# Chatbot-Ollama - System Documentation

## 1. Overview

Chatbot-Ollama is a system that allows users to upload files, create searchable indexes, and interact with a chatbot to query indexed data. The system consists of a backend API, built with FastAPI, and a frontend UI developed using Streamlit. The chatbot utilizes Ollama's LLM models for text generation and Pinecone for vector storage.

## 2. Architecture

The system consists of two main components:

* • Backend: Handles file uploads, index creation, and chatbot responses.
* • Frontend: A Streamlit-based UI for interacting with the system.

## 3. Embedding Process

1. User uploads files through the frontend.  
2. Files are sent to the backend, where they are processed and embedded using an NLP model.  
3. The embeddings are stored in Pinecone as vector representations.  
4. These embeddings are used to fetch relevant information when queried in the chatbot.

## 4. Chatbot Interaction Flow

1. The user enters a query in the chatbot.  
2. The query is converted into an embedding and matched against stored vectors.  
3. The most relevant data is retrieved and sent to the LLM for response generation.  
4. The chatbot streams the response back to the user.

## 5. API Endpoints

|  |  |  |
| --- | --- | --- |
| Method | Endpoint | Description |
| POST | /upload/ | Uploads files and creates an index. |
| GET | /list/ | Retrieves the list of created indexes. |
| DELETE | /delete/{index\_id} | Deletes an index. |
| POST | /chat/{index\_id} | Sends a query and retrieves a chatbot response. |

## 6. API Usage Examples

* • Upload Files

```python  
import requests  
files = {'files': open('sample.txt', 'rb')}  
response = requests.post('http://127.0.0.1:8000/upload/', files=files)  
print(response.json())  
```

* • List Available Indexes

```python  
response = requests.get('http://127.0.0.1:8000/list/')  
print(response.json())  
```

* • Delete an Index

```python  
index\_id = 'your\_index\_id'  
response = requests.delete(f'http://127.0.0.1:8000/delete/{index\_id}')  
print(response.json())  
```

* • Chat with the Chatbot

```python  
index\_id = 'your\_index\_id'  
data = {'query': 'Hello, how does this work?'}  
response = requests.post(f'http://127.0.0.1:8000/chat/{index\_id}', json=data)  
print(response.json())  
```

## 7. Setup & Execution

1. Run `setup.bat` to install dependencies and download the LLM model.  
2. Start the backend using `run\_backend.bat`.  
3. Start the frontend using `run\_frontend.bat`.

## 8. Batch Files Explanation

* • `setup.bat`

• Creates a Conda environment named `llm`.  
• Installs dependencies from `requirements.txt`.  
• Pulls the latest `llama3.2` model via Ollama.

* • `run\_backend.bat`

• Activates the `llm` environment.  
• Runs the FastAPI backend (`api.py`).

* • `run\_frontend.bat`

• Activates the `llm` environment.  
• Launches the Streamlit application (`App.py`).