Chatbot using API with GUI

Overview

Here I created a chatbot using the API of gemini and set it up to create an output in the front end using the streamlit.

Description

In this project we developed a chatbot using the API key of Gemini and then we connected it to streamlit for the GUI. The chatbot will allow to ask questions and give the responses accordingly .The front end is also user-friendly

First we are importing all the necessary libraries:-

```
import streamlit as st #importing necessry libraries
import google.generativeai as genai
```

Then we are configuring the API key

```
# configuring the API
genai.configure(api_key="AIzaSyAg1aAE0qGMXLnYN7F9wiYlRf4GJ_cZYxQ")
genrative_config = {"temperature": 0.9, "top_p": 1, "top_k": 1} #s
```

Then we are connecting the frontend with the output generated from the input

```
# setting session state for question and answer
if "question" not in st.session_state:
    st.session_state.question = ""
if "answer" not in st.session_state:
    st.session_state.answer = ""

# Input space for asking quesiton
st.session_state.question = st.text_input("Ask your question:", value=st.session_state.question)

col1, col2 = st.columns([1, 8.5])

# Enter button to generate the response
with col1:
    if st.button("Enter"):
            # Generate the answer
            response = model.generate_content(st.session_state.question)
            st.session_state.answer = response.text
            st.markdown(f"**Answer:** {st.session_state.answer}")

# Clear button to reset the input
with col2:
    if st.button("Clear"):
        st.session_state.question = ""
        st.session_state.answer = ""
        st.rerun()
```

These all things are happening in the main page.

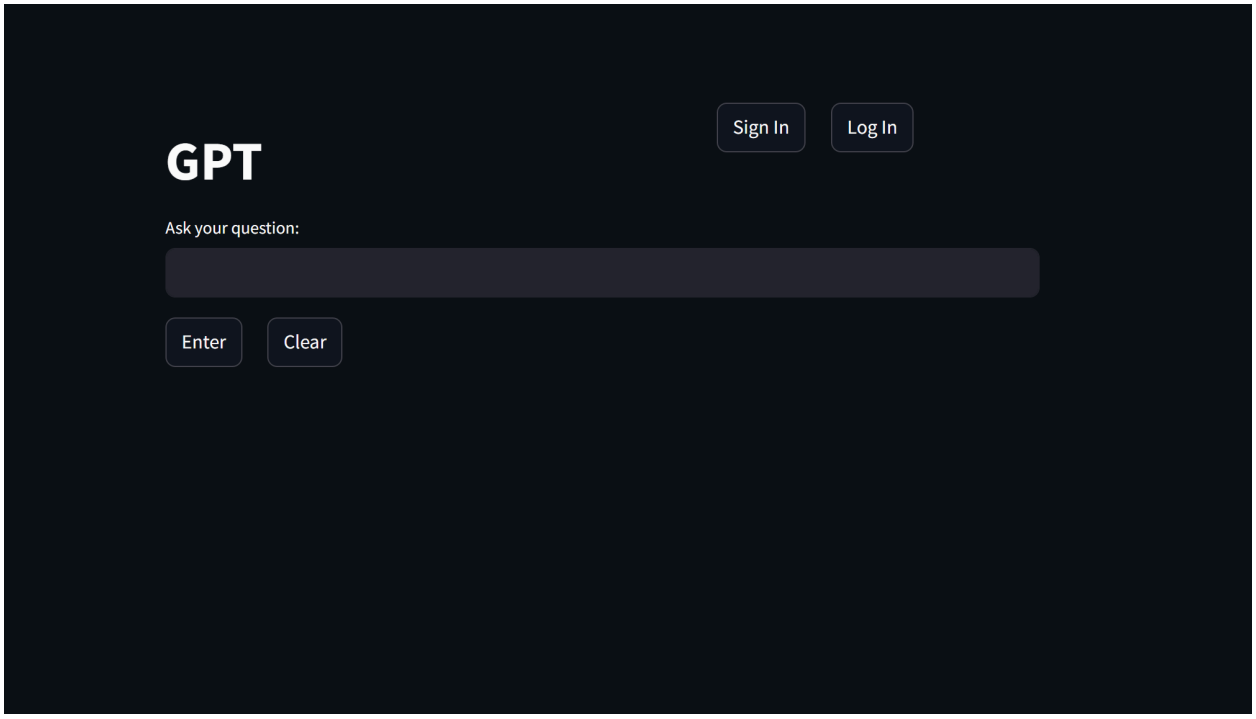
```
def main_page(): #In the title section we are also adding sign in and login buttons
    col1, col2, col3, col4 = st.columns([5, 1, 1, 1])  # Adjust the ratio for proper spacing
    with col1:
        st.title("GPT")

    # Sign In button in the second column
    with col2:
        if st.button("Sign In"):
            st.session_state.page = "sign_in" #This will move to different page called signin
    with col3:
        if st.button("Log In"):
            st.session_state.page = "log_in" #This will also move to a different page called login

    # setting session state for question and answer
    if "question" not in st.session_state:
        st.session_state.question = ""
    if "answer" not in st.session_state:
        st.session_state.answer = ""
```

There you can find session pages moving to sign-in and login so there I created another functions to move to different pages.

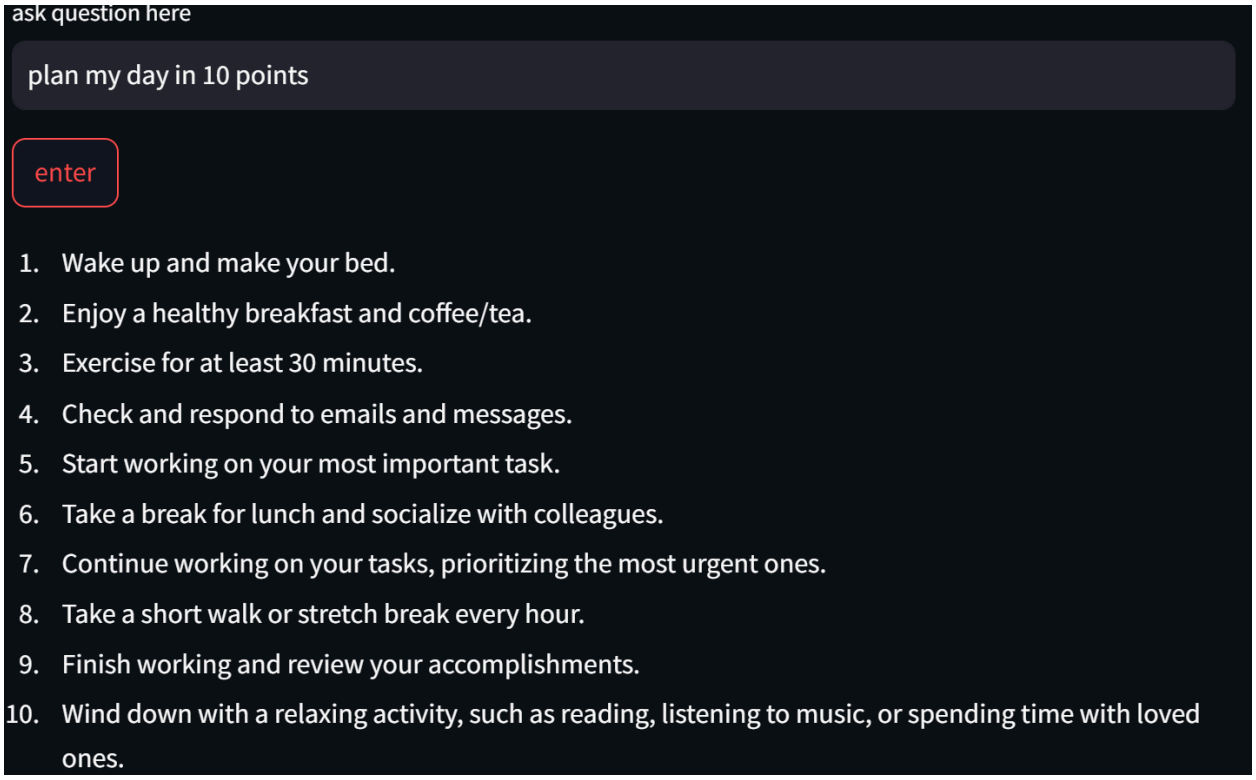
```
            st.rerun()
def sign_in_page():
    st.title("Sign In")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")

    if st.button("Submit"):
        # Signin logic
        if len(username)<8:
            st.text("incorrect username")
        else:
            st.success("Sign In successful!")  #Gives a message successful signin
            st.session_state.page = "main"  # Return to the main page after sign-in
def log_in_page():
    st.title("Log In")
    username = st.text_input("GPT username")
    password = st.text_input("password",type= 'password')
    if st.button("Submit"):
        #login logic
        if len(username)<8:
            st.text("incorrect username")
        else:
            st.success("Logged IN") #Gives a message successful login
            st.session_state.page = 'main' # Return to the main page after sign-in
```

These are the code snippets for the respective tasks in the project. The multi page GUI can further be developed by adding the database and other things and also improving the user interface and also can be used to maintain the history of the chat .

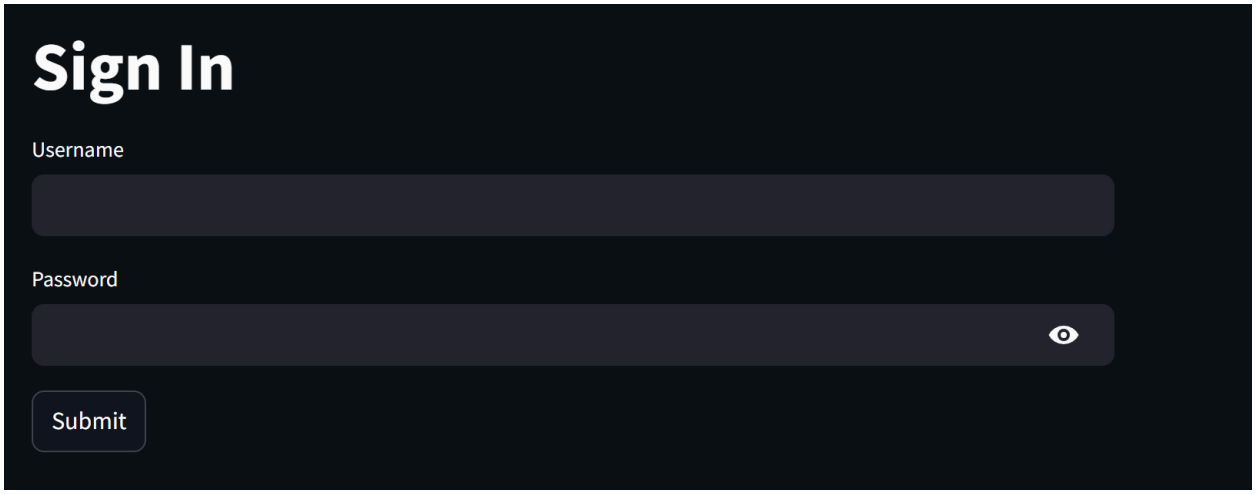
Output



Output when the question is asked:-



Sign-in page:-(when clicked sign in)



Login page:-(when clicked login)

Github link:- https://github.com/viswa0028