AI-Powered Language Learning Chatbot Architecture

Technologies Used :-

The project utilizes the following technologies:

- **Python**: Core programming language for backend logic.
- **Streamlit**: Web framework used for building the interactive UI.
- **LangChain**: Framework for interacting with OpenAI's GPT-4.O model.
- **SQLite**: Lightweight database for storing user mistakes.
- **OpenAI API**: GPT model for generating responses and corrections.
- **Typing module**: Used for type hints and better code readability.

Project Architecture :-

The project has various component adhering to object oriented programming :

1. User Interface (UI) – Streamlit ->

- The chatbot is embedded within a web-based UI using **Streamlit**.
- Users enter their API key and select learning preferences (language, proficiency level, etc.).
- Messages between the user and AI are displayed interactively.
- The sidebar shows mistake analysis and improvement recommendations.

2. Chatbot Backend - LanguageLearningChatbot Class ->

- **Initialization (__init__)**: Sets up OpenAI API (GPT-4.O) and connects to an SQLite database.
- **Database Management (create_mistakes_table, record_mistake)**: Handles storing and retrieving user mistakes for personalized feedback.
- **Conversation Handling (generate_response)**: Processes user input, sends requests to OpenAI API, and manages conversation history.
- **Mistake Analysis (analyze_mistakes)**: Retrieves mistake data from the database and provides learning recommendations.
- **Contextual Adaptation (get_conversation_scene)**: Adjusts chat topics based on proficiency level.

3. OpenAI Integration – LangChain ->

- The chatbot leverages **LangChain** to interact with OpenAI's GPT-4.O model.
- The system message defines the tutor's role, ensuring responses remain educational.
- Conversations are stored in a structured format (system messages, user input, AI responses).

4. Data Storage - SQLite Database ->

- **Tracks language mistakes**:
    - Mistake Type (Grammar, Vocabulary, Pronunciation)
    - Original and corrected text
    - Explanation for correction
- Used to provide insights into user progress and generate improvement tips.

Workflow :-

1. **User Configuration** → User selects learning language, proficiency level, and enters OpenAI API key.
2. **Chat Interaction** → Users interact with the chatbot, receiving AI-generated responses.
3. **Mistake Detection** → AI identifies language mistakes in user messages.
4. **Error Recording** → Mistakes are logged into the SQLite database.
5. **Learning Analytics** → User mistakes are analysed and provides improvements.

Installation & Setup :-

1. **Install required dependencies:**
    - pip install openai langchain sqlite3 streamlit
2. **Set up OpenAI API key**
3. **Run the Streamlit app:**
    - streamlit run language_learning_chatbot.py

Usage Instructions :-

1. Enter OpenAI API Key.
2. Select learning language.
3. Choose native language.
4. Select proficiency level.
5. Click "Start Learning Session".
6. Begin chatting in target language.

Recommendations for Future Upgrades :-

To enhance the chatbot further, consider the following improvements:

1. Speech Recognition & Synthesis

- Integrate **Google Speech-to-Text** or **Whisper API** to enable **voice-based interaction**.
- Use **Text-to-Speech (TTS)** to provide spoken feedback for better pronunciation learning.

2. Real-time Feedback & Gamification

- Introduce a **real-time grammar checker**.
- Implement **point-based rewards** or **streak tracking** to keep users engaged.

3. Personalized Learning Paths

- Create **customized quizzes** based on frequent mistakes.
- Add **AI-generated lesson plans** for structured learning.

By implementing these enhancements, the chatbot can evolve into a **comprehensive AI-powered language tutor** that is engaging, interactive, and highly effective for learners worldwide.

About the Developer :-

- Tanmay Vijayvargiya (INDIA)
- tanmay.vijayvargiya2003@gmail.com , www.linkedin.com/in/tanmay-vijayvargiya