Assignment Report

Title: "Carbon Reporting Project in Python"

Name: Esha Agarwal GH1031345

University Name: Gisma University of Applied Science

Module Name: M602 Computer Programming

Professor: William Baker Morrison

Table of Contents:

Abstract

With growing environmental concerns, organizations are increasingly focusing on reducing their carbon footprint to meet sustainability goals. However, many lack accessible tools for monitoring, analyzing, and providing actionable insights to mitigate carbon emissions effectively.

The need for a user-friendly software solution arises to help clients input relevant data, track their carbon footprint, and receive tailored recommendations to reduce it. This tool should simplify complex environmental metrics, enhance decision-making, and promote eco-conscious behaviours among users.

To tackle this issue, a Carbon Footprint Monitoring Tool will be developed. This program will leverage Python to create an efficient, interactive system for data input, report generation, and implementable suggestions for clients.

Problem Statement

Develop a Carbon Footprint Monitoring Tool that allows users to input data and generate reports providing suggestions on how to reduce their carbon footprint.

Approach and Methodology

To achieve the objectives of the Carbon Footprint Monitoring Tool, a structured approach is used, combining data collection, calculation methodologies, and user-driven insights. The strategy is executed using the following key components:

1. **Approach:**
    - **Tool Selection**:
        - For the development, I have used **Python** as the core programming language because I find it easier to use compared to C++. Python offers better readability of code and intuitiveness of syntax, making it a preferred choice for my project.
        - To build the user interface, I have used **Streamlit**, as it pairs well with Python for creating web applications. Streamlit's simplicity and

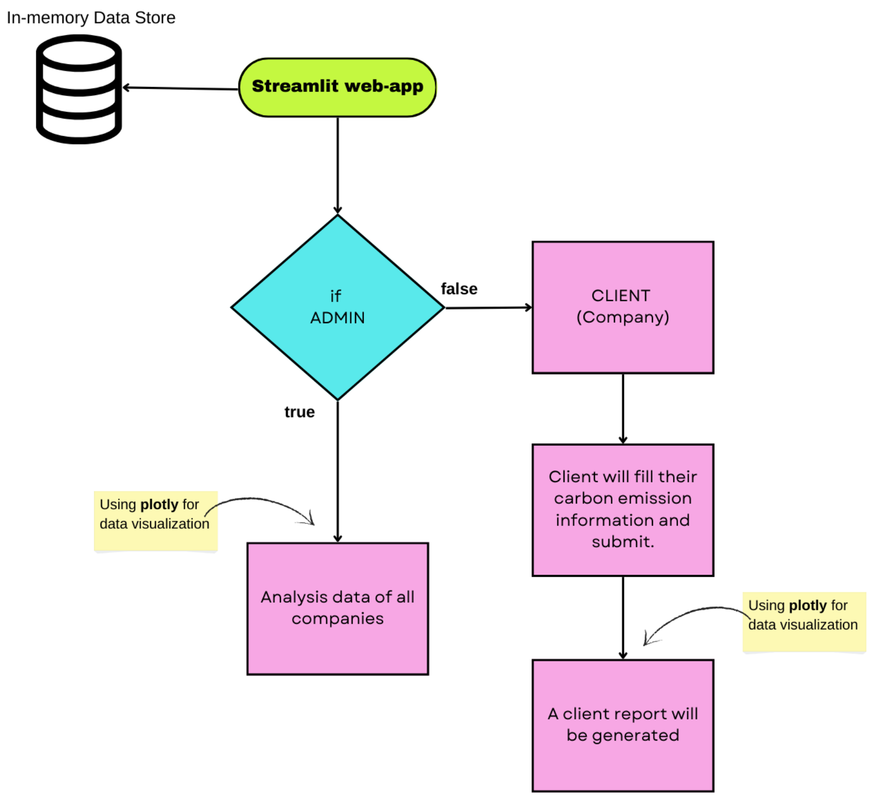
    flexibility enhance the user experience, from a simple-to-use tool up to rapid building of complex interactive interfaces.

    - I have used **Plotly** for visualization, displaying reports in the form of interactive charts. I chose Plotly because it allows users to explore and understand data in a dynamic way. It also offers a wide range of chart types with a high level of customization.

- **Carbon Footprint Calculation:**
    - The tool calculates the carbon footprint by using pre-defined emission factors based on the user's input. These formulas are designed to process various types of data collected (e.g., energy usage, travel, waste) and convert them into measurable carbon emissions.

2. **Methodology:**
    - **Data Collection**: Data will be gathered through direct user input.
    - **Calculation Process**: Once the data is collected, the monitoring tool will use a predefined set of carbon emission factors to calculate the total carbon footprint.
    - **Report Generation**: The monitoring tool will then analyze the results and generate visual report that clearly illustrate the company's carbon footprint and provide suggestions for reducing emissions.
    - **Admin Section**: The admin section aggregates the carbon footprint data from all companies that have calculated their emissions. This data is displayed using a bar graph, allowing comparisons between different companies' footprints and helping to identify trends across various organisations.

System Design


In-memory Data Store
Streamlit web-app
if ADMIN
false
true
CLIENT (Company)
Client will fill their carbon emission information and submit.
Using plotly for data visualization
Analysis data of all companies
Using plotly for data visualization
A client report will be generated

Data Flow Diagram: This is the data flow diagram for the project.

- **User Interface (Streamlit Web App):**
  - The client or admin accesses the monitoring tool through a web interface built with Streamlit. There are two separate tabs: one for **Admin** and one for **Client (Company User)**. The user selects the appropriate tab based on their role.
- **Decision Point (Tabs):**
  - If the user selects the **Admin** tab, the system will analyze and visualize the data for all companies. It uses **plotly** to display the graph of the data.
  - If the user selects the **Client (Company User)** tab, they are prompted to input their **carbon emission data** and submit it.
- **Data Analysis and Visualization**:
  - For the **Admin**, the system performs a comprehensive analysis of all companies' data and generates visualizations using **Plotly**.

    - For the **Client (Company User)**, the system analyzes the submitted carbon emissions data and generates personalized visualizations for that company.
- **Report Generation**:
    - After the analysis, the system generates a **report** for the client, summarizing the carbon emissions data and providing insights.

Key Components:

- **Streamlit**: A Python library used to create the web interface with separate tabs for Admin and Client.
- **Plotly**: A library used to generate interactive visualizations.
- **In-memory Data Store**: A temporary data store used for holding data during processing.

Development Process

1. Overview of Implementation

The development began with implementing the core formulae in Jupyter Notebook to test calculations using sample data. This step was essential for understanding the formulae's behavior and provided a clear roadmap for integrating these calculations into the web application.

2. Environment Setup

The project is developed using Python 3.9.6 in Visual Studio Code and hosted on HuggingFace Spaces. The primary libraries used are Jupyter Notebook (for initial formulation), Streamlit (for user interface), and Plotly (for data visualization). These dependencies were installed using the `pip` package manager within a virtual environment created using Python's `venv` module.

3. Code Structure

Main Project Directory: `Carbon Reporting Project`

The root folder contains the primary project files and subfolders:

1. `.streamlit/`

- Contains the configuration file to customize the Streamlit theme.
- The theme has been set to light mode by default, overriding Streamlit's built-in option for user selection between light and dark modes.

2. `carbon_calculator/`
    - This folder contains the entire web application and is further divided into roles:
        - `admin/`
            - `admin_view.py`: The UI and logic for admin functionalities are implemented here.
        - `user/`
            - `user_view.py`: The main rendering file for user interactions.
            - `generate_suggestions.py`: Contains logic for suggesting ways to reduce carbon emissions.
            - `validate_inputs.py`: Handles input validations to ensure data accuracy.
        - `calculator.py`
            - Contains functions for carbon emission calculations.
        - `styles.py`
            - Contains custom stylings for the web app.
        - `utils.py`
            - Contains utility functions (e.g., data download, PDF/CSV report generation).
3. `Icons/`
    - Stores all the icons used throughout the project for UI elements.
4. `learning/`
    - Contains the Jupyter Notebook `formulation.ipynb`, used during the initial stages to experiment with calculations and test dummy data.
5. `app.py`
    - The main renderer file that integrates and calls `admin_view.py` and `user_view.py` to launch the web application.

6. `requirements.txt`
    - Lists all the dependencies required for the project, ensuring a consistent setup across environments.
7. `README.md`
    - Provides a comprehensive overview of the project, including its purpose, features, and setup instructions.

```
Carbon Reporting Project/
│
├── .streamlit/
│   └── config.toml
│
├── carbon_calculator/
│   ├── admin/
│   │   └── admin_view.py
│   ├── user/
│   │   ├── generate_suggestions.py
│   │   ├── user_view.py
│   │   └── validate_inputs.py
│   ├── calculator.py
│   ├── styles.py
│   ├── utils.py
│
├── Icons/
│   └── [Icons used in the project]
│
├── learning/
│   └── formulation.ipynb
│
├── app.py
├── README.md
└── requirements.txt
```

4. Core Features Implemented

1. **Company Information Input**:

- Users input - essential company data such as the company name, energy usage, waste, and business travel details.
- **Code**: (the following code snippet is from `user_view.py`)

```
st.text_input("Company Name", key="company_name")
st.number_input("Electricity Bill", min_value=0.0, key="electricity_bill")
st.number_input("Natural Gas Bill", min_value=0.0, key="natural_gas_bill")
st.number_input("Fuel Bill", min_value=0.0, key="fuel_bill")
st.number_input("Waste per Month (kg)", min_value=0.0, key="waste_per_month")
st.number_input("Recycling Percentage", min_value=0.0, max_value=100, value=30, key="recycling_percent_input")
st.number_input("Business Travel Distance (km)", min_value=0.0, key="distance_km")
st.number_input("Fuel Efficiency (km/l)", min_value=0.1, value=8.0, key="fuel_efficiency")
```

2. **Carbon Footprint Calculation**:

- The tool calculates CO2 emissions based on energy usage, waste generation, and business travel.
  - Energy usage is multiplied by predefined emission factors for each type of energy.
  - Waste emissions are calculated by considering the recycling percentage.
  - Business travel emissions are computed using distance traveled and fuel efficiency.
- **Code**: (the following code snippet is from `calculator.py`)

```
def calculate_CO2_from_energy_usage(electricity_bill, natural_gas_bill, fuel_bill):
    CO2_from_electricity_usage = electricity_bill * 12 * 0.0005
    CO2_from_natural_gas_usage = natural_gas_bill * 12 * 0.0053
    CO2_from_fuel_usage = fuel_bill * 12 * 2.32
    return CO2_from_electricity_usage + CO2_from_natural_gas_usage + CO2_from_fuel_usage

def calculate_CO2_from_waste(waste_per_month, recycling_percent):
    return waste_per_month * 12 * 0.57 - recycling_percent

def calculate_CO2_from_business_travel(distance_km, fuel_efficiency):
    return distance_km * 1 / fuel_efficiency * 2.31
```

3. **Results Visualization**:
    - After calculating the carbon footprint, the tool visualizes the data using interactive charts and tables.
    - **Code**: (the following code snippet is from `user_view.py`)

```
import plotly.graph_objects as go

# Create pie chart using Plotly
fig = go.Figure(
    data=[
        go.Pie(
            labels=[
                "Energy Usage",
                "Waste Generated",
                "Business Travel",
            ],
            values=[
                CO2_from_energy_usage,
                CO2_from_waste,
                CO2_from_business_travel,
            ],
            hole=0.3,
        )])
st.plotly_chart(fig)
```

4. **Download Options**:
   - Users can download both the carbon footprint chart as a PDF and the data as a CSV file for further analysis or record-keeping.
   - **Code**: (the following code snippet is from `utils.py`)

```
def get_image_download_link(fig, filename, icon, tex
t):
    buf = BytesIO()
    fig.write_image(buf, format="pdf")
    buf.seek(0)
    b64 = base64.b64encode(buf.read()).decode()
    href = f"""
    <a href="data:application/pdf;base64,{b64}" downl
oad="{filename}.pdf">{text}</a>
    """
    return href

def get_csv_download_link(df, filename, icon, text):
    csv = df.to_csv(index=False)
    b64 = base64.b64encode(csv.encode()).decode()
    href = f"""
    <a href="data:file/csv;base64,{b64}" download="{f
ilename}">{text}</a>
    """
    return href
```

5. **Suggestions for Reducing Carbon Footprint**:
   - Based on the user's data, the system provides suggestions for reducing carbon emissions, such as using renewable energy, recycling more, or reducing business travel.
   - **Code**: (the following code snippet is from `generate_suggestions.py`)

```
def generate_suggestions(energy_usage, waste, busines
s_travel, electricity_bill, natural_gas_bill, fuel_bi
ll, waste_per_month, recycling_percent):
    suggestions = {"energy": [], "waste": [], "trave
l": []}

    # Energy Suggestions
    if electricity_bill > 1000:
        suggestions["energy"].extend(["Install LED li
ghting", "Conduct an energy audit"])

    # Waste Suggestions
    if recycling_percent < 50:
        suggestions["waste"].extend(["Increase recycl
ing rate", "Implement a recycling program"])

    # Travel Suggestions
    if business_travel > 1000:
        suggestions["travel"].extend(["Promote virtua
l meetings", "Consider carbon offsetting"])

    return suggestions
```

6. **Input Validation**:
    - The inputs are validated to ensure all necessary fields are filled out. If any required fields are missing or incorrect, error messages are displayed.
    - **Code**: (the following code snippet is from `validate_inputs.py`)

```
def validate_inputs():
    is_valid = True
    error_messages = []

    # Company name validation
    if not st.session_state.company_name:
        is_valid = False
        error_messages.append("Please enter a company
name.")
```

```
    # Energy usage, Waste and travel validations
    if st.session_state.electricity_bill == 0:
        is_valid = False
        error_messages.append("Please enter the elect
ricity bill.")

    if st.session_state.waste_per_month == 0:
        is_valid = False
        error_messages.append("Please enter the amoun
t of waste generated.")

    if st.session_state.distance_km == 0:
        is_valid = False
        error_messages.append("Please enter the dista
nce traveled.")

    return is_valid, error_messages
```

7. **Admin Dashboard for Monitoring**:
    - The admin dashboard allows the administrator to monitor the carbon footprint data from all companies, offering key metrics like total companies, reports, and emissions.
    - **Code**: (the following code snippet is from `admin_view.py`)

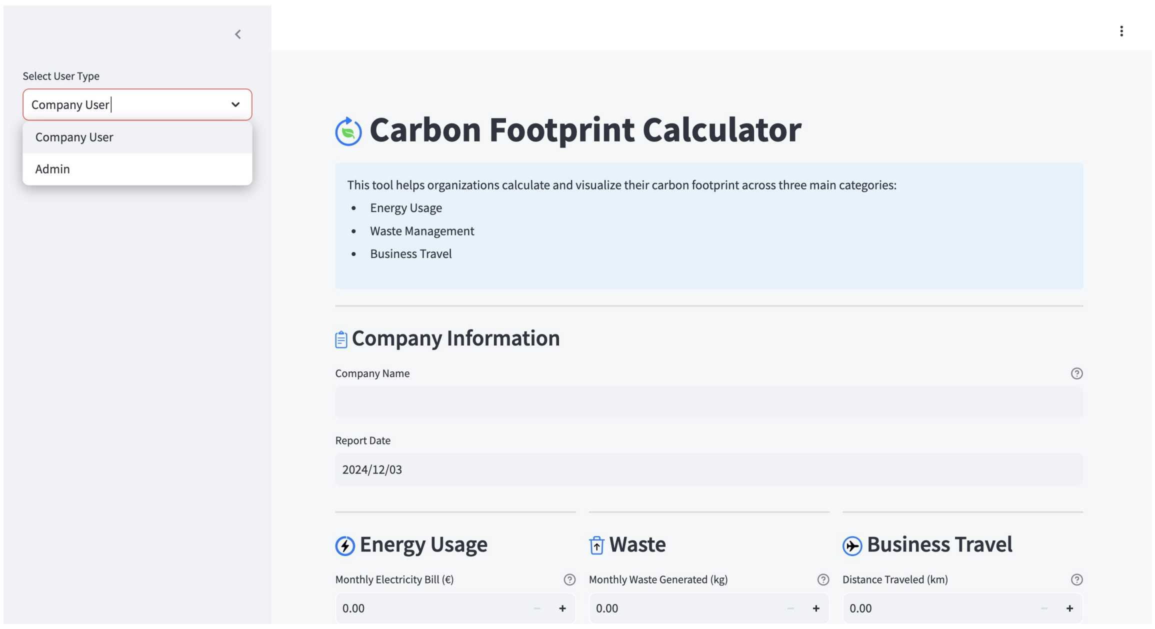
```
def admin_view():
    df = pd.DataFrame(st.session_state.companies_dat
a)
    st.metric("Total Companies", len(df["company_nam
e"].unique()))
    st.metric("Total Reports", len(df))
    st.metric("Total Emissions", f"{df['total'].sum
():.2f} kgCO2")

    fig = px.bar(
        df, x="company_name", y=["energy_usage", "was
te", "business_travel"], barmode="stack"
```

```
    )
    st.plotly_chart(fig)
```

Results

- **Company User view:** Company Information, Carbon Footprint Results, Emission Reduction Suggestions, Input Validations Errors



Select User Type

Company User

Company User

Admin

Carbon Footprint Calculator

This tool helps organizations calculate and visualize their carbon footprint across three main categories:

- Energy Usage
- Waste Management
- Business Travel

Company Information

Company Name

Report Date

2024/12/03

Energy Usage

Monthly Electricity Bill (€)

0.00

Waste

Monthly Waste Generated (kg)

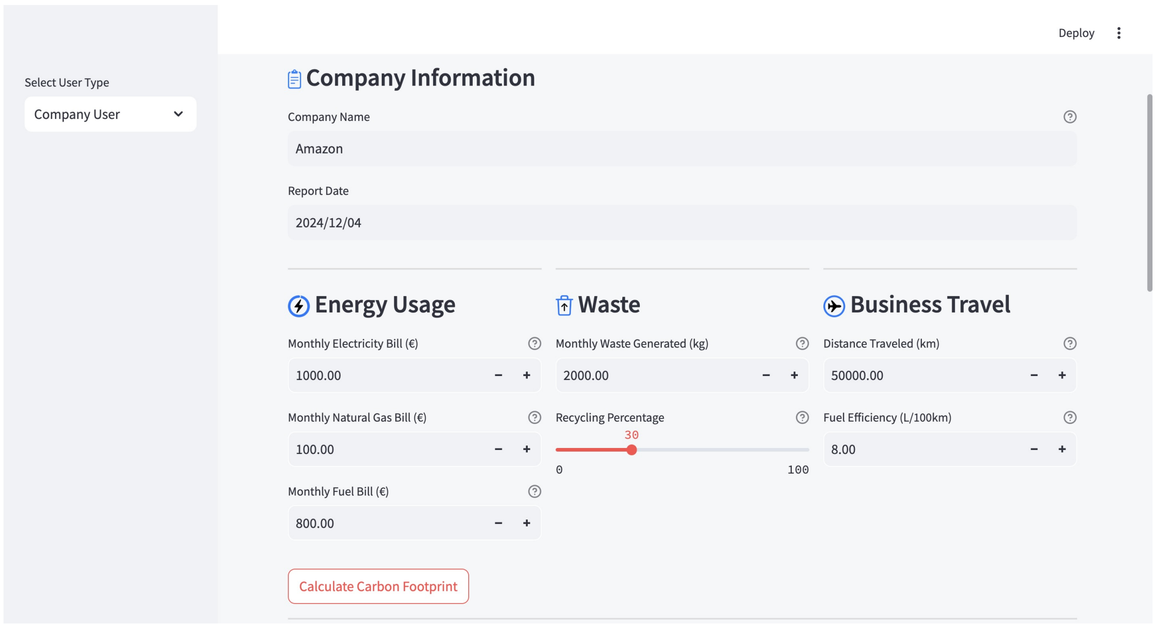
0.00

Business Travel

Distance Traveled (km)

0.00

Company Information



Select User Type

Company User

Company Information

Company Name

Amazon

Report Date

2024/12/04

Energy Usage

Monthly Electricity Bill (€)

1000.00

Monthly Natural Gas Bill (€)

100.00

Monthly Fuel Bill (€)

800.00

Waste

Monthly Waste Generated (kg)

2000.00

Recycling Percentage

30

0 100

Business Travel

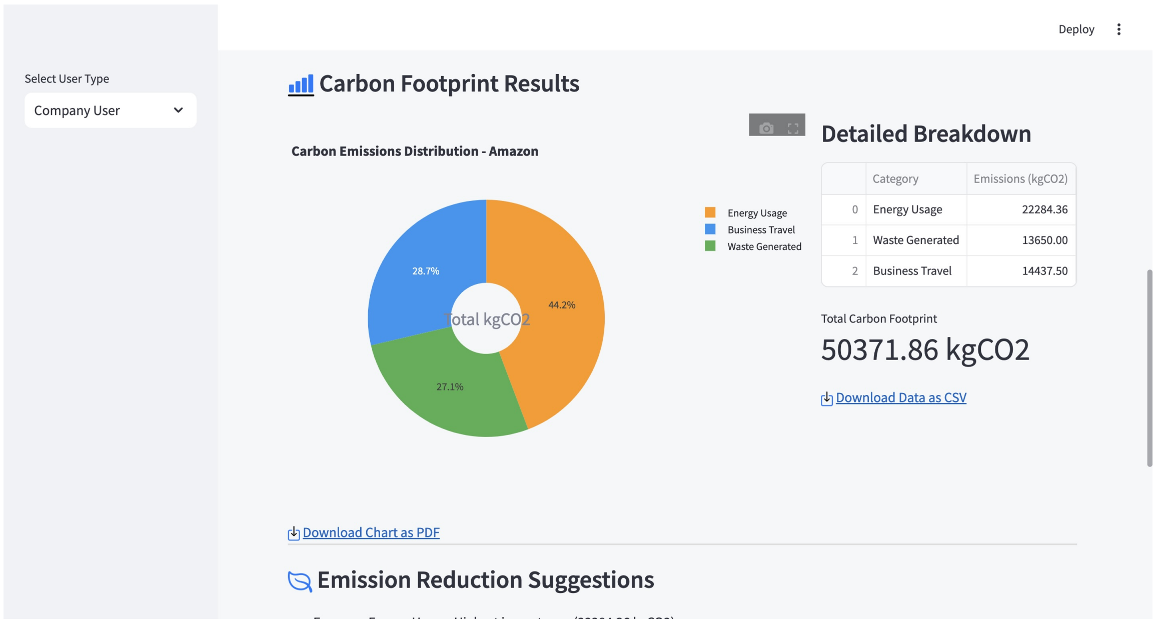
Distance Traveled (km)

50000.00

Fuel Efficiency (L/100km)

8.00

Company Information



Carbon Footprint Results

Carbon Emissions Distribution - Amazon

Detailed Breakdown

	Category	Emissions (kgCO2)
0	Energy Usage	22284.36
1	Waste Generated	13650.00
2	Business Travel	14437.50

Total Carbon Footprint

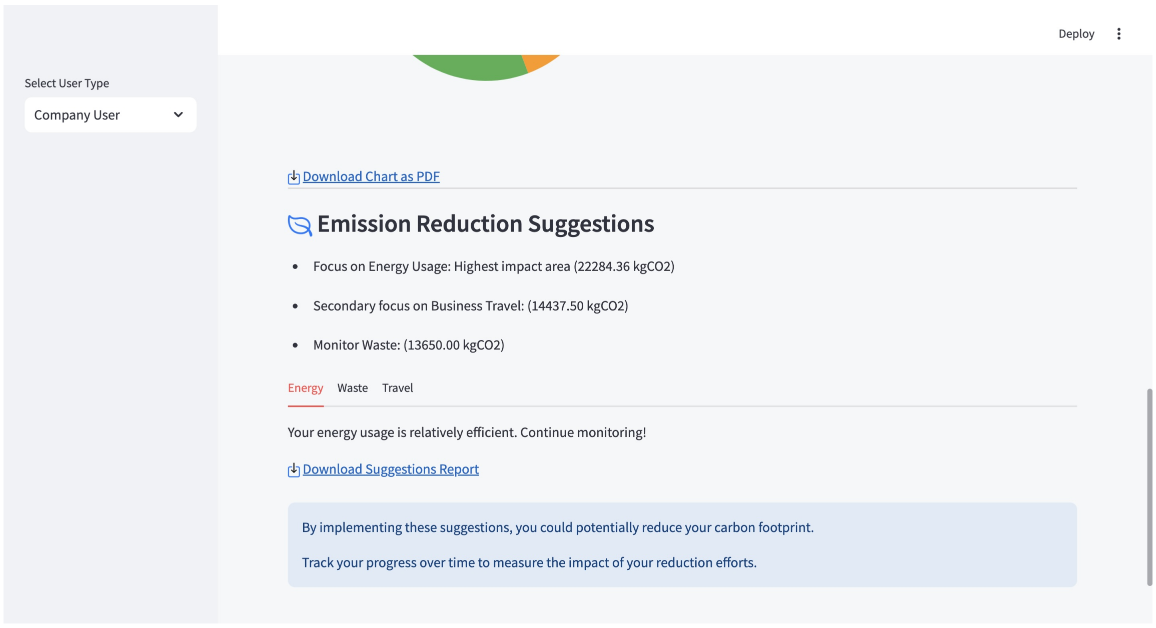
50371.86 kgCO2

Download Data as CSV

Download Chart as PDF

Emission Reduction Suggestions

Carbon Footprint Results



Download Chart as PDF

Emission Reduction Suggestions

- Focus on Energy Usage: Highest impact area (22284.36 kgCO2)
- Secondary focus on Business Travel: (14437.50 kgCO2)
- Monitor Waste: (13650.00 kgCO2)

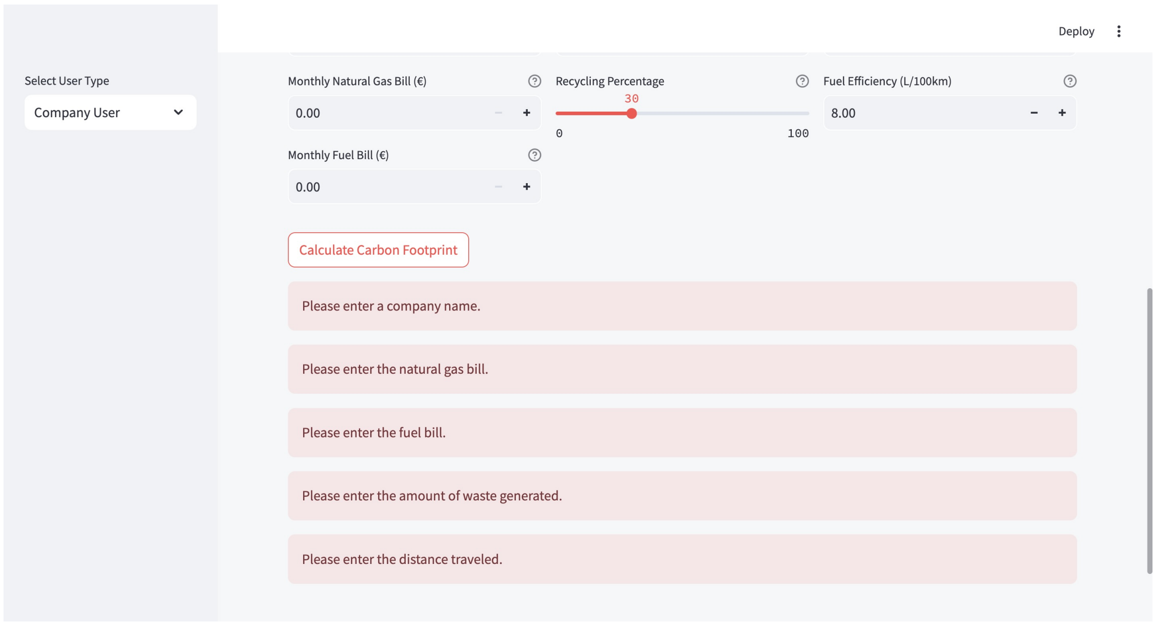
Your energy usage is relatively efficient. Continue monitoring!

Download Suggestions Report

By implementing these suggestions, you could potentially reduce your carbon footprint.

Track your progress over time to measure the impact of your reduction efforts.

Emission Reduction Suggestions



Please enter a company name.

Please enter the natural gas bill.

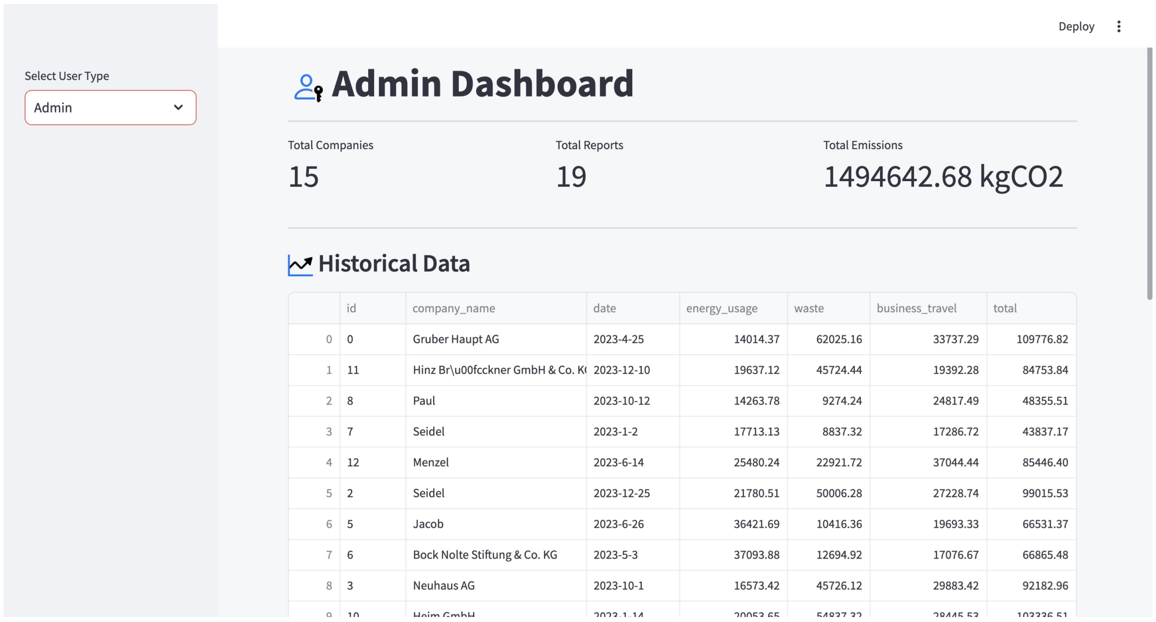
Please enter the fuel bill.

Please enter the amount of waste generated.

Please enter the distance traveled.

Input Validations Errors

- **Admin View:** Admin Dashboard, Company Comparison



Admin Dashboard

Total Companies 15

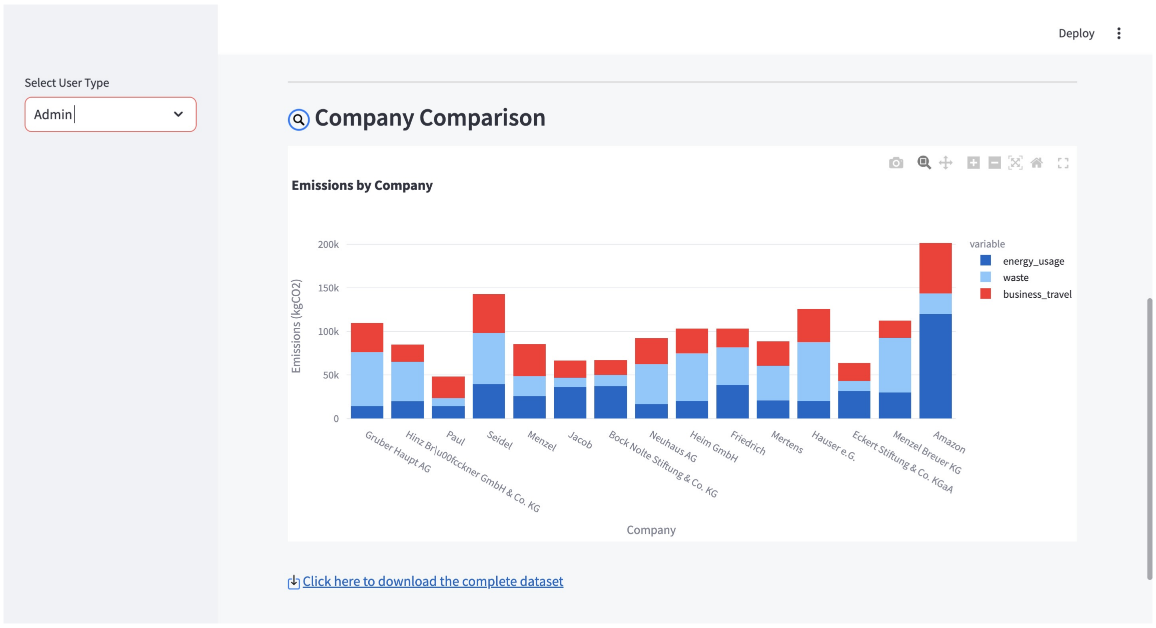
Total Reports 19

Total Emissions 1494642.68 kgCO2

Historical Data

	id	company_name	date	energy_usage	waste	business_travel	total
0	0	Gruber Haupt AG	2023-4-25	14014.37	62025.16	33737.29	109776.82
1	11	Hinz Br\u00fcckner GmbH & Co. K(	2023-12-10	19637.12	45724.44	19392.28	84753.84
2	8	Paul	2023-10-12	14263.78	9274.24	24817.49	48355.51
3	7	Seidel	2023-1-2	17713.13	8837.32	17286.72	43837.17
4	12	Menzel	2023-6-14	25480.24	22921.72	37044.44	85446.40
5	2	Seidel	2023-12-25	21780.51	50006.28	27228.74	99015.53
6	5	Jacob	2023-6-26	36421.69	10416.36	19693.33	66531.37
7	6	Bock Nolte Stiftung & Co. KG	2023-5-3	37093.88	12694.92	17076.67	66865.48
8	3	Neuhaus AG	2023-10-1	16573.42	45726.12	29883.42	92182.96
9	10	Heim GmbH	2023-1-14	20053.65	54837.32	28445.53	103336.51

Admin Dashboard



Company Comparison

Challenges

Several challenges were encountered during this project:

- **Custom Icons**: Incorporating custom icons as SVGs from a local path was tricky. To solve this, I first uploaded the SVG files and then used their corresponding links.
- **Custom CSS**: Streamlit does not support separate `styles.css` files, so I created a dedicated node to manage all custom styling.
- **UI Design**: Ensuring the UI layout was both aesthetic and easy to read required careful planning and adjustments.
- **Input Validation**: Validating user inputs to ensure they fell within the correct ranges was essential for maintaining accuracy and functionality.

Summary

This project highlights the growing need for accessible and user-friendly tools to address environmental sustainability challenges. Organizations increasingly prioritize reducing their carbon footprint, but often face obstacles due to the complexity of tracking and analyzing emissions. The proposed solution bridges this gap by providing an intuitive platform that enables users to input data, monitor their carbon footprint, and receive tailored recommendations for improvement. By simplifying complex environmental metrics into actionable insights, this tool empowers organizations to achieve their sustainability goals more effectively, making a significant contribution toward a greener future.

Future Work

Looking ahead, the project could be further enhanced by developing a separate REST API backend and a standalone frontend. Implementing this architecture using the MERN stack (MongoDB, Express.js, React.js, and Node.js) would allow for improved scalability, modularity, and flexibility in the development process.

Additional Suggestions:

1. **User Authentication and Roles**: Add user authentication with role-based access control, allowing for features tailored to individual users or organizations.
2. **Advanced Analytics Dashboard**: Integrate an interactive dashboard with detailed visualizations to provide deeper insights into carbon footprint trends, comparisons, and projections.
3. **Available to Companies as a Service**: This API can be integrated into internal tools, offering continuous monitoring and analysis, with flexible monetization options such as per-token usage or licensing.
4. **Gamification Elements**: Introduce gamified elements, such as rewards for achieving sustainability milestones, to incentivize user participation.

These advancements will not only improve the usability and functionality of the application but also make it a comprehensive tool for addressing environmental sustainability.

GitHub Link

```
https://github.com/eshagarwal/Carbon-Reporting-Project
```