Pi Chat - Global

Pi Chat – Global utilizes OpenAI functions and provides various kinds of chatbots. The application was developed using HTML, CSS and JavaScript as frontend and Flask as the backend. Langchain and Langflow modules are used for optimal utilization of OpenAI functions. It includes basic authentication – user has to register and then login to access all apps. The app was developed in the ‘intern’ conda environment. The file ‘requirements.txt’ contains all the required python modules. The ‘README.MD’ file explains the procedure to run the app.

The apps offered:

1. General chatbot – ask any general queries
2. Custom chatbot - query your own documents
3. Interview Assistant – takes your interview based on your selection and provides a score at the end.
4. Time Travel using Langflow – Travel back in time with this chatbot
5. Dash Apps – Displays various dash apps and how they can be rendered
6. Langflow demo – A video showcasing how we can use langflow chains in our own apps.

# Breakdown of the files:

1. **.env and .env.example**

The API Keys are stored in these files for security purposes.

1. **app.py**

This is the backbone file of the whole app, created using the Flask framework. All routes and forms are handled here. It also handles registration and login.

User authentication is handled using Flask-Login and bcrypt for password hashing.

Various chatbots, including a general chatbot, custom chatbot, interview assistant, and Langflow chatbot, are integrated into the application and have their respective routes and prediction endpoints.

Routes are:

* / - The home page.
* /login - The login page for users.
* /main - The main page that displays all apps after a user logs in.
* /logout - Logs out the user and redirects to the home page.
* /register - The registration page for new users.
* /chatbot - A general chatbot page using OpenAI's GPT-3.5-Turbo.
* /predict - An API endpoint for receiving chatbot responses.
* /custom - A custom chatbot page with file upload functionality.
* /custom\_predict - An API endpoint for custom chatbot responses.
* /interview - An interview preparation page.
* /interview\_questions - An API endpoint for interview question generation.
* /lf\_chatbot - A chatbot using the Langflow framework.
* /lf\_predict - An API endpoint for Langflow chatbot responses.

1. **chat.py**

This file is responsible for predicting chat responses for the general chatbot. It utilizes OpenAI gpt-3.5-turbo model. The OpenAI API key is called using the load\_dotenv() function. The session\_messages list stores the conversation between the user and the AI assistant. The get\_response function is defined to interact with the chatbot. It takes a user message as input, uses the ChatOpenAI class to generate a response, and appends the messages to the session\_messages list.

1. **customchat.py**

This file contains the functions that predict responses to uploaded documents. Embeddings of the document are created using ChromaDB and stored in the database. Buffer memory is used to keep track of the conversation and prompt templates are used to structure responses. Conversation chain class is used to link the messages. A vector query retriever is used to extract relevant documents and generate an answer. OpenAI’s gpt-3.5-turbo model is used for the same.

1. **interview.py**

The code is intended for conducting a technical interview where the chatbot asks questions, evaluates user responses, and awards points based on correctness. The chatbot's behaviour, including question difficulty, scoring, and instructions, is provided through system messages and prompts. The OpenAI GPT-3.5 Turbo model is used to generate responses and conduct the interview. The code is designed to interact with the chatbot by providing user input and receiving responses.

1. **langflow\_import.py**

The code is used to demonstrate how langflow chat models can be integrated with a Flask app and used to generate responses in real time. The model is built on the langflow server and a json file of the code is downloaded and integrated with the app. The langflow model is a Time Travel Guide.

1. **dash\_apps.py**

This Python script utilizes the Dash framework to create web applications for data visualization and interaction. Dash is a Python web application framework that is built on top of Flask, Plotly, and React, making it a powerful tool for building interactive, data-driven web applications. The code demonstrates the creation of multiple Dash applications, each with different features and layouts.

1. **Templates (folder):**

This folder contains all the html files that are rendered using Flask as the backend.

1. **Static (folder):**

This folder contains all the JavaScript and CSS files for the web page.

1. **static\images and static\videos**

These folders contain the images and videos used in the development of the app.

1. **static\files**

This folder is the directory used to save all the chromaDB embeddings and files uploaded by the user in the custom chatbot.

1. **instance (folder)**

This folder contains the database for all the users’ login id and password.

Top of Form