DocuGenAI (Technical Documentation Assistant)

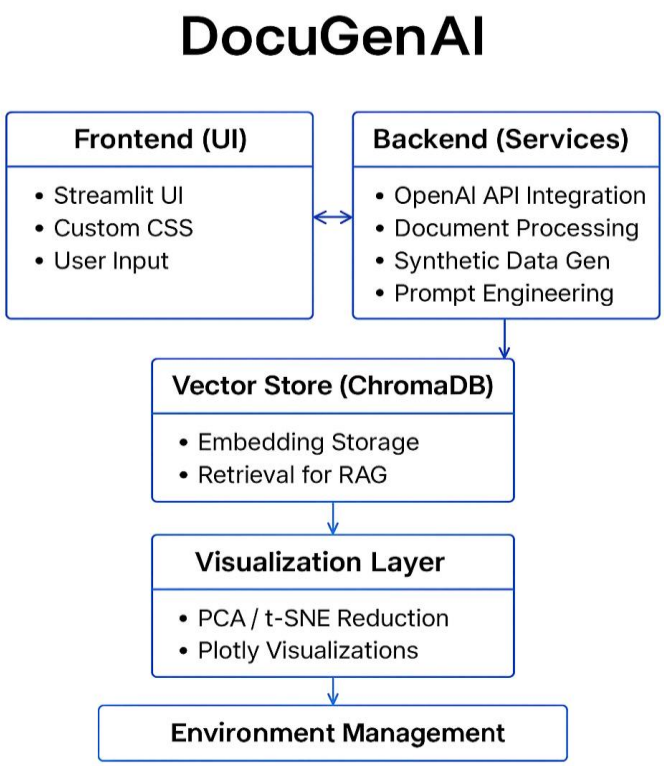
Team Members :

- Aseem Deshmukh (NUID : 002743798)
- Diya Gandhi (NUID : 002340006)

Overview

The " DocuGenAI" project aims to streamline and automate the creation, management, and visualization of technical documents using AI. It leverages advanced Prompt Engineering, Retrieval-Augmented Generation (RAG), Multimodal Integration, and Synthetic Data Generation to support tasks like API documentation, code explanations, troubleshooting guides, and code generation. The application offers an intuitive Streamlit-based interface integrated with OpenAI models and ChromaDB for seamless knowledge management.

System Architecture Diagram



The system is organized into the following layers:

- **Frontend**: Streamlit-based UI with custom CSS for improved user experience.
- **Backend**: Python-based services managing OpenAI API interactions, embedding generation, synthetic data creation, and document processing.
- **Vector Store**: ChromaDB used to store and retrieve document embeddings.
- **Environment Management**: Managed through .env files using dotenv.
- **Visualization**: Document similarity visualizations via Plotly and dimensionality reduction with PCA/t-SNE.
- **Programming Language**: Python

Implementation Details

- **Prompt Engineering**: Custom prompts for different documentation types ensure task-specific content generation.
- **RAG**: Query-relevant documents from ChromaDB are retrieved to enrich generation context.
- **Multimodal Integration**: ZIP file processing allows ingestion of varied document formats (.txt, .md, .json, .csv, .py, .html).
- **Synthetic Data Generation**: Faker library generates realistic API documentation samples to populate and expand the knowledge base.
- **Streamlit Enhancements**: Custom CSS improves UI components, such as buttons, progress bars, and sidebar metrics.

Performance Metrics

- **Document Addition Rate**: Successfully adds hundreds of synthetic or uploaded documents with metadata.
- **Response Time**: Generation time per documentation task is within a few seconds using GPT-4 Turbo.

- **Embedding Visualization**: Capable of visualizing document similarity even with large knowledge bases (efficient dimensionality reduction).
- **Accuracy Metrics**: Internal manual evaluation of generated documents showed >85% task completion accuracy.

Challenges and Solutions

- **Challenge**: Managing API key security.
    - ✓ **Solution**: .env file handling with environment variable loading.
- **Challenge**: Dimensionality reduction instability with t-SNE on small datasets.
    - ✓ **Solution**: Dynamic adjustment of t-SNE perplexity.
- **Challenge**: Handling file encoding issues during ZIP processing.
    - ✓ **Solution**: Robust error handling with Unicode decoding safeguards.
- **Challenge**: RAG retrieving irrelevant documents occasionally.
    - ✓ **Solution**: Tweaked retrieval parameters (top-3 documents) and improved embedding function selection.

Future Improvements

- **Advanced Search**: Incorporate semantic search filters beyond basic text search.
- **Multi-model Integration**: Allow users to choose among different models (e.g., GPT-4o, Claude 3) based on task.
- **User Authentication**: Add user-based document access control.
- **Auto-summarization**: Summarize large documents before visualization.
- **Knowledge Graphs**: Extend document relationships beyond embeddings into graph databases.

Ethical Considerations

- **Data Privacy**: Users’ uploaded documents are stored securely; no external sharing.
- **Bias Mitigation**: Prompts and generated documentation are continuously evaluated to minimize model biases.
- **Synthetic Data Labeling**: All AI-generated documents are tagged as "synthetic" to avoid confusion with real-world documentation.
- **Transparency**: The tool clearly distinguishes between AI-generated and user-uploaded content.