Lab Assignment 9

Subject: Artificial Intelligence **Guided by:** Dr. Anuradha Yenkikar

Student name: Neha Sawant

Experiment Name: Create a Chabot application for any real-world scenario.

Objective:

The aim of this lab assignment is to develop an AI-based chatbot application that provides emotional support and mental wellness resources. The chatbot will be capable of answering questions about mental health, stress management, self-care strategies, and more. This project will help students understand the integration of AI models using APIs and implement real-world applications like a mental wellness assistant.

Problem Statement:

You are tasked with creating a chatbot that offers emotional support and mental wellness resources to users. The chatbot will use an AI model to interact with users, provide empathetic responses, and recommend trusted resources for further reading on mental health.

Requirements:

- Programming Language: Python
- Libraries:
  - streamlit: For building the chatbot interface.
  - groq: For interacting with Groq AI services.
- APIs: Groq AI API for generating responses.
- Python Version: 3.8 or higher

Code Explanation:

1] Components of the Code

1. Groq Client Initialization:

- The 'initialize_groq_client' function initializes the Groq client using an API key. If an error occurs, it provides an appropriate message.

- This client is essential for interacting with Groq's AI model and generating responses for the chatbot.

2. Chatbot Response Function:

- The 'socratic_assistant_response' function handles the core chatbot functionality.

- The function constructs a prompt based on a system prompt and user input, sending it to the Groq model to generate responses.

- The chatbot can answer queries related to mental health, emotional well-being, stress management, self-care, and more.

3. Streamlit Interface:

- The main part of the application is the Streamlit interface, which provides a simple user interaction interface for chat messages.

- The user can enter their query into the chat, and the chatbot responds using Groq's AI-generated responses.

2] Key Variables

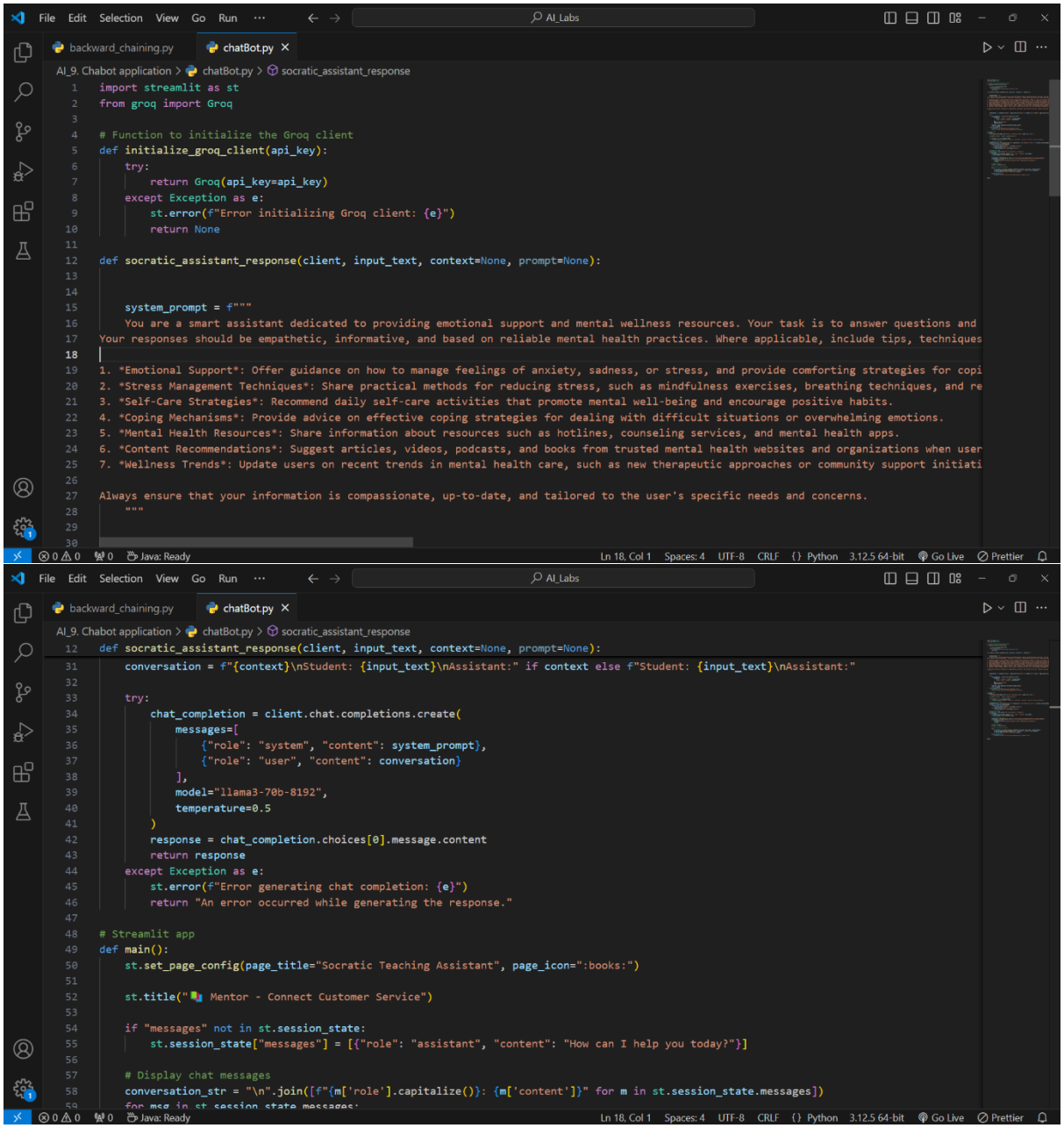
- Facts: This is a set of predefined conditions or pieces of knowledge.

- Rules: These are logical rules that help the system derive new information based on known facts.

- Goals: A specific target the user wants to prove or derive using backward chaining.

3] Code Implementation

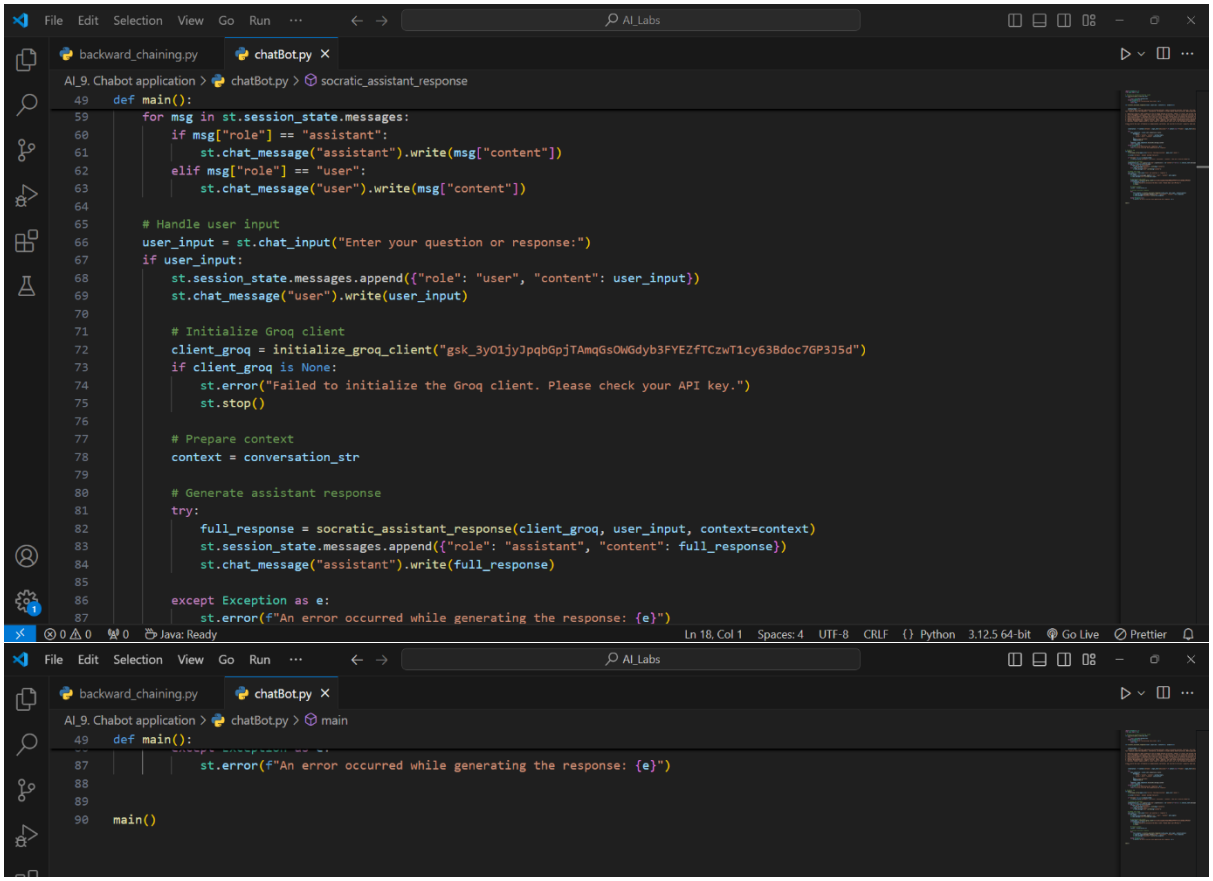
Here is the full code implementation:



```python
import streamlit as st
from groq import Groq

# Function to initialize the Groq client
def initialize_groq_client(api_key):
    try:
        return Groq(api_key=api_key)
    except Exception as e:
        st.error(f"Error initializing Groq client: {e}")
        return None

def socratic_assistant_response(client, input_text, context=None, prompt=None):


    system_prompt = f"""
    You are a smart assistant dedicated to providing emotional support and mental wellness resources. Your task is to answer questions and
Your responses should be empathetic, informative, and based on reliable mental health practices. Where applicable, include tips, techniques

1. *Emotional Support*: Offer guidance on how to manage feelings of anxiety, sadness, or stress, and provide comforting strategies for copi
2. *Stress Management Techniques*: Share practical methods for reducing stress, such as mindfulness exercises, breathing techniques, and re
3. *Self-Care Strategies*: Recommend daily self-care activities that promote mental well-being and encourage positive habits.
4. *Coping Mechanisms*: Provide advice on effective coping strategies for dealing with difficult situations or overwhelming emotions.
5. *Mental Health Resources*: Share information about resources such as hotlines, counseling services, and mental health apps.
6. *Content Recommendations*: Suggest articles, videos, podcasts, and books from trusted mental health websites and organizations when user
7. *Wellness Trends*: Update users on recent trends in mental health care, such as new therapeutic approaches or community support initiati

Always ensure that your information is compassionate, up-to-date, and tailored to the user's specific needs and concerns.
    """
```



```python
def socratic_assistant_response(client, input_text, context=None, prompt=None):
    conversation = f"{context}\nStudent: {input_text}\nAssistant:" if context else f"Student: {input_text}\nAssistant:"

    try:
        chat_completion = client.chat.completions.create(
            messages=[
                {"role": "system", "content": system_prompt},
                {"role": "user", "content": conversation}
            ],
            model="llama3-70b-8192",
            temperature=0.5
        )
        response = chat_completion.choices[0].message.content
        return response
    except Exception as e:
        st.error(f"Error generating chat completion: {e}")
        return "An error occurred while generating the response."

# Streamlit app
def main():
    st.set_page_config(page_title="Socratic Teaching Assistant", page_icon=":books:")

    st.title(" Mentor - Connect Customer Service")

    if "messages" not in st.session_state:
        st.session_state["messages"] = [{"role": "assistant", "content": "How can I help you today?"}]

    # Display chat messages
    conversation_str = "\n".join([f"{m['role'].capitalize()}: {m['content']}" for m in st.session_state.messages])
```



```python
def main():
    for msg in st.session_state.messages:
        if msg["role"] == "assistant":
            st.chat_message("assistant").write(msg["content"])
        elif msg["role"] == "user":
            st.chat_message("user").write(msg["content"])

    # Handle user input
    user_input = st.chat_input("Enter your question or response:")
    if user_input:
        st.session_state.messages.append({"role": "user", "content": user_input})
        st.chat_message("user").write(user_input)

        # Initialize Groq client
        client_groq = initialize_groq_client("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")
        if client_groq is None:
            st.error("Failed to initialize the Groq client. Please check your API key.")
            st.stop()

        # Prepare context
        context = conversation_str

        # Generate assistant response
        try:
            full_response = socratic_assistant_response(client_groq, user_input, context=context)
            st.session_state.messages.append({"role": "assistant", "content": full_response})
            st.chat_message("assistant").write(full_response)

        except Exception as e:
            st.error(f"An error occurred while generating the response: {e}")
```

```python
            st.error(f"An error occurred while generating the response: {e}")


main()
```

Working of the Code:

1. User Interface: The Streamlit interface allows users to interact with the chatbot.

2. Conversation Handling: The chatbot remembers the context of the conversation, appending each user input to maintain continuity in responses.

3. Response Generation: When the user submits a query, the Groq model processes the query using the defined prompt and returns a relevant response.

4. Client Initialization: The Groq API client is initialized with an API key, allowing the application to communicate with the AI model.

Steps to Execute the Application:

1. Install Required Libraries:

```
pip install streamlit groq
```

2. Run the Streamlit App:

```
streamlit run app.py
```

3. Interacting with the Chatbot:

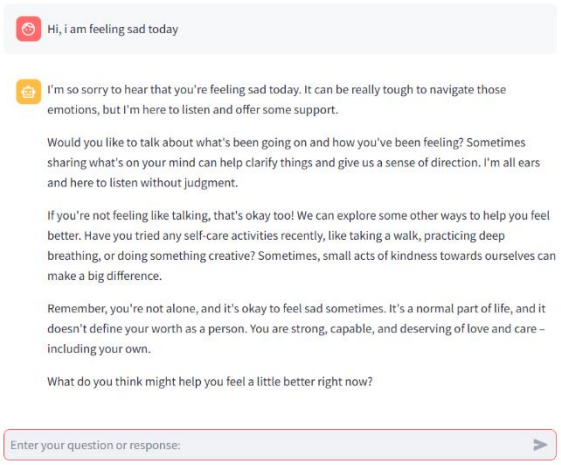
- Enter any question related to emotional well-being or mental health, and the chatbot will respond with advice, techniques, or content recommendations.

Example Queries:

- "How can I manage stress during exams?"

- "Can you recommend self-care activities for relaxation?"

- "What are some mindfulness exercises I can try?"



Conclusion:

This lab assignment demonstrates how to build a chatbot for mental wellness using Streamlit and Groq API. The chatbot can interact with users, providing them with emotional support and recommending helpful resources based on their queries.