ChatGPT - Uz

請參考
https://justdone.com/ai-detector?utm_adset_id=180489509478&utm_network=g&utm_matchtype=e&gad_campaignid=22559092043
的版面
我要做使用者輸入一段話，判斷它是AI的程度有多少，要上streamlit線上部屬

我先給你一個**可以直接丟到 Streamlit Cloud 的完整 app.py**，版面走 Justdone 那種「中間一張卡片＋上面說明、下面結果」的風格，偵測模型用 Hugging Face 的 **`openai-community/roberta-base-openai-detector`**（不用付費 API key，只要能上網抓模型即可）。 Hugging Face+1

一、Streamlit App 範例（`app.py`）

```python
import streamlit as st
from transformers import pipeline

# ---- Page config ----
st.set_page_config(
    page_title="AI 文字偵測器",
    page_icon="🤖",
    layout="centered"
)

# ---- Simple custom style，參考 JustDone 那種居中的卡片感 ----
st.markdown(
    """
    <style>
    [data-testid="stAppViewContainer"] {
        background: radial-gradient(circle at top, #eef3ff 0, #f9fbff 40%, #ffffff 100%);
    }
    [data-testid="stHeader"] {
        background-color: rgba(0,0,0,0);
    }
    .detector-card {
        background-color: #ffffff;
        border-radius: 20px;
        padding: 2rem 2.5rem;
        box-shadow: 0 20px 40px rgba(59, 113, 246, 0.18);
    }
    .small-muted {
        font-size: 0.85rem;
        color: #6b7280;
    }
    .result-tag {
        display: inline-block;
        padding: 0.2rem 0.7rem;
        border-radius: 999px;
        font-size: 0.85rem;
        font-weight: 600;
    }
    .tag-ai {
        background: #fee2e2;
        color: #b91c1c;
    }
```

```
    .tag-human {
        background: #dcfce7;
        color: #166534;
    }
    </style>
    """,
    unsafe_allow_html=True,
)

# ---- Load model once ----
@st.cache_resource(show_spinner="正在載入 AI 偵測模型，首次會比較久…")
def load_detector():
    # RoBERTa Base OpenAI Detector：偵測 GPT-2 風格的 AI 文字
    # Fake = AI 生成，Real = 人類文字:contentReference[oaicite:1]{index=1}
    clf = pipeline("text-classification", model="openai-community/roberta-base-openai-detector")
    return clf


def compute_ai_score(detector, text: str) -> float:
    """
    使用 roberta-base-openai-detector 的輸出，換算成「AI 可能性」(0~1)
    label = 'Fake' 代表 AI 生成，score 就是 AI 機率
    label = 'Real' 代表人寫，AI 機率 = 1 - score（粗略對稱處理）
    """
    result = detector(text)[0]
    label = result["label"]
    score = float(result["score"])

    if label.lower() == "fake":
        ai_prob = score
    else:
        ai_prob = 1.0 - score

    # 安全夾在 [0,1]
    ai_prob = max(0.0, min(1.0, ai_prob))
    return ai_prob, label, score


# ---- Layout ----
st.markdown("### 🤖 AI 文字偵測器（Demo）")
st.markdown(
    "參考 Justdone AI Detector 的版面設計：輸入一段文字，估計其中 **AI 生成的可能性**。\n\n"
    "<span class='small-muted'>⚠ 僅供教學／示範使用，請勿拿來做學術違規或懲處判斷。</span>",
    unsafe_allow_html=True,
)

col_left, col_center, col_right = st.columns([1, 2.2, 1])

with col_center:
    st.markdown('<div class="detector-card">', unsafe_allow_html=True)

    st.markdown("#### 檢查這段文字是不是 AI 寫的")
    st.markdown(
        "<span class='small-muted'>貼上文章、作業、段落等文字，系統會給出「較像 AI」或「較像人類」的分數。
</span>",
        unsafe_allow_html=True,
    )

    text = st.text_area(
        "輸入要檢查的文字",
        height=220,
        placeholder="例：\n這是一段測試文字，用來示範 AI 文字偵測工具的效果……",
        label_visibility="collapsed",
    )

    col_btn1, col_btn2 = st.columns([1, 1])
    with col_btn1:
        analyze = st.button("🚀 分析這段文字", use_container_width=True)
    with col_btn2:
        min_len = st.slider("最少字元（建議 ≥ 200）", 50, 600, 200, 50)
```

```
    if analyze:
        if not text.strip():
            st.warning("請先輸入一段文字。")
        elif len(text) < min_len:
            st.warning(f"目前字數約 {len(text)}，少於建議長度 {min_len}，偵測結果可能不太可靠。")
        else:
            # 做推論
            detector = load_detector()
            ai_prob, raw_label, raw_score = compute_ai_score(detector, text)

            percent = round(ai_prob * 100, 1)

            # 判斷主要訊息
            if ai_prob >= 0.7:
                verdict = "🧠 **這段文字「很有可能」是 AI 生成。**"
                tag_html = '<span class="result-tag tag-ai">偏向 AI 文字</span>'
            elif ai_prob <= 0.3:
                verdict = "✍️ **這段文字「比較像」是人類撰寫。**"
                tag_html = '<span class="result-tag tag-human">偏向人類文字</span>'
            else:
                verdict = "🤝 **這段文字介於 AI 與人類風格之間，難以明確判斷。**"
                tag_html = '<span class="result-tag">模糊區間</span>'

            st.markdown("---")
            st.markdown("##### 分析結果")
            st.markdown(tag_html, unsafe_allow_html=True)
            st.markdown(verdict)

            st.markdown(f"**AI 生成可能性：{percent} %**")
            st.progress(ai_prob)

            with st.expander("查看模型原始輸出（給開發者看）"):
                st.json(
                    {
                        "model_label": raw_label,
                        "model_score": raw_score,
                        "ai_probability_used": ai_prob,
                    }
                )

            st.markdown(
                "<span class='small-muted'>說明：本 Demo 使用 Hugging Face 的 "
                "<code>openai-community/roberta-base-openai-detector</code> 模型，"
                "主要是針對 GPT-2 類型輸出訓練，因此對 GPT-3.5 / GPT-4 等新模型與中文內容的準確度有限。
</span>",
                unsafe_allow_html=True,
            )

    st.markdown("</div>", unsafe_allow_html=True)


with st.sidebar:
    st.markdown("### ℹ️ 工具說明")
    st.write(
        "- **模型**：`roberta-base-openai-detector`（開源、免費，不需付費 API key）。\n"
        "- **輸出**：估計文字是 AI 生成的機率（0–100%）。\n"
        "- **用途**：課堂示範、研究測試、網路素養教材。\n"
        "- **限制**：偵測結果可能誤判，不能做為處罰或正式審查依據。\n"
    )
    st.caption(
        "建議：把這個工具當成「網路素養教材」的一部分，搭配討論與人工判讀一起使用。"
    )
```

二、需求套件（requirements.txt 範例）

放在同一個 repo 裡，讓 Streamlit Cloud 看到就會自動安裝：

```
txt


streamlit>=1.32
transformers>=4.40
torch
```

> 註：第一次在雲端跑會花一點時間下載模型權重。

三、部署到 Streamlit Cloud（快速版步驟）

1. **建立 GitHub 專案**
   - 新建一個 repo，例如 **`ai-detector-demo`**
   - 把 **`app.py`** 和 **`requirements.txt`** 推上去
2. **到 Streamlit Cloud**
   - 登入 https://share.streamlit.io↗
   - 「New app」→ 選你的 repo → 指定 main branch & **`app.py`**
3. **按 Deploy**
   - 部署完成後會得到一個網址
   - 之後改 **`app.py`** 推到 GitHub，Streamlit 會自動重新部署

如果你想再「更像 Justdone」一點（例如上面有工具介紹、下面加「重新改寫/人性化」按鈕），我也可以幫你把這個 app 擴充成「偵測＋人性化改寫」的兩段式流程。