⭐ Milestone Documentation: RAG + LLM Pipeline for Visa Eligibility Screening

Project: SwiftVisa – AI-Based Visa Eligibility Screening Agent

---

📌 Milestone Objective

Build a **Retrieval-Augmented Generation (RAG)** pipeline that:

1. **Collects** visa/immigration documents
2. **Extracts, preprocesses, and chunks text**
3. **Generates embeddings**
4. **Stores embeddings in a vector store (FAISS/Chroma)**
5. **Retrieves relevant policy information for queries**
6. **Uses an LLM (OpenAI/Gemini/Mistral)** to generate grounded visa eligibility explanations
7. **Produces confidence scores and citations**

This milestone validates your understanding of **RAG**, **embeddings**, **vector databases**, and **LLM inference**.

1. Supported Visa Categories & Fields

The system supports multiple countries and visa types.
Example:

Countries Included (Week 1–2 Project):

- USA
- Canada
- Germany
- Australia
- UK

Visa Types (per country)

(Example — based on available PDFs)

USA

- F1 Student Visa
- H1B Work Visa
- B1/B2 Tourist Visa
- L1 Intra-Company Visa

Canada

- Express Entry
- Study Permit
- Work Permit
- Provincial Nominee Program

Germany

- Job Seeker Visa
- Student Visa
- Work Visa (Blue Card)

Australia

- Skilled Independent Visa (189)
- Student Visa (500)

UK

- Skilled Worker Visa
- Student Visa

📂 **2. Document Collection**

Manual Collection Requirements

You must gather **official immigration PDFs**, such as:

- Government-issued visa policy guides
- Eligibility criteria documents
- Application checklists
- Work permit instructions
- Country-specific PR guidelines

Save them into the folder:

/data/<country_name>/<visa_type>/

Example:

/data/canada/express_entry/express_entry_guide.pdf

/data/us/h1b/h1b_policy_handbook.pdf

/data/germany/job_seeker/germany_job_seeker.pdf

This maintains a professional structure and allows easy extension.

🖌️ 3. Extract, Clean, and Chunk Documents

Step 1 — Extract Text

Use:

- pdfplumber
- PyPDF2
- Fallback OCR (if PDF is scanned)

Step 2 — Clean Text

Remove:

- Headers, footers
- Extra whitespaces
- Non-ASCII characters
- Table formatting artifacts
- Empty lines

Step 3 — Chunking

Use sliding-window chunking:

- **Chunk size:** 400–800 tokens
- **Overlap:** 100–150 tokens

Reason:
LLMs do not use entire PDFs at once; they require small sections so the RAG model can retrieve relevant policy blocks.

4. Generate Embeddings

Available Models:

Option A — SentenceTransformer

model = SentenceTransformer("all-MiniLM-L6-v2")

Pros:

- Free

- Fast
- Good for medium accuracy

Option B — OpenAI Embeddings

text-embedding-3-large

text-embedding-3-small

Pros:

- Highest quality
- Works with multilingual data
- Best retrieval accuracy

Option C — Gemini Embeddings

models/text-embedding-004

Pros:

- Fast
- Google-optimized for document policy search

🗄️ 5. Store Embeddings in FAISS or Chroma

Recommended: FAISS

Fastest for similarity search.

Process:

1. Convert chunks → embeddings
2. Store them into FAISS Index
3. Save as:

⛓️ 6. RAG Pipeline Implementation

Below is the core logic:

🔥 Step 1 — Retrieve Relevant Chunks

Input:
User query → “What are the requirements for Canada PR?”

Process:

- Convert query → embedding

- Perform FAISS similarity search
- Return top K chunks (K = 5 or 7)

🔥 **Step 2 — Construct a Prompt**

You are SwiftVisa, an expert immigration assistant.

Answer ONLY using the policy context provided.

[User Question]

What are the requirements for Canada PR?

Provide:

1. Final answer

2. Eligibility check

3. Bullet-point supporting rules

4. Confidence score

5. Source citations

Deliverables for This Milestone

---

1. Working RAG + LLM Pipeline

That can answer:

- Visa requirements
- PR eligibility
- Document checklists
- Work visa rules
- Study visa requirements

2. Eligibility Outputs

Generated using LLM + retrieved document chunks.

3. Explanation + Citations

Every answer grounded in:

- Country PDFs
- Policy rules
- Visa guidelines

4. Confidence Assessments

LLM + FAISS-based scoring.

5. Logged Decision History

For auditability and demo purposes.

🎓 **Key Learnings (Week 1 + Week 2)**

----

1. How RAG Works

RAG =
User Query → Vector Search → Relevant Chunk Retrieval → LLM Answer grounded in context

It removes hallucinations by forcing the model to answer ONLY using real policy documents.

---

2. What Are Embeddings?

- Numerical vector representation of text
- Captures meaning & context
- Used for similarity search
- Essential for FAISS vector search
- LLMs understand through embeddings, not text

---

3. Different Embedding Models

Model	Strength	Use Case
SentenceTransformer	Free, fast	Good baseline
OpenAI Embeddings	Best accuracy	Production RAG
Gemini Text Embedding	Fast, scalable	Google-rich content
LLaMA Local Embeddings	Offline	Data privacy

---

4. Why FAISS or Chroma?

Vector DB	Pros
FAISS	Fastest, GPU support, ideal for large PDF collections
ChromaDB	Easy to use, persistent, great for local RAG apps

FAISS = Best for high-accuracy visa screening.

🎉 Final Note

This documentation covers the **complete lifecycle** of your milestone:

✓ PDF → Extraction → Cleaning → Chunking
✓ Embeddings → FAISS storage
✓ Retrieval → Prompt building
✓ LLM answer → Explanation → Confidence
✓ Logging + citations

This is exactly what mentors and companies expect.