Integration Documentation

INTRO

This documentation outlines the integration of a generative AI model into a Streamlit application designed for an e-commerce negotiation chatbot. The chatbot allows users to negotiate product prices or ask other questions, providing responses based on user input.

Environment Setup

- **Python Libraries**:
    - textwrap: Used for formatting text.
    - dotenv: For loading environment variables.
    - markdown_it: Converts text into markdown format.
    - streamlit: For building the web interface.
    - google.generativeai: For interfacing with Google's Generative AI model.

Loading Environment Variables

The dotenv library is used to load API keys and other environment variables from a .env file:

```
import dotenv
from dotenv import load_dotenv
load_dotenv()
```

Configuring the AI Model

The google.generativeai library is configured with an API key loaded from the environment variables:

```
import google.generativeai as genai
import os
genai.configure(api_key=os.getenv("YOUR_API_KEY_HERE"))
```

AI Model Integration

The get_gemini_response function uses the generative AI model to generate responses based on user input. The gemini-pro model is specified, and the generate_content method is used to get the response:

```
def get_gemini_response(question):
  model = genai.GenerativeModel('gemini-pro')
  response = model.generate_content(question)
  return response.text
```

Price Negotiation Logic

The negotiate_price function handles price negotiation based on user input. It compares the user's offer with the current price and returns a response:

```
def negotiate_price(user_price, current_price):
    if user_price >= current_price:
        return f"Great! Your offer of ${user_price} has been accepted."
    elif user_price >= MIN_PRICE:
        counter_offer = (current_price + user_price) / 2
        return f"How about we meet halfway at ${counter_offer:.2f}?"
    else:
        return f"Sorry, the lowest I can go is ${MIN_PRICE}. Would you like to accept that?"
```

Streamlit Application

The Streamlit app is set up to provide a user interface for the chatbot. It includes:

- **Page Configuration**:

  st.set_page_config(page_title="E-commerce Negotiation Chatbot")

- **User Input**: A text input field allows users to enter their offer or ask questions:

  input_text = st.text_input("Enter your offer or start a conversation:", key="input")

- **Submit Button**: A button is provided to submit the input:

  submit = st.button("Send")

- **Response Handling**: When the button is clicked, the application processes the input. If the input is a valid float (interpreted as a price), it uses the negotiate_price function. Otherwise, it uses the get_gemini_response function to generate a response:

```
if submit:
    try:
        user_price = float(input_text)
        response = negotiate_price(user_price, START_PRICE)
    except ValueError:
        response = get_gemini_response(input_text)


    st.subheader("The Response is")
    st.write(response)
```