CHATBOT COMPLETE WEB-APP

Streamlit code and develop a complete AI-driven chatbot application for mental well-being support for students, we can add several new functionalities. Below is the enhanced version of the code with the following additional features:

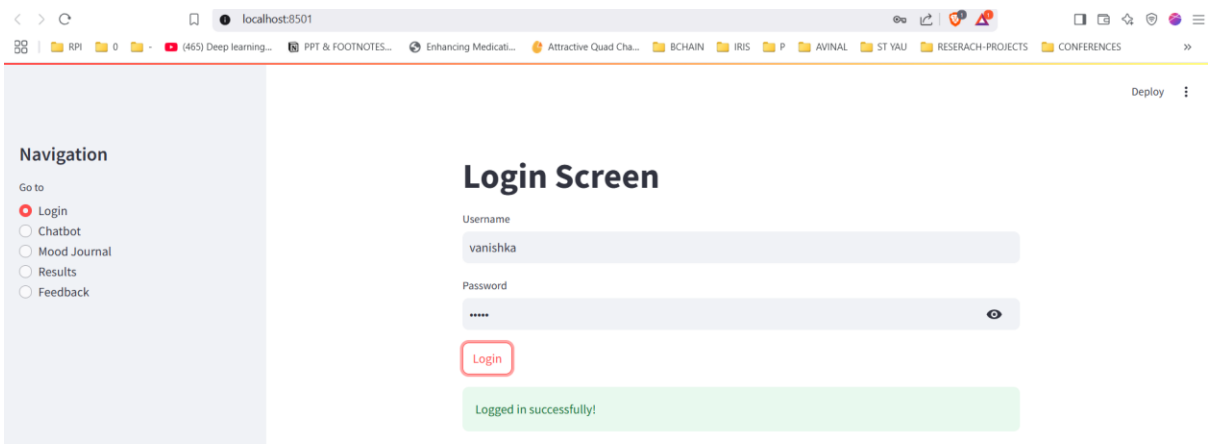
1. **Login Screen**: A simple login screen to authenticate users.
2. **Results Screen**: A screen to display the user's mood analysis results.
3. **Mood Journal**: A feature to allow users to journal their moods over time.
4. **Resource Recommendations**: A feature to recommend resources based on the user's mood.
5. **Feedback Mechanism**: A feature to collect user feedback on the chatbot's performance.

Explanation of Screen Features :

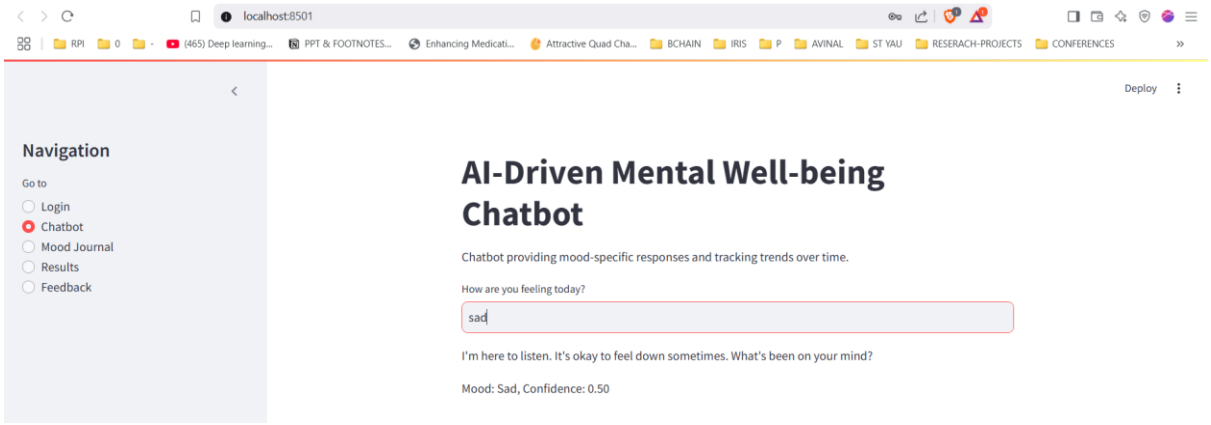
1. **Login Screen**: A simple login screen is added to authenticate users. The username and password are hardcoded for simplicity, but in a real-world application, you would use a more secure authentication method.
2. **Results Screen**: After interacting with the chatbot, users can view their mood analysis results and get personalized resource recommendations.
3. **Mood Journal**: Users can track their mood over time and visualize trends using a bar chart.
4. **Resource Recommendations**: Based on the user's mood, the chatbot recommends relevant resources such as books, articles, and videos.
5. **Feedback Mechanism**: Users can provide feedback on the chatbot's performance, which is saved for future improvements.

This enhanced version provides a more comprehensive and user-friendly experience for students seeking mental well-being support.

1. Login Screen :

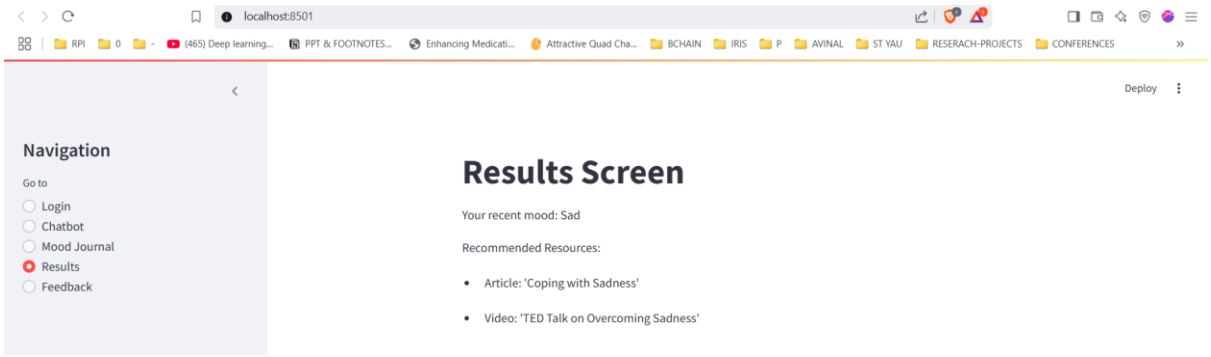


2. Main Chatbot Screen :

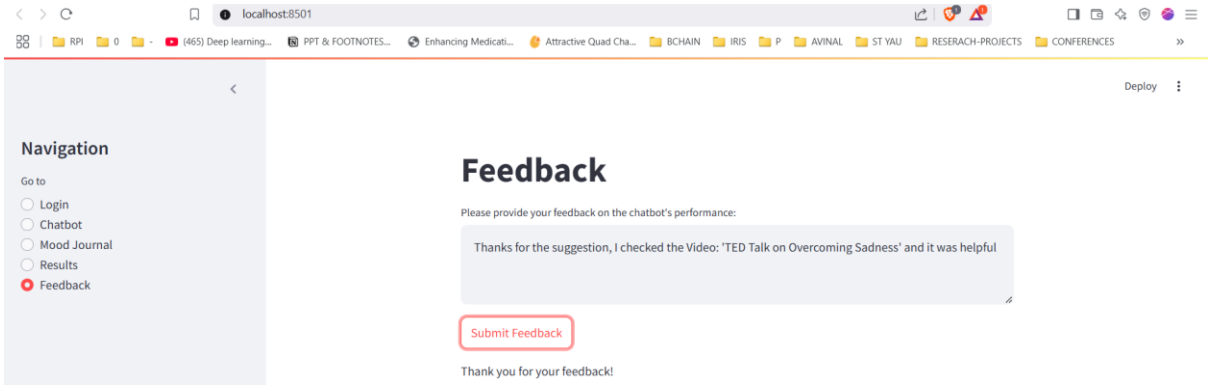


3. Mood Journal Screen :

4. Mood based Results Screen :



5. Feedback Screen :



Complete Code:

```
# Importing Required Libraries
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from textblob import TextBlob

# Function to analyze mood using TextBlob
def analyze_mood(user_input):
    blob = TextBlob(user_input)
    polarity = blob.sentiment.polarity

    if polarity > 0.5:
        mood = "Very Happy"
    elif polarity > 0:
        mood = "Happy"
    elif polarity == 0:
        mood = "Neutral"
    elif polarity < -0.5:
        mood = "Very Sad"
    else:
        mood = "Sad"

    return mood, abs(polarity)

# Define chatbot responses based on mood levels
def chatbot_response(user_input):
    mood, score = analyze_mood(user_input)
    response = ""
```

```
    if mood == "Very Happy":
        response = "You seem to be in great spirits! Keep shining and spread the positivity."
    elif mood == "Happy":
        response = "I'm glad to hear you're feeling good. What made your day brighter?"
    elif mood == "Neutral":
        response = "It seems like you're feeling okay. Let me know if there's something specific you'd like to talk about."
    elif mood == "Sad":
        response = "I'm here to listen. It's okay to feel down sometimes. What's been on your mind?"
    elif mood == "Very Sad":
        response = "I'm sorry to hear that you're feeling this way. Please know you're not alone. How can I support you?"

    return f"{response}\n\nMood: {mood}, Confidence: {score:.2f}"

# Function to save mood data
def save_mood_data(user_input):
    mood, score = analyze_mood(user_input)
    timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    data = {'Timestamp': [timestamp], 'Input': [user_input], 'Mood': [mood], 'Confidence': [score]}
    df = pd.DataFrame(data)
    df.to_csv('mood_tracking_data.csv', mode='a', header=not pd.io.common.file_exists('mood_tracking_data.csv'), index=False)
    return mood

# Function to visualize mood trends
def plot_mood_trends():
    df = pd.read_csv('mood_tracking_data.csv')
    df['Timestamp'] = pd.to_datetime(df['Timestamp'])
    df['Date'] = df['Timestamp'].dt.date
    mood_counts = df.groupby('Date')['Mood'].value_counts().unstack().fillna(0)
    mood_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
```

```
    plt.title('Mood Trends Over Time')
    plt.xlabel('Date')
    plt.ylabel('Mood Count')
    plt.legend(title="Moods")
    st.pyplot(plt)

# Function to recommend resources based on mood
def recommend_resources(mood):
    resources = {
        "Very Happy": ["Book: 'The Happiness Advantage' by Shawn Achor", "Article: 'The Science of Happiness'"],
        "Happy": ["Podcast: 'The Happiness Lab'", "Video: 'TED Talk on Happiness'"],
        "Neutral": ["Article: 'Mindfulness and Well-being'", "Book: 'The Power of Now' by Eckhart Tolle"],
        "Sad": ["Article: 'Coping with Sadness'", "Video: 'TED Talk on Overcoming Sadness'"],
        "Very Sad": ["Hotline: National Suicide Prevention Lifeline", "Article: 'Dealing with Depression'"]
    }
    return resources.get(mood, ["No specific recommendations available."])

# Function to collect user feedback
def collect_feedback():
    feedback = st.text_area("Please provide your feedback on the chatbot's performance:")
    if st.button("Submit Feedback"):
        timestamp = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        data = {'Timestamp': [timestamp], 'Feedback': [feedback]}
        df = pd.DataFrame(data)
        df.to_csv('feedback_data.csv', mode='a', header=not pd.io.common.file_exists('feedback_data.csv'), index=False)
        st.write("Thank you for your feedback!")

# Streamlit app
def main():
```

```
st.sidebar.title("Navigation")
page = st.sidebar.radio("Go to", ["Login", "Chatbot", "Mood Journal", "Results", "Feedback"])

if page == "Login":
    st.title("Login Screen")
    username = st.text_input("Username")
    password = st.text_input("Password", type="password")
    if st.button("Login"):
        if username == "student" and password == "password":
            st.success("Logged in successfully!")
            st.session_state.logged_in = True
        else:
            st.error("Invalid username or password")

elif page == "Chatbot":
    if not st.session_state.get('logged_in', False):
        st.warning("Please login to access the chatbot.")
        return

    st.title("AI-Driven Mental Well-being Chatbot")
    st.write("Chatbot providing mood-specific responses and tracking trends over time.")

    user_input = st.text_input("How are you feeling today?", "")
    if user_input:
        mood_response = chatbot_response(user_input)
        st.write(mood_response)
        mood = save_mood_data(user_input)
        st.session_state.mood = mood

elif page == "Mood Journal":
    if not st.session_state.get('logged_in', False):
```

```
            st.warning("Please login to access the mood journal.")
            return

        st.title("Mood Journal")
        st.write("Track your mood over time.")

        if st.button("Show Mood Trends"):
            plot_mood_trends()

    elif page == "Results":
        if not st.session_state.get('logged_in', False):
            st.warning("Please login to access the results.")
            return

        st.title("Results Screen")
        mood = st.session_state.get('mood', 'Unknown')
        st.write(f"Your recent mood: {mood}")
        st.write("Recommended Resources:")
        resources = recommend_resources(mood)
        for resource in resources:
            st.write(f"- {resource}")

    elif page == "Feedback":
        if not st.session_state.get('logged_in', False):
            st.warning("Please login to provide feedback.")
            return

        st.title("Feedback")
        collect_feedback()

if __name__ == "__main__":
```

main()