---

title: "GenAI Doc Assistant" subtitle: "RAG + Multimodal Technical Documentation Assistant" author: "Yagna Patel" date: "2025-08-15" fontsize: 11pt geometry: margin=1in mainfont: "Helvetica" monofont: "Courier New" toc: true toc-depth: 3 colorlinks: true urlcolor: blue linkcolor: blue

Executive Summary

GenAI Doc Assistant is a production-ready, local-first system for retrieval-augmented generation (RAG) and multimodal summarization across PDFs, images, and videos. It pairs a Streamlit UI with a FastAPI backend, supports a local FAISS knowledge base, and optionally integrates Pinecone namespaces for scalable retrieval.

Key Capabilities:

- Ask natural-language questions over your docs with cited answers
- Summarize PDFs (text + tables), screenshots (OCR), and videos (audio transcription)
- Index content to FAISS locally or Pinecone remotely
- Clean prompts, guardrails, sticky namespaces, and a modern, responsive UI

Features

RAG (Retrieval-Augmented Generation)

- **Local FAISS KB:** POST `/ingest`, POST `/ask`
- **Optional Pinecone namespaces:** POST `/upsert_pinecone`, POST `/ask_pinecone`
- **Configurable top-K, simple reranking, source citations**

Multimodal Processing

- **PDF text extraction** (PyMuPDF) + tables (pdfplumber)
- **OCR for scanned PDFs & images** (Tesseract)
- **Image description** via `/describe_image`
- **Video → audio transcription** using OpenAI Whisper (optional)

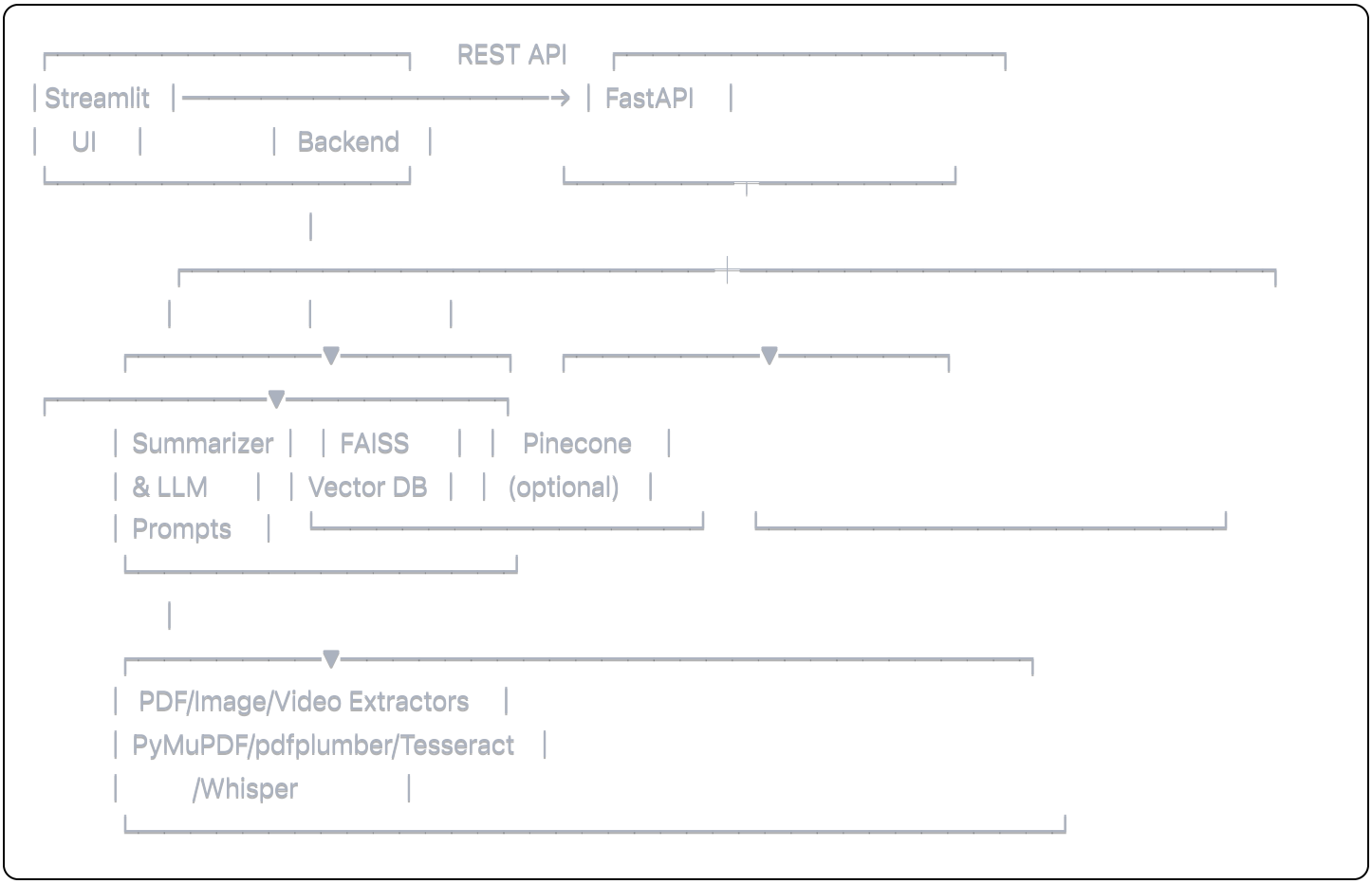
Prompt Engineering

- Structured, concise prompts with clear instructions and refusal cases
- Compact, source-aware context windows

Streamlit UI

- **Tabs:** Ask · Upload (PDF/Image/Text/Video) · Photo Summary
- **Features:** Sticky Pinecone namespace, dark theme, charts & table previews

System Architecture



Project Structure

```
genai-doc-assistant/
├── app/
│   ├── api.py              # FastAPI endpoints
│   ├── streamlit_app.py    # Streamlit UI
│   ├── rag_pipeline.py     # Retrieval + answer synthesis
│   ├── pinecone_embeds.py  # Pinecone upsert/query helpers
│   ├── chunkers.py         # Chunking strategies
│   ├── prompts.py          # Prompt templates
│   ├── llm_chat.py         # LLM wrappers/adapters
│   └── utils.py            # Settings, env, logging
├── data/
│   ├── raw/                # Source docs (md/pdf/txt/images)
│   └── kb/                 # FAISS index (auto generated)
├── docs/
│   ├── index.html          # Landing page (GitHub Pages)
│   └── assets/
│       └── architecture.png  # Architecture diagram
├── tests/
│   ├── test_api_smoke.py   # Basic endpoint sanity
│   ├── test_health.py
│   ├── test_retrieval.py
│   └── test_generation.py
├── .env.example
├── requirements.txt
├── README.md
└── LICENSE
```

Quickstart

1. Environment Setup

bash

```
# Clone repository
git clone <your-repo-url>
cd genai-doc-assistant

# Create virtual environment
python -m venv .venv
source .venv/bin/activate

# Install dependencies
pip install --upgrade pip
pip install -r requirements.txt

# Configure environment
cp .env.example .env
# Edit .env and add OPENAI_API_KEY for Whisper/image LLM features
```

macOS Additional Setup (for OCR/video)

```
bash

brew install tesseract ffmpeg

# If Tesseract isn't discovered automatically:
echo "TESSERACT_CMD=/opt/homebrew/bin/tesseract" >> .env
```

Pinecone Setup (Optional)

Set `PINECONE_API_KEY` and `PINECONE_INDEX` in `.env` file.

2. Run the Application

Terminal A - API Server

```
bash

source .venv/bin/activate
uvicorn app.api:app --reload --port 8000
```

Terminal B - Streamlit UI

```
bash
```

```
source .venv/bin/activate
export API_BASE="http://127.0.0.1:8000"
python -m streamlit run app/streamlit_app.py --server.port 8503
```

Sanity Checks

```bash
# Check API health
curl -s http://127.0.0.1:8000/health

# Test ingest endpoint
curl -s -X POST http://127.0.0.1:8000/ingest | jq
```

Using the Application

Ask Tab

1. Enter a question (e.g., "How do I rotate Snowflake keys?")
2. Choose FAISS (local) or toggle Use Pinecone
3. (Optional) Specify a namespace (sticky across tabs)
4. Click **Ask** — you'll get a cited answer

Upload Tab

1. Upload PDF/PNG/JPG/TXT/MD/MP4/MOV/M4V
2. For scanned PDFs, enable **Try OCR for scanned PDFs**
3. Click **Summarize this document** for a concise summary & highlights
4. Optionally **Index to Pinecone** (choose namespace) or **Save to local KB** (FAISS)

Photo Summary Tab

1. Upload a photo or paste a direct image URL (.png/.jpg/.jpeg/.gif/.webp)
2. Provide an instruction (optional)
3. Click **Summarize photo**

API Reference

Core Endpoints

Method	Endpoint	Description
GET	`/health`	Health check
GET	`/config`	Get configuration
POST	`/ingest`	Rebuild FAISS KB from data/raw
POST	`/ask`	Query FAISS knowledge base
POST	`/upsert_pinecone`	Add documents to Pinecone
POST	`/ask_pinecone`	Query Pinecone index
POST	`/summarize`	Summarize text
POST	`/describe_image`	Describe/analyze image

Example API Calls

Ask (FAISS)

```bash
curl -s -X POST http://127.0.0.1:8000/ask \
  -H 'Content-Type: application/json' \
  -d '{"question":"How do I rotate Snowflake keys?","k":5}' | jq
```

Ingest Local Documents

```bash
curl -s -X POST http://127.0.0.1:8000/ingest | jq
```

Upsert to Pinecone

```bash
curl -s -X POST http://127.0.0.1:8000/upsert_pinecone \
  -H 'Content-Type: application/json' \
  -d '{"texts":["hello"],"ids":["t1"],"namespace":"demo"}' | jq
```

Ask (Pinecone)

```bash
curl -s -X POST http://127.0.0.1:8000/ask_pinecone \
  -H 'Content-Type: application/json' \
  -d '{"question":"How do I rotate Snowflake keys?","top_k":8,"namespace":"demo"}' | jq
```

Describe an Image

```bash
curl -s -X POST http://127.0.0.1:8000/describe_image \
  -F 'prompt=Describe key numbers if visible.' \
  -F "file=@/path/to/image.png;type=image/png" | jq
```

Configuration

Environment Variables

Configure via `.env` file:

Variable	Required	Purpose
`OPENAI_API_KEY`	Optional*	Needed for Whisper and OpenAI LLM features
`PINECONE_API_KEY`	Optional	Enables Pinecone indexing/query
`PINECONE_INDEX`	Optional	Pinecone index name
`API_BASE`	Optional	Streamlit → API base (default: http://127.0.0.1:8000)
`TESSERACT_CMD`	Optional	Path to tesseract if not on PATH

*Core FAISS RAG works without OpenAI; video transcription & some vision features require it.

Example .env File

```
OPENAI_API_KEY=sk-...
PINECONE_API_KEY=
PINECONE_INDEX=
API_BASE=http://127.0.0.1:8000
TESSERACT_CMD=/opt/homebrew/bin/tesseract
```

Evaluation & Quality

- **RAGAS:** Tests in `tests/` and `eval_ragas.py` measure faithfulness and context relevance
- **Synthetic Data:** `synthetic_data.py` bootstraps Q–A pairs for regression/eval
- **Chunking:** Tune window/stride in `chunkers.py`; adjust top-K in UI and API

Mapping to Assignment Rubric

Technical Implementation (40%)

- RAG over FAISS/Pinecone
- Robust extractors
- Clear prompts
- Tested endpoints

Creativity & Application (20%)

- Technical doc assistant with multimodal capabilities
- Namespace-aware retrieval

Documentation & Presentation (20%)

- Comprehensive README (PDF-ready)
- Architecture diagram
- Landing page under `docs/index.html`

User Experience & Output Quality (20%)

- Clean UI (Streamlit)
- Cited answers
- Charts
- Stable ingestion & summarization

Troubleshooting

Port Already in Use

```bash
lsof -nP -iTCP:8503 | grep LISTEN
kill -9 <PID>
```

Streamlit Command Not Found

```bash
source .venv/bin/activate
python -m streamlit run app/streamlit_app.py --server.port 8503
```

422 Error on /summarize

Text must be ≥ 20 chars. For scanned PDFs/images, enable OCR and re-upload.

Multipart Form Error

```bash
pip install python-multipart
```

Tesseract/ffmpeg Missing (macOS)

```bash
brew install tesseract ffmpeg
export TESSERACT_CMD=/opt/homebrew/bin/tesseract
```

Image URL Fails

Must be a direct image URL ending with .png/.jpg/.jpeg/.gif/.webp

Security & Ethics

- Process only content you have rights to use
- Avoid sending sensitive data to external services
- Provide safety filters and refusal cases when deploying broadly
- Document limitations and potential misuse

Roadmap

- [ ] Hybrid retrieval (BM25 + embeddings)
- [ ] Better reranking & answer calibration
- [ ] Doctype-specific chunkers (slides/HTML/tables)
- [ ] Batch ingestion jobs & progress UI
- [ ] Conversation exports with citation maps

Acknowledgements

Built with ❤️ using:

- **Streamlit** - Interactive web UI

- **FastAPI** - High-performance API backend
- **PyMuPDF** - PDF text extraction
- **pdfplumber** - Table extraction from PDFs
- **Tesseract** - OCR capabilities
- **FAISS/Pinecone** - Vector search
- **moviepy/ffmpeg** - Video processing
- **OpenAI Whisper** - Audio transcription (optional)

License

MIT — see LICENSE file for details.

Appendix A: Local PDF Export

To export this README to a professional PDF:

macOS

```bash
# Install LaTeX
brew install --cask mactex  # Full LaTeX distribution
# OR
brew install basictex       # Minimal LaTeX distribution

# Convert to PDF
pandoc README.md -o README.pdf --pdf-engine=xelatex
```

Ubuntu

```bash
# Install LaTeX
sudo apt-get install texlive-xetex

# Convert to PDF
pandoc README.md -o README.pdf --pdf-engine=xelatex
```
