```
import streamlit as st
import joblib
import pandas as pd
import os
from pathlib import Path
from model_training import
preprocess_text

# Set page config
st.set_page_config(
    page_title="Intelligent Chatbot",
    page_icon="🤖",
    layout="centered"
)

# Define paths
BASE_DIR =
Path(__file__).parent.absolute()
```

```
MODEL_PATH = BASE_DIR /
'chatbot_model.joblib'

# Custom CSS
st.markdown("""
    <style>
    .stTextInput>div>div>input {
        background-color: #f0f2f6;
    }
    .chat-message {
        padding: 1.5rem;
        border-radius: 0.5rem;
        margin-bottom: 1rem;
        display: flex;
        flex-direction: column;
    }
    .chat-message.user {
        background-color: #2b313e;
        color: white;
    }
    .chat-message.bot {
        background-color: #f0f2f6;
        color: #0f1112;  /* Dark text color for
bot messages */
        font-weight: 500;  /* Medium font
weight for better readability */
    }
    /* Additional styles for better visibility
*/
    .stMarkdown {
        color: #0f1112;
    }
    </style>
""", unsafe_allow_html=True)

def load_model():
    """Load the trained model."""
    try:
        if not MODEL_PATH.exists():
            st.error(f"Model file not found at
{MODEL_PATH}")
            st.sidebar.error("Debug Info:")
            st.sidebar.info(f"Looking for
model at: {MODEL_PATH}")
            st.sidebar.info(f"Current
directory: {os.getcwd()}")
            st.sidebar.info(f"Directory
contents: {list(BASE_DIR.glob('*'))}")
            return None

        model = joblib.load(MODEL_PATH)
        st.sidebar.success("Model loaded
successfully!")
        return model
    except Exception as e:
        st.error(f"Error loading model:
{str(e)}")
        st.sidebar.error("Debug Info:")
        st.sidebar.info(f"Error type:
{type(e).__name__}")
        st.sidebar.info(f"Error details:
{str(e)}")
        return None
```

```
def get_response(model, query):
    """Get response from the model."""
    processed_query =
preprocess_text(query)
    return
model.predict([processed_query])[0]

def main():
    st.title("🤖 Intelligent Chatbot")
    st.markdown("Welcome! I'm your AI
assistant. How can I help you today?")

    # Initialize chat history
    if "messages" not in st.session_state:
        st.session_state.messages = []

    # Load the model
    model = load_model()
    if model is None:
        return
    # Display chat history
    for message in
st.session_state.messages:
        with st.container():
            st.markdown(f"""
                <div class="chat-message
{message['role']}">
                    <div>{message['content']}</
div>
                </div>
            """, unsafe_allow_html=True)

    # Chat input
    if prompt := st.chat_input("Type your
message here..."):
        # Add user message to chat history

st.session_state.messages.append({"role"
: "user", "content": prompt})
```

```
        # Get bot response
        response = get_response(model,
prompt)

        # Add bot response to chat history

st.session_state.messages.append({"role"
: "bot", "content": response})

        # Rerun to update the chat
        st.rerun(if __name__ ==
"__main__":
    main()
```