**Step 1:**

!pip install streamlit pyngrok

!pip install folium streamlit-folium

!pip install streamlit folium streamlit-folium requests geopy pyngrok

!pip install streamlit\_javascript

!pip install --upgrade streamlit

**Step 2:**

%%writefile app.py

import streamlit as st

import matplotlib.pyplot as plt

import folium

from streamlit\_folium import st\_folium

import requests

from geopy.geocoders import Nominatim

import random

import urllib.parse

import json

import datetime

import pandas as pd

# ================= CONFIG ===================

GOOGLE\_PLACES\_API\_KEY = "AIzaSyC\_Rw\_jhoIFnGjMmcuGwS7xpHMiSkNb7pw"

# ================= SESSION STATE ===================

st.set\_page\_config(page\_title="CO₂ Planner Pro", layout="wide")

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

    st.session\_state.page = "home"

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

    st.session\_state.choice = None

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

    st.session\_state.emission = 0.0

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

    st.session\_state.target = 0

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

    st.session\_state.city = ""

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

    st.session\_state.state = ""

if "filter\_type" not in st.session\_state:

    st.session\_state.filter\_type = "park"

if "chat\_open" not in st.session\_state:

    st.session\_state.chat\_open = False

if "chat\_history" not in st.session\_state:

    st.session\_state.chat\_history = []

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

    st.session\_state.history = []

if "eco\_score" not in st.session\_state:

    st.session\_state.eco\_score = 0

# ================= FUNCTIONS ===================

def go\_home():

    st.session\_state.page = "home"

def choose\_reduction():

    st.session\_state.choice = "reduction"

    st.session\_state.page = "form"

def choose\_place():

    st.session\_state.choice = "place"

    st.session\_state.page = "form"

def get\_coordinates(city, state):

    geolocator = Nominatim(user\_agent="co2\_planner")

    location = geolocator.geocode(f"{city}, {state}")

    if location:

        return location.latitude, location.longitude

    return None, None

def fetch\_low\_co2\_places(lat, lng, filter\_type="park"):

    places = []

    url = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"

    params = {

        "key": GOOGLE\_PLACES\_API\_KEY,

        "location": f"{lat},{lng}",

        "radius": 50000,

        "keyword": filter\_type,

        "type": filter\_type

    }

    res = requests.get(url, params=params)

    if res.status\_code == 200:

        results = res.json().get("results", [])

        for place in results[:10]:

            places.append({

                "place": place.get("name"),

                "lat": place["geometry"]["location"]["lat"],

                "lng": place["geometry"]["location"]["lng"],

                "address": place.get("vicinity")

            })

    return places

def chatbot\_reply(user\_input):

    replies = [

        "Try entering your city and state to find eco-friendly places.",

        "Use the reduction option to calculate your target emissions.",

        "Consider walking or cycling to reduce emissions.",

        "Trees and plants help absorb CO₂ from the atmosphere!",

        "Did you know? Transportation is one of the biggest CO₂ contributors."

    ]

    return random.choice(replies)

def save\_history(action, data):

    st.session\_state.history.append({"time": datetime.datetime.now(), "action": action, "data": data})

def generate\_report():

    df = pd.DataFrame(st.session\_state.history)

    filename = f"CO2\_Report\_{datetime.datetime.now().strftime('%Y%m%d\_%H%M%S')}.csv"

    df.to\_csv(filename, index=False)

    return filename

def eco\_score\_calculator(reduced\_percentage):

    if reduced\_percentage >= 50:

        return "Eco Hero 🌿", 100

    elif reduced\_percentage >= 30:

        return "Green Champion 🍃", 70

    elif reduced\_percentage >= 10:

        return "Eco Starter 🌱", 40

    else:

        return "Eco Beginner 🌏", 10

def daily\_challenge():

    challenges = [

        "🚶 Walk instead of taking a car for short distances today!",

        "♻️ Recycle one item you usually throw away.",

        "💡 Turn off lights when not in use for a day.",

        "🍽️ Reduce meat consumption for a meal today.",

        "🌱 Plant a seed or small plant."

    ]

    return random.choice(challenges)

# ================= MAIN APP ===================

def main():

    st.title("🌱 CO₂ Planner Pro")

    menu = ["Home", "My Dashboard", "History", "Report"]

    choice = st.sidebar.selectbox("Menu", menu)

    if choice == "Home":

        if st.session\_state.page == "home":

            st.write("Choose what you want to explore:")

            st.button("💡 Know My Reduction", on\_click=choose\_reduction)

            st.button("📍 Find Low CO₂ Places", on\_click=choose\_place)

            # Daily Challenge

            st.markdown("### 🎯 Daily Carbon Challenge")

            st.info(daily\_challenge())

        elif st.session\_state.page == "form":

            if st.session\_state.choice == "reduction":

                st.header("Know Your CO₂ Reduction Plan")

                st.session\_state.emission = st.number\_input("Enter your current CO₂ emission (kg/year):", min\_value=0.0, step=0.1)

                st.session\_state.target = st.number\_input("Enter your target reduction (%)", min\_value=0, max\_value=100, step=1)

                if st.button("Calculate"):

                    st.session\_state.page = "result"

            elif st.session\_state.choice == "place":

                st.header("Find Places with Low CO₂ Emission")

                st.session\_state.city = st.text\_input("Enter City:")

                st.session\_state.state = st.text\_input("Enter State:")

                filter\_type = st.selectbox("Filter by Type", ["park", "nature", "garden", "sanctuary"])

                if st.button("Find Places"):

                    st.session\_state.page = "result"

                    st.session\_state.filter\_type = filter\_type

            st.button("🔙 Back", on\_click=go\_home)

        elif st.session\_state.page == "result":

            if st.session\_state.choice == "reduction":

                reduced = st.session\_state.emission \* (1 - st.session\_state.target / 100)

                st.success(f"✅ Your target CO₂ emission: \*\*{reduced:.2f} kg/year\*\* 🌿")

                st.subheader("How to reduce your CO₂ emissions:")

                tips = [

                    "🚶 Walk or cycle instead of using a car for short distances.",

                    "🔌 Use energy-efficient appliances.",

                    "🌱 Reduce meat consumption.",

                    "♻️ Recycle and reuse materials.",

                    "💡 Use renewable energy sources if possible."

                ]

                for tip in tips:

                    st.write(tip)

                # Eco Score

                score\_label, score\_value = eco\_score\_calculator(st.session\_state.target)

                st.markdown(f"### 🌟 Your Eco Score: {score\_label}")

                st.session\_state.eco\_score = score\_value

                # Virtual Tree Planting

                trees\_planted = int(score\_value / 20)

                st.markdown(f"🌳 You’ve planted \*\*{trees\_planted} virtual trees\*\*!")

                fig, ax = plt.subplots()

                ax.bar(["Current Emission", "Target Emission"], [st.session\_state.emission, reduced], color=["orange", "green"])

                ax.set\_ylabel("CO₂ Emission (kg/year)")

                st.pyplot(fig)

                save\_history("Reduction Plan", {"emission": st.session\_state.emission, "target": st.session\_state.target, "reduced": reduced})

            elif st.session\_state.choice == "place":

                lat, lng = get\_coordinates(st.session\_state.city, st.session\_state.state)

                if lat and lng:

                    places = fetch\_low\_co2\_places(lat, lng, st.session\_state.filter\_type)

                else:

                    st.error("Could not find location coordinates.")

                    places = []

                if places:

                    st.write(f"Places with lowest CO₂ near \*\*{st.session\_state.city}, {st.session\_state.state}\*\*:")

                    for p in places:

                        query = urllib.parse.quote(p['place'] + " " + st.session\_state.city)

                        link = f"[{p['place']}](https://www.google.com/maps/search/{query})"

                        st.markdown(f"🌿 {link} — {p.get('address','')}")

                    m = folium.Map(location=[lat, lng], zoom\_start=12)

                    for p in places:

                        query = urllib.parse.quote(p['place'] + " " + st.session\_state.city)

                        popup\_html = f"<a href='https://www.google.com/maps/search/{query}' target='\_blank'>{p['place']}</a>"

                        folium.Marker(location=[p['lat'], p['lng']], popup=popup\_html, icon=folium.Icon(color="green")).add\_to(m)

                    st\_folium(m, width=700, height=500)

                    save\_history("Low CO₂ Places", {"city": st.session\_state.city, "state": st.session\_state.state, "places": places})

                else:

                    st.warning("No eco-friendly places found nearby.")

    elif choice == "My Dashboard":

        st.header("📊 My CO₂ Dashboard")

        if st.session\_state.history:

            df = pd.DataFrame(st.session\_state.history)

            st.dataframe(df)

            st.markdown(f"### 🌟 Eco Score: {st.session\_state.eco\_score}")

            st.bar\_chart(df.index)

        else:

            st.write("No data yet. Start exploring!")

    elif choice == "History":

        st.header("📜 History")

        for h in st.session\_state.history:

            st.write(f"{h['time']} — {h['action']}: {h['data']}")

    elif choice == "Report":

        st.header("📄 Generate Report")

        if st.button("Generate CSV Report"):

            filename = generate\_report()

            st.success(f"Report generated: {filename}")

            with open(filename, "rb") as f:

                st.download\_button("📥 Download Report", f, file\_name=filename)

    # ===== CHATBOT =====

    st.markdown("<style>.chatbot-container{position:fixed;bottom:20px;right:20px;}</style>", unsafe\_allow\_html=True)

    if st.button("💬 Chatbot", key="chat\_toggle"):

        st.session\_state.chat\_open = not st.session\_state.chat\_open

    if st.session\_state.chat\_open:

        with st.container():

            st.markdown("<div style='position:fixed; bottom:80px; right:20px; "

                        "background-color:white; padding:10px; border:1px solid gray; border-radius:5px; width:300px;'>"

                        "<h4>CO₂ Planner Chatbot</h4></div>", unsafe\_allow\_html=True)

            for speaker, message in st.session\_state.chat\_history[-10:]:

                st.markdown(f"\*\*{speaker}:\*\* {message}")

            user\_input = st.text\_input("Type your message here", key="chat\_input")

            if user\_input:

                reply = chatbot\_reply(user\_input)

                st.session\_state.chat\_history.append(("You", user\_input))

                st.session\_state.chat\_history.append(("Bot", reply))

                st.session\_state.page = st.session\_state.page

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**step 3:**

from pyngrok import ngrok

import getpass

import os

import time

# Ask for ngrok auth key securely

NGROK\_AUTH = getpass.getpass("Enter your ngrok auth key: ")

# Set ngrok auth token

ngrok.set\_auth\_token(NGROK\_AUTH)

# Kill old tunnels

ngrok.kill()

# Start streamlit app in background

get\_ipython().system\_raw("streamlit run app.py &")

time.sleep(5)  # wait for streamlit to start

# Create tunnel (correct syntax)

public\_url = ngrok.connect(addr="8501", proto="http")

print("🔗 Click this URL to open your app:", public\_url)