```
# biasapp.py - Updated version with admin panel removed
import streamlit as st
import json
import random
import time
import pandas as pd
from sentence_transformers import SentenceTransformer
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Import your database module
from database import create_database, get_condition_assignment, save_survey_response

# ----------------------
# CONFIGURATION - Easy to change settings
# ----------------------
MAX_RESPONSES = 3  # Change this number to set the maximum number of responses allowed

# ----------------------
# 1. Dark Mode + ChatCSS (unchanged)
# ----------------------
st.set_page_config(page_title="AI Response Evaluation", layout="wide")
st.markdown(
   """
   <style>
     /* Page background */
     .reportview-container, .main {
       background-color: #343541;
       color: #d1d5db;
     }
     /* Chat container styling */
     .chat-container {
       max-width: 700px;
       margin: 0 auto 2rem auto;
       padding: 1rem;
       background-color: #444654;
       border-radius: 8px;
     }
     /* User bubble */
     .user-message {
       background-color: #444654;
       color: #f8f8f2;
```

```
      padding: 0.6rem 1rem;
      border-radius: 12px;
      margin: 0.5rem 0;
      text-align: left;
    }
    /* Assistant bubble */
    .assistant-message {
      background-color: #10a37f;
      color: #ffffff;
      padding: 0.6rem 1rem;
      border-radius: 12px;
      margin: 0.5rem 0;
      text-align: left;
    }
    /* Thinking text */
    .thinking {
      font-style: italic;
      color: #a0aec0;
    }
    /* Diagnostic info styling */
    .diagnostic-info {
      background-color: #2d3748;
      padding: 1rem;
      border-radius: 8px;
      margin: 1rem 0;
      border-left: 4px solid #10a37f;
    }
    @keyframes pulse {
      0%   { transform: scale(1); }
      50%  { transform: scale(1.05); }
      100% { transform: scale(1); }
    }
    .assistant-thinking.pulse {
      animation: pulse 1s infinite;
      display: inline-block;
    }
  </style>
  """, unsafe_allow_html=True
)

# Load your cases from a JSON file (unchanged)
@st.cache_data
```

```
def load_cases():
    with open("cases.json") as f:
        return json.load(f)

cases = load_cases()
case_prompts = [c["prompt"] for c in cases]

# Creating an embedder (unchanged)
@st.cache_resource
def load_embedder():
    return SentenceTransformer("all-MiniLM-L6-v2")

embedder = load_embedder()

# Create embeddings (unchanged)
@st.cache_data
def create_embeddings():
    raw_embs = embedder.encode(case_prompts, convert_to_tensor=True)
    list_embs = [emb.detach().cpu().tolist() for emb in raw_embs]
    return np.array(list_embs)

case_embeddings = create_embeddings()

# ---------- Database Integration ---------

# Initialize database
@st.cache_resource
def init_db():
    return create_database()

db = init_db()

conditions = [
    "Control",
    "Group A - Warning Label",
]

# ——— Initialize session state with database integration ———
if "group" not in st.session_state:
    # Get condition assignment from database
    condition, participant_id = get_condition_assignment()
    st.session_state.group = condition
```

```
    st.session_state.participant_id = participant_id

if "remaining" not in st.session_state:
    st.session_state.remaining = cases.copy()

if "current" not in st.session_state:
    st.session_state.current = None

if "history" not in st.session_state:
    st.session_state.history = []

if "search_feedback" not in st.session_state:
    st.session_state.search_feedback = ""

if "search_feedback_type" not in st.session_state:
    st.session_state.search_feedback_type = ""

if "user_query" not in st.session_state:
    st.session_state.user_query = ""

if "response_counter" not in st.session_state:
    st.session_state.response_counter = 0

# ------ Diagnostic information session state ------
if "terms_conditions_complete" not in st.session_state:
    st.session_state.terms_conditions_complete = False

if "diagnostic_complete" not in st.session_state:
    st.session_state.diagnostic_complete = False

if "user_age" not in st.session_state:
    st.session_state.user_age = ""

if "user_profession" not in st.session_state:
    st.session_state.user_profession = ""

group = st.session_state.group

# ------ ADMIN PANEL REMOVED FOR PARTICIPANTS ------
# The admin panel has been moved to a separate admin_dashboard.py file
# Participants will not see any admin functionality
```

```
def display_instructions():
   st.html("""
<div style="background-color:#f0f2f6;padding:20px;border-radius:10px;border-left:5px
solid #1f77b4;">
 <h2 style="color:#1f77b4;margin-top:0;">📋 Instructions</h2>

 <ol style="font-size:16px;line-height:1.6;">
   <li><strong>Copy and paste your clinical vignette and question</strong> into
the search bar below
     <!-- muted quick-tip -->
     <div style="background-color:#f7f9fc;border-left:3px solid #c6dafc;padding:8px
10px;margin-top:6px;border-radius:4px;font-size:10px;">
       <ul style="margin:4px 0 0 20px;padding-left:0;">
           <li>Press <code>Cmd + Enter</code> (Mac) to submit</li>
           <li>Press <code>Ctrl + Enter</code> (Windows) to submit</li>
       </ul>
     </div>
   </li>
   <li>Click <strong>Send to Chatbot</strong>
   <li><strong>Rate the response</strong> using the provided answer choices</li>

   <li><strong>Leave a comment</strong> with your feedback</li>
 </ol>

 <!-- subtle one-shot warning -->
 <div style="background-color:#f7f9fc;border-left:3px solid #c6dafc;padding:8px
10px;margin-top:6px;border-radius:4px;font-size:10px;">
   <ul style="margin:4px 0 0 20px;padding-left:0;">
     <li><strong>No edits</strong> after you press
<em>Submit &amp;nbsp;Next</em>.</li>
     <li>Double-check that your ratings and comments are final.</li>
   </ul>
 </div>
</div>
""")




# ——— Consent Form ———

if not st.session_state.terms_conditions_complete:
```

```
   st.markdown("### 📋 Study Consent")

   st.html("""
<div style="background-color:#f8f9fa;padding:20px;border-radius:10px;border-left:5px solid #007bff;">
 <h3 style="color:#0056b3;margin-top:0;">Research Participation Agreement</h3>

 <p style="font-size:16px;line-height:1.6;">
   <strong>Study Purpose:</strong> Evaluating AI-generated medical recommendations
 </p>

 <p style="font-size:16px;line-height:1.6;">
   <strong>What you'll do:</strong> Review AI responses to clinical cases and provide your professional assessment (10-15 minutes)
 </p>

 <!-- Ethics note in faint-blue tip style -->
 <div style="
     background-color:#f7f9fc;        /* faint grey-blue */
     border-left:3px solid #c6dafc;   /* thin blue accent */
     padding:8px 10px;
     margin:8px 0;
     border-radius:4px;
     font-size:14px;
 ">
   <span style="color:#1565c0;">
     <strong>✓ Ethics Note:</strong> All clinical vignettes are fictional and created for research purposes only. No real patient information is used.
   </span>
 </div>

 <p style="font-size:16px;line-height:1.6;">
   <strong>Privacy:</strong> Your responses are anonymous and will be used solely for research purposes
 </p>
</div>
""")


   # Simple consent
   agree_to_participate = st.checkbox(
```

```
        "I have read the above information and agree to participate in this research
study",
        key="simple_consent"
    )

    if agree_to_participate:
        if st.button("Continue to Participant Information", type="primary"):
            st.session_state.terms_conditions_complete = True
            st.rerun()
    else:
        st.info("Please check the consent box to continue.")

# ——— Diagnostic Information Collection (modified to update database) ———
elif not st.session_state.diagnostic_complete:
    st.markdown(f"### 📋 Participant Information")
    st.markdown("Please provide some basic information before proceeding:")

    col1, col2 = st.columns(2)

    with col1:
        age = st.number_input(
            "Enter your age", min_value=10, max_value=100,
            key="age_input"
        )

    with col2:
        profession = st.selectbox(
            "Profession",
            ["", "Medical Student", "Resident", "Attending Physician", "Nurse", "Other
Healthcare Worker", "Non-Healthcare Professional"],
            key="profession_input"
        )

    # Optional: Add a text input for "Other" profession
    if profession == "Other Healthcare Worker":
        other_profession = st.text_input("Please specify your healthcare role:",
key="other_profession")
        if other_profession:
            profession = f"Other Healthcare Worker: {other_profession}"
    elif profession == "Non-Healthcare Professional":
        other_profession = st.text_input("Please specify your profession:",
key="other_profession")
```

```
        if other_profession:
            profession = f"Non-Healthcare Professional: {other_profession}"

    if age and profession:
        if st.button("Continue to Study", type="primary"):
            # Update database with participant info
            try:
                db.update_participant_info(st.session_state.participant_id, age,
profession)
                st.session_state.user_age = age
                st.session_state.user_profession = profession
                st.session_state.diagnostic_complete = True
                st.rerun()
            except Exception as e:
                st.error(f"Database error: {e}")
    else:
        st.info("Please complete both fields to continue.")

# ——— Search Interface (unchanged from your original) ———
elif st.session_state.current is None and len(st.session_state.history) <
MAX_RESPONSES:
    display_instructions()
    st.markdown(f"### Rate a Vignette Response ({len(st.session_state.history) +
1}/{MAX_RESPONSES})")

    if st.session_state.search_feedback:
        if st.session_state.search_feedback_type == "success":
            st.success(st.session_state.search_feedback)
        elif st.session_state.search_feedback_type == "error":
            st.error(st.session_state.search_feedback)

    query = st.text_area(
        "🔍 Give vignette and question",
        key="search_query",
        height=200,
        placeholder="Enter your clinical vignette and question here..."
    )

    if query:
        if st.button("▶ Send to Chatbot", type="primary"):
            # Perform semantic search (unchanged)
            raw_q = embedder.encode([query], convert_to_tensor=True)
```

```
            q_list = raw_q[0].detach().cpu().tolist()
            q_np = np.array(q_list).reshape(1, -1)
            sims = cosine_similarity(q_np, case_embeddings)[0]
            best_idx = int(np.argmax(sims))
            score = float(np.max(sims))

            if score > 0.8:
                matched_case = cases[best_idx]
                st.session_state.current = matched_case
                st.session_state.response_counter += 1
                st.session_state.search_feedback = ""
                st.session_state.search_feedback_type = ""
                st.session_state.user_query = query
                st.rerun()
            else:
                st.session_state.search_feedback = f"Not enough information. Please
give the clinical vignette and question again."
                st.session_state.search_feedback_type = "error"
                st.rerun()

# ——— Chat Interface (modified submit button to save to database) ———
elif st.session_state.current:
    case = st.session_state.current
    case_id = case["id"]
    anim_flag = f"anim_done_response_{st.session_state.response_counter}"

    display_instructions()
    if anim_flag not in st.session_state:
        st.session_state[anim_flag] = False

    st.markdown(f"## Response {len(st.session_state.history) + 1}")

    # User bubble
    st.markdown(f'<div class="user-message">{st.session_state.user_query}</div>',
unsafe_allow_html=True)

    if not st.session_state[anim_flag]:
        # Animation (unchanged)
        placeholder = st.empty()
        placeholder.markdown('<div class="assistant-thinking pulse">🤖
Thinking…</div>', unsafe_allow_html=True)
        time.sleep(2)
```

```
        placeholder.markdown('<div class="assistant-thinking">🛠️ Finishing
reasoning…</div>', unsafe_allow_html=True)
        time.sleep(1)

        ai_typed = ""
        for ch in case["llm_response"]:
            ai_typed += ch
            placeholder.markdown(f'<div class="assistant-message">{ai_typed}</div>',
unsafe_allow_html=True)
            time.sleep(0.02)

        st.session_state[anim_flag] = True
    else:
        st.markdown(f'<div class="assistant-message">{case["llm_response"]}</div>',
unsafe_allow_html=True)

    # Group-specific interventions (unchanged)
    if group == "Group A - Warning Label":
        st.warning("⚠️ WARNING: Please check the validity of AI responses")

    # Rating widgets (unchanged)
    st.markdown("### Your Assessment")
    agree = st.radio(
        "How much do you agree with this recommendation?",
        ["1 Strongly Disagree", "2 Disagree", "3 Neutral", "4 Agree", "5 Strongly
Agree"],
        key=f"agree_response_{st.session_state.response_counter}",
        horizontal=True
    )

    trust = st.checkbox("Overall, would you follow that recommendation on your real
practice?")

    comment = st.text_area(
        "Give one sentence or a few words to explain your ratings",
        key=f"comment_response_{st.session_state.response_counter}",
        height=100
    )

    # Modified submit button - saves to database
    if st.button("✅ Submit & Next", type="primary"):
        response_data = {
```

```
            "case_id": case_id,
            "response_number": st.session_state.response_counter,
            "group": group,
            "user_age": st.session_state.user_age,
            "user_profession": st.session_state.user_profession,
            "agree": agree,
            "trust": trust,
            "comment": comment
        }

        try:
            # Save to database
            save_survey_response(st.session_state.participant_id, response_data)

            # Also keep in session state for UI
            st.session_state.history.append(response_data)

            # Mark as completed if this was the last response
            if len(st.session_state.history) == MAX_RESPONSES:
                db.mark_participant_completed(st.session_state.participant_id)

            # Reset for next case
            st.session_state.current = None
            st.session_state.user_query = ""
            st.session_state.search_feedback = ""
            st.session_state.search_feedback_type = ""
            st.rerun()

        except Exception as e:
            st.error(f"Failed to save response: {e}")

# ——— Study Complete (unchanged except for database completion marking) ———
if len(st.session_state.history) == MAX_RESPONSES:
    st.markdown("""
<div style="background-color:#d4edda;padding:20px;border-radius:10px;border-left:5px
solid #28a745;">
 <h2 style="color:#155724;margin-top:0;">🎉 Thank You for Completing the Study!</h2>

 <ul style="color:#155724;font-size:16px;line-height:1.5;margin:0 0 0 1em;padding:0;">
   <li>Your responses have been <strong>successfully recorded</strong>.</li>
   <li>Your participation is greatly appreciated and will provide <strong>valuable
insights</strong> to our research.</li>
```

```
   <li><strong>Reminder:</strong> please do not share this website or the results of
this study with anyone.</li>
 </ul>
</div>
""", unsafe_allow_html=True)
```