MINI PROJECT

SENTIMENT AWARE CHATBOT USING NATURAL LANGUAGE PROCESSSING

AIM:

The aim of this project is to develop an emotionally intelligent chatbot capable of understanding the sentiment behind user input and responding accordingly using Natural Language Processing (NLP) techniques. The project seeks to create an interactive web-based chatbot that enhances human–computer interaction by providing context-aware and empathetic responses based on user emotions.

ABSTRACT:

In recent years, chatbots have become an essential part of modern digital communication, serving in customer support, healthcare, and education. However, most traditional chatbots lack emotional understanding, leading to monotonous or irrelevant responses. This project addresses that limitation by developing a Sentiment-Aware Chatbot capable of analyzing user emotions and adapting its replies accordingly.

The system leverages Natural Language Processing (NLP) to process textual input and determine the emotional tone using VADER (Valence Aware Dictionary and sEntiment Reasoner), a pre-trained sentiment analysis model available in the NLTK library. Based on the computed sentiment polarity (positive, negative, or neutral), the chatbot generates empathetic and context-appropriate responses to simulate natural human conversation.

The chatbot is implemented in Python and deployed using Streamlit, allowing users to interact with it through an intuitive and user-friendly web interface. The application dynamically captures user input, performs real-time sentiment analysis, and displays responses in an engaging chat format. This project demonstrates how combining NLP with simple deployment tools can create intelligent systems that improve user experience by understanding and responding to emotions, rather than just words.

SOURCE CODE

COLLAB CODE:

✅ Step 1: Install dependencies

```
!pip install nltk
```

✅ Step 2: Import libraries

```
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
# Download sentiment lexicon (only once)
nltk.download('vader_lexicon')
# Initialize analyzer
sia = SentimentIntensityAnalyzer()
```

```
# ✅ Step 3: Define chatbot function
def chatbot_response(user_input):
    sentiment_score = sia.polarity_scores(user_input)
    compound = sentiment_score['compound']

 if compound >= 0.5:
        return "😊 I'm glad to hear that! Sounds like you're feeling positive!"
    elif compound <= -0.5:
        return "😔 I'm sorry you feel that way. Want to talk about what’s bothering you?"
    else:
        return "😐 Hmm, I see. Tell me more!"
✅ Step 4: Test chatbot manually
while True:
    user_input = input("You: ")
    if user_input.lower() in ["exit", "quit", "bye"]:
        print("Bot: 👋 Goodbye! Take care!")
        break
    print("Bot:", chatbot_response(user_input))
```

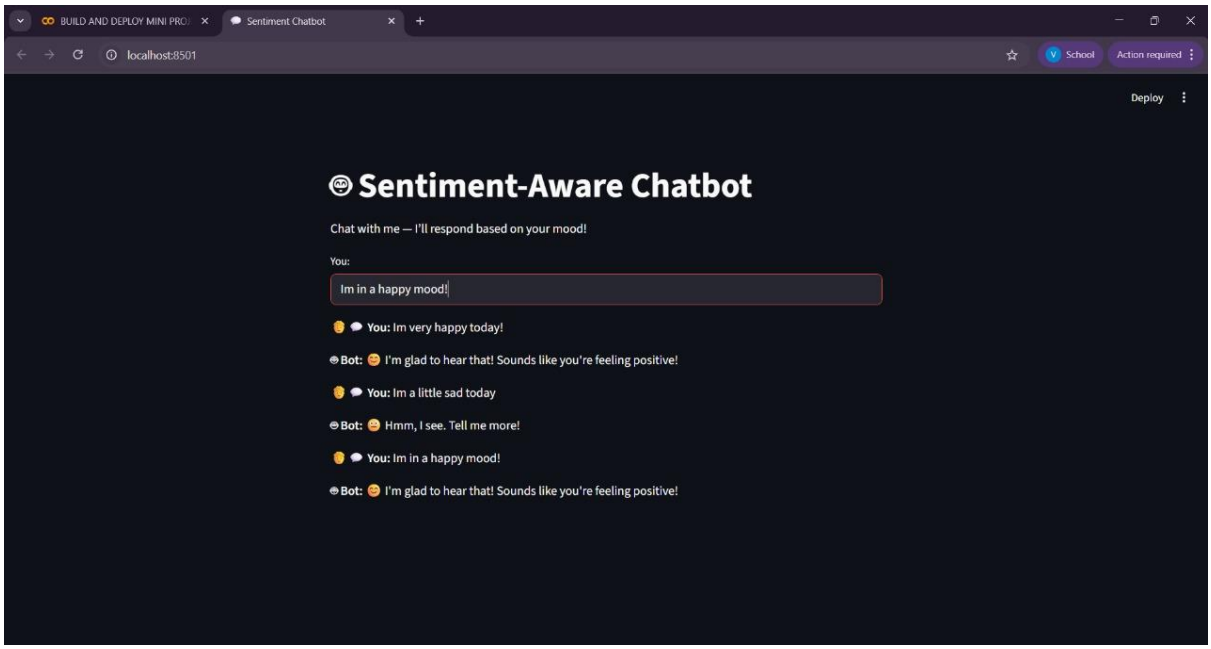
VS CODE:

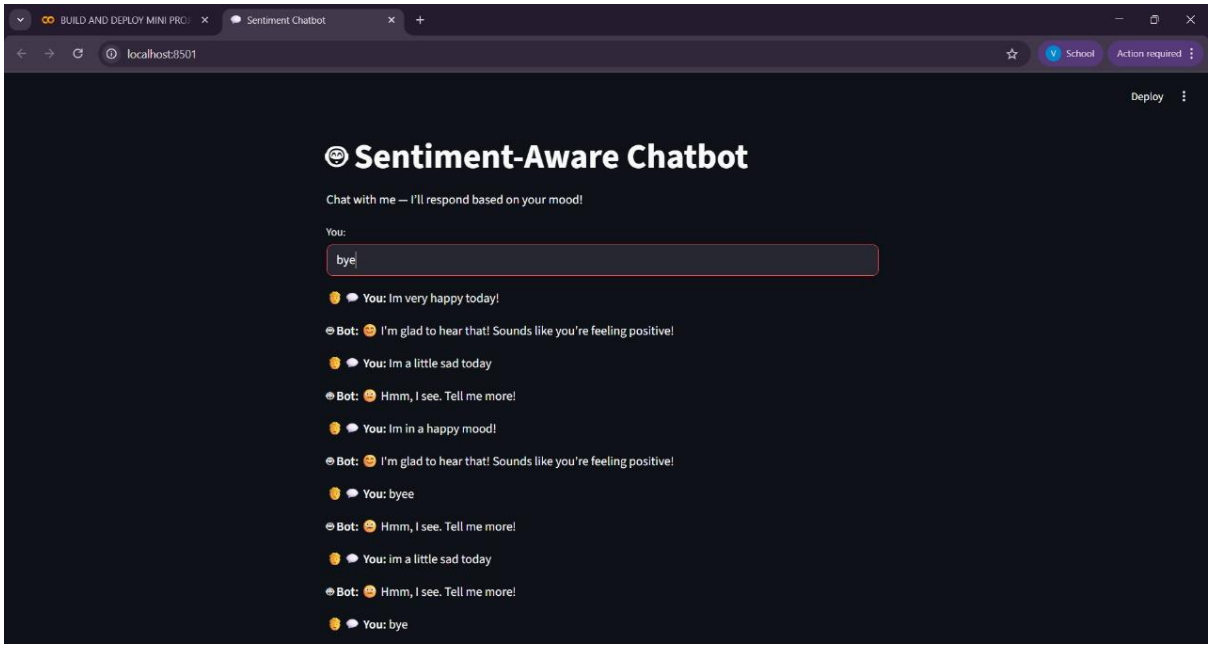
```
import streamlit as st
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer
# Download sentiment lexicon (only first time)
nltk.download('vader_lexicon')
# Initialize analyzer
sia = SentimentIntensityAnalyzer()
# Function: Generate chatbot response
def chatbot_response(user_input):
    sentiment_score = sia.polarity_scores(user_input)
    compound = sentiment_score['compound']

    if compound >= 0.5:
```

```
        return "😊 I'm glad to hear that! Sounds like you're feeling positive!"
    elif compound <= -0.5:
        return "😔 I'm sorry you feel that way. Want to talk about what's bothering you?"
    else:
        return "😐 Hmm, I see. Tell me more!"
# Streamlit App UI
st.set_page_config(page_title="Sentiment Chatbot", page_icon="💬")
st.title("🤖 Sentiment-Aware Chatbot")
st.write("Chat with me — I'll respond based on your mood!")
# Maintain chat history
if "history" not in st.session_state:
    st.session_state.history = []
# Input box
user_input = st.text_input("You:", "")
# Handle input
if user_input:
    bot_reply = chatbot_response(user_input)
    st.session_state.history.append(("You", user_input))
    st.session_state.history.append(("Bot", bot_reply))
# Display chat history
for speaker, text in st.session_state.history:
    if speaker == "You":
        st.markdown(f"**🧑💬 You:** {text}")
    else:
        st.markdown(f"**🤖 Bot:** {text}")
```

OUTPUT:





CONCLUTION:

The Sentiment-Aware Chatbot successfully analyzes the emotional sentiment of user inputs and generates responses that align with the detected mood. Positive statements trigger cheerful replies, negative inputs prompt empathetic responses, and neutral messages receive balanced replies. The chatbot performs effectively in identifying sentiment across a variety of user expressions, and the Streamlit interface ensures seamless, interactive communication through a web browser.

The final system validates that integrating sentiment analysis with chatbot design significantly enhances the quality of machine–human interactions, making digital conversations more natural, personalized, and emotionally engaging.