Final Technical Documentation

Salud Revenue Partners

Dynamic Payer Knowledge Base

1. Project Overview

The Salud Dynamic Payer Knowledge Base Agent is an AI-powered policy intelligence platform designed to support healthcare organizations in navigating the complexity of payer policy documentation. Built using **Streamlit** and modern **Retrieval-Augmented Generation (RAG)** architecture, the system enables users to ingest, analyze, compare, and query large volumes of payer policy PDFs (e.g., Medicaid, Medicare Advantage, and commercial payer rules).

Healthcare policy teams traditionally rely on **manual reviews, static summaries, and spreadsheet-based tracking** to interpret payer rules. This approach is time-intensive, error-prone, and poorly suited for handling frequent policy updates across multiple payers.

Salud addresses this gap by:

- Structuring unstructured policy PDFs into searchable, versioned knowledge assets
- Enabling **natural language querying** with policy-grounded answers
- Providing **page-level citations and traceability** suitable for audit and compliance contexts
- Automatically detecting and summarizing policy changes across versions

The platform is designed to support **policy analysts, compliance teams, revenue cycle managers, and healthcare consultants** who require fast, reliable, and defensible access to payer policy information.

2. Core Functional Capabilities

2.1. Intelligent Policy Ingestion & Structuring

We implement a robust ingestion pipeline that transforms raw policy PDFs into structured, query-ready data assets.

Key capabilities include:

- Automated ingestion of policy PDFs organized by:
    - run_date: Timestamp of ingestion, enabling version control

    - payer_id: Unique identifier for each payer (e.g., Medicaid state, commercial insurer)

- Page-level PDF parsing to preserve contextual granularity
- Extraction and storage of structured metadata, including:
    - Source file name
    - Page number
    - Payer identifier
    - Ingestion run date
    - Extracted text content

Each policy document is decomposed into semantically meaningful text chunks, ensuring that downstream retrieval and citation remain precise.

Outcome:
Every policy is transformed into a fully indexed, metadata-rich knowledge object, enabling fast retrieval, historical comparison, and audit-grade traceability.

2.2. Semantic Search & Retrieval-Augmented Generation (RAG)

To overcome the limitations of keyword-based search, our project leverages **semantic embeddings** to capture the meaning of policy language.

Technical approach:

- Policy text is converted into dense vector embeddings using **Sentence-Transformers (MiniLM)**
- Embeddings are stored in a **Chroma vector database**, optimized for similarity search
- Queries are embedded in the same semantic space, enabling:
    - Conceptual matching (not just exact wording)
    - Retrieval of relevant policy sections even when terminology differs

Advanced retrieval controls:

- Metadata-based filtering:
    - Restrict searches to a specific payer

    - Filter by ingestion date or most recent policy version
- Top-k retrieval ensures only the most relevant policy passages are passed to the LLM

Outcome:
Before generating any response, the system retrieves **the most contextually relevant policy excerpts**, ensuring factual grounding and minimizing hallucinations.

2.3. Complexity, Uncertainty, and Sense-Making in Data-Driven Projects

Our approach integrates **Anthropic Claude** as its language generation engine, operating strictly within a RAG-based constraint system.

The platform supports **two distinct response modes**:

Mode	Description	Primary Use Case
Strict Mode	Generates answers exclusively from retrieved policy text, without external knowledge	Legal, compliance, audit, and regulatory workflows
Hybrid Mode	Combines retrieved policy text with general healthcare domain knowledge (clearly labeled)	Interpretive analysis, consulting discussions

Key safeguards:

- All answers are grounded in retrieved policy passages
- Inline citations reference:
    - Exact PDF file
    - Page number(s)
- Responses clearly differentiate between policy-derived facts and general context (Hybrid Mode only)

Outcome:
Users receive **trustworthy, explainable answers** that can be defended in compliance reviews and stakeholder discussions.

2.4. Change Intelligence & Policy Version Comparison

Healthcare payer policies change frequently, often with subtle wording modifications that have significant downstream impact. We are incorporating a dedicated Change Intelligence Engine to address this challenge.

How it works:

- Compares the latest ingestion run against prior runs for the same payer
- Identifies:
  - Added clauses
  - Removed sections
  - Modified policy language
- Summarizes changes in clear, actionable language
- Avoids speculative interpretation by anchoring all differences in source text

Use cases enabled:

- Rapid review of policy updates
- Proactive identification of reimbursement or compliance risks
- Reduced dependency on manual redlining and document diffing

Outcome:
Policy teams can track and understand policy evolution in minutes instead of hours, improving responsiveness and decision quality.

2.5. Traceability, Auditing & Compliance Readiness

Our project is designed with healthcare compliance standards in mind, emphasizing full transparency across the AI workflow.

Traceability features include:

- Direct linkage from AI-generated answer → retrieved text → original PDF
- In-app preview of cited policy pages
- Download access to original policy documents
- Display of contextual metadata (payer, run date, page number)

Audit support:

- Ensures all insights are verifiable

- Enables documentation of decision rationale
- Supports internal audits and external regulatory reviews

Outcome:
The system meets the expectations of **audit-ready, enterprise-grade healthcare analytics tooling**.

3. System Architecture Overview

User Query

↓

Streamlit UI (BigAgentFinal_Streamlit.py)

↓

Chroma Vector Database ← Policy Embeddings (MiniLM)

↓

Anthropic Claude API

↓

Policy-Grounded Answer + Citations + PDF Page Preview

3.1. Core Components

Component	Role
Streamlit UI	Front-end interface for document upload, querying, and visualization
PDF Parser	Extracts page-level text and metadata from policy PDFs
MiniLM	Converts policy text into semantic embeddings
Chroma DB	Stores and retrieves embeddings for semantic search
Claude LLM	Generate grounded, context-aware responses
Change Engine	Detects and summarizes policy updates across versions

4. Repository Structure & Design Rationale

File / Folder	Description
BigAgentFinal_Streamlit.py	Main application entry point managing UI, ingestion, retrieval, and response generation
requirements.txt	Python dependency list ensuring environment reproducibility
.streamlit/config.toml	UI configuration (custom Salud green theme)
Charlie Output/	Local data processing artifacts (excluded from version control)
Salud_main_1/	Root directory for all ingestion runs
Salud_main_1/<run_date>/<payer_id>/	Structured storage for versioned payer policies
README.md	Setup instructions and project overview

This structure enforces **clear separation between code, configuration, data, and outputs**, supporting maintainability and scalability.

5. Local Execution Instructions

1. Clone the repository

   git clone https://github.com/sarthakc123/BIG_KnowledgeBase.git

   cd BIG_KnowledgeBase

2. Install dependencies

   pip install -r requirements.txt

3. Launch the application

   streamlit run BigAgentFinal_Streamlit.py

4. Configure secrets

   - Navigate to **Streamlit → App Settings → Secrets**
   - Add:

     ANTHROPIC_API_KEY = "sk-ant-..."

6. Deploying Streamlit Cloud

- **Main file:** *BigAgentFinal_Streamlit.py*
- Secrets managed through Streamlit Cloud's secure environment
- Deployed application is accessible via a shareable URL
- No sensitive credentials are stored in the repository

7. Data Refresh & Re-Indexing Workflow

To update or expand the policy corpus:

1. Open *navigation.ipynb*
2. Update policy URLs or document sources
3. Insert Anthropic API key (securely)
4. Execute all cells to:
   - Re-ingest PDFs
   - Recompute embeddings
   - Update vector store

This design supports **continuous policy monitoring** with minimal manual intervention.

8. Security, Privacy & Compliance Considerations

- No hardcoded API keys or credentials
- All secrets managed via Streamlit's encrypted secrets manager
- Processes public payer policy documents only
- No handling of PHI or PII
- Full citation trail ensures explainability and accountability

9. Technology Stack Summary

Layer	Technology
Interface	Streamlit
AI Engine	Anthropic Claude
Embeddings	Sentence-Transformers (MiniLM)
Vector Store	Chroma
Language	Python
Document Parsing	PyPDF, LangChain utilities
Deployment	Streamlit Community Cloud

10. Project Value & Differentiation

Our approach distinguishes itself by combining:

- Healthcare-domain RAG architecture
- Policy version intelligence
- Audit-grade traceability
- Practical deployment via lightweight cloud tooling

The system demonstrates how LLMs can be responsibly operationalized in regulated domains, balancing innovation with compliance, transparency, and reliability.