Case Study: RAG Chatbot Powered by Google Gemini for Smart Document Q&A

Project Title: Intelligent Document Q&A Assistant using Retrieval-Augmented Generation (RAG) with Gemini

GitHub Repository: https://github.com/mukul-mschauhan/RAG-Using-Gemini
Live Demo: https://gemini-rag2025.streamlit.app/

Problem Statement

Across industries such as legal, finance, healthcare, and construction, professionals are required to extract insights from massive document repositories—contracts, product manuals, policies, reports, regulations, and emails.

Traditional keyword-based search and static FAQs fail to deliver contextual, accurate answers. Employees waste hours scanning PDFs and notes, leading to operational inefficiencies, poor decision-making, and knowledge silos.

There's a critical need for an intelligent assistant that can understand natural language questions, reason over domain-specific documents, and deliver precise responses—instantly.

Business Objective

To build an enterprise-grade, scalable, and cost-efficient RAG-powered AI assistant that enables:

- **Smart retrieval and summarization** of large-scale PDF/text content
- **Natural language understanding** of user queries
- **Domain-specific knowledge augmentation** via vector search
- **Real-time Q&A** over user-uploaded documents

This tool is aimed at: - Legal & Compliance teams automating contract review - Analysts querying internal policy documents - Researchers navigating technical papers - Support teams handling customer manuals and SOPs

---

Proposed Solution

We built a fully functional web application that allows users to:

1. Upload one or more PDF/text documents
2. Extract content and embed it using state-of-the-art vectorization (Google Gemini-compatible)
3. Store and retrieve relevant chunks using vector search (FAISS)
4. Ask questions in natural language
5. Get contextual answers generated by **Google Gemini 1.5 Flash** using the retrieved documents

---

Architecture Overview

1. **Frontend:** Streamlit web UI for uploading files and chat interface
2. **Document Processing:** Text extraction using PyMuPDF and chunking logic
3. **Embeddings:** Google-compatible embeddings (e.g., Gemini/Vertex-compatible or from SentenceTransformers)
4. **Vector Store:** FAISS for similarity search on embedded text chunks
5. **LLM Integration:** Google Gemini 1.5 Flash for natural language generation using retrieved chunks as context
6. **Prompt Engineering:** Carefully crafted system and user prompts to ensure contextual relevance and safety

---

Tech Stack

Layer	Tools Used
LLM	Google Gemini 1.5 Flash
Vector DB	FAISS
Embeddings	SentenceTransformers / Gemini
Text Extraction	PyMuPDF (fitz)
Frontend	Streamlit
Backend Integration	LangChain
Deployment	Streamlit Cloud
Programming Language	Python

Business Impact

- Reduced document navigation time by 90%
- Enabled 24x7 AI assistant for contract, policy, and research Q&A

- Democratized document access for non-technical users
- Scalable for internal or customer-facing use cases

Example Use Cases: - Ask a policy question like: *“What is the refund timeline for cancelled trips?”* - Upload 5 contracts and ask: *“Which clause discusses penalty on late delivery?”* - Upload medical SOPs and ask: *“When to escalate Stage 2 hypertension?”*

Conclusion & Future Roadmap

This RAG solution bridges the gap between static document stores and intelligent, real-time assistance.

Next Enhancements: - Multi-user support with user-based document history - Support for audio documents via speech-to-text - API endpoints for enterprise SaaS integration - In-built analytics dashboard

---

For the complete implementation, visit the GitHub repo: 👉 https://github.com/mukul-mschauhan/RAG-Using-Gemini

Try the live chatbot: 🚀 https://gemini-rag2025.streamlit.app/