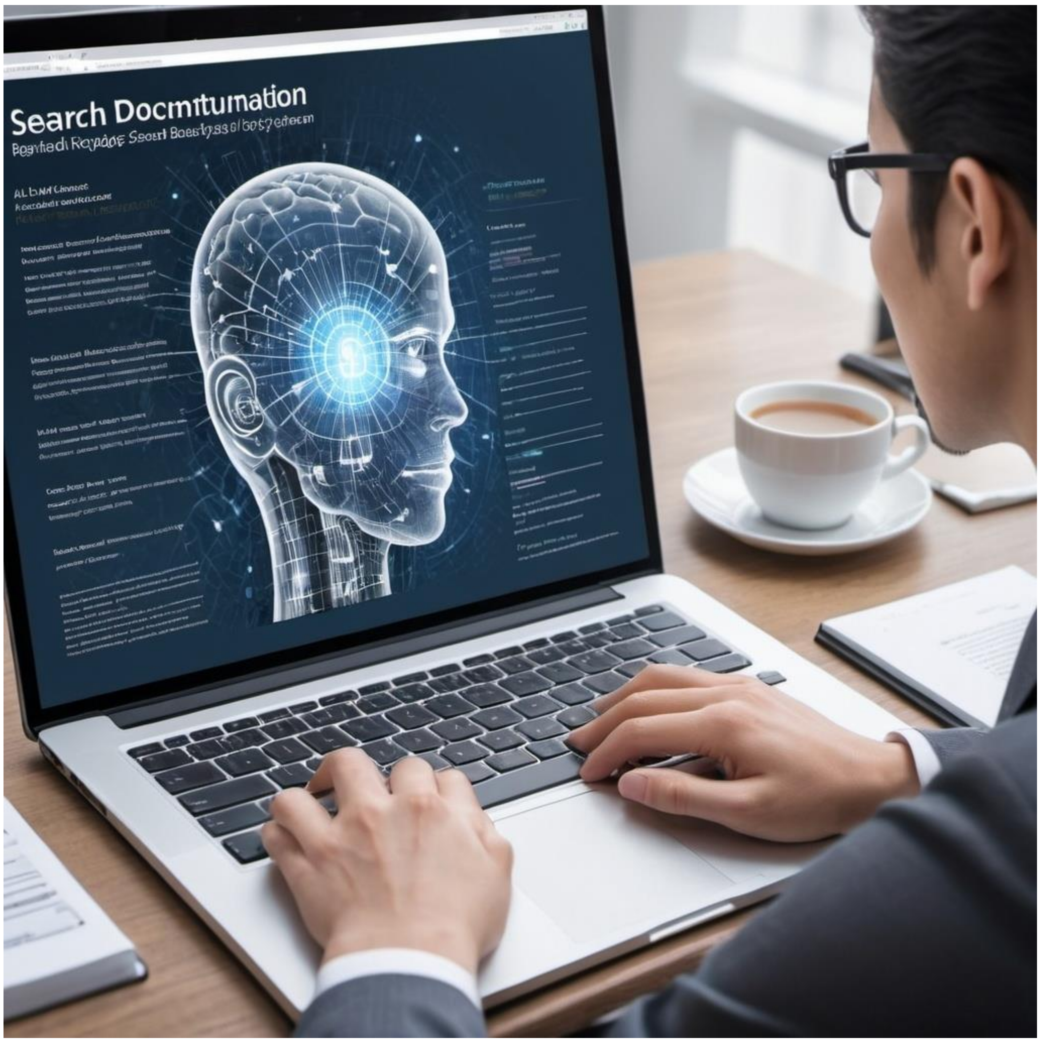


# AI-Based Knowledge-Based Search Retrieval System Documentation

Raeesul Islam – AI Engineer

MSc in Big Data Analytics (Reading),

BSc in IT (Hons) Specialization in Data Science

# Table of Contents

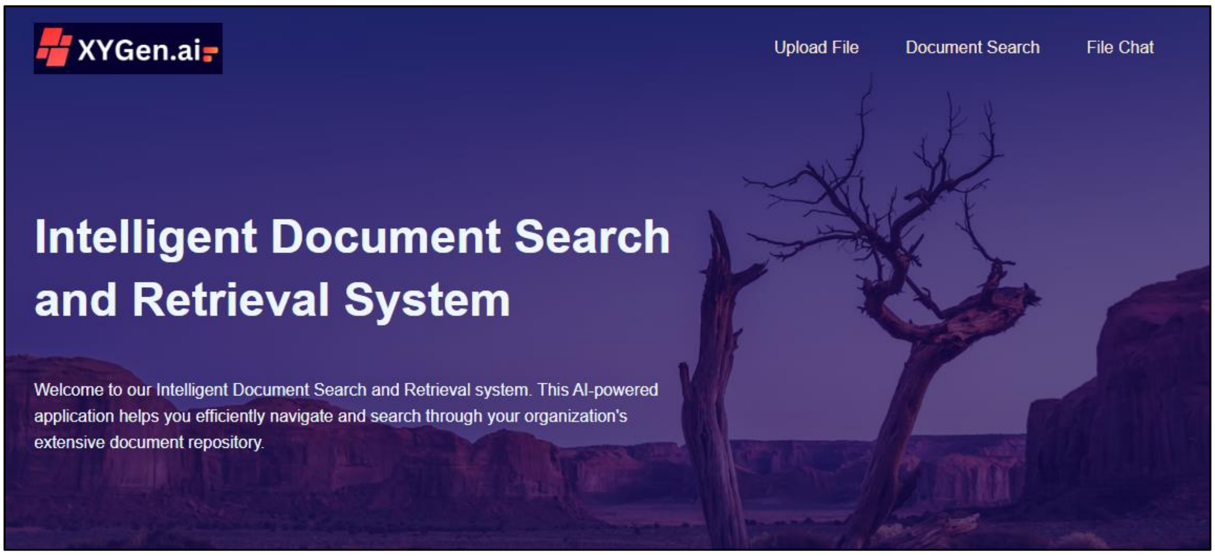
## Contents

## Overview

The **AI-Powered Knowledge-Based Search Retrieval System** combines Large Language Models (LLMs) with vector databases and provides accurate responses, genuinely context-aware and relevant to complex domain queries. This system will be developed based on the Retrieval-Augmented Generation (RAG) architecture that allows document upload, search among uploaded documents, and chat. This system is designed for any company looking to enhance its document search and retrieval capability with AI-driven insight.

### Core Functionalities

- **Document Upload:** It enables the user to send documents in the form of a PDF to the system.
- **Document Search:** This allows searching through the terms in uploaded documents to retrieve relevant sections.
- **Chat with Documents**: Allows users to interact with documents via a chat interface to ask questions and get context-aware responses.



Intelligent Document Search and Retrieval System

Welcome to our Intelligent Document Search and Retrieval system. This AI-powered application helps you efficiently navigate and search through your organization's extensive document repository.

## System Components

### Knowledge Base Search System

This module handles document processing, vector storage, and querying using Google's Generative AI and FAISS (Facebook AI Similarity Search).

**Key Classes & Methods:**

- **`KnowledgeBaseSearchSystem`**: The main class managing document loading, search, and query response.
  - **`load_documents()`**: Loads and processes documents (Uses `PyPDFLoader` and `RecursiveCharacterTextSplitter` for PDF handling and text chunking), storing vectors in FAISS.
  - **`search_documents()`**: Searches documents for occurrences of a query and generates LLM responses.
  - **`qa_conversation()`**: Handles interactive Q&A with documents using context retrieval and LLM.
  - **`llm_process_query()`**: Processes queries with LLM and returns detailed responses.

### Database Processor

Manages interactions with the MongoDB database where document metadata is stored.

**Key Functions:**

- **`insert_document()`**: Inserts document metadata (e.g., name, type, notes, page count) into the MongoDB database.

### Application Interface (app.py)

This module sets up the web application using Flask, providing routes for document upload, search, and chat.

**Key Routes:**

- **`/upload`**: Handles the uploading of PDF documents, storing them locally, and processing them in the knowledge base system.
- **`/search`**: Enables users to search for terms within the uploaded documents and retrieve results.
- **`/pdfchat`**: Provides a chat interface where users can ask questions related to the uploaded documents.

## System Setup and Execution

### Prerequisites

1. **Python Environment**: Ensure Python 3.8+ is installed.
2. **Libraries**: Install required Python libraries using:

```
pip install -r requirements.txt
```

3. **Database**: Set up MongoDB locally or on a server, and ensure it is running.
4. **API Keys**: The system requires Google API keys for the LLM functionality. Set the `‘GOOGLE_API_KEY’` environment variable before running the system.

### Running the System

1. **Start MongoDB:**
   - Ensure MongoDB is running and accessible at `mongodb://localhost:27017/`.
2. **Set Environment Variables:**

```
export GOOGLE_API_KEY='your-google-api-key'
```

3. **Start the Flask Application:**

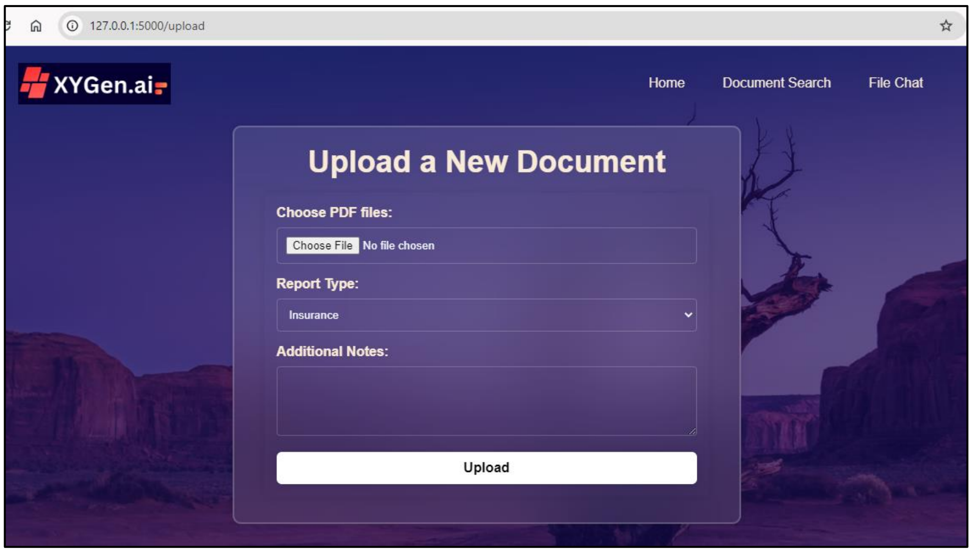
```
python app.py
```

   - The Flask server will start in debug mode, and the application will be accessible at `http://127.0.0.1:5000/`.

## Usage

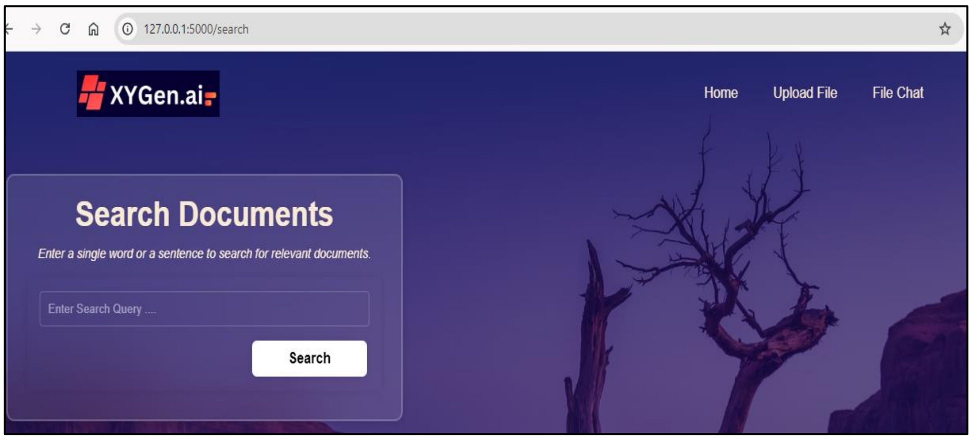
### Document Upload

- Access the upload interface at `http://127.0.0.1:5000/upload`.
- Choose a PDF file, enter the document type and notes, and upload it.
- The document will be processed, and the metadata will be stored in MongoDB.



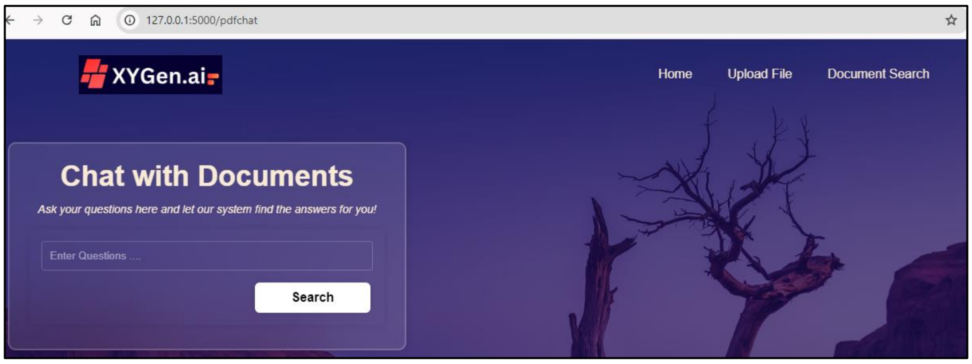
### Document Search

- Navigate to `http://127.0.0.1:5000/search.`
- Enter a search term to find relevant sections across uploaded documents.
- The system will return a list of occurrences, and a summary generated by the LLM.



### Chat with Documents

- Visit `http://127.0.0.1:5000/pdfchat.`
- Ask questions related to the content of the uploaded documents.
- The system will provide a context-aware response based on the document content.



## Dependencies and File Structure

### Dependencies

- **Python Libraries**: `‘Flask’`, `‘pymongo’`, `‘faiss-cpu’`, `‘langchain’`, etc.
- **Database**: `MongoDB`
- **External Services**: `Google Generative AI` for embeddings and LLM processing.

## File Structure

```
│
├──── uploads/                       # Uploaded files
├──── faiss-index/                   # FAISS index
├────  backend/
│     ├──── app.py                    # Flask application
│     ├──── knowledge_base_search_system.py # Knowledge base system
│     └──── database.py                   # Database interaction module
├──── templates/
│     ├──── index.html               # Homepage template
│     ├──── upload.html              # Upload page template
│     ├──── search.html              # Search page template
│     └──── pdfchat.html             # Chat page template
├────  static/                       # Static files (CSS, JS, images)
└──── requirement.txt                # required libraries and dependencies
```

## Conclusion

This **AI-based Knowledge-Based Search Retrieval System** uses the latest AI techniques for innovative document searching and retrieval. It provides a robust backend with an easy-to-use user interface and thus effectively handles and interactively communicates large volumes of documents within an organization having extensive documentation.