import streamlit as st

import requests

import json

from datetime import datetime, timedelta

# Load API key from Streamlit Secrets (secure - set this in Streamlit Cloud dashboard)

WEATHER\_API\_KEY = st.secrets.get("WEATHER\_API\_KEY", "YOUR\_WEATHERAPI\_KEY\_HERE") # Fallback for local testing

# Crop to pesticide mapping (expand as needed)

pesticide\_suggestions = {

"wheat": "Fungicide XYZ (e.g., Carbendazim) - Protects against rust and smut.",

"rice": "Insecticide ABC (e.g., Imidacloprid) - Controls stem borers and leaf folders.",

"maize": "Herbicide DEF (e.g., Atrazine) - Manages weeds like grass and broadleaf.",

"cotton": "Pesticide GHI (e.g., Endosulfan) - Targets bollworms and aphids.",

"sugarcane": "Pesticide JKL (e.g., Chlorpyrifos) - Fights borers and termites.",

# Add more: e.g., "potato": "Some pesticide"

}

# Streamlit App

st.set\_page\_config(page\_title="10-Day Weather & Pesticide Chatbot", page\_icon="🌤️", layout="centered")

st.title("🌤️ 10-Day Weather Forecast & Pesticide Chatbot for Ludhiana Farmers")

st.markdown("---")

# Initialize session state for chat history and conversation flow

if "messages" not in st.session\_state:

st.session\_state.messages = []

if "step" not in st.session\_state:

st.session\_state.step = 0 # 0: Greeting, 1: After forecast (ask crop), 2: After pesticide

if "crop" not in st.session\_state:

st.session\_state.crop = ""

# Function to fetch 10-day forecast

@st.cache\_data(ttl=1800) # Cache for 30 minutes

def get\_10day\_forecast():

city = "Ludhiana"

days = 10

url = f"http://api.weatherapi.com/v1/forecast.json?key={WEATHER\_API\_KEY}&q={city}&days={days}"

try:

response = requests.get(url)

data = response.json()

if response.status\_code == 200:

forecast\_list = []

for i in range(days):

day\_data = data["forecast"]["forecastday"][i]

date = day\_data["date"]

max\_temp = day\_data["day"]["maxtemp\_c"]

min\_temp = day\_data["day"]["mintemp\_c"]

avg\_temp = day\_data["day"]["avgtemp\_c"]

condition = day\_data["day"]["condition"]["text"]

forecast\_list.append({

"date": date,

"max\_temp": max\_temp,

"min\_temp": min\_temp,

"avg\_temp": avg\_temp,

"condition": condition

})

return forecast\_list

else:

return None

except Exception as e:

st.error(f"Error fetching forecast: {e}")

return None

# Function to get pesticide suggestion

def get\_pesticide\_suggestion(crop):

crop\_lower = crop.lower().strip()

if crop\_lower in pesticide\_suggestions:

return pesticide\_suggestions[crop\_lower]

else:

return "No specific suggestion available for this crop. Consult a local expert for tailored advice."

# Display chat messages

for message in st.session\_state.messages:

with st.chat\_message(message["role"]):

st.markdown(message["content"])

# Chat input and logic

if prompt := st.chat\_input("Type your message here..."):

# Add user message to chat history

st.session\_state.messages.append({"role": "user", "content": prompt})

with st.chat\_message("user"):

st.markdown(prompt)

# Bot response logic based on step

if st.session\_state.step == 0:

# Greet and fetch 10-day forecast

forecast\_data = get\_10day\_forecast()

if forecast\_data:

forecast\_msg = "\*\*10-Day Weather Forecast for Ludhiana (starting today):\\n\\n"

for day in forecast\_data:

forecast\_msg += f"- \*\*{day['date']}\*\*: Max {day['max\_temp']}°C / Min {day['min\_temp']}°C | Avg {day['avg\_temp']:.1f}°C | {day['condition']}\\n"

forecast\_msg += "\\nWhich crop are you growing? (e.g., wheat, rice, maize) I can suggest suitable pesticides based on the weather trends."

st.session\_state.messages.append({"role": "assistant", "content": forecast\_msg})

st.session\_state.step = 1

else:

error\_msg = "Sorry, I couldn\\'t fetch the 10-day forecast right now. Please try again later.\\n\\nWhich crop are you growing?"

st.session\_state.messages.append({"role": "assistant", "content": error\_msg})

st.session\_state.step = 1

elif st.session\_state.step == 1:

# User provides crop

st.session\_state.crop = prompt

pesticide = get\_pesticide\_suggestion(st.session\_state.crop)

pesticide\_msg = f"\*\*Suggested Pesticide for {st.session\_state.crop} (considering upcoming weather):\*\*\\n{pesticide}\\n\\n\*Note: Weather forecast suggests planning applications during milder conditions. Always follow local guidelines and safety instructions.\*\\n\\nThank you for using the chatbot! If you need more help, refresh the page."

st.session\_state.messages.append({"role": "assistant", "content": pesticide\_msg})

st.session\_state.step = 2

elif st.session\_state.step == 2:

# Conversation ended

end\_msg = "Our conversation has ended. Refresh the page to start a new one with the latest 10-day forecast and crop advice."

st.session\_state.messages.append({"role": "assistant", "content": end\_msg})

# Rerun to display bot response

st.rerun()

# Initial greeting if no messages

if not st.session\_state.messages:

greeting = "Hello! 👋 I\\'m your AI assistant for farmers in Ludhiana, Punjab. I provide 10-day weather forecasts and suggest pesticides based on your crop.\\n\\nType anything to get started (e.g., \\'Hi\\' or \\'Start\\')."

st.session\_state.messages.append({"role": "assistant", "content": greeting})

st.rerun()

# Sidebar for info

with st.sidebar:

st.markdown("### About")

st.markdown("- \*\*Weather Source:\*\* WeatherAPI (10-day forecast)")

st.markdown("- \*\*Crops Supported:\*\* Wheat, Rice, Maize, Cotton, Sugarcane (add more in code)")

st.markdown("- \*\*Disclaimer:\*\* Forecasts are estimates; pesticide suggestions are general. Consult agricultural experts.")

# Footer (optional: add your contact for farmers)

st.markdown("---")

st.markdown("\*Built for Ludhiana farmers. Questions? Contact [your-email@example.com](mailto:your-email@example.com). 🌾\*")