TECHNIK NEST

INNOVATIVE MINDS, NESTING SUCCESS

Name: Maaz Sher Muhammad

Intern ID: TN/1N01/003

Email ID : maazshermuhammadofficial@gmail.com

Internship Domain : Python Internee

Task : Project Report

Instructor Name : Hassan

📄 Mood Music Suggester Report

by Maaz Sher Muhammad

📌 Table of Contents

1. Introduction
2. Task Descriptions
3. Screenshots
4. Challenges and Solutions
5. Conclusion

1. Introduction

This project, titled Mood Music Suggester, is a Python-based application built using the Gradio framework. It takes a user's mood as a textual input, identifies the underlying emotion, and recommends a song that matches the mood along with a motivational message.

The purpose of this project is to explore the use of basic Natural Language Processing (NLP) techniques for understanding user emotions and delivering uplifting multimedia suggestions. It simulates an emotional assistant using simple keyword logic and showcases how interactive web UIs can be built quickly with Gradio.

2. Task Descriptions

Task 1: Designing the Mood-to-Music Mapping Logic
I created a dedicated Python file named data.py that stores a dictionary mapping moods such as happy, sad, stressed, relaxed, and excited to:

- A list of two songs per mood (title and YouTube link)
- A matching motivational message

This mapping allows the app to return varied results and simulate real-time musical empathy.

Code:

```
# data.py

mood_song_map = {

   "happy": {

      "songs": [

         ("Pharrell Williams - Happy",
"https://www.youtube.com/watch?v=ZbZSe6N_BXs"),

         ("Katrina & The Waves - Walking on Sunshine",
"https://www.youtube.com/watch?v=iPUmE-tne5U"),

      ],

      "message": "Keep smiling! Music makes happiness louder 🎵"

   },

   "sad": {

      "songs": [

         ("Adele - Someone Like You",
"https://www.youtube.com/watch?v=hLQl3WQQoQ0"),

         ("Billie Eilish - everything i wanted",
"https://www.youtube.com/watch?v=EgBJmlPo8Xw"),

      ],

      "message": "It's okay to feel down. Music heals the soul 💙"

   },
    "relaxed": {
        "songs": [
            ("Coldplay - Strawberry Swing",
"https://www.youtube.com/watch?v=h3pJZSTQqIg"),
            ("Norah Jones - Come Away With Me",
"https://www.youtube.com/watch?v=lbjZPFBD6JU"),
        ],
        "message": "Take a deep breath and enjoy the calm 🌿"
    },
    "stressed": {
        "songs": [
            ("Bob Marley - Don't Worry Be Happy",
"https://www.youtube.com/watch?v=d-diB65scQU"),
            ("Moby - Porcelain", "https://www.youtube.com/watch?v=13EifDb4GYs"),
        ],
        "message": "Hang in there. Music is your stress-buster 🧘‍♀️"
    },
    "excited": {
        "songs": [
            ("Avicii - Wake Me Up", "https://www.youtube.com/watch?v=IcrbM1l_BoI"),
            ("Queen - Don't Stop Me Now",
"https://www.youtube.com/watch?v=HgzGwKwLmgM"),
        ],
        "message": "You're on fire! Let the music fuel your energy 🚀"
    }
}
```

Task 2: Implementing Mood Detection

In mood_logic.py, I implemented a function that takes user input and checks for keywords like “happy”, “sad”, or “anxious”. The input string is converted to lowercase for consistency, and the code loops through predefined mood keywords to find a match.

This function provides a lightweight, rule-based NLP alternative to sentiment analysis.

Code:

```
# mood_logic.py

import random

from data import mood_song_map

mood_keywords = {

    "happy": ["happy", "joyful", "cheerful", "smiling"],

    "sad": ["sad", "down", "crying", "depressed"],

    "relaxed": ["relaxed", "calm", "chill", "peaceful"],

    "stressed": ["stressed", "anxious", "tense", "worried"],

    "excited": ["excited", "thrilled", "pumped", "energetic"]

}

def detect_mood(user_input):

    user_input = user_input.lower()

    for mood, keywords in mood_keywords.items():

        for keyword in keywords:

            if keyword in user_input:

                return mood

    return None


def suggest_song(mood):

    if mood not in mood_song_map:

        return None, None, None

    songs = mood_song_map[mood]["songs"]

    message = mood_song_map[mood]["message"]

    song_title, song_url = random.choice(songs)
    return song_title, song_url, message
```

Task 3: Song Suggestion and Output Formatting

Another function within mood_logic.py takes the detected mood and randomly selects a song and message from the corresponding mood group. It returns:

- The song title
- The YouTube URL
- A motivational message

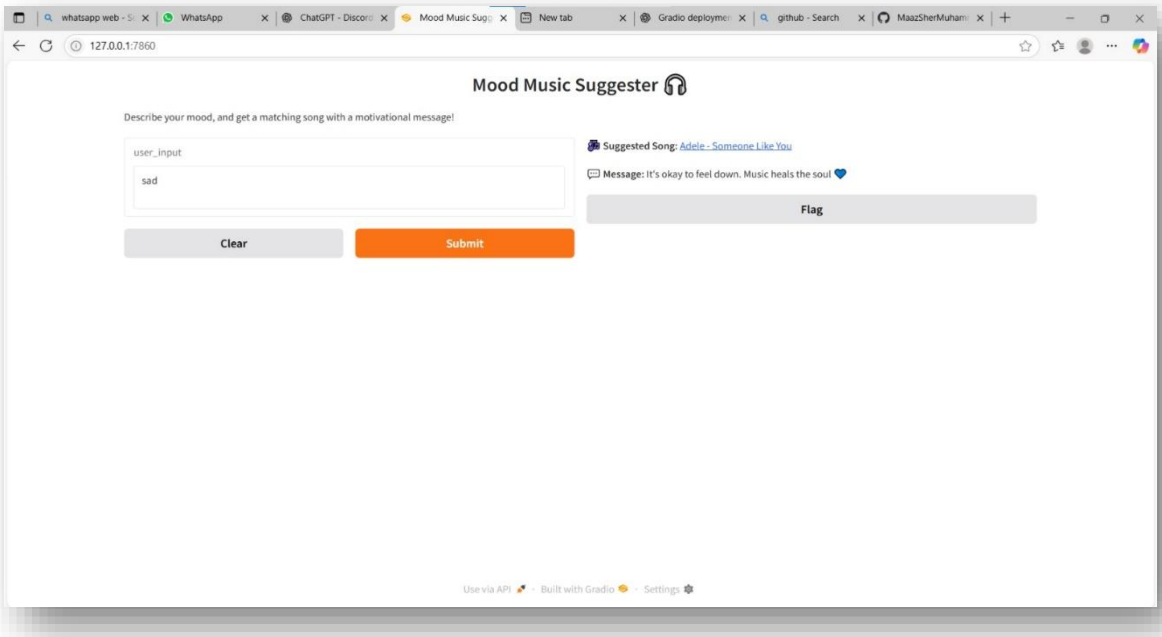
This helps the app feel more dynamic and less repetitive.

Task 4: Building the Gradio Interface

In app.py, I created a simple Gradio interface with:

- A textbox input (for users to type how they're feeling)
- A markdown output area (for showing song + message)
- A title and description for the UI

This interface is launched using Gradio's Interface class and provides real-time interaction in the browser.



Mood Music Suggester 🎧

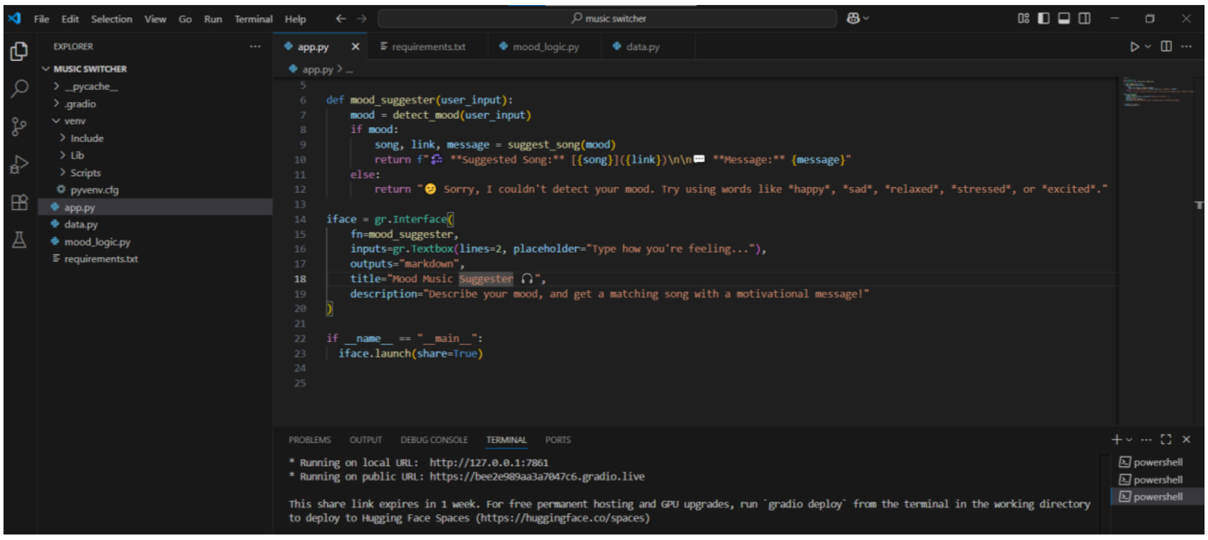
Describe your mood, and get a matching song with a motivational message!

user_input

sad

Suggested Song: Adele - Someone Like You

Message: It's okay to feel down. Music heals the soul 💙



```python

def mood_suggester(user_input):
    mood = detect_mood(user_input)
    if mood:
        song, link, message = suggest_song(mood)
        return f"🎶 **Suggested Song:** [{song}]({link})\n\n💬 **Message:** {message}"
    else:
        return "😕 Sorry, I couldn't detect your mood. Try using words like *happy*, *sad*, *relaxed*, *stressed*, or *excited*."

iface = gr.Interface(
    fn=mood_suggester,
    inputs=gr.Textbox(lines=2, placeholder="Type how you're feeling..."),
    outputs="markdown",
    title="Mood Music Suggester 🎧",
    description="Describe your mood, and get a matching song with a motivational message!"
)

if __name__ == "__main__":
  iface.launch(share=True)

```

```
* Running on local URL:  http://127.0.0.1:7861
* Running on public URL: https://bee2e989aa3a7047c6.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory
to deploy to Hugging Face Spaces (https://huggingface.co/spaces)
```

4. Challenges and Solutions

🔶 Problem: Gradio CLI Confusion
Initially, the gradio login and deploy commands failed due to the wrong version being used (Python package instead of CLI tool).
✅ Solution: Installed the correct CLI using pipx and used the direct path to execute the login.

🔶 Problem: Basic Keyword Matching
The app initially only responded to exact matches like "happy" or "sad".
✅ Solution: Enhanced the keyword matching logic with a synonym list for each emotion, like ["joyful", "cheerful", "delighted"] for "happy".

🔶 Problem: Structuring the App
Combining logic and UI code made the project messy.
✅ Solution: Refactored into three modular files (data.py, mood_logic.py, and app.py), improving readability and future maintainability.

5. Conclusion

This project demonstrates how emotional intelligence can be simulated using simple Python tools. While not AI-driven, the app still delivers engaging, personalized song suggestions using rule-based mood detection.

The final product is an interactive Gradio web app that provides users with emotional support through music. This project lays the foundation for future enhancements like:

- Sentiment analysis via TextBlob or NLP libraries
- Spotify/YouTube API integration
- Voice input or speech-to-text
- Enhanced UI with buttons, emoji, or visual themes

Overall, the Mood Music Suggester fulfills its goal of combining empathy and entertainment through code.
