AI Chat-Based Documentation Finder

Setup Instructions

1. Clone the repository:

```
git clone <repository-url>
cd <repository-name>
```

2. Create a virtual environment and activate it:

```
python -m venv venv
# On Windows:
.\venv\Scripts\activate
# On macOS/Linux:
source venv/bin/activate
```

3. Install dependencies:

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

4.Add your OpenAI API key in a .env file:

```
OPENAI_API_KEY=your_api_key_here
```

5. Run the application:

```
streamlit run app.py
```

Architecture Explanation

The application consists of the following components:

- Frontend: Built with Streamlit for an interactive chat interface.
- LLM Integration: Uses OpenAI's GPT API to answer queries based on document content.
- Document Processing: Utilizes PyPDF2 and LangChain to process PDF/text files.
- Vector Store: FAISS is used to index document embeddings for fast semantic search.
- Data Management: Uses the local file system and SQLite for data storage and logs.

🗀 Project Structure

AI Chat-Based Documentation Finder

```
.
├── app.py                    # Main application entry point
├── utils/
│   ├── document_processor.py    # Parses and chunks uploaded documents
│   ├── llm_handler.py           # GPT API integration logic
│   ├── db_handler.py            # SQLite operations and logging
│   └── vector_store.py          # FAISS vector indexing and searching
├── data/
│   ├── documents/               # Stored uploaded documents
│   └── index/                # Vector store index files
├── requirements.txt             # Dependency list
└── README.md                    # Documentation
```

AI Chat-Based Documentation Finder

API Usage Guidelines

The application communicates with OpenAI's GPT API to process user queries.

Requirements:

- A valid OpenAI API key
- Sufficient credits on your OpenAI account

Best Practices:

- Store API keys securely using environment variables
- Handle rate limits and exceptions gracefully
- Monitor API usage and log errors for debugging