```
import streamlit as st
import openai
import os
import json
import re

# Load your OpenAI API key from environment variable
openai.api_key = os.getenv("OPENAI_API_KEY")

def simple_rule_based_analysis(text):
    """
    Simple offline analysis:
    - intent_score: based on presence of keywords (0 to 1)
    - discordance: based on presence of contradictory words (0 to 1)
    """
    text_lower = text.lower()

    # Define positive intent keywords (customize as needed)
    positive_keywords = ["help", "need", "want", "request", "please", "urgent"]
    negative_keywords = ["not", "don't", "never", "no", "can't"]

    # Count keywords occurrences
    pos_count = sum(text_lower.count(word) for word in positive_keywords)
    neg_count = sum(text_lower.count(word) for word in negative_keywords)

    # Calculate intent score as ratio of positive words to total keywords found
    total = pos_count + neg_count
    if total == 0:
        intent_score = 0.0
    else:
        intent_score = pos_count / total

    # Discordance is higher if both positive and negative keywords appear
    discordance = 0.5 if (pos_count > 0 and neg_count > 0) else 0.0

    return intent_score, discordance

def call_gpt_api(text):
    """
    Call OpenAI GPT API to analyze intent and discordance
    """
    prompt = f"""
You are an assistant that analyzes the user's text.
Please provide a JSON response with two scores between 0.0 and 1.0:
- "intent_score": How clear and strong is the user's intent (1.0 is very clear).
- "discordance": How contradictory or uncertain the user's message is (0.0 is no contradictio

Analyze this text:
"""{text}"""

Respond ONLY with a JSON like:
```

```
{{
  "intent_score": 0.85,
  "discordance": 0.1
}}
"""
    response = openai.ChatCompletion.create(
        model="gpt-4o-mini",
        messages=[
            {"role": "system", "content": "You analyze user intent and discordance in text."}
            {"role": "user", "content": prompt}
        ],
        temperature=0,
        max_tokens=100,
    )
    content = response.choices[0].message.content.strip()

    try:
        data = json.loads(content)
        intent_score = float(data.get("intent_score", 0))
        discordance = float(data.get("discordance", 1))
        return intent_score, discordance
    except Exception as e:
        st.error(f"Error parsing GPT response: {e}")
        st.write("Raw GPT response:", content)
        return 0.0, 1.0

def main():
    st.title("Quantum Consent Engine - Hybrid Intent Analysis")
    st.write("Enter your text. The system will analyze intent and discordance using a hybrid

    user_input = st.text_area("Enter text here", height=150)

    if st.button("Analyze Intent and Discordance"):
        if not user_input.strip():
            st.warning("Please enter some text first.")
            return

        with st.spinner("Performing offline analysis..."):
            intent_score, discordance = simple_rule_based_analysis(user_input)

        st.write(f"Offline analysis results - Intent score: {intent_score:.2f}, Discordance:

        # Thresholds for offline analysis
        intent_threshold = 0.6
        discordance_threshold = 0.4

        if intent_score < intent_threshold:
            st.info("Intent score low, calling GPT for detailed analysis...")
            with st.spinner("Calling GPT API..."):
                intent_score, discordance = call_gpt_api(user_input)
            st.write(f"GPT analysis results - Intent score: {intent_score:.2f}, Discordance:
```

```
        # Final decision thresholds
        final_intent_threshold = 0.7
        final_discordance_threshold = 0.3

        if intent_score >= final_intent_threshold and discordance <= final_discordance_thresh
            st.balloons()
            st.success("Passed the intent and discordance thresholds!")
        else:
            st.warning("Threshold not passed · Please reconsider your mental field.")

if __name__ == "__main__":
    main()
```