📘 Agentic AI RAG System – Documentation

1. Introduction

The **Agentic AI RAG (Retrieval-Augmented Generation) System** is designed to provide intelligent, context-aware answers by integrating multiple sources of information. This project combines **website scraping, video transcript processing, and vector database storage** with an agent-based approach powered by **Gemini Flash** as the language model.

Unlike traditional RAG systems that rely on a single source of data, this system leverages **multiple knowledge bases**—documentation from the OpenAI Agent SDK and educational video transcripts—to deliver more accurate, relevant, and contextually appropriate answers.

---

2. Objectives

The main goals of the project are:

1. To implement **full website scraping and document chunking** for structured knowledge retrieval.
2. To process and store **video transcripts** for context-rich question answering.
3. To utilize **multiple vector storage solutions** (ChromaDB and FAISS).
4. To build an **agentic system** that intelligently selects the right knowledge base depending on the query.
5. To ensure the solution is **UI-ready** by making it compatible with **Streamlit**.

---

3. System Architecture

The system follows a modular architecture consisting of:

1. **Data Acquisition**
    - **Website Scraping**: The OpenAI Agent SDK documentation is scraped and divided into chunks for efficient retrieval.
    - **Video Transcript Extraction**: Educational or tutorial videos are transcribed, and the text is stored in ChromaDB.
2. **Vector Database Storage**

    - **FAISS**: Used for storing and retrieving embeddings from the website documentation.
    - **ChromaDB**: Used for storing and retrieving embeddings from video transcripts.
3. **Agentic Layer**
    - An **Agent** is built that can decide which vector database to query (FAISS or ChromaDB) depending on the user's question.
    - The agent selects the right tool automatically and returns the answer.
4. **Language Model**
    - **Gemini Flash** is used as the core reasoning engine to generate natural, human-like responses from the retrieved data.
5. **User Interface (Optional)**
    - The system is **Streamlit-compatible**, meaning a front-end can be built for user interaction without changing the backend logic.

---

4. Key Features

- **Full Website Scraping & Chunking**
    - Automatically extracts knowledge from OpenAI Agent SDK documentation.
    - Splits data into smaller, retrievable chunks for accurate responses.
- **Video Transcript Integration**
    - Extracts spoken content from videos.
    - Stores transcript data in **ChromaDB** for semantic search.
- **Multiple Vector Databases**
    - **FAISS** for document-based queries.
    - **ChromaDB** for video-based queries.
- **Intelligent Tool Selection**
    - Agent dynamically decides which knowledge source to use.
    - Avoids irrelevant responses by selecting the correct database.

- **Streamlit Compatibility**
    - Ready for deployment with an interactive web UI.

---

5. Workflow

1. **User Query** → The user asks a question through the interface.
2. **Agent Decision** → The agent determines whether the query relates to documentation or video context.
3. **Vector Search** →
    - If documentation-related → search FAISS database.
    - If video-related → search ChromaDB database.
4. **Response Generation** → Gemini Flash generates a contextual response.
5. **Answer Delivery** → Final answer is returned to the user.

---

6. Technology Stack

- **Web Scraping**: Python (BeautifulSoup / Requests)
- **Vector Storage**: FAISS, ChromaDB
- **Embedding Models**: Hugging Face models
- **LLM**: Gemini Flash (OpenAI-compatible API)
- **Agent Framework**: OpenAI Agent SDK
- **Frontend**: Streamlit (optional)

---

7. Example Use Case

User Query: “How can I extend an Agent with a custom tool?”

- Agent identifies this as documentation-related.
- Searches FAISS (OpenAI SDK Docs).
- Retrieves relevant section and generates explanation.

User Query: “What was explained about tool selection in the YouTube video?”

- Agent identifies this as video-related.
- Searches ChromaDB (video transcript).
- Retrieves explanation from video context.

---

9. Conclusion

The **Agentic AI RAG System** demonstrates a robust and flexible approach to **multi-source knowledge retrieval**. By integrating both **documentation** and **video transcripts** into a single agentic framework, it ensures accurate and context-aware answers. The modular architecture makes it scalable, adaptable, and ready for production deployment with minimal adjustments