Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 local 2222 / 2223
E-mail Address: cics.alangilan@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

EcoTrack - Carbon Footprint Calculator and Reducer App

TITLE

CS121: ADVANCED COMPUTER PROGRAMMING

Submitted By:

JALOS, RANIEL CLARENCE C.

IT – 2106

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 local 2222 / 2223
E-mail Address: cics.alangilan@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Informatics and Computing Sciences

I. PROJECT OVERVIEW

EcoTrack is a user-friendly application designed to help individuals and businesses measure and reduce their carbon footprint. The app estimates carbon emissions based on various activities like transportation, energy usage, diet, and shopping. It then provides actionable recommendations for reducing emissions and offers a carbon offset marketplace for users who wish to neutralize their environmental impact.

EcoTrack aligns with the principles of sustainability by promoting eco-friendly habits, educating users on the environmental impact of their choices, and helping them track progress towards a greener lifestyle. By utilizing data analysis and machine learning, EcoTrack offers precise insights and personalized reduction plans.

II. PYTHON CONCEPTS AND LIBRARIES

In developing EcoTrack, several Python concepts and libraries were employed to enhance functionality and ensure a seamless user experience:

1. **Data Handling and Analysis:**
   - The application uses **Pandas** and **NumPy** for data manipulation and calculations. These libraries help process user input, estimate carbon emissions, and generate insights based on real-time data.
   - **Matplotlib** and **Seaborn** are used for creating visualizations, such as bar charts and pie charts, that display a user's carbon footprint breakdown.
2. **Machine Learning Model:**
   - **Scikit-Learn** was used to build a predictive model that analyzes user behavior patterns and suggests the best actions to reduce carbon emissions. The model is trained on a dataset of common activities and their associated carbon impacts.
3. **API Integration:**
   - The app uses external APIs (e.g., transportation emission APIs) to fetch real-time data for accurate carbon footprint calculations.
   - **Requests** library in Python is used for making API calls and handling responses.
4. **User Interface:**
   - The backend is developed using **Flask**, a lightweight web framework, which handles user requests, processes data, and serves results.
   - **Streamlit** can also be used as an alternative for a quick and interactive frontend interface.
5. **Data Storage:**
   - User data and activity logs are stored using **SQLite** or **PostgreSQL**, ensuring efficient and scalable data management.

Python Tkinter

```
import tkinter as tk
from tkinter import messagebox
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

# Function to calculate carbon footprint
def calculate_footprint():
    try:
        transportation = float(entry_transport.get())
        electricity = float(entry_electricity.get())
        diet = float(entry_diet.get())
        shopping = float(entry_shopping.get())

        # Carbon footprint calculation (example formula)
        total_emissions = (
            transportation * 0.21 +  # kg CO2 per km
            electricity * 0.5 +     # kg CO2 per kWh
            diet * 1.5 +           # kg CO2 per meal
            shopping * 2.0         # kg CO2 per item
        )

        result_label.config(text=f"Total Carbon Footprint: {total_emissions:.2f} kg CO2")
        suggest_tips(total_emissions)
    except ValueError:
        messagebox.showerror("Input Error", "Please enter valid numeric values.")

# Function to suggest tips based on total emissions
def suggest_tips(total_emissions):
    tips_text.delete(1.0, tk.END)
    tips_text.insert(tk.END, "Suggestions to reduce your footprint:\n\n")

    if total_emissions > 1000:
        tips_text.insert(tk.END, "- Consider carpooling or using public transport.\n")
        tips_text.insert(tk.END, "- Switch to energy-efficient appliances.\n")
    elif total_emissions > 500:
        tips_text.insert(tk.END, "- Reduce meat consumption in your diet.\n")
        tips_text.insert(tk.END, "- Minimize online shopping.\n")
    else:
        tips_text.insert(tk.END, "- Keep up the good work! Aim for even smaller
emissions.\n")

# Function to show a pie chart of contributions
def show_pie_chart():
```

```
        try:
            transportation = float(entry_transport.get())
            electricity = float(entry_electricity.get())
            diet = float(entry_diet.get())
            shopping = float(entry_shopping.get())

            labels = ['Transportation', 'Electricity', 'Diet', 'Shopping']
            values = [
                transportation * 0.21,
                electricity * 0.5,
                diet * 1.5,
                shopping * 2.0
            ]

            fig, ax = plt.subplots()
            ax.pie(values, labels=labels, autopct='%1.1f%%', startangle=140)
            ax.set_title("Carbon Footprint Breakdown")

            # Display the pie chart in the Tkinter window
            canvas = FigureCanvasTkAgg(fig, master=window)
            canvas.draw()
            canvas.get_tk_widget().pack()
        except ValueError:
            messagebox.showerror("Input Error", "Please enter valid numeric values.")

    # Create the Tkinter window
    window = tk.Tk()
    window.title("EcoTrack - Carbon Footprint Calculator")
    window.geometry("500x600")

    # Input fields
    tk.Label(window, text="Transportation (km traveled):").pack()
    entry_transport = tk.Entry(window)
    entry_transport.pack()

    tk.Label(window, text="Electricity Usage (kWh):").pack()
    entry_electricity = tk.Entry(window)
    entry_electricity.pack()

    tk.Label(window, text="Diet (meals per day):").pack()
    entry_diet = tk.Entry(window)
    entry_diet.pack()

    tk.Label(window, text="Shopping (items purchased):").pack()
    entry_shopping = tk.Entry(window)
    entry_shopping.pack()
```

```
# Calculate button
calculate_button = tk.Button(window, text="Calculate Footprint",
command=calculate_footprint)
calculate_button.pack()

# Result label
result_label = tk.Label(window, text="Total Carbon Footprint: --")
result_label.pack()

# Suggestions section
tk.Label(window, text="Suggestions to Reduce Your Footprint:").pack()
tips_text = tk.Text(window, height=10, width=50)
tips_text.pack()

# Pie chart button
pie_chart_button = tk.Button(window, text="Show Pie Chart",
command=show_pie_chart)
pie_chart_button.pack()

# Run the application
window.mainloop()
```

MySQL

```
import tkinter as tk
from tkinter import messagebox
import mysql.connector
from datetime import datetime

# Connect to MySQL database
db = mysql.connector.connect(
    host="localhost",
    user="root",
    password="yourpassword",  # Replace with your MySQL root password
    database="ecotrack"
)

cursor = db.cursor()

# Function to calculate carbon footprint and save to MySQL
def calculate_and_save():
    try:
        username = entry_username.get()
        transportation = float(entry_transport.get())
        electricity = float(entry_electricity.get())
        diet = float(entry_diet.get())
        shopping = float(entry_shopping.get())
```

```
            # Carbon footprint calculation (example formula)
            total_emissions = (
                transportation * 0.21 +  # kg CO2 per km
                electricity * 0.5 +     # kg CO2 per kWh
                diet * 1.5 +           # kg CO2 per meal
                shopping * 2.0         # kg CO2 per item
            )

            # Save to database
            query = """
            INSERT INTO users (username, transportation, electricity, diet, shopping,
total_footprint)
            VALUES (%s, %s, %s, %s, %s, %s)
            """
            data = (username, transportation, electricity, diet, shopping, total_emissions)
            cursor.execute(query, data)
            db.commit()

            result_label.config(text=f"Total Carbon Footprint: {total_emissions:.2f} kg
CO2")
            messagebox.showinfo("Success", "Data saved successfully!")
            load_data()  # Refresh the data display
        except ValueError:
            messagebox.showerror("Input Error", "Please enter valid numeric values.")

    # Function to load and display data from MySQL
    def load_data():
        cursor.execute("SELECT * FROM users")
        rows = cursor.fetchall()

        data_display.delete(1.0, tk.END)
        for row in rows:
            user_data = f"User: {row[1]}, Footprint: {row[6]:.2f} kg CO2, Date:
{row[7]}\n"
            data_display.insert(tk.END, user_data)

    # Create the Tkinter window
    window = tk.Tk()
    window.title("EcoTrack - Carbon Footprint Calculator")
    window.geometry("600x600")

    # Input fields
    tk.Label(window, text="Username:").pack()
    entry_username = tk.Entry(window)
    entry_username.pack()

    tk.Label(window, text="Transportation (km traveled):").pack()
```

```
        entry_transport = tk.Entry(window)
        entry_transport.pack()

        tk.Label(window, text="Electricity Usage (kWh):").pack()
        entry_electricity = tk.Entry(window)
        entry_electricity.pack()

        tk.Label(window, text="Diet (meals per day):").pack()
        entry_diet = tk.Entry(window)
        entry_diet.pack()

        tk.Label(window, text="Shopping (items purchased):").pack()
        entry_shopping = tk.Entry(window)
        entry_shopping.pack()

        # Calculate and Save button
        calculate_button = tk.Button(window, text="Calculate & Save",
command=calculate_and_save)
        calculate_button.pack()

        # Result label
        result_label = tk.Label(window, text="Total Carbon Footprint: --")
        result_label.pack()

        # Data display
        tk.Label(window, text="Stored User Data:").pack()
        data_display = tk.Text(window, height=10, width=70)
        data_display.pack()

        # Load data from the database
        load_data()

        # Run the application
        window.mainloop()
```

III. SUSTAINABLE DEVELOPMENT GOALS

EcoTrack directly supports SDG 13: Climate Action, which aims to take urgent action to combat climate change and its impacts. The app contributes to this goal by:

- Raising Awareness: Educating users about their carbon footprint and the environmental impact of everyday actions.
- Encouraging Action: Providing personalized recommendations for reducing emissions and promoting sustainable practices.
- Enabling Offsetting: Offering a platform for users to purchase verified carbon credits to offset their remaining emissions, supporting projects that contribute to climate resilience.

By empowering users to make informed decisions, EcoTrack helps foster a culture of sustainability and climate responsibility.

IV. PROGRAM/SYSTEM INSTRUCTIONS

Using the Program:

1. Input Fields:
    - Enter the following data in the input fields:
        - Username: Your name or identifier.
        - Transportation: Number of kilometers traveled in a day.
        - Electricity: Electricity usage in kilowatt-hours (kWh).
        - Diet: Number of meals consumed daily.
        - Shopping: Number of items purchased.
2. Calculate and Save:
    - Click the "Calculate & Save" button to:
        - Compute your total carbon footprint.
        - Save the data to the MySQL database.
3. View Results:
    - Your total carbon footprint (in kg CO2) will be displayed below the button.
    - Suggestions for reducing your footprint will appear in the "Suggestions" section.
4. View Stored Data:
    - The Stored User Data section will display all records saved in the database, including username, total footprint, and the timestamp.
5. Exit the Application:
    - Close the window to terminate the program.

Error Handling:

1. Invalid Input: If non-numeric values are entered in numeric fields, an error message will pop up.
2. Database Connection Issue: Ensure the MySQL service is running and your credentials are correct.
3. Missing Libraries: If a library is missing, install it using `pip`.

Customization Options:

1. Modify Calculation Formula: Update the formula in the `calculate_and_save` function to match your requirements.
2. Add New Features:
    - User authentication for personalized data.
    - Charts to visualize trends over time.
3. Integrate Additional Data Sources:
    - APIs for transportation or energy-specific emissions.