SwiftVisaAI - Based Visa Eligibility Screening Agent

Yamini Sai Chodavarapu

Infosys Internship — Milestone 3

Contents

Chapter 1

Project Statement

1.1 Problem Definition

Visa application procedures involve multiple eligibility criteria, including financial requirements, documentation checks, and country-specific rules. Applicants often find it difficult to interpret these requirements accurately before applying, leading to application delays or rejections. Manual verification of eligibility is also time-consuming and error-prone.

SwiftVisaAI addresses this challenge by introducing an AI-driven visa eligibility screening agent that analyzes user-provided information and evaluates it against predefined visa rules. By leveraging retrieval-augmented generation (RAG), the system provides reliable, context-aware responses to user queries, helping applicants make informed decisions before initiating the official visa application process.

1.2 Objectives

1. To design an AI-based visa eligibility screening agent.
2. To collect and preprocess visa-related eligibility data.
3. To generate vector embeddings for efficient information retrieval.
4. To implement a Top-k retrieval mechanism for relevant document selection.
5. To integrate a large language model using retrieval-augmented generation (RAG).
6. To document the complete technical pipeline in a formal report format.

Chapter 2

Pipeline Overview

2.1 Overview of the System Architecture

The SwiftVisaAI system follows a structured and modular pipeline designed to efficiently process user queries and provide accurate visa eligibility responses. The pipeline combines data preprocessing, vector-based retrieval, and large language model reasoning to ensure reliable and context-aware outputs.

The overall workflow begins with the preparation of visa-related data, which includes eligibility rules, financial thresholds, and documentation requirements. This data is processed and converted into vector embeddings to enable efficient similarity-based retrieval. When a user submits a query, the system retrieves the most relevant information from the database and augments it with a large language model to generate a final response.

2.2 Pipeline Flow

The SwiftVisaAI pipeline consists of the following key stages:

1. **User Input:** The user provides personal, financial, or visa-related information through a query interface.

2. **Dataset Preparation:** Visa eligibility rules and guidelines are collected and preprocessed to remove noise and ensure consistency.

3. **Embedding Generation:** The preprocessed textual data is converted into numerical vector representations using a pre-trained embedding model.

4. **Retrieval (Top-k):** The system performs similarity search to retrieve the top-k most relevant documents based on the user query.

5. **LLM Augmented with Retrieval (RAG):** The retrieved documents are combined with the user query and passed to a large language model to generate a contextual and accurate eligibility response.

6. **Final Output:** The system presents the eligibility result and supporting explanation to the user.

2.3 Advantages of the Proposed Pipeline

The proposed pipeline offers several advantages:

- Improved accuracy through retrieval-augmented generation.
- Efficient handling of large volumes of visa-related textual data.
- Modular design that allows easy updates to data or models.
- Reduced manual effort in preliminary visa eligibility screening.
- This pipeline ensures scalability and reliability while maintaining interpretability of eligibility decisions.

Chapter 3

Implementation Steps

3.1 Dataset Preparation

The dataset preparation stage forms the foundation of the SwiftVisaAI system. This step involves collecting, organizing, and preprocessing visa-related information to ensure high-quality input for the downstream components of the pipeline.

3.1.1 Data Sources

The dataset is constructed using visa eligibility guidelines, financial requirements, and documentation rules collected from reliable and publicly available sources. These sources provide structured and semi-structured textual information required for eligibility assessment.

3.1.2 Data Components

The dataset is organized into the following data components:

- Personal information requirements
- Financial eligibility criteria
- Documentation and supporting evidence
- Country-specific visa rules

3.1.3 Preprocessing Steps

To improve data quality and retrieval efficiency, the following preprocessing steps are applied:

- Removal of irrelevant symbols and formatting noise
- Text normalization such as lowercasing and whitespace handling
- Segmentation of large documents into smaller text chunks

3.2 Embeddings

Embeddings play a crucial role in enabling semantic search within the SwiftVisaAI system. They transform textual information into numerical vector representations that capture the semantic meaning of visa-related documents and user queries.

3.2.1 Embedding Model Selection

A pre-trained embedding model is used to generate dense vector representations for both the dataset documents and user queries. The model is selected based on its ability to capture semantic similarity and its efficiency in handling large volumes of text data.

3.2.2 Embedding Generation Process

During this stage, the preprocessed text chunks are passed through the embedding model to obtain fixed-length vector representations. These vectors are stored in a vector database, allowing efficient similarity-based retrieval during query processing.

The same embedding process is applied to user queries to ensure consistency between document vectors and query vectors.

3.3 Retrieval (Top-k)

The retrieval component is responsible for identifying the most relevant visa-related documents required to answer a user query. This step ensures that only the most contextually appropriate information is provided to the language model.

3.3.1 Similarity Search Mechanism

The system employs a vector-based similarity search mechanism to compare the embedded user query with the stored document embeddings. A similarity metric such as cosine similarity is used to measure the closeness between vectors and identify relevant documents.

3.3.2 Top-k Document Selection

Based on the similarity scores, the system selects the top-k documents that are most relevant to the user query. These selected documents provide focused and reliable context for the subsequent response generation stage.

Limiting the retrieval to top-k documents improves both efficiency and response accuracy by reducing noise and irrelevant information.

3.4 LLM Augmented with Retrieval (RAG)

The RAG (Retrieval-Augmented Generation) component combines the retrieved documents with the user query to generate a context-aware response. This ensures that the final output is accurate, relevant, and grounded in the retrieved data.

3.4.1 Integration with Retrieved Documents

The top-k documents retrieved in the previous step are passed to the large language model (LLM) along with the user query. This allows the model to generate responses based on relevant context rather than relying solely on pre-trained knowledge.

3.4.2 Model Selection

A suitable large language model is chosen based on its ability to:

- Understand natural language queries
- Generate coherent and context-aware responses
- Handle structured and unstructured text effectively

3.4.3 Confidence Evaluation

The system evaluates the relevance and reliability of the retrieved documents, ensuring that the LLM-generated response is supported by high-confidence information from the database.

3.4.4 Final Output Generation

Once the LLM processes the query with the retrieved context, the system outputs a concise and informative eligibility result to the user, along with supporting explanations.

Appendix: Implementation Code

```
!pip install google-genai python-dotenv
```

```
Requirement already satisfied: google-genai in /usr/local/lib/python3.12/dist-packages (1.55.0)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.12/dist-packages (1.2.1)
Requirement already satisfied: anyio<5.0.0,>=4.8.0 in /usr/local/lib/python3.12/dist-packages (from google-genai) (4.12.0)
Requirement already satisfied: google-auth<3.0.0,>=2.14.1 in /usr/local/lib/python3.12/dist-packages (from google-auth[requests]
Requirement already satisfied: httpx<1.0.0,>=0.28.1 in /usr/local/lib/python3.12/dist-packages (from google-genai) (0.28.1)
Requirement already satisfied: pydantic<3.0.0,>=2.9.0 in /usr/local/lib/python3.12/dist-packages (from google-genai) (2.12.3)
Requirement already satisfied: requests<3.0.0,>=2.28.1 in /usr/local/lib/python3.12/dist-packages (from google-genai) (2.32.4)
Requirement already satisfied: tenacity<9.2.0,>=8.2.3 in /usr/local/lib/python3.12/dist-packages (from google-genai) (9.1.2)
Requirement already satisfied: websockets<15.1.0,>=13.0.0 in /usr/local/lib/python3.12/dist-packages (from google-genai) (15.0.1
Requirement already satisfied: typing-extensions<5.0.0,>=4.11.0 in /usr/local/lib/python3.12/dist-packages (from google-genai) (
Requirement already satisfied: distro<2,>=1.7.0 in /usr/local/lib/python3.12/dist-packages (from google-genai) (1.9.0)
Requirement already satisfied: sniffio in /usr/local/lib/python3.12/dist-packages (from google-genai) (1.3.1)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.12/dist-packages (from anyio<5.0.0,>=4.8.0->google-genai) (3.
Requirement already satisfied: cachetools<7.0,>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>=2.14.
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>=2.14.1
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.12/dist-packages (from google-auth<3.0.0,>=2.14.1->google
Requirement already satisfied: certifi in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0,>=0.28.1->google-genai) (202
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.12/dist-packages (from httpx<1.0.0,>=0.28.1->google-genai
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.12/dist-packages (from httpcore==1.*->httpx<1.0.0,>=0.28.1->g
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.9.0->g
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.9.0->go
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic<3.0.0,>=2.9.0-
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.28.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.28.1->goog
Requirement already satisfied: pyasn1<0.7.0,>=0.6.1 in /usr/local/lib/python3.12/dist-packages (from pyasn1-modules>=0.2.1->goog
```

```
import os

# Option 1: Set Gemini API key directly
os.environ["GEMINI_API_KEY"] = "AIzaSyCYSeshyN9vsjJq6a2C8fUdUX6qeffFWL4"

# Option 2: If you have .env file
# from dotenv
```

```
import google.generativeai as genai

API_KEY = os.environ["GEMINI_API_KEY"]
genai.configure(api_key=API_KEY)
```

```
from google.colab import files

uploaded = files.upload()

combined_text = ""

for filename in uploaded.keys():
    with open(filename, "r", encoding="utf-8") as f:
        combined_text += f"\n\n--- Content from {filename} ---\n"
        combined_text += f.read()

print("First file loaded")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving uaevisaeligibity.txt to uaevisaeligibity (5).txt
First file loaded
```

```
uploaded = files.upload()

for filename in uploaded.keys():
    with open(filename, "r", encoding="utf-8") as f:
        combined_text += f"\n\n--- Content from {filename} ---\n"
        combined_text += f.read()

print("Second file added")
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving visaeligibility.txt to visaeligibility.txt
Second file added
```

```
print(combined_text[:1000])
```

```
--- Content from uaevisaeligibity (5).txt ---
.

🇦🇪 1. Dubai / UAE Visa Types & Eligibility
✅ Tourist Visa (Visit / eVisa)

Dubai offers online e-Visas for short stays:

Types
✓ Transit visa (96 hours / 4 days)
✓ 30-day tourist visa
✓ 90-day tourist visa

Eligibility

Valid passport (≥ 6 months validity)

Completed online application

Pay visa fee

Some applications require confirmed flight/hotel bookings
vishikhamedia.in

Typical Fees & Duration

30-day visa approval ~ ₹6,710 (approx)

Express visa is slightly higher
vishikhamedia.in

🧳 2. Visit Visa Eligibility for Family & Friends

If you live/work in UAE and want to sponsor visitors:

Minimum Salary Requirements (2025)
✓ Immediate family (parents, spouse, kids): ≈ AED 4,000/month
✓ Second/third degree relatives (siblings, grandparents, cousins): ≈ AED 8,000/month
✓ Friends: ≈ AED 15,000/month
(Proof of salary needed: salary certificate / bank statements)
Dubai UAE
+1

Documents Required

Sponsor’s valid UAE resident visa & Emi
```

```
from google.colab import files
files.upload()   # upload background.jpg
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving background to background
{'background': b'RIFFh\xf2\x04\x00WEBPVP8
\\\xf2\x04\x00P\xfa\x11\x9d\x01*\xb8\x0b\x8d\t>=\x1e\x8cD\xa2!\xa3\x11*U\\0\x03\xc4\xa6\xef\xf9\xced\xf8\x1b\xcc\xf1\xc8\xf9\x
\xc8\xfc\xff\\\xd1\xe7\r\xf8\x17\xa0\x07\xf2_W_S\xfe\xde\xbf\xd2\xf4\xa7\xfa!\xbc\x01\xe93\xcb\xfb7\xf6\x8f\xf1\xff\xe3\xbf\xb
\xdf\x7f\xc7\x7ft\xfe\xc7\xff\x83\xfc\xff\xce\x8d\x8d\xfaW\xf7o\xf3\x1f\xd9\x7f\xb3\x7f\xde\xff3\xf8\x01\xfco\xf7_\xe9?
1\xbd\xe7\xf7\xdf\xf5\xff\xf0\xbf\xda~\xef\x7f\xa2\xff\xff\xf1\xc1\xe6?
\x9b\x7f\x8d\xfe\xdb\xfeG\xfe\'\xf8\x9f\xff\xff\xf9\xbe\xef\xffz\xff\t\xfd\xe7\xfc\x9f\xfaO\xef_\xff\xff\xeb~\x15\xfe\x8f\xfd\
\xef?\x7f\xff\xef\xfe\x80\xfe\x91\xffy\xfe\xdb\xfe3\xfd\xcf\xf7\xbf\xff\xff\xf9\xfe\xb2?
\xe1\xff\xa0\xfd\xe3\xf9G\xfds\xfew\xfc\x9f\xf4\x1f\xe7\xbf\xec~\xff\xff\xff\xfc\x05\xfc\x9f\xfa\x9f\xfb\xcf\xef\xbf\xe7\x7f\x
\xe5\x7f\xd0\x7f\xf6\xffo\xf4\xf7\xfe\xf3\xffG\xfa\xef\xf7\xbf\xfe\x7f\xdb\xff\xff\xff\xff\xfam\xfd3\xfd\x87\xfe\xbf\xf5\xbf\x
\xff\xff\xf5\xbe\x19?\x80\x7f\xba\xff\xf3\xff3\xfe\xe7\xca\x7f\xa7\x7f\x8c\xffw\xf9\x07\xfb\xa1\xf3W\xe8?
\xc3\xff\x86\xff\x19\xfe\x93\xfc\x87\xf7O\xfd>\xbd\xfeS\xf4\xcf\xdb\xff\xbb\xff\x8b\xff)\xfd\xbf\xff\x87\xfa\xff\xc4/\xd2?
\xcc\xfc\xda\xf4\xf1\xf3\xff\xd9\xff\xda\xffg\xfb\xc1\xeeo\xf3o\xb7?\x83\xfe\xe5\xfe?
\xfd\xe7\xf8/\xdc\x8f\xc5_\xd1\x7f\xb9\xff)\xfe\xb3\xfd\xa7\xf9\xaf\xfb\xff\xee\xfd\xa5\xfdg\xf7?\xf5\x9f\xe2?
\xca\xff\xc7\xff\x13\xfba\xf6\x0b\xf8\xe7\xf2/\xee\x7f\xda\x7f\xc8\xff\xa5\xfe\xe5\xfb\x97\xf7\xed\xfa?
\xf9\x7f\xd3\xf7\x83\xed\xbf\xec?
\xf0\x7f\xb6\xf6\x05\xf6\x1b\xe8\xbf\xea\x7f\xbf\xff\x96\xff\xb7\xfe\'\xf7\x87\xed\xe7\xec\xff\xdf\x7f\xa6\xfd\xd2\xff7\xff\xf
\xd1\xbf\xae\x7f\xa8\xfe\xfd\xfe\x87\xfe\xbf\xf8\xbf\xff\xff\xf8\x7f\x0b\xff\xa9\xff\x83\xfd\xb7\xee\xcf\xa6W\xe6\x7f\xde\x7f\
\xec\xff\x98\xff\xff\xff\xf7\xf2;\xfc\x1f\xfc\xdf\xe9\xbf\xd8\x7f\xec\xffk\xff\xff\xfe\x9f\xcc/\xa6\xff\xf1\x7f\x9e\xff_\xff\x
\x14\xc2\xcc\x98\x8c|\xa0\x80S\x0b2b1\xf2\x82\x01L,\xc9\x88\xc7\xca\x08\x050\xb3&#\x1f(
\x14\xc2\xcc\x98\x8c|\xa0\x80S\x0b2b1\xf2\x82\x01L,\xc9\x88\xc7\xca\x08\x050\xb3&#\x1f(
\x14\xc2\xcc\x98\x8c|\xa0\x80S\x0b2b1\xf1u\x08D\x92a\x89\x17\x82{M-
3=\xa9\xf5\x04\xcd\xac]\xeb4p\xca\xf1.L\xbcF>P@)\x85\x9901|ipQ\xb8\xc6\xbe/\x9e\x08\xf0Q+\xb7\xf9\xa7^\x1b?`\xd3w\x83P\xeb
Z\xa2\x18\xce\xff%\xf3\xa1\x94\xfe\x14\xcc\xe2\xc6\xe1\xdbTJ\xea\xbbD\x1fs\x9d\xfb\xfc\xea\xd3,m\x1a\xb6l\xbb\x13\xa8\xc5\xb8\
\xe8\xbc\x90\xd1<\x11\xe0\xa2X\x01\x99\x0f\xe2\x9d{\x981\x1e\x8a\xa2\xb8\xdcmH`\xdcu\xdb\xb7\x8a\x82|\'\x86~=];\xef\xdb\xc5\xb8
\x0b\xe7\xb1\\\xdc;ibv\xf2A\xe4PF\xc3\xeb\x0e\x83\x1f\xb7\xafI\x03$l\x8b\xaf\xfb\x96\xd2\x12\x11\x9c\x96a\xb4\xceHp\xbf3\n\xef'
Bo\xc9\xe7$\xb4\x99>#\x95>\xa8<\xe2R;\xf3!\xfd\xfc\x82\xfc0\x94\xf8\xb9r\xd6F\xaeY\x0f\x8e\xf0\xf7\xc8so\x17\x8c\x98\xb4\xf1\x
\x95\xc8\x8c\xae\n\x1dz\x9d$\xf7r\x05\x14z>mC\xfd~\x08\xcf\xc6)\xf4\xc4Z\xbe\xd3\xcd\x0b\xa2\xd4\x0b\x80\x97\xa4\x83\xe1`\x9b\
\x02\x12Da\xdc\xee\x1e\x08\xf0Q\x01\xbd\x03J}\x95\xcb<g\xab\xd1\x92\xfd\x9f+L\x98l\x95j\x08d\x12(t_\xb5Nd\x99\xdbgF\x0c\x9b\x1
z\x02\x19\xee\x92/\x8b\xdf\x08r\xad\xa6rC\x85\xf9C\x80\xd4P"w\xb1\x04\x9cti\x98\x91\xcf\x8aqcKr\xaf\xf5ZM\xc82\x10\xab\xb1\xa1'
\xc58D\x82\x1f\xbdJu\x02\x94\x98\t\xe6\xbcz\'\xf2f\x80\x99<\x10\x04@\x1f5\xf1\x04$\xe1\xc6\x18\xa3/xDnD\xf7<\xf5;\xcf\xb8c\x17'
\x1d\x98\xc6\xd7\xb5\x19\x99\xcc\xf3\xee\xbd\xfc\xcb\xe43M\xe5\xce\x9f\xea\x84\n\xc6\x94U<\x90\xabE\xae\x1d\xb5D\xb0\x01\x8a\x
\x9e\xbd\xb7\xf6\x7f\xf0\xb3T\xf6uU\xebj\xef\xefR\x05\xeaK\xf1\xb7\xee\\W\xc0\xcc\xce}\xf5\xee\xa4\xf2E\\\xb5\x87\x06\x8e*\x9d'
\xf7\x15\xf6t\xcd\xee\x08\x8f\xfd\xb8\xfa=\x90\x80\x1b\xbb\xd0\x9d\xe4-
\x9d\xc9\x16\xb0a\xe1\xd2\xfd\x1b\n\x0f|\xdc3\x9e\xd4\xb0\xda)\xba\xaa\xbc\xe7A3C\xc9\xe3\x0e\x87}pP\xdb\x1dzI\x8b\xa4\xb4\xba'
\xdbU\xe8\xc9~\xd59\xb0\xd2t\x86\xfd?\x93O?\xac9\xfaL0\xd6\xc5\x81\x03\xb55\xb9S\xca\xaa?\xb5\xe3\xc7k\x01e\xdf\x9d\x97?
\xc3\xdfsc[\x16\xc8\r\xb6\xbbT\x9eb\xae\xcb\x01\x9e\x04\xb3\x8b\xbe\xcc\x156I\'>K\x7f\xdeQ\xb6w\xf9\xba\xa1\x9b\xe5\xba\x89`\x
^\xbb K!\x0caENX\x01\x99\x10s\xba\xe1\xb74
,\x8c\xc4\x08\xec\x7f]\xe6\xca*b\\\xecI\xf6\xe3\xb2\xca\xb9\xb8v\xd2\x17\xd6\x94\xfd\x18JL\xe4Lx\xa7\xd0\xb7/\x95\xc0\x11\xe9\
\xfe\x05\xebB\x89\xf6\xd7\xe0\\\xc4h\x9ee\xf4\xdb\xc7\x9e\x08v\x9d\xf2\x9c\x8aE\xa8\xd3n\\\x99\x16\xbbB3*\xc0\x12\xae\xcf\x0f{'
g\xc2hih2\xc5!,\xc9\x88\xc7\xca\x08\x050\xb3&#\x1f-
>MT\x99I\xc9\x03Q\xc9\x82\x06\xfb\xc0Nq\x9cK&i\x9e\xcc\x9b\xd2\xac\x91.\x02F\x91\xb3`d\x9f~\x9f\xaf\xcd\x1c\xed\xaa%x\x0c\xef\
\xa3\xed\nb\x10\xca\xfa?
x\xd6U\xcd\xc3\xb6\x93\xeeX\x97\xb6%g\xf1Szt\xd8\xb8<\x85\xfc*93I\x86d\xce\x19\x91\x07)\xdf\xa8d\xc5\xbf\xe7\xda\xa0\xe7u\xc3p'
\x93{\xcc7\x0e\xdaE\xb2\xf0\x91\xf3\xe2F\r^\x8c\x97\xedS\x8d\xe4\x17\t\xe7\xce\x8a:\x1b\x87mQ,\x00\xcc&H+\x89\xf92\xf1\x18\xfc'
\x14\xc2\xcc\x98\x8c|\xa0\x80S\x0b2b1\xf2\xc73>\x16d\xc5+\xf2\x82\x01L,\xc9\xaf]\xc9A\x00\xa6\x16uK\xc4\xb92\xf1\x18\xf9A\x00\
[N\xe6\x89\x8c/e@\xed\x93\xe8\x10t\xdei\xf0G\x82\x87*\xb2\xdfg\x14N\x7f2LXQ[n\xc7\xaay\xed\x85v\xdb\x9a\xe8nl\xa2\x97\xcf\x04x'
\x04\xddIM\xd1\x06\x01L\xe3\x1cP\x86\xaa\xae\xeb\x9b\x87mQ,\x00\xcc\x87\xd1N\xeeqw*\xcf_\xbd\x9c\xd3\x1a\xa2(\xf2\x8b\x88<\xb3'
\xca\x901\xe2\xab\xf9\xedy\xa8\x9e\x1d\x04\xce\xb5\xda\x7f\xc88j\xee\xe3_\xb1
\xb5Ndm\xe7\\\x88\xa0\x02\xc9\n\x12b\t\xadL\x99\x04P\xc3
@w;\xaa}A5\xa9\xb3\xf4\xd7\xdf\x1d]s\xdd\x91\xe7\xbc\xc0\x16\x0fQ\x97G\xb7K\x8f\xa5\x9a\x8a\xcbC\x1csG\xd3\xf9\x8cP\xbb\xe5\xa
\xbd@#\xe5\xdf\x9f\x7f\x84N\x01\xdb\xffaGU\xbb\xbf\xb8\x01&\x9f\x81\xbe\x8b\xbfP\x8b\xfe\xbf\x1cTP\x9c\xc0[\xff\xfd\xc3s\x10\x
\xbd\x87Pm\x03\xadU\xadZ\xb3\xee\xbaE[TJ\xea-
\xff(\xd8\x85\r\xe7\x0e\xf2H\xe1\xd8\xde/\x10\r\x9c\xda\xbfzI\xd7\xd9N\xf9\xa0\xd3\r\x1b\xa9\x97\xf3\xd1"\x0ew\\7\x0e\xdaL\xe0'
\xb0\x0c{\xa90\x96\xbc\x90\x87:9\x99\xcaK\xd8,\xfc\xe1\xce\x8d\x80]\xd1\xf72\xc3\x84\xe9\x0e\x9fN\xf8\xa8\x18\xa3b\xcf\xb2EI8\
```

```
question = "i have 200000 in my bank account .am i eligible for tourist visa"

prompt = f"Here is the file content:\n{text}\n\nQuestion: {question}"

model = genai.GenerativeModel("gemini-2.5-flash")
response = model.generate_content(contents=prompt)

print("Answer from Gemini AI:")
print(response.text)
```

```
\xa9\x1d\x01\x08\xf3\xdf\xe4O\xa4\\\xa8K\xc8Q\xb1\xaa\xf1.L\xb8\x11\xea"\xa1\xae*\xe6\xe1\xdbTJ\xf3&\xbf\xf2oR\x82\x01L,\xea\x
Answer from Gemini AI:
\x0c\x9bPRpaw\xd5\xea\xb9|\x90AZ$\x82\xda,\xc6\xd2E<g\xab\xd1\x92\xfd\xaap\xf2\xae\x9d\xe2\x06\x15\xf0\xbb\x91-
Based on the provided information, there is **no specific minimum bank balance** explicitly stated for a Dubai tourist visa.
\xb1\xf7\xc8*\xae\xa2\x99\x87k\x9d\xb6HUnum\xca\x8ea\x99\xf0\xb3&#\x1f&\xebp<\'w\xde2\x95h\xb5\xc3\xb6\xa8yg\xa53\xf7j\x11\xd2'
\x14\xc2\xcc\x98\x8c|\xb2e\x16\x80\t<\xba/\xd3rY\x93\x11\x8f\x94\x10\nafLF>P@)\x85\x991\x18\xf9A\x00\xa6\x16d\xc4c\xe5\x04\x02'
The text mentions:
*xf3\xc2\xa8\x8aI\x888
* **sufficient funds** as an eligibility requirement.
[\xf9\x04\x0b8\xea\xb1Cy"\x92b\t\xac%xfa\x04s\xb3\x1eAVk\xcf\xb4\xe6\x85\x91\x94J\xeeL\xea\x1d\x12\xf3g1Mjl\xfd\x9f\x06z\xabj\
* **Bank statement** (last 3-6 months) ** as a required document.
\x14\xc2\xcc\x98\x8c|\xa0\x80S\x0b2b1\xf2\x82\x01k\x8eh\xeb\xc6\x14\xec\x9a~x\xfcB\xd4\x82\xe9\xf1\xb1\xd7\n\xce\x8a\x97$\xbf\
* Under "Salary / Financial Guidelines", it notes: **"Tourist: Depends on stay & itinerary"**
\x9a\x9a\xc3\x9fQ\xc9\xab\xfDCt\xf0v\x08X\xd5\xb5\xe8\xc9}\x0eB)\x03\xcc\x9f\xb5NlGK\xce\xa4k\x8epG\x99x\x08\x9fH\xb9]\x8f#\x1
\xd5\x0f\xc6R<\xe3;3.\x02\xf2\xb0\xf7\xd0mS\x9b\x11\xc8uQ\x0c\x08\xaehIIMjl\xfd\x9f\x0e\x92W\xc7\xbb\xa7\xd16\xb4\xda\xad\xa6\;
Having ₹200,000 (which is approximately AED 8,800 - 9,000 depending on the current exchange rate) in your bank account *could* b
\x99\xacq\x93D:\xbf\x8c3\x06\xe1*%J\xef\x0f\xdd\x81\x0b\xe0\xe3\xd4\xfd\x00%z>\x9c\xc1\xc6w+\xaa3\xfaRN\x1c\xb8O\x03\xda\xaf\x
\xcc\x02\x98X\xb6G^z.\xe8\xa4\x9c\xb2A\xb9\x9b\x92\xaa\x88;]K\xe2\x14\xfd\x1d\xc7=P\n\xf4\x9d@m\xafh\xe8\xf1=\xe2Q\x05\xa5F\xb
However, whether it's truly "sufficient" will depend on:
*xa7|\x8b\x8bs&\xc8:\xc5Xw,\xac;\x*fem;
* **The duration of your stay:** ₹200,000 might be enough for a 7-10 day trip but likely insufficient for a 30 or 90-day visa.
`\xb6\xfc\x8ch\xd8\x17\x02\x1d\x82\x18~7\x9d\x8a\x14\xbc\xc4_\xc5\x0b\x93\xac\x9eao@W\x82\x85\x99I\x0f\xbc\xb3\xb0iC\x90\xf5\'
* **Your planned itinerary:** High-end hotels, expensive activities, and lavish spending will require more funds.
\x98mk\xa0\x9c\x00V\xa2`w\xf4\xa0Q%\xc3c\xad\xb9@k\x82\x8f\x7f\x9c4\xa4\xde\x90\xf4\x8f\xBdq\xa3\x98XD7\xbdebh\x90\xe1(e\xc533'
* **Whether flights and accommodation are prepaid:** If these major costs are already covered, the remaining funds are for dai
\xae\x91D\x8b\xe1K\xc7>\x19|\x06\x1e\x83/\x0e\x00\xfc\x1eC2\x1f\xcf\xbc6T\xc6\xd0F\x00\x17aju\x15\xc4\xb9%\xa4\xb1@\x87)\xa9;'
**The consistency of your bank statements:** Immigration authorities often look for a stable financial history, not just a r
\x0b2b\x1fr\xf4\x90\n\x1a1\x8b\x17QE$V\x93\xa7`_\xe8K\xa6&\x16\xe5\xfd\x1a\x07\xc8\xd3\xa0\'\xD3\x13\xce\xaa-
```

```
!streamlit run app.py &>/content/logs.txt &
```

```
!pip install streamlit
```

```
%%writefile app.py
import streamlit as st
import google.generativeai as genai
import os

genai.configure(api_key=os.environ["GEMINI_API_KEY"])

@st.cache_data
def load_files():
    combined = ""
    for fname in ["uaevisaeligibility.txt", "visaeligibility.txt"]:
        if os.path.exists(fname):
            with open(fname, "r", encoding="utf-8") as f:
                combined += f"\n\n--- {fname} ---\n"
                combined += f.read()
    return combined

data = load_files()

st.set_page_config(page_title="Visa Eligibility Assistant")
st.title("🌎 Visa Eligibility Assistant")

question = st.text_input("Ask your visa question:")
```

```
    if st.button("Get Answer"):
        if not question.strip():
            st.warning("Please ask a question")
        else:
            prompt = f"""
    Answer ONLY using the following documents:

    {data}

    Question: {question}
    """
            model = genai.GenerativeModel("gemini-2.5-flash")
            response = model.generate_content(prompt)
            st.success(response.text)
```

```
!pip install pyngrok
from pyngrok import ngrok

ngrok.set_auth_token("36huqtjOWepnnMApsJ8Muc7zE02_WZUfyoQCdPHzbESmDqiA")
```

```
!streamlit run app.py &>/content/logs.txt &
```

```
from pyngrok import ngrok

# Kill old tunnels if any
ngrok.kill()

# Create public URL
public_url = ngrok.connect(8501, "http")
print("Streamlit app public URL:", public_url)
```

```
!streamlit run app.py --server.port 8501 --server.address 0.0.0.0 &
```

Chapter 4

Conclusion and Future Work

4.1 Conclusion

This project presented **SwiftVisaAI**, an AI-driven visa eligibility screening agent designed to assist applicants in understanding visa requirements before initiating the official application process. The system leverages a structured pipeline consisting of dataset preparation, embedding generation, Top-k retrieval, and retrieval-augmented generation (RAG) using a large language model.

By integrating vector-based retrieval with a generative model, SwiftVisaAI is able to provide accurate, context-aware, and reliable responses to user queries related to visa eligibility. The modular design of the system ensures scalability and allows future enhancements without significant changes to the core architecture. Overall, the project successfully demonstrates the practical application of AI techniques in solving real-world problems related to visa screening and decision support.

4.2 Future Work

Although SwiftVisaAI achieves its intended objectives, several enhancements can be considered for future development:

- Integration of real-time visa policy updates from official government sources.
- Expansion of the dataset to include a wider range of countries and visa types.
- Incorporation of user feedback mechanisms to improve response accuracy.
- Deployment of the system as a web or mobile application for broader accessibility.
- Implementation of confidence scoring to quantify the reliability of eligibility predictions.