**Documentation and User Guide for Content Engine**

**Overview**

The Content Engine is a system designed to analyze, compare, and highlight differences across multiple PDF documents. It uses Retrieval Augmented Generation (RAG) techniques and integrates local embedding models and a Large Language Model (LLM) to ensure data privacy and high performance. The engine also provides an interactive chatbot interface for querying insights from the documents.

**System Architecture**

**Components:**

1. **Backend Framework**:
   * **LlamaIndex** or **LangChain** for retrieval tasks.
2. **Frontend Framework**:
   * **Streamlit** for building an interactive web application.
3. **Vector Store**:
   * Options include ChromaDB, Pinecone, Faiss, etc. ChromaDB is used in this implementation for managing document embeddings.
4. **Embedding Model**:
   * A local embedding model generates vector representations of document content.
5. **Local LLM**:
   * Processes queries and generates contextual insights locally.

**Installation and Setup**

**Prerequisites:**

* Python 3.8 or higher
* Packages: streamlit, llama-index, langchain, chromadb, pypdf2

**Steps:**

1. Clone the GitHub repository containing the Content Engine code.
2. Install required dependencies:

pip install -r requirements.txt

1. Ensure the local embedding model and LLM are installed and configured.
2. Start the Streamlit application:

streamlit run app.py

**Features**

**Document Parsing**

* Upload PDF files for content extraction and analysis.

**Embedding Generation**

* Generates vector embeddings from the content of the uploaded documents.

**Vector Store**

* Stores embeddings for efficient querying and retrieval.

**Query Engine**

* Processes user queries to retrieve and compare document insights.

**Chatbot Interface**

* Provides an intuitive platform for interacting with the system.

**User Guide**

**Step 1: Upload Documents**

1. Launch the Streamlit app using the command:

streamlit run app.py

1. Use the file uploader to upload one or more PDF documents.
2. Wait for confirmation that the documents have been successfully uploaded and parsed.

**Step 2: Generate Embeddings**

* The system will automatically generate vector embeddings for the uploaded documents.
* A success message will confirm completion.

**Step 3: Query the Engine**

1. Enter your query in the text input box, e.g., “What are the risk factors associated with Google and Tesla?”
2. Submit the query.
3. View the generated response displayed below the input box.

**Step 4: Interpret Results**

* The chatbot provides contextual insights and compares relevant information across the documents.

**Sample Queries**

1. **Risk Factors**: “What are the risk factors associated with Google and Tesla?”
2. **Revenue Comparison**: “What is the total revenue for Google Search?”
3. **Business Model Differences**: “What are the differences in the business of Tesla and Uber?”

**Troubleshooting**

**Common Issues:**

1. **File Upload Failure**:
   * Ensure the uploaded file is in PDF format.
   * Check file permissions.
2. **Embedding Generation Errors**:
   * Verify the embedding model is correctly installed and configured.
   * Check system memory availability.
3. **Query Issues**:
   * Ensure queries are within the scope of the document content.
   * Restart the application if the query engine fails to respond.

**Future Enhancements**

1. Integration with advanced LLMs for improved context understanding.
2. Support for additional document formats (e.g., Word, Excel).
3. Enhanced visualization for document comparisons.

**Conclusion**

The Content Engine offers a robust platform for document comparison and analysis. Its integration of local models ensures privacy while providing efficient insights. Follow the installation steps and user guide for a seamless experience.

For further assistance, refer to the GitHub repository documentation or contact the support team.