📘 SwiftVisa – AI-Based Visa Eligibility Screening Agent

Milestone 1: Creating a Vector Database

📌 Objective of Milestone

The goal of this milestone is to build the foundational **vector database** using documents related to different countries. This vector DB will empower the SwiftVisa Agent to efficiently search, retrieve, and analyze visa-related information during user interactions.

This milestone includes:

1. Collecting country-wise documents
2. Preprocessing and cleaning the text
3. Generating embeddings using SentenceTransformers
4. Storing the embeddings inside FAISS for fast retrieval

📁 2. Create a Folder With List of Countries the Agent Caters To

Step Details

The SwiftVisa agent needs country-specific immigration and visa documents to answer user queries.
Create a folder named **Data/** containing all relevant PDFs.

Example Documents

Country	Examples of Documents
Canada	Express Entry Guide, Post-Study Visa Rules, Work Permit Guide
USA	F1 Visa Guide, H1B Rules, B1/B2 Information
Australia	Skilled Visa Handbook, Student Visa Information

Purpose of This Step

- Organize raw inputs for the pipeline
- Enable the agent to map each country to its visa rules
- Prepare the source for embeddings and retrieval

🧹 3. Preprocess and Clean the Documents

Why Preprocessing Is Needed?

Visa PDFs are generally unstructured and contain:

- Headers & footers

- Irregular spacing
- Line breaks
- Hyphens
- Page numbers

This noise will reduce embedding quality.

Steps Included

1. Extract text from PDF using **pdfplumber** (fallback: PyPDF2)
2. Normalize text
3. Remove unwanted characters
4. Save cleaned output to /raw/cleaned_text.txt

Outcome

✓ Clean, consistent text
✓ Higher-quality chunks
✓ Better embeddings

✂️ 4. Chunking the Text

Why Chunking?

- Large documents can't be embedded directly
- Splitting text into 300–500 word chunks improves retrieval quality
- Helps the AI match user queries to specific visa-related sections

Chunk Format

```
{
 "chunk_id": 1,
 "text": "Eligibility criteria for Canada Express Entry..."
}
```

Outputs Generated

- raw/chunks.json
- Printed visual logs showing:
    - Number of chunks
    - Size of each chunk

🧠 5. Generating Embeddings Using Sentence Transformers

What Are Embeddings?

Embeddings are **numerical vector representations** of text.
They capture meaning and semantic relationships, enabling the AI to:

- Understand user queries
- Match questions to relevant visa rules
- Perform similarity search

Why Sentence Transformers?

Because ST models:

- Are lightweight and fast
- Provide state-of-the-art semantic understanding
- Work well with multilingual and visa-related content

Embedding Model Used

all-MiniLM-L6-v2

Output

- raw/embeddings.npy
- Logged outputs:
  - Embedding shape (number_of_chunks × embedding_dimension)
  - Dimension = 384

🗂 6. Store Embeddings Inside FAISS

What is FAISS?

FAISS (Facebook AI Similarity Search) is:

- A high-performance vector database
- Built for fast similarity search
- Optimized for large datasets

Why Use FAISS Instead of ChromaDB?

FAISS	Chroma DB
Fast, GPU-accelerated search	Easy to use, beginner-friendly
Industry standard for RAG systems	Good for prototyping

FAISS	**Chroma DB**
Highly scalable to millions+ vectors	Slower for large data
Used in production for semantic search, RAG, QA	Often used in local/dev environments

FAISS Index Built

- Type: IndexFlatL2
- Stores: All embeddings
- Output file: raw/faiss_index.bin

Reality of This Step

The FAISS index becomes the **backend memory** for the SwiftVisa agent.

📘 7. Learnings From Week 1–2

🔍 1. How RAG (Retrieval Augmented Generation) Works

RAG = Retrieval + Generation

Pipeline

1. User asks:
   "Am I eligible for Canada PR?"
2. System embeds the query
3. FAISS retrieves top-k relevant chunks
4. LLM (GPT) reads retrieved chunks
5. LLM generates a trusted, grounded answer

Why RAG Is Important in SwiftVisa

- Ensures answers follow official guidelines
- Avoids hallucinations
- Provides document-grounded visa recommendations

🧠 2. What Are Embeddings

Embeddings are:

- Numerical vector representations of text
- Enable semantic search
- Capture meaning beyond keywords

Example:

- “Work Permit Canada”
- “Canadian employment visa”

These should be close in vector space → embeddings make this possible.

🧰 3. Different Embedding Generation Modules

Library	**Examples**	**Notes**
SentenceTransformers (SBERT)	all-MiniLM-L6-v2, mpnet-base	Best for semantic RAG
OpenAI Embeddings	text-embedding-3-small	Best accuracy but paid
Google USE	Universal Sentence Encoder	Older model but still useful
HuggingFace Models	msmarco, e5-base	Domain-tuned models available

---

📚 4. Importance of FAISS or ChromaDB

Why We Need a Vector Database

LLMs cannot store or remember documents.
We need a vector database to:

- Embed documents
- Search similar text
- Retrieve relevant content fast
- Feed documents to LLM for RAG

FAISS Benefits

- Super fast retrieval
- Supports millions of vectors
- Used by Meta, Amazon, NVIDIA
- Works offline (your project requirement)

ChromaDB Benefits

- Simple Python API
- Persistent and user-friendly

- Good for small projects

Why FAISS for SwiftVisa

- Speed is critical
- Country documents may grow large
- Supports scaling
- Production-ready backend for the agent

🎯 FINAL SUMMARY

Milestone 1 built the essential foundation for RAG-based visa screening:

Stage	Output
Document Collection	/Data folder
Preprocessing	/raw/cleaned_text.txt
Chunking	/raw/chunks.json
Embeddings	/raw/embeddings.npy
Vector Store	/raw/faiss_index.bin

This milestone ensures that the SwiftVisa Agent can:

- Understand user queries semantically
- Retrieve the correct visa rules
- Provide grounded and accurate responses