Assignment 1: Chatbot using API

We have implemented the whole process in several ways t**o build a chatbot using different API integration methods and continuously improved its functionalities.**

Initially we used specifically Gemini API with the model gemini-2.5-flash, accessed directly by the terminal. In this version-01, the user can ask a question and the bot will generate an answer and then when the user enters the specific keyword—"exit" or "quit"—specifying the end of the usage. Next, in Version-02, we implemented the process using Streamlit. To keep the privacy of the API, we directly ask it from the user. This version worked as a **single-turn chatbot**. Just work on a query, not a conversational one! In version-03, we added a continuous chatting version with the st.chat_message function in streamlit. In both versions 2 and 3, the chatbot sends only the *latest* user input each time, not the entire chat history as context. The model doesn't "remember" previous messages, as it treats each input as a new conversation. But in the 4th version, we add the feature named "Chat Memory" to remember the previous responses and also generate context-aware responses.
In the next two phases, we enhanced the chatbot to support multiple AI providers, making it more flexible and powerful. Users can now select any API provider, such as **Gemini**, **OpenAI**, **Groq**, or others, and specify the **model name** (e.g., *gpt-4*, *llama3-7b*, etc.) along with their **API key**.

Feature	Old Version	The last Version
Provider	Only Gemini	Multiple (Gemini, OpenAI, Groq, HF, etc.)
API Handling	Static	Dynamic with importlib + REST
Conversation	Context-aware	Full memory retained
UI	Simple Streamlit	Fully interactive with chat bubbles
Extensibility	Fixed	Can add any new provider easily

Code Link: assignment-1: chatbout using api.ipynb