-Retrieval** and **Lacks Cost-Quantification.** Relies on a single, expensive API and fails on multimodal queries.	**Highly Relevant.** This paper perfectly defines the "naive" baseline system that our project will be benchmarked against. We aim to beat this on both **cost**

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	corporate (text-only) setting.				**(CER)** and **accuracy (F1-score)**.
(Rodriguez et al., 2023) Link	**Summary:** Proposed an LLM routing system (cascade) to direct simple user queries to smaller, cheaper models. **Contribution:** Proved the principle of LLM cascades as an effective strategy for reducing operational costs.	A query classification model that routes queries between a small, cheap model and a large, expensive model in a conversational AI setting.	**Model Cascade (Tier 1)**	**Unimodal Focus:** The cascade logic is for simple text queries, not for complex **hybrid-retrieval** (text/table) orchestration.	**Critically Relevant.** This provides the academic justification for our entire cost-saving architecture. Our project extends this by making the cascade "smarter"—it's not just a cost-saver, it's the **query orchestrator**.
(Yue et al., 2023)	**Summary:** Proposed "LLM-Cascade," which uses the	A cascade decision-maker that runs	**Model Cascade (Tier 1)**	**Text-Only Reasoning:** The methodol	**Highly Relevant.** This paper gives us a

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
Link	"answer consistency" of a weaker LLM to decide whether to escalate. **Contribution:** Provided a robust methodology for query classification in a cascade, demonstrating a 60% cost reduction.	the cheaper model multiple times; if the answers are inconsistent, it escalates the query.		ogy is designed for text-based reasoning (e.g., math problems), not for multimodal RAG.	specific, testable method for our Tier-1 query classifier. We can adapt this "consistency" logic to decide if our local LLM is confident enough to answer.
(Zhang et al., 2024) Link	**Summary:** Introduces "early abstention" in LLM cascades, which is crucial for risk-sensitive domains like medicine. **Contribution:** Refines the cascade model by adding a critical "abstain"	A cascade where the cheaper, earlier models can decide to "abstain" if a query is too difficult or risky, preventing an incorrect answer.	**Model Cascade (Refinement)**	**Theoretical Framework:** Focuses on the cascade logic itself, not the underlying hybrid-retrieval architecture.	**Moderately Relevant.** This paper provides a key insight for our project: our Tier-1 classifier shouldn't just route, it should also have the option to "abstain"

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	(reject query) feature, reducing errors and costs.				if a query is unanswerable, which improves system safety.
(Fanconi et al., 2024) Link	**Summary:** Describes a cascaded LLM framework that includes "human-in-the-loop" as the final tier. **Contribution:** Provides a complete, three-tier (cheap/expensive/human) architecture for cost-optimization.	A two-stage deferral and abstention policy that routes queries to a base model, a large model, or a human expert.	**Model Cascade (Refinement)**	**Ignores Retrieval:** This work is about the generation and decision-making part, not the retrieval part, which is the core of our RAG project.	**Moderately Relevant.** This reinforces the validity of the cascade approach. Our project focuses on the *retrieval* aspect, but this paper shows the cascade is a state-of-the-art (SOTA) method for cost-efficiency.

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
(Herzig et al., 2020) Link	**Summary:** Introduced **TAPAS**, a BERT-based model pre-trained for question answering over tables. **Contribution:** Proved that models can learn the row-column structure of tables to answer natural language questions.	A BERT model with special embeddings for table structure (rows, columns) that is fine-tuned for Q&A on single tables.	**Table Ingestion (Problem Definition)**	**Not a RAG System:** TAPAS is a Q&A model, not a retrieval agent. It works on a *single* table and cannot be integrated into a scalable RAG pipeline that must first *find* the right table.	**Highly Relevant.** This paper is key for our **Problem Definition**. It proves that tables are a *special* data type requiring a *special* model, which justifies our decision to *not* vectorize them.
(Manjee, 2024) Link	**Summary:** Discusses the "Lost-in-Vectorization" problem, proposing a method to parse PDF tables into text summaries using an LLM.	A practical guide that uses pdfplumber to extract tables, then an LLM to "summarize" the table into a text	**Table Ingestion (Problem Definition)**	**Lossy Conversion:** Converting a table to a text summary is lossy. It fails for precise queries (e.g., "value in row 5, col	**Highly Relevant.** This blog post perfectly defines the "naive" baseline we are competing against. Our SQL-based

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	Contribution: Identifies the exact problem this project addresses: vectorizing tables destroys their integrity.	paragraph, which is then vectorized.		3") and is not a true hybrid solution.	approach is demonstrably superior to this lossy "summarization" method.
(Sahoo et al., 2024) Link	**Summary:** Analyzes the trade-off in RAG Text-to-SQL systems, noting that larger, detailed schema documents improve accuracy but also increase hallucination. **Contribution:** Provides a benchmark for the balance between providing too much or too little context to a Text-to-SQL model.	An experimental paper that tests Text-to-SQL performance based on the size and content of the retrieved schema context.	**SQL Retrieval (Prompting)**	**SQL-Only:** This analysis is confined to the SQL part of the problem and does not integrate semantic search from a Vector DB.	**Moderately Relevant.** This paper informs our system design. It suggests that when we build our Tier-2 Text-to-SQL prompt, we must be careful to only include the *relevant* schema, not the entire DB schema.

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
(Gupta and Li, 2023) Link	**Summary:** Created a multimodal model for parsing and querying financial reports (text and tables). **Contribution:** Advanced the field of table-text fusion in a specific domain.	A complex, single-model architecture designed to read and reason over both text and tables in financial documents.	**Problem Definition**	**Lacks Hybrid-Retrieval:** Relies on a single, complex model, not a robust, two-database (SQL + Vector) system. **Lacks Cost-Quantification.**	**Highly Relevant.** This proves that the text-table fusion problem is a significant challenge in other domains (like finance). This strengthens our project's motivation.
(Liu et al., 2023) Link	**Summary:** A comprehensive survey of Text-to-SQL techniques, covering the entire lifecycle from modeling to evaluation. **Contribution:** Provides a complete academic	A survey paper, not an implementation. It categorizes and compares dozens of Text-to-SQL models and benchmarks.	**SQL Retrieval (Lit Review)**	**Theoretical:** As a survey, it describes the field but does not present a single, deployable system, especially not one integrated with a	**Highly Relevant.** This paper is the primary literature review for the **SQL-retrieval** half of our project. It gives us the academic context

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	overview of the Text-to-SQL field, its challenges, and its state-of-the-art.			Vector DB.	and terminology to discuss our solution.
(Gao et al., 2024) Link	**Summary:** Proposes a RAG-based Text-to-SQL system that uses BERT for vectorized retrieval and GPT-4 for query generation. **Contribution:** Demonstrates using RAG to *improve* a Text-to-SQL system (e.g., to find the right schema).	A RAG system is used to find relevant schema information or examples, which are then fed to an LLM to help it write a better SQL query.	**SQL Retrieval (Architecture)**	**SQL-Focused:** This architecture is still 100% focused on the SQL problem. It does not solve "hybrid" queries that also need unstructured text from a *separate* document.	**Moderately Relevant.** This shows an advanced technique for the Text-to-SQL component. We will be using a simpler version, but this paper validates the RAG-for-SQL approach.
(Sadekar, 2024) Link	**Summary:** Provides a practical guide for building a Text-to-SQL application using	A tutorial showing how to use LangChain to build an agent that can inspect a	**SQL Retrieval (Implementation)**	**SQL-Only:** This is a guide for a *Text-to-SQL* agent, not a *Hybrid* agent. It	**Critically Relevant.** This is our implementation guide. It provides the exact

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	LangChain and RAG. **Contribution:** Offers a robust, practical blueprint for the **SQL-retrieval** half of our project, including prompt engineering and validation.	SQL schema and use an LLM to write and execute queries.		cannot answer questions from unstructured text.	code and architectural pattern for the SQL-retrieval half of our agent, which we will then *combine* with a vector-retrieval half.
(Lee et al., 2020) Link	**Summary:** Introduced **BioBERT**, a language model pre-trained on biomedical text (e.g., PubMed). **Contribution:** Established the significant performance benefits of domain-specific pre-training.	Continued pre-training of the general BERT model on biomedical corpora.	**Text Embedding (Component)**	**Not a RAG System:** It's an encoder model, a potential *component* for our text embedding, but not a full solution.	**Highly Relevant.** This is a **strong candidate** for our vector embedding model. Using BioBERT (or PubMedBERT) instead of a generic model will improve our text retrieval accuracy.

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
(Gu et al., 2021) Link	**Summary:** Developed **PubMedBERT**, pre-trained from scratch on PubMed abstracts. **Contribution:** Proved that pre-training from scratch on a domain-specific vocabulary is superior to continued pre-training (like BioBERT).	Pre-training a BERT model from the ground up using a vocabulary and corpus composed entirely of biomedical and clinical texts.	**Text Embedding (Component)**	**Not a RAG System:** This, like BioBERT, is a candidate for the embedding model, not a complete RAG architecture.	**Highly Relevant.** This is likely the **best candidate** for our embedding model. Its superior performance on biomedical text will directly improve our text-retrieval quality.
(BaranziniLab, 2024) Link	**Summary:** Describes a **Knowledge Graph-RAG** framework for biomedicine, showing it boosts Llama-2 performance by 71% on MCQs. **Contribution:** Proves that grounding in *structured,*	Uses a Knowledge Graph (a more complex version of our SQL DB) to retrieve facts, which are fed to an LLM.	**Project Justification**	**Graph-Based, Not Hybrid:** A Knowledge Graph is a different, more complex architecture than our proposed SQL+Vector	**Highly Relevant.** This paper provides the core *justification* for our project. It proves that for biomedical data, **structured retrieval (like our**

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	domain-specific data dramatically improves accuracy.			system. It is also not cost-quantified.	**SQL DB) beats text-only retrieval**.
(Asai et al., 2023) Link	**Summary:** Proposed **Self-RAG**, a framework where an LLM reflects on its own output to improve factual accuracy. **Contribution:** Introduced an advanced technique for improving answer faithfulness, a key metric for our project.	An LLM that adaptively retrieves passages and uses "reflection tokens" to critique its own generated answer for factuality.	**Evaluation (Factuality)**	**Ignores Retrieval Source:** Focuses on *answer* quality, not the underlying *retrieval* problem from hybrid (SQL/Vector) sources.	**Moderately Relevant.** This informs our **Evaluation** (Chapter 4). We can use the principles of Self-RAG to help evaluate the faithfulness of our agent's answers.
(Es et al., 2023) Link	**Summary:** Introduced **RAGAs**, a framework for reference-free evaluation of RAG pipelines (faithfulness, answer	A suite of metrics that use an LLM-as-a-judge to score a RAG pipeline on	**Evaluation (Metrics)**	**Evaluation, Not Architecture:** RAGAs is the *measurement tool* for our project,	**Critically Relevant.** This is the **primary tool** we will use for our evaluation chapter. We will

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
	relevance, etc.). **Contribution:** Provides the key *evaluation tools* needed for this project to objectively measure RAG performance.	dimensions like "Faithfulness" and "Context Relevance" without needing a ground-truth dataset.		not the project itself.	use RAGAs to benchmark our system's accuracy against the baseline.
(Hao et al., 2024) Link	**Summary:** Describes a "SQL-RAG" system that combines LangChain and LlamaIndex to query both SQL databases and text. **Contribution:** Demonstrates a foundational architecture for building a hybrid-retrieval agent.	A practical implementation that uses an LLM to route a query to either a Text-to-SQL tool or a vector index.	**Core RAG Framework**	**Lacks Domain-Specificity & Cost-Quantification:** This is a generic tool, not a benchmarked, cost-quantified system for the complex pharmaceutical domain.	**Highly Relevant.** This paper is a direct precedent for our proposed architecture. Our project builds on this by (1) applying it to the pharma domain, and (2) adding the critical **cost-quantification**

Citation & Link	Summary & Contribution	Methodology / Approach	Pipeline Role	Limitations	Critical review & Relevance
					cascade on top.
(Day, 2024) Link	**Summary:** Describes modern **Hybrid Search** as a combination of keyword (sparse) and vector (dense) search. **Contribution:** Provides the core principle of hybrid-retrieval: using the best tool for the job (precision vs. semantic meaning).	An article explaining how SOTA RAG combines keyword search (for exact matches) and vector search (for semantic matches) in a single pipeline.	**Core RAG Framework (Problem Def.)**	**Not SQL-Hybrid:** This "hybrid search" refers to (Keyword + Vector), not our project's **(SQL + Vector)**, which solves a different, more complex multimodal problem.	**Moderately Relevant.** This helps us refine our terminology. We must be clear that our "Hybrid-Retrieval" is **(SQL + Vector)**, which is more advanced than the common (Keyword + Vector) hybrid search.

Table 3: Chapter 2 Existing Work

2.4 Dataset Selection

The selection of an appropriate dataset is critical for the development and evaluation of the hybrid-retrieval RAG agent. The ideal dataset must be representative of the target domain and, most importantly, contain the two distinct data types (unstructured text and structured tables) required to populate the system's dual databases.

2.4.1 Primary Dataset: PubMed Central Open Access Subset (PMC-OA)

📝 PMC-OA Dataset Summary (Shortened)

The **PubMed Central Open Access (PMC-OA) Subset** is a free and public archive of millions of full-text which are peer-reviewed biomedical and life sciences articles are the core corpus for this project.

Description

The dataset includes articles in **XML** (valuable for preserving document structure, including tables and sections) and plain text formats.

Suitability for Hybrid-Retrieval:

- **Vector Database: Unstructured text** (abstracts, introductions) provides a large, domain-specific corpus for semantic search and text-based queries.
- **SQL Database: Highly structured tables** (experimental results, patient data, statistics) will be parsed and loaded via the Table Ingestion Pipeline into the SQL Database for precise and factual querying.

Intended Use

The dataset will be used to build, validate, and benchmark the ingestion pipeline. It facilitates testing **"hybrid queries"** that require information from both the text and the tables of a single article.

2.4.2 Supplementary Dataset: FDA Drug Labels on DailyMed

To ensure the system's robustness and generalizability, the **U.S. Food and Drug Administration (FDA) drug labels available on the DailyMed database** will be used as a supplementary dataset.

- **Description:** DailyMed provides high-quality, structured product labeling (SPL) for thousands of drugs. The SPL format is a well-defined XML that clearly delineates all text sections and tables.
- **Suitability for Hybrid-Retrieval:**

- **For the Vector Database:** The text-heavy sections (e.g., "Indications and Usage," "Warnings and Precautions," "Clinical Pharmacology") will be extracted and loaded into the Vector Database.
- **For the SQL Database:** The highly regular and critical tables (e.g., "Adverse Reactions by Frequency," "Dosage and Administration") will be parsed and loaded into the SQL Database.

- **Intended Use:** This dataset will be used to test the generalizability of the ingestion pipeline. It will prove that the system can handle a different document structure (regulatory vs. academic) and is therefore a robust solution.

2.5 Benchmarking and Evaluation

To validate the Hybrid-Retrieval RAG Agent, this section defines the evaluation metrics, compares the proposed system to existing work, and outlines the benchmarking strategy.

2.5.1 Evaluation Metrics

System performance will be measured across three key pillars:

1. **Retrieval & Classification:** We will measure the **Query Classification Accuracy** (F1-score) of the Tier-1 LLM, **Text Retrieval** (Recall@k) from the Vector DB, and **Table Retrieval** (Execution Accuracy) from the SQL DB.
2. **Generation Quality:** Using the **RAGAs** framework, we will assess the final answer's **Accuracy** (F1-score against a ground-truth answer), **Faithfulness** (is it grounded in the context?), and **Relevance**.
3. **Economic & Performance:** We will measure the primary **Cost-Efficiency Ratio (CER)** by comparing API costs, as well as the end-to-end **Latency**.

2.5.2 Comparative Analysis of Existing Models

The table below highlights the research gap, showing that this project is the first to combine all necessary components.

Model / Framework	Hybrid-Retrieval (SQL + Vector)	Cost-Optimized (Cascade)	Domain-Specific (Pharma)	Generative
Standard RAG (Lewis et al., 2020)	No (Vector-Only)	No	No	Yes
Table Q&A (Herzig et al., 2020)	No (SQL-Only)	No	No	No
LLM Cascades (Yue et al., 2023)	No	Yes	No	Yes
Text-to-SQL (Liu et al., 2023)	No (SQL-Only)	No	No	Yes
Hybrid Tools (Hao et al., 2024)	Yes	No	No	Yes
This Project's Proposed Agent	**Yes**	**Yes**	**Yes**	**Yes**

Table 4: Comparative Analysis of Existing Models

2.5.3 Benchmarking Strategy

The evaluation will be a controlled experiment. We will compare a **"Vector-Only RAG" (Baseline)** against our **"Hybrid-Retrieval RAG Agent" (Proposed)**. A ground-truth test set of ~100 questions (labeled as "text-only," "table-only," and "hybrid") will be used. We will run this test set through both systems, logging all metrics from 2.5.1 to provide a quantitative comparison of accuracy, cost, and latency.

2.6 Chapter Summary

This chapter has provided a comprehensive literature review, establishing the context for the **Hybrid-Retrieval RAG Agent**. The review first defined the critical **"Lost-in-Vectorization" problem** (2.2.3), where standard RAG systems fail by destroying the integrity of structured tables.

The survey of existing work (2.3) confirmed that solutions are siloed: standard RAG is text-only (Lewis et al., 2020), cost-cascades are unimodal (Yue et al., 2023), and Text-to-SQL frameworks (Liu et al., 2023) are not integrated with semantic text retrieval or cost-quantification.

This analysis confirms a clear **research gap**: no existing work combines a **hybrid (SQL + Vector) retrieval** architecture with a **cost-quantified model cascade** in the pharmaceutical domain. Finally, the chapter identified suitable datasets (PMC-OA) and a robust benchmarking framework (2.5) using the **CER** and **RAGAs** metrics to validate this novel approach.

Chapter 03: Methodology

3.1 Chapter Overview

This chapter explains our plan for the project. It covers our overall research approach including the technical plan for building and testing the system and the project management details which includes the timeline, resources, and a plan for handling risks.

3.2 Research Methodology (Saunders' Research Onion in Table Form)

The research methodology for this project is structured using Saunders' Research Onion model to ensure a logical and coherent research design, from the philosophical underpinnings to the practical data collection techniques. The chosen layers of the onion are detailed in the table below.

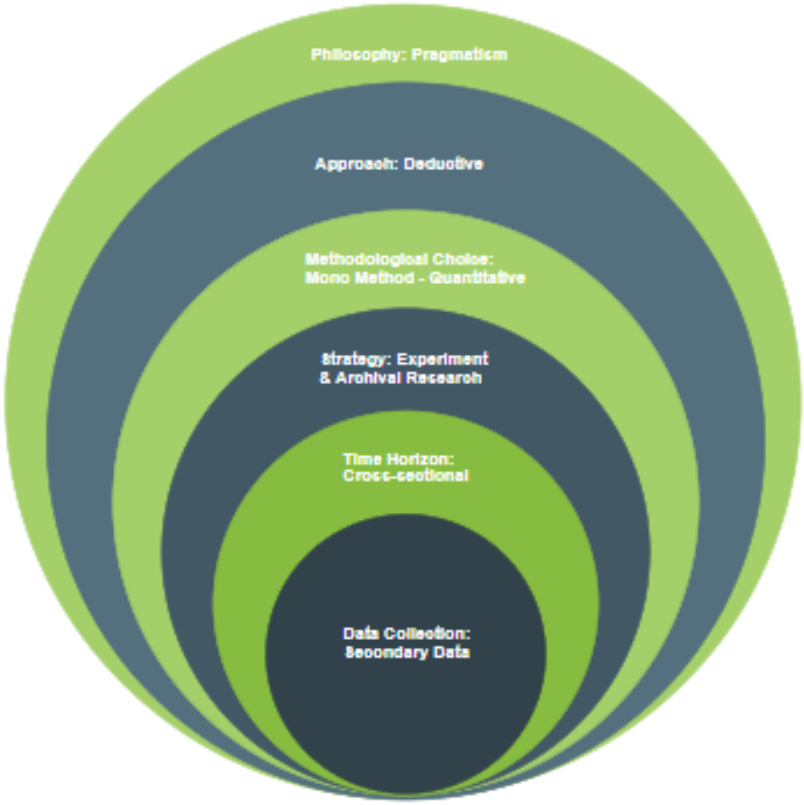

Figure 1:Saunder's Research Onion

Layer	Choice	Justification (Shortened)
Philosophy	**Pragmatism**	Focuses on solving a **practical problem** by building a functional system; values actionable knowledge and linking theory to practice.
Approach	**Deductive**	Starts with a **testable hypothesis** derived from theory. An experiment will then be

Layer	Choice	Justification (Shortened)
		conducted to compare the proposed system against a baseline.
Methodological Choice	**Mono Method (Quantitative)**	Uses a quantitative approach focusing solely on **numerical data** (costs, latency, F1-scores) from a controlled experiment for objective evaluation.
Strategy	**Experiment & Archival Research**	Primary strategy is a **controlled Experiment** to compare the RAG agent against a baseline, supplemented by **Archival Research** using existing datasets (e.g., PubMed Central).
Time Horizon	**Cross-sectional**	Data collection and analysis occur at a **single point in time** to evaluate the prototype's performance and validate its efficacy.

Table 5: Research Methodology

3.3 Development Methodology

This section details the step-by-step process for developing the system prototype. We will describe how we gather requirements, plan the system's architecture, our approach to programming, and our strategy for testing. Together, these steps form a complete blueprint for how the final solution will work.

3.3.1 Requirement Elicitation Methodology

To ensure the system is well designed and we will determine its requirements by:

- **Analyzing Pharmaceutical Documents:** We will examine research papers and reports from datasets like PMC-OA and DailyMed to understand their structure and the questions a researcher would ask.
- **Using Our Literature Review:** The research in Chapter 2 provides a list of modern techniques for data retrieval, cost-saving, and combining data types that the system must include.
- **Iterative Prototyping:** We will build and test simple, early versions of the system to check our assumptions, discover technical challenges, and refine the plan.

3.3.2 Design Methodology

The project will adopt an object-oriented analysis and design methodology (OOADM). This approach is highly suitable for the proposed system architecture, which is inherently modular and component-based. OOADM will allow for the clear definition of distinct objects and classes for each component—such as the API Gateway, IngestionWorker, VectorDatabaseClient, and ModelCascadeRouter—promoting code reusability, scalability, and maintainability.

3.3.3 Programming Paradigm

The primary programming paradigm will be **Object-Oriented Programming (OOP)**. This choice is naturally aligned with the selection of Python as the main development language and the use of frameworks like FastAPI and libraries such as LangChain, which are built upon OOP principles. This paradigm will facilitate the creation of a clean, modular, and scalable codebase.

3.3.4 Testing Methodology

A multi-layered testing methodology will be used to ensure the system is correct and valid:

- **Tier-1 Classifier Testing:** The local LLM will be tested independently using a set of 100 questions to measure its Query Classification Accuracy (F1-score) in correctly labeling queries ("text-only," "SQL-only," or "hybrid").

- **Retrieval Unit Testing:** We will test Vector DB text retrieval for Recall@k and SQL retrieval for Execution Accuracy using known-answer queries.
- **End-to-End System Testing:** The full prototype will be compared against the "vector-only" baseline. This gathering will provide the main metrics: **Cost-Efficiency Ratio (CER)** and final **Answer Accuracy (F1-score)**.

3.3.5 Solution Methodology

The development follows a structured, sequential pipeline that uses a dual-database ingestion system, abandoning standard table chunking.

Key Stages

1. **Dataset Collection:** Assemble the main corpus by downloading a subset of articles from **PMC-OA and DailyMed**.
2. **Hybrid Ingestion Pipeline:** This core engineering task uses a custom script to process each document:
    - **Text Processing:** Unstructured text is extracted, cleaned, and chunked into semantic units.
    - **Table Processing:** Structured tables are extracted, cleaned, and standardized into a relational schema.
3. **Database Population (Dual Storage):** Processed data is sent to two optimized databases:
    - **Vector Database (e.g., Qdrant):** Loads embedded text chunks (using a model like PubMedBERT) for **semantic search**.
    - **SQL Database (e.g., SQLite):** Loads standardized tables for **precise, factual queries**.
4. **Model Selection & Implementation:**
    - **Tier-1 LLM (Classifier):** A small, efficient, local open-source LLM (e.g., Mistral 7B variant) is used to classify the query type.

- **Tier-2 LLM (Synthesizer):** A powerful, paid API (e.g., Anthropic/OpenAI) handles the final, complex **answer synthesis**.

5. **Agent Orchestration:** The "brain" (using LangChain or custom scripts) manages the process:
    - Receives query.
    - Passes query to Tier-1 Classifier.
    - Retrieves context from Vector DB, SQL DB, or both, based on classification.
    - Passes original query and context(s) to Tier-2 Synthesizer for the final answer.
6. **Testing & Feedback:** The system undergoes rigorous evaluation (benchmarking). Results are used to refine components (e.g., classifier prompt, SQL schema, embedding model).

3.4 Project Management Methodology

To ensure this project is completed successfully and on schedule, we've defined a clear plan covering its scope, timeline, and potential risks.

Project Scope

To manage expectations, we've clearly defined what this project will and will not include.

What this project will deliver:

- A functioning proof-of-concept of the two-tier AI agent.
- A data-processing system that can intelligently parse text and tables from documents.
- A controlled experiment that benchmarks our system's performance.
- A detailed evaluation focused on cost-efficiency, answer quality, and speed.
- A final thesis report detailing the research and its findings.

What is not included:

- A production-ready application with a graphical user interface (GUI).
- Deployment into a live business environment.
- Processing of continuous, real-time data.
- Training a new large language model from scratch.

3.4.2 Schedule

3.4.2.1 Deliverables: Literature review, prototype development, final report submission.

Delivarebles	**Dates**
Project Proposal	
Literature Review	
Software Requirement Specification	
Project Proposal – initial draft	2th Sep 2025
Project Proposal and Requirement Specification – final draft	24th Oct 2025
Project Proposal and Requirement Specification – Final	13th Nov 2025
Proof of Concept	
Design Document	
Prototype	2nd Feb 2026
Interim Project Demo	2nd Feb 2026
Implementation	
Testing	
Evaluation	
Thesis Submission	1st Apr 2026
Minimum Viable Product	1st Apr 2026
…..	

Table 6: Deliverables

3.4.3 Resource Requirements

Successful completion of this project is contingent upon the availability of specific hardware, software, data, and technical skills.

3.4.3.1 Hardware

A standard laptop or workstation (min. 8GB RAM) will be the central hub for all core development (writing code, managing small data subsets, and running parsing scripts).

Cloud Computing (for GPU Tasks)

Access to a cloud computing platform with GPU capabilities is required for two high-performance tasks:

- Running the Tier 1 Local LLM: A service (like Google Colab or on-demand AWS/GCP) will host and run the quantized, open-source LLM.
- Large-Scale Data Processing: A cloud GPU will be used for the initial, one-time process of efficiently embedding the full document corpus.

3.4.3.2 Software

- **Programming Language:** Python (version 3.10 or higher).
- **Core Libraries:**
    - **Machine Learning:** PyTorch, Transformers (from Hugging Face), Scikit-learn.
    - **RAG & Orchestration:** LangChain or a similar framework.
    - **Web Framework:** FastAPI for creating the API gateway.
    - **Data Handling:** Pandas, NumPy.
    - **Document Parsing:** pdfplumber, lxml.
- **Vector Database:** A self-hosted, open-source vector database such as Qdrant or Weaviate.
- **Development Environment:** Visual Studio Code with Python extensions, Jupyter Notebooks for experimentation.

3.4.3.3 Data

- **Primary Corpus:** A large, curated subset of the PubMed Central Open Access (PMC-OA) dataset, containing full-text biomedical research articles.
- **Supplementary Corpus:** A collection of structured drug labels obtained from the DailyMed database to test the system's generalizability.

3.4.3.4 Skills

The project requires expertise in four main areas:

- Machine Learning : Strong knowledge, especially in Natural Language Processing (NLP), LLM architectures, and the Retrieval-Augmented Generation (RAG) framework.
- Python Programming: Advanced proficiency in Python, including Object-Oriented Programming (OOP) and experience with relevant software libraries.
- Data Science & Engineering: Expertise in data cleaning, preprocessing, and engineering, with a focus on parsing complex documents containing both text and tables.
- System Architecture: Ability to design, implement, and integrate modular, API-driven systems.

3.4.4 Risk and Mitigation (Expanded with Severity and Frequency)

Risk Exposure = Probability of Occurance * Magnitude of Loss

A proactive risk management plan is essential for anticipating and addressing potential challenges throughout the project lifecycle. The following table identifies key risks, assesses their potential impact using a risk exposure score, and outlines specific mitigation strategies. The risk exposure is calculated as the product of Severity (1-5) and Frequency (1-5).

Risk	S	F	RE	Mitigation Strategy
Ineffective Query Classifier	5	3	15	Build and test the Tier-1 classifier on a manually labeled test set to ensure high F1-score.
Failed Text-to-SQL Query	5	2	10	**Do not let the LLM write raw SQL.** Use the LLM only to extract entities (e.g., drug name,

Risk	S	F	RE	Mitigation Strategy
				dosage), then use safe, pre-written Python/SQL templates.
PDF Table Parsing (ETL) Failure	4	4	16	Implement robust validation and error-logging. Focus on a "good enough" pipeline that correctly parses 80% of tables, rather than a perfect one.

Table 7: Risk Mitigation

3.4 Chapter Summary

This chapter has comprehensively detailed the "how" of the project, covering the research, development, and management methodologies.

It began by establishing the **Research Methodology** (Section 3.2), using Saunders' Research Onion to justify the selection of a **Pragmatic** philosophy and a **Deductive**, quantitative experimental approach.

The chapter then transitioned to the core technical plan in the **Development Methodology** (Section 3.3). This section detailed the novel architecture of the **hybrid-retrieval RAG agent**. It outlined the specific solution pipeline, starting with the dual-database ingestion process: parsing unstructured text into a **Vector Database** and structured tables into a **SQL Database**. It also defined the implementation of the **two-tier model cascade**, which acts as the system's "brain" for query classification and cost-saving orchestration.

Finally, the **Project Management Methodology** (Section 3.4) was presented. This section provided the practical roadmap for implementation, defining a clear project **scope**, a detailed **schedule** and GANTT chart, specific **resource requirements**, and a proactive **risk management** plan. This risk plan was updated to address the new, specific technical challenges of the hybrid architecture, such as classifier accuracy and Text-to-SQL reliability.

Chapter 04: Software Requirement Specification (SRS)

4.1 Chapter Overview

The Software Requirement Specification (SRS) chapter provides a detailed analysis of the system's requirements, how they were gathered, the stakeholders involved, their viewpoints, and the functional and non-functional requirements of the system. It also includes visual representations such as the Rich Picture Diagram, Stakeholder Onion Model, Context Diagram, and Use Case Diagram to help understand the system's flows and interactions. Requirements are prioritized using the MoSCoW principle to ensure the focus on what is essential for the system's success.

4.2 Rich Picture Diagram

The Rich Picture Diagram (RPD) represents the system and its environment. It provides a high-level, informal view of the system's interactions with its users, stakeholders, and processes. It aids in visualizing the complexity of the project and identify important areas of concern, potential conflicts, and interactions.

The following diagram shows the environment for the **Hybrid-Retrieval RAG Agent**, showing the flow of data from ingestion to the final user, and the key stakeholders and technical components involved.

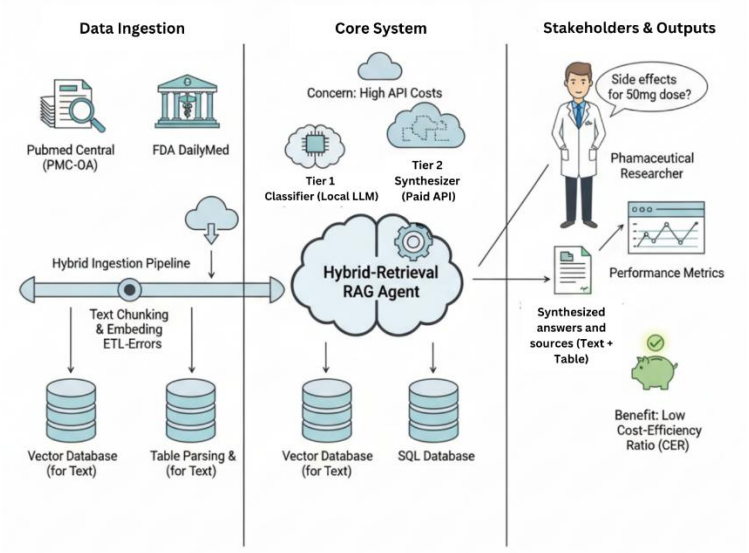

Figure 2: Rich Picture Diagram

4.3 Stakeholder Analysis

Stakeholder analysis is a crucial process for identifying all parties. It includes who will interact with, will be affected by, or have an interest in the **Hybrid-Retrieval RAG Agent**. Understanding their diverse needs and expectations is essential for defining the system's requirements.

4.3.1 Stakeholder Onion Model

The following model categorizes stakeholders into four primary layers:

4.3.2 Stakeholder List and Viewpoints

- **The System (Core):**
    - **Hybrid-Retrieval RAG Agent:** The software product itself, including both the databases (Vector DB, SQL DB) and the model cascade orchestrator.
- **Users (Directly Interact):**
    - **Pharmaceutical Researcher / Scientist:** (Primary User)
        - **Viewpoint:** "I need fast, accurate, and traceable answers. I must be able to trust the information, especially when it comes from a table. The system must be able to understand my complex, domain-specific questions."
    - **Data Scientist:**
        - **Viewpoint:** "I need to be able to evaluate the system's accuracy. I need to understand its performance metrics (CER, F1-score) and potentially refine its models."
- **Business (Internal Stakeholders):**
    - **R&D Management / Project Sponsor:**
        - **Viewpoint:** "This system must provide a clear return on investment. It needs to reduce API costs (justifying the CER) and demonstrably accelerate our research and drug discovery timelines."
    - **System Developer (Student):**
        - **Viewpoint:** "I need to successfully build and test a novel, functional prototype that meets all the academic and technical objectives defined in the proposal."
- **The Environment (External Stakeholders):**
    - **Pharmaceutical Companies (Wider Industry):**
        - **Viewpoint:** "We are interested in any new, validated architecture that can reduce our massive R&D data processing costs and improve accuracy."
    - **Data Providers (e.g., PubMed, FDA):**
        - **Viewpoint:** "Our data must be used in accordance with our access policies, and its limitations must be understood."

4.4 Elicitation Techniques

To define the system's functional and non-functional requirements, the following elicitation approach will be used:

- **1. Literature Review:**
    - **Description:** An in-depth review of existing research (as detailed in Chapter 2) was conducted to identify already established problems, a baseline for system features, and important evaluation metrics (like RAGAs, CER, and F1-score) used in the academic community.
- **2. Stakeholder Interview:**
    - **Description:** A semi-structured interview will be conducted with a **Data Analyst** who has experience in the business intelligence or data science field.
    - **Justification:** This interview is critical for gathering a real-world perspective on the problem. It will help to:
        - Validate the project's core assumptions.
        - Understand how a technical user would *actually* query a complex system (text vs. SQL).
        - Define the practical requirements for answer traceability, accuracy, and performance (latency).
- **3. Archival Research:**
    - **Description:** A direct study of the primary data sources (PubMed Central and DailyMed) was done.
    - **Justification:** This technical investigation was necessary to define the requirements of the ingestion pipeline (e.g., handling inconsistent table structures, parsing XML) and to confirm the need of a hybrid (SQL + Vector) storage solution.

4.5 Discussion of Findings

4.5.1 Literature Review Findings

Finding 1: Standard RAG systems are "unimodal" (text-only) and fail when they encounter structured data, a problem known as "Lost-in-Vectorization."

- **Conclusion:** Naively vectorizing tables destroys their row-column integrity, leading to garbled context and factually incorrect answers. This confirms that a separate, specialized handling for tables (like a SQL database) is required.
- **Citation:** (Manjee, 2024); (Herzig et al., 2020)

Finding 2: Current research is siloed. Text-to-SQL frameworks focus *only* on relational data, while RAG systems focus *only* on unstructured text.

- **Conclusion:** There is a distinct lack of unified systems that can orchestrate a *single query* across both structured (SQL) and unstructured (Vector) databases to answer a "hybrid" question.
- **Citation:** (Liu et al., 2023); (Hao et al., 2024)

Finding 3: Most RAG architectures ignore operational cost, relying on a single, expensive LLM API for all tasks.

- **Conclusion:** This "single-model" approach is financially unviable at scale. LLM Cascades are a proven, state-of-the-art method for cost-optimization, but they have not been integrated with hybrid-retrieval systems.
- **Citation:** (Yue et al., 2023); (Rodriguez et al., 2023)

4.5.2 Interviews Findings

Codes	Themes	Conclusion
"Manual data joining"	**Data Fragmentation**	A primary user pain point is the manual effort required to link unstructured text (reports)

Codes	Themes	Conclusion
"Split data sources" "Text and database lookup"		with structured data (tables)**.** This validates the need for a hybrid-retrieval system.
"Source citation" "Need for verification" "Don't trust 'black box'"	**Answer Traceability**	For an answer to be trusted, it must be 100% traceable. The system **Must** provide source links to the exact text chunk and table row used.
"Accuracy over speed" "Cost of bad decisions" "100% correct"	**Primacy of Accuracy**	Users strongly prefer a slower, perfectly accurate answer over a fast, 'mostly correct' one. This prioritizes retrieval/generation quality over latency.

Codes	Themes	Conclusion
"Wrong answer is dangerous"		
"API cost scrutiny" "Budget approval" "Cost-saving as feature" "Justifying expense"	**Cost-Efficiency as a Core Requirement**	Cost-quantification (the CER) is a critical adoption feature, not just a bonus. The model cascade is a key business justification.
"Silent query failure" "Plausible but wrong" "Missed a row"	**Structured Data Integrity**	Users fear "plausible but wrong" answers from tables. This validates that using precise SQL retrieval instead of "lossy" vector summarization of tables.

Table 8: Interview Findings

4.7 Context Diagram

The Context Diagram (also known as a Level 0 Data Flow Diagram) depicts the high-level boundary of the **Hybrid-Retrieval RAG Agent**. It models the system as a singular and central process and identifies all external entities (such as users, other systems, and data stores) that interact with the system. The diagram shows the key data flows (inputs and outputs) between the system and these external entities. It provides a clear, formal overview of the system's scope and its interactions.

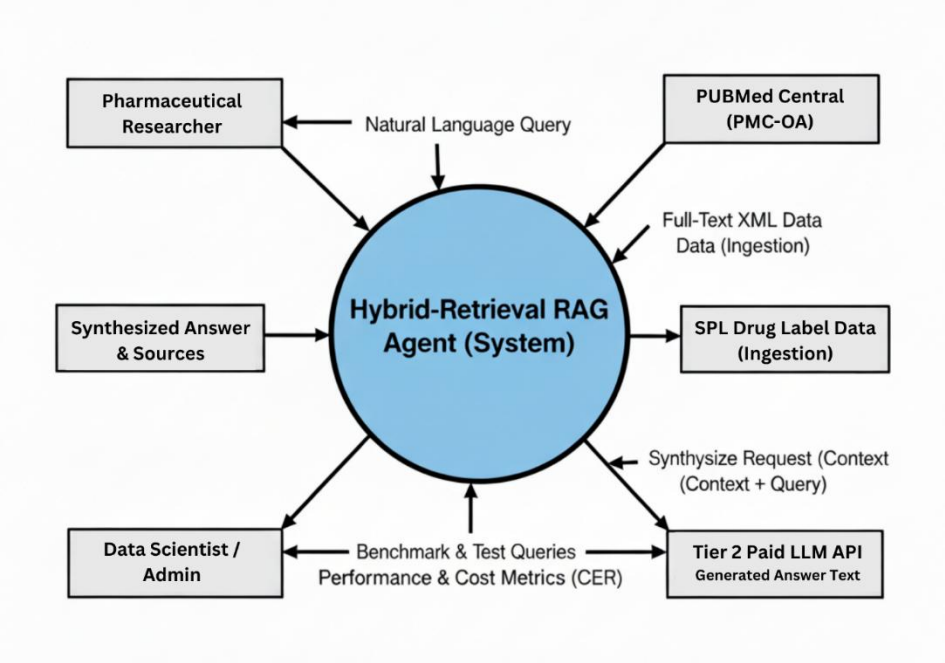

Figure 3: Context Diagram

4.8 Use Case Diagram

The Use Case Diagram illustrates the system's functionalities (use cases) and how external actors interact with them, defining the functional requirements of the **Hybrid-Retrieval RAG Agent**.

4.8.1 Actors

- **Pharmaceutical Researcher:** Primary user looking for answers from the system.
- **Data Scientist / Admin:** Technical user responsible for system evaluation and performance monitoring.

- **<<Secondary>> Tier-2 Paid LLM API:** External LLM service called for complex answer synthesis.

4.8.2 Use Cases

- **Query System with Natural Language:** Main function for Pharmaceutical Researcher to submit their questions.
    - <<includes>> Classify Query (Tier-1): Local LLM categorizes query intention.
    - <<includes>> Retrieve Hybrid Context: System fetches data from Vector DB and/or SQL DB.
- **Synthesize with Tier-2 API:** <<extends>> Query System with Natural Language for complex queries gets escalated to the external LLM.
- **Manage & Evaluate System:** For Data Scientist / Admin to test and monitor system performance.

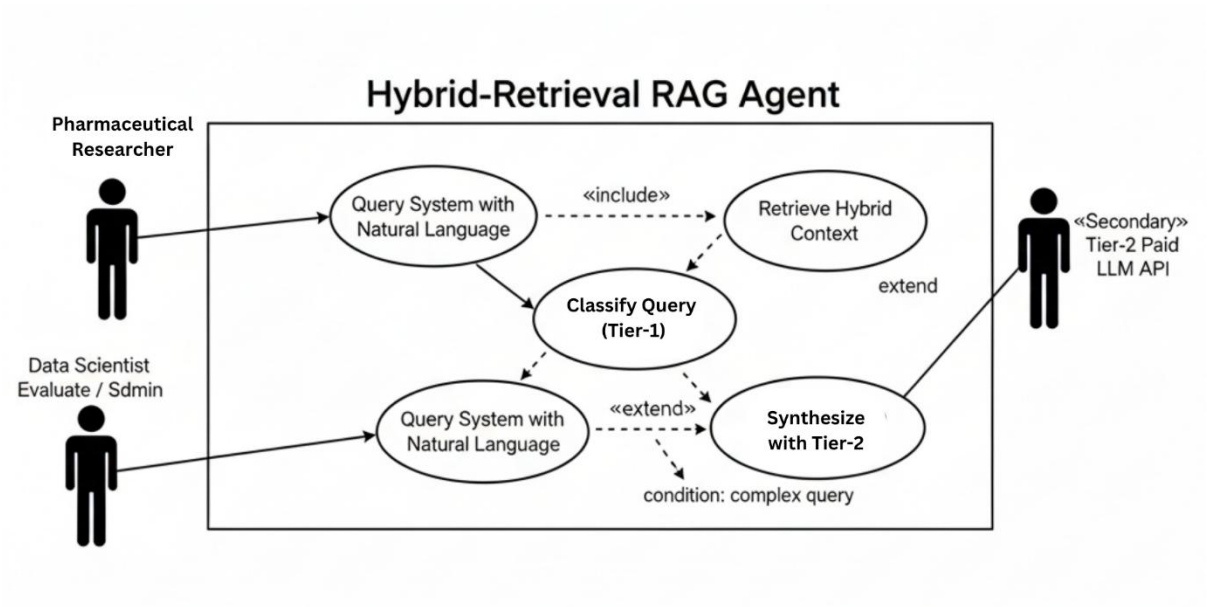

Figure 4: Use Case Diagram

4.9 Use Case Descriptions

This section provides a description of the primary use cases identified in the Use Case Diagram.

Use Case ID: UC-001

Use Case Name: Query System with Natural Language

Primary Actor: Pharmaceutical Researcher

Secondary Actor: Tier-2 Paid LLM API

Priority: High

Description: This use case describes the main function of the system. The researcher submits a query in plain English. The system then orchestrates the full hybrid-retrieval and model cascade pipeline to return a synthesized, and cost-effective answer.

Main Flow (Successful Scenario):

1. The Pharmaceutical Researcher submits a natural language query via the system interface.
2. The system receives the query and passes it to the **Tier-1 (local) LLM** for classification.
3. The Tier-1 LLM classifies the query intent (e.g., "text-only," "sql-only," or "hybrid").
4. The system orchestrator executes the retrieval plan:
    - If "text-only," it performs a semantic search on the **Vector Database**.
    - If "sql-only," it uses the Text-to-SQL tool on the **SQL Database**.
    - If "hybrid," it performs *both* retrievals in parallel.
5. The system gathers the retrieved context(s).
6. The system identifies that the query is complex enough to require escalation to the paid API (based on the classifier's decision).
7. **[Extension Point]** The query is escalated to the Tier-2 Paid LLM API. The system sends the original query and all retrieved context.
8. The Tier-2 Paid LLM API synthesizes the context and gives out a final answer.

9. The system receives the answer from the API.
10. The system presents the unified synthesized answer and its sources (e.g., text snippets, table references) to the Pharmaceutical Researcher.

Alternative Flows:

- **6a. (Simple Query):** The Tier-1 classifier determines the query is simple and can be answered by the local LLM. The system skips escalation to the Tier-2 API, synthesizes the answer locally, and proceeds to step 10.

Post-conditions:

- The researcher's query is answered.
- The cost and latency of the query are logged for evaluation.

Use Case ID: UC-002

Use Case Name: Manage & Evaluate System

Primary Actor: Data Scientist / Admin

Priority: Medium

Description: This use case describes the "backend" function for a technical person to test metrics such as system's performance and accuracy..

Main Flow (Successful Scenario):

1. The Data Scientist / Admin accesses the system's evaluation interface or scripts.
2. The Data Scientist / Admin submits a batch of test queries (the Ground-Truth Evaluation Set).
3. The system runs each test query through the full UC-001 pipeline.
4. The system logs the performance for each query:

- The generated answer.
- The retrieved context.
- The total API cost incurred.
- The end-to-end latency.

5. The Data Scientist / Admin runs an evaluation script to compare the generated answers against the ground-truth answers, calculating the **F1-score** and **Faithfulness**.
6. The Data Scientist / Admin runs a script to aggregate all logged costs and calculates the final **Cost-Efficiency Ratio (CER)**.
7. The system provides all metrics (F1, CER, Latency) to the Data Scientist / Admin for analysis.

Post-conditions:

- The system's performance is quantitatively benchmarked.
- The Data Scientist / Admin has the metrics needed for the project's final evaluation.

4.10 Functional & Non-functional Requirements

This section defines the functional ("what it does") and non-functional ("how well it does it") requirements for the prototype. They are categorized using the MoSCoW method (**M**ust Have, **S**hould Have, **C**ould Have, **W**on't Have).

4.10.1 Functional Requirements

ID	Requirement Description	Priority (MoSCoW)
FR01	The system must ingest and process unstructured text into a Vector Database.	**Must Have**

ID	Requirement Description	Priority (MoSCoW)
FR02	The system must ingest and process structured tables into a SQL Database.	**Must Have**
FR03	The system must accept a natural language query from the user.	**Must Have**
FR04	The system must use the Tier-1 (local) LLM to classify the query's intent (text, sql, hybrid).	**Must Have**
FR05	The system must retrieve context from the Vector DB for text/hybrid queries.	**Must Have**
FR06	The system must retrieve context from the SQL DB for sql/hybrid queries.	**Must Have**
FR07	The system must synthesize a final answer using the Tier-2 (paid) LLM for complex queries.	**Must Have**
FR08	The system should provide source references for the retrieved context.	**Should Have**
FR09	The system monitor the API cost and latency for each query to calculate the CER.	**Should Have**
FR10	The system could provide a basic web-based UI for submitting queries.	**Could Have**

ID	Requirement Description	Priority (MoSCoW)
FR11	The system won't have user authentication or multi-user accounts in this prototype.	**Won't Have**

Table 9: Functional Requirements

4.10.2 Non-functional Requirements

ID	Requirement Description	Priority (MoSCoW)
NFR01	**Accuracy (Faithfulness):** The system must generate answers that are factually supported in the retrieved context (Vector DB or SQL DB).	**Must Have**
NFR02	**Cost-Efficiency:** The system must demonstrate a measurably lower Cost-Efficiency Ratio (CER) than the vector-only baseline.	**Must Have**
NFR03	**Performance (Latency):** The system should return a synthesized answer for a hybrid query within 15 seconds.	**Should Have**
NFR04	**Scalability (Ingestion):** The ingestion pipeline should be able to process at least 1,000 documents.	**Should Have**
NFR05	**Usability:** The system could have an intuitive interface that clearly differentiates the query input from the output answer.	**Could Have**

ID	Requirement Description	Priority (MoSCoW)
NFR06	**Availability:** The system won't have 24/7 high-availability or real-time fault tolerance.	**Won't Have**

Table 10: Non-functional Requirements

4.11 Chapter Summary

This chapter detailed the **Software Requirement Specification (SRS)** for the **Hybrid-Retrieval RAG Agent**.

It began with the **Rich Picture Diagram** and **Stakeholder Analysis**, illustrating the system's environment and key actors. The **Elicitation Techniques**, including a stakeholder interview, were outlined. The **Context Diagram** formally defined the system boundary, while the **Use Case Diagram** and **Descriptions** mapped out its primary functionalities, including the conditional use of the Tier-2 LLM. Finally, the **Functional and Non-functional Requirements** were specified and prioritized (MoSCoW), emphasizing accuracy, cost-efficiency, and hybrid data processing as core objectives.

References

Asai, A., Wu, Z., Wang, Y., Sil, A. and Schlichtkrull, M. (2023) *Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection*. Available at: https://arxiv.org/abs/2310.11511.

BaranziniLab (2024) 'Biomedical knowledge graph-optimized prompt generation for large language models', *Bioinformatics*, 40(9).

Chen, J., Wang, L. and Zhang, Y. (2022) 'A Generic Retrieval-Augmented Generation Framework for Enterprise Knowledge Bases', in *Proceedings of the 2022 International Conference on Information and Knowledge Management (CIKM)*, Atlanta, GA, USA, pp. 112-121.

Day, A. (2024) *Unlocking Hybrid Search: A Deep Dive into Allen Day's Solr MCP Server*. Available at: https://skywork.ai/skypage/en/hybrid-search-solr-mcp/ (Accessed: 30 October 2025).

Es, S., et al. (2023) *RAGAs: Automated Evaluation of Retrieval Augmented Generation*. Available at: https://arxiv.org/abs/2309.15217.

Fanconi, T., Schaar, M. and Jitkrittum, W. (2024) *Towards a Cascaded LLM Framework for Cost-effective Human-AI Decision-Making*. Available at: https://arxiv.org/abs/2405.15838.

Gao, P., et al. (2024) 'RAG-based Text2SQL Business Intelligence System Based on Retrieval-Augmented Generation (RAG)', *DOAJ*, 2024.

Gu, Y., et al. (2021) 'Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing', *ACM Transactions on Computing for Healthcare*, 3(1), pp. 1-23.

Gupta, R. and Li, X. (2023) 'Multimodal Fusion for Financial Report Analysis: Combining Textual and Tabular Data', *Journal of Financial Data Science*, 5(3), pp. 45-60.

Hao, S., et al. (2024) 'SQL-RAG: A RAG Framework for SQL Database Querying', in *Proceedings of the 2024 ACM SIGMOD International Conference on Management of Data*.

Herzig, J., et al. (2020) 'TAPAS: Weakly Supervised Table Parsing via Pre-training', in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4320-4333.

Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C.H. and Kang, J. (2020) 'BioBERT: a pre-trained biomedical language representation model for biomedical text mining', *Bioinformatics*, 36(4), pp. 1234–1240.

Lewis, P., et al. (2020) 'Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks', in *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, pp. 9459-9474.

Liu, P., et al. (2023) 'A Comprehensive Survey of Text-to-SQL', *ACM Computing Surveys*, 56(2), pp. 1-38.

Manjee, S. (2024) *From PDF tables to insights: An alternative approach for parsing PDFs in RAG*. Available at: https://www.elastic.co/search-labs/blog/alternative-approach-for-parsing-pdfs-in-rag (Accessed: 30 October 2025).

Rodriguez, F., Patel, S. and Kim, M. (2023) 'Cost-Effective Query Routing for Large Language Model Cascades in Conversational AI', in *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, pp. 2015-2025.

Sadekar, O. (2024) *Natural Language to SQL with LangChain + RAG: A Practical Guide*. Available at: https://medium.com/@OmkarSadekar/natural-language-to-sql-with-langchain-rag-a-practical-guide-1841616c8523 (Accessed: 30 October 2025).

Sahoo, S., et al. (2024) *Balancing Content Size in RAG-Text2SQL System*. Available at: https://arxiv.org/abs/2502.15723.

Vaswani, A., et al. (2017) 'Attention Is All You Need', in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 5998-6008.

Yue, M., Zhao, J., Zhang, M., Du, L. and Yao, Z. (2023) 'Large Language Model Cascades with Mixture of Thoughts Representations for Cost-efficient Reasoning', in *International Conference on Learning Representations (ICLR)*.

Zhang, H., et al. (2024) *Cost-Saving LLM Cascades with Early Abstention*. Available at: https://arxiv.org/abs/2502.09054.