VerbaVerse: Multilingual Dialogue Dynamo using Gemini Model

Project Description

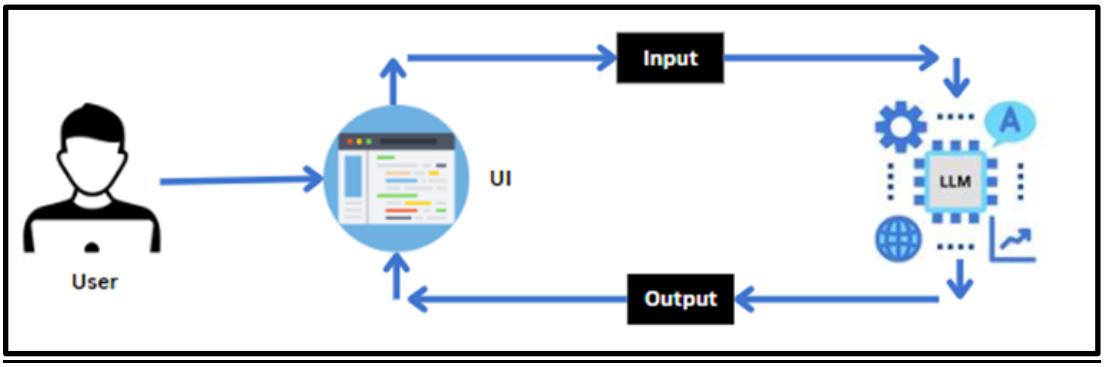
VerbaVerse is an advanced chatbot project powered by the Gemini Pro LLM (Large Language Model) architecture. This innovative system is designed to facilitate seamless communication across multiple languages, offering users a versatile platform for interaction and information exchange.

Scenario 1: Multilingual Customer Support
VerbaVerse enhances customer support services by providing multilingual assistance to users. Whether customers need help troubleshooting issues, seeking product information, or making inquiries, VerbaVerse can efficiently handle their queries in their preferred language. This feature ensures a smooth and satisfactory customer experience, regardless of linguistic diversity.

Scenario 2: Language Learning and Practice
For language learners and enthusiasts, VerbaVerse serves as a valuable tool for practicing and improving language skills. Users can engage in conversations with the chatbot in different languages, receive real-time feedback on grammar and vocabulary usage, and access relevant learning resources. This feature promotes active language learning and cultural exchange, enhancing users' proficiency and confidence.

Scenario 3: Multilingual Content Translation and Summarization
Professionals and individuals dealing with multilingual content can leverage VerbaVerse for translation and summarization purposes. Whether it's documents, articles, or textual data in various languages, VerbaVerse's advanced capabilities enable accurate translation and concise summarization, facilitating efficient information processing and analysis across language barriers.

Technical Architecture



Project Flow:

- User interacts with the UI to enter the input.
- User input is collected from the UI and transmitted to the backend using the Google API key.
- The input is then forwarded to the Gemini Pro pre-trained model via an API call.
- The Gemini Pro pre-trained model processes the input and generates the output in the selected language.
- The results are returned to the frontend for formatting and display.

To accomplish this, we have to complete all the activities listed below:

- **Requirements Specification**
  - Create a requirements.txt file to list the required libraries.
  - Install the required libraries
- **Initialization of Google API Key**
  - Generate Google API Key
  - Initialize Google API Key
- **Interfacing with Pre-trained Model**
  - Load the Gemini Pro pre-trained model
  - Implement a function for text translation
- **Model Deployment**
  - Integrate with Web Framework
  - Host the Application

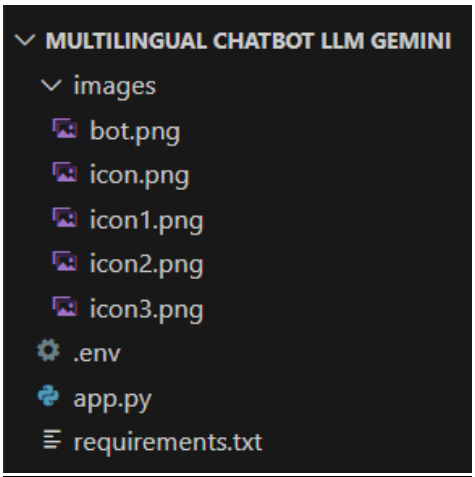
Prior Knowledge:

You must have the prior knowledge of the following topics to complete this project.

- Generative AI Concepts
- NLP: https://www.tutorialspoint.com/natural_language_processing/index.htm
- Generative AI: https://en.wikipedia.org/wiki/Generative_artificial_intelligence
- About Gemini: https://deepmind.google/technologies/gemini/#introduction
- Gemini API: https://ai.google.dev/gemini-api/docs/get-started/python
- Gemini Demo: https://colab.research.google.com/github/google/generative-ai-docs/blob/main/site/en/gemini-api/docs/get-started/python.ipynb
- Streamlit: https://www.geeksforgeeks.org/a-beginners-guide-to-streamlit/

Project Structure

Create the Project folder which contains files as shown below:



- images folder: It is established to store the images utilized in the user interface.
- .env file: It securely stores the Google API key.
- app.py: It serves as the primary application file housing both the model and Streamlit UI code.
- requirements.txt: It enumerates the libraries necessary for installation to ensure proper functioning.
- Additionally, ensure proper file organization and adhere to best practices for version control.

Milestone 1: Requirements Specification

Specifying the required libraries in the requirements.txt file ensures seamless setup and reproducibility of the project environment, making it easier for others to replicate the development environment.

Activity 1: Create a requirements.txt file to list the required libraries.

```
# libraries need to be installed
streamlit
streamlit_extras
google-generativeai
python-dotenv
googletrans==4.0.0-rc1
os
Pillow
```

- streamlit: Streamlit is a powerful framework for building interactive web applications with Python.
- streamlit_extras: Additional utilities and enhancements for Streamlit applications.
- google-generativeai: Python client library for accessing the GenerativeAI API, facilitating interactions with pre-trained language models like Gemini Pro.
- python-dotenv: Python-dotenv allows you to manage environment variables stored in a .env file for your Python projects.
- googletrans==4.0.0-rc1: Googletrans is a Python library for Google Translate API, enabling text translation between languages.
- os: The os module in Python provides a way of using operating system dependent functionality, allowing interaction with the operating system in a portable way.
- Pillow: Pillow is a Python Imaging Library (PIL) fork that adds support for opening, manipulating, and saving many different image file formats.

Activity 2: Install the required libraries

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS                                   powershell

PS C:\Users\dhurvi patel\Desktop\SCIT academic\Smart_Bridge_Internship\SB_Company work\May_Month\Multilingual Chatbot LLM Gemini> pip install -r requirements.txt
```

- Open the terminal.
- Run the command: pip install -r requirements.txt
- This command installs all the libraries listed in the requirements.txt file.

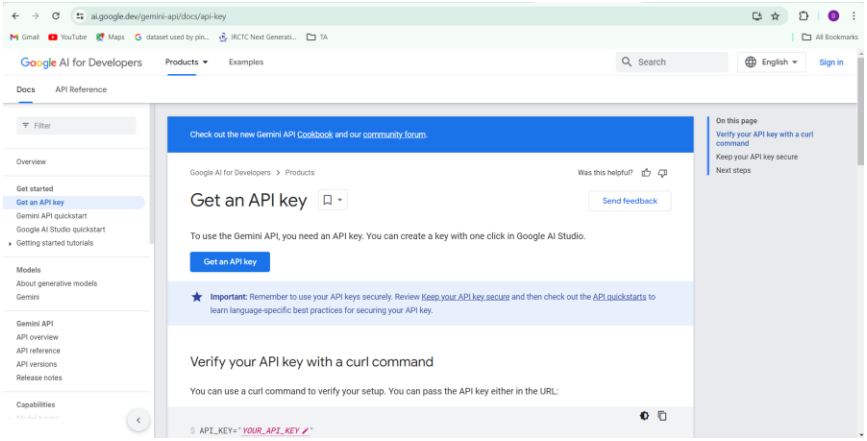
Milestone 2: Initialization of Google API Key

The Google API key is a secure access token provided by Google, enabling developers to authenticate and interact with various Google APIs. It acts as a form of identification, allowing users to access specific Google services and resources. This key plays a crucial role in authorizing and securing API requests, ensuring that only authorized users can access and utilize Google's services.

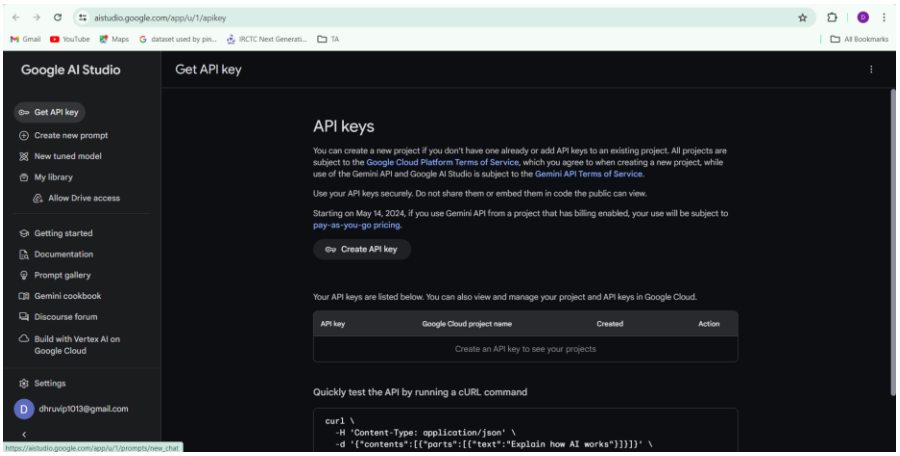
Activity 1: Generate Google API Key

Click the provided link to access the following webpage.

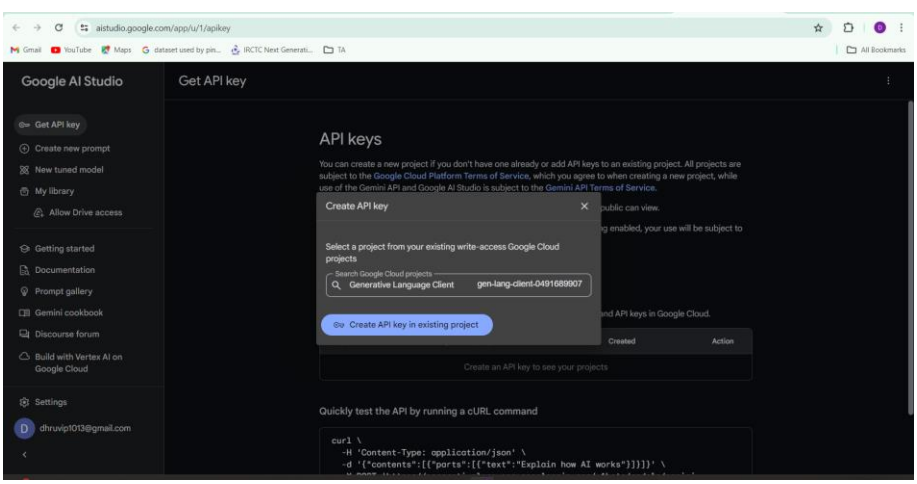
Link: https://ai.google.dev/gemini-api/docs/api-key



After signing in to your account, navigate to the 'Get an API Key' option. Clicking on this option will redirect you to another webpage as shown below.



Next, click on 'Create API Key' and choose the generative language client as the project. Then, select 'Create API key in existing project'.



Copy the newly generated API key as it is required for loading the Gemini Pro pre-trained model.

Activity 2: Initialize Google API Key

```
GOOGLE_API_KEY = "<Enter the copied Google API Key>"
```

- Create a .env file and define a variable named GOOGLE_API_KEY.
- Assign the copied Google API key to this variable.
- Paste the API key obtained from the previous steps here.

Milestone 3: Interfacing with Pre-trained Model

To interface with the pre-trained model, we'll start by creating an app.py file, which will contain both the model and Streamlit UI code.

Activity 1: Load the Gemini Pro pre-trained model

```
from dotenv import load_dotenv
import streamlit as st
import os
from googletrans import Translator
import google.generativeai as genai
from PIL import Image
from streamlit_extras import add_vertical_space as avs

load_dotenv()

# Configure the GenerativeAI module
genai.configure(api_key=os.getenv('GOOGLE_API_KEY'))

# Function to load Gemini Pro model and get responses
model = genai.GenerativeModel("gemini-pro")
```

- The code begins by importing necessary libraries and modules, including dotenv, Streamlit, os, Googletrans, GenerativeAI from Google, PIL (Python Imaging Library), and a custom module for adding vertical space in Streamlit.
- It loads environment variables from the .env file using the load_dotenv() function.
- The GenerativeAI module is configured with the Google API key stored in the environment variable GOOGLE_API_KEY.

- A GenerativeModel object named "model" is created using the Gemini Pro pre-trained model from Google.
- The code is essentially setting up the environment, configuring the GenerativeAI module with the API key, and loading the Gemini Pro model for generating responses to user inputs in the Streamlit app.

Activity 2: Implement a function for text translation

```
def get_gemini_response(question, language="en"):
    translator = Translator()
    question_translated = translator.translate(question, dest=language).text
    response = model.generate_content(question_translated)
    return response.text
```

- The function get_gemini_response takes two parameters: question (input text) and language (target language for translation).
- It initializes a Translator object from the Googletrans library to translate the input question to the specified language.
- The translated question is obtained using the translate method and stored in the variable question_translated.
- The translated question is then passed to the pre-trained model model.generate_content() to generate a response.
- The response generated by the model is returned as text using response.text.

Milestone 4: Model Deployment

We deploy our model using the Streamlit framework, a powerful tool for building and sharing data applications quickly and easily. With Streamlit, we can create interactive web applications that allow users to interact with our models in real-time, providing an intuitive and seamless experience.

Activity 1: Integrate with Web Framework

The webpage is organized into four main sections to provide users with a comprehensive experience:

- Introduction:
    - This section provides information about the application, highlighting its purpose and key features.
    - It configures the page title, sets a custom font style for the header, and displays a welcome message to users.

- Additionally, it includes an image to visually represent the application's branding.

```python
# Streamlit UI
st.set_page_config(page_title="Multilingual Chatbot")
# Header with custom font style
st.title("VerbaVerse")
st.header("Your Multilingual Dialogue Dynamo")
st.markdown("""<p style='text-align: justify;'>
            Welcome to VerbaVerse! Our platform enables seamless multilingual communication, allowing users to engage in conversations and
            receive responses effortlessly across languages. Whether it's personal or professional, VerbaVerse fosters clear communication
            and global connectivity. Join us today to unlock the power of multilingual dialogue!
            </p>""", unsafe_allow_html=True)

img = Image.open("images/icon.png")
col1, col2, col3 = st.columns([1, 3, 1])
with col2:
    st.image(img, use_column_width=True)
```

- Offering:
  - Here, users can explore what the application offers, including its capabilities and functionalities.
  - It displays three columns, each containing an icon and a description of a specific feature offered by the VerbaVerse application.
  - The features include Multilingual Customer Support, Language Learning and Practice, and Multilingual Content Translation and Summarization.
  - Each description is aligned and formatted using HTML markup for justified text, enhancing readability.
  - Additionally, vertical spacing is added between each feature description for better visual separation.

```python
st.header("What we Offer?")

col1, col2 = st.columns([1, 3])
with col1:
    img1 = Image.open("images/icon1.png")
    st.image(img1, width=150)

with col2:
    st.markdown("""<p style='text-align: justify;'><b>Multilingual Customer Support: </b>
            VerbaVerse offers multilingual assistance for troubleshooting, product inquiries, and general support, ensuring a
            smooth customer experience across languages.
            </p>""", unsafe_allow_html=True)

avs.add_vertical_space(4)

col1, col2 = st.columns([1, 3])
with col1:
    img2 = Image.open("images/icon2.png")
    st.image(img2, width=150)

with col2:
    st.markdown("""<p style='text-align: justify;'><b>Language Learning and Practice: </b>
            Users can practice and improve language skills by engaging in conversations with the chatbot, receiving real-time
            feedback, and accessing learning resources, fostering proficiency and confidence.
            </p>""", unsafe_allow_html=True)

avs.add_vertical_space(4)

col1, col2 = st.columns([1, 3])
with col1:
    img3 = Image.open("images/icon3.png")
    st.image(img3, width=150)

with col2:
    st.markdown("""<p style='text-align: justify;'><b>Multilingual Content Translation and Summarization: </b>
            VerbaVerse provides accurate translation and concise summarization of documents, articles, and textual data in
            multiple languages, facilitating efficient information processing and analysis.
            </p>""", unsafe_allow_html=True)


avs.add_vertical_space(4)
```

- Text Translation:

- This area allows users to input text for translation into multiple languages using the chatbot interface.
- The provided code segment creates an interactive user interface using Streamlit. It includes a header titled "Let's Engage" centered using HTML markup.
- An image of a chatbot is displayed below the header, and users can input text into a text field labeled "Input."
- Additionally, there's a dropdown menu for selecting the desired language for communication.
- Upon clicking the "Ask the Question" button, the input text is translated into the selected language using Google Translate, and the translated text is passed to the Gemini Pro pre-trained model to generate a response.
- Finally, the response is displayed in the Streamlit UI.

```python
st.markdown("<h1 style='text-align: center;'>Let's Engage</h1>", unsafe_allow_html=True)
img4 = Image.open("images/bot.png")
st.image(img4, use_column_width=True)
input_text = st.text_input("Input: ", key="input")

# Dropdown for selecting language
selected_language = st.selectbox("Select Language:", options=["Arabic", "Bengali", "Chinese", "Dutch", "English",
                                                              "French", "German", "Greek", "Hindi", "Italian",
                                                              "Japanese", "Korean", "Polish", "Portuguese",
                                                              "Russian", "Spanish", "Swedish", "Turkish", "Urdu"])


# Map language names to language codes
language_map = {
    "Arabic": "ar",
    "Bengali": "bn",
    "Chinese": "zh-CN",
    "Dutch": "nl",
    "English": "en",
    "French": "fr",
    "German": "de",
    "Greek": "el",
    "Hindi": "hi",
    "Italian": "it",
    "Japanese": "ja",
    "Korean": "ko",
    "Polish": "pl",
    "Portuguese": "pt",
    "Russian": "ru",
    "Spanish": "es",
    "Swedish": "sv",
    "Turkish": "tr",
    "Urdu": "ur"
}

# Convert selected language to language code
language_code = language_map.get(selected_language, "en")

# Button to ask the question
submit_button = st.button("Ask the Question")

# When submit button is clicked
if submit_button:
    response = get_gemini_response(input_text, language=language_code)
    st.write(response)
```

- FAQ:
  - Users can find answers to frequently asked questions about the application and its features in this section.
  - The provided code segment creates a FAQ section in the Streamlit UI using Markdown and Streamlit's `info` function.
  - It displays frequently asked questions along with their answers.
  - The questions are styled as informative info boxes.Each FAQ item consists of a question followed by its corresponding answer.
  - This section provides users with helpful information about VerbaVerse and its capabilities.

```
st.markdown("<h1 style='text-align: center;'>FAQ</h1>", unsafe_allow_html=True)
st.info("""What languages does VerbaVerse support for customer support? \n
        VerbaVerse supports a wide range of languages, ensuring efficient
    assistance for users regardless of their preferred language.""")
st.info("""Can VerbaVerse provide real-time feedback on language usage during practice sessions? \n
        Yes, VerbaVerse offers real-time feedback on grammar and vocabulary
    usage, helping users improve their language skills effectively.""")
st.info("""Is VerbaVerse suitable for professionals dealing with multilingual content? \n
        Absolutely! VerbaVerse is designed to meet the needs of professionals,
    enabling efficient information processing and analysis across language
    barriers.""")
```

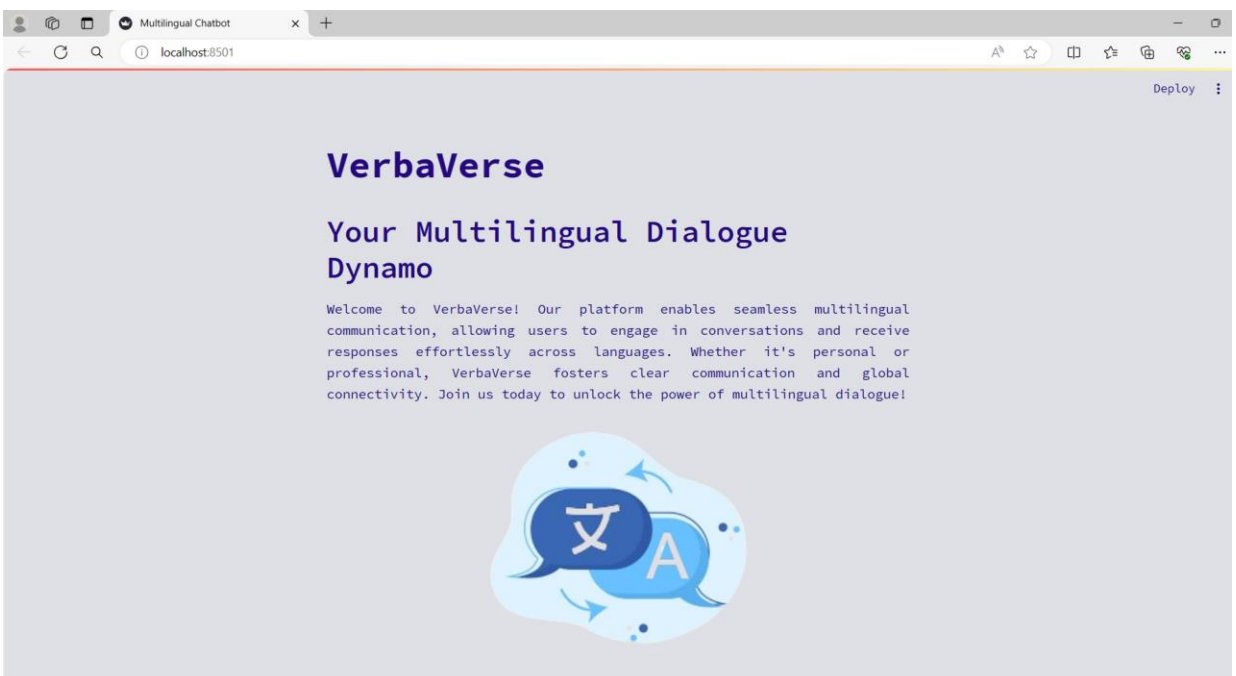
Activity 2: Host the Application

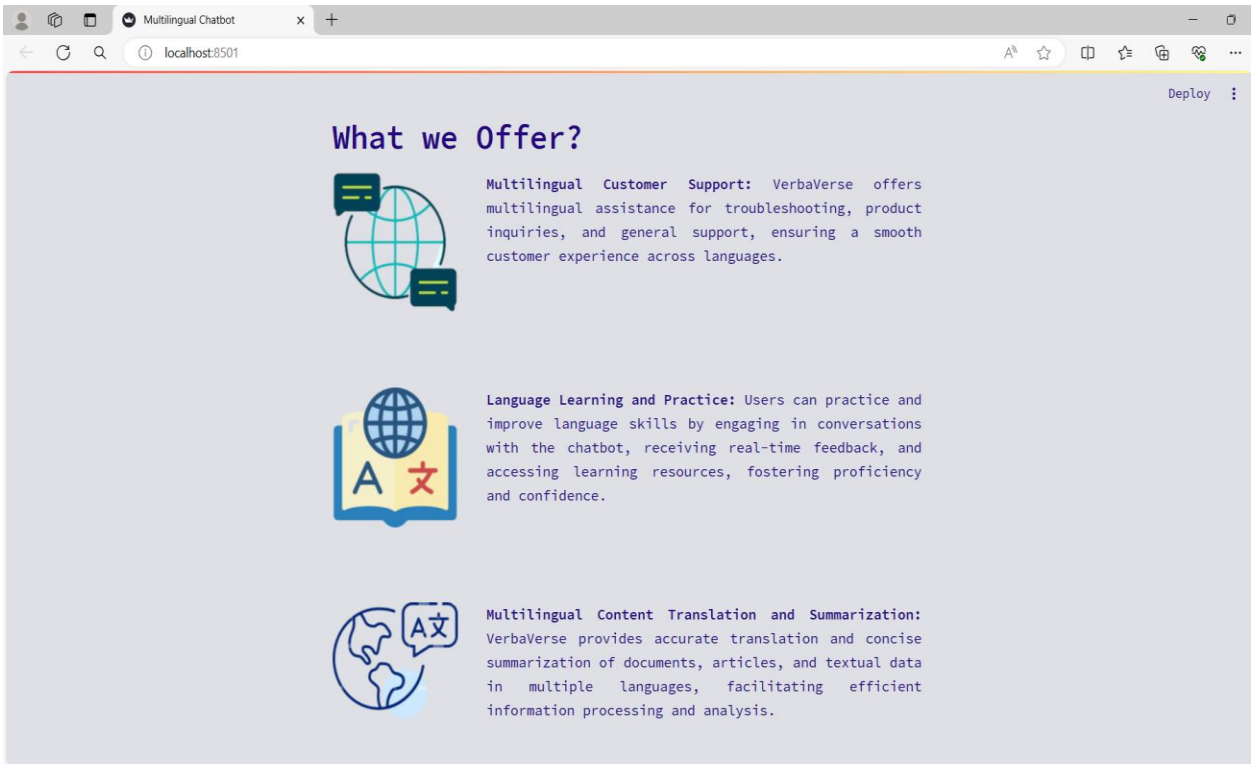
Launching the Application:

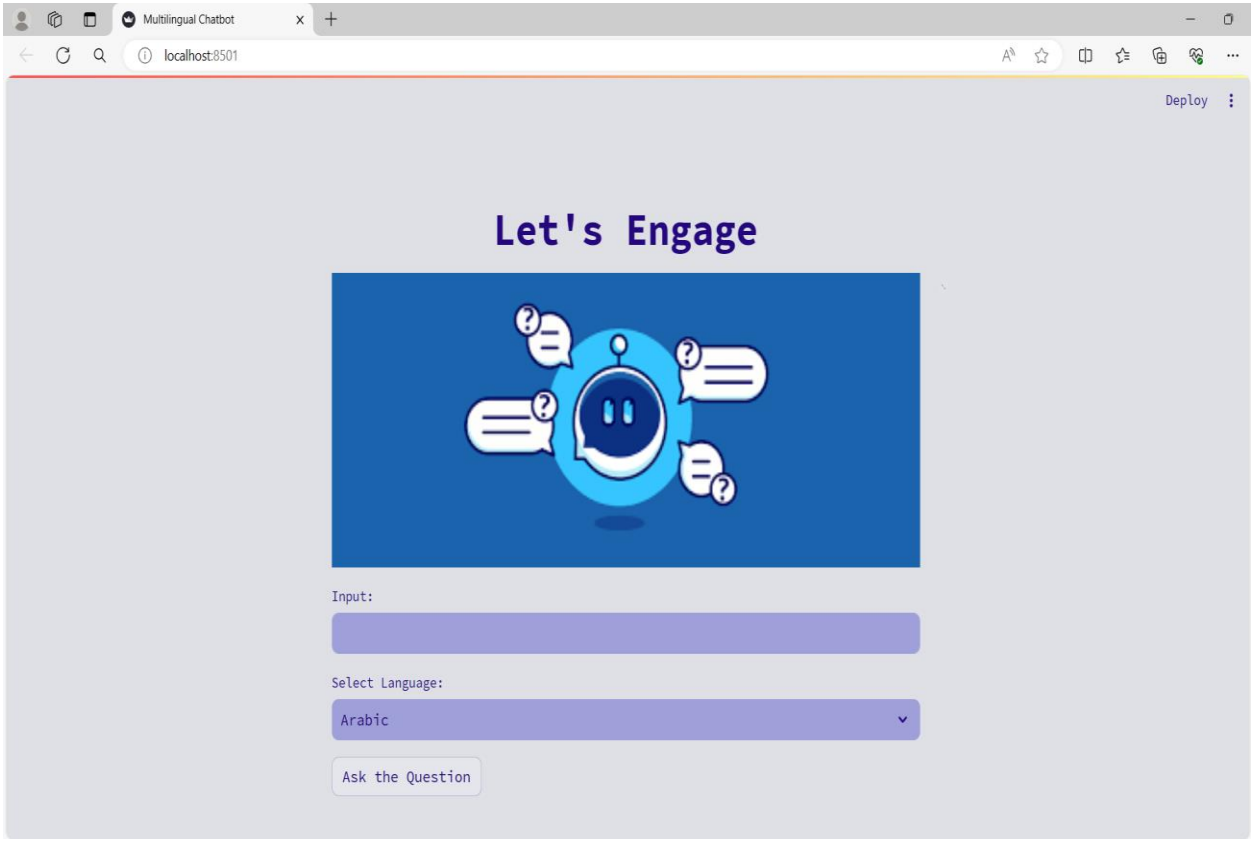
- To host the application, go to the terminal, type - streamlit run app.py
- Here app.py refers to a python script.

```
PS C:\Users\dhurvi patel\Desktop\SCIT academic\Smart_Bridge_Internship\SB_Company work\May_Month\Multilingual Chatbot LLM Gemini> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.29.80:8501
```

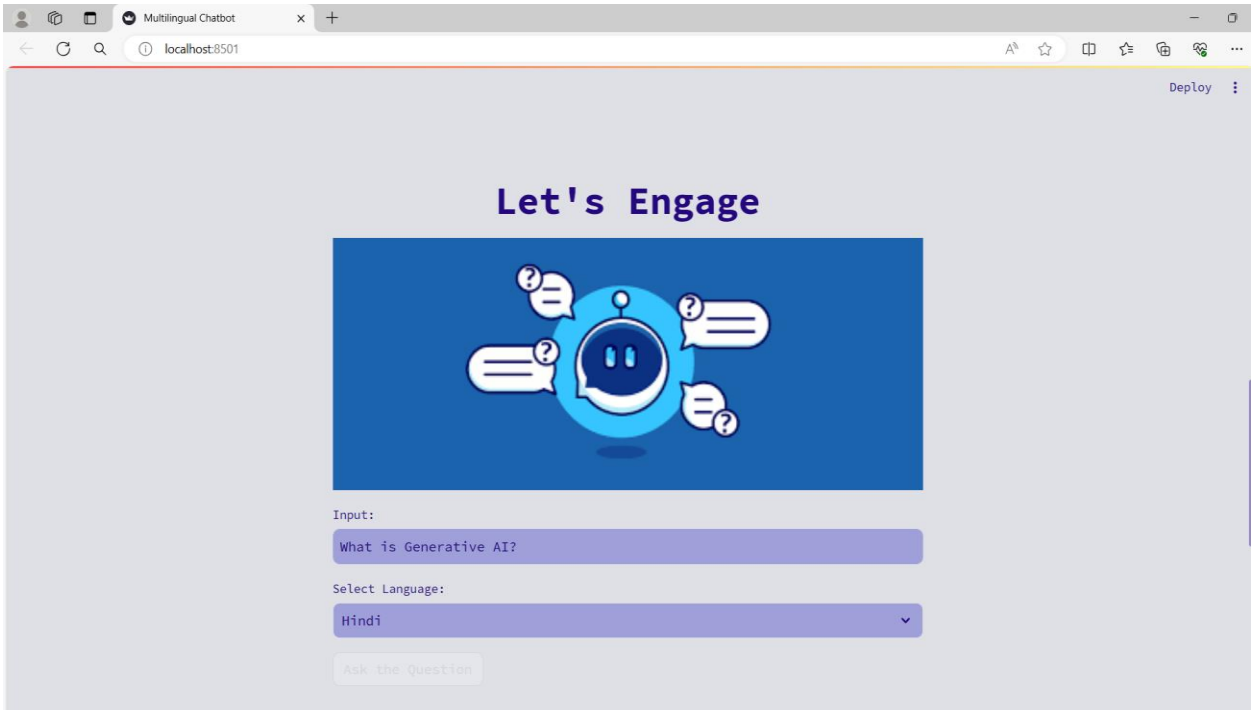
Run the command to get the below results



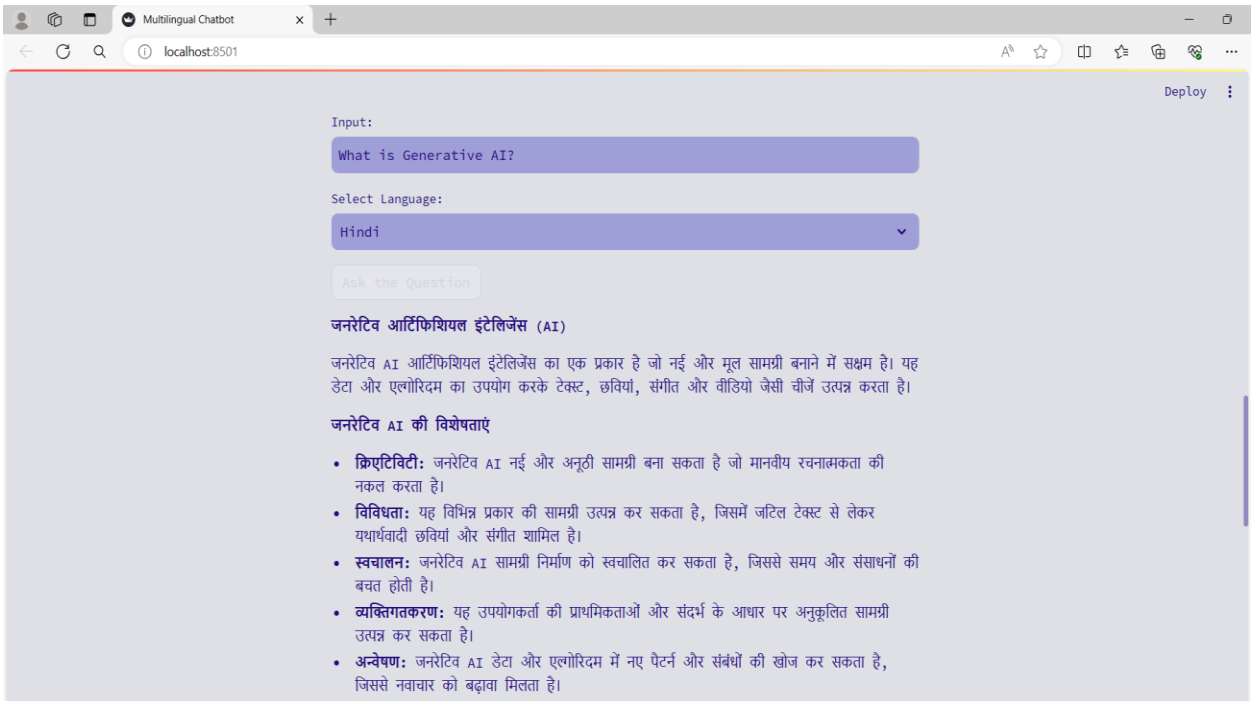

Multilingual Chatbot
localhost:8501
Deploy
What we Offer?
Multilingual Customer Support: VerbaVerse offers multilingual assistance for troubleshooting, product inquiries, and general support, ensuring a smooth customer experience across languages.
Language Learning and Practice: Users can practice and improve language skills by engaging in conversations with the chatbot, receiving real-time feedback, and accessing learning resources, fostering proficiency and confidence.
Multilingual Content Translation and Summarization: VerbaVerse provides accurate translation and concise summarization of documents, articles, and textual data in multiple languages, facilitating efficient information processing and analysis.


Multilingual Chatbot
localhost:8501
Deploy
Let's Engage
Input:
Select Language:
Arabic
Ask the Question

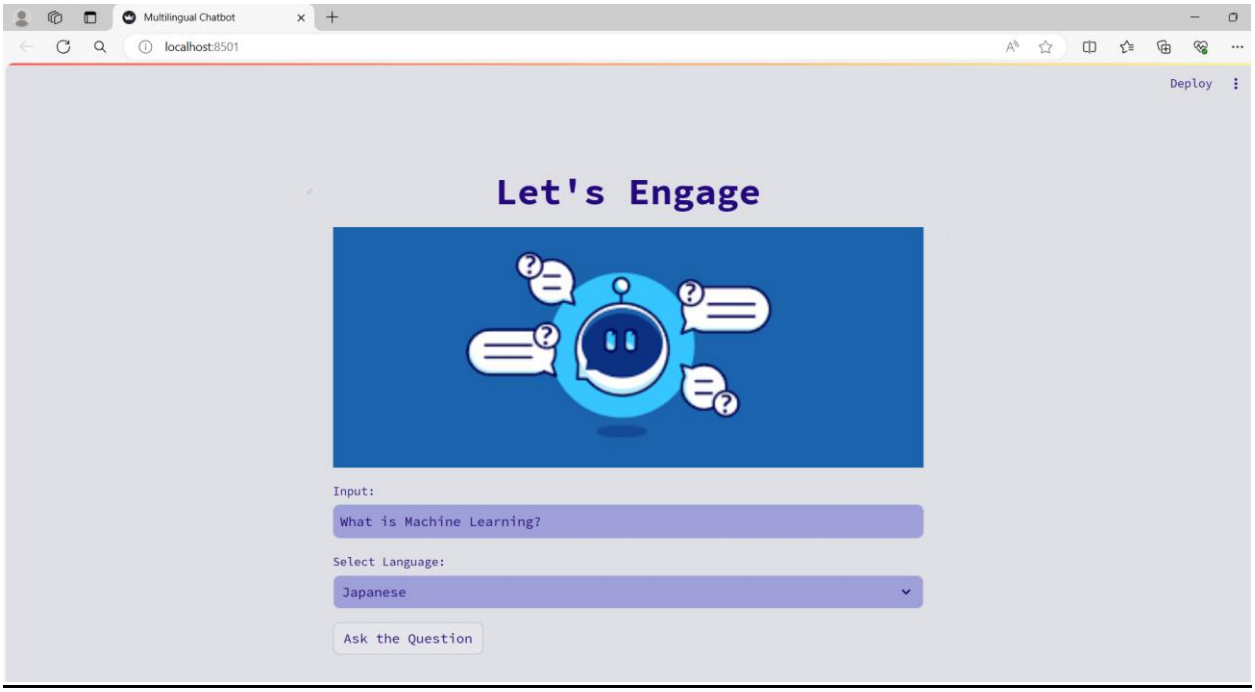
INPUT 1



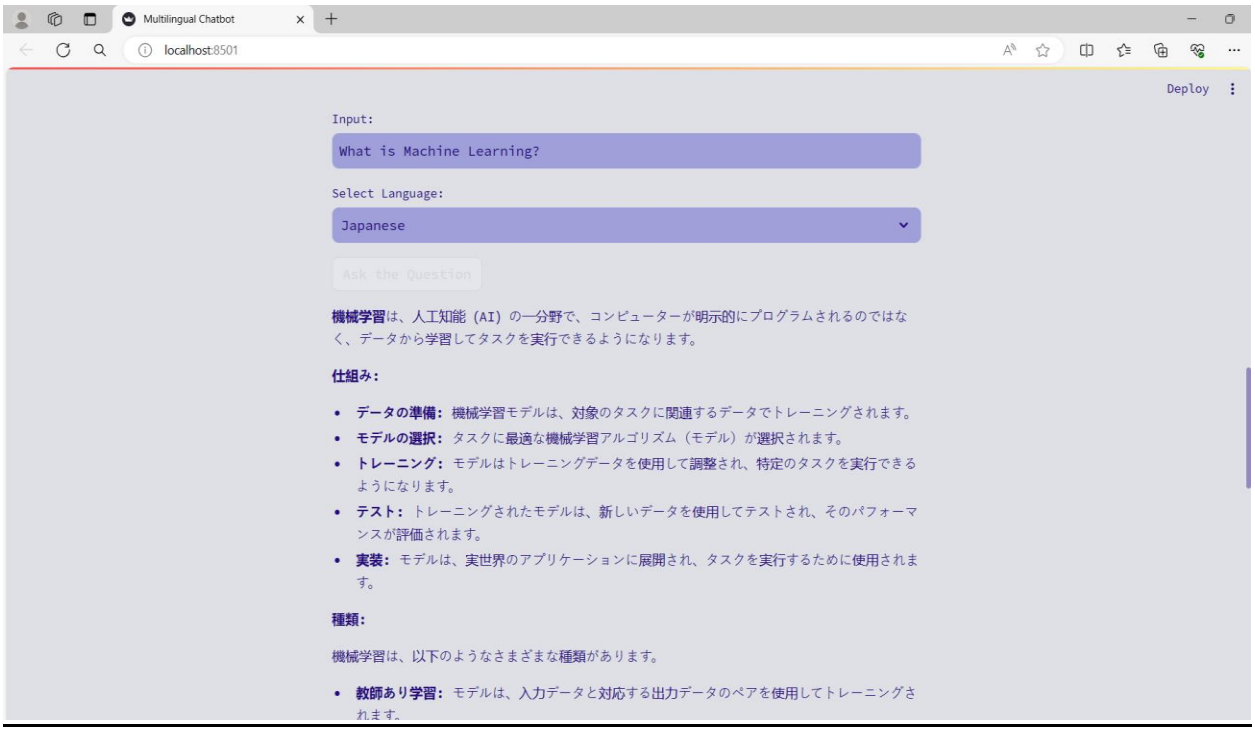
OUTPUT 1



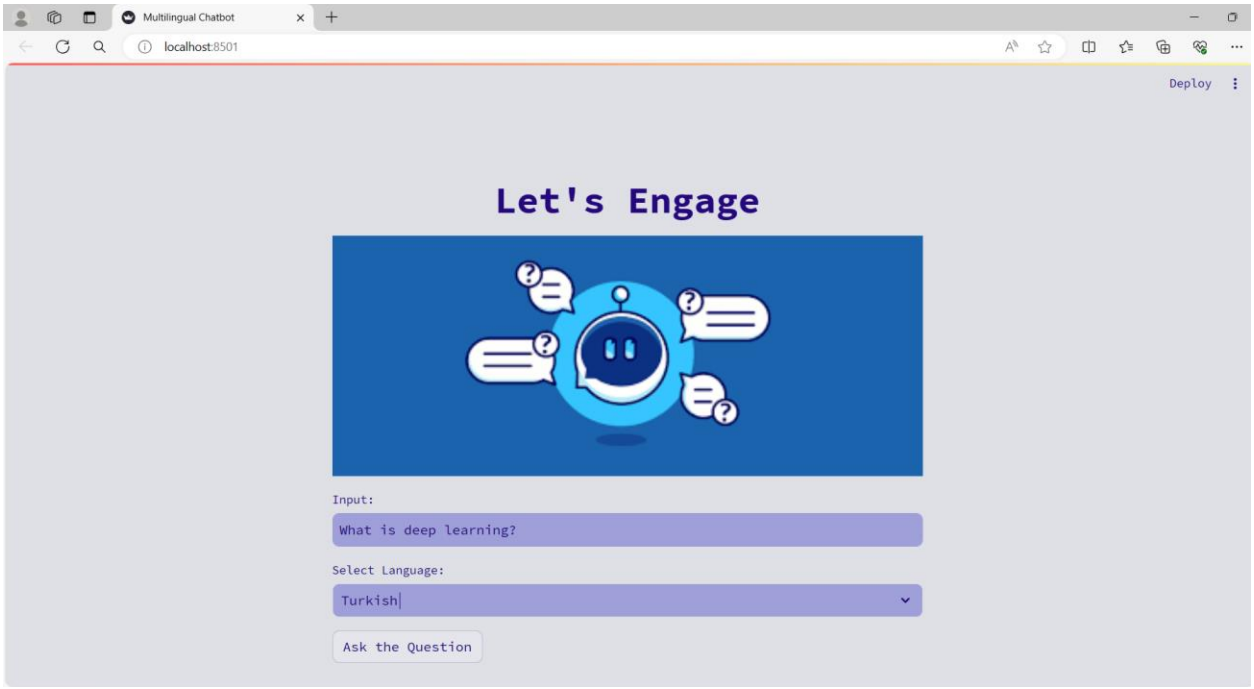
INPUT 2



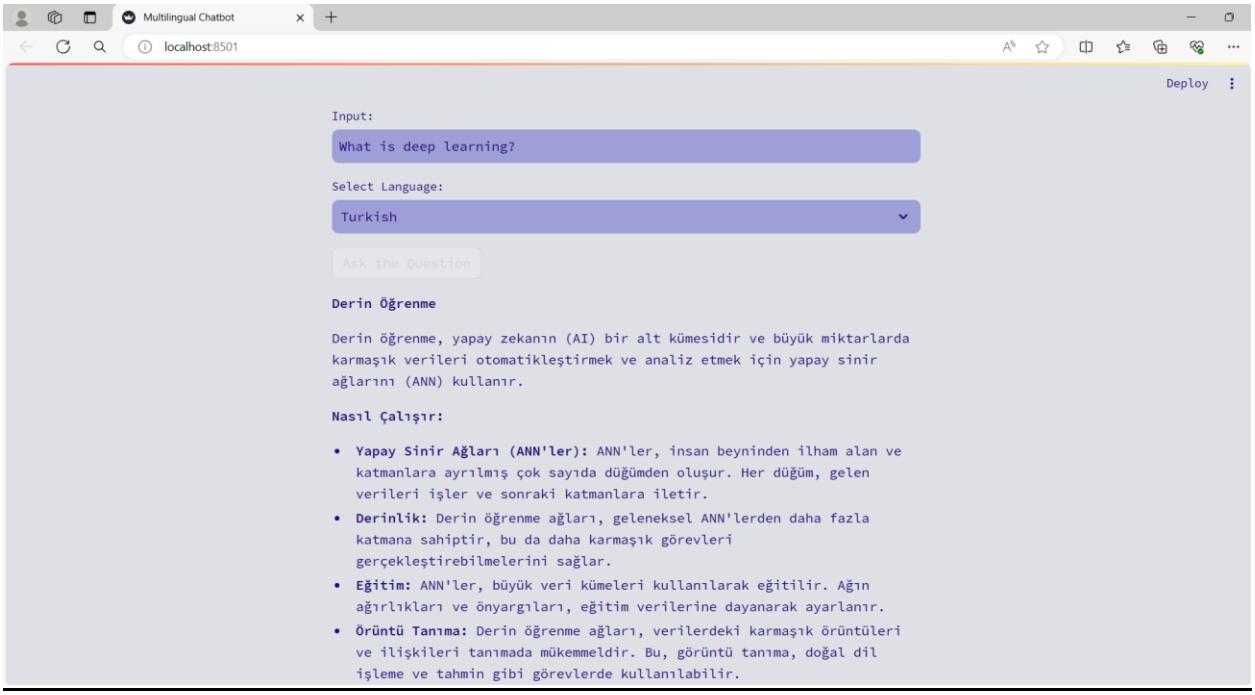
OUTPUT 2

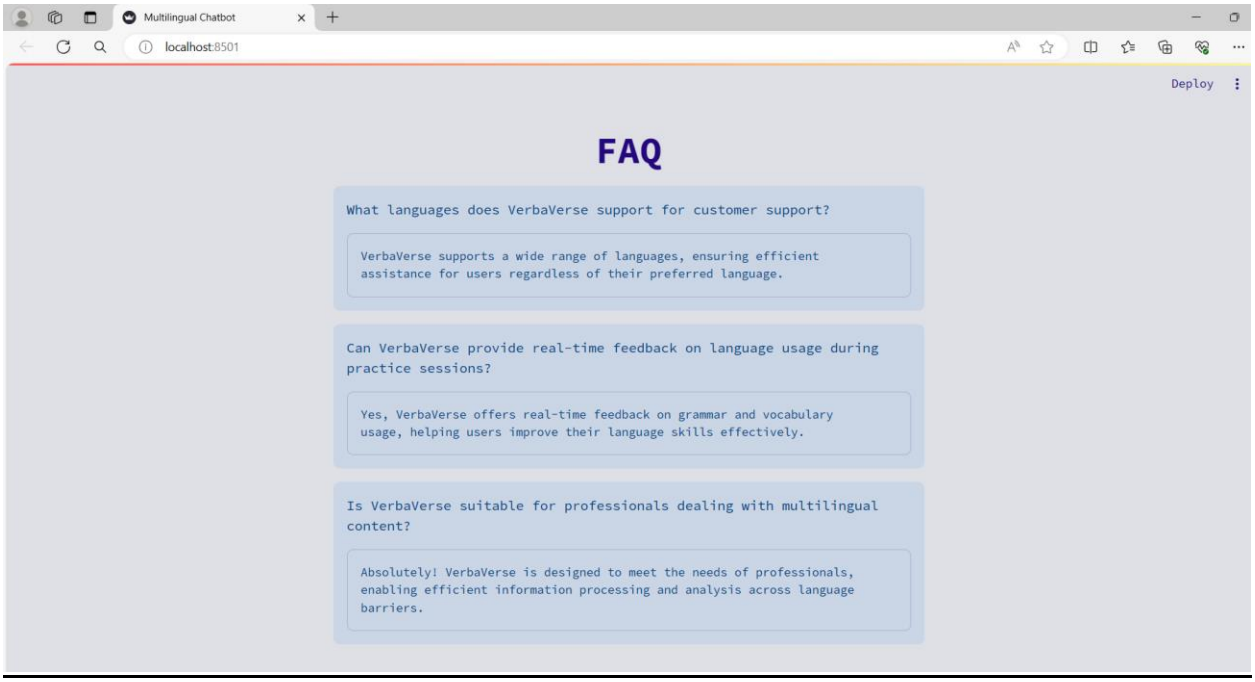


INPUT 3



OUTPUT 3




Multilingual Chatbot
localhost:8501
Deploy
FAQ
What languages does VerbaVerse support for customer support?
VerbaVerse supports a wide range of languages, ensuring efficient assistance for users regardless of their preferred language.
Can VerbaVerse provide real-time feedback on language usage during practice sessions?
Yes, VerbaVerse offers real-time feedback on grammar and vocabulary usage, helping users improve their language skills effectively.
Is VerbaVerse suitable for professionals dealing with multilingual content?
Absolutely! VerbaVerse is designed to meet the needs of professionals, enabling efficient information processing and analysis across language barriers.