Retrieval-Augmented AI Assistants for Healthcare: System Design and Evaluation

Mark Meleka

Kellogg College
University of Oxford

A dissertation submitted for the
MSc in Software Engineering

Abstract

This dissertation explores the application of Retrieval-Augmented Generation (RAG) in AI models to address the challenges posed by fragmented personal health data. Contemporary healthcare systems often struggle with scattered patient information, hindering efficient and informed decision-making. This work investigates whether a RAG-enhanced AI assistant can reliably answer natural-language queries over personal health records, focusing on accuracy, transparency, and auditability using only synthetic data for privacy.

The primary contribution is a modular, local-first prototype system designed to ingest, chunk, embed, and retrieve information from PDF health records to ground AI-generated answers. A comprehensive evaluation was conducted, systematically varying RAG parameters like chunk size, retrieval depth, and embedding models across different query types (factual, multi-document, contradiction). This dissertation presents findings on the trade-offs between retrieval precision, answer correctness, latency, and cost. A detailed failure mode analysis links design choices to risks like fact fabrication and missed context, revealing that AI model reasoning, rather than information retrieval, is often the main bottleneck for complex queries. Ethical, legal, and social implications are examined, proposing safeguards such as verifiable citations and calibrated uncertainty. The work concludes by reflecting on the design, evaluation, and potential for scaling such systems responsibly.

Acknowledgements

I am deeply grateful to my supervisor, Professor Andrew Markham, for his invaluable guidance—his feedback helped shape this dissertation and challenged me to think about the broader implications of my work. My thanks extend to the Department of Computer Science for a rigorous MSc programme and to Kellogg College, particularly the MCR committee, for the friendship and support. To my family and friends, especially my parents, thank you for your patience, understanding, and encouragement throughout this endeavour. Finally, thank you to the AI research community for making this field such an exciting space to learn.

Contents

List of Acronyms

AI Artificial Intelligence
ANN Approximate Nearest Neighbour
API Application Programming Interface
EHR Electronic Health Record
ELSI Ethical, Legal, and Social Implications
GDPR General Data Protection Regulation
GPU Graphics Processing Unit
HIPAA Health Insurance Portability and Accountability Act
LLM Large Language Model
OCR Optical Character Recognition
PHI Protected Health Information
RAG Retrieval-Augmented Generation
UI User Interface
USMLE United States Medical Licensing Exam
WHO World Health Organization

List of Figures

List of Tables

1 Introduction

1.1 Context and Motivation

Personal health data are increasingly digital, yet remain scattered across patient portals, email attachments, laboratory portals, and scanned documents. This fragmentation obscures clinical context, delays diagnoses, and encourages redundant tests. Existing consumer tools provide storage, not synthesis; they force patients and medical professionals to read every document or memorize medical history. The resulting cognitive load jeopardises informed decision-making and shared care.

Artificial Intelligence (AI) models, including *Large Language Models (LLMs)*, offer a promising remedy. Instruction-tuned variants such as `Med-PaLM` already answer medical questions at near-clinician accuracy on benchmark datasets [43]. Unfortunately, standalone AI models hallucinate, forget long-range context, and embed outdated knowledge in their parameters [20]. *Retrieval-Augmented Generation (RAG)* mitigates those flaws by grounding each answer in external documents [17, 23]. In clinical summarization, combining RAG with an open-weight LLM raised factual accuracy from 93% to 99% while reducing hallucinations [1]. Yet we lack systematic evidence on how core RAG design choices—chunk size, retrieval depth, and embedding model—interact when the corpus is small, noisy, and patient-specific.

Our dissertation tackles this gap. We explore whether a RAG-enhanced AI assistant can reliably answer natural-language queries over fragmented personal health records, using only synthetic data for privacy. The work is technically motivated by the need to quantify trade-offs among accuracy, latency, and token cost, and ethically motivated by the imperative to ground high-stakes answers in verifiable evidence [16].

1.2 Project Objectives and Contributions

Our overarching aim is to demonstrate that a document-grounded assistant can answer health questions both accurately and transparently while remaining auditable on a single laptop. To realize this aim we pursued four mutually reinforcing objectives. First, we designed and implemented a modular pipeline that ingests PDFs, slices them into provenance-rich chunks, embeds them, retrieves evidence on demand, and passes a safety-conscious prompt to an external language model. Second, we conducted a full-factorial evaluation—varying *chunk size*, *retrieval depth* and *embedding model* across 540 query–configuration pairs—to quantify the trade-offs among retrieval precision, answer correctness, latency, and token economy. Third, we analysed failure modes in depth, producing a taxonomy that links technical design decisions to factual fabrication, missed context, and contradiction handling. Fourth, we examined the Ethical, Legal, and Social Implications (ELSI) of deploying such a system, proposing concrete safeguards—local storage, calibrated uncertainty, and human-in-the-loop review—that align with World Health Organization (WHO) guidance.

This dissertation contributes a reproducible, local-first framework that reveals how retrieval

parameters shape reliability and cost in medical RAG pipelines; it provides a fine-grained failure-mode taxonomy that maps engineering choices to ethical risk; and it offers a set of practical design patterns for privacy-preserving, traceable AI assistants in healthcare.

1.3 Scope and Limitations

We intentionally constrain scope to maximize depth. All experiments use synthetic Electronic Health Records (EHRs); no real patient data enter the pipeline, eliminating regulatory overhead while limiting ecological validity. The prototype supports a single user and stores embeddings in `SQLite`; it forgoes multi-user security, distributed search indices, and image modalities. We evaluate only two OpenAI embedding models and one OpenAI generation model. Reasoning-optimized AI models and domain-specific embeddings, though likely beneficial [46, 51], remain future work. Finally, we analyse offline questions, not live dialogue, so we do not measure longitudinal memory effects.

1.4 Research Questions

The primary question in our evaluation is, "How do chunk size, number of chunks used (*top-k*), and embedding model affect retrieval precision and answer correctness across different query types (factual, multi-document, contradiction)?" The supporting questions are:

1. Which retrieval parameters most strongly correlate with each category of failure (e.g. hallucination, omission, contradiction) across query types?
2. How do chunk size, top-k, and embedding choice influence token consumption and end-to-end latency?
3. What are the practical trade-offs between accuracy (retrieval precision, correctness) and efficiency (tokens, latency) when tuning RAG parameters?

The primary question is framed as a falsifiable hypothesis in Chapter 4 and subjected to formal statistical testing; the supporting questions guide targeted descriptive analyses, visualizations, and correlation studies to contextualize our quantitative results.

1.5 Dissertation Structure Overview

This dissertation unfolds over the subsequent chapters in a deliberate arc from motivation to reflection. Chapter 2 surveys the landscape of healthcare information challenges and RAG, anchoring the work in clinical need and technical context. Chapter 3 details the system design, justifying architectural choices like local-first storage and modular adapters against project goals and constraints. Chapter 4 presents the rigorous evaluation methodology and results, quantifying the impact of RAG parameters (chunk size, retrieval depth, embedding model) on performance and failure modes across different query types. Chapter 5 examines the crucial ethical, legal, and social implications, mapping risks to established principles and proposing mitigations. Chapter 6 offers reflections on the design process, evaluation findings, and risk management lessons learned. Finally, Chapter 7 synthesizes these contributions, answers the research questions, and discusses limitations and future directions. Appendices provide supplementary data and the full system prompt to aid reproducibility. Through this structure, the dissertation aims to demonstrate mastery of software engineering principles applied to a significant real-world problem.

2 Background

2.1 Challenges in Contemporary Healthcare

Modern healthcare stands at a paradoxical juncture. Chronic, non-communicable diseases now account for the majority of deaths worldwide and are projected to cost US $47 trillion by 2030 [18, 53]. Yet the systems intended to manage this burden struggle under their own weight. The United States spends almost twice as much on care as peer nations while delivering lower life expectancy and higher infant mortality [35]. Globally, an estimated 4.3 million clinicians are missing from the workforce, and where staff are available they are increasingly exhausted: nearly half of US physicians report at least one symptom of burnout [3, 42]. These pressures erode quality, inflate costs, and leave little slack for innovation.

Against this strained backdrop, health outcomes remain unevenly distributed. When clinical evidence is ambiguous, stereotypes—conscious or otherwise—creep into diagnostic and treatment choices, widening racial and ethnic disparities [4]. Justice therefore demands not only more resources but also better tools for consistent, evidence-based decision-making that resist bias and preserve fair equality of opportunity [12].

At the same time, digital health technologies are advancing. Telehealth, wearables, and mobile apps expanded rapidly during the COVID-19 pandemic and promise more continuous patient-centered care [9]. Classical clinical-decision-support (CDS) systems offer one proven avenue forward. By linking patient data to structured knowledge they reduce medication errors and improve adherence to guidelines, but they suffer from brittle rule bases [6].

Recent advances in AI models raise the prospect of more adaptive support: systems that can retrieve pertinent evidence from unstructured records, reason over it in real time and present concise, source-linked guidance. Such retrieval-augmented assistants could ease cognitive overload, surface otherwise overlooked context and provide transparent citations—directly addressing the twin problems of uncertainty and trust. With careful attention to transparency, cost, and fairness, these systems can narrow information gaps, support overextended clinicians, and lay groundwork for more equitable, evidence-driven care.

2.2 AI models for Retrieval Tasks

Modern AI models used for language tasks are often deep neural networks trained on enormous amounts of text, typically from the internet. To process text, they first break it down into discrete units called "*tokens*". Think of tokens as the basic vocabulary the model works with; a token usually represents a common word (like "patient") or even part of a word (like "measur" and "ing" making up "measuring"), along with punctuation. The core task the model learns during training is predicting the next token in a sequence, given the tokens that came before it. For example, if the input is "The patient's blood pressure today is ", the model uses the patterns it learned to predict the most probable next tokens, like "120/80 mmHg.". This process is often referred to as "*pre-training*". Crucially, this prediction is probabilistic, not deterministic like traditional

rule-based code. The model generates a likely continuation based on statistical patterns, not a guaranteed fact derived from logical rules. This probabilistic nature is key to their flexibility but also why they can sometimes make errors or "*hallucinate*" information.

Pre-training alone yields an autocomplete engine; post-training methods such as Reinforcement Learning from Human Feedback (RLHF) [44] or Direct-Preference Optimization (DPO) [38] realign the model into a task-oriented assistant. Ask the raw model "Is 40 °C a fever?" and it might complete with another half-sentence; ask the aligned model and it answers: "Yes—a core temperature of 40 °C constitutes a high-grade fever; seek urgent care if sustained."

The latest frontier AI models are multimodal—they process text, images, audio, and video in a single network. OpenAI's `GPT-4o` and Google's `Gemini 2.5 Pro`, for example, accept text, images, and audio [14, 33]. Older models, such as OpenAI's `GPT-3` [8], that are trained only on text were called Large Language Models. LLMs are a subset of the broader category of AI models. Since this dissertation focuses primarily on text input and output we use the terms "AI model" and "LLM" interchangeably.

Raw scale matters. Performance improves predictably as parameters (the internal variables, or *weights*, the model learns from data), data, and compute rise, a power-law result formalized by Kaplan et al. [22]. Yet bigger models incur super-linear cost and latency. A complementary strategy is knowledge distillation: a compact student network learns to mimic a large teacher, retaining much of its accuracy while shrinking memory footprint and inference cost [15]. Distillation is how OpenAI can offer slightly less intelligent versions of their models for a fraction of the cost (e.g. `GPT-4o-mini` for 6% of the cost of `GPT-4o` [33]). Distilled students can make on-device or edge deployment feasible in bandwidth-constrained clinics.

Scaling alone does not guarantee clinical competence. `Med-PaLM`, an instruction-tuned `PaLM` variant, reaches near-expert accuracy on United States Medical Licensing Exam-style questions while cutting potentially harmful replies from 30% to 6% [43]. Open-weight successors (models whose learned weights are publicly released)—`BioGPT`, `Clinical Camel` and others—replicate this pattern in biomedical corpora [46]. The implication is clear: with targeted tuning, modern AI models can reach a useful level of performance on clinical tasks like reading discharge notes, summarizing longitudinal records and answering specialist queries.

The sequence of tokens the model can consider at once is its "*context window*": currently up to one million tokens in state-of-the-art systems such as OpenAI's `GPT-4.1` [31] and Google's `Gemini 2.5 Pro` [14]. Reading input tokens or producing output tokens consumes computational resources (Graphics Processing Unit (GPU) time and electricity), which is why providers typically charge per-token fees or have usage limits on recurring revenue subscriptions.

However, these models have fundamental limitations. First, the context window, though now vast, is still finite and unevenly used. Research shows they often pay more attention to information at the very beginning or very end of the provided context, potentially missing details in the middle (referred to as "lost-in-the-middle" or primacy–recency bias) [27]. Second, the model's knowledge is essentially encoded in its network parameters (weights), learned during its initial training phase. This knowledge is static; the model doesn't automatically learn new information after training is complete. So, if medical guidelines change, the model won't know unless explicitly retrained or updated. This reliance on "frozen" knowledge is a major reason why models can hallucinate or fabricate facts—generating fluent, plausible-sounding statements that are incorrect, outdated, or simply not supported by the input evidence. Maleki, Padmanabhan, and Dutta [29] highlight the need for precise terminology around such failures. The RAG technique, discussed next, is specifically designed to mitigate this problem by providing current, relevant information within the context window.

2.3 Retrieval-Augmented Generation (RAG)

A standalone AI model cannot cite a document it has never seen. RAG addresses this gap by coupling the generator to an external knowledge store and treating generation as an "open-book" exercise. Rather than relying solely on parametric memory, the model retrieves relevant evidence at inference time, grounding its outputs on retrieved text.

Preparing documents for RAG involves several preprocessing steps, especially crucial given that clinical data often arrives in heterogeneous formats—scanned PDFs, mixed-layout discharge summaries, and hand-typed notes. Post-extraction cleaning, such as normalizing whitespace and correcting Optical Character Recognition (OCR) artefacts, is helpful to avoid injecting noise into downstream processes like embedding generation.

Once cleaned text is available, it must be divided into manageable pieces, or *chunks*. Various chunking strategies exist, as illustrated in Figure 2.1. Fixed-size overlapping chunks (e.g., 300 tokens with 50-token overlap) are common for simplicity. Alternatives include splitting by sentence or paragraph boundaries, semantic clustering based on embedding similarity, or hybrid methods. While dynamic chunking can theoretically preserve semantic cohesion better, empirical studies suggest that well-tuned fixed chunks often rival more complex approaches [37].

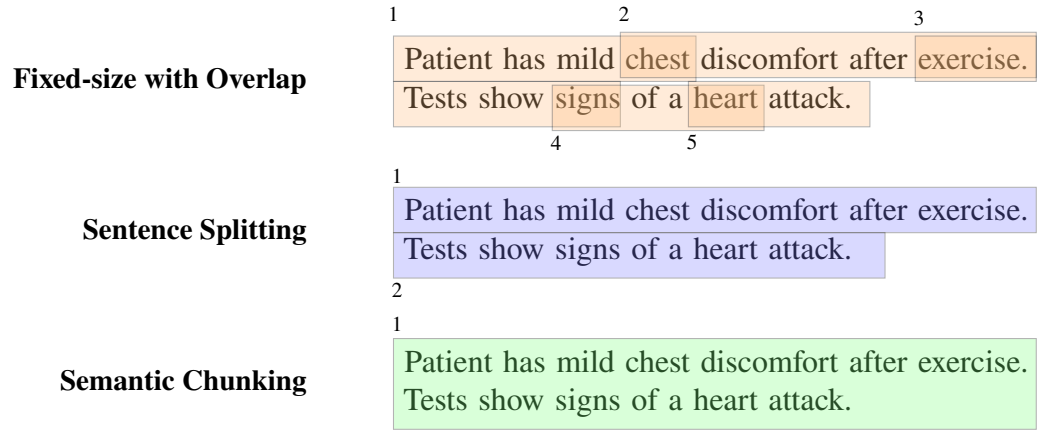


Figure 2.1: Visualization of Chunking Strategies (simplified)

Next, each text chunk is processed by an embedding model, which converts the text into a numerical vector (essentially, a list of numbers), or *embedding*, often with hundreds or thousands of dimensions. The key idea is that these embeddings capture the semantic meaning of the text. The model is trained so that text chunks with similar meanings are mapped to vectors that are close to each other in high-dimensional *embedding space*, while chunks with different meanings result in vectors that are far apart (conceptualized in Figure 2.2). This numerical representation allows the system to search for relevant chunks based on meaning (e.g., semantic search), which is much more powerful than just matching keywords. For example, a search for "heart issues" could find chunks mentioning "cardiac problems" or "arrhythmia" even if the exact words "heart issues" aren't present, because their embedding vectors would be close.

Various embedding models can perform this mapping. General-purpose encoders such as OpenAI's `text-embedding-3-small` perform well on a wide variety of tasks and at low cost [34], while domain-specific embedding models such as `BioBERT` [25] or `ClinicalBERT` [2] may better capture biomedical synonymy and abbreviations. Larger models, or ones trained on domain-specific data, aim to better represent semantic similarity in embedding space. To maintain transparency and traceability, crucial metadata—such as document IDs, page ranges,

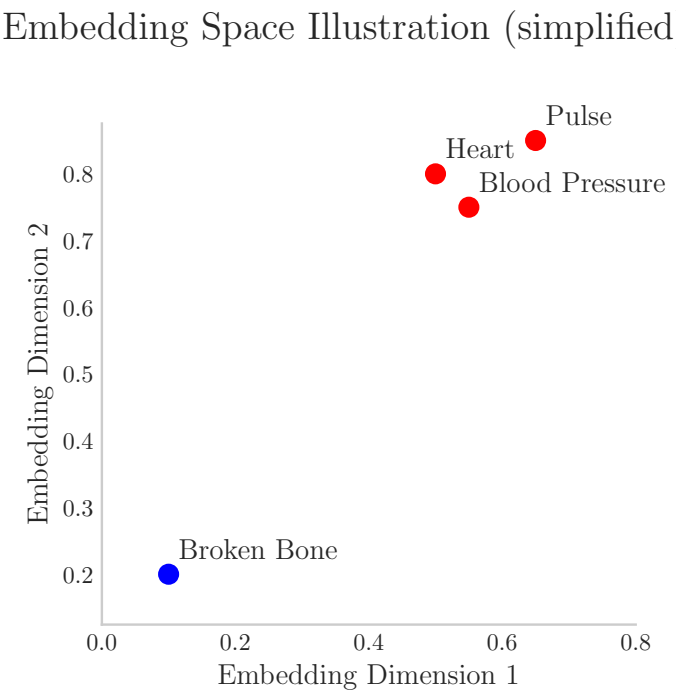


Figure 2.2: Embedding Space Illustration (simplified)

source timestamps, and original filenames—are typically stored alongside each chunk's vector embedding.

During retrieval, when a user poses a query, that query is embedded using the same encoder. The resulting query vector is then matched against the stored chunk vectors, typically using cosine similarity or an Approximate Nearest Neighbour (ANN) search, to find the most semantically relevant chunks containing potential evidence. Figure 2.3 illustrates the overall flow of the RAG pipeline, including this retrieval step. Explicit retrieval has proven beneficial even for very large models. In healthcare, adding an embedding retriever to Meta's `LLaMA-2` model slashed hallucinations and achieved over 99% structured-field accuracy when summarizing Electronic Health Records [1].

After retrieval, the system faces a bandwidth bottleneck: only a finite number of tokens fit into the AI model's prompt. Poorly ordered, redundant, or bloated context can dilute attention and trigger the lost-in-the-middle failure mode mentioned in Section 2.2. To mitigate this, prompts are constructed with strict structure: a system message defining task behaviour, retrieved chunks labelled by source, and the user query placed clearly. Chronological or relevance-based ranking further exploits primacy and recency biases. Prompt patterns such as the Fact Check List and Context Manager [52] help scaffold the model's reasoning, nudging it to synthesize faithfully grounded responses. *Chain-of-thought* prompting, where intermediate steps are made explicit, can further improve multi-step reasoning but increases token usage and latency [50]. Some models, like OpenAI's `o3` or Google's `Gemini 2.5 Pro`, are designed to spend time "thinking" before they respond.

Selecting the right RAG parameters—like chunk size, top-k (how many chunks are retrieved for a query), and the specific embedding model used—requires balancing competing factors. Small chunks can provide very focused context to the AI model, reducing irrelevant noise. However, if a key piece of information is split across two chunks, retrieving only one might miss the full picture. Large chunks have a better chance of capturing all related information within a

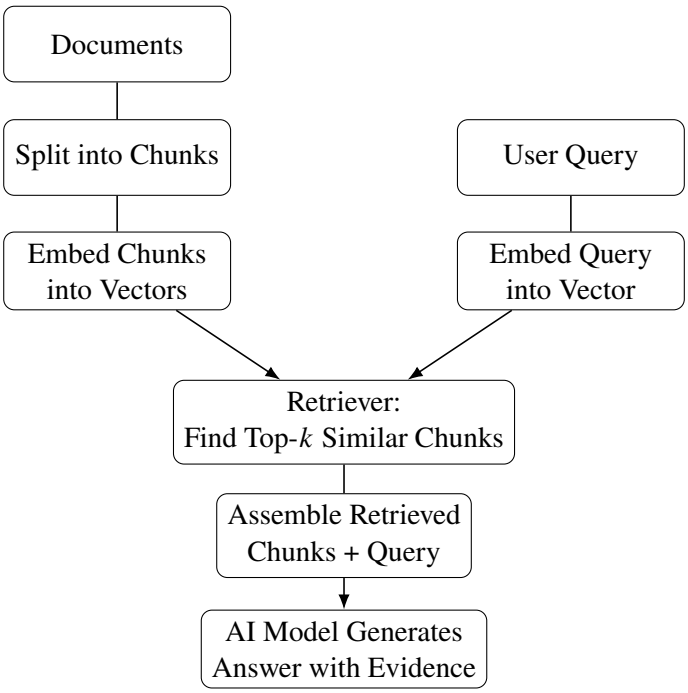


Figure 2.3: RAG Pipeline Diagram

single chunk. But, they use more tokens and might include more irrelevant details that could confuse the AI model and cause it to hallucinate [20]. Retrieving more chunks, by increasing k, makes it more likely we'll retrieve all relevant chunks (improving *recall*). However, sending too many chunks, especially if some are only loosely relevant, can dilute the important information and paradoxically lead to more hallucination, even though more information was technically provided.

The engineering emphasis thus shifts from ever-larger pre-training of models to tuning retrieval parameters carefully for the task at hand. Our evaluation, detailed in Chapter 4, treats these parameters as experimental factors and measures their interactions across factual, multi-document reasoning, and contradiction queries. By quantifying the cost–benefit curve we provide evidence-based guidance for medical RAG design.

2.4 Evaluation Strategies for Retrieval-Augmented AI Systems

The reliability of a retrieval-augmented AI assistant hinges on the integrity of each component, as failure can occur at any stage. For instance, ingesting a document might involve OCR errors transposing "0.2 mg" into "2 mg," or an embedding model could fail to distinguish between "benign cancer" and "malignant cancer." Subsequently, the similarity search might retrieve incorrect snippets, and even with accurate evidence, the AI model itself can misinterpret information or invent details. In a medical context, such slips carry severe consequences, potentially leading to adverse drug events or delayed diagnoses, which can be fatal. Therefore, evaluating these systems requires more than a single headline score; it is critical to pinpoint precisely where faults arise. This need for detailed assessment is underscored by recent surveys describing RAG evaluation as under-benchmarked and methodologically fragile, particularly in high-stakes domains [20, 46].

Current research therefore measures four interacting competencies. Retrieval quality measures how well the system surfaces relevant passages containing the requested facts, often evaluated

using metrics like *precision* (the proportion of retrieved passages that are relevant) and *recall* (the proportion of all relevant passages available that were successfully retrieved), frequently focusing these calculations on the top "k" results. Factuality, or faithfulness, then checks that each statement in the answer is entailed by those passages, guarding against hallucination. Reasoning probes the logical glue that combines multiple snippets, resolves contradictions or performs arithmetic. Multi-hop reasoning questions still expose brittleness even in cutting-edge models. Finally, *calibration* tests whether the model's declared confidence tracks its accuracy—essential when over-certainty can mislead clinicians.

Public benchmarks illuminate different slices of this space, but none cover it completely. The Factuality, Retrieval, and Reasoning Measurement Set (FRAMES) binds these three axes together, revealing how a minor retrieval lapse can cascade into a wrong conclusion [24]. The Benchmarking Information Retrieval (BEIR) benchmark offers broad retrieval tasks across twenty-one domains but stops short of checking whether answers are grounded in the retrieved sources [45]. The Automated RAG Evaluation System (ARES) adds fine-grained error provenance, labelling whether a failure stems from retrieval, grounding or reasoning [40]. MultiMedQA couples medical question-answering with expert judgements of factual harm and bias, forcing systems to justify claims in a clinical setting [43]. Together, these resources advance the field by demonstrating how retrieval and generation can be benchmarked systematically.

Scoring methodology is as important as dataset choice. Fully automated metrics can grade thousands of queries in minutes, but they infer plausibility from surface cues and struggle to verify that an answer is genuinely supported by its citations; a fluent hallucination can look correct to a pattern-matching algorithm. Human expert review remains the gold standard for clinical accuracy yet is costly and slow. The emerging compromise is a hybrid workflow: an AI model supplies a first-pass label and rationale ("AI-as-judge"), and domain experts audit a smaller stratified sample for drift and safety [56].

Our evaluation approach, detailed in Chapter 4, will evaluate the retrieval quality and factuality of our system and is inspired by existing work.

2.5 AI in Healthcare: Use Cases and Risks

Large-scale AI models have left the laboratory and are already being piloted in hospitals, clinics, and consumer apps. Within the clinical workspace, they generate first-draft radiology summaries, compose discharge letters, and answer United States Medical Licensing Exam (USMLE) questions with performance nearing that of expert physicians. The `Med-PaLM` series exemplifies this, achieving high multiple-choice accuracy while significantly reducing unsafe answers through instruction alignment [43].

On the patient side, chatbots now assist with symptom triage and medication adherence reminders; one study found ChatGPT's advice for low-acuity emergency cases met or exceeded physician recommendations [5]. Another frontier involves workflow automation, where agentic systems built on large models negotiate appointments, file authorizations, and populate order sets, promising to reclaim clinician administrative time [36]. Furthermore, early evidence suggests RAG can effectively curb ungrounded statements when summarizing EHR notes [1].

Despite these benefits, the same qualities that make these models attractive—fluency, generality, and scale—introduce four distinctive categories of risk. The first is factual fabrication: models sometimes produce confident, plausible prose that is not supported by the evidence retrieved. When a patient or junior clinician acts on such an error, responsibility becomes murky; legal scholars note that current malpractice doctrine offers no clear guidance when advice is generated by an algorithm rather than a human [49].

The second risk concerns bias and fairness. Reviews of published clinical-AI systems reveal systematic under-representation of minority groups in training data and measurable performance gaps that mirror those skews [11]. A global audit of 7,000 papers found that more than half of all clinical datasets originate from just two countries and that three-quarters of first and last authors are male, trends likely to reinforce existing inequities unless corrected through deliberate data diversification and external validation [10]. WHO therefore cautions that poorly governed AI could "automate and exacerbate existing disparities" [16].

A third challenge is privacy and consent. Unlike on-premise software, most commercial models run as cloud Application Programming Interfaces (APIs): prompts and retrieved excerpts transit the network and may be logged for service improvement. Even vector embeddings can retain latent identifiers that allow patient re-identification [26]. Analyses of real prompts show that protected health information leaks easily unless explicit redaction and encryption are enforced [41].

Finally, trust and explainability shape whether clinicians adopt or over-rely on these tools. Large-scale experiments with lay users demonstrate that detailed explanations do improve perceived understandability, yet trust can fall when the system contradicts prior belief or expresses low confidence [55]. Prompt-engineering patterns can bolster transparency, but their real-world impact on clinical decision-making remains an open question.

In short, AI models already add value in documentation retrieval, triage, and workflow support, but those benefits are inseparable from the risks of fact fabrication, bias, privacy breach, and misplaced trust. Rigorous, multi-dimensional evaluation remains a prerequisite for any future clinical deployment.

3 System Design

3.1 Goals and Requirements

This section establishes the foundation for the prototype system by defining its purpose, objectives, and limitations. The project operates as a proof-of-concept, working towards specific high-level goals and with clear constraints chosen to ensure feasibility for a single student. These are shown in Tables 3.1 and 3.2. Within these tables, short identifiers (e.g., G-1, C-1) are used for compact referencing.

Table 3.1: High-level Goals

ID	Goal	Rationale
G-1	Improve health outcomes	Faster, evidence-grounded answers help patients notice trends and ask better questions of clinicians.
G-2	Increase healthcare efficiency	Frees clinicians for higher quality patient interaction and gives patients an accessible exploration tool.
G-3	Demonstrate mastery of software engineering	Fulfil the purpose of our dissertation.

Table 3.2: Constraints

ID	Constraint	Rationale
C-1	Time: $\leq$ 80 focused hours	Keeps implementation tractable within the reasonable effort for a master's dissertation.
C-2	Deployment simplicity	Cuts operational overhead; maximizes focus on evaluation and exploring trade-offs.
C-3	Licensing: all tooling must be open or readily accessible	Removes legal friction and keeps costs low.
C-4	Cost: run on laptop with free or low-cost APIs	Ensures reproducibility for peers and lowers barriers to future adoption.
C-5	Ethics: no Institutional Review Board approval required	Avoids privacy risk and accelerates iteration.

Tables 3.3 and 3.4 articulate the system's functional requirements, expressed through user stories, and the non-functional requirements it must satisfy.

Table 3.3: User Stories

ID	User Story	Linked Goals
US-1	As a patient, I want to upload my lab reports and clinic notes so I can ask about them.	G-1, G-2
US-2	As a patient, I want to ask questions in plain language and receive concise, understandable answers.	G-1, G-2
US-3	As a patient, I want each answer to cite exactly where in my documents the information came from, so that I can verify it myself.	G-1
US-4	As a developer, I want to test how different configurations of the RAG pipeline affect system accuracy, performance, and cost so that I can improve it.	G-1, G-3
US-5	As a developer, I want to easily tweak parameters as I learn which perform best.	G-1

Table 3.4: Non-Functional Requirements

ID	Non-Functional Requirement	Rationale	Tied User Story
NFR-1	Accuracy floor: the system must correctly answer the majority of basic factual queries	A system that cannot retrieve and restate simple facts offers no health benefit.	US-2, US-4
NFR-2	Latency: 95th percentile full response <5 s	Needed for realistic user experience; long waits may invalidate efficiency claims.	US-2
NFR-3	Auditability: prompts, completions, and meta-data stored locally in human-readable database during testing	Essential for performance improvement process and debugging.	US-4
NFR-4	Modularity: easily update embedding model, AI model, and other parameters	Accelerates research iteration; future-proofs upgrades.	US-4, US-5

While this dissertation prototype demonstrates the system's core functionality, several production-critical requirements are deliberately out of scope. Their exclusion reflects a focus on feasibility for a single-student project, rather than a lack of architectural foresight. The system is designed to accommodate these future enhancements without major rework:

- Full security measures, including complete encryption of Protected Health Information (PHI) at rest and in transit, to meet the standards of the Health Insurance Portability and Accountability Act (HIPAA) and the General Data Protection Regulation (GDPR).
- Formal informed-consent workflows to ensure explicit user agreement on data usage and transmission.
- Scalable infrastructure to support multiple concurrent users beyond local-first operation.
- User-facing data deletion features to follow GDPR and California Consumer Privacy Act (CCPA) rights.
- External audit and security certification to verify data protection and safety claims.
- Comprehensive test coverage to ensure maintainability and reduce regression risks.

3.2 System Overview and Architecture

Our system architecture stems directly from the goals and constraints articulated previously, grounded in five complementary pillars: evidence traceability, local-first by default, modularity, simplicity with observability, and evaluation-driven development. These principles ensure the prototype effectively supports user decision-making (G-1) while remaining feasible (C-1), adhering to disciplined software engineering practices (G-3), and acknowledging ethical realities (C-5, NFR-3).

The system provides users a seamless workspace for interacting with their health documents. A patient can upload PDFs (illustrated in Figures 3.1 and 3.2), ask plain-language questions, and receive concise, cited answers linked back to the source passages (Figure 3.3). The core data flow, depicted in Figure 3.4, involves a document processing pipeline—extracting, normalizing, chunking, embedding text, and storing it locally with provenance metadata—and a query handling process. When a user submits a query, it is embedded, used to retrieve relevant document chunks with similarity search, combined with context into a structured prompt, sent to an AI model for answer generation, and the cited result is displayed.

Architecturally, the application is implemented as a monolithic Django process hosted locally. This choice prioritized developer familiarity (Python/Django expertise) and aligned with constraints on time, cost, and deployment simplicity, while facilitating transparent control flow and observability crucial for evaluation. Data, including text chunks and vector embeddings, is stored locally in a `SQLite` database, functioning as a simple, transparent vector store (detailed in Section 3.3).

Although designed as local-first, the prototype pragmatically utilizes external OpenAI APIs for embedding and generation due to time and deployment simplicity constraints (C-1, C-2). This introduces a privacy tension—acceptable here only due to synthetic data use (C-5)—as data leaves the local machine. However, modular adapters (NFR-4) isolate these external calls, enabling future substitution with on-premise models to ensure data sovereignty and ethical alignment before handling real patient data.

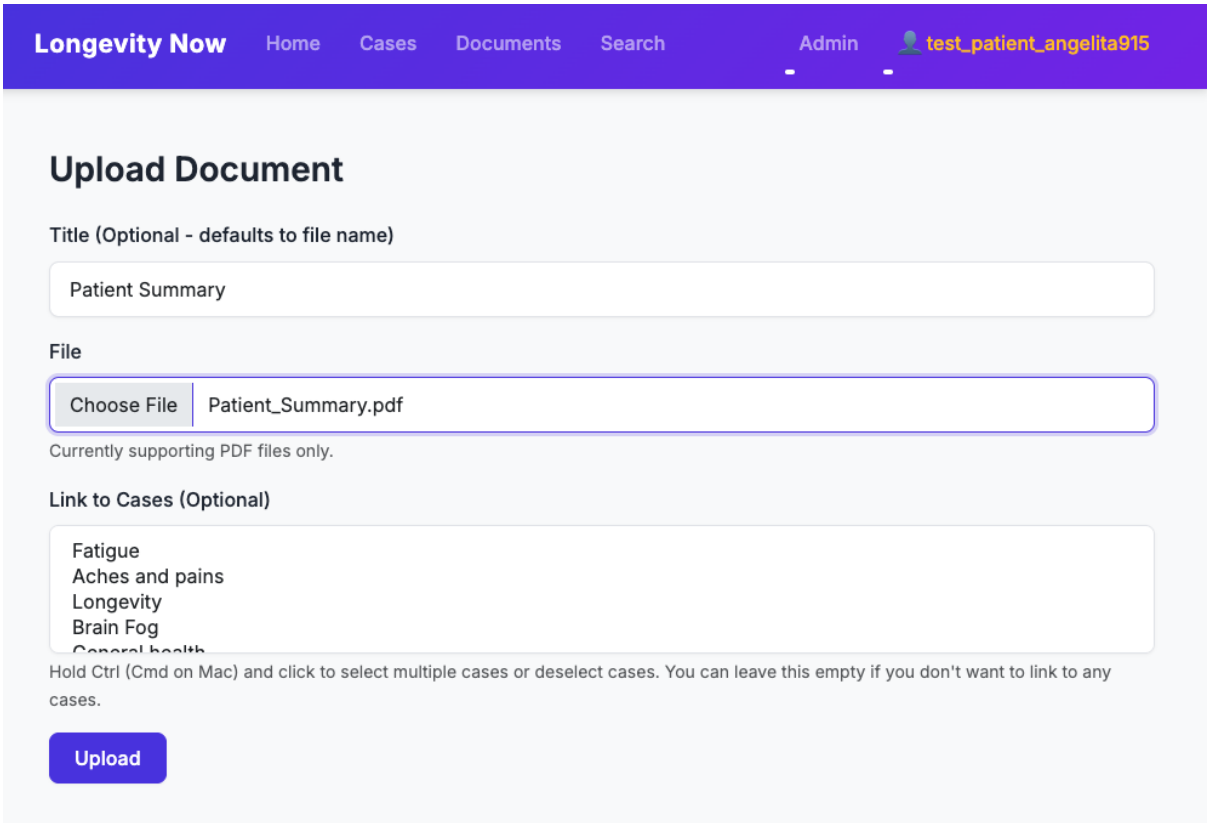


Figure 3.1: Document Upload Interface

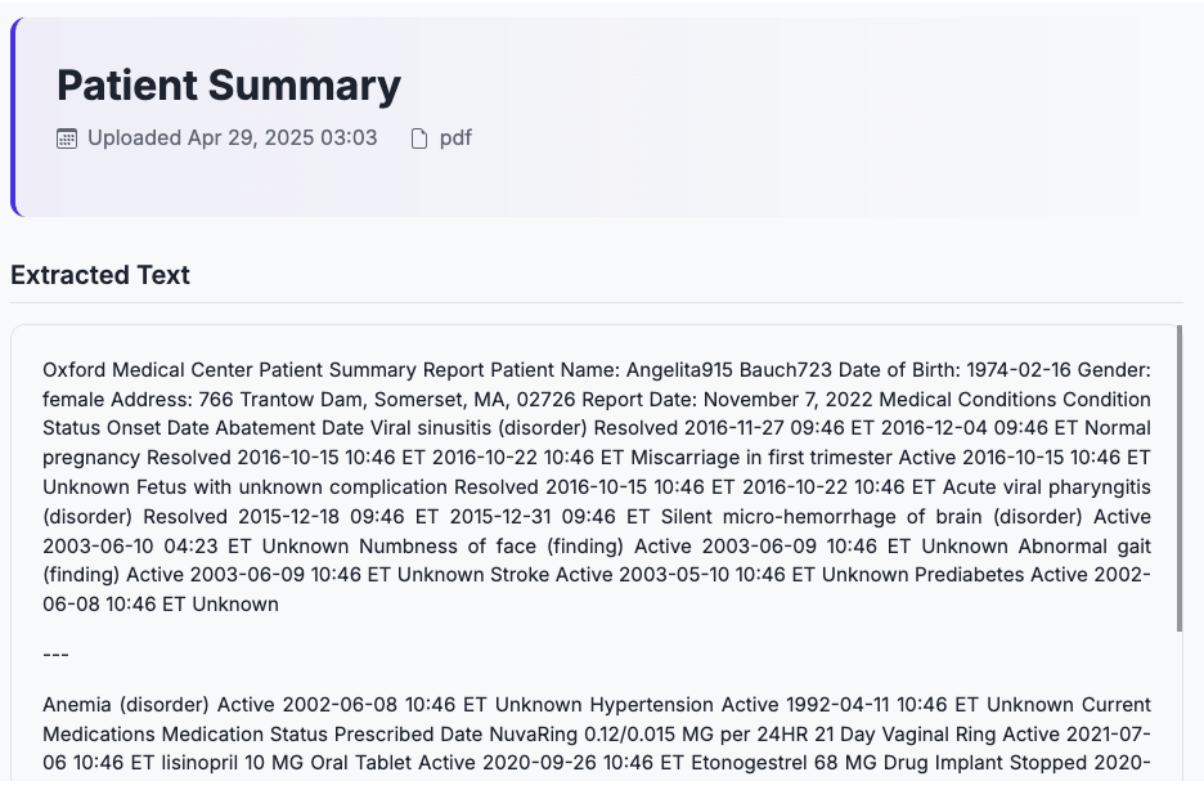


Figure 3.2: Extracted Text Shown to User

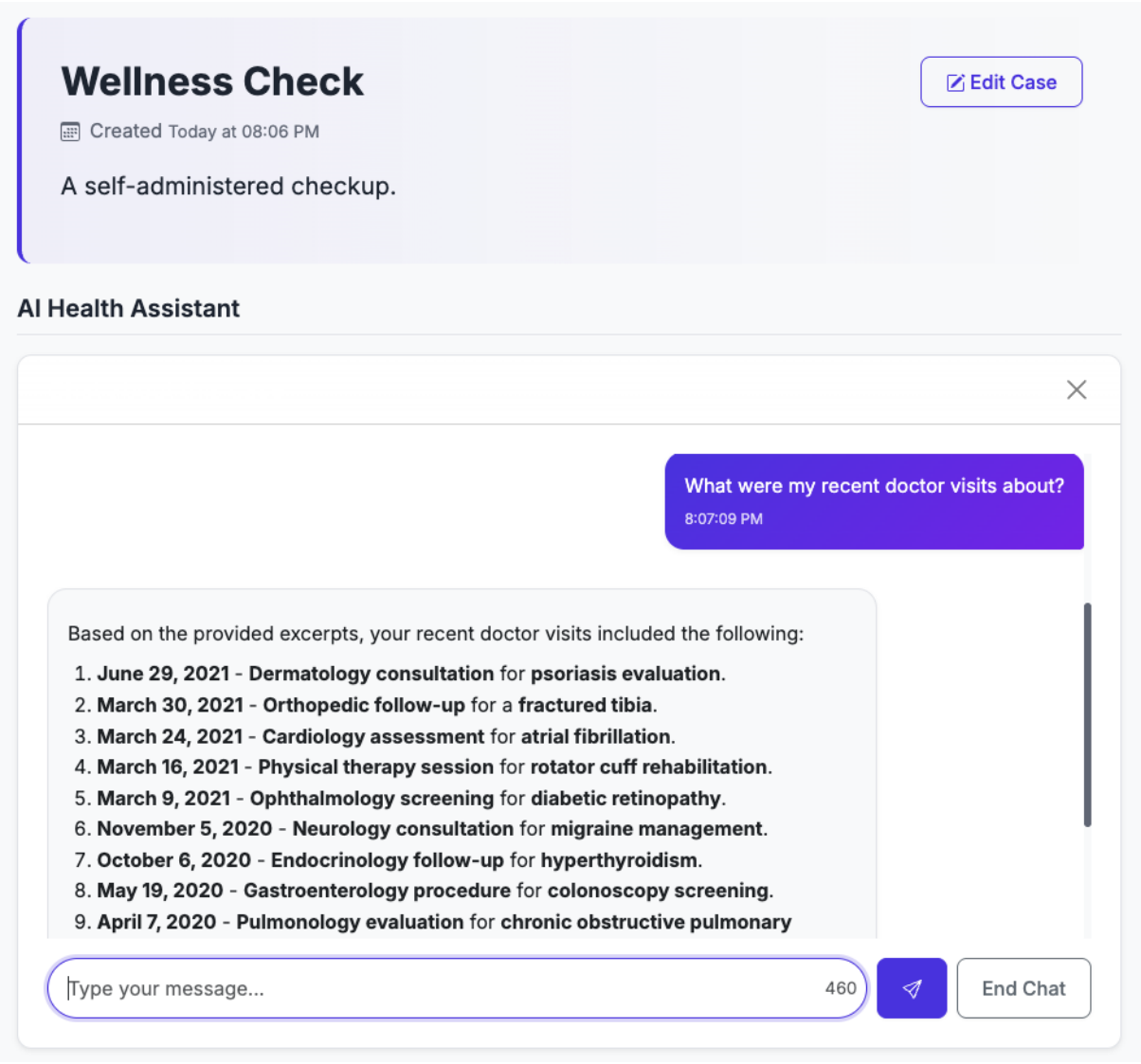


Figure 3.3: AI Chat Interface

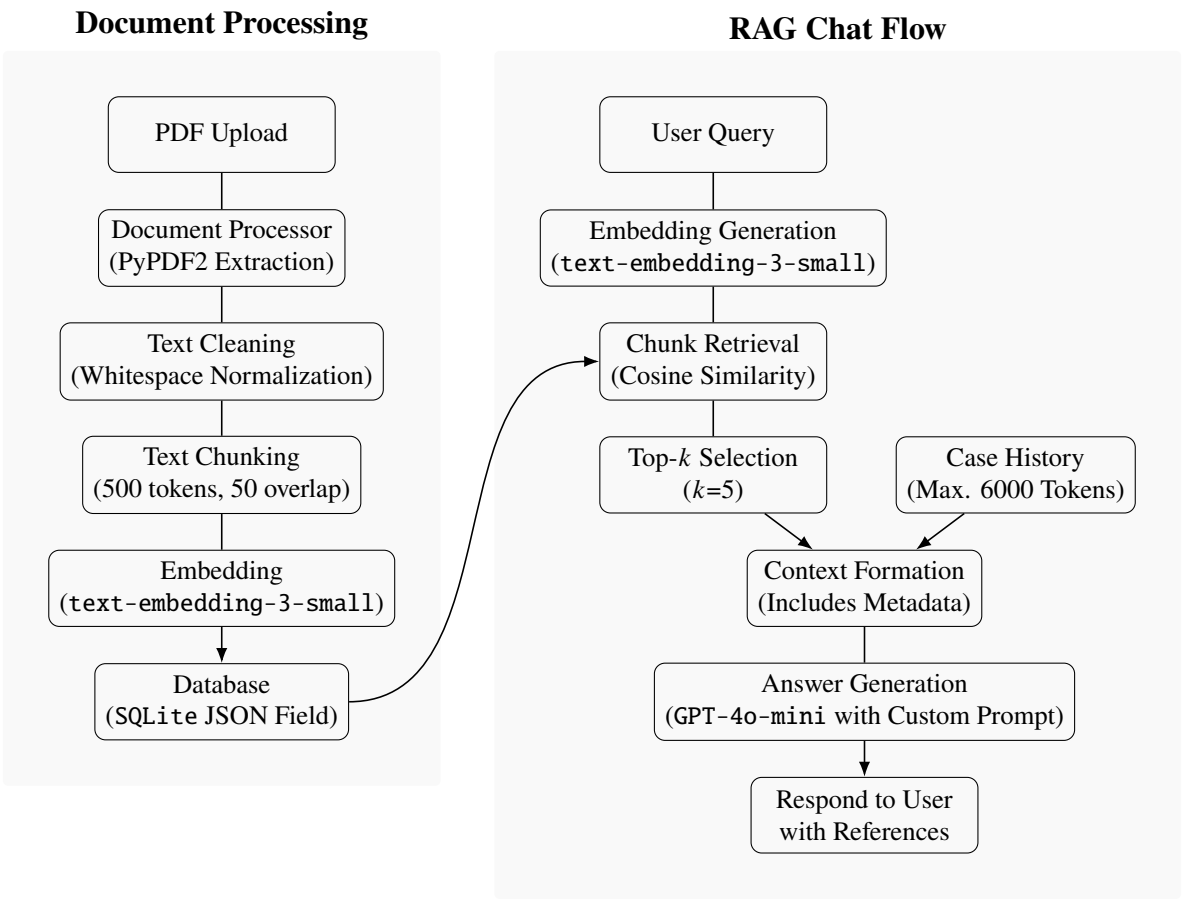


Figure 3.4: RAG Chat Pipeline with Document Processing and Query Flow

The architecture is explicitly designed to support rigorous evaluation, aligning with the evaluation-driven development pillar. During operation, key artifacts from the RAG exchange—retrieved chunk IDs, token counts, latencies, prompts, completions—are logged for offline analysis. Furthermore, dedicated administrator tools, integrated as a separate Django app reusing core code paths, facilitate parameter grid searches and failure-mode annotation, allowing quantification of accuracy, cost, and risk. These are detailed and illustrated in Section 3.6.

We acknowledge the inherent design trade-offs. The local, monolithic architecture limits horizontal scalability but is appropriate for this single-user proof-of-concept. Using cloud APIs creates a potential privacy exposure surface but significantly accelerates research iteration within budget constraints (C-4). Crucially, the modular design—incorporating abstraction layers and pluggable components—provides the necessary seams for the system to evolve safely towards multi-user support, hardened PHI governance, or enterprise scale, as discussed further in Chapter 6.

3.3 Document Ingestion and Embedding

We designed the ingestion pipeline to convert heterogeneous PDFs into semantically indexed, provenance-rich chunks with minimal manual intervention and external dependency. The process unfolds inside a single request–response loop so that each newly uploaded document is searchable within seconds and every transformation step remains inspectable. This supports our pillars of traceability, simplicity, and evaluation-driven development.

We first extract text. When the user uploads a PDF, a Django signal triggers a `PyPDF2` parser that iterates page by page, concatenating textual content while stripping headers, footers, and duplicated whitespace. More sophisticated libraries such as `pdfplumber` handle mixed-layout reports, but on our synthetic document corpus `PyPDF2` proved sufficient and yielded clean offsets we could map back to originals. This choice satisfies our deployment simplicity constraint (C-2) and respects the eighty-hour project limit (C-1). Should future deployments require scanning images or OCR artefacts, we can swap this module without touching downstream code—a deliberate consequence of keeping ingestion stateless and side-effect free, aligned with our modularity goal (NFR-4).

After extraction, we normalize characters and collapse Unicode variants. This is primarily to improve the display of the extracted text to the user. We considered passing the text through a lightweight synonym harmoniser seeded with high-frequency terms from the Unified Medical Language System (UMLS) but we decided against it. We chose to rely on the AI model's inherent ability to recognize medical synonyms, thereby avoiding unnecessary pipeline complexity for this proof-of-concept and remaining within our development-time constraint (C-1).

Chunking follows. We slide a 500-token window with a 50-token overlap, where token counts are measured with `tiktoken`, a library by OpenAI for tokenizing text, to stay consistent with their generation APIs. Fixed chunks are preferred to sentence- or semantic-based splits in our context: they guarantee deterministic boundaries, reproducible recall, and O(1) memory. Overlap mitigates boundary loss by ensuring that a lab value straddling two pages remains intact at least once. Well-tuned fixed chunks rival more complex segmenters on downstream accuracy, while avoiding their brittleness on noisy input, as discussed in Section 2.3. Our chunking decision thus balances retrieval precision (NFR-1) against latency (NFR-2) and prompt economy constraints (C-4). Our evaluation in Chapter 4 quantifies the trade-off.

For each chunk we write a provenance record: document and user IDs, upload timestamp, and any associated case. This metadata travels with the chunk through retrieval and appears verbatim in prompts, giving users an audit trail from answer back to chat window—an explicit

answer to WHO's call for explainable clinical AI [16].

We then embed the chunk with OpenAI's embeddings API and the `text-embedding-3-small` model. We selected this model because it delivers strong performance at one-fifth the token cost of its predecessor [32], balancing performance and efficiency. It also shares a cosine space with OpenAI's `GPT-4o-mini`, avoiding embedding misalignment. Domain-specific open source alternatives such as `BioBERT` or `ClinicalBERT` could improve domain-specific performance, but would require adequate hardware to run locally. Fine-tuning the models may improve performance but may require specialized hardware (e.g. GPUs) and would introduce development complexity. Our development-simplicity constraint and the small corpus size tip the balance toward a hosted, metered model that is simple to integrate and we can swap later with a configuration flag, as illustrated in Figure 3.5. Each embedding is a 1,536-dimensional vector; we persist it uncompressed in a `SQLite` JSON column next to raw text so that a single `SELECT` returns everything needed for similarity scoring or human inspection.

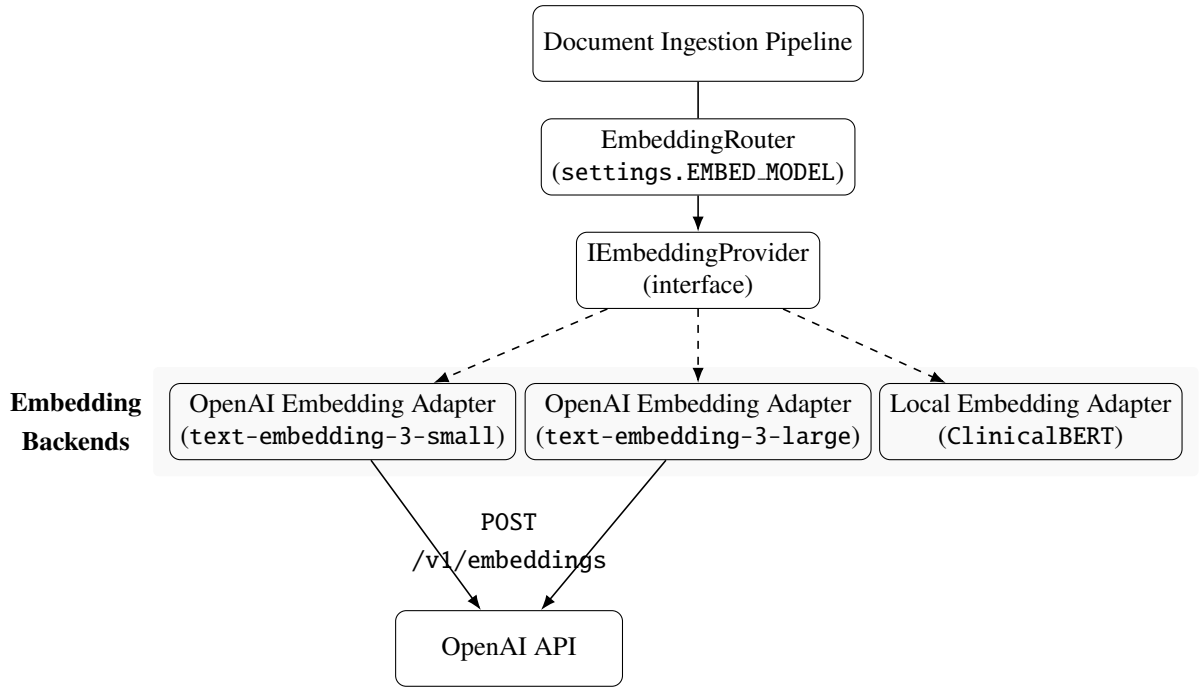


Figure 3.5: Embedding Service Abstraction

Storing vectors in `SQLite` prioritized development simplicity (easier to set up and manage for a single developer) (C-2, C-1) and traceability (keeping text and its vector representation together) (NFR-3) over achieving the absolute fastest retrieval speed. Specialized ANN indices (like Facebook AI Similarity Search (FAISS) or those in dedicated vector databases) provide faster vector search, especially with millions of vectors. However, integrating and managing them adds complexity. Given our relatively small number of documents (thousands of chunks, not millions) and the fact that the time taken by the AI model to generate the answer was much longer than the vector search time, the extra speed from an ANN index wasn't deemed necessary for this prototype. Using `SQLite` allowed us to use standard database queries (via Django's Object-Relational Mapping (ORM)) for analysis during evaluation (US-4), making the process more transparent and manageable within the project's scope and constraints (C-1, C-4). The system is designed modularly, so `SQLite` could be replaced with a faster ANN index later if needed for a much larger scale.

Since embedding is performed by an external provider, the chunk content is exposed. This is acceptable in this prototype without additional consideration because we use only synthetic documents. In a production setting, this architecture would require a different data governance strategy: encryption of embeddings at rest, role-based access control, and ideally, on-premise or self-hosted embedding models to ensure that no PHI is transmitted externally. This limitation is discussed further in Chapter 5. Nevertheless, the current implementation preserves a clean interface boundary allowing future deployments to substitute the embedding backend without affecting upstream logic.

By the end of ingestion, every document has been reduced to a set of immutable, semantically indexed, and fully traceable chunks. This representation underwrites the precision-latency experiments in Chapter 4 and the transparency commitments analysed in Chapter 5, while keeping the code path short enough to reason about—a design victory for maintainability and ethical assurance.

3.4 AI Model Integration

The final stage of the pipeline transforms retrieved evidence into a conversational answer using an external AI model. To ensure modularity (NFR-4) and support easy model swapping (US-5)—similar to the embedding model—the integration logic is encapsulated within a dedicated adapter. Currently, this adapter integrates OpenAI's `GPT-4o-mini` using their Python SDK. This transparent design allows for future replacement of the AI model or adjustments to prompting strategies without requiring invasive changes to the core application code, as illustrated in Figure 3.6.

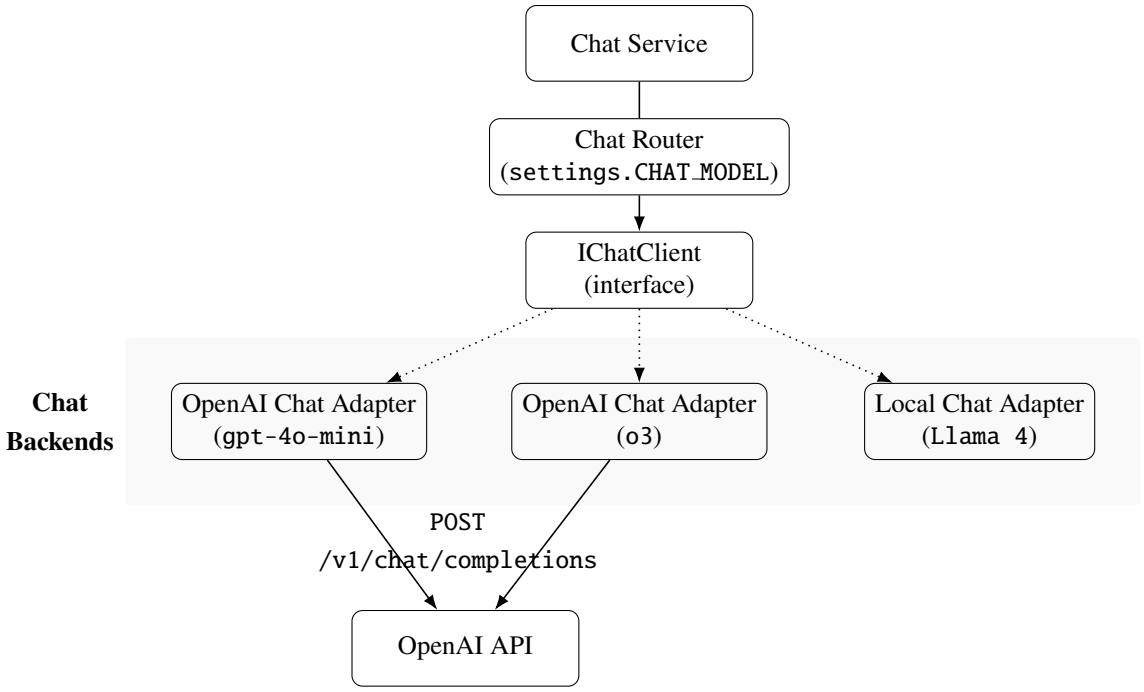


Figure 3.6: Chat Service Abstraction

At inference time, the system constructs a deterministic prompt composed of four sequential components. First is a safety-first system message that defines role, scope, emergency protocols, and mandates epistemic humility. Second, case logs submitted by the user, enclosed in Extensible Markup Language (XML) tags. Third, labeled document chunks that include document title

and upload date. Finally, the user's natural-language query. This structured layout is informed by the primacy–recency effects described in Section 2.2, ensuring that the most important parts—the safety-first system prompt and user query—receive optimal model attention. We avoid few-shot examples or chain-of-thought prompting to reduce token count and latency (NFR-2) but investigating their potential accuracy improvements would be worthwhile.

The system prompt, shown in Appendix C, is a critical anchor for model behaviour, designed across three dimensions. First, it emphasises factual grounding, instructing the model to quote document excerpts verbatim and avoid fabrication. Second, it acknowledges uncertainty, requiring the model to highlight when evidence is incomplete or contradictory. Third, it encodes safety-critical guidance, such as advising immediate emergency care in response to time-sensitive issues such as chest pain or stroke symptoms. These directives are consistent with WHO's principles of transparency and responsibility [16], and with best practices for uncertainty calibration in question-answering systems [21].

The `GPT-4o-mini` model was chosen for its cost and latency profile. It offers approximately 6% of the token cost of its larger sibling, `GPT-4o` [33] while maintaining sub-five-second response times in our use case (shown in Section 4.4). This choice satisfies our project cost constraint (C-4) while allowing practical experimentation. Our evaluation in Section 4.4 confirms that while the model performs well for basic tasks, it struggles with tasks that require reasoning, an anticipated limitation given our avoidance of chain-of-thought and plan-and-solve prompting [50, 51]. Since we expose the model through an adapter, replacing it with a more reasoning-capable model to enable future investigation would be simple.

To prioritize deterministic and repeatable outputs, we control the model's stochasticity using the API's provided temperature parameter. Empirically, we found that a setting of 0.2 balances output consistency with sufficient expressiveness. This promotes reproducibility across sessions and enables reliable evaluation of correctness and hallucination. The model's answers are returned verbatim to the browser, preserving inline citations that reference document titles and upload dates, allowing users to trace responses back to their source context.

Nevertheless, this design has clear limitations. Because the system relies on a hosted API, prompt content must leave the local device. While this study uses only synthetic documents, we explore PHI-safe deployment pathways in Section 5.2. The system currently lacks post-processing for citation alignment and source drift detection—capabilities earmarked for future development alongside richer prompt templates and confidence estimation. However, the modular adapter architecture ensures that such enhancements will remain localized, requiring no changes elsewhere in the stack.

In sum, the current AI model integration strikes a pragmatic balance between speed, cost, and traceability. It satisfies our time (C-1), cost (C-4), and development-simplicity (C-2) constraints, while preserving extensibility through modular design (NFR-4).

3.5 Retrieval-Augmented Generation

The retrieval stage bridges static embeddings and live reasoning. Upon each query we embed the user text with the same OpenAI `text-embedding-3-small` encoder used at ingestion and compute cosine similarity using the numerical computing package, `NumPy`. Because our corpus never exceeds ten thousand chunks, we can do a brute-force scan over the entire `SQLite` JSON column. This is guaranteed to retrieve the text chunks with the highest similarity. It also completes in a reasonable time, with the entire retrieval process completing in under 1 second in most cases tested in our evaluation (details in Section 4.4). This dense retrieval is transparent and easily inspected, while avoiding the infrastructure overhead of an ANN index

like FAISS and Hierarchical Navigable Small Worlds (HNSW). Should document volumes grow, the retrieval module is encapsulated behind a single interface, allowing us to swap in an ANN backend without touching ingestion, embedding, or prompt logic. This design choice directly supports our pillars of traceability and evaluation-driven development, and satisfies the auditability requirement (NFR-3).

We rank chunks exclusively at the chunk level, as opposed to the document level, and take the top five hits. Chunks scoring below an empirically tuned threshold are discarded so that the prompt contains evidence, not noise. The surviving chunks are injected into the prompt in descending similarity order. Placing the most relevant passage first exploits the primacy bias described earlier.

Users see the final answer, and referenced documents, but not the raw context. This reflects a conscious prioritization of usability over completeness for the prototype stage, while preserving the ability to satisfy transparency requirements in future deployments. In the RAG Evaluation subsystem, all retrieved IDs, scores, and token counts are logged, enabling retrieval precision and latency dashboards without bloating the production database. We omit logging in the live chat to minimize storage of potentially sensitive text captured during testing, a pragmatic compromise until more privacy and security safeguards are in place.

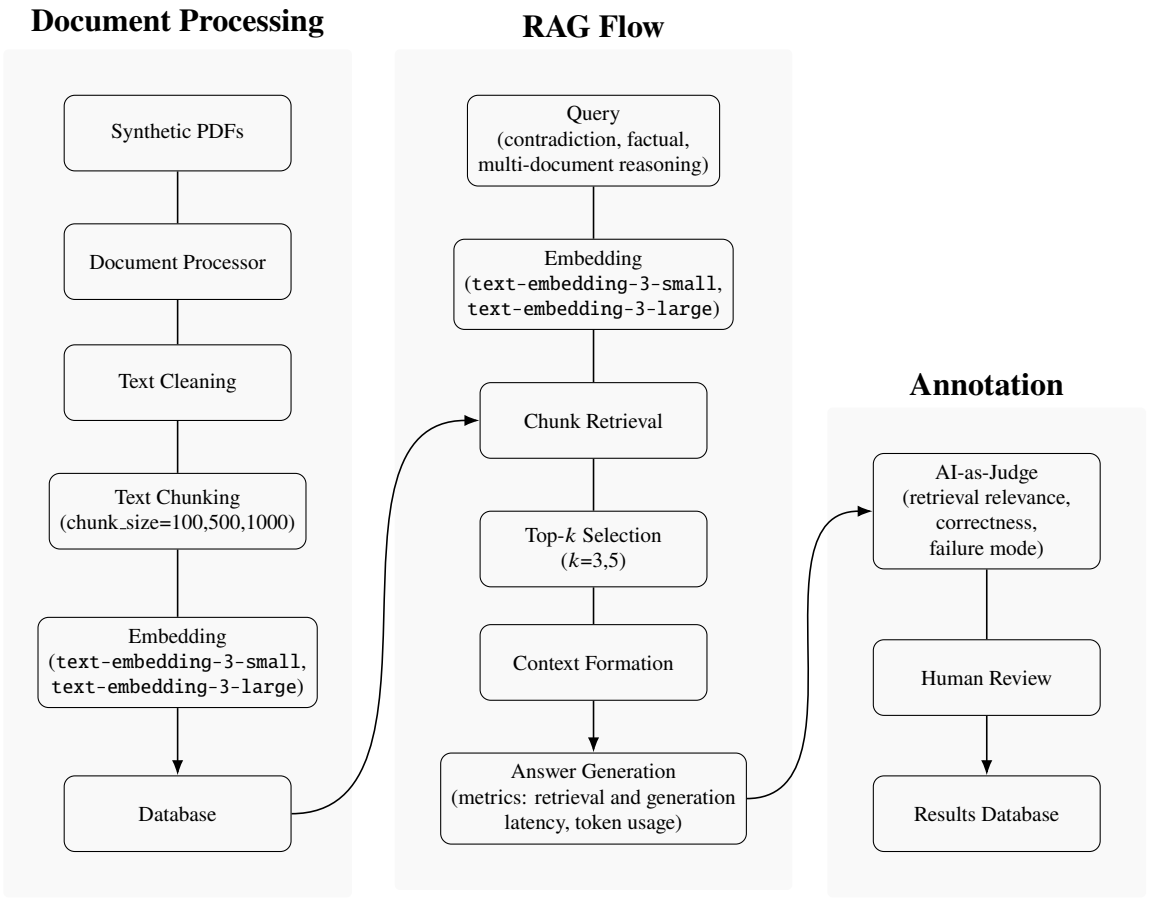
While this design delivers high precision on small corpora for certain query types, it presents limitations that scope future enhancements. The fixed chunk size, for instance, risks splitting semantically linked facts across boundaries if they span more than the 50-token overlap. Furthermore, the reliance on dense vector similarity alone means that fixed top-k retrieval may overlook relevant chunks containing rare but crucial keywords, and the pure similarity ranking ignores potentially valuable contextual cues like temporality.

To address these specific limitations, future work should explore hybrid sparse-dense retrieval techniques to capture both semantic similarity and exact keyword matches, potentially improving recall for evidence containing low-frequency terms. Implementing re-ranking of the top-k candidates using more computationally intensive query-chunk cross-encoders could increase precision by better identifying the most relevant chunks before synthesis, mitigating noise from less relevant retrieved passages. Finally, to enhance transparency and user trust—addressing the inherent risk of model misinterpretation—exposing the retrieved context directly to the user would allow for real-time verification of the generated answers against their sources. For now, however, the current pipeline provides a reproducible and inspectable baseline that effectively supports the evaluation in Chapter 4 and the ethical commitments of Chapter 5, meeting our development constraints (C-2) while establishing a foundation for these future improvements. A high-level discussion about scaling past the prototype is provided in Section 6.5.

3.6 RAG Evaluation System

To systematically expose both the strengths and limitations of our RAG system, we developed a dedicated evaluation framework. Implemented alongside the main application but operating exclusively on synthetic patient data, this framework allows for rigorous testing without compromising user privacy or interfering with the live environment. This isolation prevents debug artefacts from leaking into user sessions and preserves a clean separation of concerns. Crucially, the evaluation system utilizes the same core pipeline code as the live chat (Figure 3.7), ensuring that testing exposes realistic performance bottlenecks.

The framework supports parameter sweeps, detailed logging, and a human-in-the-loop annotation process, aligning with our principles of evaluation-driven development, traceability, and reproducibility. Furthermore, it operationalizes our ethical stance: rigorous measurement



Note: This flow is similar to the user-facing chat pipeline in Figure 3.4, with the addition of results annotation.

Figure 3.7: RAG Evaluation System Diagram

precedes any consideration of deployment, and every metric is stored with sufficient context for thorough inspection.

An evaluation run is initiated through an administrative interface presenting a grid-search form, shown in Figure 3.8. This allows selection of parameters for testing, including chunk sizes (100, 500, 1000 tokens), retrieval depths k (1, 3, 5), and embedding models (`text-embedding-3-small` and `-3-large`). The questions used were manually authored after an attempt to automatically generate question-answer pairs with AI proved insufficient for complex multi-document and contradiction scenarios. While AI could generate simple factual queries, it failed on more demanding types, foreshadowing some limitations explored in Chapter 4. This manual authoring prioritized evaluation validity and a high-quality error taxonomy over sheer query volume.

Upon initiation, the system clones the patient's vector store for the chosen configuration, executes the full RAG pipeline for each question, and persists detailed logs. These logs include the raw prompt text, the model's completion, retrieved chunk IDs and text, token counts, retrieval and generation latencies, and timestamps. Given the use of synthetic data, full text is logged unredacted to maximize auditability (NFR-3), consistent with our ethical constraints (C-5). Handling real PHI would necessitate hashing or encryption, as discussed in Section 5.2.

To assess performance, each generated answer is first automatically labelled by an AI model judge (`GPT-4o-mini`) based on retrieval relevance, answer correctness, and failure mode. The judge also provides a textual justification for its scores. Subsequently, these automated annotations are reviewed by a human evaluator using an admin interface, shown in Figure 3.9, that displays the retrieved evidence alongside the model's response. This human-in-the-loop step is crucial for mitigating potential AI miscalibration and enforcing accountability standards.

All annotated results, including performance metrics and failure modes, are stored persistently, enabling downstream analysis in tools like Jupyter notebooks without re-running experiments. While the application includes lightweight dashboards for immediate feedback (illustrated by Figure 3.10), final statistical tests and visualizations for this dissertation were performed offline for faster iteration and select plots are provided in Section 4.4. Integrating these richer visualizations back into the web User Interface (UI) would enable maintainers to make their decisions directly in the application in support of *US-4*, but remains a task for future development.

Ultimately, this dedicated evaluation system provides more than just performance metrics; it serves as an extensible laboratory for reliably investigating RAG behaviour in our specific context. It underwrites the quantitative claims made in Chapter 4 and offers a template for ongoing model governance and risk monitoring as system components—such as more advanced, reasoning-capable AI models—are integrated in the future.

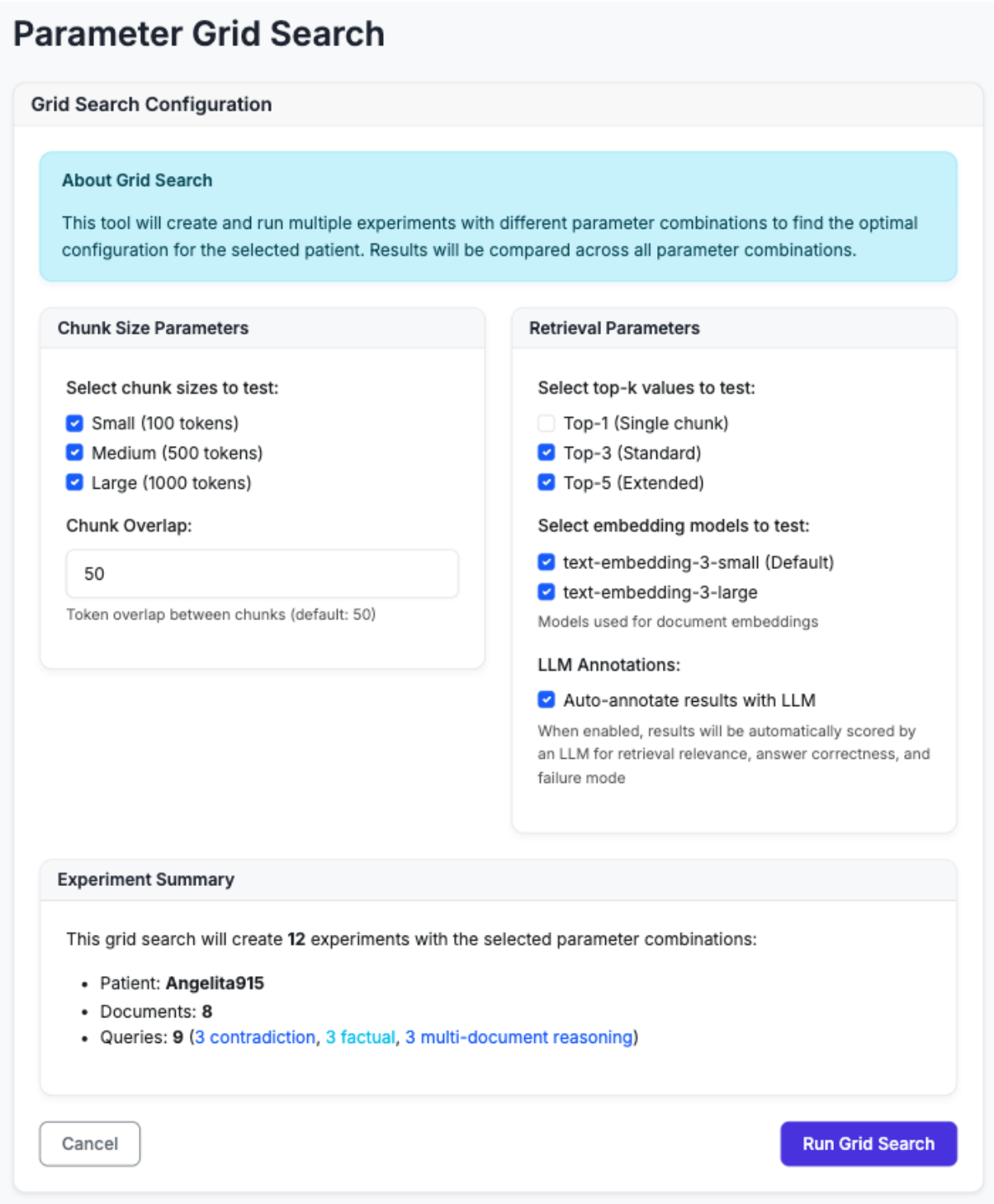


Figure 3.8: Parameter Grid Search User Interface

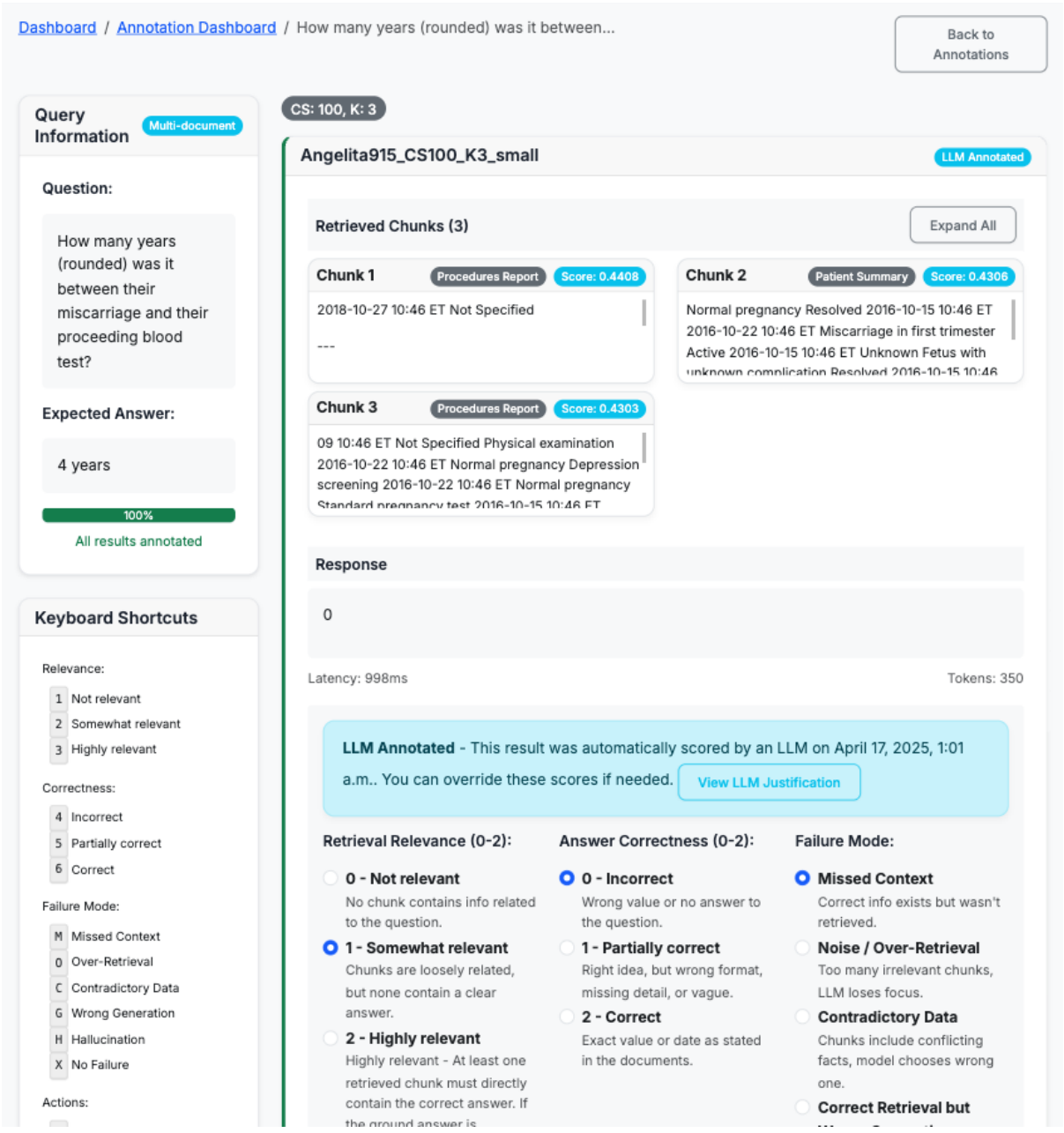


Figure 3.9: Annotation Tool User Interface

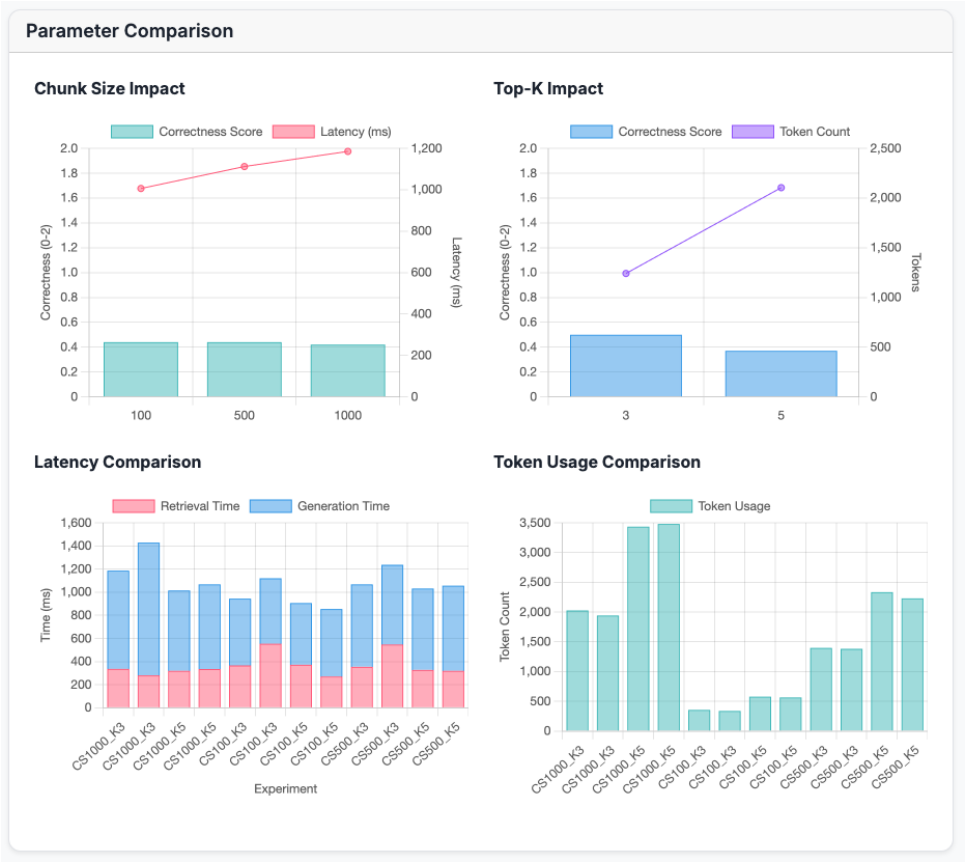


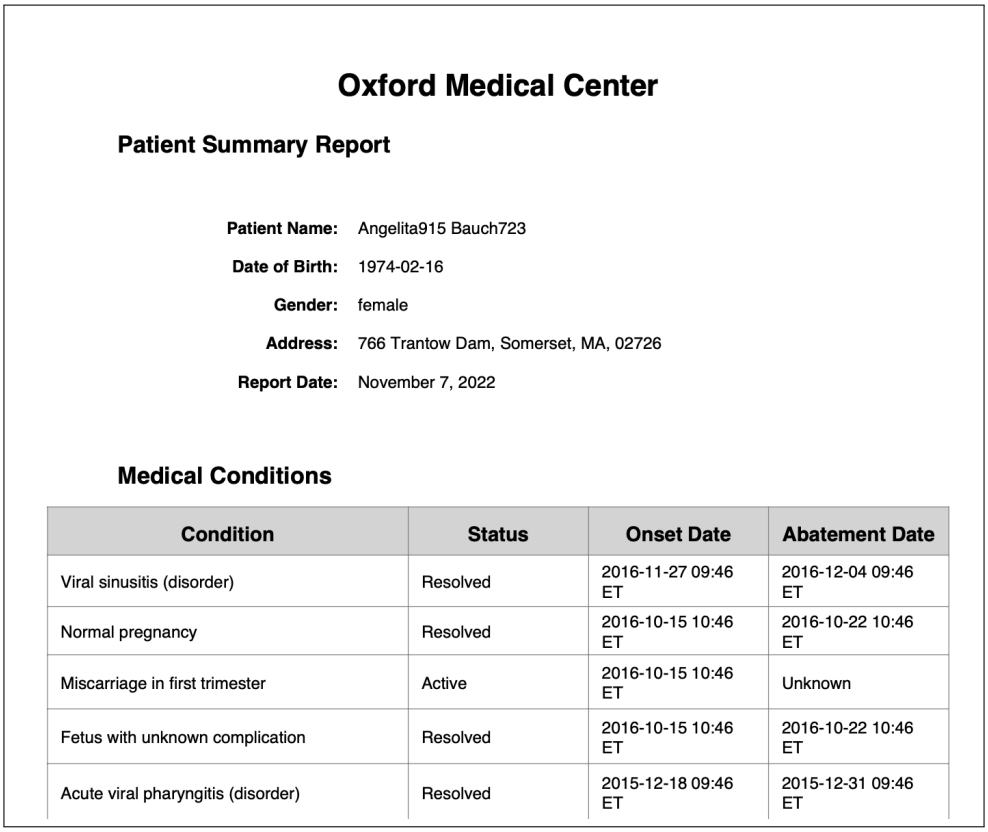
Figure 3.10: Sample Metrics Dashboard (superceded by Jupyter Notebook analysis)

4 Evaluation

4.1 Experimental Setup

This chapter evaluates how core retrieval parameters influence the accuracy and efficiency of the RAG system described in Chapter 3 when applied to fragmented personal health records. The evaluation employs a controlled testbed to isolate the effects of three independent variables: chunk size, retrieval depth (k), and embedding model.

The experimental corpus consists of synthetic health records for five distinct patients, generated using Synthea, an open-source tool that generates realistic, synthetic patient health records [48]. We wrote a standalone Python script to transform the synthetic data into PDFs to mimic common clinical document types (e.g., encounter notes, lab reports). An example is shown in Figure 4.1 and more examples are in Appendix E. The document token size ranged from 118 to 7872 tokens, with a median token count of 477 and a 90th percentile token count of 1939. Synthetic data eliminates privacy risks, the need for ethical review, and provides known ground truth. However, it lacks certain real-world noise like OCR errors, limiting ecological validity.



Oxford Medical Center

Patient Summary Report

Patient Name: Angelita915 Bauch723
Date of Birth: 1974-02-16
Gender: female
Address: 766 Trantow Dam, Somerset, MA, 02726
Report Date: November 7, 2022

Medical Conditions

Condition	Status	Onset Date	Abatement Date
Viral sinusitis (disorder)	Resolved	2016-11-27 09:46 ET	2016-12-04 09:46 ET
Normal pregnancy	Resolved	2016-10-15 10:46 ET	2016-10-22 10:46 ET
Miscarriage in first trimester	Active	2016-10-15 10:46 ET	Unknown
Fetus with unknown complication	Resolved	2016-10-15 10:46 ET	2016-10-22 10:46 ET
Acute viral pharyngitis (disorder)	Resolved	2015-12-18 09:46 ET	2015-12-31 09:46 ET

Figure 4.1: Example Synthetic PDF

Against this corpus, we tested nine manually authored natural-language queries per patient. These queries were designed to represent three distinct types: simple factual look-ups, multi-document reasoning (or "*multi-hop*" or "*synthesis*") tasks, and contradiction detection questions (requiring the system to identify contradictions that are dynamically injected at runtime for this question type). Examples are shown in Figure 4.2. Manual authoring ensured query clarity and the availability of precise reference answers for reliable scoring.

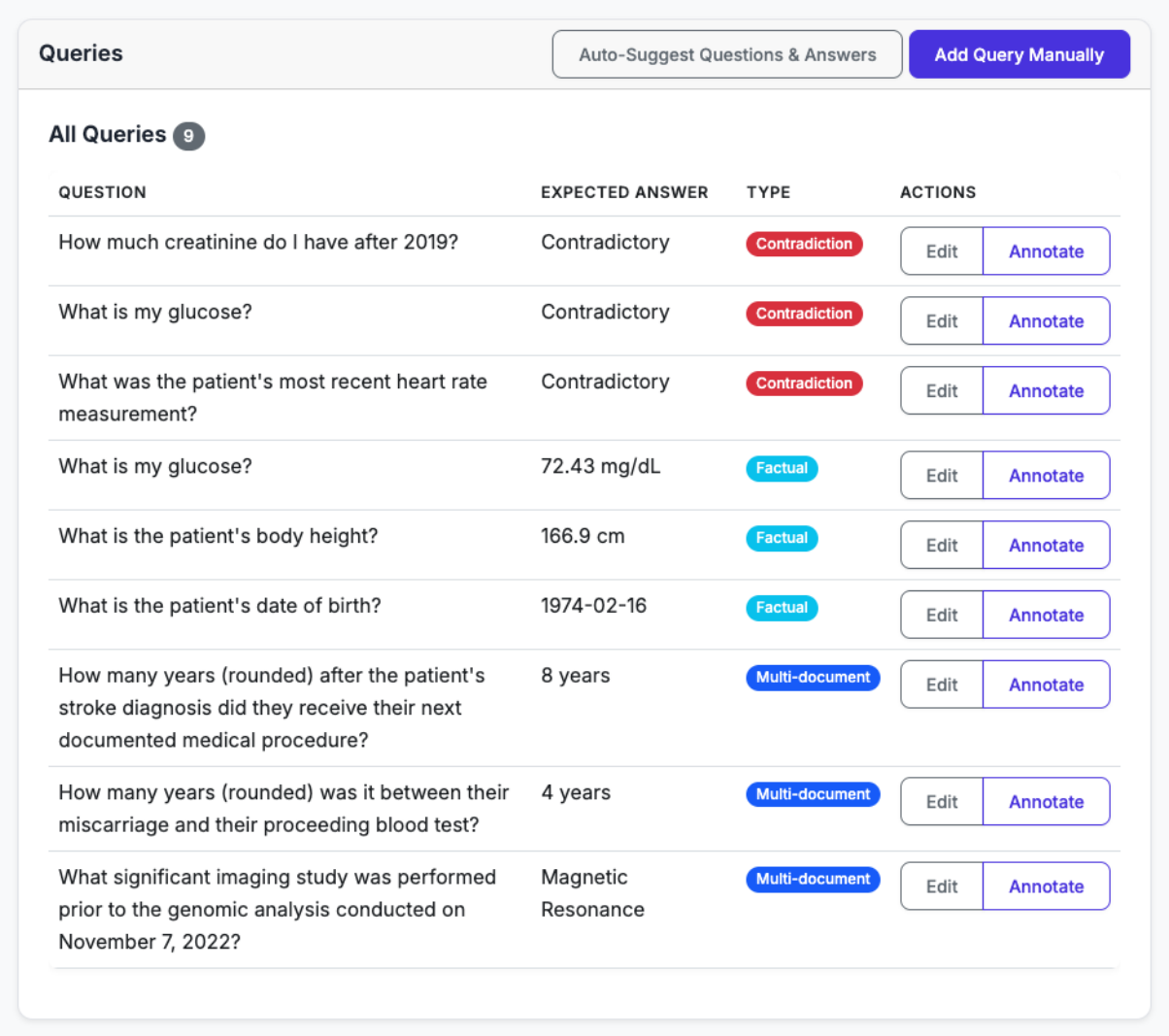


Figure 4.2: Query Creation User Interface

We employed a full factorial experimental design, systematically varying the independent variables across their chosen levels:

- Chunk Sizes: 100, 500, and 1000 tokens (each with 50-token overlap).
- Top-k Retrieval Depths: 3 and 5 chunks.
- Embedding Models: OpenAI's `text-embedding-3-small` and `text-embedding-3-large`.

The rationale for selecting these parameters and their anticipated trade-offs were discussed in Sections 2.3 and 3.3. Each of the 9 queries per patient was executed under all 12 resulting configurations ($3 \times 2 \times 2$), yielding 540 independent query-configuration pairs for analysis (5

patients × 9 queries × 12 configurations). Each run was stateless, without reusing conversation history, and utilized deterministic decoding (temperature=0.2) from the `GPT-4o-mini` model. Experiments execute on a 16 GB M1 MacBook Pro, which mirrors the resource envelope of a clinician's laptop rather than a GPU cluster.

The underlying system architecture (local Django application, `SQLite` storage, specific libraries) and the evaluation framework (parameter sweep orchestration, detailed logging, AI-assisted annotation with human review) used to execute these runs and collect data are described in Sections 3.2 and 3.6 respectively. The resulting dataset, capturing performance metrics and failure modes for each run, forms the basis for the quantitative and qualitative analyses presented in the subsequent sections of this chapter.

4.2 Metrics and Evaluation Methodology

To assess the RAG system holistically, we defined a multidimensional evaluation that spans the fidelity of retrieved evidence, the accuracy of generated responses, resource efficiency, responsiveness, and qualitative failure characterization. This approach ensures that we not only measure whether the system can locate and present relevant information, but also how effectively it uses that information to produce correct answers under realistic constraints.

To measure how well the system retrieved relevant information, we used a metric related to *precision@k*. Specifically, we evaluated the top k retrieved chunks using a graded 0–2 scale, rather than just a simple 'yes/no' for relevance or evaluating every chunk individually. Scores are defined as:

- 0 ("Not relevant"): No chunk contains information related to the question.
- 1 ("Somewhat relevant"): Retrieved chunks are topically related but none contains a clear answer.
- 2 ("Highly relevant"): At least one chunk explicitly states the correct information. Contradiction queries require both conflicting chunks for a score of 2.

This graded approach balances annotation effort against interpretive depth: it distinguishes partial recalls (score 1) from complete, direct retrievals (score 2), guiding our understanding of how chunk size and retrieval depth trade off noise against coverage.

Response correctness follows a similar 0–2 rubric, reflecting faithfulness to the retrieved evidence:

- 0 ("Incorrect"): Model output is wrong or unrelated.
- 1 ("Partially correct"): The response indicates the right idea but is vague, formatted incorrectly, or omits key details.
- 2 ("Correct"): The answer matches exactly the value or fact as stated in the documents.

By separating retrieval relevance from generation correctness, we can identify whether errors stem from missing evidence or from the model's inability to synthesize retrieved context—a distinction critical for prioritizing improvements in the retriever versus the AI model. The evaluation system prompt includes special instructions to answer only using the specific data point requested, answer "Unknown" if the context doesn't include relevant information, and "Contradictory" if the context includes contradictory information.

We counted token usage for each query—prompt tokens (including query, instructions, and retrieved chunks) plus completion tokens—using the `tiktoken` library. Tracking tokens illuminates the practical impacts of design choices: larger chunks or higher k inflate prompts,

and hence API costs, but may not yield proportional gains in accuracy, as observed in our quantitative findings.

Latency was measured end-to-end and disaggregated into retrieval and generation components. Retrieval latency encompasses query embedding, cosine-similarity scoring across the `SQLite`-stored vectors, and top-k selection. Generation latency covers the AI model call and decoding time. This split enables targeted optimizations—for example, increasing chunk size reduces the number of cosine computations but expands generation prompt length, yielding opposing effects on each latency component.

Beyond these numeric metrics, we annotated each query–response pair with a single failure mode from a predefined taxonomy, chosen for its clarity and mutual exclusivity:

- `missed_context`: Correct evidence exists but was not retrieved.
- `contradictory_data`: Retrieved chunks conflict and the model selects the wrong fact.
- `wrong_generation`: Retrieval is accurate but the model misinterprets the evidence (similar to hallucination or "fact fabrication").
- `none`: No observable failure; the response is fully correct.

This taxonomy provides actionable insights into where the pipeline breaks down—whether in retrieval coverage, prompt clarity, or model reasoning.

Annotations were generated in two stages. First, an AI model judge assessed retrieval relevance and response correctness, providing justification notes alongside each label. We then performed manual review of all labels to confirm the validity of the failure-mode assignment. While AI-based pre-labelling accelerated the process, manual oversight was essential to uphold annotation quality.

4.3 Qualitative Case Studies

To complement the aggregated metrics, we examine three representative interactions that illustrate the strengths and weaknesses of our RAG pipeline in real-world–style scenarios. These case studies highlight ideal behaviour, over-retrieval leading to hallucination, and failure to flag contradictory evidence despite clear instructions—each underlining specific design trade-offs and failure modes.

The ideal retrieval scenario is represented by a factual query—"What is my glucose?"—for patient `angelita915` using 100-token chunks, top-k=3, and the `text-embedding-3-small` model returned "72.43 mg/dL." Two of the retrieved chunks originated from the Lab Results document, due to chunk overlap, each containing the correct glucose value at the expected timestamp. The model not only located the exact data point but also reproduced it verbatim, demonstrating high retrieval fidelity and generation faithfulness. This case exemplifies how small chunks with a modest top-k focus the prompt on a narrow, noise-free context, minimizing distraction and maximizing precision—consistent with findings on the benefits of dense retrieval for single-fact lookups [23].

In contrast, we observe the multi-document reasoning query—"How many years (rounded) was it between their miscarriage and their proceeding blood test?"—for the same patient under a 500-token/top-k=3 configuration with `text-embedding-3-large`. It exposed an over-retrieval and hallucination failure mode. Despite retrieving three lengthy chunks—one from the Procedures Report spanning years of events and another from the Patient Summary listing onset dates—the model answered "6" years instead of the correct "4." Here, the inclusion of broad, overlapping context diluted the salient dates and the model invented an interval unsupported by any chunk.

This underscores how larger chunks can swamp prompts with irrelevant details, triggering the "`wrong_generation`" failure mode even when the correct evidence was present. It also suggests that further prompt engineering techniques should be explored to counteract the noise introduced by deep retrieval in reasoning tasks.

Finally, we look at a contradiction detection test—"What is the reason for the patient's dementia?"—for patient `dallas143` under 500 token chunks, top-k=3, and `text-embedding-3-large`. Recall that the system was explicitly instructed to answer "Contradictory" when conflicting reasons appeared. The retrieval stage correctly surfaced the two conflicting care-plan entries where the doctor labels the causes of the patient's dementia as "Alzheimer's disease" in one record and "Hypertension" in another record with the same timestamp. Yet the model answered "Alzheimer's disease" despite clear, prompt instructions to flag contradictions, indicating a breakdown in reasoning rather than retrieval. It highlights the limitation of `GPT-4o-mini`'s reasoning capabilities over conflicting evidence—an insight in line with the findings by Ji et al. [20] of hallucination arising from unsupported synthesis rather than missing context.

Together, these case studies illustrate that while our current pipeline configuration excels at pinpointing single data points with minimal context, it remains vulnerable to over-retrieval noise and reasoning failures when faced with multi-document reasoning or contradiction tasks. These qualitative insights reinforce the quantitative findings and motivate future work on refined prompt patterns, dynamic chunk selection, and integration of explicit inconsistency detection.

4.4 Quantitative Findings

Our primary research question is "How do chunk size, number of chunks used (top-k), and embedding model affect retrieval precision and answer correctness across different query types (factual, multi-document, contradiction)?" To formally test this, we frame a falsifiable hypothesis: the null hypothesis (H_0) posits that these retrieval parameters have no effect on precision or correctness, while the alternative hypothesis (H_1) asserts that at least one parameter does affect performance for at least one query type.

To test this hypothesis, we use a statistical method called the Kruskal–Wallis test. This test is suitable because our performance scores don't follow a normal distribution. It works by ranking all the scores across groups and calculating the Kruskal–Wallis statistic, a number that reflects how much the ranks differ between groups. A larger statistic suggests greater differences. The test then returns a p-value, which estimates the probability of observing differences this large if, in fact, there were no real effects. If the p-value is small (typically < 0.05), we reject the null hypothesis and conclude that retrieval parameters have a statistically significant effect. To complement this, we also calculate epsilon squared (ε^2), a measure of effect size that tells us how much of the total variation in the data can be explained by the retrieval parameters. This way, we can assess not only whether a difference exists, but also whether it is practically meaningful. The results are shown in Table 4.1, and a detailed breakdown can be found in Appendix A.

System performance varied most significantly by question type, with factual queries yielding consistently high retrieval and correctness scores, while contradiction and multi-document questions proved substantially more difficult on correctness. These differences reflect the underlying cognitive demands: factual lookups rely on localized evidence, whereas contradiction and multi-document synthesis require reasoning across documents, often in the presence of ambiguity or conflict. The results that follow are therefore organized by query category, highlighting how each type responded to changes in chunk size, retrieval depth, and embedding model.

We start with factual queries, with results visualized in Figure 4.3. The evaluation confirms

that retrieval is rarely the bottleneck for factual look-ups but becomes a limiting factor once evidence is dispersed across multiple documents. For factual queries, mean retrieval precision exceeded 0.93 in every configuration and rose to a perfect 1.00 when the retriever returned five chunks. The corresponding Kruskal-Wallis test shows that top-k depth, not chunk size or embedding model, was the only statistically significant driver of precision ($p = 0.0163$). In practice, adding two extra neighbours improves recall without flooding the prompt, a trade-off that suits low-risk, single-fact tasks.

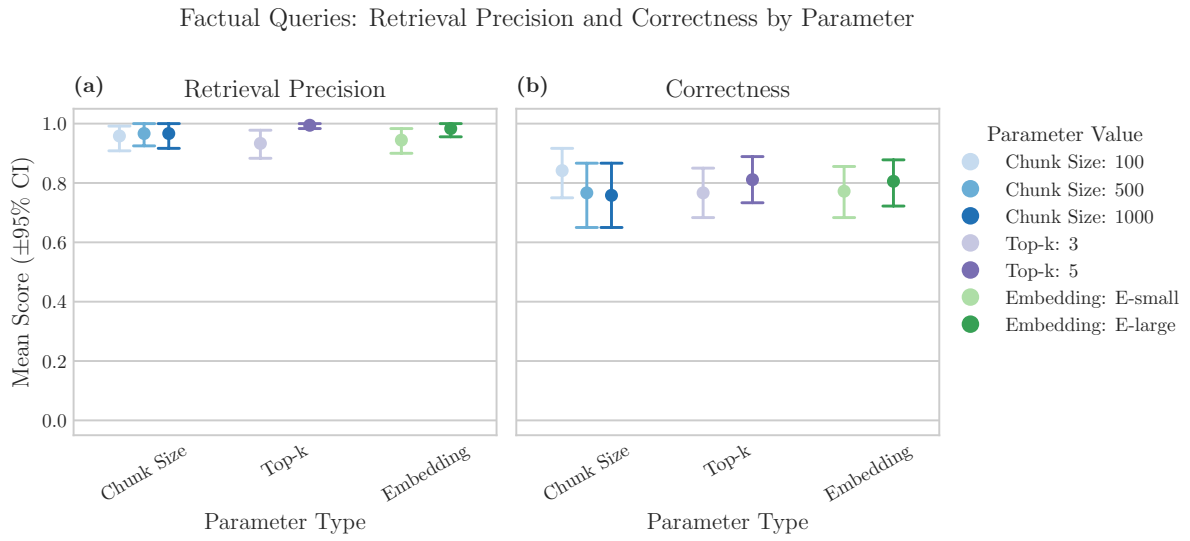


Figure 4.3: Performance for Factual Queries by Parameter Type

The results for contradiction queries, visualized in Figure 4.4, highlighted reasoning, not retrieval, as the key bottleneck. While larger chunks improved precision by capturing conflicting data ($p < 0.001$), the final answer correctness remained very low (mean score ≤ 0.33) across configurations. The AI model used (`GPT-4o-mini`) consistently failed to flag contradictions as instructed, instead asserting one version despite conflicting retrieved evidence. This points to limitations in logical consistency and instruction adherence for the tested model, suggesting improvements require better prompting strategies or models, not just better retrieval.

The results for multi-document reasoning queries, visualized in Figure 4.5, exposes the system's sharpest limits. Mean precision rises from 0.57 at 100-token chunks to 0.87 at 500 tokens, driven by the same chunk-size effect observed in contradiction queries ($p < 0.0001$). Mean correctness, however, never exceeds 0.33 and shows significance only for the embedding model: `text-embedding-3-large` outperforms its smaller counterpart ($p = 0.038$) by retrieving semantically diverse synonyms that the small model misses. Even so, the absolute gains are modest, cautioning against the intuition that "bigger embeddings fix everything".

Table 4.1: Kruskal-Wallis Test Results for Retrieval Parameters

Query Type	Metric	Factor	Kruskal-Wallis Statistic	p-Value	Epsilon Squared
Contradiction	**Precision**	**ChunkSize**	**14.4855**	**0.0007**	**0.0705**
Contradiction	Precision	TopK	1.3303	0.2488	0.0019
Contradiction	Precision	EmbeddingModel	1.3303	0.2488	0.0019
Contradiction	Correctness	ChunkSize	4.3725	0.1123	0.0134
Contradiction	Correctness	TopK	1.3166	0.2512	0.0018
Contradiction	Correctness	EmbeddingModel	0.4134	0.5203	-0.0033
Factual	Precision	ChunkSize	0.6332	0.7286	-0.0077
Factual	**Precision**	**TopK**	**5.7680**	**0.0163**	**0.0268**
Factual	Precision	EmbeddingModel	2.8929	0.0890	0.0106
Factual	Correctness	ChunkSize	1.4394	0.4869	-0.0032
Factual	Correctness	TopK	0.6934	0.4050	-0.0017
Factual	Correctness	EmbeddingModel	0.2986	0.5847	-0.0039
Multi-document	**Precision**	**ChunkSize**	**30.5183**	**0.0000**	**0.1611**
Multi-document	Precision	TopK	1.9570	0.1618	0.0054
Multi-document	Precision	EmbeddingModel	2.7697	0.0961	0.0099
Multi-document	Correctness	ChunkSize	1.1535	0.5617	-0.0048
Multi-document	Correctness	TopK	2.4572	0.1170	0.0082
Multi-document	**Correctness**	**EmbeddingModel**	**4.3055**	**0.0380**	**0.0186**

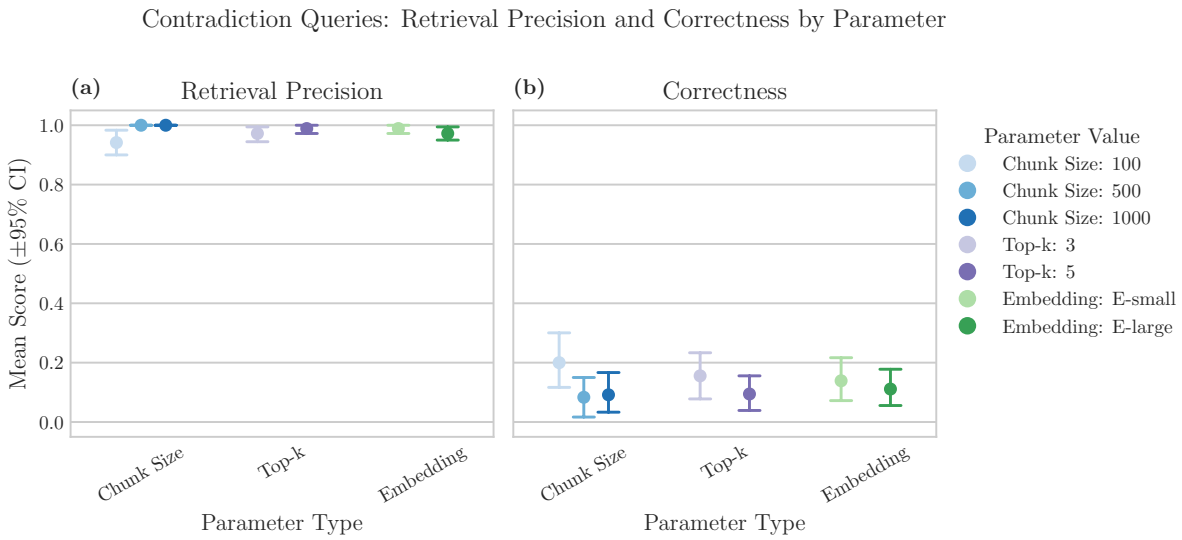


Figure 4.4: Performance for Contradiction Queries by Parameter Type

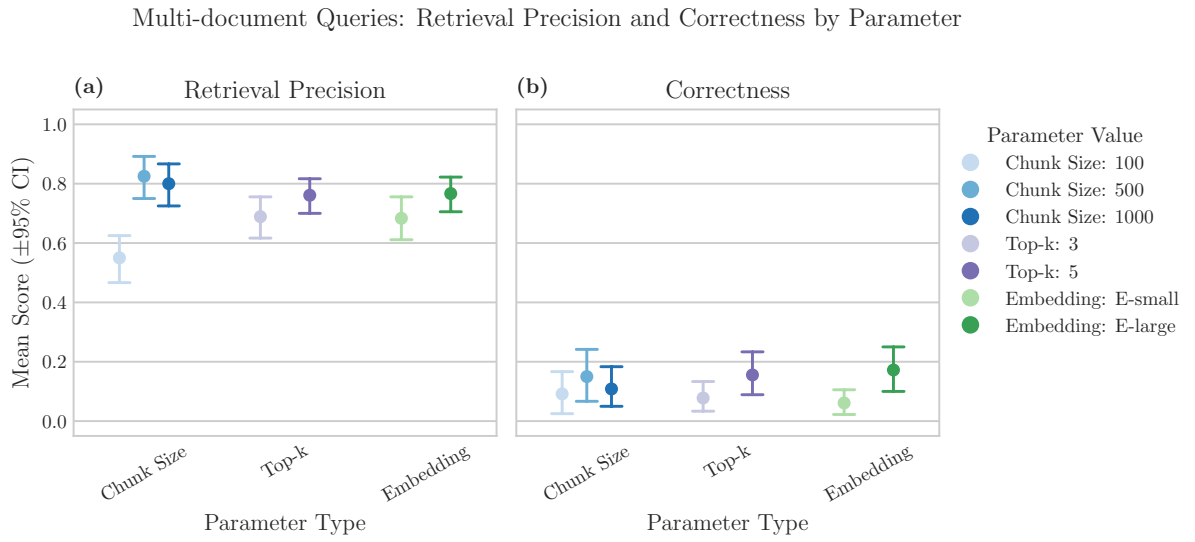


Figure 4.5: Performance for Multi-document Queries by Parameter Type

Now we explore how chunk size and top-k affect token consumption and latency, and the relationship between cost and accuracy. Resource analysis showed diminishing returns for larger contexts. Token usage grew predictably with chunk size and k, inflating costs significantly for configurations like 1000-tokens and k=5. The larger context improved retrieval precision for multi-document reasoning questions, but didn't have an affect on the other query types. However, the larger context failed to improve, and sometimes worsened, correctness for complex queries, likely due to attention dilution. These observations are illustrated in Figures 4.6 and 4.7. While total latency remained acceptable, the optimal cost-accuracy balance favoured moderate settings (e.g., 500-tokens and k=3). Maximizing context is not always optimal; tuning requires balancing recall needs against AI model limitations and token costs.

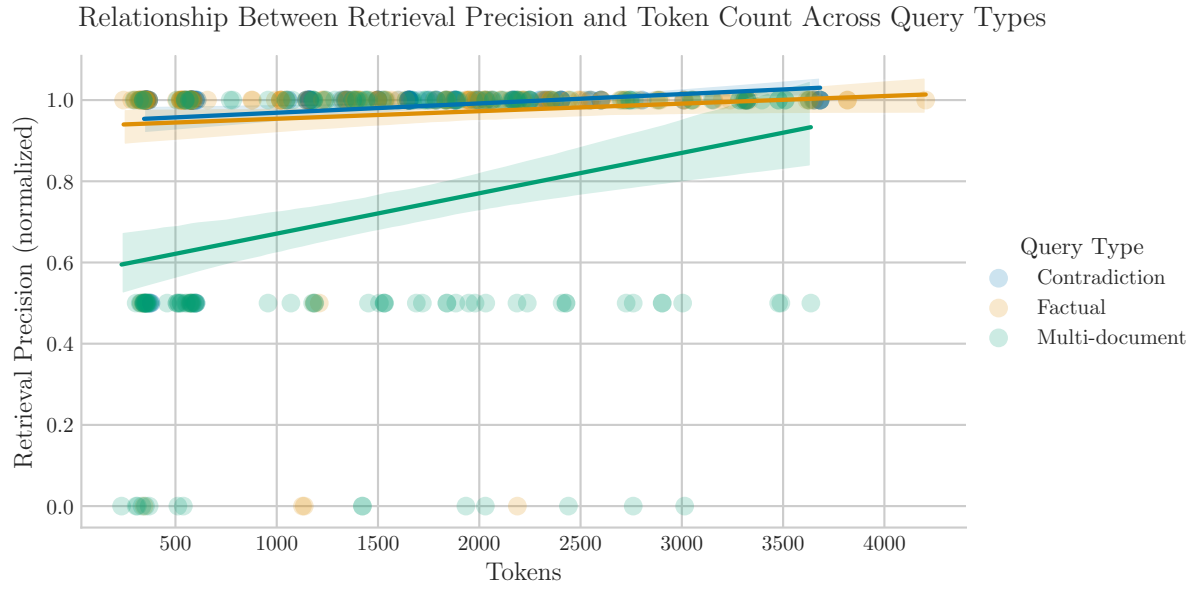


Figure 4.6: Relationship Between Retrieval Precision and Token Count Across Query Types

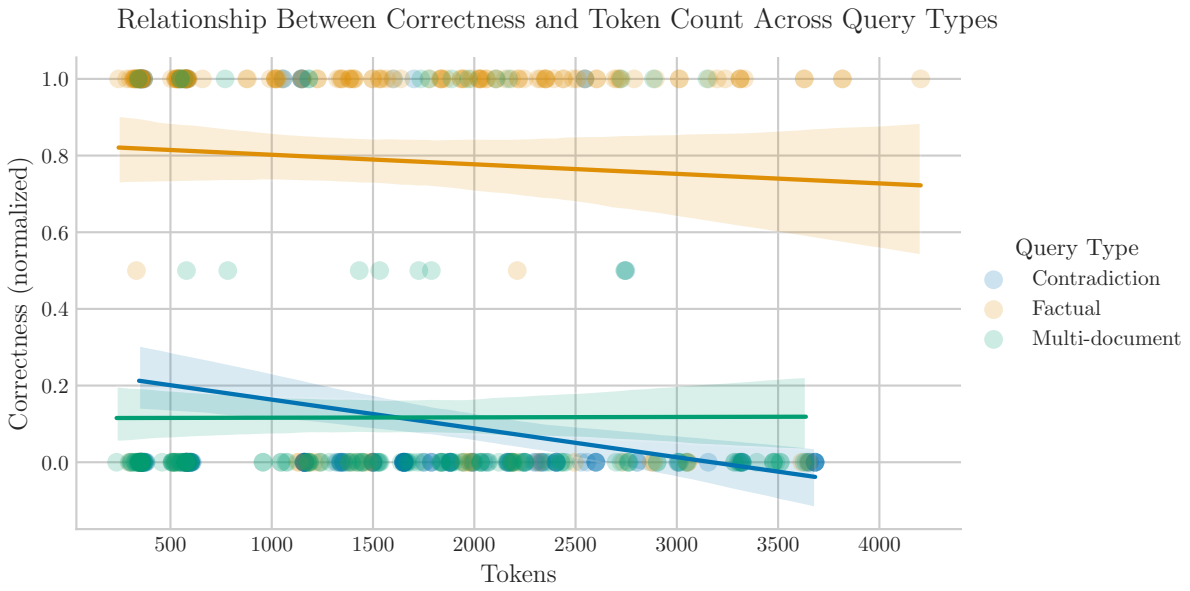


Figure 4.7: Relationship Between Correctness and Token Count Across Query Types

Latency trends mirror the token story. Retrieval latency drops as chunk size grows, because a smaller index means fewer cosine operations, but generation latency climbs sub-linearly with prompt length, as seen in Figure 4.8. Across all tests the mean end-to-end delay stayed below two seconds on an M1-equipped MacBook Pro, indicating that even the heaviest configuration remains manageable for a single user.

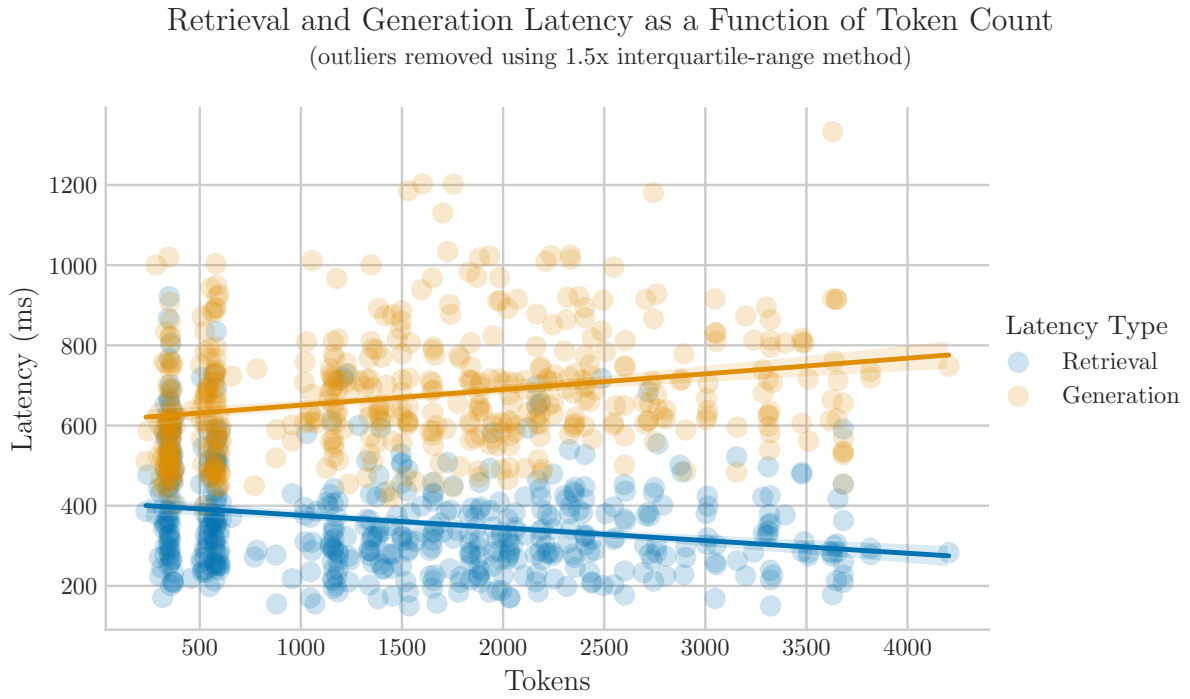


Figure 4.8: Retrieval and Generation Latency as a Function of Token Count

No latency penalty surfaced when switching from the small to the large embedding model. In practice, document embeddings are pre-computed so any cost difference at query time is

economic rather than experiential.

Together, the numbers confirm three design lessons. First, high precision alone does not guarantee correctness once queries demand reasoning or reconciliation—limitations also observed by Liu et al. [27] in their long-context study. Second, chunk size is the dominant retrieval lever for distributed evidence, but its benefits plateau quickly for factual tasks. Third, the larger embedding model earns its keep only for multi-document reasoning, suggesting that use cases default to the cheaper encoder and escalate selectively when the use case requires multi-document reasoning. These insights guide the parameter presets recommended in our reflection (Chapter 6) and underscore that future gains are more likely to come from stronger reasoning models or better prompt engineering rather than from more aggressive retrieval.

4.5 Failure Mode Analysis

Here we explore which retrieval parameters correlate with each failure category for each query type. Across the 540 question–configuration runs distinct failure patterns emerged for factual, contradiction, and multi-document queries. These patterns map directly onto our four failure-mode categories—"`contradictory_data`", "`wrong_generation`", "`missed_context`", and "`none`"—and reveal how chunk size and retrieval depth shape the locus of error. A detailed breakdown can be found in Appendix B.

For contradiction queries, retrieval almost invariably surfaced both conflicting chunks—hence `missed_context` was essentially zero. Instead, failures overwhelmingly fell into `contradictory_data`, as seen in Figure 4.9. They averaged roughly 80 percent across configurations (peaking at 93% for the 100-token, k=5, large embedding configuration as shown in Table B.1). In those cases the model consistently chose one diagnosis rather than flagging the discrepancy. A smaller share of errors, up to 20%, stemmed from `wrong_generation`, where the AI model misquoted or misinterpreted perfectly retrieved evidence (e.g. selecting a single cause without conflict resolution). Even increasing k or chunk length did little to shift this balance, underscoring that the bottleneck lies in the model's reasoning over conflicting inputs, not in retrieval coverage.

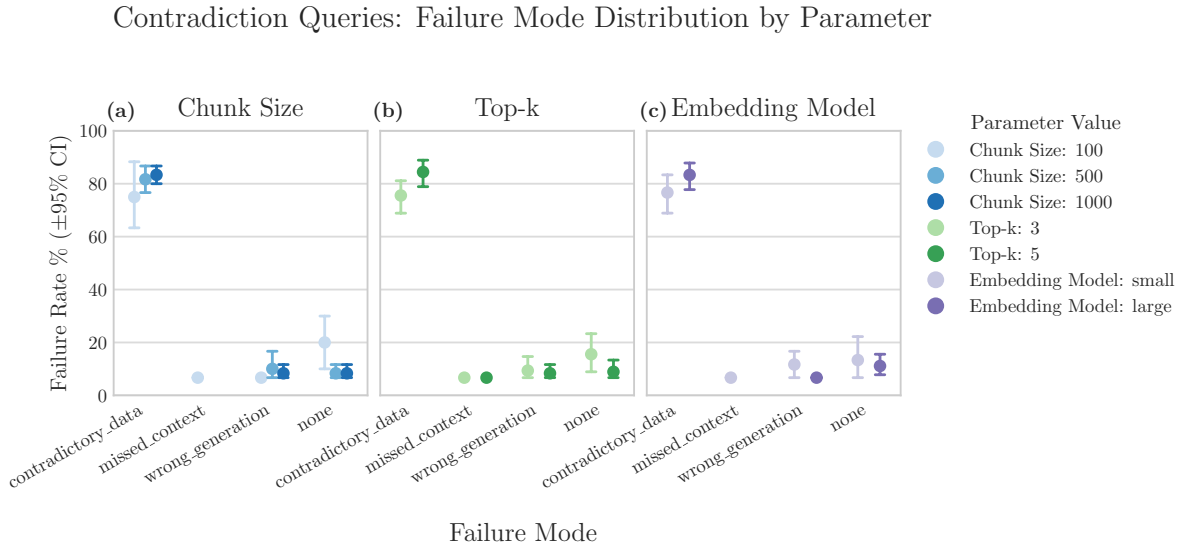


Figure 4.9: Failure Mode Distribution by Parameter for Contradiction Queries

In factual queries, shown in Figure 4.10, system performance was strongest: 66 to 87% of runs

yielded no failure, with correct answers grounded in retrieved chunks. When errors did occur, they split between `wrong_generation`—the AI model misreading numbers or omitting qualifiers—and `missed_context`, particularly in the 100-token, k=3, small-embedding configuration (about 6% `wrong_generation` and 13% `missed_context`). Larger chunks and deeper retrieval reduced `missed_context` to near zero but slightly increased `wrong_generation`, suggesting that superfluous context can dilute attention on the relevant facts.

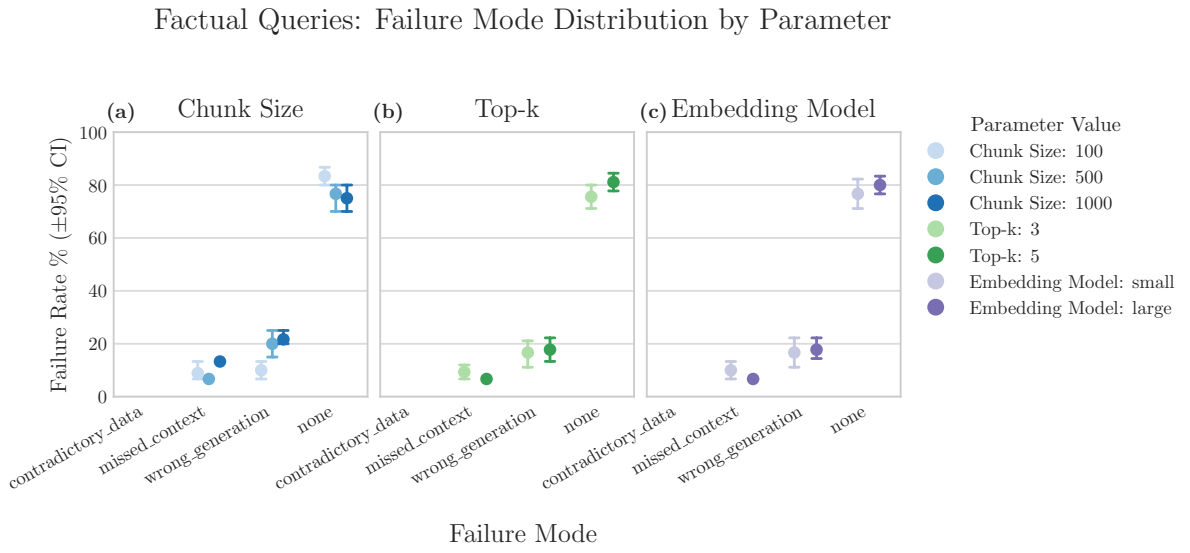


Figure 4.10: Failure Mode Distribution by Parameter for Factual Queries

The greatest challenges appeared in multi-document queries, where the system must aggregate and reason about information across separate records. With the fewest tokens (100 token per chunk, k=3 chunks) and with small embeddings, `missed_context` dominated (up to 93% of failures), as one of the two needed documents often went unretrieved. By contrast, at 500 or 1000-token chunks, `missed_context` fell sharply but `wrong_generation` rose to 60–80%, indicating that although retrieval coverage improved, the model struggled to coherently synthesize multiple inputs. Notably, `contradictory_data` did not apply here, and "`none`" remained below 15% in most tasks, as seen in Figure 4.11.

Embedding model choice had minimal impact on failure distributions, confirming that for our small corpora, retrieval depth and chunk granularity drive both retrieval and generation errors. In sum, factual lookups succeed when retrieval is precise and context is concise; contradiction handling fails at the reasoning layer despite perfect retrieval; and multi-document reasoning requires balanced chunking—large enough to capture dispersed facts, yet concise enough to avoid overwhelming the model's generative capacity.

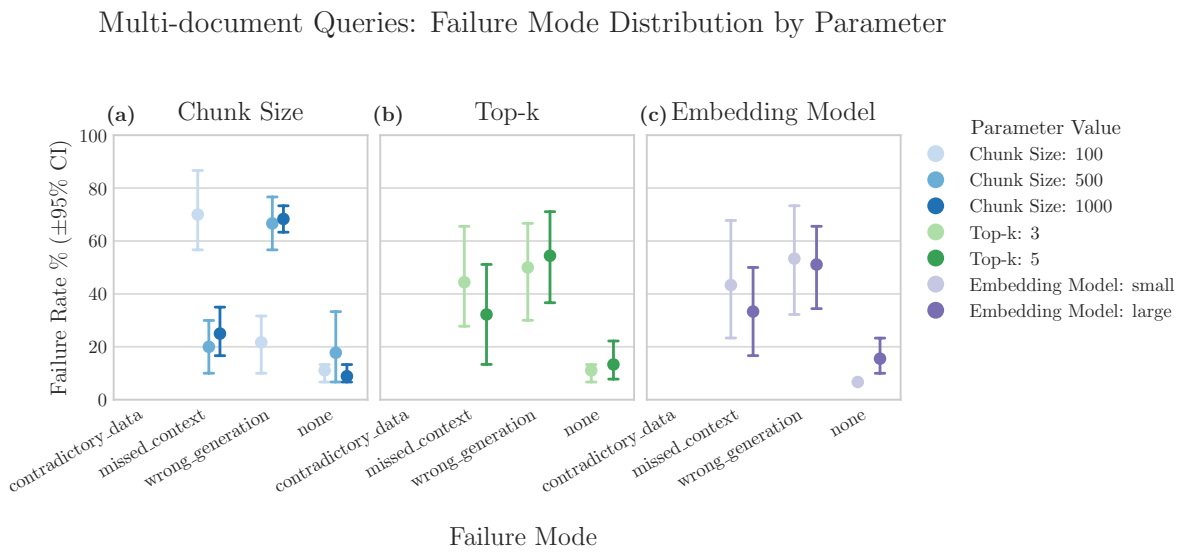


Figure 4.11: Failure Mode Distribution by Parameter for Multi-Document Queries

5 Ethical, Legal, and Social Implications

5.1 Accuracy, Hallucination, and Harm

In high-stakes domains such as personal health informatics, the accuracy and reliability of an AI assistant are not merely technical goals but ethical imperatives. Patients and clinicians must be able to trust that the information provided faithfully reflects the underlying medical records. While RAG is employed specifically to ground outputs in evidence, our evaluation in Chapter 4 demonstrated that significant risks remain, particularly concerning the generation process itself.

The key finding from our evaluation relevant to ethics is this: even though the RAG system was often successful at retrieving the correct document sections, it struggled to accurately synthesize or reason about that information. This happened particularly for complex tasks like combining information from multiple documents or handling contradictions. This led to instances where the system produced confidently stated, plausible-sounding inaccuracies—or "fact fabrication" as described in Section 2.2. This risk is ethically significant because, as literature suggests, the fluency of AI outputs can mask underlying errors, potentially leading users to place undue trust in incorrect information, which could have serious consequences for health decisions.

Recognizing this potential for harm, several design choices were implemented as ethical safeguards. The insistence on inline citations referencing specific source documents aims to provide transparency and allow users to verify claims. The system prompt explicitly instructs the model towards epistemic humility—mandating caution when evidence is absent or conflicting—and includes safety protocols for emergency situations. Furthermore, a low generation temperature was used to promote output consistency so that results are more reproducible.

However, these mitigations represent a balancing act and are not foolproof. Enforcing strict citation and caution can sometimes lead to overly conservative responses or increased latency and cost, potentially reducing the system's utility. More fundamentally, they primarily address grounding, but do not fully prevent failures in the AI model's reasoning process itself, which was identified as the main source of error for complex tasks.

Therefore, while RAG and careful prompt engineering are necessary steps, they are insufficient alone to guarantee safety in this sensitive domain. Document grounding constrains fabrication but does not eliminate reasoning errors. Ethically responsible deployment would necessitate a more layered approach. This includes incorporating more robust mechanisms for handling uncertainty and contradiction, potentially through more advanced prompting techniques or dedicated checks, and integrating features that clearly communicate the system's confidence levels to the user. Ultimately, acknowledging the residual risks and limitations requires maintaining human oversight in any safety-critical application of this technology.

5.2 Data Privacy and Consent

The ethical handling of PHI necessitates robust data privacy measures and informed user consent. This project's prototype, developed rapidly using synthetic data, employed a hybrid architecture combining local processing with external API calls for embedding and generation. While pragmatic for the prototype, this model immediately surfaces critical privacy considerations that must be addressed before handling real PHI.

Transmitting potentially sensitive data to third-party services requires navigating regulations like HIPAA and GDPR. Compliance pathways include formal agreements (such as "Business Associate Agreements (BAAs)" or "Data Processing Agreements (DPAs)") with API providers (e.g. OpenAI's BAA [30]). These agreements mandate safeguards, contingent on provider availability and rigorous due diligence. They may require implementing robust de-identification procedures based on standards like HIPAA safe harbour, though this may impact utility and involve significant technical and procedural hurdles.

In light of the complexities and risks associated with external API use for PHI, the alternative architectural approach—a fully local-first system achieved by substituting external calls with on-premise models—emerges as preferable if data privacy is prioritized over development simplicity. This method most directly ensures data sovereignty, reduces third-party dependencies, simplifies compliance, and aligns with data minimization principles.

Independent of the chosen architecture, securing truly informed consent for RAG systems presents unique difficulties. Users must grasp complex processes beyond simple data storage, including algorithmic document chunking, conversion to abstract embeddings, and the possibility of the generative model producing novel but potentially flawed outputs (as discussed in Section 5.1). Standard consent forms likely require significant adaptation to effectively communicate these specific risks.

Ultimately, implementing effective privacy safeguards—be they technical (encryption, de-identification), architectural (local deployment), or procedural (enhanced consent)—involves balancing competing concerns. Measures increasing privacy may add system complexity, introduce latency, or negatively impact usability. Navigating these trade-offs between protection, performance, and user experience is a persistent ethical challenge in system design and deployment.

5.3 Misuse and Security Risks

Beyond unintentional errors, AI systems like the RAG assistant developed here can be deliberately misused or attacked. Understanding the specific vulnerabilities inherent in the RAG pipeline—spanning ingestion, retrieval, and generation—is crucial for designing effective security controls and maintaining user trust, especially within the sensitive domain of personal health.

The RAG architecture presents distinct attack vectors beyond generic system security concerns. Retrieval poisoning [54] poses a significant threat: an adversary could intentionally upload documents containing manipulated or dangerously incorrect health information (e.g., wrong dosages, fake diagnoses). If these malicious documents are retrieved during a query, the system could generate misleading or harmful advice, grounded in seemingly legitimate, but poisoned, sources. Prompt injection [28] represents another vector, where an attacker could craft a malicious query designed to trick the AI model into ignoring its original instructions (the "system prompt"). For example, a query might try to make the AI bypass safety rules, reveal confidential parts of its own prompt, or execute unintended commands, essentially hijacking the AI's intended function through clever user input. Furthermore, sophisticated adversaries

might attempt data exfiltration attacks [47] that exploit how the embedding and similarity search works. The goal wouldn't be just to retrieve text, but to infer potentially sensitive patterns or relationships between documents based on which ones have similar embeddings, potentially revealing more information than intended.

The current prototype's local-first, single-session design inherently mitigates certain risks, such as unauthorized access to a central user database or cross-user attacks. However, it remains vulnerable to misuse by the primary user operating the system (e.g., self-poisoning their document set or attempting prompt injection) and offers limited protection should the underlying components, like the external AI model API, be compromised. Scaling this system to a multi-user or web-accessible service would drastically expand the attack surface, making robust security measures paramount.

Therefore, a defence-in-depth strategy, grounded in security engineering principles, is necessary. For any scaled deployment, this must include strong authentication and session isolation. Critically, robust input sanitization is required at the query interface to mitigate prompt injection risks, while document integrity checks (e.g., checksums or digital signatures upon upload) are needed to counter retrieval poisoning. If external APIs remain part of the architecture, secure API key management, usage monitoring, and anomaly detection are vital to prevent unauthorized access or large-scale exfiltration. Comprehensive audit logs, recording uploads, queries, and retrieved context IDs, are essential not just for accountability but specifically for forensic analysis following a suspected security incident.

Implementing these safeguards involves navigating ethical and practical trade-offs. For instance, overly strict input sanitization might inadvertently reject valid, complex medical queries or text containing necessary special characters. Enhanced logging for security forensics potentially increases privacy risks if not managed carefully. Achieving robust security without unduly hindering usability or compromising privacy requires careful design and continuous evaluation. Ultimately, anticipating potential misuse and embedding security considerations throughout the development lifecycle are ethical responsibilities necessary to ensure the assistant remains a trustworthy tool for health exploration rather than an exploitable vulnerability.

5.4 Legal Accountability and Responsibility

Our RAG-based AI health assistant introduces significant legal and ethical questions regarding accountability and responsibility, particularly when outputs may influence health-related decisions. Unlike deterministic software with predictable outcomes, this system synthesizes responses probabilistically, blending potentially imperfect retrieved data with the generative capabilities of an AI model. This makes assigning liability in the event of erroneous or harmful advice exceptionally complex compared to traditional software or direct human error. Determining fault—whether it lies with the original data source, the retrieval mechanism, the AI model's generation process, the system's implementation, or the user's interpretation—challenges existing legal frameworks for product liability and professional negligence.

Given this ambiguity, establishing clear lines of responsibility is crucial. A potential approach involves a shared responsibility framework, acknowledging that accountability is distributed [50]. System designers and implementers bear responsibility for the system's architecture, safety features, and transparent communication of its capabilities and limitations. This project attempted to address this through specific design choices: embedding clear disclaimers about the system not being a substitute for professional medical advice in the UI, grounding outputs in cited sources to allow verification, and deliberately instructing the AI model to limit the scope, avoiding diagnostic claims. These choices align with professional ethical codes demanding

integrity, honesty about limitations, and a safety-first mindset [7, 39].

However, disclaimers alone are insufficient to absolve responsibility. Mechanisms to ensure practical accountability are essential. Robust audit trails, meticulously logging data provenance, queries, retrieved evidence, prompts, and generated outputs, serve a critical function beyond security and privacy compliance. Specifically for accountability, these logs provide the necessary forensic evidence to reconstruct the system's behaviour preceding an adverse event, potentially clarifying the chain of causation. Furthermore, incorporating human-in-the-loop workflows for reviewing critical outputs, as suggested for future work, acts not only as a safety check but also as an explicit mechanism for involving human judgment and clarifying the point at which responsibility for action is assumed.

Looking forward, the legal landscape for AI in healthcare is evolving. Anticipating and engaging with emerging governance frameworks and standards, such as those emphasizing accountability and transparency from organisations like the WHO, is vital. Designing systems with robust accountability mechanisms from the outset is not only ethically prudent but also essential for mitigating legal risks and fostering the trust necessary for responsible adoption by patients, clinicians, and regulators.

5.5 Algorithmic Bias and Fairness

Fairness in AI healthcare systems is an ethical imperative. Bias can infiltrate the RAG pipeline primarily through the inherited characteristics of the external OpenAI embedding and generation models, likely trained on demographically skewed corpora (as discussed in Section 2.5). This inherited bias, potentially amplified during synthesis, could lead to misinterpretations or under-representation of health information for certain demographic groups. Furthermore, design choices like fixed-size chunking and dense vector retrieval might inadvertently favour dominant linguistic patterns or fail to capture culturally specific terminology, potentially impacting retrieval relevance and overall system equity.

The consequences in a personal health context could involve marginalizing specific health concerns or undermining trust for diverse users. Critically, the current evaluation's reliance on synthetic data prevents a rigorous assessment of these fairness risks across real-world populations, highlighting a significant ethical limitation. Fairness also encompasses accessibility; the system's dependence on commercial APIs presents cost barriers, while its current monolingual focus excludes non-English speakers, further limiting equitable access to its potential benefits.

Addressing these concerns requires embedding fairness as a core design principle. Future work must involve evaluation on diverse datasets (with ethical oversight) to identify and mitigate performance disparities. This includes exploring bias mitigation techniques within the RAG pipeline and actively addressing accessibility through multilingual support and exploring cost-effective open-source alternatives. Proactive attention to bias and equitable access is essential for responsible development and deployment.

6 Reflections

6.1 System Design Reflections

Reflecting on the system design process provides an opportunity to critically evaluate key architectural decisions against our initial pillars—traceability, local-first privacy, modularity, simplicity with observability, and evaluation-driven development—alongside software engineering principles and project outcomes. The ambition was a pragmatic, auditable RAG assistant, and the design choices largely stemmed from balancing this goal against strict project constraints (Section 3.1).

The selection of a single-process Django monolith backed by `SQLite` exemplified adherence to the You Ain't Gonna Need It (YAGNI) principle and prioritized development velocity (C-1) and observability (NFR-3) for this proof-of-concept. While consciously sacrificing horizontal scalability, this decision proved advantageous: the unified structure simplified debugging across the RAG pipeline and facilitated the detailed evaluation logging central to the evaluation. This experience underscored that architectural simplicity, when aligned with project scope (G-3), can directly enable deeper investigation.

Similarly, employing brute-force cosine similarity search over embeddings stored in `SQLite`, rather than integrating an ANN index, was a deliberate choice against premature optimization. Profiling confirmed that AI model inference was the majority of end-to-end latency, validating that the added complexity of a dedicated vector store was unnecessary at this scale. This approach maximized transparency (NFR-3) and simplified the evaluation framework (US-4), although the modular design ensures this retrieval component could be replaced if scalability becomes a future requirement (NFR-4).

Managing the tension between the local-first ideal and the practical need for powerful cloud-based APIs for embedding and generation was a core design challenge. The compromise—using external APIs with synthetic data—was enabled by implementing a modular adapter layer. This software engineering pattern successfully isolated external dependencies, preserving architectural flexibility to substitute local models in future iterations requiring stringent PHI handling, thereby managing the conflict between development constraints (C-1, C-4) and ethical/privacy imperatives.

Design gaps also became apparent during implementation. The initial design treated the retriever and generator as loosely coupled stages, passing retrieved chunks wholesale. This passive handover proved insufficient for complex queries involving conflicting evidence, where the generator lacked guidance on reconciling inconsistencies—a limitation reflected in the evaluation results. In retrospect, incorporating a re-ranking layer or more sophisticated prompt engineering, like the Fact Check List pattern, earlier could have better bridged retrieval and synthesis, highlighting the importance of interface design between pipeline stages.

Conversely, the stateless ingestion pipeline, meticulously tagging each chunk with provenance metadata, stands out as a design success. It directly fulfilled the traceability pillar and WHO principles, enabling straightforward auditing and providing the foundation for verifiable, citation-

backed answers—a critical feature for trustworthy clinical AI.

Looking forward, reflections on the system's limitations directly motivate specific design enhancements. The evaluation's clear indication that AI model reasoning, not just retrieval, is a bottleneck necessitates exploring reasoning-optimized models or prompting strategies (e.g., chain-of-thought, plan-and-solve). To address recall limitations for dispersed facts without overwhelming the generator, hybrid sparse-dense retrieval with intelligent re-ranking could be explored. Finally, achieving true ethical alignment requires migrating embeddings on-premise to close the privacy loop and enhancing the UI with features like confidence scores and source chunk previews to bolster transparency and user trust. These future directions stem directly from evaluating the current design against its goals and limitations.

6.2 Evaluation Reflections

Our evaluation was conceived as a controlled experiment to isolate the impact of core RAG parameters, rather than a simple benchmark chase. The factorial design, varying chunk size, retrieval depth, and embedding models across distinct query types, provided the structure for this investigation. This section critically reflects on that evaluation process, the validity and limitations of its findings, and the lessons learned about assessing RAG systems in this domain.

The most significant finding was the identification of AI model reasoning, not retrieval recall, as the primary bottleneck for complex queries involving multi-document reasoning or contradiction. This aligns with emerging literature highlighting fact fabrication challenges even in retrieval-augmented systems, as described in Section 2.2, and shifts the focus for future RAG improvements towards enhancing generator reliability and consistency. We also observed diminishing, sometimes negative, returns from providing excessive context, suggesting that simply maximizing retrieval depth is not always optimal.

The evaluation framework itself yielded granular insights, particularly through the failure mode taxonomy which helped pinpoint error sources. However, the multi-dimensional assessment (precision, correctness, latency, cost, failure mode) generated substantial annotation overhead, even with AI assistance for initial labelling. In retrospect, focusing annotation efforts more tightly on correctness and retrieval precision, treating other metrics as diagnostics, might have yielded a more efficient process without sacrificing core insights.

The most significant limitation impacting the findings' trustworthiness and generalizability was the reliance on synthetic patient data. While necessary for privacy and feasibility, this clean dataset lacks the inherent noise, OCR errors, formatting inconsistencies, and longitudinal ambiguities of real-world EHRs. Consequently, the system's performance under realistic conditions remains untested, constraining the applicability of our quantitative results and potentially masking failure modes prevalent in practice.

Through this process, we learned valuable lessons about evaluating RAG systems. It underscored the necessity of designing tasks that specifically probe reasoning and synthesis capabilities, not just factual recall. It also highlighted the practical challenges of creating robust, reliable evaluation datasets and annotation schemes, particularly for nuanced aspects like correctness and failure attribution in generative systems.

Therefore, future evaluations must prioritize ecological validity by incorporating real-world, albeit appropriately anonymized, patient data—requiring careful ethical consideration and likely Institutional Review Board (IRB) approval. Methodologically, advancements should include metrics assessing reasoning coherence and calibration, alongside tasks specifically designed to challenge synthesis and contradiction handling. Furthermore, developing more efficient, potentially semi-automated, methods for annotation review will be crucial for scaling evaluation

efforts effectively.

6.3 ELSI Reflections

This project drove home that navigating ELSI is less about applying static principles and more about confronting emergent complexities during development. Moving beyond the theoretical risks detailed previously, the practical experience of building and evaluating this RAG prototype surfaced critical lessons in ethical engineering, revealing how initial assumptions were challenged and perspectives shifted through the course of the work.

The early commitment to a local-first architecture prioritized user data autonomy over simpler cloud-based designs. While this was the goal, practical constraints meant relying on external APIs for core AI capabilities. This necessary reliance on external services introduces the future challenge of managing these API dependencies securely and mitigating the inherent privacy risks they pose to the local-first ideal. Consequently, the design of modular adapters became a direct exercise in embedding future ethical possibilities for PHI handling. Implementation details also revealed deeper ELSI challenges. Crafting effective system prompts (Appendix C) proved surprisingly difficult, requiring meticulous effort to define safe behaviour precisely for the AI model itself. This difficulty foreshadows the immense practical challenge of achieving truly informed consent, which requires clearly communicating the system's complex operations and risks to users.

It was the evaluation phase, however, that provided the most sobering insights. Witnessing the supposedly "grounded" model fabricate answers or ignore contradictions, even with relevant evidence readily available (Section 4.3), made the limitations of retrieval augmentation viscerally clear. This empirical reality forced a critical reconsideration of RAG's safety profile, shifting the understanding of robust uncertainty handling from merely a desirable feature to an absolute, non-negotiable prerequisite for trustworthy deployment in sensitive domains.

Finally, project constraints inevitably shaped the ethical landscape. The necessary use of synthetic data, while upholding non-maleficence for this stage, simultaneously created a frustrating inability to empirically investigate potential algorithmic fairness, leaving a significant known risk unexplored. Resource limitations also meant navigating a constant, tangible tension between implementing ideal security or privacy safeguards and the pragmatic need to deliver. This experience underscored the dynamic interplay between technical feasibility, ethical aspiration, and the crucial role of iterative evaluation—coupled with epistemic humility—when developing increasingly potent AI tools.

6.4 Risk Management Reflections

Reflecting on this project's risk management involved comparing initial plans against the often-unpredictable realities of a research-heavy software endeavour (Table 6.1). This comparison wasn't just an academic exercise; it yielded critical, hard-won lessons about navigating uncertainty in software engineering research.

Initial technical concerns, such as potential API instability, proved largely unfounded in practice. The proactive design of modular adapters provided effective insulation, confirming the value of this specific mitigation strategy against dependency risks. However, while deliberate scoping contained core implementation complexity, the project significantly underestimated the effort required for the evaluation component. This scope creep, particularly around the annotation workload for hundreds of query-pairs, demanded strict adherence to deadlines to

Table 6.1: Risk Audit Table

Initial Risk (from project proposal)	**Anticipated Mitigation**	**What Occurred**	**Response & Lesson**
Technical integration (API limitations or changes)	Early API testing, modular adapter layer, fallback stubs	No breaking changes; API surface only expanded	Modular design paid off; integration risk proved negligible, confirming value of interface isolation
Implementation complexity (RAG + case management)	Incremental development, clear MVP, regular reviews	Dropped unnecessary add-ons early; RAG pipeline integrated smoothly	Simplicity trumped ambition; tight MVP boundaries avoided runaway complexity
Performance degradation (latency, token cost)	Early profiling, optimization budget, performance targets	Majority of latency from AI model call; brute-force retrieval acceptable	Premature optimization avoided; real profiles should precede infra work
Scope management (feature creep)	Weekly scope reviews, prioritized backlog	Evaluation ambitions ballooned annotation workload	Deadline discipline and feature cuts re-focused effort; scoping is an ongoing, not one-off, activity
Contingency events (major API failure, severe performance regressions)	Feature reduction plan, timeline extension triggers	No trigger events; contingency dormant	Having a written fallback increased confidence even unused
Unanticipated: dataset realism gap	Not in original plan	Synthetic records proved too tidy, masking real-world noise	Early realism audit needed; IRB pathway should be explored sooner

manage and served as a stark reminder of how difficult it is to accurately forecast effort for research-intensive tasks within fixed timelines.

Perhaps the most crucial learning came not from managing anticipated risks, but from confronting an unforeseen one: the dataset realism gap. The ethically mandated decision to use synthetic data created a significant blind spot, masking the noise and ambiguity inherent in real-world clinical documents. Recognizing, deep into the evaluation phase, that this fundamentally limited the generalizability of the findings highlighted a critical oversight in the initial risk assessment: the failure to fully consider the technical validation risks introduced by non-technical, ethical constraints. This underscored how non-technical factors can introduce substantial project risk.

While agile practices like maintaining a living risk register and leveraging modularity as inherent risk buffers were beneficial, the experience also exposed the limitations of static, upfront risk analysis for projects dominated by research uncertainty. Standard methodologies struggle to anticipate "unknown-unknowns," particularly when, as seen here, an early ethical decision profoundly impacts later technical validation.

This project's journey, therefore, offers concrete illustrations of vital risk management principles. The struggle with evaluation scope creep vividly reinforces the necessity of continuous, disciplined scope control, especially for research components. The unexpected impact of the dataset realism gap demonstrates why risk assessment must be holistic, actively probing the complex interdependencies between technical, ethical, and research factors. Finally, having explicit contingency plans provided valuable psychological resilience, confirming the benefit of fallback strategies even if specific trigger events don't materialize.

6.5 Scaling from Prototype to Population

The retrieval-augmented AI assistant developed in this project served its purpose as a local-first prototype, enabling detailed evaluation and ensuring user privacy by design (Section 4.3). This approach met the dissertation's objectives but, as acknowledged in Section 1.3, does not address the challenges of deploying such a service widely. This reflection explores the journey required to evolve the prototype into a service for progressively larger groups, moving from a few households to potentially globally, highlighting the specific technical, financial, and ethical hurdles that arise and the trade-offs involved in overcoming them.

To make this scaling discussion concrete, we start with some basic estimates (detailed in Appendix D). Let's assume a typical user asks about 20 questions per month. For these queries, we assume the system uses commercial APIs from a provider like OpenAI—specifically, their state-of-the-art reasoning model, `o3`, for generating answers and the `text-embedding-3-large` model for creating embeddings. We estimate an average query uses about 1500 input tokens (representing the retrieved chunks plus instructions) and generates 500 output tokens (the AI's answer). Based on OpenAI's current pricing tiers [33] the cost for just the answer generation would be around $0.70 per user per month. We also factor in re-calculating embeddings for the user's documents periodically (estimated at 25,000 tokens monthly), but this cost is negligible (less than a cent). Therefore, the core API cost is roughly $0.71 per user per month. Crucially, this figure only covers the direct API usage—it does not include costs for servers, databases, network bandwidth, development, or support staff, which would add significant expense in a real deployment. Storage needs are estimated at 26 MB per user.

For a single user or a few households, the prototype operates effectively on individual laptops. Storage (tens of megabytes), API costs (under a dollar monthly), and peak load (< 0.001 Queries Per Second (QPS)) are minimal. Data is stored locally but transmitted to OpenAI for embedding

and AI model inference. Performance is gated by external API latency, not local processing. The key trade-off is the lack of shared access or oversight by a medical professional—it is suitable only for personal use.

Deploying within a doctor's clinic (perhaps one thousand users) forces a move to the cloud. The local architecture cannot support concurrent access; `SQLite`, while handling concurrency via locking, would serialize requests, effectively freezing the UI under load. Migrating to a centralized cloud server with a database like `PostgreSQL` using its `pgvector` extension becomes necessary to handle the modest peak load (below 0.1 QPS). This introduces infrastructure costs and operational duties. Centralizing 25 GB of patient data creates significant privacy risks and demands HIPAA compliance. The accepted trade-off is reduced privacy for shared clinical utility, with API costs rising to $710 per month.

Scaling to a hospital system (one hundred thousand users) reveals new bottlenecks. The vector index size (2.6 TB) likely exceeds the capacity of simple database extensions like `pgvector`, requiring migration to a dedicated, scalable vector database service (e.g., `Qdrant`, `Weaviate`) that can handle the load without query latency ballooning. The peak query load reaches 8 QPS, necessitating an auto-scaling cluster of application servers. Asynchronous processing for ingestion becomes critical. API costs hit $71,000 per month, making optimization via caching vital. The trade-off involves adopting significantly more complex and costly infrastructure to maintain performance. Security requirements intensify, demanding robust encryption and audit logs to protect the large dataset (Sections 5.3 and 5.4).

Serving a million users nationally (26 TB storage, 80 QPS peak) makes the $710,000 monthly API bill the dominant constraint. Simply scaling the infrastructure is insufficient; architectural changes driven by cost become paramount. A key strategy involves fine-tuning open-source language models on healthcare data to handle generation internally, drastically cutting recurring inference costs compared to the external OpenAI API. This trades substantial upfront engineering investment and internal infrastructure complexity for GPU rental costs and long-term savings. It also allows removing the need to send sensitive patient data to an external AI vendor, reducing data privacy and security risk. Multi-region deployment becomes necessary for reliability and latency requirements. The system clearly falls under regulations like U.S. Food and Drug Administration (FDA) Software as a Medical Device (SaMD) [19] or the European Union AI Act [13], mandating rigorous validation and bias monitoring.

A global deployment reaching one billion users faces extreme challenges. Storage requirements hit 26 petabytes (PB) and peak QPS in the tens of thousands. To reduce server load our system now exclusively uses in-house fine-tuned language models, processing roughly 80% of queries on-device using small quantized models, and the remaining 20% escalated to a central inference cluster. This architectural shift—toward federation and edge RAG—creates immense challenges in distributed systems management, edge security, and ensuring equity across diverse hardware. Optimizations like embedding compression and delta updates become mandatory. The energy draw from training and running these models begins to raise environmental concerns. However, compliance is the primary concern, with global data localization laws, international regulatory regimes, and risks such as model monoculture shaping everyday operations.

This scaling journey reveals that moving from prototype to population requires navigating successive breakpoints where the primary challenge shifts—from basic concurrency to vector search performance, then to API cost, and ultimately to the limits of centralized cloud models. Each solution introduces new complexities and trade-offs across technical, financial, and ethical domains. The prototype's value lies in demonstrating core feasibility and establishing an evaluation framework that can be continuously re-run as the system scales and evolves—surfacing foundational issues, such as the AI reasoning bottleneck, that demand sustained attention for

responsible deployment.

6.6 Project Reflections

When we began this dissertation the goal was ambitious but simple: build a system that lets users "talk" to their medical documents and study how RAG behaves under controlled, reproducible conditions. Eighteen weeks later we have a working, auditable application and an evaluation harness that can swap in fresh embedding models or AI models with a single configuration file. I am most proud of that end-to-end artefact. The chat interface may look modest, yet beneath it lies a data-pipeline tracer, a latency profiler, and a scoring rig that re-grades every answer against multiple criteria. Knowing I can drop in tomorrow's open-weight model and immediately generate a full performance report feels like a durable contribution—one that outlives the specific numbers in this document.

The most intellectually rewarding moments came when the literature leapt off the page and into my console logs. Papers on primacy–recency bias, calibration failure, and fact-fabrication all predicted trouble, and the evaluation proved them right. Conversely, the recent idea of "AI-as-Judge" suggested letting a model pre-label retrieval relevance before human review; implementing that trick may have cut annotation time by half. These experiences reshaped my view of academic reading: citations are not rhetorical ornaments but design inputs, and the gulf between theory and practice narrows when you invite the literature into your build loop.

Two software engineering lessons now feel visceral. First, simplicity scales; clever abstractions did not accelerate progress nearly as much as a clear, modular boundary between ingestion, retrieval, generation, and evaluation. Second, scope is an engineering parameter, not a managerial after-thought. Every week spent refining the factorial grid was a week not available for something else, and recognizing that trade-off early saved the project from drowning in half-finished features.

My perception of RAG systems has also changed. I once assumed retrieval was the bottleneck; after watching a model hallucinate in the face of perfect evidence, I see that overconfidence and poor calibration are the real barriers to trust. The experience has made me a more cautious consumer of AI systems in daily life—quicker to ask what evidence underpins a fluent answer and more willing to accept "I don't know" as the safest output.

Finally, the project reframed how I see myself as an engineer–researcher. Depth emerged from iteration, not from grand architectural leaps; each tight feedback loop—write code, run experiments, read failures—moved the work further than a week of speculation. I also discovered that scepticism is productive: asking "how do we know this works?" before writing the next line of code prevented hours of misdirected effort. Most of all, I learned that focus is a super-power. Cutting scope felt like surrender at first, yet it created the space to polish what mattered and to finish on time with a system, an experiment, and a narrative that fit together.

7 Conclusion

Personal health information still lives in frustrating silos: laboratory PDFs here, scanned letters there, and clinic notes buried in proprietary portals. At the outset we asked whether a retrieval-augmented assistant could help users interrogate that fragmented record while preserving privacy. Across the preceding chapters we have answered in the affirmative, and in doing so have delivered four tightly coupled contributions.

First, we built a transparent, modular pipeline that ingests PDFs, normalizes text, embeds overlapping chunks and retrieves evidence on demand. The entire stack runs on a laptop, stores every artefact in `SQLite` and exposes a single adapter layer for swapping models—concretely realizing our architectural objective. Second, we executed a full-factorial experiment sweeping chunk size, retrieval depth and embedding model over 540 query–configuration pairs. The resulting dataset let us isolate the physics of chunking from the economics of token cost, fulfilling the experimental objective. Third, we derived a fine-grained failure-mode taxonomy—missed context, wrong generation and contradictory data—and traced each error to either retrieval or reasoning, meeting our analytical objective. Fourth, we threaded ethical reflection through design and evaluation, demonstrating how local storage, provenance links and calibrated uncertainty operationalize WHO principles of autonomy, safety, and accountability.

How do chunk size, top-k and embedding choice shape performance? For single-fact look-ups, 100-token chunks with $k = 3$ gave 97% precision while keeping prompts lean. For multi-document reasoning, 500-token chunks improved recall but only when the generator could still synthesize; larger chunks simply inflated latency and provoked "lost-in-the-middle" errors. Chunk size, not the embedding model, was the dominant driver of retrieval precision, while the more expensive `text-embedding-3-large` embedding model boosted correctness only in synthesis tasks.

Which parameters correlate with failure modes? Small chunks create missed-context errors when evidence straddles a boundary; wide chunks cure that but trigger wrong-generation failures because irrelevant detail distracts the model. Contradictory-data errors arise not from retrieval but from the language model's overconfidence when faced with conflicting snippets.

How do design choices affect cost and latency? Token usage grows linearly with k and chunk size, yet the majority of end-to-end latency is caused by the AI-model call; thrift therefore lies in trimming context rather than accelerating cosine search. A shallow k with disciplined chunking strikes the best balance between user-perceived speed and budget.

What trade-offs emerge? Precision, recall, and cognitive load cannot be optimized simultaneously. Adaptive chunking—small for factual queries, larger for multi-document reasoning—and dynamic k repay their complexity by sidestepping one-size-fits-all penalties.

Our findings rest on deliberate constraints: synthetic Synthea records, single-user operation, two embedding models and `GPT-4o-mini`. Real EHRs carry OCR noise, handwriting artefacts and longitudinal contradictions that our corpus omits. We evaluated offline queries, not interactive dialogue, and we stored embeddings in clear text. These boundaries keep the work feasible for a master's dissertation yet limit ecological validity.

Ethically, the project shows that infrastructure choices are moral choices: a local-first design narrows the privacy attack surface, but cloud-based embeddings re-introduce exposure. Our evaluation also confirms that document-grounding constrains—but does not eliminate—fact fabrication; confident error remains the chief clinical hazard. Mitigations therefore extend beyond better retrieval: they require reasoning-optimized prompting, contradiction detectors, calibrated confidence scores and, critically, a clinician-in-the-loop workflow.

Technically, the next step is realism. We must move from synthetic PDFs to genuine patient records, from single-shot queries to longitudinal conversations and from text-only evidence to multimodal inputs. That shift will demand ethics approval, on-premise language models and efficient annotation pipelines, but it is essential if retrieval-augmented assistants are ever to earn clinical trust.

Eighteen weeks ago we framed scope, simplicity, and scepticism as guiding principles. They proved decisive. By pruning ambition early we produced a system that not only works but can be reasoned about; by logging every prompt and latency reading we turned anecdote into evidence; by asking "How do we know?" before "What else can we add?" we converted curiosity into disciplined discovery. Our local-first RAG assistant is therefore more than a prototype: it is a reproducible research instrument that others can extend, critique and deploy. If future iterations pair stronger reasoning with the safeguards outlined here, Retrieval-Augmented Generation could indeed move from promising toy to reliable clinical tool—helping patients and professionals alike to see coherence where today they see only fragments.

Bibliography

[1] Mohammad Alkhalaf et al. “Applying generative AI with retrieval augmented generation to summarize and extract key clinical information from electronic health records”. In: *Journal of biomedical informatics* 156 (2024), p. 104662.

[2] Emily Alsentzer et al. “Publicly available clinical BERT embeddings”. In: *arXiv preprint arXiv:1904.03323* (2019).

[3] Christoph Aluttis, Tewabech Bishaw, and Martina W Frank. “The workforce for health in a globalized context–global shortages and international migration”. In: *Global health action* 7.1 (2014), p. 23611.

[4] Ana I Balsa et al. “Clinical uncertainty and healthcare disparities”. In: *American Journal of Law & Medicine* 29.2-3 (2003), pp. 203–219.

[5] Suhana Bedi et al. “Testing and evaluation of health care applications of large language models: a systematic review”. In: *JAMA* (2024).

[6] Patrick Emanuel Beeler, David Westfall Bates, and Balthasar Luzius Hug. “Clinical decision support systems”. In: *Swiss medical weekly* 144 (2014), w14073.

[7] British Computer Society. *BCS Code of Conduct*. Accessed: 2025-04-28. June 2022. URL: https://www.bcs.org/media/2211/bcs-code-of-conduct.pdf.

[8] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[9] Charles JT Butcher and Wajid Hussain. “Digital healthcare: the future”. In: *Future healthcare journal* 9.2 (2022), pp. 113–117.

[10] Leo Anthony Celi et al. “Sources of bias in artificial intelligence that perpetuate healthcare disparities—A global review”. In: *PLOS Digital Health* 1.3 (2022), e0000022.

[11] Richard J Chen et al. “Algorithmic fairness in artificial intelligence for medicine and healthcare”. In: *Nature biomedical engineering* 7.6 (2023), pp. 719–742.

[12] Norman Daniels. “Justice, health, and healthcare”. In: *American Journal of Bioethics* 1.2 (2001), pp. 2–16.

[13] European Union. *EU Artificial Intelligence Act)*. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng. Published in the Official Journal of the European Union, L 2024/1689, 12 July 2024. Accessed: 30 April 2025. July 2024.

[14] Google. *Gemini Developer API Pricing*. Accessed: 2025-04-28. Apr. 2025. URL: https://ai.google.dev/gemini-api/docs/pricing.

[15] Jianping Gou et al. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[16] WHO Guidance. “Ethics and governance of artificial intelligence for health”. In: *World Health Organization* (2021).

[17] Kelvin Guu et al. “Retrieval augmented language model pre-training”. In: *International conference on machine learning*. PMLR. 2020, pp. 3929–3938.

[18] Karen Hacker. “The burden of chronic disease”. In: *Mayo Clinic Proceedings: Innovations, Quality & Outcomes* 8.1 (2024), pp. 112–119.

[19] International Medical Device Regulators Forum. *Software as a Medical Device (SaMD): Clinical Evaluation*. Tech. rep. IMDRF/SaMD WG/N41FINAL:2017. Accessed: 2025-04-30. U.S. Food and Drug Administration, June 2017. URL: https://www.fda.gov/media/100714/download.

[20] Ziwei Ji et al. “Survey of hallucination in natural language generation”. In: *ACM computing surveys* 55.12 (2023), pp. 1–38.

[21] Zhengbao Jiang et al. “How can we know when language models know? on the calibration of language models for question answering”. In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 962–977.

[22] Jared Kaplan et al. “Scaling laws for neural language models”. In: *arXiv preprint arXiv:2001.08361* (2020).

[23] Vladimir Karpukhin et al. “Dense Passage Retrieval for Open-Domain Question Answering.” In: *EMNLP (1)*. 2020, pp. 6769–6781.

[24] Satyapriya Krishna et al. “Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation”. In: *arXiv preprint arXiv:2409.12941* (2024).

[25] Jinhyuk Lee et al. “BioBERT: a pre-trained biomedical language representation model for biomedical text mining”. In: *Bioinformatics* 36.4 (2020), pp. 1234–1240.

[26] Haoran Li, Mingshi Xu, and Yangqiu Song. “Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence”. In: *arXiv preprint arXiv:2305.03010* (2023).

[27] Nelson F Liu et al. “Lost in the middle: How language models use long contexts”. In: *Transactions of the Association for Computational Linguistics* 12 (2024), pp. 157–173.

[28] Yi Liu et al. “Prompt Injection attack against LLM-integrated Applications”. In: *arXiv preprint arXiv:2306.05499* (2023).

[29] Negar Maleki, Balaji Padmanabhan, and Kaushik Dutta. “AI hallucinations: a misnomer worth clarifying”. In: *2024 IEEE conference on artificial intelligence (CAI)*. IEEE. 2024, pp. 133–138.

[30] OpenAI. *How can I get a Business Associate Agreement (BAA) with OpenAI for the API Services?* Accessed: 2025-04-28. Mar. 2025. URL: https://help.openai.com/en/articles/8660679-how-can-i-get-a-business-associate-agreement-baa-with-openai-for-the-api-services.

[31] OpenAI. *Introducing GPT-4.1 in the API*. Accessed: 2025-04-28. Apr. 2025. URL: https://platform.openai.com/docs/models/gpt-4.1.

[32] OpenAI. *New embedding models and API updates*. Accessed: 2025-04-28. Jan. 2024. URL: https://openai.com/index/new-embedding-models-and-api-updates/.

[33] OpenAI. *OpenAI API Pricing*. Accessed: 2025-04-28. Apr. 2025. URL: https://platform.openai.com/docs/pricing.

[34] OpenAI. *text-embedding-3-small*. Accessed: 2025-04-28. Apr. 2025. URL: https://platform.openai.com/docs/models/text-embedding-3-small.

[35] Irene Papanicolas, Liana R Woskie, and Ashish K Jha. "Health care spending in the United States and other high-income countries". In: *Jama* 319.10 (2018), pp. 1024–1039.

[36] Jianing Qiu et al. "LLM-based agentic systems in medicine and healthcare". In: *Nature Machine Intelligence* 6.12 (2024), pp. 1418–1420.

[37] Renyi Qu, Ruixuan Tu, and Forrest Bao. "Is Semantic Chunking Worth the Computational Cost?" In: *arXiv preprint arXiv:2410.13070* (2024).

[38] Rafael Rafailov et al. "Direct preference optimization: Your language model is secretly a reward model". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 53728–53741.

[39] Royal Academy of Engineering. *Ethics in the Engineering Profession*. Accessed: 2025-04-28. June 2023. URL: https://raeng.org.uk/media/x0lbgvco/ethics-in-the-engineering-profession.pdf.

[40] Jon Saad-Falcon et al. "Ares: An automated evaluation framework for retrieval-augmented generation systems". In: *arXiv preprint arXiv:2311.09476* (2023).

[41] Glorin Sebastian. "Privacy and data protection in chatgpt and other ai chatbots: Strategies for securing user information". In: *International Journal of Security and Privacy in Pervasive Computing (IJSPPC)* 15.1 (2023), pp. 1–14.

[42] Tait D Shanafelt et al. "Burnout and satisfaction with work-life balance among US physicians relative to the general US population". In: *Archives of internal medicine* 172.18 (2012), pp. 1377–1385.

[43] Karan Singhal et al. "Large language models encode clinical knowledge". In: *arXiv preprint arXiv:2212.13138* (2022).

[44] Nisan Stiennon et al. "Learning to summarize with human feedback". In: *Advances in neural information processing systems* 33 (2020), pp. 3008–3021.

[45] Nandan Thakur et al. "Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models". In: *arXiv preprint arXiv:2104.08663* (2021).

[46] Shubo Tian et al. "Opportunities and challenges for ChatGPT and large language models in biomedicine and health". In: *Briefings in Bioinformatics* 25.1 (2024), bbad493.

[47] Faheem Ullah et al. "Data exfiltration: A review of external attack vectors and countermeasures". In: *Journal of Network and Computer Applications* 101 (2018), pp. 18–54.

[48] Jason Walonoski et al. "Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record". In: *Journal of the American Medical Informatics Association* 25.3 (2018), pp. 230–238.

[49] Changyu Wang et al. "Ethical considerations of using ChatGPT in health care". In: *Journal of Medical Internet Research* 25 (2023), e48009.

[50] Lei Wang et al. "Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models". In: *arXiv preprint arXiv:2305.04091* (2023).

[51] Jason Wei et al. "Chain-of-thought prompting elicits reasoning in large language models". In: *Advances in neural information processing systems* 35 (2022), pp. 24824–24837.

[52] Jules White et al. "A prompt pattern catalog to enhance prompt engineering with chatgpt". In: *arXiv preprint arXiv:2302.11382* (2023).

[53] Derek Yach et al. “The global burden of chronic diseases: overcoming impediments to prevention and control”. In: *Jama* 291.21 (2004), pp. 2616–2622.

[54] Quan Zhang et al. “Human-imperceptible retrieval poisoning attacks in LLM-powered applications”. In: *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. 2024, pp. 502–506.

[55] Zhan Zhang et al. “Effect of AI explanations on human perceptions of patient-facing AI-powered healthcare systems”. In: *Journal of Medical Systems* 45.6 (2021), p. 64.

[56] Lianmin Zheng et al. “Judging llm-as-a-judge with mt-bench and chatbot arena”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 46595–46623.

A Breakdown of RAG Pipeline Performance

Table A.1: Performance by Configuration

Query Type	Configuration (Chunk, k, Emb)	Retrieval Precision (mean ± SE)	Correctness (mean ± SE)	Tokens (mean ± SD)	Retrieval Latency (ms, mean ± SD)	Generation Latency (ms, mean ± SD)
Contradiction	100-3-large	0.87 ± 0.06	0.20 ± 0.11	362 ± 9	554 ± 133	609 ± 101
Contradiction	100-3-small	0.97 ± 0.03	0.33 ± 0.13	358 ± 7	638 ± 533	564 ± 80
Contradiction	100-5-large	0.97 ± 0.03	0.07 ± 0.07	581 ± 18	478 ± 88	648 ± 164
Contradiction	100-5-small	0.97 ± 0.03	0.20 ± 0.11	586 ± 9	413 ± 210	640 ± 134
Contradiction	500-3-large	1.00	0.13 ± 0.09	1257 ± 149	358 ± 65	890 ± 830
Contradiction	500-3-small	1.00	0.07 ± 0.07	1286 ± 152	322 ± 118	638 ± 92
Contradiction	500-5-large	1.00	0.07 ± 0.07	2099 ± 270	361 ± 68	885 ± 618
Contradiction	500-5-small	1.00	0.07 ± 0.07	2112 ± 243	325 ± 126	763 ± 202
Contradiction	1000-3-large	1.00	0.13 ± 0.09	1588 ± 288	303 ± 59	715 ± 246
Contradiction	1000-3-small	1.00	0.07 ± 0.07	1745 ± 274	830 ± 1452	863 ± 511
Contradiction	1000-5-large	1.00	0.07 ± 0.07	3079 ± 535	557 ± 912	634 ± 106
Contradiction	1000-5-small	1.00	0.10 ± 0.07	3070 ± 475	345 ± 112	620 ± 118
Factual	100-3-large	0.93 ± 0.07	0.80 ± 0.11	328 ± 32	381 ± 146	604 ± 148
Factual	100-3-small	0.93 ± 0.05	0.83 ± 0.09	340 ± 17	408 ± 126	643 ± 160
Factual	100-5-large	1.00	0.87 ± 0.09	540 ± 55	419 ± 187	673 ± 142
Factual	100-5-small	0.97 ± 0.03	0.87 ± 0.09	557 ± 22	366 ± 170	707 ± 121
Factual	500-3-large	0.97 ± 0.03	0.80 ± 0.11	1237 ± 257	362 ± 110	773 ± 163
Factual	500-3-small	0.90 ± 0.07	0.67 ± 0.13	1274 ± 192	391 ± 336	673 ± 102
Factual	500-5-large	1.00	0.80 ± 0.11	2071 ± 354	324 ± 70	734 ± 307
Factual	500-5-small	1.00	0.80 ± 0.11	2082 ± 189	393 ± 204	695 ± 146
Factual	1000-3-large	1.00	0.83 ± 0.09	1805 ± 478	391 ± 126	914 ± 431
Factual	1000-3-small	0.87 ± 0.09	0.67 ± 0.13	1861 ± 478	375 ± 129	772 ± 135
Factual	1000-5-large	1.00	0.73 ± 0.12	3088 ± 406	329 ± 93	762 ± 144
Factual	1000-5-small	1.00	0.80 ± 0.11	3174 ± 478	368 ± 258	966 ± 693
Multi-document	100-3-large	0.60 ± 0.09	0.13 ± 0.09	345 ± 17	599 ± 684	605 ± 141
Multi-document	100-3-small	0.40 ± 0.07	0.00	333 ± 35	468 ± 310	523 ± 74
Multi-document	100-5-large	0.63 ± 0.06	0.13 ± 0.09	557 ± 31	646 ± 1136	574 ± 94
Multi-document	100-5-small	0.57 ± 0.08	0.10 ± 0.07	550 ± 39	377 ± 220	620 ± 158
Multi-document	500-3-large	0.80 ± 0.08	0.13 ± 0.09	1228 ± 211	291 ± 83	593 ± 81
Multi-document	500-3-small	0.80 ± 0.08	0.03 ± 0.03	1278 ± 217	476 ± 673	678 ± 146
Multi-document	500-5-large	0.87 ± 0.08	0.33 ± 0.13	2095 ± 220	342 ± 127	850 ± 426
Multi-document	500-5-small	0.83 ± 0.08	0.10 ± 0.07	2037 ± 194	389 ± 353	711 ± 124
Multi-document	1000-3-large	0.87 ± 0.06	0.13 ± 0.08	1951 ± 462	342 ± 100	717 ± 223
Multi-document	1000-3-small	0.67 ± 0.09	0.03 ± 0.03	1900 ± 449	525 ± 902	1171 ± 1222
Multi-document	1000-5-large	0.83 ± 0.08	0.17 ± 0.09	3167 ± 319	349 ± 97	879 ± 538
Multi-document	1000-5-small	0.83 ± 0.06	0.10 ± 0.07	3065 ± 407	324 ± 99	729 ± 128

B Breakdown of Failure Modes

Table B.1: Failure Mode Distribution by Configuration

Query Type	Configuration (Chunk, k, Emb)	Contradictory Data (%)	Correct Retrieval but Wrong Gen (%)	Missed Context (%)	No Failure (%)
Contradiction	100-3-large	73.3	6.7	0.0	20.0
Contradiction	100-3-small	60.0	0.0	6.7	33.3
Contradiction	100-5-large	93.3	0.0	0.0	6.7
Contradiction	100-5-small	73.3	0.0	6.7	20.0
Contradiction	500-3-large	80.0	6.7	0.0	13.3
Contradiction	500-3-small	73.3	20.0	0.0	6.7
Contradiction	500-5-large	86.7	6.7	0.0	6.7
Contradiction	500-5-small	86.7	6.7	0.0	6.7
Contradiction	1000-3-large	80.0	6.7	0.0	13.3
Contradiction	1000-3-small	86.7	6.7	0.0	6.7
Contradiction	1000-5-large	86.7	6.7	0.0	6.7
Contradiction	1000-5-small	80.0	13.3	0.0	6.7
Factual	100-3-large	0.0	13.3	6.7	80.0
Factual	100-3-small	0.0	6.7	13.3	80.0
Factual	100-5-large	0.0	13.3	0.0	86.7
Factual	100-5-small	0.0	6.7	6.7	86.7
Factual	500-3-large	0.0	13.3	6.7	80.0
Factual	500-3-small	0.0	26.7	6.7	66.7
Factual	500-5-large	0.0	20.0	0.0	80.0
Factual	500-5-small	0.0	20.0	0.0	80.0
Factual	1000-3-large	0.0	20.0	0.0	80.0
Factual	1000-3-small	0.0	20.0	13.3	66.7
Factual	1000-5-large	0.0	26.7	0.0	73.3
Factual	1000-5-small	0.0	20.0	0.0	80.0
Multi-document	100-3-large	0.0	33.3	53.3	13.3
Multi-document	100-3-small	0.0	6.7	93.3	0.0
Multi-document	100-5-large	0.0	20.0	66.7	13.3
Multi-document	100-5-small	0.0	26.7	66.7	6.7
Multi-document	500-3-large	0.0	53.3	33.3	13.3
Multi-document	500-3-small	0.0	80.0	20.0	0.0
Multi-document	500-5-large	0.0	60.0	6.7	33.3
Multi-document	500-5-small	0.0	73.3	20.0	6.7
Multi-document	1000-3-large	0.0	66.7	26.7	6.7
Multi-document	1000-3-small	0.0	60.0	40.0	0.0
Multi-document	1000-5-large	0.0	73.3	13.3	13.3
Multi-document	1000-5-small	0.0	73.3	20.0	6.7

C Full AI Model System Prompt

This appendix contains the full system prompt used for the AI model integration described in Section 3.4.

```
"""
You are a document-grounded AI assistant for health assistance.
You help users understand their medical records, test results,
and recommend evidence-based approaches to maximize healthspan
and lifespan. While not a doctor, your role is to analyse data
from users' medical documents and provide personalized insights
based solely on retrieved evidence.

Core Responsibilities:
- Ground all responses in retrieved document content; do not
  fabricate information.
- When evidence is incomplete or contradictory, clearly explain
  the limitations of available data.
- Translate medical terminology into accessible language while
  preserving accuracy.
- Balance scientific rigor with compassionate, practical guidance.
- Respect your limitations - for diagnosis, treatment plans, or
  urgent medical concerns, advise consultation with healthcare
  professionals.
- When discussing longevity interventions, clearly distinguish
  between well-established approaches and emerging research.

Instructions:
Use the document excerpts verbatim to answer the user's query.
Cite sources inline using metadata. Do not fabricate or add
information not present in the excerpts. If excerpts conflict or
are incomplete:
- Describe the inconsistency.
- Outline what additional information is needed.
- Recommend consulting a healthcare professional.

You may:
- Explain relevant anatomy, physiology, risk factors, and typical
diagnostic approaches (informational only).
- Suggest questions for the user to ask their doctor.
- Recommend evidence-based longevity practices when relevant to
  the query.
```

```
- Analyse trends in health markers across multiple documents
  when available.

Never:
- Provide diagnoses, treatment plans, medication advice, legal or
  insurance guidance.
- Confirm or deny a medical condition.

Always:
- Emphasize traceability and safety.
- Acknowledge uncertainty and your limitations.

Safety Protocol:
If the user describes symptoms suggesting a medical emergency
(e.g., chest pain, stroke symptoms, severe bleeding, sudden
severe pain), respond: "This may be a medical emergency. Please
seek immediate medical care or call emergency services."

<case_logs>
{case_logs}
</case_logs>
"""
```

D Scaling Assumptions

Table D.1: System Assumptions and Derived Metrics

Parameter/Metric	Assumption/Value	Notes/Derivation
Usage Assumptions		
Queries per User per Month	20	Estimated user activity
Peak Query Factor	10x	Assumed ratio of peak to average load
Corpus Size for Re-embedding	25,000 tokens	50 docs × 500 tokens/doc
Model & Pricing Assumptions		
AI Generation Model	`o3`	OpenAI model assumed for generation
AI Model Input Cost Rate	$10.00 / 1M tokens	OpenAI pricing
AI Model Output Cost Rate	$40.00 / 1M tokens	OpenAI pricing
Embedding Model	`text-embedding-3-large`	Model used for retrieval embeddings
Embedding Dimensions	3072	Dimensionality of model output
Embedding Cost Rate	$0.13 / 1M tokens	OpenAI pricing
Derived Per-User Metrics		
AI Model Input Tokens per Query	1500 tokens	Estimated from prompt structure
AI Model Output Tokens per Query	500 tokens	Estimated average answer length
AI Model Input Cost / User / Month	$0.30	$20 \times 1500/10^6 \times 10$
AI Model Output Cost / User / Month	$0.40	$20 \times 500/10^6 \times 40$
Embedding Cost / User / Month	$0.00	$25,000/10^6 \times 0.13$
Total API Cost / User / Month	$0.703 (≈ $0.71)	Sum of above, infra excluded
Chunks per User (Estimate)	100	Derived from corpus size
Embedding Vector Size (fp32)	12.3 kB	3072×4 bytes
Embeddings Storage / User	1.2 MB	100×12.3 kB
Raw Document Storage / User	25 MB	50 docs × 500 KB/doc
Total Storage / User	26 MB	Sum of raw and vector storage
Average QPS / User	7.7×10^{-6} QPS	20 queries / month in seconds
Peak QPS / User	7.7×10^{-5} QPS	Avg QPS × 10x peak factor

E Synthetic Patient Record PDFs

This appendix provides example synthetic patient record PDFs for synthetic patient `Angelita915 Bauch723` during system evaluation. The previously provided example, Figure 4.1, shows the top half of an example report, showing a document header and preamble before the details of the record. The following examples focus on the content after the preamble.

Figure E.1 displays an example where we see the patient's current medications. There's a title for the table and a table with three columns of data. Some text wraps within its row. Figure E.2 displays lab results in a table with a footnote after. Figure E.3 is a "contradictory" version of this report which is injected into the corpus only when a "contradiction" query is being tested during evaluation mode. The report is identical, but the results have different values. Figure E.5 shows a synthetic imaging report, which demonstrates that our OCR system doesn't break when an image is in an ingested PDF. Figure E.5 shows a table broken over two pages, which our ingestion pipeline must handle. Figure E.6 shows another edge case: a column of text where a single word, "Completed," is wrapped three times in the same cell and must be handled during ingestion.

Current Medications

Medication	Status	Prescribed Date
NuvaRing 0.12/0.015 MG per 24HR 21 Day Vaginal Ring	Active	2021-07-06 10:46 ET
lisinopril 10 MG Oral Tablet	Active	2020-09-26 10:46 ET
Etonogestrel 68 MG Drug Implant	Stopped	2020-07-10 10:46 ET
lisinopril 10 MG Oral Tablet	Stopped	2019-09-21 10:46 ET
lisinopril 10 MG Oral Tablet	Stopped	2018-09-15 10:46 ET
1 ML medroxyprogesterone acetate 150 MG/ML Injection	Stopped	2018-07-20 10:46 ET
lisinopril 10 MG Oral Tablet	Stopped	2017-09-09 10:46 ET

Figure E.1: Synthetic PDF: Medications

Laboratory Test Results

Patient: Angelita915 Bauch723

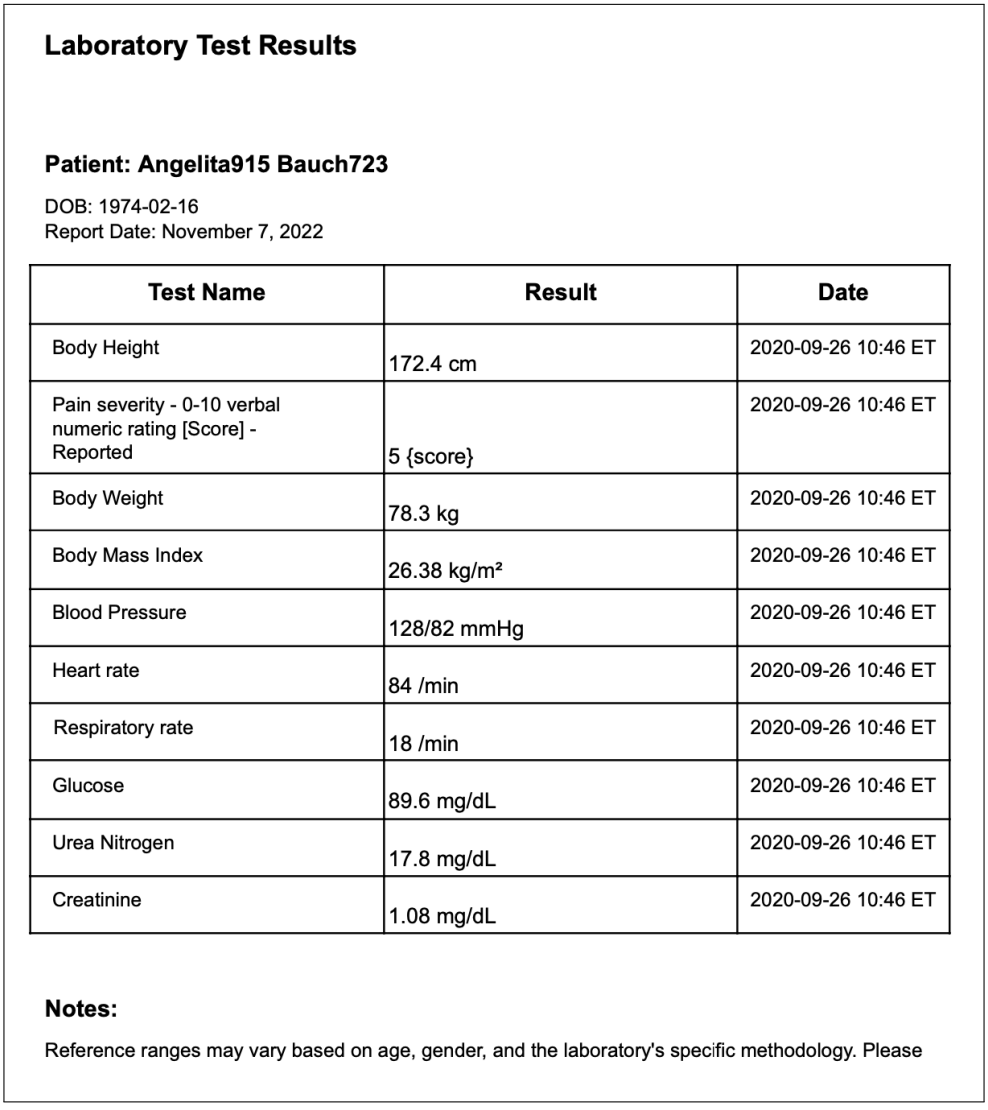
DOB: 1974-02-16
Report Date: November 7, 2022

Test Name	Result	Date
Body Height	166.9 cm	2020-09-26 10:46 ET
Pain severity - 0-10 verbal numeric rating [Score] - Reported	3 {score}	2020-09-26 10:46 ET
Body Weight	85 kg	2020-09-26 10:46 ET
Body Mass Index	30.51 kg/m2	2020-09-26 10:46 ET
Blood Pressure	Unknown	2020-09-26 10:46 ET
Heart rate	92 /min	2020-09-26 10:46 ET
Respiratory rate	16 /min	2020-09-26 10:46 ET
Glucose	72.43 mg/dL	2020-09-26 10:46 ET
Urea Nitrogen	14.23 mg/dL	2020-09-26 10:46 ET
Creatinine	0.94 mg/dL	2020-09-26 10:46 ET

Notes:

Reference ranges may vary based on age, gender, and the laboratory's specific methodology. Please consult with your healthcare provider to interpret these results.

Figure E.2: Synthetic PDF: Lab Results



Laboratory Test Results

Patient: Angelita915 Bauch723

DOB: 1974-02-16
Report Date: November 7, 2022

Test Name	Result	Date
Body Height	172.4 cm	2020-09-26 10:46 ET
Pain severity - 0-10 verbal numeric rating [Score] - Reported	5 {score}	2020-09-26 10:46 ET
Body Weight	78.3 kg	2020-09-26 10:46 ET
Body Mass Index	26.38 kg/m²	2020-09-26 10:46 ET
Blood Pressure	128/82 mmHg	2020-09-26 10:46 ET
Heart rate	84 /min	2020-09-26 10:46 ET
Respiratory rate	18 /min	2020-09-26 10:46 ET
Glucose	89.6 mg/dL	2020-09-26 10:46 ET
Urea Nitrogen	17.8 mg/dL	2020-09-26 10:46 ET
Creatinine	1.08 mg/dL	2020-09-26 10:46 ET

Notes:

Reference ranges may vary based on age, gender, and the laboratory's specific methodology. Please

Figure E.3: Synthetic PDF: Lab Results Contradiction

Imaging Studies Report

Patient: Angelita915 Bauch723

DOB: 1974-02-16
Report Date: November 7, 2022

Imaging Studies Summary

Modality	Date	Status
Magnetic Resonance	2003-06-10 03:23 ET	Available

Representative Image

Below is a representative image from the patient's most recent imaging study:

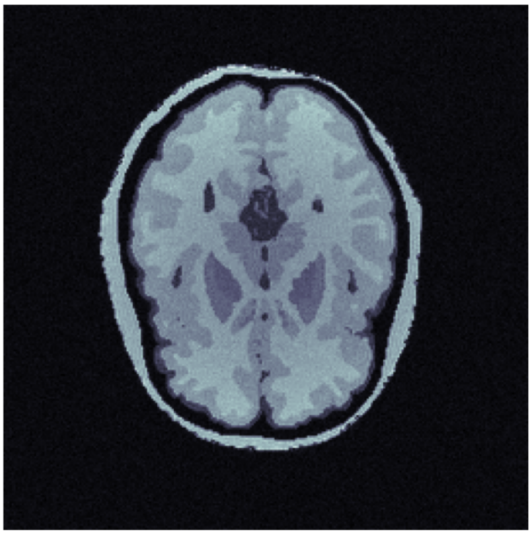
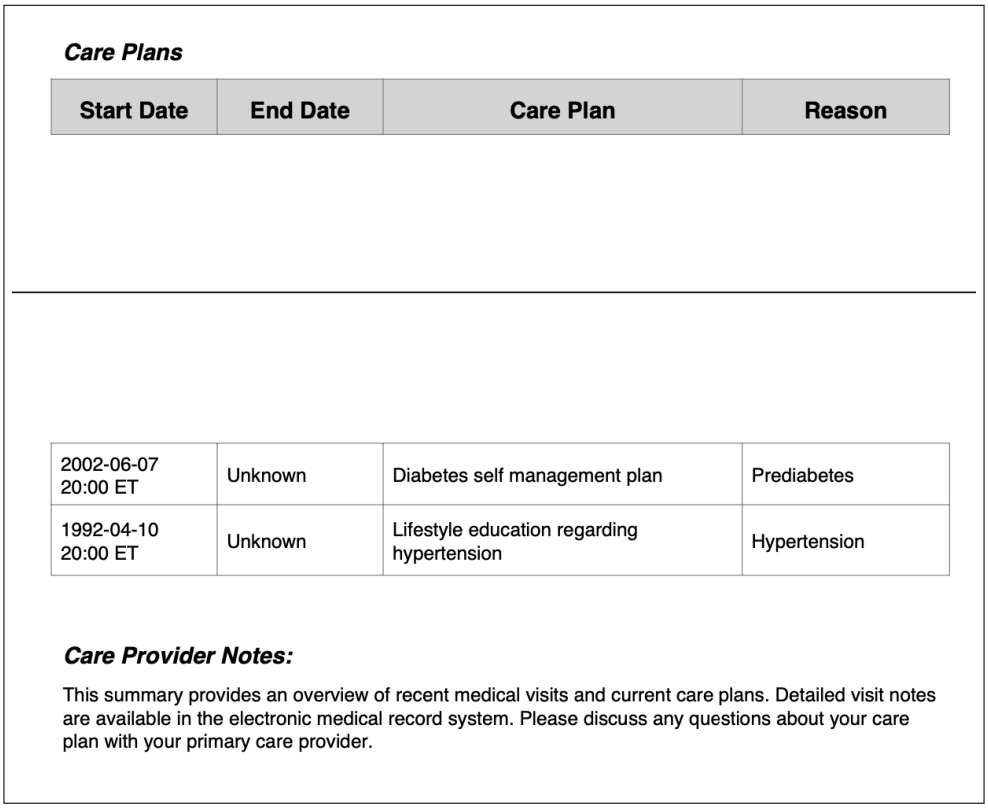


Figure E.4: Synthetic PDF: Imaging Studies



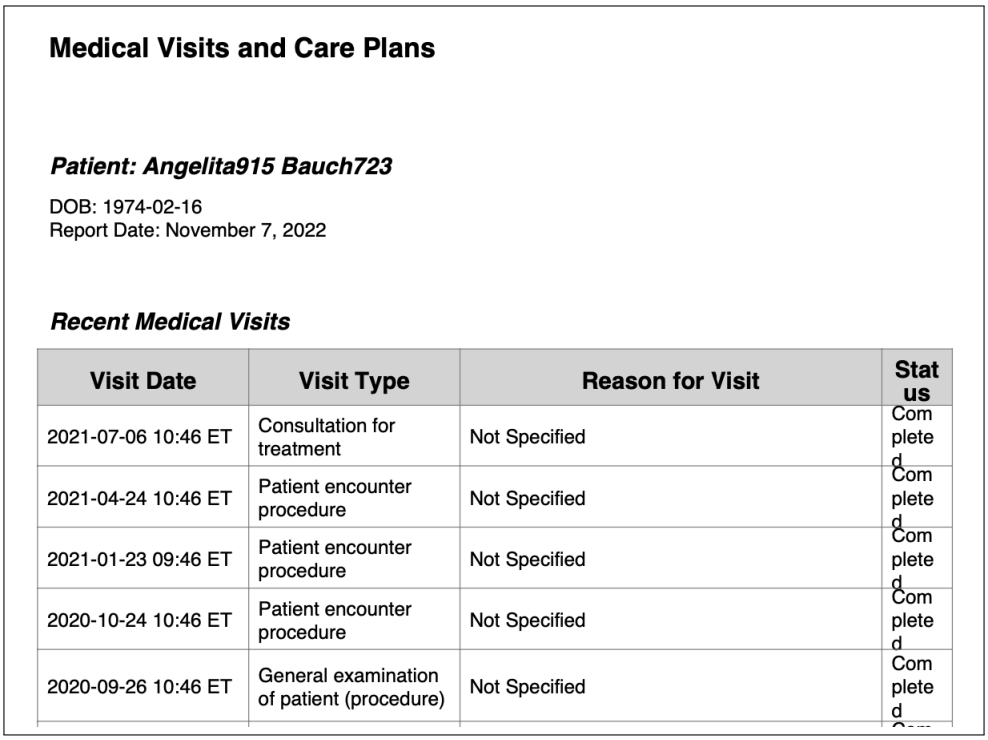
Care Plans

Start Date	End Date	Care Plan	Reason
2002-06-07 20:00 ET	Unknown	Diabetes self management plan	Prediabetes
1992-04-10 20:00 ET	Unknown	Lifestyle education regarding hypertension	Hypertension

Care Provider Notes:

This summary provides an overview of recent medical visits and current care plans. Detailed visit notes are available in the electronic medical record system. Please discuss any questions about your care plan with your primary care provider.

Figure E.5: Synthetic PDF: Care Plans



Medical Visits and Care Plans

Patient: Angelita915 Bauch723

DOB: 1974-02-16
Report Date: November 7, 2022

Recent Medical Visits

Visit Date	Visit Type	Reason for Visit	Status
2021-07-06 10:46 ET	Consultation for treatment	Not Specified	Completed
2021-04-24 10:46 ET	Patient encounter procedure	Not Specified	Completed
2021-01-23 09:46 ET	Patient encounter procedure	Not Specified	Completed
2020-10-24 10:46 ET	Patient encounter procedure	Not Specified	Completed
2020-09-26 10:46 ET	General examination of patient (procedure)	Not Specified	Completed

Figure E.6: Synthetic PDF: Recent Visits