Chat Service Installation Guide

Introduction

This guide provides detailed instructions for setting up a Python-based chat service leveraging a microservices architecture. It covers environment configuration, Python environment setup, and how to run the chat and ingestion services.

The Chat Service employs a microservices architecture to offer scalable, real-time document chat functionalities. This installation guide will help you set up the necessary components.

Prerequisites

- Python 3.8 or later: Installed on your system.
- Network Access: For HTTP communication.

Setting Up Global Environment Variables

Configuring the application correctly involves setting up global environment variables, crucial for the operation of the ingestion service, chat service, frontend, and admin console.

Preparing the .env File

Copy and Rename the env File: Take the provided env file and distribute it to the backend, ingestion, frontend, and admin directories, renaming it to .env within each location.

Content of the .env File: The file should include key variables like:

```
GOVBOTIC_INGESTION_STORAGE="/path/to/ingestion/storage"
GOVBOTIC_FILE_SANDBOX="/path/to/sandbox/storage"
DEEPINFRA_API_TOKEN="your_deepinfra_api_token_here"
OPENAI_API_KEY="your_openai_api_key_here"
GOVBOTIC_LLM="OpenAI:gpt-4-turbo-preview"
```

Environment Variable Reference

A comprehensive list and explanation of the environment variables:

GOVBOTIC_INGESTION_STORAGE:
Directory for storing ingested documents.

GOVBOTIC_FILE_SANDBOX:
Path for temporary files or sandbox environments.

DEEPINFRA_API_TOKEN:
Security token for DeepInfra backend operations.

OPENAI_API_KEY: API
key for accessing OpenAI services.

GOVBOTIC_API_BASE_URL:
The base URL for the application's API.

GOVBOTIC_API_TOKEN:
Token for API request authentication within the application.

GOVBOTIC_ADMIN_SERVER:
URL for the application's admin server.

GOVBOTIC_LLM:
Specifies the LLM for backend service use, formatted as
PLATFORM:LANGUAGE_MODEL.

Supported options include:

- OpenAI: gpt-4-0125-preview
- OpenAI: gpt-4-turbo-preview
- DeepInfra: mistralai/Mixtral-8x7B-Instruct-v0.1
- ChatOpenAI: gpt-4-0125-preview
- ChatOpenAI:gpt-4-turbo-preview
- Groq: mixtral-8x7b-32768, llama2-70b-4096
- Anthropic:claude-2.1

This structure allows the backend service to utilize a wide range of large language models by configuring a single environment variable.

Setup Python Environment

Creating the Virtual Environment

Navigate to your project root and create a virtual environment:

```
python -m venv .venv
```

Activating the Virtual Environment

macOS/Linux: source .venv/bin/activate

Windows: .venv\Scripts\activate

Installing Dependencies

Install required packages:

pip install -r requirements.txt

Running the Chat and Ingestion Services

To get the chat and ingestion services up and running, follow these steps for each service. Ensure that you have completed the setup of your Python environment and the `.env` file distribution as described in previous sections.

Starting the Chat Service

Navigate to the Backend Directory:
Change into the directory containing the backend service:
Jim: cd backend

Start the Chat Service:
Execute the startup script provided with the application. If you encounter any permission-related errors, ensure the script is executable:
./runserver.sh

If necessary, make the script executable with:
chmod +x runserver.sh
./runserver.sh

Starting the Ingestion Service
Navigate to the Ingestion Service Directory: Change into the directory containing the ingestion
cd ingestion

Start the Ingestion Service: Like with the chat service, execute the provided startup script. Adjust permissions if necessary:
bash

./runingestserver.sh

Make the script executable if you encounter a permission error:

```
chmod +x runingestserver.sh
 ./runingestserver.sh
```

Additional Resources and Sample Code

For examples of interacting with the Chat and Ingestion Services, refer to the provided sample code. These can serve as a basis for developing your application's backend integration.