Streamlit Chatbot using Gemini Pro

Setup:

- Install required packages such as Streamlit. Google Generative AI and python-dotenv.
- Put it in a file named as requirements
- Create a .env file with the Gemini API key in it.

Since I used Streamlit, the design aspect of this task was done using Streamlit's framework which is obtained from their official documentation.

Code Explanation:

```
import os
import streamlit as st
from dotenv import load_dotenv
import google.generativeai as gen_ai
```

```
load_dotenv()
st.set_page_config(
page_title="Chat with Gemini-Pro!"
page_icon=":brain:",
layout="centered",
)
```

Configures the Streamlit page settings, including the page title, icon, and layout.
The layout="centered" setting centers the content of the page, which is often preferred for chat applications.

```
GOOGLE_API_KEY = os.getenv("GOOGLE_API_KEY")
gen_ai.configure(api_key=GOOGLE_API_KEY)
model = gen_ai.GenerativeModel('gemini-pro')
```

Sets up the model by loading the API key from the '.env' file and using it to enable Gemini Pro

```
if "chat_session" not in st.session_state:
    st.session_state.chat_session =
model.start_chat(history=[])
```

Initializes the chat session if it is not already present by creating an empty chat.

```
# Display the chatbot's title on the page
st.title("🤖 Gemini Pro - ChatBot")

# Display the chat history
for message in st.session_state.chat_session.history:
    with
st.chat_message(translate_role_for_streamlit(message.role)):
        st.markdown(message.parts[0].text)
```

Displays each message in a chat message container, while making sure that the correct sender role is determined for each message.

```
# Input field for user's message
user_prompt = st.chat_input("Ask Gemini-Pro...")
if user_prompt:
    # Add user's message to chat and display it
    st.chat_message("user").markdown(user_prompt)

    # Send user's message to Gemini-Pro and get the response
    gemini_response =
st.session_state.chat_session.send_message(user_prompt)

    # Display Gemini-Pro's response
    with st.chat_message("assistant"):
        st.markdown(gemini_response.text)
```

This code builds the basic of sending and receiving messages using the chatbot. When a message is entered, it is sent to the Gemini-Pro model for processing using 'send_message'. The response is stored in 'gemini_response' variable. And the assistant is role is given to the responses given back.

```
#Generate chat history
def get_chat_history_as_string():
    chat_history_str = ""
    for message in st.session_state.chat_session.history:
        role = translate_role_for_streamlit(message.role)
        chat_history_str += f"{role.capitalize()}:
{message.parts[0].text}\n\n"
    return chat_history_str
```

This function creates a chat history which can be downloaded later. It iterates through the chat session's history, translates the roles, and formats the messages into a string.

```
#Downloads chat as a text
chat_history_str = get_chat_history_as_string()
st.download_button(
    label="Download Conversation",
    data=chat_history_str,
    file_name="chat_history.txt",
    mime="text/plain"
```

This code adds a download button that allows users to download the chat history as a text file.