mainfont: ”Noto Sans” monofont: ”JetBrains Mono” geometry: margin=1in header-includes:

- | \usepackage{newunicodechar} \usepackage{amssymb} % for \heartsuit % Map missing code points to LaTeX equivalents or strip VS16 \newunicodechar{→}{→} \newunicodechar{ }{≥} \newunicodechar{ }{\ensuremath{\heartsuit}} \newunicodechar{}{} % strip VARIATION SELECTOR-16 (U+FE0F)

---

title: ”GenAI Doc Assistant” subtitle: ”RAG + Multimodal Technical Documentation Assistant” author: ”Yagna Patel” date: ”2025-08-15” fontsize: 11pt geometry: margin=1in toc: true toc-depth: 3 colorlinks: true

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.

- **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**
  - Local FAISS KB: `POST /ingest`, `POST /ask`
  - Optional **Pinecone** namespaces: `POST /upsert_pinecone`, `POST /ask_pinecone`
  - Configurable top-K, simple reranking, source citations
- **Multimodal**
  - **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**
  - Sticky Pinecone namespace, dark theme, charts & table previews

---

System Architecture

```
flowchart LR
  A[Streamlit UI] -- REST --> B[FastAPI app]
  B --> C[Summarizer & LLM Prompts]
  B --> D[FAISS Vector DB]
  B --> E[Pinecone (optional)]
  B --> F[PDF/Image/Video Extractors\nPyMuPDF/pdfplumber/Tesseract/Whisper]
  D & E --> G[Retriever/Ranker]
  G --> C
```

Image fallback (optional): place a diagram at `docs/assets/architecture.png` and reference it with:

```
![Architecture](docs/assets/architecture.png)
```

Project Structure

```
genai-doc-assistant/
  app/
    api.py                # FastAPI (health, ask, summarize, image describe, ingest, pineco
    streamlit_app.py      # Streamlit UI (Ask / Upload / Photo Summary)
    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  # (optional) architecture image
  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

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

python -m venv .venv
source .venv/bin/activate

pip install --upgrade pip
pip install -r requirements.txt

cp .env.example .env
# Fill OPENAI_API_KEY for Whisper/image LLM if you want those features
```

macOS extras (for OCR/video):

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

Pinecone (optional): set `PINECONE_API_KEY` and `PINECONE_INDEX` in `.env`.

2) Run

Terminal A – API

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

Terminal B – UI

```
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

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

Using the App

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

```
GET  /health
GET  /config
POST /ingest                          # rebuild FAISS KB from data/raw
POST /ask            {"question": "...", "k": 5}
POST /upsert_pinecone {"texts":[...], "ids":[...], "namespace":"demo"}
POST /ask_pinecone   {"question":"...", "top_k":8, "namespace":"demo"}
POST /summarize      {"text":"...", "max_words":250}
POST /describe_image (multipart: file + prompt)
```

Examples

- Ask (FAISS)

```
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 docs

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

- Upsert to Pinecone

```
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)

```
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

```
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

Set via `.env`:

Variable	Required	Purpose
`OPENAI_API_KEY`	Optional*	Needed for Whisper and any OpenAI LLM features you enable.
`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 (e.g., `/opt/homebrew/bin/tesseract`)

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

Example:

```
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/` and `eval_ragas.py` help 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 and namespace-aware retrieval.
- **Documentation & Presentation (20%)**
  - This 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**

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

- **`streamlit: command not found`**

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

- **422 on `/summarize`**

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

- **Multipart form error**

  ```
  pip install python-multipart
  ```

- **Tesseract / ffmpeg missing (macOS)**

  ```
  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** · **FastAPI** · **PyMuPDF** · **pdfplumber** · **Tesseract** · **FAISS/Pinecone** · **moviepy/ffmpeg** · *(optional)* **OpenAI Whisper**.

License

Appendix A — Local PDF Export

To export this README to a professional PDF:

```
# macOS: brew install --cask mactex  (for xelatex)  OR  brew install basictex
# Ubuntu: sudo apt-get install texlive-xetex

pandoc README.md -o README.pdf --pdf-engine=xelatex
```