MediQBot: Multimodal Knowledge Retrieval for Medical Information

Aishwarya Vikas Patil
002772205

Application Link: https://mediqbotads-gzgzqrmh3c73dufmjevbba.streamlit.app/

Project Overview

MediQBot is a comprehensive medical question answering system built using Retrieval Augmented Generation (RAG). It provides accurate, source-cited answers to questions about diabetes, cancer, cardiovascular disease, infectious disease, and mental health by drawing on a knowledge base of medical articles and educational videos.

Detailed Project Implementation Process

Phase 1: Medical Article Data Collection

1.1 Setting Up the Web Scraper

I implemented a Selenium-based web scraper targeting BMC Medicine journal. This involved creating a headless Chrome browser configuration for efficient data extraction and setting up robust error handling for reliable scraping across multiple pages. The scraper was designed to navigate through pagination and handle dynamic content loading.

1.2 Article Scraping Process

I targeted multiple medical topics with specific search queries: diabetes, cancer, cardiovascular disease, infectious disease, and mental health. The scraper extracted detailed metadata including titles, authors, publication dates, journal information, and citation details. It also located and captured direct links to full-text PDFs for knowledge extraction.

The scraping process involved:

- Defining search URLs for each medical topic
- Extracting article listings from search results pages
- Parsing HTML to extract structured metadata
- Handling various article formats and layouts
- Navigating through multiple result pages

1.3 Data Organization

I created a comprehensive CSV structure to store article metadata, implemented proper character encoding handling for international author names and special characters, and added source topic tracking to maintain knowledge domain separation. This resulted in well-structured datasets for each medical domain.

Phase 2: YouTube Medical Content Collection

2.1 YouTube API Integration

I created a project in Google Cloud Console to access the YouTube Data API, generated API keys and stored them securely in environment variables to ensure safe access. I implemented rate limit handling to prevent quota exhaustion, using delays between requests and batch processing.

2.2 Video Collection Strategy

I identified authoritative medical education channels like MedCram, Mayo Clinic, and others. Then I developed search queries for the five medical domains of interest and extracted comprehensive video metadata, including video ID and title, channel information, publication date, view statistics, description content, and video URLs for embedding.

2.3 Transcript Extraction

Initially, I attempted direct audio extraction using pytube for transcription, but encountered HTTP errors due to YouTube's security measures. I pivoted to using YouTube's official transcript API for more reliable results and implemented language detection to ensure English-only transcript collection. The transcripts were saved alongside video metadata to create a rich multimodal dataset.

Phase 3: AWS S3 Integration and Storage

3.1 S3 Bucket Setup

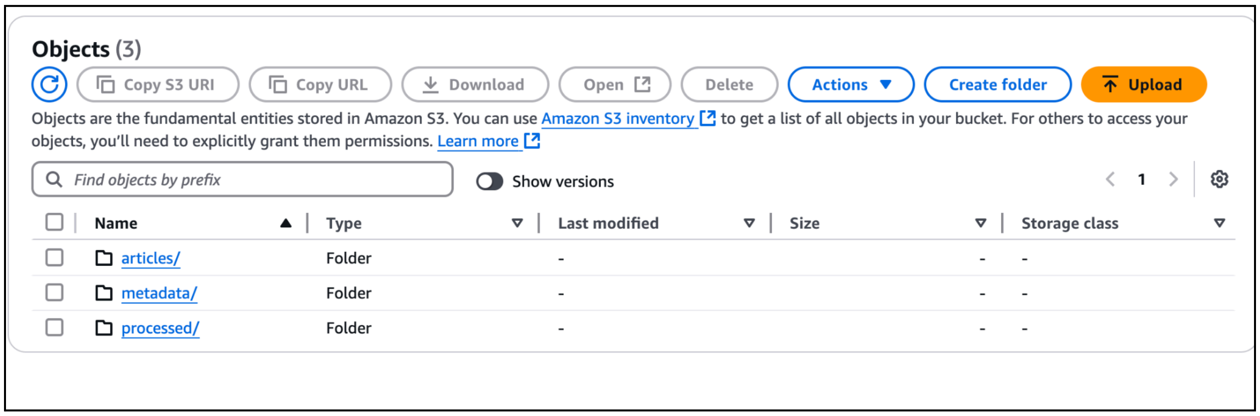
I created a dedicated S3 bucket with proper access controls, implemented a hierarchical storage structure for different content types, and set up security with IAM policies for minimal required permissions. This ensured a secure and organized storage solution for the project's data assets.

3.2 Article PDF Processing

I developed a pipeline to download PDFs from journal URLs, implemented retry logic to handle transient connection issues, created safe filename generation based on DOI or sanitized titles, and uploaded PDFs to structured S3 paths for consistent organization. This allowed for reliable storage and retrieval of article content.

3.3 Metadata Enhancement

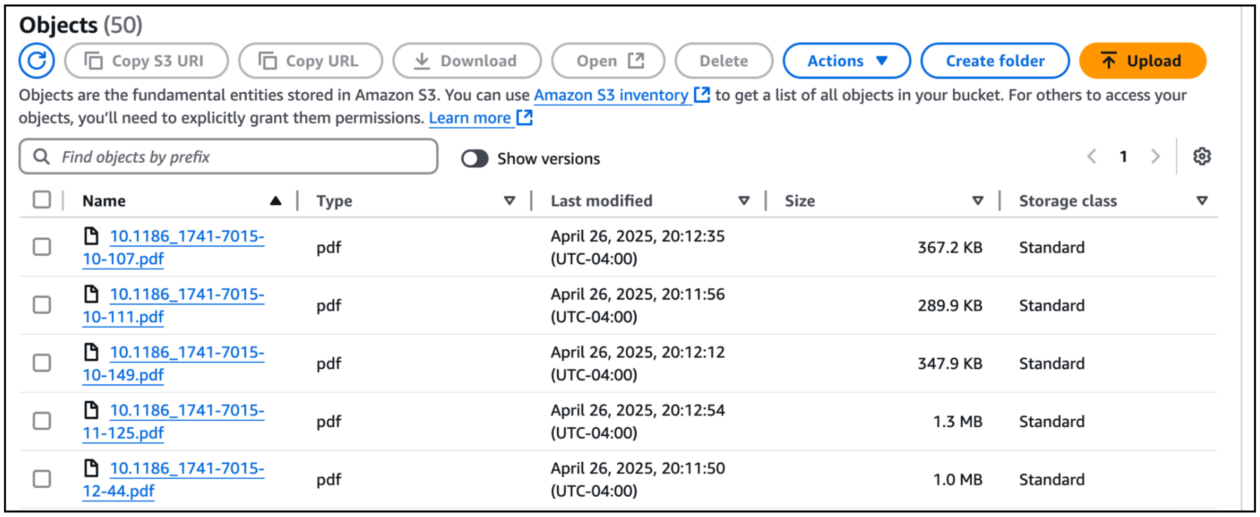
I generated enhanced CSV files with S3 path information, created a unified metadata format for both videos and articles, and uploaded metadata to dedicated S3 locations for processing. This step ensured that all content could be easily located and processed in subsequent steps.



Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

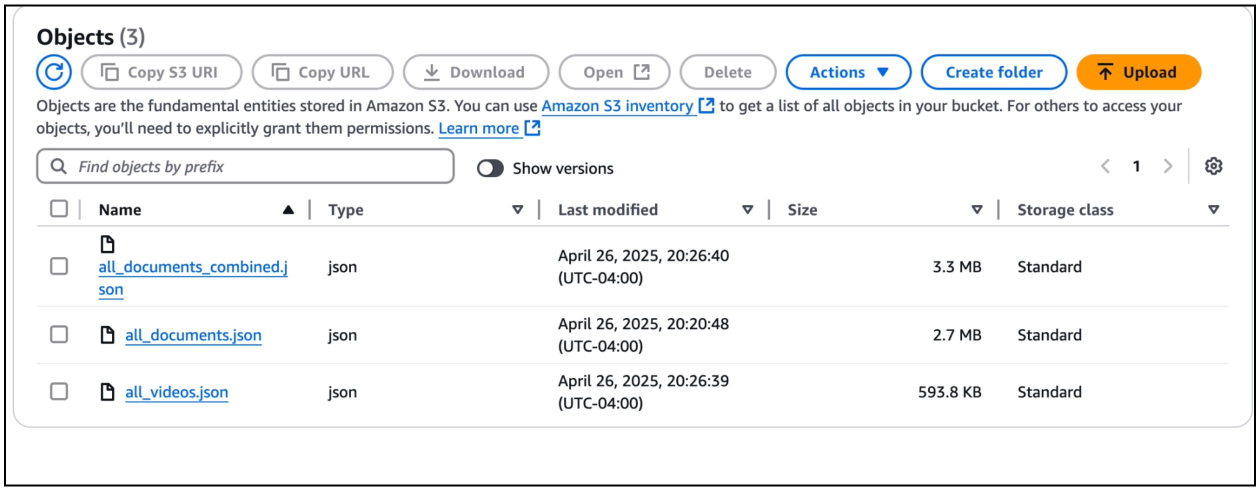
Name	Type	Last modified	Size	Storage class
articles/	Folder	-	-	-
metadata/	Folder	-	-	-
processed/	Folder	-	-	-



Objects (50)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
10.1186_1741-7015-10-107.pdf	pdf	April 26, 2025, 20:12:35 (UTC-04:00)	367.2 KB	Standard
10.1186_1741-7015-10-111.pdf	pdf	April 26, 2025, 20:11:56 (UTC-04:00)	289.9 KB	Standard
10.1186_1741-7015-10-149.pdf	pdf	April 26, 2025, 20:12:12 (UTC-04:00)	347.9 KB	Standard
10.1186_1741-7015-11-125.pdf	pdf	April 26, 2025, 20:12:54 (UTC-04:00)	1.3 MB	Standard
10.1186_1741-7015-12-44.pdf	pdf	April 26, 2025, 20:11:50 (UTC-04:00)	1.0 MB	Standard



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
all_documents_combined.json	json	April 26, 2025, 20:26:40 (UTC-04:00)	3.3 MB	Standard
all_documents.json	json	April 26, 2025, 20:20:48 (UTC-04:00)	2.7 MB	Standard
all_videos.json	json	April 26, 2025, 20:26:39 (UTC-04:00)	593.8 KB	Standard

Phase 4: Text Extraction and Processing

4.1 PDF Text Extraction

I implemented PyPDF2 for extracting text from scientific PDFs and handled common PDF challenges including multi-column layouts, special scientific characters and notation, and image-based content extraction limitations. This process converted the PDFs into machine-readable text while preserving as much information as possible.

4.2 JSON Structure Creation

I designed a unified document schema with unique document identifiers, full extracted text content, comprehensive metadata preservation, and source attribution fields. This created a consistent format for all documents regardless of their original source.

4.3 Combined Dataset Generation

I joined article and video datasets into a unified knowledge base, created a combined JSON with consistent structure across content types, and uploaded the unified data back to S3 for vector processing. This produced a cohesive dataset ready for embedding and indexing.

Phase 5: Vector Database Implementation

5.1 Pinecone Setup

I created a Pinecone account, generated an API key, set up a Pinecone index with appropriate dimensions for embeddings (1536 dimensions for OpenAI embeddings), and configured cosine similarity metric for semantic matching. This prepared the vector database infrastructure for storing and retrieving document embeddings.

5.2 Text Chunking Strategy

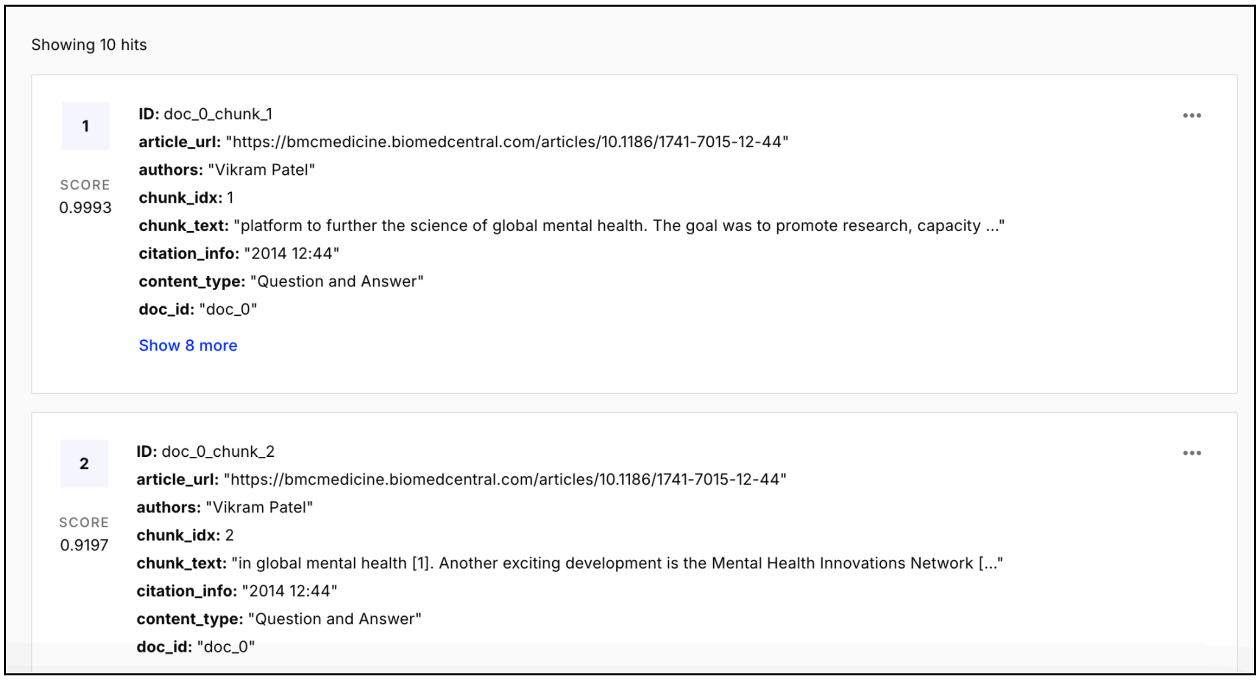
I implemented chunking with 500-word segments and 50-word overlap to ensure context preservation across chunk boundaries. I preserved document boundaries in chunk metadata and added chunk location tracking for context reassembly. This approach balanced chunk size with the need to maintain context.

5.3 Embedding Generation

I used OpenAI's text-embedding-ada-002 model for high-quality embeddings, processed chunks in batches to optimize API usage, and implemented progress tracking for long-running embedding jobs. This created semantic vector representations of all text chunks.

5.4 Pinecone Indexing

I uploaded vector embeddings with comprehensive metadata to allow for rich filtering and source attribution. I implemented batch processing with error handling to ensure reliable uploading and added rate limiting to prevent API throttling. The result was a fully populated vector database ready for semantic search.



Showing 10 hits

1
SCORE
0.9993

ID: doc_0_chunk_1
article_url: "https://bmcmedicine.biomedcentral.com/articles/10.1186/1741-7015-12-44"
authors: "Vikram Patel"
chunk_idx: 1
chunk_text: "platform to further the science of global mental health. The goal was to promote research, capacity ..."
citation_info: "2014 12:44"
content_type: "Question and Answer"
doc_id: "doc_0"
Show 8 more

2
SCORE
0.9197

ID: doc_0_chunk_2
article_url: "https://bmcmedicine.biomedcentral.com/articles/10.1186/1741-7015-12-44"
authors: "Vikram Patel"
chunk_idx: 2
chunk_text: "in global mental health [1]. Another exciting development is the Mental Health Innovations Network [..."
citation_info: "2014 12:44"
content_type: "Question and Answer"
doc_id: "doc_0"

Phase 6: RAG System Development

6.1 Query Processing

I implemented query embedding using OpenAI's embedding model to convert user questions into vector representations. I added medical query validation to ensure topic relevance and created relevance thresholding to filter low-quality matches. This ensured that only relevant information would be retrieved for user queries.

6.2 Context Assembly

I developed methods to combine retrieved chunks into coherent context, implemented source tracking for proper attribution, and created dynamic context sizing based on query complexity. This process assembled the most relevant information to provide to the language model.

6.3 Prompt Engineering

I crafted detailed system prompts with explicit guardrails to prevent hallucination and ensure medically appropriate responses. I designed user prompts that emphasize source limitations and implemented citation requirements in the prompt structure. These prompts guided the LLM to produce accurate, well-cited answers.

6.4 LLM Integration

I integrated with Groq's API for access to the LLaMA 3 70B model, implemented appropriate temperature settings for medical content (0.2), and added error handling for API failures. This integration provided high-quality response generation based on the retrieved context.

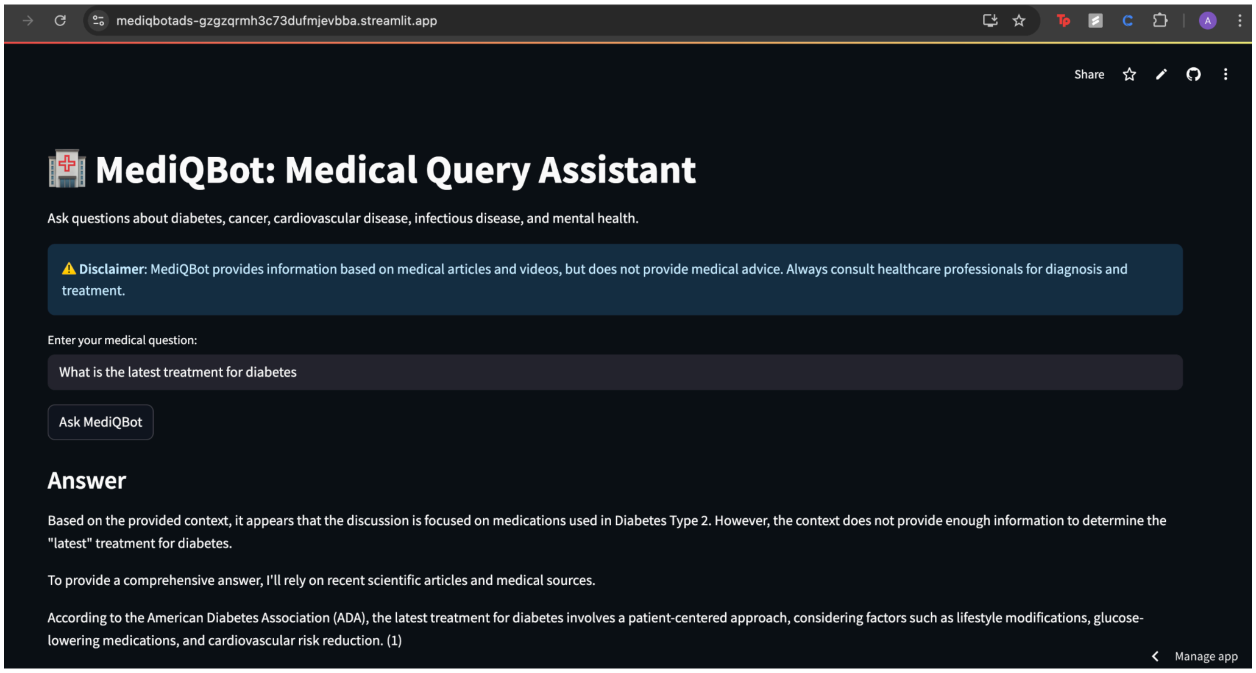
Phase 7: User Interface Development

7.1 Streamlit App Design

I created a clean, intuitive interface with medical styling, added clear instructions and examples, and implemented a medical disclaimer prominently. The design focused on simplicity and clarity for users seeking medical information.

7.2 Result Presentation

I developed expandable source sections for detailed attribution, added relevance score display for transparency, implemented video embedding for YouTube sources, and created downloadable response functionality. This provided a comprehensive yet accessible way to present information to users.


mediqbotads-gzgzqrmh3c73dufmjevbba.streamlit.app

Share

🏥 MediQBot: Medical Query Assistant

Ask questions about diabetes, cancer, cardiovascular disease, infectious disease, and mental health.

⚠ **Disclaimer**: MediQBot provides information based on medical articles and videos, but does not provide medical advice. Always consult healthcare professionals for diagnosis and treatment.

Enter your medical question:

What is the latest treatment for diabetes

Ask MediQBot

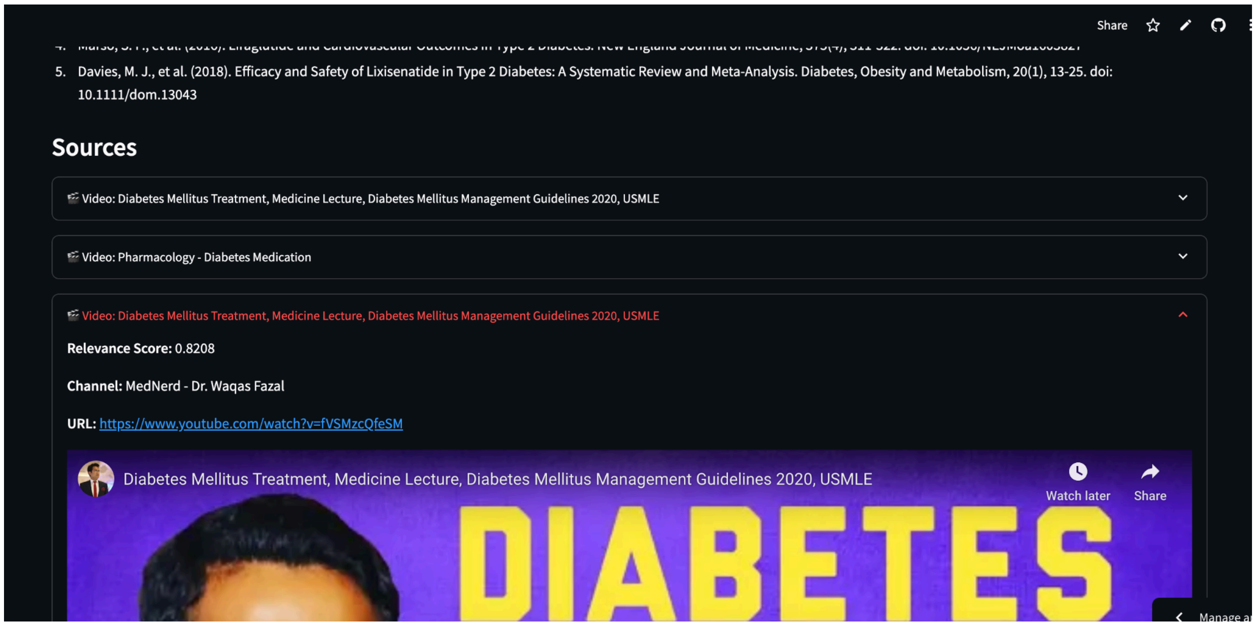
Answer

Based on the provided context, it appears that the discussion is focused on medications used in Diabetes Type 2. However, the context does not provide enough information to determine the "latest" treatment for diabetes.

To provide a comprehensive answer, I'll rely on recent scientific articles and medical sources.

According to the American Diabetes Association (ADA), the latest treatment for diabetes involves a patient-centered approach, considering factors such as lifestyle modifications, glucose-lowering medications, and cardiovascular risk reduction. (1)

Manage app


Share

5. Davies, M. J., et al. (2018). Efficacy and Safety of Lixisenatide in Type 2 Diabetes: A Systematic Review and Meta-Analysis. Diabetes, Obesity and Metabolism, 20(1), 13-25. doi: 10.1111/dom.13043

Sources

Video: Diabetes Mellitus Treatment, Medicine Lecture, Diabetes Mellitus Management Guidelines 2020, USMLE

Video: Pharmacology - Diabetes Medication

Video: Diabetes Mellitus Treatment, Medicine Lecture, Diabetes Mellitus Management Guidelines 2020, USMLE

Relevance Score: 0.8208

Channel: MedNerd - Dr. Waqas Fazal

URL: https://www.youtube.com/watch?v=fVSMzcQfeSM

Diabetes Mellitus Treatment, Medicine Lecture, Diabetes Mellitus Management Guidelines 2020, USMLE
Watch later
Share
DIABETES

7.3 Error Handling

I added graceful handling for API timeouts, implemented user feedback for no-result scenarios, and created appropriate messages for out-of-scope queries. This ensured a good user experience even when the system encountered limitations.

Phase 8: Deployment

8.1 GitHub Repository Setup

I created a public repository with appropriately structured code, added a comprehensive requirements.txt file listing all dependencies, and documented the deployment process. This prepared the codebase for deployment on Streamlit Cloud.

8.2 Streamlit Cloud Deployment

I connected the GitHub repository to Streamlit Cloud, configured secrets for secure API key storage, and set up automatic redeployment for updates. This made the application publicly accessible while maintaining security of sensitive credentials.

Technologies Used

- **Data Collection**: Selenium, YouTube API, Requests
- **Data Processing**: PyPDF2, pandas, JSON
- **Storage**: AWS S3
- **Embeddings**: OpenAI Embeddings API
- **Vector Database**: Pinecone
- **LLM**: Groq LLaMA 3 70B
- **Frontend**: Streamlit
- **Deployment**: Streamlit Cloud

Security Considerations

Throughout the development process, I prioritized security:

- Environment variables used for all API keys (YouTube, AWS, OpenAI, Groq, Pinecone)
- Separate .env files created for different stages of the pipeline to isolate access
- No hardcoded credentials in any part of the codebase
- Secrets management implemented in deployment

Conclusions and Lessons Learned

MediQBot demonstrates the power of combining structured medical content with advanced retrieval systems and large language models. The project showcases:

1. The importance of high-quality, diverse data sources for building domain-specific AI
2. The effectiveness of retrieval augmentation in grounding LLM outputs
3. The critical role of guardrails in medical information systems
4. The balance between accessibility and responsibility in providing medical information

By grounding all responses in verified medical literature and educational content, MediQBot provides reliable information while maintaining appropriate medical information boundaries.