```
import spotipy

from spotipy.oauth2 import SpotifyClientCredentials

from spotipy.oauth2 import SpotifyOAuth

from textblob import TextBlob

import random

import pandas as pd

from IPython.display import display, HTML

!pip install transformers torch
```

```
Requirement already satisfied: transformers in c:\anaconda3\lib\site-
packages (4.48.2)
Requirement already satisfied: torch in c:\anaconda3\lib\site-packages
(2.7.0)
Requirement already satisfied: filelock in c:\anaconda3\lib\site-
packages (from transformers) (3.17.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in c:\
anaconda3\lib\site-packages (from transformers) (0.32.3)
Requirement already satisfied: numpy>=1.17 in c:\anaconda3\lib\site-
packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in c:\anaconda3\lib\
site-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\anaconda3\lib\site-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in c:\anaconda3\lib\
site-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in c:\anaconda3\lib\site-
packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\anaconda3\
lib\site-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.1 in c:\anaconda3\lib\
site-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in c:\anaconda3\lib\site-
packages (from transformers) (4.67.1)
Requirement already satisfied: fsspec>=2023.5.0 in c:\anaconda3\lib\
site-packages (from huggingface-hub<1.0,>=0.24.0->transformers)
(2025.3.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\
anaconda3\lib\site-packages (from huggingface-hub<1.0,>=0.24.0-
>transformers) (4.12.2)
Requirement already satisfied: sympy>=1.13.3 in c:\anaconda3\lib\site-
packages (from torch) (1.13.3)
Requirement already satisfied: networkx in c:\anaconda3\lib\site-
packages (from torch) (3.4.2)
Requirement already satisfied: jinja2 in c:\anaconda3\lib\site-
```

```
packages (from torch) (3.1.6)
Requirement already satisfied: setuptools in c:\anaconda3\lib\site-
packages (from torch) (78.1.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\anaconda3\lib\
site-packages (from sympy>=1.13.3->torch) (1.3.0)
Requirement already satisfied: colorama in c:\anaconda3\lib\site-
packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\anaconda3\lib\
site-packages (from jinja2->torch) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\
anaconda3\lib\site-packages (from requests->transformers) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\anaconda3\lib\site-
packages (from requests->transformers) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\anaconda3\lib\
site-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda3\lib\
site-packages (from requests->transformers) (2025.4.26)
```

```
from transformers import pipeline
```

```
!pip install sentence_splitter
```

```
Requirement already satisfied: sentence_splitter in c:\anaconda3\lib\
site-packages (1.4)
Requirement already satisfied: regex>=2017.12.12 in c:\anaconda3\lib\
site-packages (from sentence_splitter) (2024.11.6)
```

```
from sentence_splitter import SentenceSplitter
```

```
!pip install numpy scikit-learn
```

```
Requirement already satisfied: numpy in c:\anaconda3\lib\site-packages
(1.26.4)
Requirement already satisfied: scikit-learn in c:\anaconda3\lib\site-
packages (1.6.1)
Requirement already satisfied: scipy>=1.6.0 in c:\anaconda3\lib\site-
packages (from scikit-learn) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in c:\anaconda3\lib\site-
packages (from scikit-learn) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\anaconda3\
lib\site-packages (from scikit-learn) (3.5.0)
```

```
import numpy as np
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
from sklearn.metrics.pairwise import cosine_similarity
```

```
!pip install gliclass
```

```
Requirement already satisfied: gliclass in c:\anaconda3\lib\site-
packages (0.1.11)
```

```
Requirement already satisfied: numpy<2.0.0,>=1.26.4 in c:\anaconda3\
lib\site-packages (from gliclass) (1.26.4)
Requirement already satisfied: scikit-learn<2.0.0,>=1.0.0 in c:\
anaconda3\lib\site-packages (from gliclass) (1.6.1)
Requirement already satisfied: torch<3.0.0,>=2.0.0 in c:\anaconda3\
lib\site-packages (from gliclass) (2.7.0)
Requirement already satisfied: transformers<=4.48.2,>=4.37.2 in c:\
anaconda3\lib\site-packages (from gliclass) (4.48.2)
Requirement already satisfied: scipy>=1.6.0 in c:\anaconda3\lib\site-
packages (from scikit-learn<2.0.0,>=1.0.0->gliclass) (1.15.3)
Requirement already satisfied: joblib>=1.2.0 in c:\anaconda3\lib\site-
packages (from scikit-learn<2.0.0,>=1.0.0->gliclass) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\anaconda3\
lib\site-packages (from scikit-learn<2.0.0,>=1.0.0->gliclass) (3.5.0)
Requirement already satisfied: filelock in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (3.17.0)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\
anaconda3\lib\site-packages (from torch<3.0.0,>=2.0.0->gliclass)
(4.12.2)
Requirement already satisfied: sympy>=1.13.3 in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (1.13.3)
Requirement already satisfied: networkx in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (3.4.2)
Requirement already satisfied: jinja2 in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (3.1.6)
Requirement already satisfied: fsspec in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (2025.3.2)
Requirement already satisfied: setuptools in c:\anaconda3\lib\site-
packages (from torch<3.0.0,>=2.0.0->gliclass) (78.1.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in c:\
anaconda3\lib\site-packages (from transformers<=4.48.2,>=4.37.2-
>gliclass) (0.32.3)
Requirement already satisfied: packaging>=20.0 in c:\anaconda3\lib\
site-packages (from transformers<=4.48.2,>=4.37.2->gliclass) (24.2)
Requirement already satisfied: pyyaml>=5.1 in c:\anaconda3\lib\site-
packages (from transformers<=4.48.2,>=4.37.2->gliclass) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in c:\anaconda3\lib\
site-packages (from transformers<=4.48.2,>=4.37.2->gliclass)
(2024.11.6)
Requirement already satisfied: requests in c:\anaconda3\lib\site-
packages (from transformers<=4.48.2,>=4.37.2->gliclass) (2.32.4)
Requirement already satisfied: tokenizers<0.22,>=0.21 in c:\anaconda3\
lib\site-packages (from transformers<=4.48.2,>=4.37.2->gliclass)
(0.21.1)
Requirement already satisfied: safetensors>=0.4.1 in c:\anaconda3\lib\
site-packages (from transformers<=4.48.2,>=4.37.2->gliclass) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in c:\anaconda3\lib\site-
packages (from transformers<=4.48.2,>=4.37.2->gliclass) (4.67.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in c:\anaconda3\lib\
```

```
site-packages (from sympy>=1.13.3->torch<3.0.0,>=2.0.0->gliclass)
(1.3.0)
Requirement already satisfied: colorama in c:\anaconda3\lib\site-
packages (from tqdm>=4.27->transformers<=4.48.2,>=4.37.2->gliclass)
(0.4.6)
Requirement already satisfied: MarkupSafe>=2.0 in c:\anaconda3\lib\
site-packages (from jinja2->torch<3.0.0,>=2.0.0->gliclass) (3.0.2)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\
anaconda3\lib\site-packages (from requests-
>transformers<=4.48.2,>=4.37.2->gliclass) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\anaconda3\lib\site-
packages (from requests->transformers<=4.48.2,>=4.37.2->gliclass)
(3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\anaconda3\lib\
site-packages (from requests->transformers<=4.48.2,>=4.37.2->gliclass)
(2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\anaconda3\lib\
site-packages (from requests->transformers<=4.48.2,>=4.37.2->gliclass)
(2025.4.26)
```

```
from gliclass import GLiClassModel, ZeroShotClassificationPipeline

from transformers import AutoTokenizer

SPOTIPY_CLIENT_ID = 'f17bf600f0bb478d9a2d4a92abe94cba'

SPOTIPY_CLIENT_SECRET = '15202bade1954c7fb791b19d60824a11'

auth_manager = SpotifyOAuth(
    client_id=SPOTIPY_CLIENT_ID,
    client_secret=SPOTIPY_CLIENT_SECRET,
    redirect_uri="http://127.0.0.1:8888/callback",
    scope="user-read-private",
    cache_path=".cache"
)
sp = spotipy.Spotify(auth_manager=auth_manager)

emotion_pipeline = pipeline(
    "text-classification",
    model="boltuix/bert-emotion",
    return_all_scores=False,
    device=0
)
```

```
Device set to use cpu
C:\Anaconda3\Lib\site-packages\transformers\pipelines\
text_classification.py:106: UserWarning: `return_all_scores` is now
deprecated,  if want a similar functionality use `top_k=None` instead
of `return_all_scores=True` or `top_k=1` instead of
`return_all_scores=False`.
  warnings.warn(
```

```
zero_shot = pipeline(
    "zero-shot-classification",
    model="valhalla/distilbart-mnli-12-3",
    device=0
)
```

```
Device set to use cpu
```

```
def classify_emotion_basic(text):
    result = emotion_pipeline(text)[0]
    return result['label'].lower(), result['score']

zero_shot_labels = [
    "anger", "disgust", "fear", "sadness", "joy", "surprise",
"neutral",
    "love", "embarrassment", "confusion", "curiosity", "excitement",
    "gratitude", "grief", "hope", "pride", "relief", "romance",
"anxiety",
    "loneliness", "disappointment", "shame", "guilt", "trust"
]

def classify_emotion_refined(text):
    output = zero_shot([text], zero_shot_labels)[0]
    scored = list(zip(output["labels"], output["scores"]))
    top3 = sorted(scored, key=lambda x: x[1], reverse=True)[:3]
    return [(label.lower(), score) for label, score in top3]

def map_emotion_to_genre(emotion_list):
    for emotion, score in emotion_list:
        if emotion in emotion_map:
            return emotion_map[emotion]
    return ("confused", "experimental")

test_inputs = [
    "I just got dumped and everything tastes like dust",
    "My friend surprised me with coffee this morning!",
    "I feel a bit hopeful about the future",
    "I’m embarrassed I tripped on my own shoelace"
]
for text in test_inputs:
    print(f"\nInput: {text}")
    emotions = classify_emotion_refined(text)
    for label, score in emotions:
        print(f"  {label}: {score:.2%}")
```

```
Input: I just got dumped and everything tastes like dust
  disappointment: 39.27%
  disgust: 13.84%
  embarrassment: 9.62%
```

```
Input: My friend surprised me with coffee this morning!
  surprise: 60.07%
  joy: 13.34%
  excitement: 8.76%

Input: I feel a bit hopeful about the future
  excitement: 54.42%
  hope: 14.73%
  joy: 7.64%

Input: I’m embarrassed I tripped on my own shoelace
  embarrassment: 52.88%
  shame: 36.74%
  disgust: 1.92%

emotion_map = {
    "anger":         ("furious",    "punk-rock"),
    "disgust":       ("nauseated",  "goth"),
    "fear":          ("anxious",    "ambient"),
    "sadness":       ("heartbroken","blues"),
    "joy":           ("euphoric",   "electro-house"),
    "surprise":      ("curious",    "experimental"),
    "neutral":       ("balanced",   "chill"),
    "love":          ("romantic",   "indie-pop"),
    "embarrassment":("mortified",  "disco"),
    "confusion":     ("lost",       "electronic"),
    "curiosity":     ("exploratory","world-music"),
    "excitement":    ("energetic",  "edm"),
    "gratitude":     ("uplifted",   "acoustic"),
    "grief":         ("devastated", "ambient"),
    "hope":          ("optimistic", "pop"),
    "pride":         ("confident",  "hip-hop"),
    "relief":        ("liberated",  "soul"),
    "romance":       ("loving",     "singer-songwriter"),
    "anxiety":       ("tense",      "downtempo"),
    "loneliness":    ("isolated",   "acoustic"),
    "disappointment":("pensive",   "indie"),
    "shame":         ("withdrawn",  "classical"),
    "guilt":         ("heavy-hearted","jazz"),
    "trust":         ("secure",     "r-n-b")
}

genre_fallbacks = {
    "punk-rock": "rock",
    "goth": "alternative",
    "ambient": "chill",
    "electro-house": "electronic",
    "experimental": "electronic",
    "indie-pop": "indie-pop",
    "disco": "dance",
```

```
    "edm": "electronic",
    "acoustic": "singer-songwriter",
    "indie": "alternative",
    "classical": "classical",
    "jazz": "jazz",
    "hip-hop": "hip-hop",
    "pop": "pop",
    "electronic": "electronic",
    "world-music": "world-music",
    "downtempo": "electronic",
    "r-n-b": "r-n-b",
    "folk": "folk",
    "soul": "soul"
}

VALID_SEEDS = {
    "acoustic", "afrobeat", "alt-rock", "alternative", "ambient",
"black-metal", "bluegrass",
    "blues", "bossanova", "brazil", "breakbeat", "british",
"cantopop", "chicago-house", "children",
    "chill", "classical", "club", "comedy", "country", "dance",
"dancehall", "death-metal", "deep-house",
    "detroit-techno", "disco", "disney", "drum-and-bass", "dub",
"dubstep", "edm", "electro", "electronic",
    "emo", "folk", "forro", "french", "funk", "garage", "german",
"gospel", "goth", "grindcore", "groove",
    "grunge", "guitar", "happy", "hard-rock", "hardcore", "hardstyle",
"heavy-metal", "hip-hop", "holidays",
    "honky-tonk", "house", "idm", "indian", "indie", "indie-pop",
"industrial", "iranian", "j-dance",
    "j-idol", "j-pop", "j-rock", "jazz", "k-pop", "kids", "latin",
"latino", "malay", "mandopop", "metal",
    "metal-misc", "metalcore", "minimal-techno", "movies", "mpb",
"new-age", "new-release", "opera",
    "pagode", "party", "philippines-opm", "piano", "pop", "pop-film",
"post-dubstep", "power-pop",
    "progressive-house", "psych-rock", "punk", "punk-rock", "r-n-b",
"rainy-day", "reggae", "reggaeton",
    "road-trip", "rock", "rock-n-roll", "rockabilly", "romance",
"sad", "salsa", "samba", "sertanejo",
    "show-tunes", "singer-songwriter", "ska", "sleep", "songwriter",
"soul", "soundtracks", "spanish",
    "study", "summer", "swedish", "synth-pop", "tango", "techno",
"trance", "trip-hop", "turkish",
    "work-out", "world-music"
}

def validate_emotion_pipeline():
    for emotion, (_, genre) in emotion_map.items():
        fallback = genre_fallbacks.get(genre, genre)
```

```
        if fallback not in VALID_SEEDS:
            print(f"Genre '{genre}' (fallback: '{fallback}') is NOT
valid.")
        else:
            print(f"{emotion}: '{genre}' → '{fallback}'")
validate_emotion_pipeline()
```

```
anger: 'punk-rock' → 'rock'
disgust: 'goth' → 'alternative'
fear: 'ambient' → 'chill'
sadness: 'blues' → 'blues'
joy: 'electro-house' → 'electronic'
surprise: 'experimental' → 'electronic'
neutral: 'chill' → 'chill'
love: 'indie-pop' → 'indie-pop'
embarrassment: 'disco' → 'dance'
confusion: 'electronic' → 'electronic'
curiosity: 'world-music' → 'world-music'
excitement: 'edm' → 'electronic'
gratitude: 'acoustic' → 'singer-songwriter'
grief: 'ambient' → 'chill'
hope: 'pop' → 'pop'
pride: 'hip-hop' → 'hip-hop'
relief: 'soul' → 'soul'
romance: 'singer-songwriter' → 'singer-songwriter'
anxiety: 'downtempo' → 'electronic'
loneliness: 'acoustic' → 'singer-songwriter'
disappointment: 'indie' → 'alternative'
shame: 'classical' → 'classical'
guilt: 'jazz' → 'jazz'
trust: 'r-n-b' → 'r-n-b'
```

```
def format_tracks_to_df(tracks):
    return pd.DataFrame([{
        "Track": t['name'],
        "Artist": ", ".join(a['name'] for a in t['artists']),
        "Preview": t['preview_url'],
        "Spotify Link": t['external_urls']['spotify']
    } for t in tracks])

def _fallback_to_playlist(sp, genre, limit=10):
    print(f"\nFalling back to public playlists for genre: '{genre}'")
    try:
        results = sp.search(q=genre, type='playlist', limit=5)
        if not results or not isinstance(results, dict):
            print(" Spotify search returned nothing useful.")
            return pd.DataFrame()
        playlists_section = results.get('playlists')
        if not playlists_section or not isinstance(playlists_section,
dict):
```

```
            print(" 'playlists' section missing or invalid in
result.")
            return pd.DataFrame()
        playlists = playlists_section.get('items', [])
        if not playlists or not isinstance(playlists, list):
            print(" No playlist items found.")
            return pd.DataFrame()
        for pl in playlists:
            pid = pl.get('id')
            if not pid:
                continue
            name = (pl.get('name') or '').strip() or 'Unnamed'
            print(f" Inspecting playlist: {name} (ID: {pid})")
            try:
                tracks_data = sp.playlist_tracks(pid)
            except Exception as e:
                print(f"  Could not fetch playlist {pid}: {e}")
                continue
            track_items = tracks_data.get('items', [])
            if not track_items or not isinstance(track_items, list):
                continue
            cleaned = [
                t['track'] for t in track_items
                if isinstance(t, dict)
                and isinstance(t.get('track'), dict)
                and t['track'].get('preview_url')
            ]
            if cleaned:
                print(f" Found {len(cleaned)} previewable tracks in
'{name}'.")
                return pd.DataFrame([{
                    "Track": t["name"],
                    "Artist": ", ".join(a["name"] for a in
t["artists"]),
                    "Preview": t["preview_url"],
                    "Spotify Link": t["external_urls"]["spotify"]
                } for t in cleaned[:limit]])
        print(" No usable tracks found in any fallback playlists.")
    except Exception as e:
        print(f" Total playlist fallback failure: {e}")
    return pd.DataFrame()

def fetch_spotify_tracks(sp, genre, limit=10):
    VALID_GENRE = genre_fallbacks.get(genre.lower(), genre.lower())
    if VALID_GENRE not in VALID_SEEDS:
        print(f" Genre '{VALID_GENRE}' invalid; using 'chill'
instead.")
        VALID_GENRE = "chill"
    all_found = []
    try:
```

```
        print(f" Searching for genre:'{VALID_GENRE}' tracks")
        res = sp.search(q=f'genre:"{VALID_GENRE}"', type='track',
limit=50)
        items = res.get('tracks', {}).get('items', []) or []
        all_found = [t for t in items if t.get('preview_url')]
    except Exception as e:
        print(f"  Genre search failed: {e}")
    if len(all_found) < limit:
        try:
            print(f" Expanding with keyword: {VALID_GENRE}")
            res2 = sp.search(q=VALID_GENRE, type='track', limit=50)
            items2 = res2.get('tracks', {}).get('items', []) or []
            extra = [t for t in items2 if t.get('preview_url') and t
not in all_found]
            all_found.extend(extra)
        except Exception as e:
            print(f"  Keyword search failed: {e}")
    if len(all_found) >= limit:
        selected = random.sample(all_found, limit)
        return pd.DataFrame([{
            "Track": t["name"],
            "Artist": ", ".join(a["name"] for a in t["artists"]),
            "Preview": t["preview_url"],
            "Spotify Link": t["external_urls"]["spotify"]
        } for t in selected])
    print(" No previewable tracks found via standard search.")
    return pd.DataFrame()

def _ai_emotion_search(sp, emotion_list, limit=10):
    """
    Search Spotify by emotion keywords using emotion labels as
queries.
    This fallback uses the AI-classified emotions directly as search
terms.
    """
    print("\nAI-powered emotion search activated.")
    collected = []
    for emotion, score in emotion_list:
        try:
            print(f" • Searching for tracks related to: '{emotion}'
({score:.1%})")
            res = sp.search(q=emotion, type="track", limit=20)
            tracks = res.get("tracks", {}).get("items", []) or []
            previewables = [t for t in tracks if t.get("preview_url")]
            for t in previewables:
                if t not in collected:
                    collected.append(t)
            if len(collected) >= limit:
                break
        except Exception as e:
```

```
            print(f"   Failed query for '{emotion}': {e}")
    if not collected:
        print("No tracks found using AI emotion fallback.")
        return pd.DataFrame()
    sampled = random.sample(collected, min(limit, len(collected)))
    return format_tracks_to_df(sampled)

def mood_bot():
    user_input = input("Tell me how you're feeling: ")
    try:
        refined_emotions = classify_emotion_refined(user_input)
        print("\nTop Detected Emotions:")
        for label, score in refined_emotions:
            print(f"  {label}: {score:.2%}")
        mood, genre = map_emotion_to_genre(refined_emotions)
        print(f"\nMapped Mood: {mood}")
        print(f"Suggested Genre: {genre}")
        playlist_df = fetch_spotify_tracks(sp, genre)
        if not playlist_df.empty:
            print("\nHere's your personalized playlist:\n")
            display(playlist_df)
            return
        print("\nNo luck with direct track search. Trying public
playlists...")
        playlist_df = _fallback_to_playlist(sp, genre)
        if not playlist_df.empty:
            print("\nHere's your playlist scraped from public
collections:\n")
            display(playlist_df)
            return
        try:
            print("\nTrying Spotify's built-in recommendations...")
            rec = sp.recommendations(seed_genres=[genre], limit=50)
            rec_tracks = rec.get("tracks", []) or []
            rec_filtered = [t for t in rec_tracks if
t.get("preview_url")]
            if rec_filtered:
                sampled = random.sample(rec_filtered, min(10,
len(rec_filtered)))
                playlist_df = format_tracks_to_df(sampled)
                print("\nSpotify's algorithm came through after all:\
n")
                display(playlist_df)
                return
        except Exception as e:
            print(f"Spotify recommendations failed: {e}")
        print("\nAll Spotify logic failed. Using AI to find tracks
based on emotion keywords...")
        ai_df = _ai_emotion_search(sp, refined_emotions)
        if not ai_df.empty:
```

```
            print("\nHere's your AI-curated playlist:\n")
            display(ai_df)
            return
        print("\nAll systems failed. AI and Spotify have nothing left
to give you. Good luck out there.")
    except Exception as e:
        print(f"\n Unexpected mood bot failure: {e}")

mood_bot()

Tell me how you're feeling:  I just got dumped and everything tastes
like dust

Couldn't read cache at: .cache
Couldn't read cache at: .cache
Couldn't read cache at: .cache
Couldn't read cache at: .cache
Couldn't read cache at: .cache
Couldn't read cache at: .cache
Couldn't read cache at: .cache


Top Detected Emotions:
  disappointment: 39.27%
  disgust: 13.84%
  embarrassment: 9.62%

Mapped Mood: pensive
Suggested Genre: indie
 Searching for genre:'alternative' tracks
  Genre search failed: [WinError 10013] An attempt was made to access
a socket in a way forbidden by its access permissions
 Expanding with keyword: alternative
  Keyword search failed: [WinError 10013] An attempt was made to
access a socket in a way forbidden by its access permissions
 No previewable tracks found via standard search.

No luck with direct track search. Trying public playlists...

Falling back to public playlists for genre: 'indie'
 Total playlist fallback failure: [WinError 10013] An attempt was made
to access a socket in a way forbidden by its access permissions

Trying Spotify's built-in recommendations...
Spotify recommendations failed: [WinError 10013] An attempt was made
to access a socket in a way forbidden by its access permissions

All Spotify logic failed. Using AI to find tracks based on emotion
keywords...

AI-powered emotion search activated.
```

```
 • Searching for tracks related to: 'disappointment' (39.3%)
   Failed query for 'disappointment': [WinError 10013] An attempt was
made to access a socket in a way forbidden by its access permissions
 • Searching for tracks related to: 'disgust' (13.8%)
   Failed query for 'disgust': [WinError 10013] An attempt was made to
access a socket in a way forbidden by its access permissions
 • Searching for tracks related to: 'embarrassment' (9.6%)
   Failed query for 'embarrassment': [WinError 10013] An attempt was
made to access a socket in a way forbidden by its access permissions
No tracks found using AI emotion fallback.

All systems failed. AI and Spotify have nothing left to give you. Good
luck out there.
```