Gisma University of Applied Sciences

Assessment Submission Form

Student Number (If this is group work, please include the student numbers of all group participants)	Jithin Raghunathan (GH1032193)
Assessment Title	**Carbon FootPrint Calculator Project(M602A Computer Programming)**
Module Code	**M602A**
Module Title	Computer Programming
Module Tutor	Professor William Baker Morrison
Date Submitted	18-Dec-24

Carbon FootPrint Calculator Project(M602A Computer Programming)

Introduction:

The Carbon Footprint Calculator is a web based tool primarily built using the Python framework Flask and HTML5,CSS, Bootstrap and javascript. This Project offers a user interface where company representatives can signup and input their daily activities like travel, energy usage and waste generation etc and receive a comprehensive report regarding their carbon Footprint. Also, there is a summary page where they can see their individual performance as well as other company's performance and also they can compare the performance based on different trends like energy, travel etc. Through this project company owners can gain valuable inputs to improve their overall carbon efficiency which in turn can help in producing an eco-friendly world.

This project is developed as a part of M602A Computer Programming (WS0124) course aiming to create a practical programming experience.

Problem statement:

Recently there are growing reports of increasing carbon emission on global level leading to increasing global warming.Increase in development and growth of technology has only caused further addition to this problem.
It is important for companies to have the opportunity to monitor their carbon footprint and also get suggestions in real time to counter those effects of carbon emission. A light weight application designed for this purpose can be highly beneficial.

Solution:

This project when used by companies can be very valuable in assisting them to mitigate the effects of carbon emissions by providing them reports which have their categories like energy ,waste and travel footprints highlighted in a color based on their magnitude.Also, they can edit the report in real time and get a newly generated report and recommendations on the spot.Also they can see and compare their companies performance against other companies based on different trends like energy, travel and waste footprints

Project Stakeholder:

This project is developed as an initiative of environmental organizations to create awareness among companies across the globe who can register through this application and get a detailed analysis regarding their overall carbon emission in the form of bar charts.

Project Overview

Carbon Emission Calculator is a tool which is designed to help companies monitor their carbon footprint based on Total Footprint. **Total Footprint** is divided into 1. **Energy Footprint** 2. **Waste Footprint** and **3. Travel Footprint**

1.Energy footprint can be subcategorized into

    1.1 Electricity Bill
    1.2 Fuel Bill
    1.3 Natural gas Bill

2.Waste footprint can be subcategorized into

    2.1 Waste Generated
    2.2 Recycling Percentage

3.Travel footprint can be subcategorized into

    3.1 Distance Travelled
    3.2 Fuel Efficiency of Vehicles

SourceLinks:

GithubRepo:https://github.com/jithin6384/carbonEmissionCalculatorTool

Git Clone url:

git@github.com:jithin6384/carbonEmissionCalculatorTool.git

You can set up the application in your local or just enter the following live url to use the application in real time

LiveUrl:

1. https://carbonemissioncalculatortool.onrender.com/register deployed using https://dashboard.render.com/
2. http://198.199.83.64:8000/register deployed using digital ocean droplet

Instructions to set the application in Windows, Mac and Linux

1.requirements
 Python 3.8 should be installed in respective machine
 Git should be installed
 Git bash app installed terminal for Windows
 Terminal for Linux or Mac
 VS code , Jupyter Notebook, Atom

2. Clone the repository

```
git clone  git@github.com:jithin6384/carbonEmissionCalculatorTool.git
cd carbonEmissionCalculatorTool
```

3. Setup Local requirement
 On windows

```
python -m venv env
env/Scripts/activate (to activate the virtual enviroment)
```

 On Linux/MacOs

```
python -m venv env
source  env/bin/activate (to activate the virtual enviroment)
```

4. Install requirements(from requirements.txt)

```
pip install -r requirements.txt
```

5. Initialize the database:

```
flask db init
flask db migrate
flask db upgrade
```

6.set the FLASK_APP

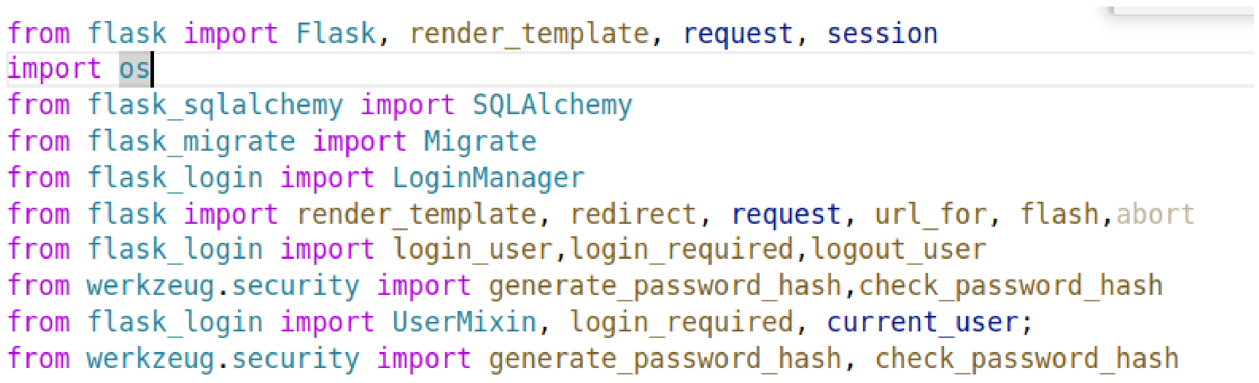
```
export FLASK_APP=app.py (for Linux/MacOS)
set FLASK_APP=app.py (for Windows)
```

7. Run the app in Local
   python3 app.py

Code structure:

app.py (Applications entry point)

Import requirements



```
from flask import Flask, render_template, request, session
import os
from flask_sqlalchemy import SQLAlchemy
from flask_migrate import Migrate
from flask_login import LoginManager
from flask import render_template, redirect, request, url_for, flash,abort
from flask_login import login_user,login_required,logout_user
from werkzeug.security import generate_password_hash,check_password_hash
from flask_login import UserMixin, login_required, current_user;
from werkzeug.security import generate_password_hash, check_password_hash
```

1. from flask import Flask, render_template, request, session
   flask is required to create the application.
   render_template is used to render html files for user interface
   request is used capture incoming HTTP request
   session is used to store user specific data

2.import os
   Provides operation level functionalities like environment variables and file paths

3. Flask_sqlalchemy for ORM related features for database management

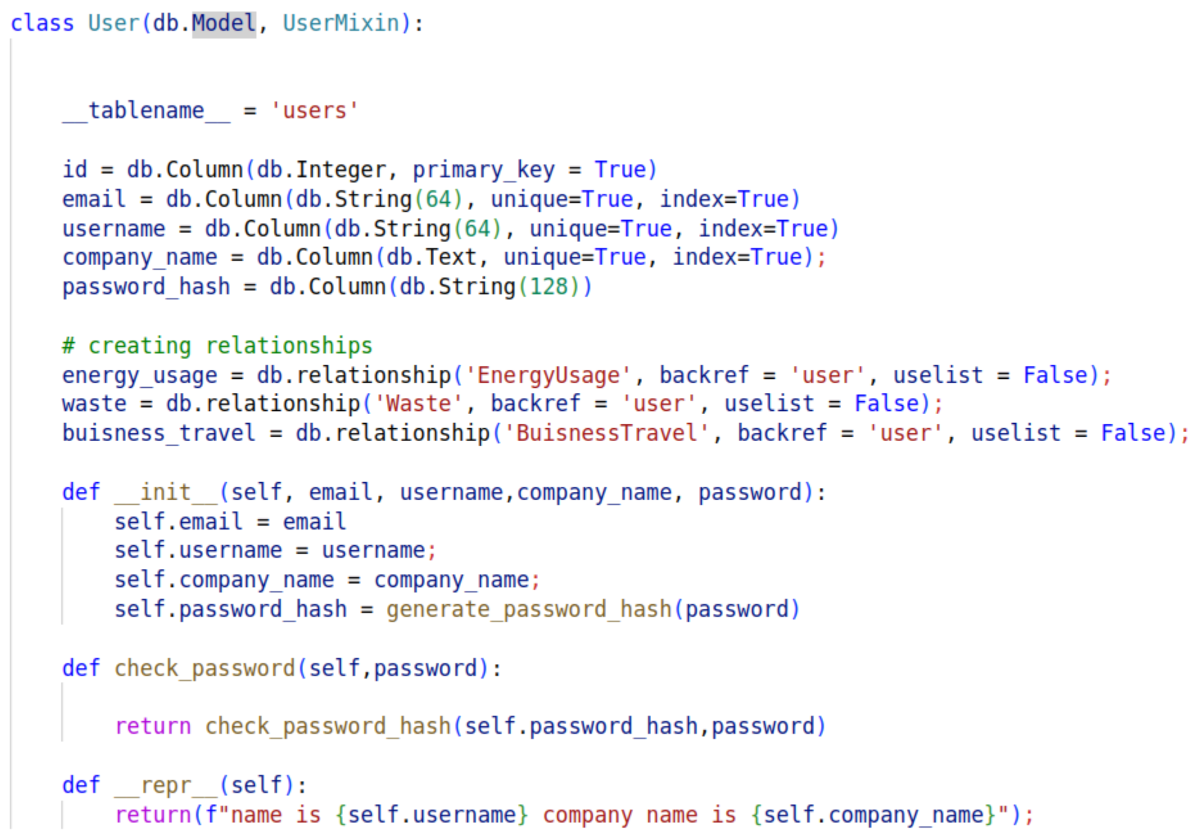
4.flask_migrate to create migrations for database

5.flask_login to manage user authentication

6.werkzeug.security for password verification

Models

1. **User model**



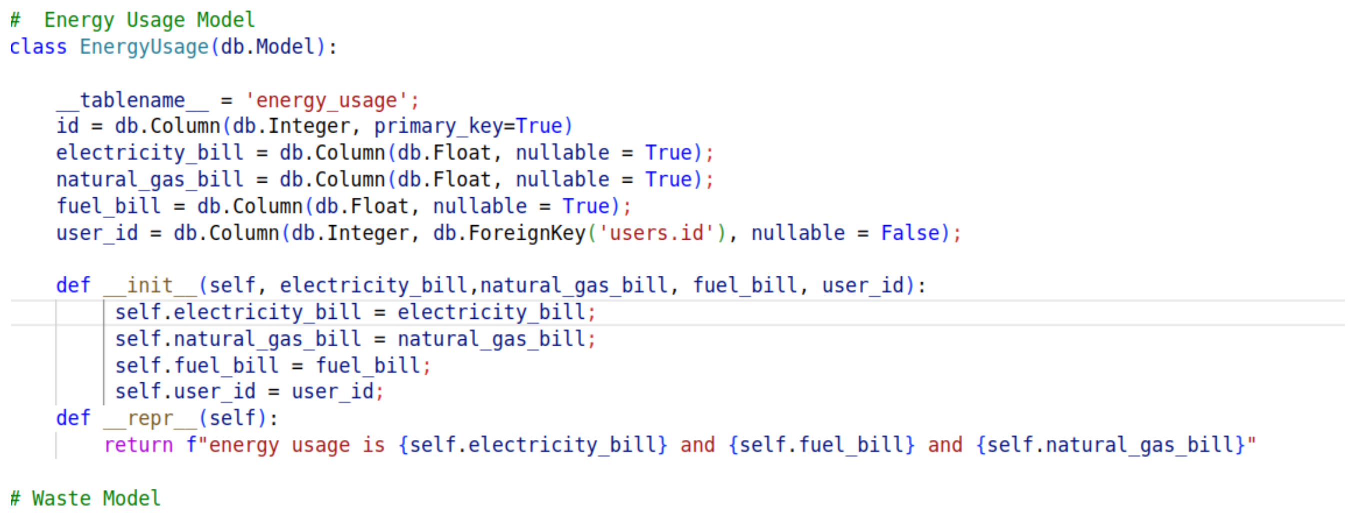
```
class User(db.Model, UserMixin):


    __tablename__ = 'users'

    id = db.Column(db.Integer, primary_key = True)
    email = db.Column(db.String(64), unique=True, index=True)
    username = db.Column(db.String(64), unique=True, index=True)
    company_name = db.Column(db.Text, unique=True, index=True);
    password_hash = db.Column(db.String(128))

    # creating relationships
    energy_usage = db.relationship('EnergyUsage', backref = 'user', uselist = False);
    waste = db.relationship('Waste', backref = 'user', uselist = False);
    buisness_travel = db.relationship('BuisnessTravel', backref = 'user', uselist = False);

    def __init__(self, email, username,company_name, password):
        self.email = email
        self.username = username;
        self.company_name = company_name;
        self.password_hash = generate_password_hash(password)

    def check_password(self,password):

        return check_password_hash(self.password_hash,password)

    def __repr__(self):
        return(f"name is {self.username} company name is {self.company_name}");
```

User Model represents the individual users who are clients and they must have following attributes namely
1.id (primary_key): gives unique representation to users
2.Username : gives unique username for different users for identification
3.Email: Another unique identifier which helps location individual user during authentication
4.Company name: Name of users company
5.Password : password for authentication
Their respective data types are defined under tablename.
This basically creates ORM between User object and user table in database to store and access individual users

Relationships
  User have one-to-one relationship with EnergyUsage, Waste and BusinessTravel to connect with their respective footprint data

__init__() acts as initializer and check_password is used password verification during authentication

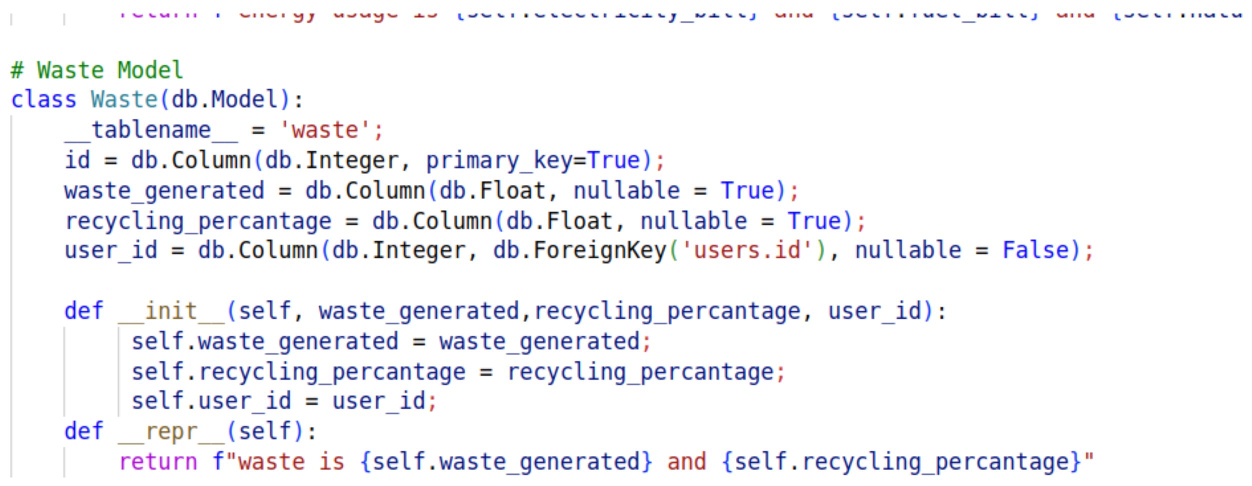
2.**EnergyUsage Model**



```
#  Energy Usage Model
class EnergyUsage(db.Model):

    __tablename__ = 'energy_usage';
    id = db.Column(db.Integer, primary_key=True)
    electricity_bill = db.Column(db.Float, nullable = True);
    natural_gas_bill = db.Column(db.Float, nullable = True);
    fuel_bill = db.Column(db.Float, nullable = True);
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable = False);

    def __init__(self, electricity_bill,natural_gas_bill, fuel_bill, user_id):
         self.electricity_bill = electricity_bill;
         self.natural_gas_bill = natural_gas_bill;
         self.fuel_bill = fuel_bill;
         self.user_id = user_id;
    def __repr__(self):
        return f"energy usage is {self.electricity_bill} and {self.fuel_bill} and {self.natural_gas_bill}"

# Waste Model
```

Energy Usage Model tracks the energy usage of company in the form of electricity natural gas and fuel bill respectively
  This has following attributes
  1.id acts as unique identifier for all energy usage objects
  2.electricity_bill represents the electricity bill for a particular user ie company owner
  3.natural_gas_bill represents natural gas bill per user
  4.fuel_bill represents fuel bill
  5.user_id foreign key linking energy usage to user model
   __init__() for initializing the energy usage object
  __repr__() string representation for debugging

3. **Waste Model**



```
        return f"energy usage is {self.electricity_bill} and {self.fuel_bill} and {self.natu

# Waste Model
class Waste(db.Model):
    __tablename__ = 'waste';
    id = db.Column(db.Integer, primary_key=True);
    waste_generated = db.Column(db.Float, nullable = True);
    recycling_percantage = db.Column(db.Float, nullable = True);
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable = False);

    def __init__(self, waste_generated,recycling_percantage, user_id):
        self.waste_generated = waste_generated;
        self.recycling_percantage = recycling_percantage;
        self.user_id = user_id;
    def __repr__(self):
        return f"waste is {self.waste_generated} and {self.recycling_percantage}"
```

The Waste model tracks the waste generated for a specific user. It has following attributes

Id: primary identifier

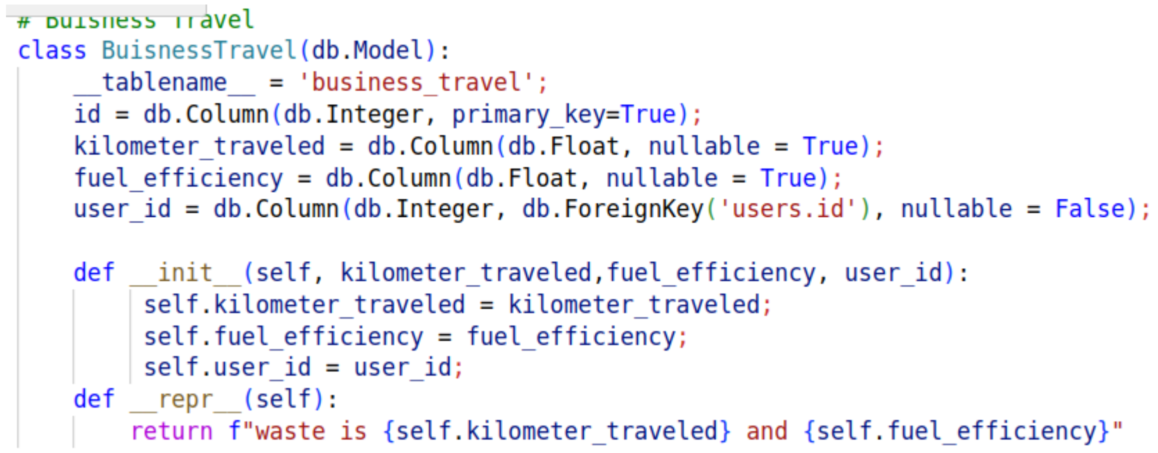
Waste_generated: to calculate the exact waste generated in the process

Recycling percentage: calculates the recycling of waste in percentage

User_id: as foreign key to connect with user model

__init__() and __repr__() are used for initialization and debugging purpose

4. **BusinessTravel Model**



```
# Buisness Travel
class BuisnessTravel(db.Model):
    __tablename__ = 'business_travel';
    id = db.Column(db.Integer, primary_key=True);
    kilometer_traveled = db.Column(db.Float, nullable = True);
    fuel_efficiency = db.Column(db.Float, nullable = True);
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable = False);

    def __init__(self, kilometer_traveled,fuel_efficiency, user_id):
         self.kilometer_traveled = kilometer_traveled;
         self.fuel_efficiency = fuel_efficiency;
         self.user_id = user_id;
    def __repr__(self):
        return f"waste is {self.kilometer_traveled} and {self.fuel_efficiency}"
```

Business travel model tracks the distance travelled and fuel efficiency for business purposes.
It has following attributes
Id: primary key
Kilometers_traveled: tracks the distance traveled by employees(user represents the entire employee set of their company)
Fuel efficiency: the fuel efficiency of vehicles used by company
User_id: foreign key to connect to User model

DB Schema Setup

User Table

1.Table name: users

Column_names

1.1 id: Integer (Primary Key)

1.2 email: String (64, Unique)

1.3 username: String (64, Unique)

1.4 company_name: Text (Unique)

1.5 password_hash: String (128)

Relationships:

One-to-One: energy_usage

One-to-One: waste

One-to-One: business_travel

Energy Usage Table

2. Table Name: energy_usage

Column names

2.1 id: Integer (Primary Key)

2.2 electricity_bill: Float (nullable)

2.3 natural_gas_bill: Float (nullable)

2.4 fuel_bill: Float (nullable)

2.5 user_id: Integer (foreign Key referring to users table)

Relationships:

One-to-One: connected to users via user_id as foreign key

Waste Table

3.Table Name: energy_usage

Column names

3.1 id: Integer (primary Key)

3.2 waste_generated: Float (nullable)

3.3 recycling_percentage: Float (nullable)

3.4 user_id: Integer (Foreign Key referencing users.id)
Relationships:
One-to-One: connected to users via user_id.

<u>Business Travel Table</u>
4.Table Name: business_travel
Column names
4.1 id: Integer (Primary Key)
4.2 kilometer_traveled: Float (nullable)
4.3 fuel_efficiency: Float (nullable)
4.4 user_id: Integer (Foreign Key referencing users.id)
Relationships:
One-to-One: connected to users via user_id.

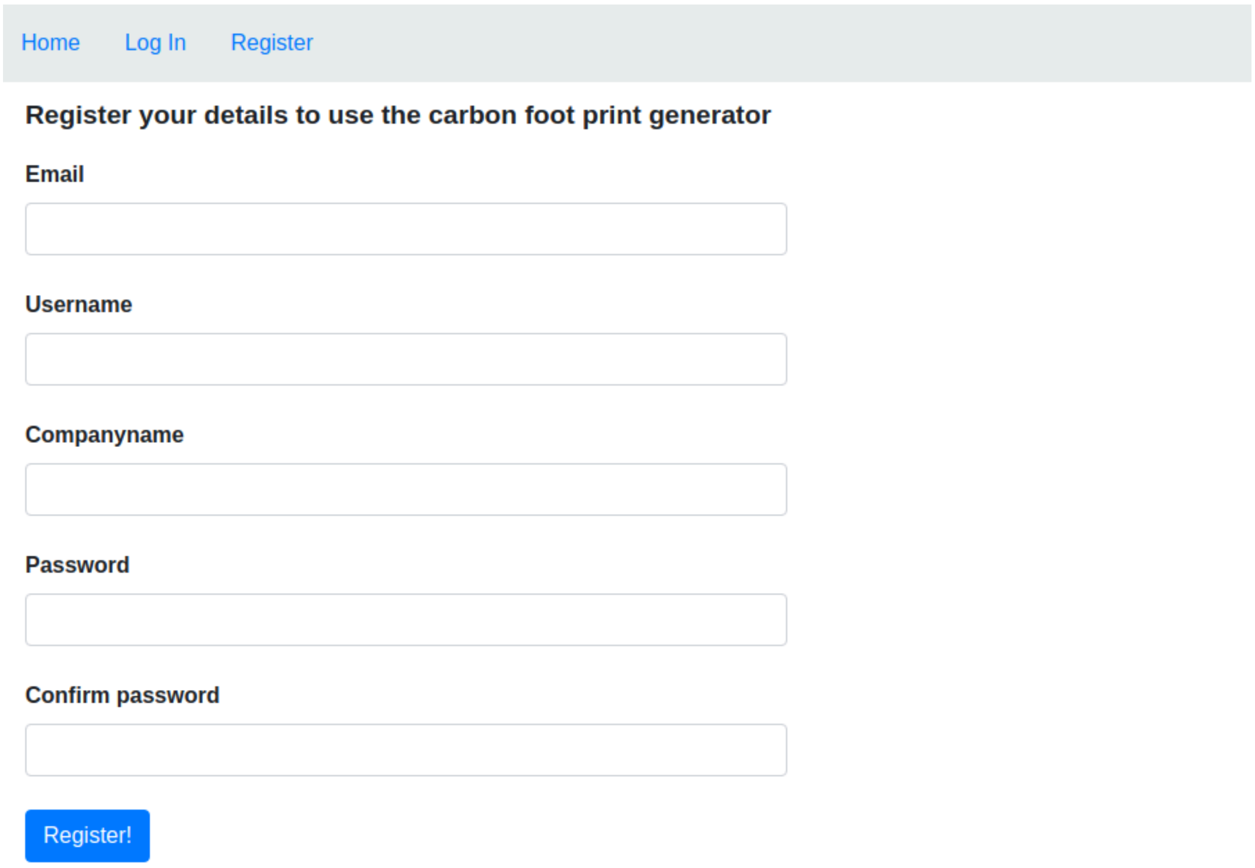
WorkFlow Explanation

When users(i.e. company owners) are created through the registration process using the application they are connected to energy usage waste and business travel through user_id as foreign key which is then used to store the respective data of the individual user regarding their company.

Application Features

After running the application in local environment go to signup page by clicking on register or by following url below
User Signup(url : localhost:5000/register or https://carbonemissioncalculatortool.onrender.com/register)
Here User signups by Filling their Data which is (Email, Username, CompanyName, Password)
Screenshot

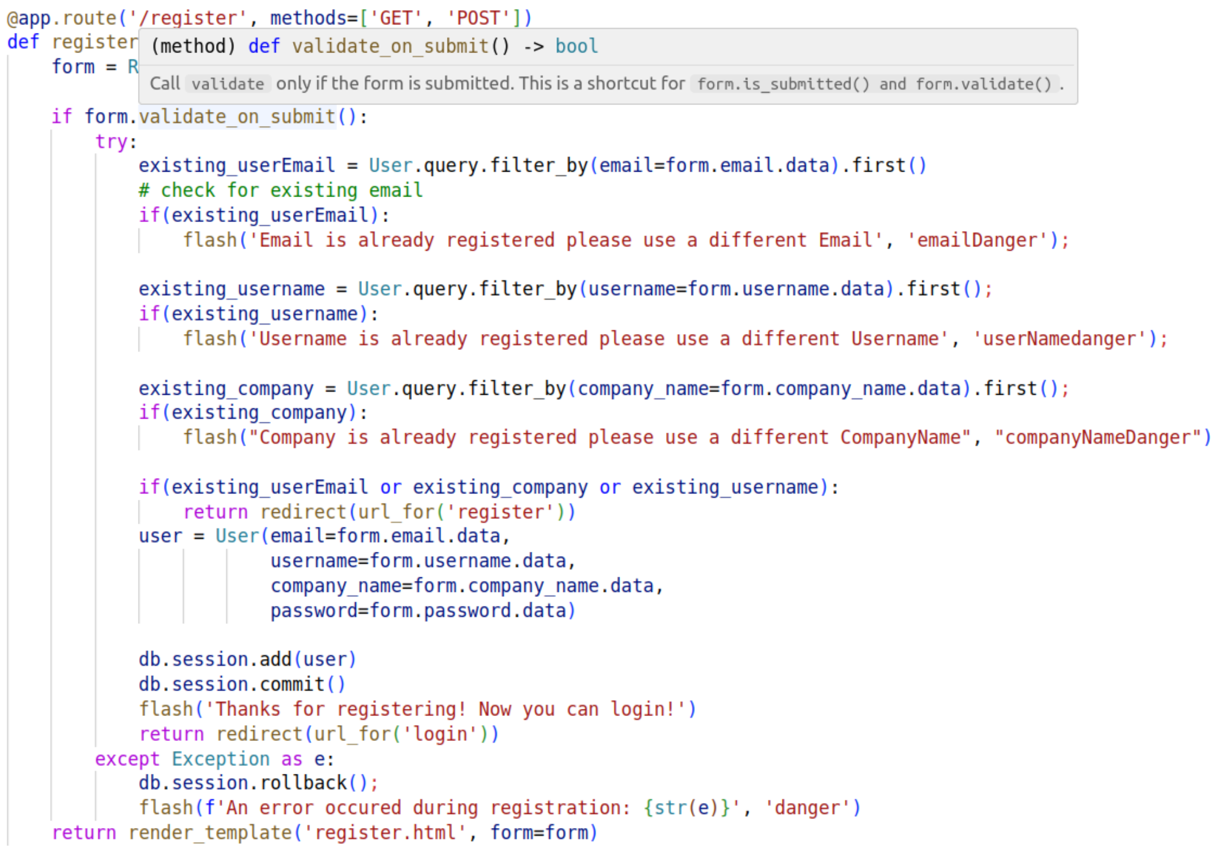
Registration page overview



Register your details to use the carbon foot print generator

Email	
Username	
Companyname	
Password	
Confirm password	

Code snippet



```
@app.route('/register', methods=['GET', 'POST'])
def register
    form = R
```

(method) def validate_on_submit() -> bool

Call validate only if the form is submitted. This is a shortcut for form.is_submitted() and form.validate().

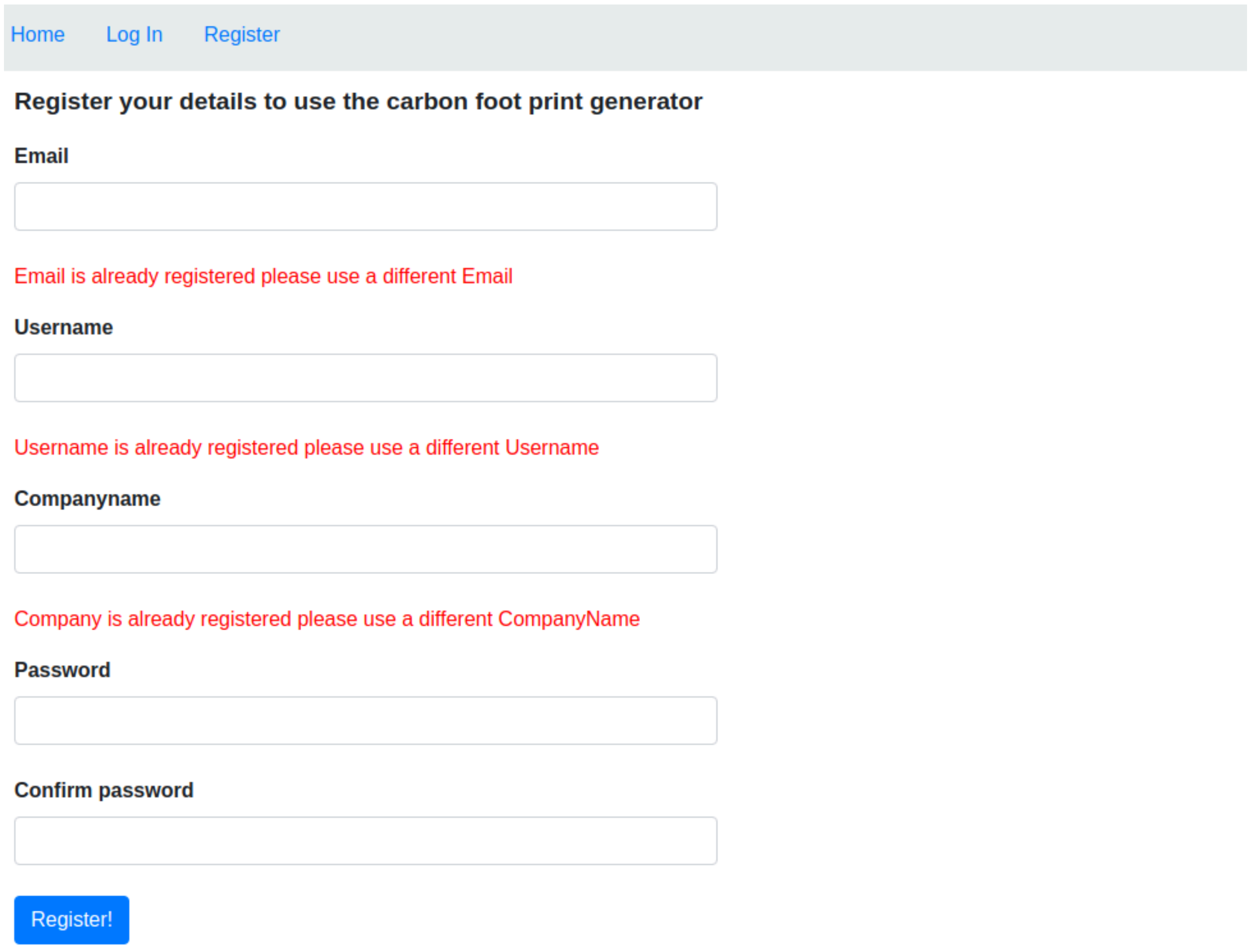
```
    if form.validate_on_submit():
        try:
            existing_userEmail = User.query.filter_by(email=form.email.data).first()
            # check for existing email
            if(existing_userEmail):
                flash('Email is already registered please use a different Email', 'emailDanger');

            existing_username = User.query.filter_by(username=form.username.data).first();
            if(existing_username):
                flash('Username is already registered please use a different Username', 'userNamedanger');

            existing_company = User.query.filter_by(company_name=form.company_name.data).first();
            if(existing_company):
                flash("Company is already registered please use a different CompanyName", "companyNameDanger")

            if(existing_userEmail or existing_company or existing_username):
                return redirect(url_for('register'))
            user = User(email=form.email.data,
                        username=form.username.data,
                        company_name=form.company_name.data,
                        password=form.password.data)

            db.session.add(user)
            db.session.commit()
            flash('Thanks for registering! Now you can login!')
            return redirect(url_for('login'))
        except Exception as e:
            db.session.rollback();
            flash(f'An error occured during registration: {str(e)}', 'danger')
    return render_template('register.html', form=form)
```

The form is validated using inbuilt validation method of flask form and then the data of email password username and company name are extracted from the form and saved in database

Error handling in registration

Error handling here is done using the try exception method. Error occurs if already registered email address, company_name or username is used and this rollback the database as shown in expect method

Here is the screenshot of Error page when above mentioned things happen



Register your details to use the carbon foot print generator

Email

Email is already registered please use a different Email

Username

Username is already registered please use a different Username

Companyname

Company is already registered please use a different CompanyName

Password

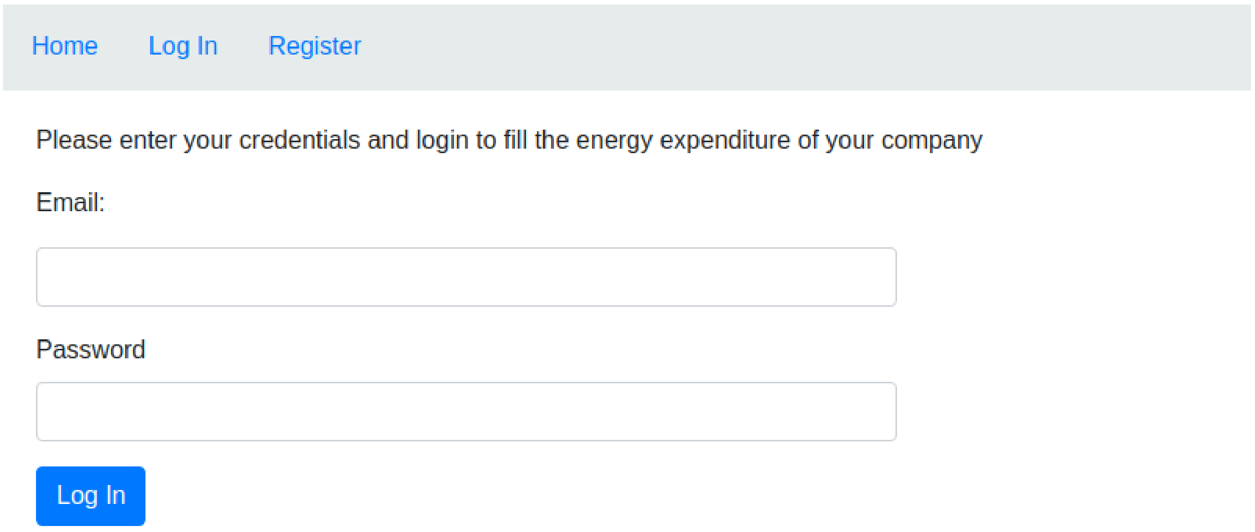
Confirm password

User Login

After signup success they are redirected to login page

Url: https://carbonemissioncalculatortool.onrender.com/login

Login page overview



Home Log In Register

Please enter your credentials and login to fill the energy expenditure of your company

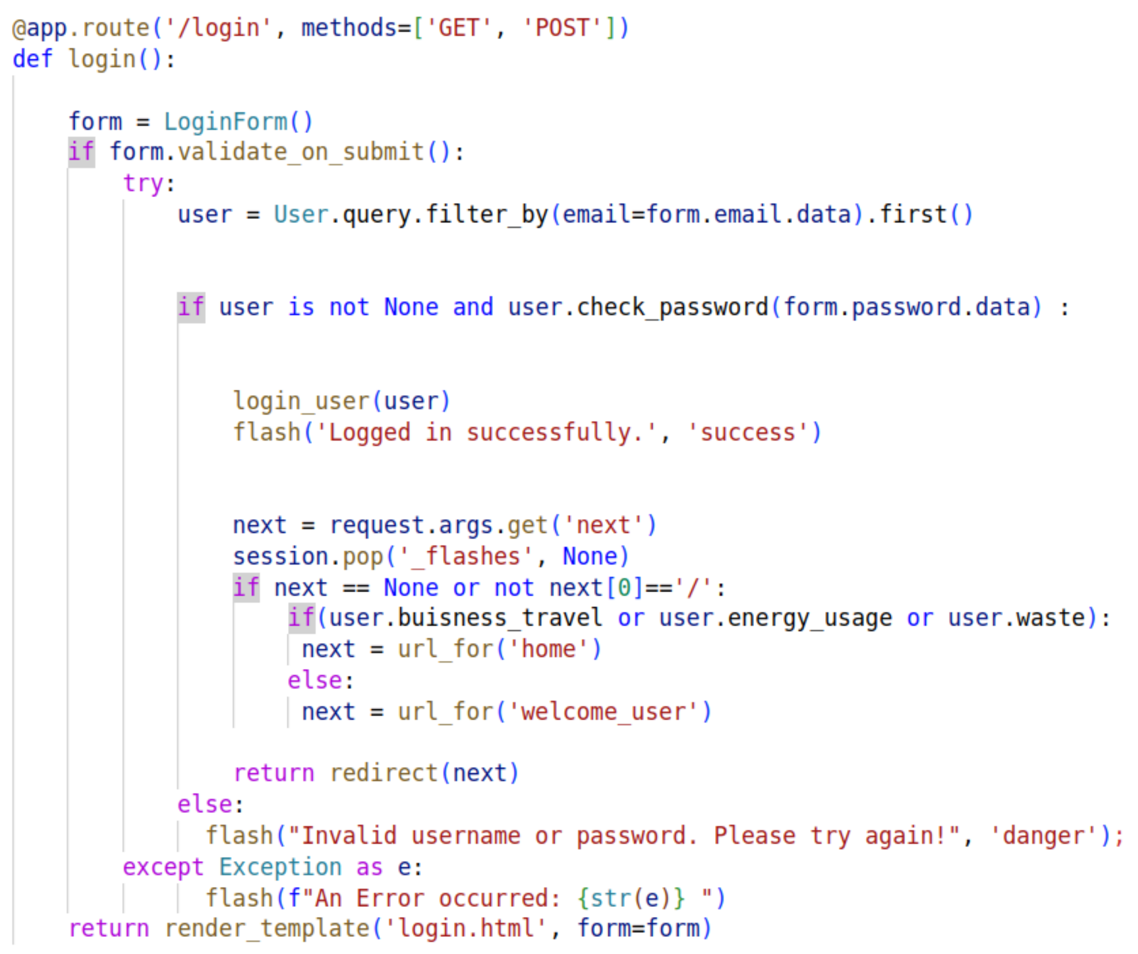
Email:

Password

Data Input

Url: https://carbonemissioncalculatortool.onrender.com/welcome
here they can enter their email and password.

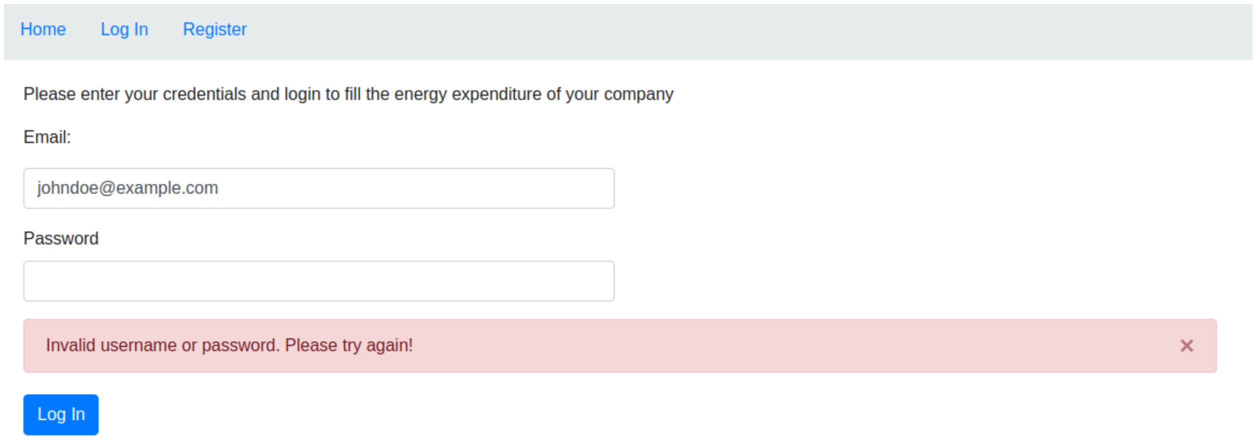
Code snippet


```python
@app.route('/login', methods=['GET', 'POST'])
def login():

    form = LoginForm()
    if form.validate_on_submit():
        try:
            user = User.query.filter_by(email=form.email.data).first()


            if user is not None and user.check_password(form.password.data) :


                login_user(user)
                flash('Logged in successfully.', 'success')


                next = request.args.get('next')
                session.pop('_flashes', None)
                if next == None or not next[0]=='/':
                    if(user.buisness_travel or user.energy_usage or user.waste):
                     next = url_for('home')
                    else:
                     next = url_for('welcome_user')

                return redirect(next)
            else:
              flash("Invalid username or password. Please try again!", 'danger');
        except Exception as e:
              flash(f"An Error occurred: {str(e)} ")
    return render_template('login.html', form=form)
```

Here the code checks if the user exists and then checks the password if they match then user validation is successful.
Also if the user enters the url then after validation the user is redirected to the same page.

Error handling in Login

If the user enters an email that is not registered or if the password is wrong then an error is generated which is handled through try expectation method.


Home Log In Register

Please enter your credentials and login to fill the energy expenditure of your company

Email:

johndoe@example.com

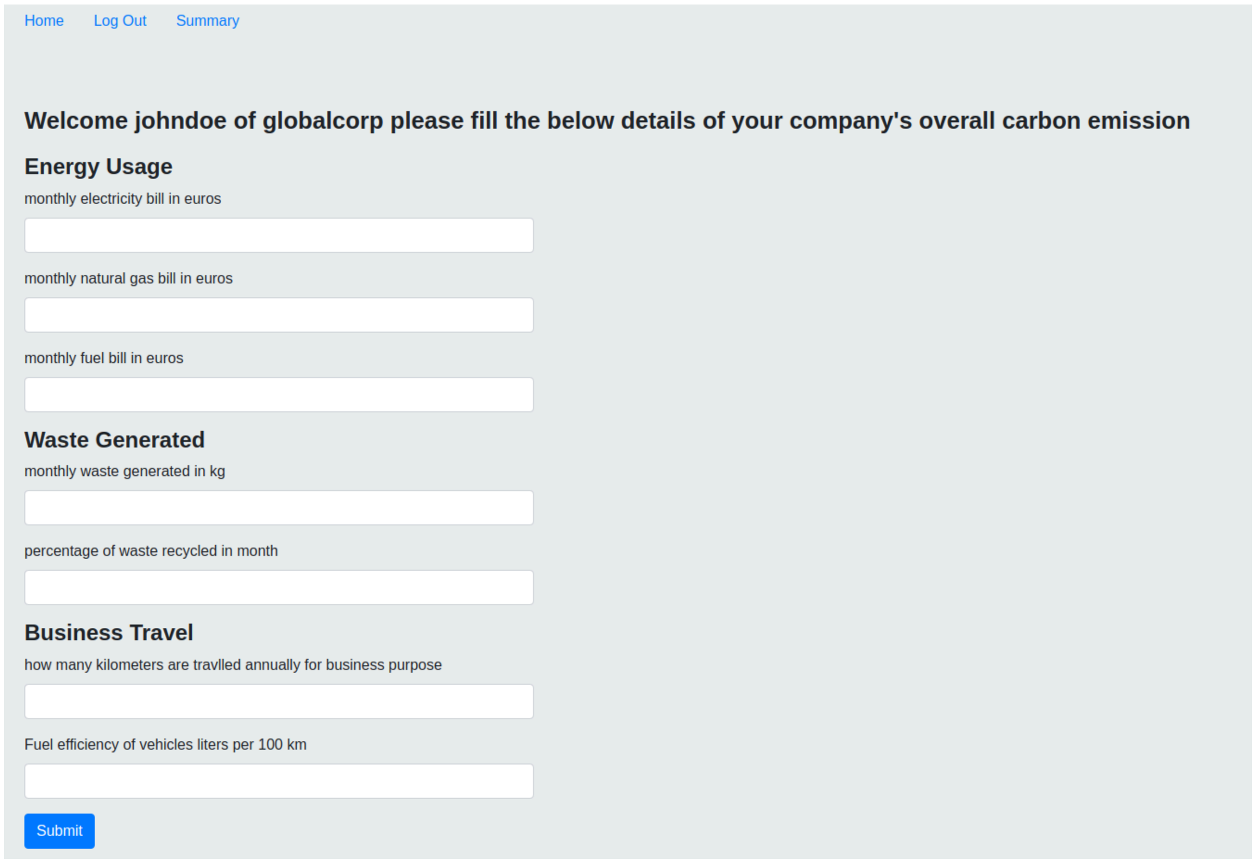
Password

Invalid username or password. Please try again!

Log In

Welcome page overview

After successful Login the first Time users are redirected to the welcome page where they can fill the data regarding their companies different parameters(Energy, Waste and Travel) after submitting their data they are redirected to the Home page.



Welcome johndoe of globalcorp please fill the below details of your company's overall carbon emission

Energy Usage

monthly electricity bill in euros

monthly natural gas bill in euros

monthly fuel bill in euros

Waste Generated

monthly waste generated in kg

percentage of waste recycled in month

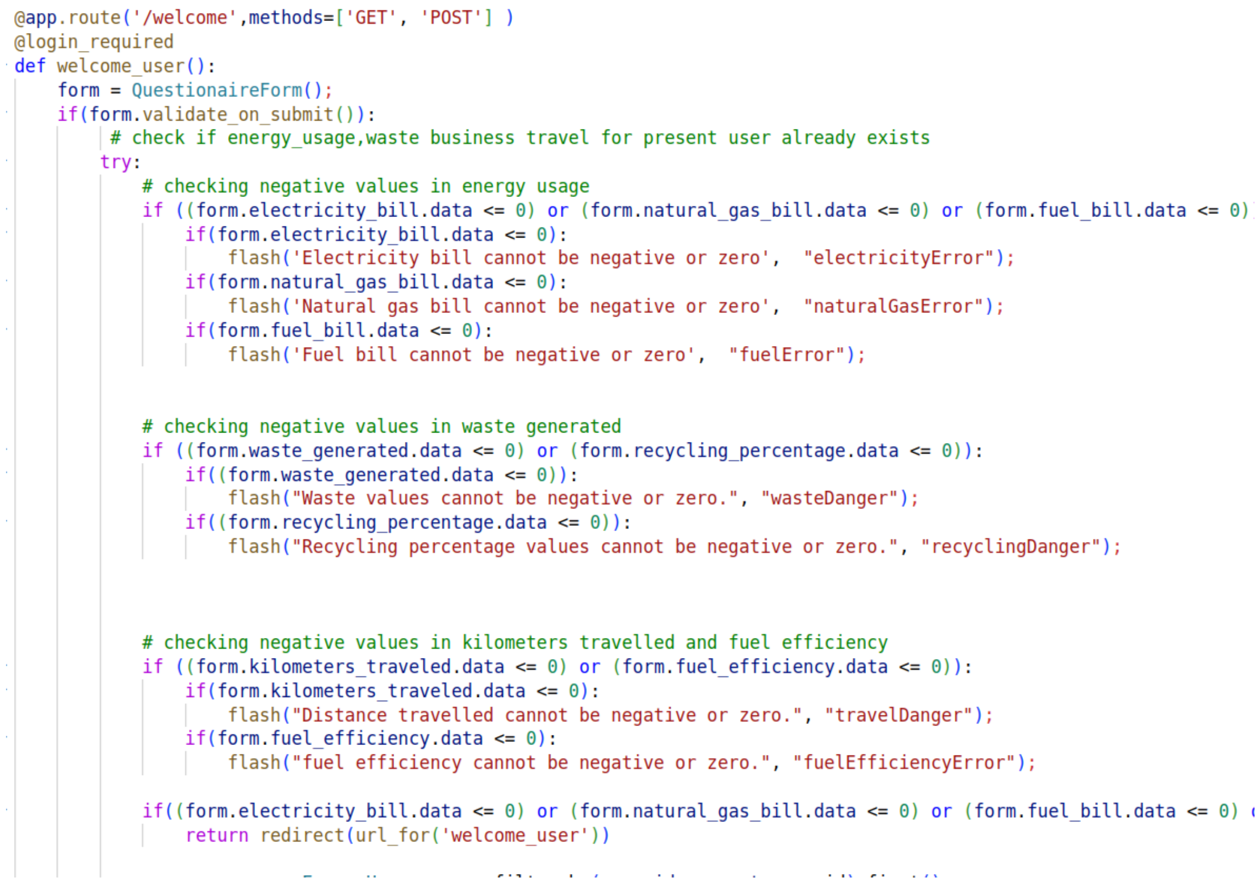
Business Travel

how many kilometers are travlled annually for business purpose

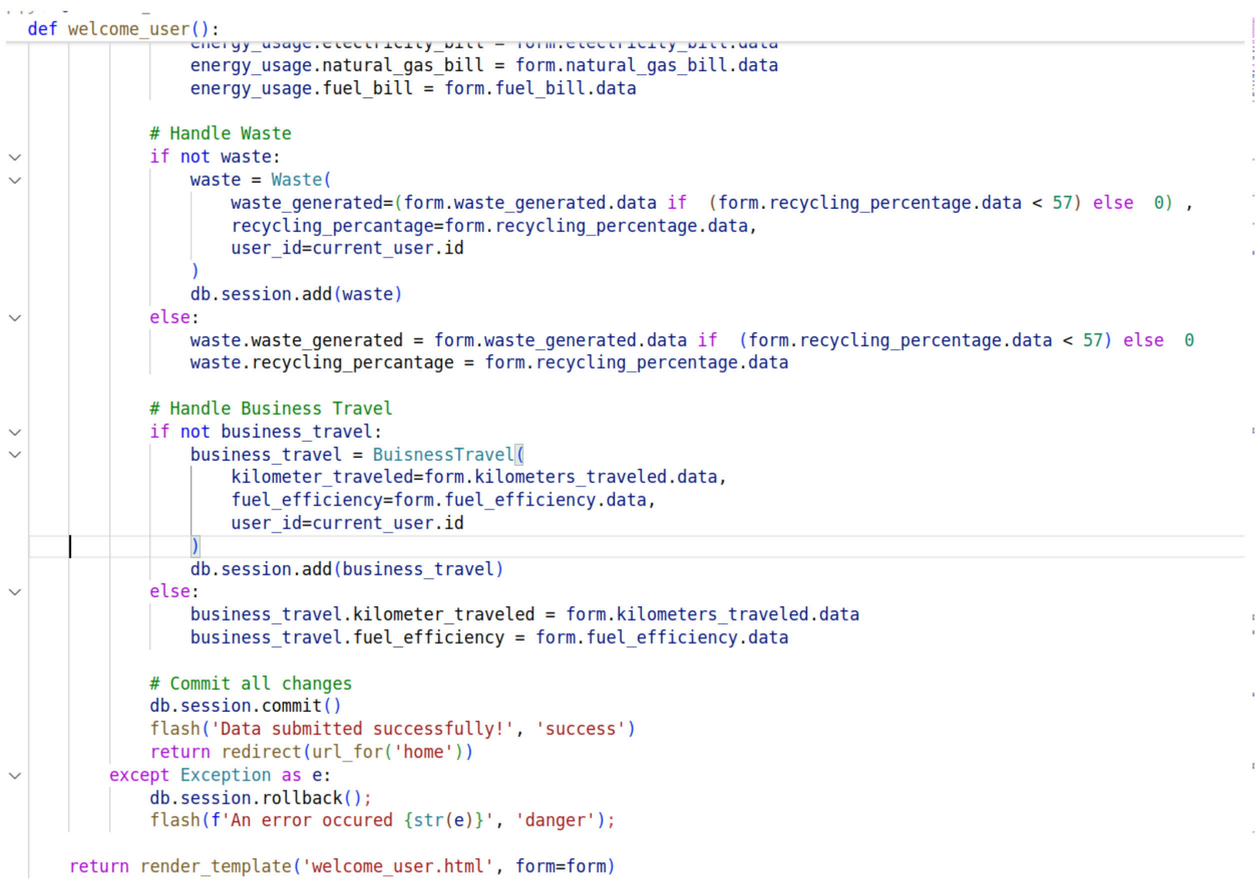
Fuel efficiency of vehicles liters per 100 km

Note: if the user has already entered their data then they are directly redirected to home page where they have option to edit the data

Code snippet



```
@app.route('/welcome',methods=['GET', 'POST'] )
@login_required
def welcome_user():
    form = QuestionaireForm();
    if(form.validate_on_submit()):
        # check if energy_usage,waste business travel for present user already exists
        try:
            # checking negative values in energy usage
            if ((form.electricity_bill.data <= 0) or (form.natural_gas_bill.data <= 0) or (form.fuel_bill.data <= 0)
                if(form.electricity_bill.data <= 0):
                    flash('Electricity bill cannot be negative or zero',  "electricityError");
                if(form.natural_gas_bill.data <= 0):
                    flash('Natural gas bill cannot be negative or zero',  "naturalGasError");
                if(form.fuel_bill.data <= 0):
                    flash('Fuel bill cannot be negative or zero',  "fuelError");


            # checking negative values in waste generated
            if ((form.waste_generated.data <= 0) or (form.recycling_percentage.data <= 0)):
                if((form.waste_generated.data <= 0)):
                    flash("Waste values cannot be negative or zero.", "wasteDanger");
                if((form.recycling_percentage.data <= 0)):
                    flash("Recycling percentage values cannot be negative or zero.", "recyclingDanger");



            # checking negative values in kilometers travelled and fuel efficiency
            if ((form.kilometers_traveled.data <= 0) or (form.fuel_efficiency.data <= 0)):
                if(form.kilometers_traveled.data <= 0):
                    flash("Distance travelled cannot be negative or zero.", "travelDanger");
                if(form.fuel_efficiency.data <= 0):
                    flash("fuel efficiency cannot be negative or zero.", "fuelEfficiencyError");

            if((form.electricity_bill.data <= 0) or (form.natural_gas_bill.data <= 0) or (form.fuel_bill.data <= 0)
                return redirect(url_for('welcome_user'))
```



```
def welcome_user():
                energy_usage.electricity_bill = form.electricity_bill.data
                energy_usage.natural_gas_bill = form.natural_gas_bill.data
                energy_usage.fuel_bill = form.fuel_bill.data

            # Handle Waste
            if not waste:
                waste = Waste(
                    waste_generated=(form.waste_generated.data if  (form.recycling_percentage.data < 57) else  0) ,
                    recycling_percantage=form.recycling_percentage.data,
                    user_id=current_user.id
                )
                db.session.add(waste)
            else:
                waste.waste_generated = form.waste_generated.data if  (form.recycling_percentage.data < 57) else  0
                waste.recycling_percantage = form.recycling_percentage.data

            # Handle Business Travel
            if not business_travel:
                business_travel = BuisnessTravel(
                    kilometer_traveled=form.kilometers_traveled.data,
                    fuel_efficiency=form.fuel_efficiency.data,
                    user_id=current_user.id
                )
                db.session.add(business_travel)
            else:
                business_travel.kilometer_traveled = form.kilometers_traveled.data
                business_travel.fuel_efficiency = form.fuel_efficiency.data

            # Commit all changes
            db.session.commit()
            flash('Data submitted successfully!', 'success')
            return redirect(url_for('home'))
        except Exception as e:
            db.session.rollback();
            flash(f'An error occured {str(e)}', 'danger');

    return render_template('welcome_user.html', form=form)
```

Here after submitting the data the regarding energy waste and business travel this code parses the data and adds it to the respective database columns with user id as the foreign key

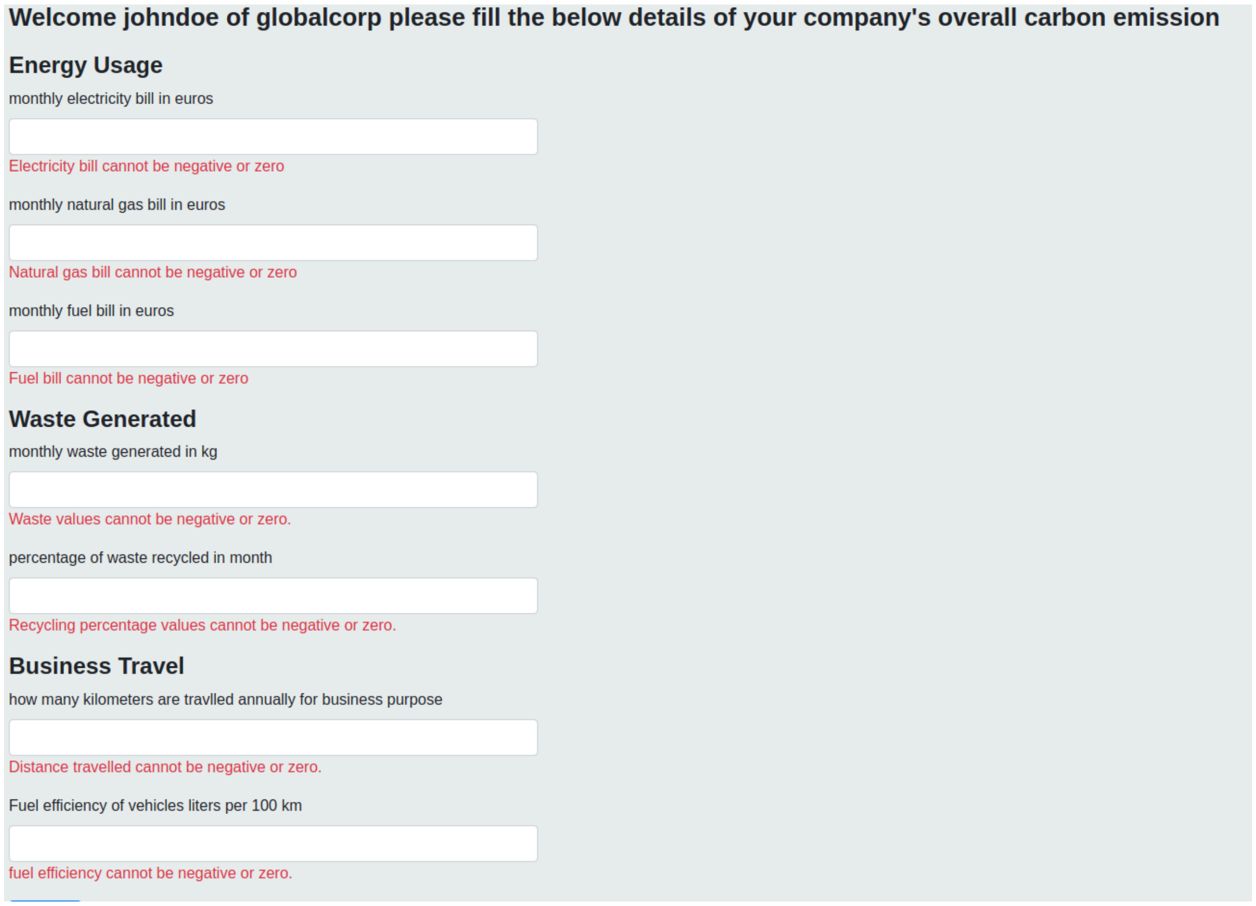
Error handling

If user enters data which is negative then it shows error which is handled by try exception as shown below in the screen shot

```
  if ((form.electricity_bill.data <= 0) or (form.natural_gas_bill.data <=
0) or (form.fuel_bill.data <= 0)):
                if(form.electricity_bill.data <= 0):
                    flash('Electricity bill cannot be negative or zero',
"electricityError");
                if(form.natural_gas_bill.data <= 0):
                    flash('Natural gas bill cannot be negative or zero',
"naturalGasError");
```

```
                if(form.fuel_bill.data <= 0):
                    flash('Fuel bill cannot be negative or zero', 
"fuelError");


            # checking negative values in waste generated
            if ((form.waste_generated.data <= 0) or 
(form.recycling_percentage.data <= 0)):
                if((form.waste_generated.data <= 0)):
                    flash("Waste values cannot be negative or zero.", 
"wasteDanger");
                if((form.recycling_percentage.data <= 0)):
                    flash("Recycling percentage values cannot be negative 
or zero.", "recyclingDanger");



            # checking negative values in kilometers travelled and fuel 
efficiency
            if ((form.kilometers_traveled.data <= 0) or 
(form.fuel_efficiency.data <= 0)):
                if(form.kilometers_traveled.data <= 0):
                    flash("Distance travelled cannot be negative or zero.", 
"travelDanger");
                if(form.fuel_efficiency.data <= 0):
                    flash("fuel efficiency cannot be negative or zero.", 
"fuelEfficiencyError");

            if((form.electricity_bill.data <= 0) or 
(form.natural_gas_bill.data <= 0) or
            (form.fuel_bill.data <= 0) or (form.waste_generated.data <= 0) 
or
            (form.recycling_percentage.data <= 0) or
            (form.kilometers_traveled.data <= 0) or
              (form.fuel_efficiency.data <= 0)):
                return redirect(url_for('welcome_user'))
```

Here is the screenshot of its UI representation of error



Welcome johndoe of globalcorp please fill the below details of your company's overall carbon emission

Energy Usage

monthly electricity bill in euros

Electricity bill cannot be negative or zero

monthly natural gas bill in euros

Natural gas bill cannot be negative or zero

monthly fuel bill in euros

Fuel bill cannot be negative or zero

Waste Generated

monthly waste generated in kg

Waste values cannot be negative or zero.

percentage of waste recycled in month

Recycling percentage values cannot be negative or zero.

Business Travel

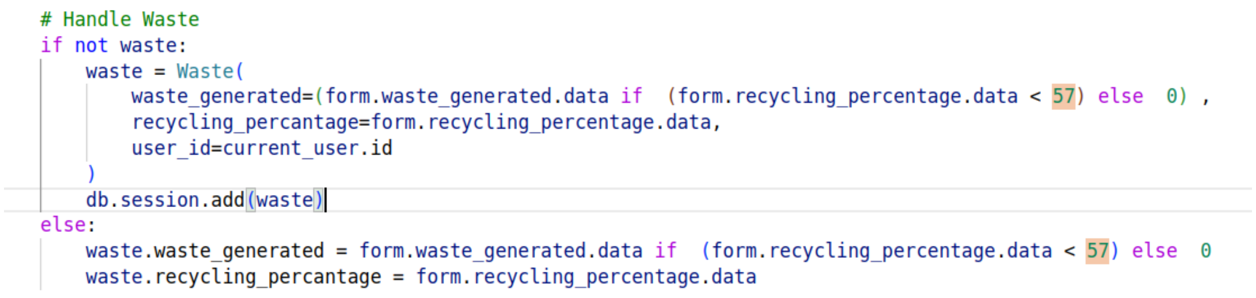
how many kilometers are travlled annually for business purpose

Distance travelled cannot be negative or zero.

Fuel efficiency of vehicles liters per 100 km

fuel efficiency cannot be negative or zero.

Error handling when recycling waste is more than or equal to 57 percent then the waste generated becomes zero



```
# Handle Waste
if not waste:
    waste = Waste(
        waste_generated=(form.waste_generated.data if  (form.recycling_percentage.data < 57) else  0) ,
        recycling_percantage=form.recycling_percentage.data,
        user_id=current_user.id
    )
    db.session.add(waste)
else:
    waste.waste_generated = form.waste_generated.data if  (form.recycling_percentage.data < 57) else  0
    waste.recycling_percantage = form.recycling_percentage.data
```

Analytics and reporting

In home page the user can see the performance in the form of bar chart based on the data they have entered and calculations done in backend

Max footprint will be marked in red and then blue and green for the lowest

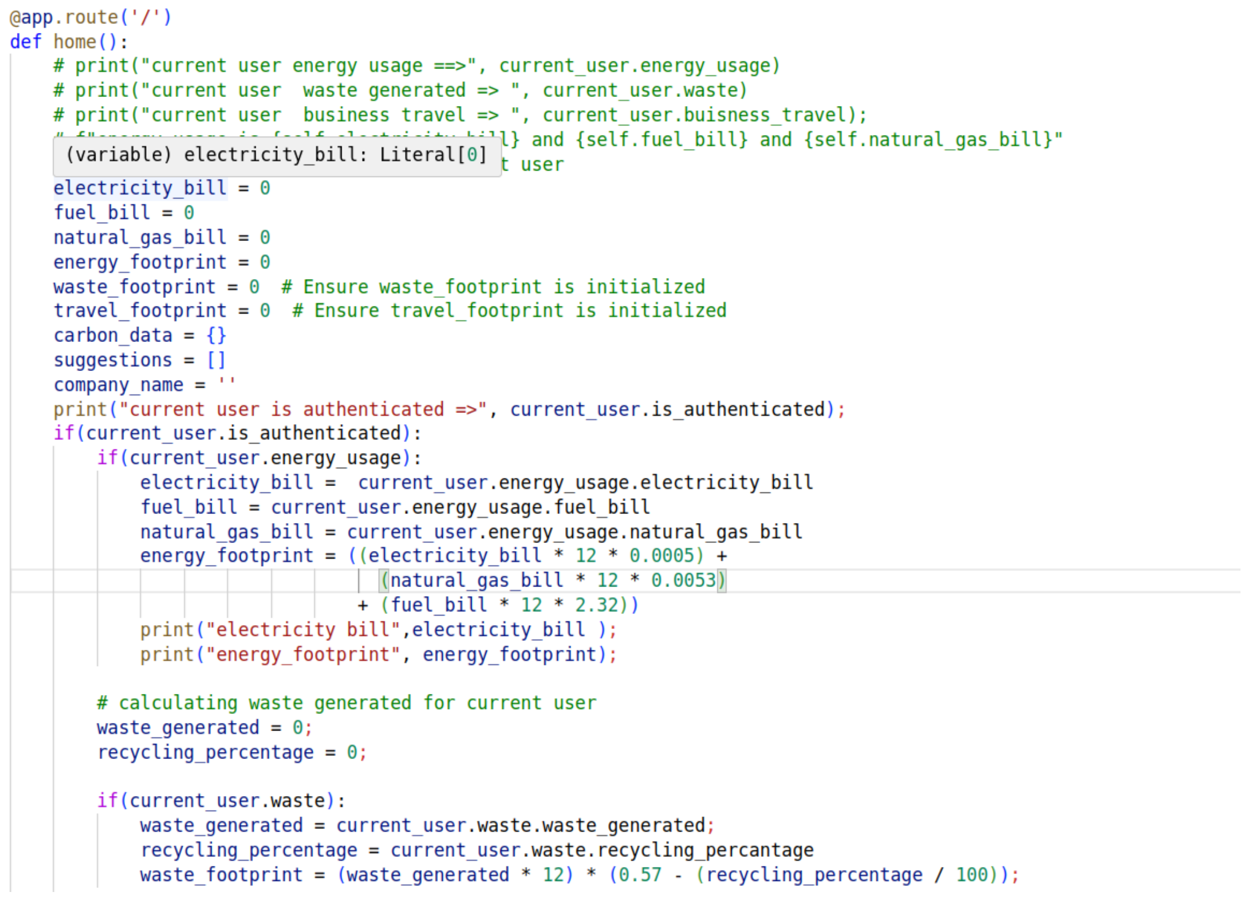
Here suggestions are also marked in red, blue and green based on their overall magnitude respectively. Red being the highest and green for lowest

Also there is an edit option if the user wants to edit their data.

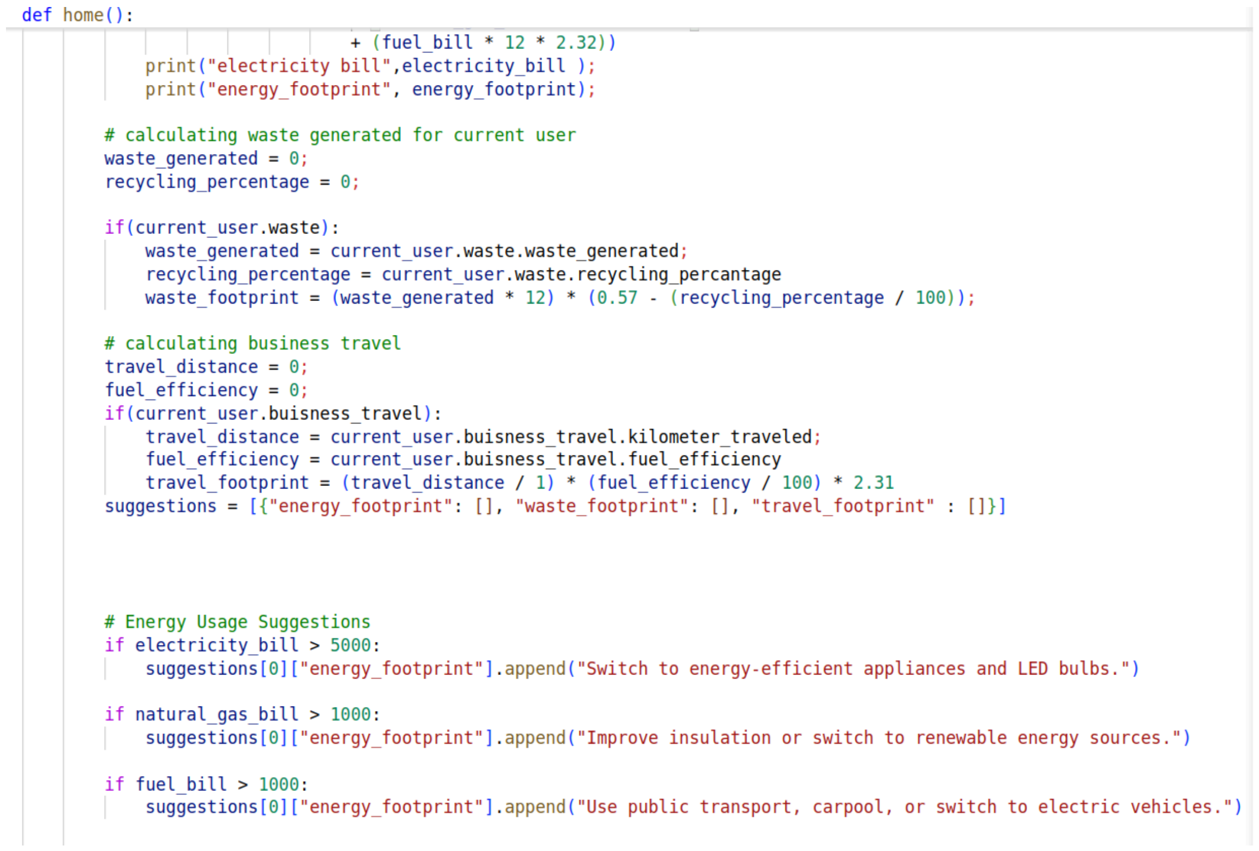
Download button for download the chart and suggestion

Here is a snippet of downloaded pdf

Code snippet

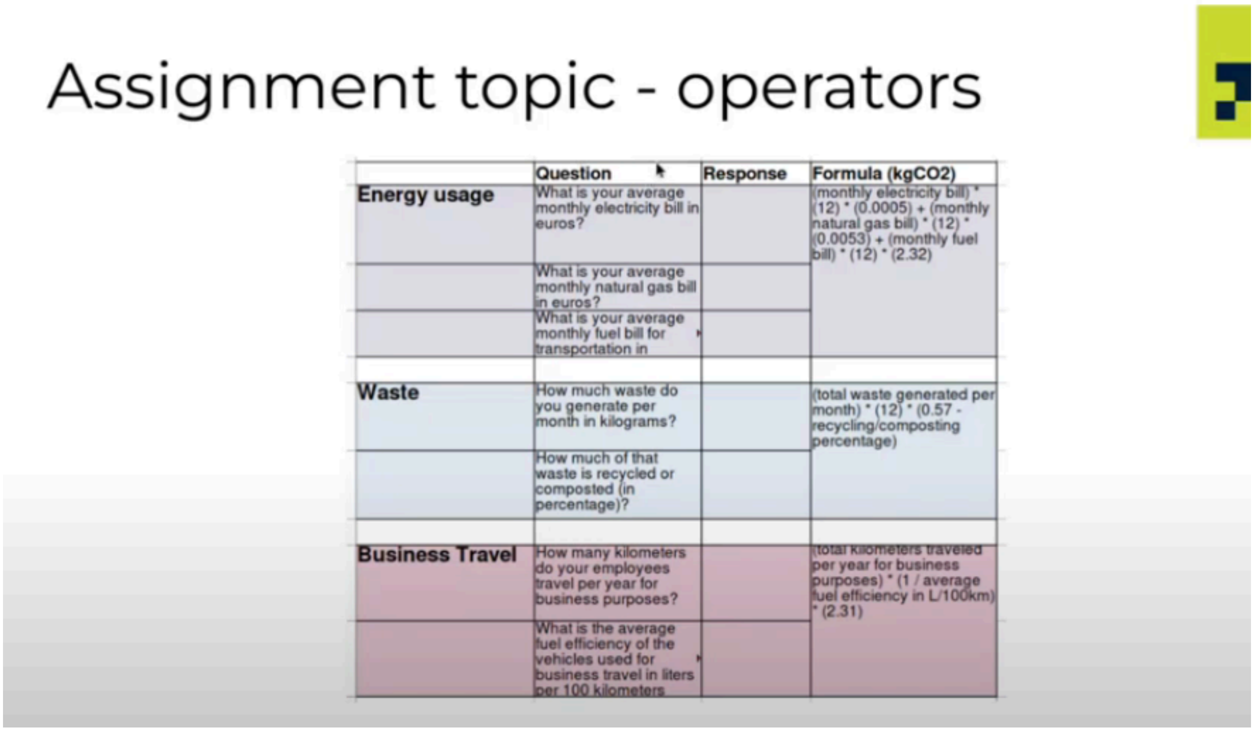


```
@app.route('/')
def home():
    # print("current user energy usage ==>", current_user.energy_usage)
    # print("current user  waste generated => ", current_user.waste)
    # print("current user  business travel => ", current_user.buisness_travel);
    # f"energy usage is {self.electricity_bill} and {self.fuel_bill} and {self.natural_gas_bill}"
    (variable) electricity_bill: Literal[0]  t user
    electricity_bill = 0
    fuel_bill = 0
    natural_gas_bill = 0
    energy_footprint = 0
    waste_footprint = 0  # Ensure waste_footprint is initialized
    travel_footprint = 0  # Ensure travel_footprint is initialized
    carbon_data = {}
    suggestions = []
    company_name = ''
    print("current user is authenticated =>", current_user.is_authenticated);
    if(current_user.is_authenticated):
        if(current_user.energy_usage):
            electricity_bill =  current_user.energy_usage.electricity_bill
            fuel_bill = current_user.energy_usage.fuel_bill
            natural_gas_bill = current_user.energy_usage.natural_gas_bill
            energy_footprint = ((electricity_bill * 12 * 0.0005) +
                                 (natural_gas_bill * 12 * 0.0053)
                                + (fuel_bill * 12 * 2.32))
            print("electricity bill",electricity_bill );
            print("energy_footprint", energy_footprint);

        # calculating waste generated for current user
        waste_generated = 0;
        recycling_percentage = 0;

        if(current_user.waste):
            waste_generated = current_user.waste.waste_generated;
            recycling_percentage = current_user.waste.recycling_percantage
            waste_footprint = (waste_generated * 12) * (0.57 - (recycling_percentage / 100));
```



```
def home():
                                + (fuel_bill * 12 * 2.32))
            print("electricity bill",electricity_bill );
            print("energy_footprint", energy_footprint);

        # calculating waste generated for current user
        waste_generated = 0;
        recycling_percentage = 0;

        if(current_user.waste):
            waste_generated = current_user.waste.waste_generated;
            recycling_percentage = current_user.waste.recycling_percantage
            waste_footprint = (waste_generated * 12) * (0.57 - (recycling_percentage / 100));

        # calculating business travel
        travel_distance = 0;
        fuel_efficiency = 0;
        if(current_user.buisness_travel):
            travel_distance = current_user.buisness_travel.kilometer_traveled;
            fuel_efficiency = current_user.buisness_travel.fuel_efficiency
            travel_footprint = (travel_distance / 1) * (fuel_efficiency / 100) * 2.31
        suggestions = [{"energy_footprint": [], "waste_footprint": [], "travel_footprint" : []}]



        # Energy Usage Suggestions
        if electricity_bill > 5000:
            suggestions[0]["energy_footprint"].append("Switch to energy-efficient appliances and LED bulbs.")

        if natural_gas_bill > 1000:
            suggestions[0]["energy_footprint"].append("Improve insulation or switch to renewable energy sources.")

        if fuel_bill > 1000:
            suggestions[0]["energy_footprint"].append("Use public transport, carpool, or switch to electric vehicles.")
```

Following Image has been used to calculate the above mentioned different parameters and the code with features will be discussed in the following section

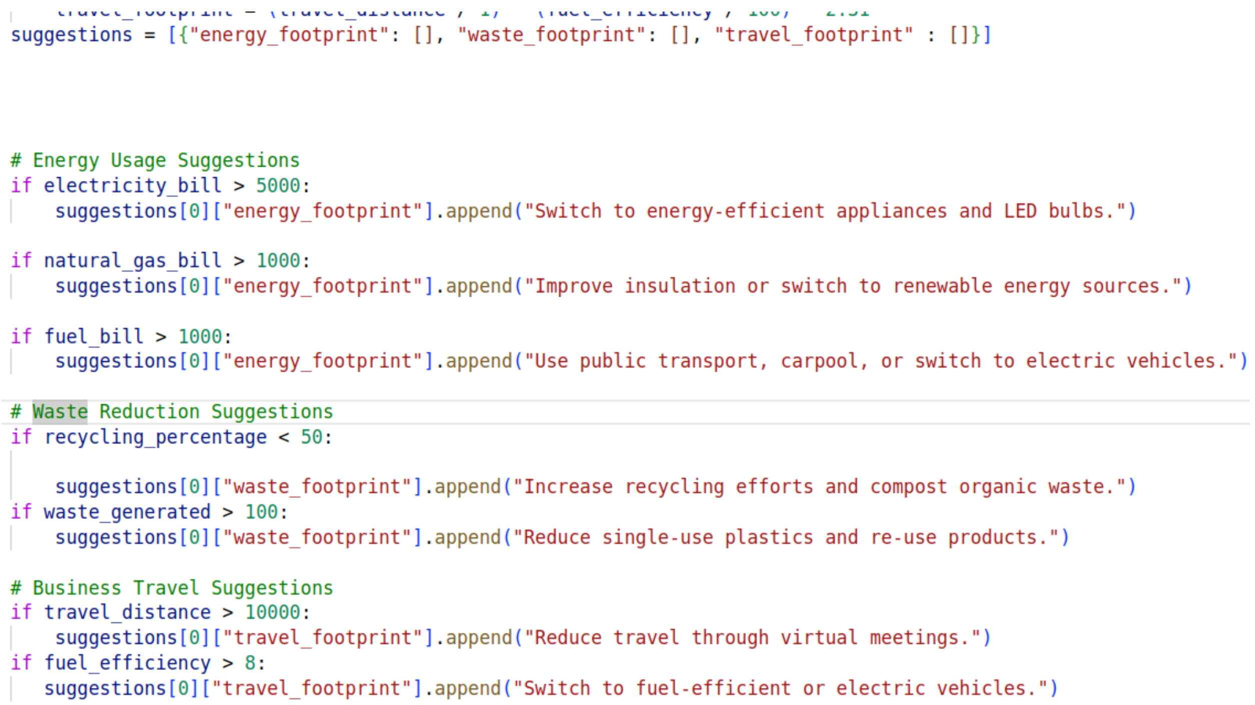


Assignment topic - operators

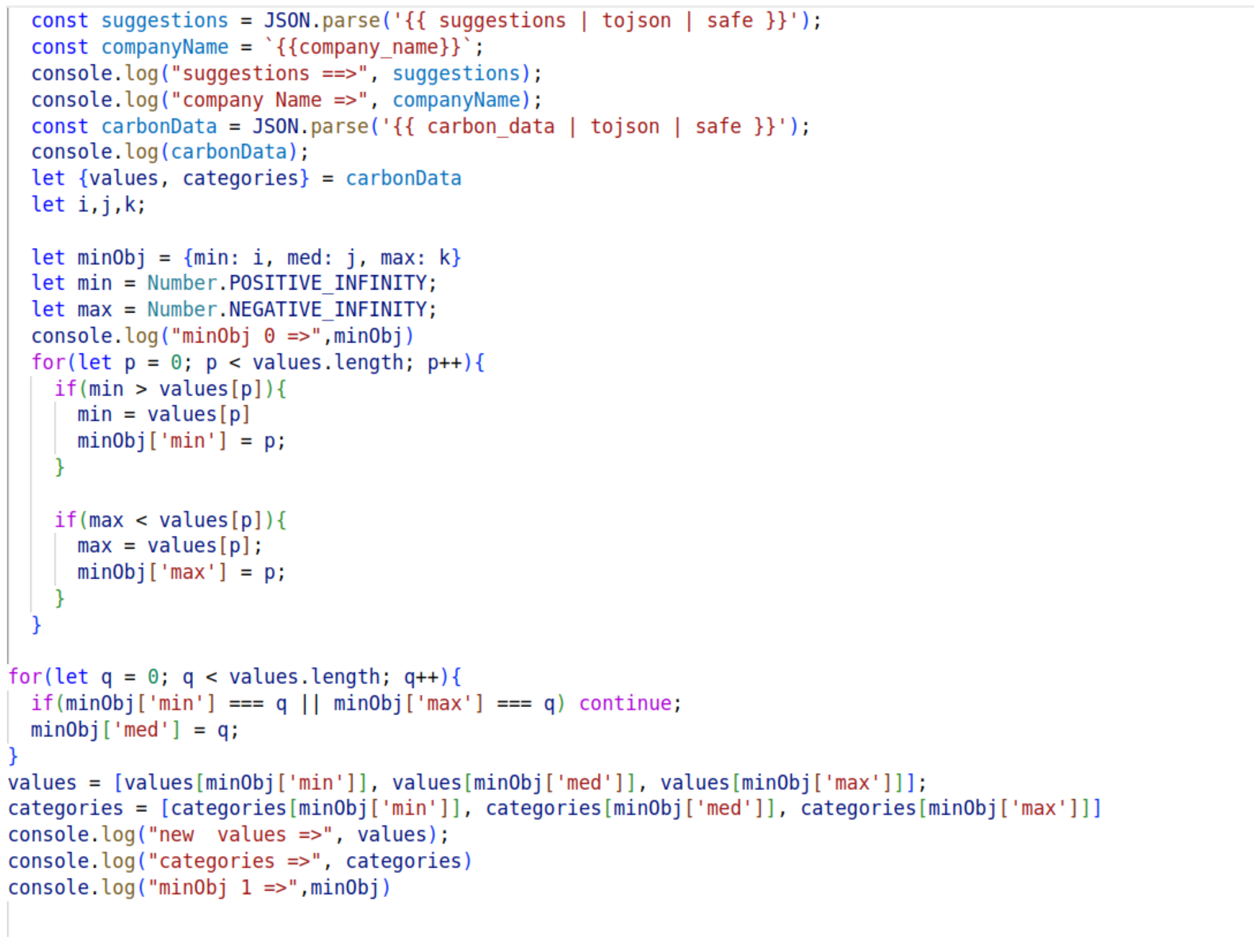
	Question	Response	Formula (kgCO2)
Energy usage	What is your average monthly electricity bill in euros?		(monthly electricity bill) * (12) * (0.0005) + (monthly natural gas bill) * (12) * (0.0053) + (monthly fuel bill) * (12) * (2.32)
	What is your average monthly natural gas bill in euros?		
	What is your average monthly fuel bill for transportation in		
Waste	How much waste do you generate per month in kilograms?		(total waste generated per month) * (12) * (0.57 - recycling/composting percentage)
	How much of that waste is recycled or composted (in percentage)?		
Business Travel	How many kilometers do your employees travel per year for business purposes?		(total kilometers traveled per year for business purposes) * (1 / average fuel efficiency in L/100km) * (2.31)
	What is the average fuel efficiency of the vehicles used for business travel in liters per 100 kilometers		

Also suggestion is calculated and then added based on the below logic

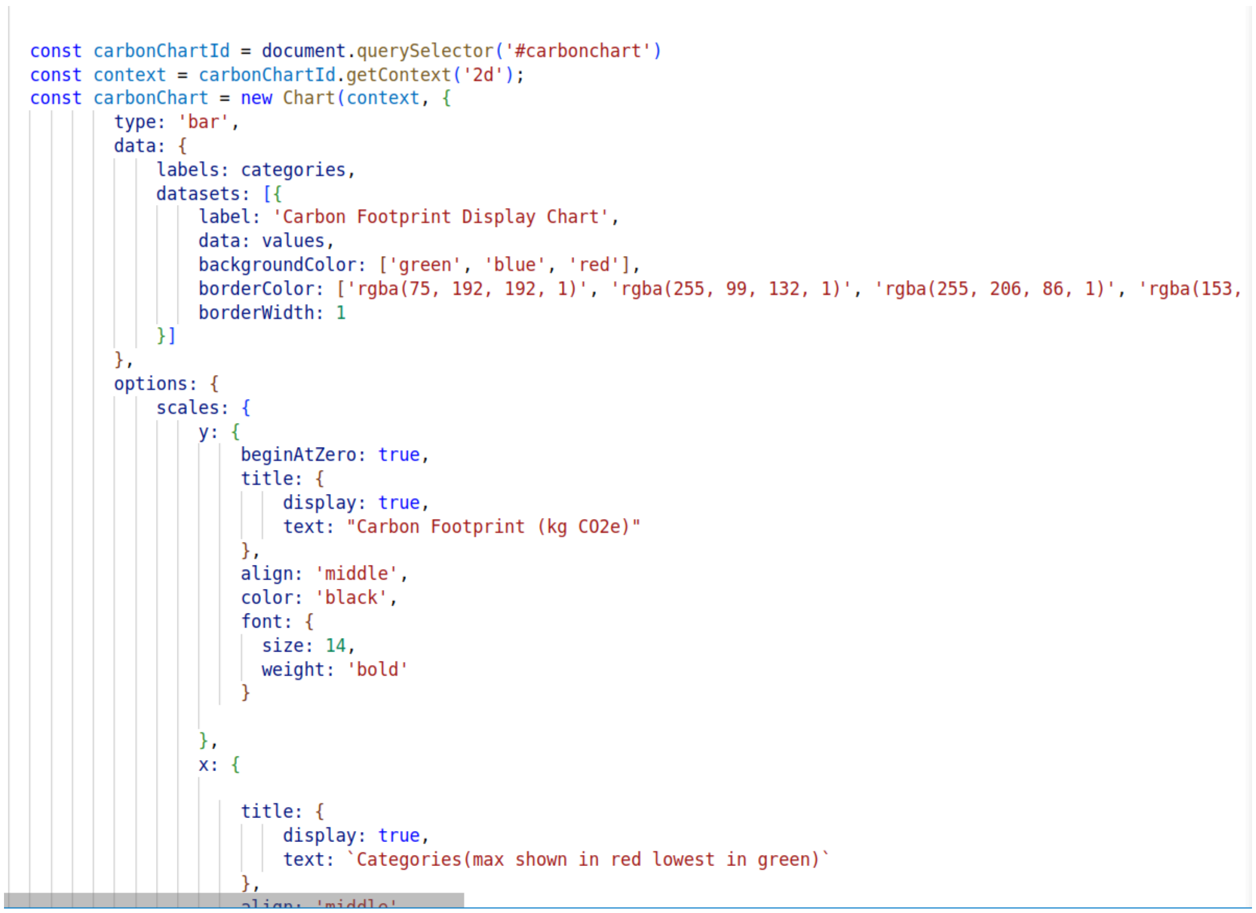
I have used Javascript for displaying and downloading the chart because the chart is displayed as part of user interface in web application so controlling the design using HTML, CSS and Javascript is easier which also makes the user experience better.Also using backend servers for creating the report creates extra load on application server which can be avoided using javascript
However, I have created a separate file using plotly that can create the same report by taking user input in the form of email and password
Here is the code snippet for display and download button



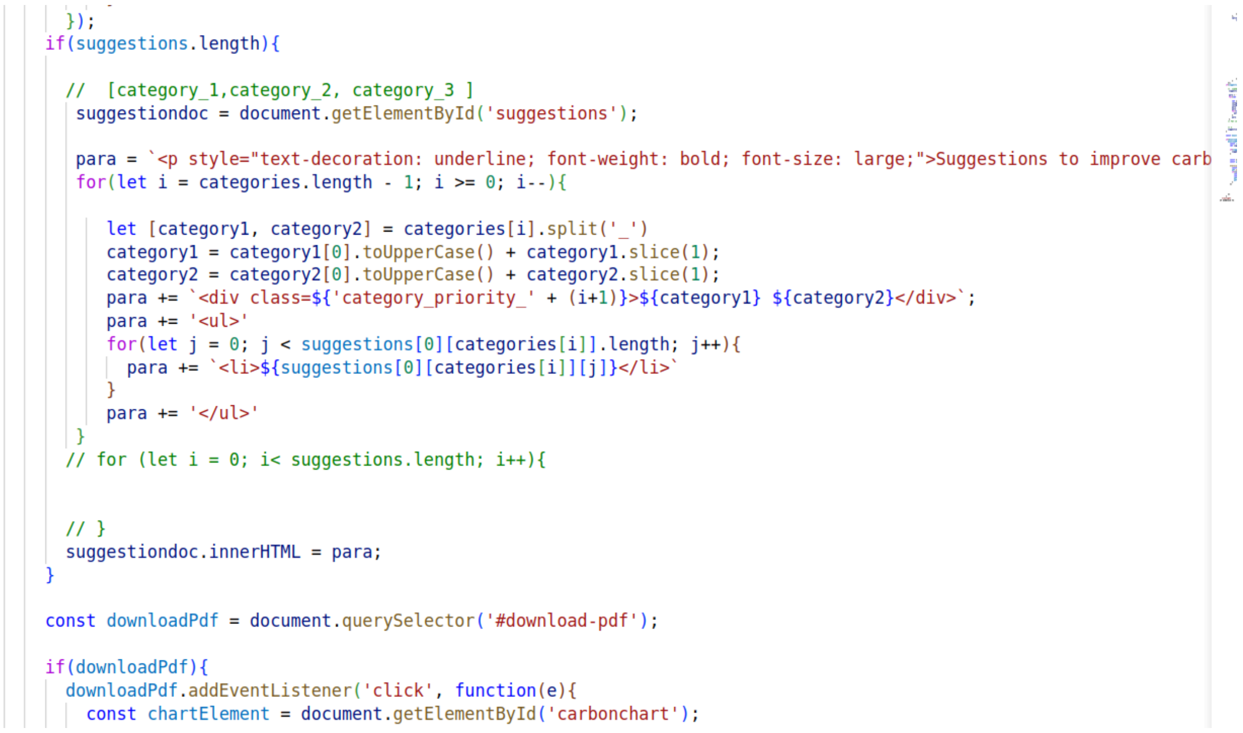
```python
    travel_footprint = (travel_distance / 1) * (fuel_efficiency / 100) * 2.31
suggestions = [{"energy_footprint": [], "waste_footprint": [], "travel_footprint" : []}]



# Energy Usage Suggestions
if electricity_bill > 5000:
    suggestions[0]["energy_footprint"].append("Switch to energy-efficient appliances and LED bulbs.")

if natural_gas_bill > 1000:
    suggestions[0]["energy_footprint"].append("Improve insulation or switch to renewable energy sources.")

if fuel_bill > 1000:
    suggestions[0]["energy_footprint"].append("Use public transport, carpool, or switch to electric vehicles.")

# Waste Reduction Suggestions
if recycling_percentage < 50:

    suggestions[0]["waste_footprint"].append("Increase recycling efforts and compost organic waste.")
if waste_generated > 100:
    suggestions[0]["waste_footprint"].append("Reduce single-use plastics and re-use products.")

# Business Travel Suggestions
if travel_distance > 10000:
    suggestions[0]["travel_footprint"].append("Reduce travel through virtual meetings.")
if fuel_efficiency > 8:
   suggestions[0]["travel_footprint"].append("Switch to fuel-efficient or electric vehicles.")
```



```javascript
  const suggestions = JSON.parse('{{ suggestions | tojson | safe }}');
  const companyName = `{{company_name}}`;
  console.log("suggestions ==>", suggestions);
  console.log("company Name =>", companyName);
  const carbonData = JSON.parse('{{ carbon_data | tojson | safe }}');
  console.log(carbonData);
  let {values, categories} = carbonData
  let i,j,k;

  let minObj = {min: i, med: j, max: k}
  let min = Number.POSITIVE_INFINITY;
  let max = Number.NEGATIVE_INFINITY;
  console.log("minObj 0 =>",minObj)
  for(let p = 0; p < values.length; p++){
    if(min > values[p]){
      min = values[p]
      minObj['min'] = p;
    }

    if(max < values[p]){
      max = values[p];
      minObj['max'] = p;
    }
  }
for(let q = 0; q < values.length; q++){
  if(minObj['min'] === q || minObj['max'] === q) continue;
  minObj['med'] = q;
}
values = [values[minObj['min']], values[minObj['med']], values[minObj['max']]];
categories = [categories[minObj['min']], categories[minObj['med']], categories[minObj['max']]]
console.log("new  values =>", values);
console.log("categories =>", categories)
console.log("minObj 1 =>",minObj)
```



```
const carbonChartId = document.querySelector('#carbonchart')
const context = carbonChartId.getContext('2d');
const carbonChart = new Chart(context, {
        type: 'bar',
        data: {
            labels: categories,
            datasets: [{
                label: 'Carbon Footprint Display Chart',
                data: values,
                backgroundColor: ['green', 'blue', 'red'],
                borderColor: ['rgba(75, 192, 192, 1)', 'rgba(255, 99, 132, 1)', 'rgba(255, 206, 86, 1)', 'rgba(153,
                borderWidth: 1
            }]
        },
        options: {
            scales: {
                y: {
                    beginAtZero: true,
                    title: {
                        display: true,
                        text: "Carbon Footprint (kg CO2e)"
                    },
                    align: 'middle',
                    color: 'black',
                    font: {
                      size: 14,
                      weight: 'bold'
                    }

                },
                x: {

                    title: {
                        display: true,
                        text: `Categories(max shown in red lowest in green)`
                    },
                    align: 'middle'
```

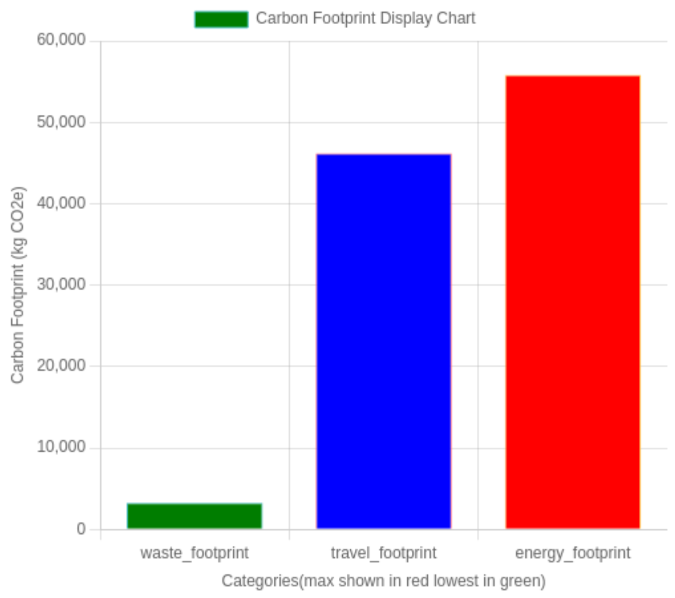


```
    });
    if(suggestions.length){

      //  [category_1,category_2, category_3 ]
       suggestiondoc = document.getElementById('suggestions');

       para = `<p style="text-decoration: underline; font-weight: bold; font-size: large;">Suggestions to improve carb
       for(let i = categories.length - 1; i >= 0; i--){

          let [category1, category2] = categories[i].split('_')
          category1 = category1[0].toUpperCase() + category1.slice(1);
          category2 = category2[0].toUpperCase() + category2.slice(1);
          para += `<div class=${'category_priority_' + (i+1)}>${category1} ${category2}</div>`;
          para += '<ul>'
          for(let j = 0; j < suggestions[0][categories[i]].length; j++){
            para += `<li>${suggestions[0][categories[i]][j]}</li>`
          }
          para += '</ul>'
       }
      // for (let i = 0; i< suggestions.length; i++){


      // }
      suggestiondoc.innerHTML = para;
    }

    const downloadPdf = document.querySelector('#download-pdf');

    if(downloadPdf){
      downloadPdf.addEventListener('click', function(e){
        const chartElement = document.getElementById('carbonchart');
```

In this carbon data and suggestion objects are parsed from backend and then they are used to create the following chart and download functionality

Hi johndoe here is the energy output of your company **GLOBALCORP**

If you want you can edit your information by visiting here



Suggestions to improve carbon footprints for Globalcorp

Energy Footprint

- Improve insulation or switch to renewable energy sources.
- Use public transport, carpool, or switch to electric vehicles.

Travel Footprint

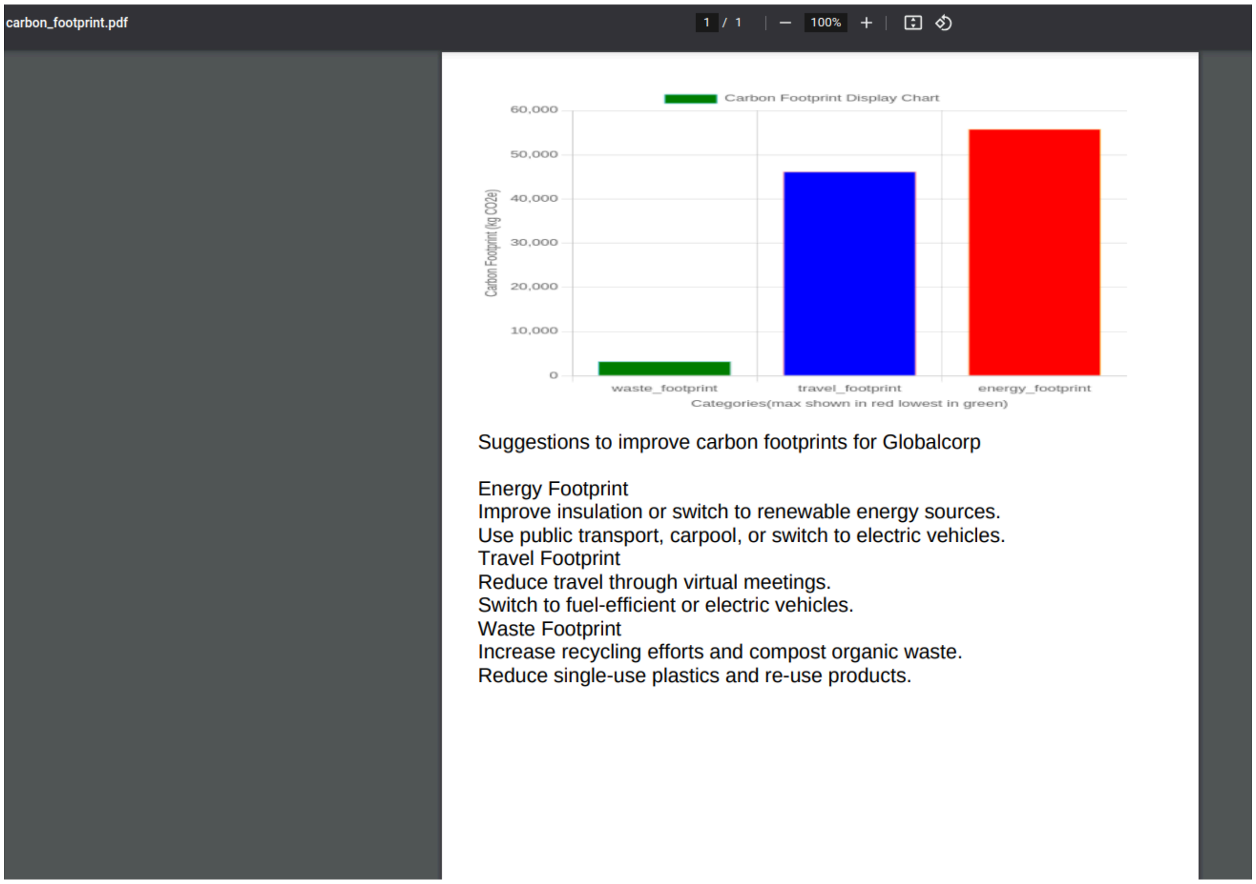
- Reduce travel through virtual meetings.
- Switch to fuel-efficient or electric vehicles.

Waste Footprint

- Increase recycling efforts and compost organic waste.
- Reduce single-use plastics and re-use products.

Download pdf

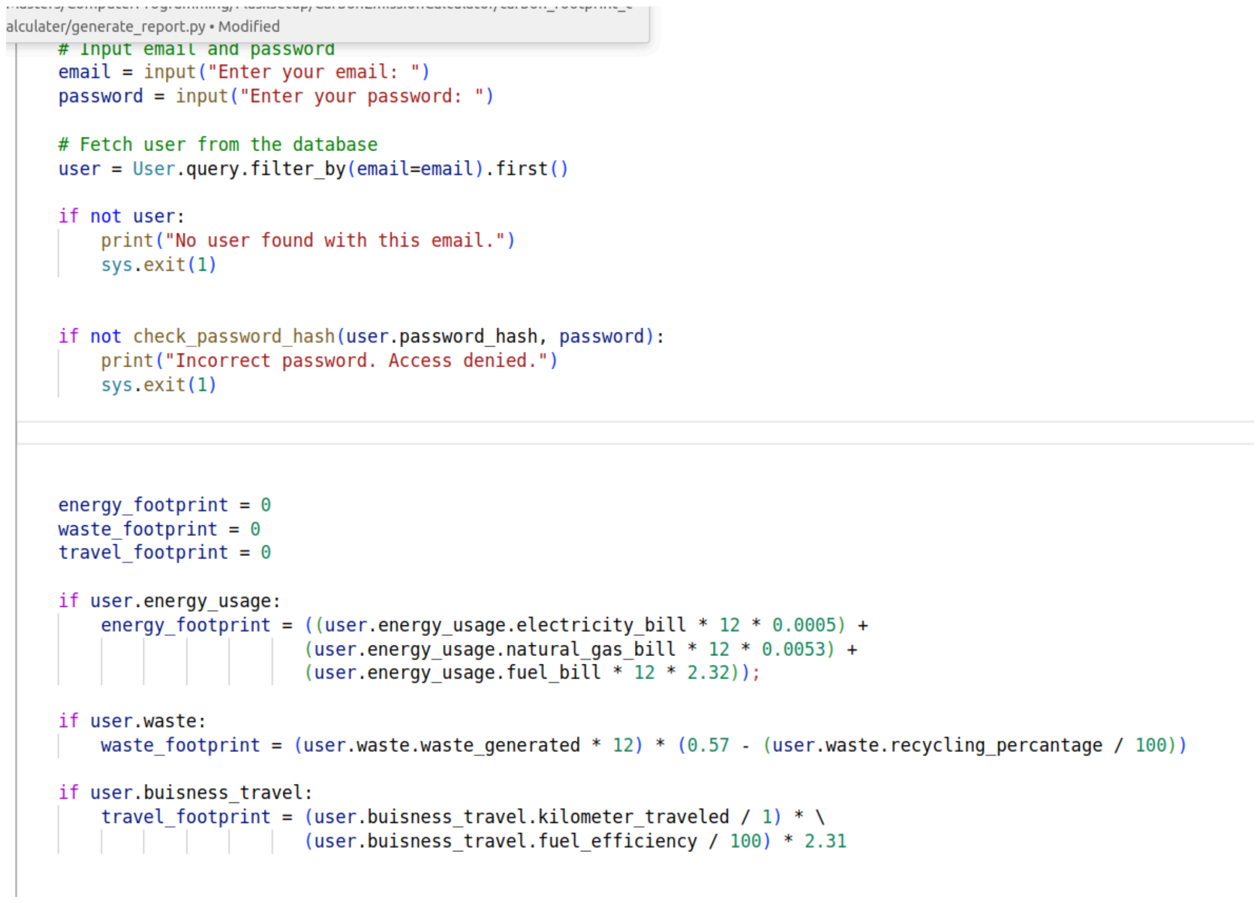
Which has both a bar chart and suggestions. In this bar chart the maximum footprint will be shown in red color in both chart and suggestion box and then blue and then green will follow. These colors will dynamically adjust based on size of data.
After clicking on download here is the downloaded chart



Suggestions to improve carbon footprints for Globalcorp

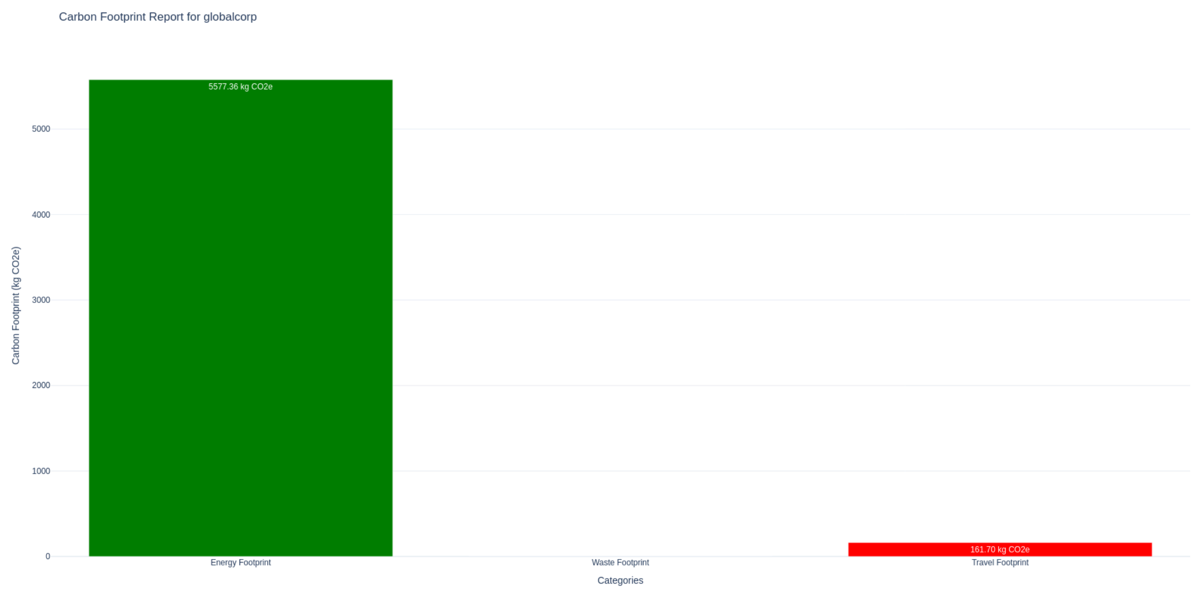
Energy Footprint
Improve insulation or switch to renewable energy sources.
Use public transport, carpool, or switch to electric vehicles.
Travel Footprint
Reduce travel through virtual meetings.
Switch to fuel-efficient or electric vehicles.
Waste Footprint
Increase recycling efforts and compost organic waste.
Reduce single-use plastics and re-use products.

If the user wants to generate the report using python's plotly then there is a separate file generate_report.py after setting up the system they can run the file using this command python3 generate_report.py then this will ask for input in the form of email and password and the password verification will be done in the same way using as in app.py using werkzeug library and a report will be generated here is the screenshot of the code


alculater/generate_report.py • Modified

```
    # Input email and password
    email = input("Enter your email: ")
    password = input("Enter your password: ")

    # Fetch user from the database
    user = User.query.filter_by(email=email).first()

    if not user:
        print("No user found with this email.")
        sys.exit(1)


    if not check_password_hash(user.password_hash, password):
        print("Incorrect password. Access denied.")
        sys.exit(1)



    energy_footprint = 0
    waste_footprint = 0
    travel_footprint = 0

    if user.energy_usage:
        energy_footprint = ((user.energy_usage.electricity_bill * 12 * 0.0005) +
                            (user.energy_usage.natural_gas_bill * 12 * 0.0053) +
                            (user.energy_usage.fuel_bill * 12 * 2.32));

    if user.waste:
        waste_footprint = (user.waste.waste_generated * 12) * (0.57 - (user.waste.recycling_percantage / 100))

    if user.buisness_travel:
        travel_footprint = (user.buisness_travel.kilometer_traveled / 1) * \
                            (user.buisness_travel.fuel_efficiency / 100) * 2.31
```

Here is the sample report generated



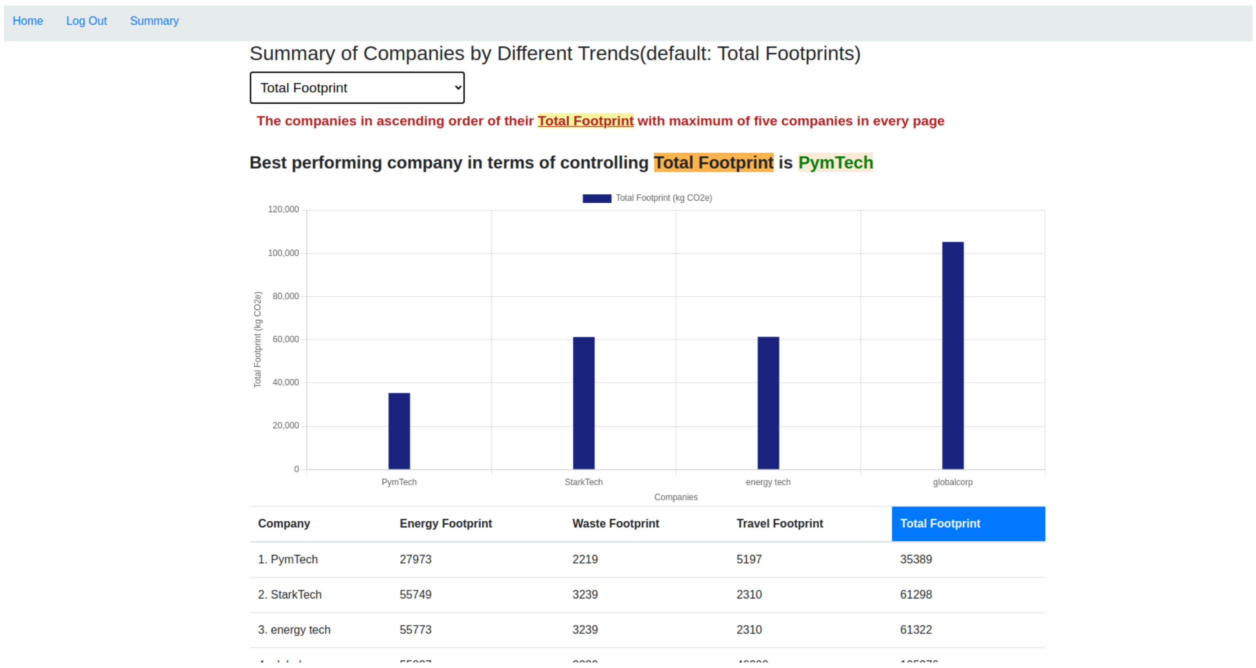
Summary page

url:https://carbonemissioncalculatortool.onrender.com/summary

where user can see the summary of the companies which are registered using this app and also they can compare trends based on

1.**Total footprint**
2. **Energy Footprint**
3. **Waste Footprint**
4. **Travel Footprint**

Summary page overview



Summary of Companies by Different Trends(default: Total Footprints)

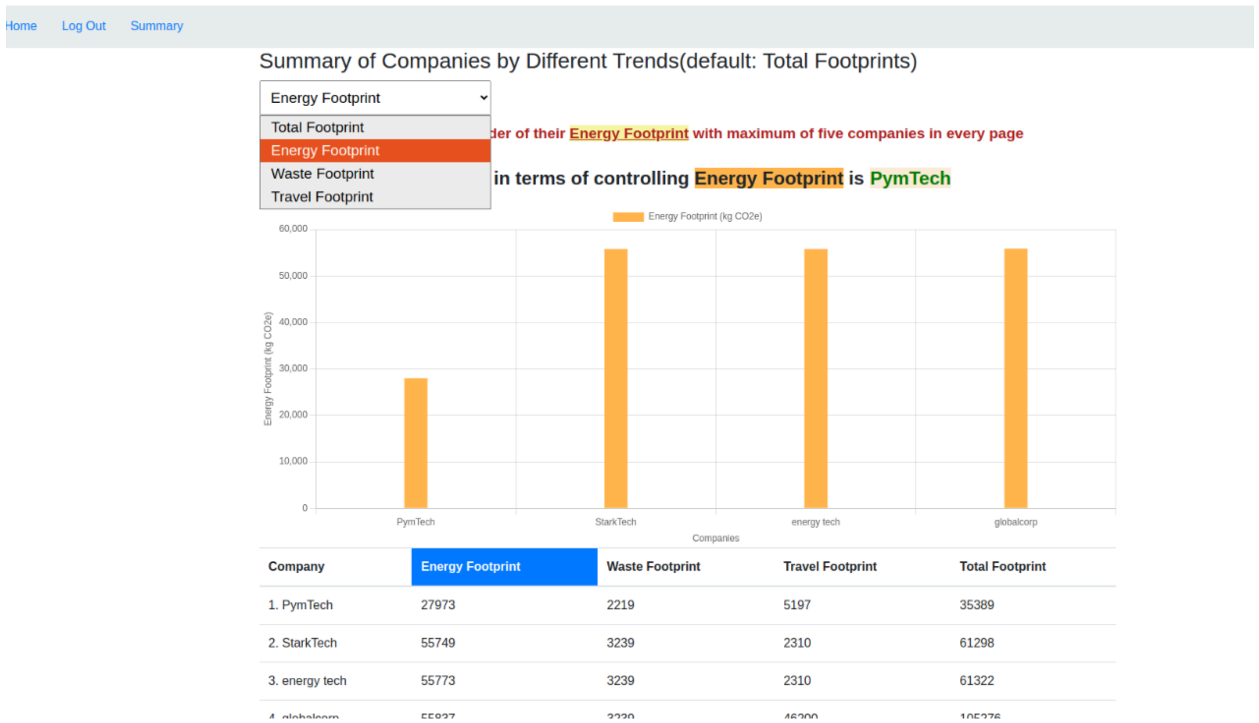
Total Footprint

The companies in ascending order of their Total Footprint with maximum of five companies in every page

Best performing company in terms of controlling Total Footprint is PymTech

Company	Energy Footprint	Waste Footprint	Travel Footprint	Total Footprint
1. PymTech	27973	2219	5197	35389
2. StarkTech	55749	3239	2310	61298
3. energy tech	55773	3239	2310	61322

Here you can see the summary based on different trend where the default selected trend is Total footprint. Selected trend will highlight on the table We can use the dropdown to select different trend and the graph will change based on the trend selected as shown. Similarly this will adjust based on the different trend selected and the name of the company will change based on which company has performed better in their respective trend.



Summary of Companies by Different Trends(default: Total Footprints)

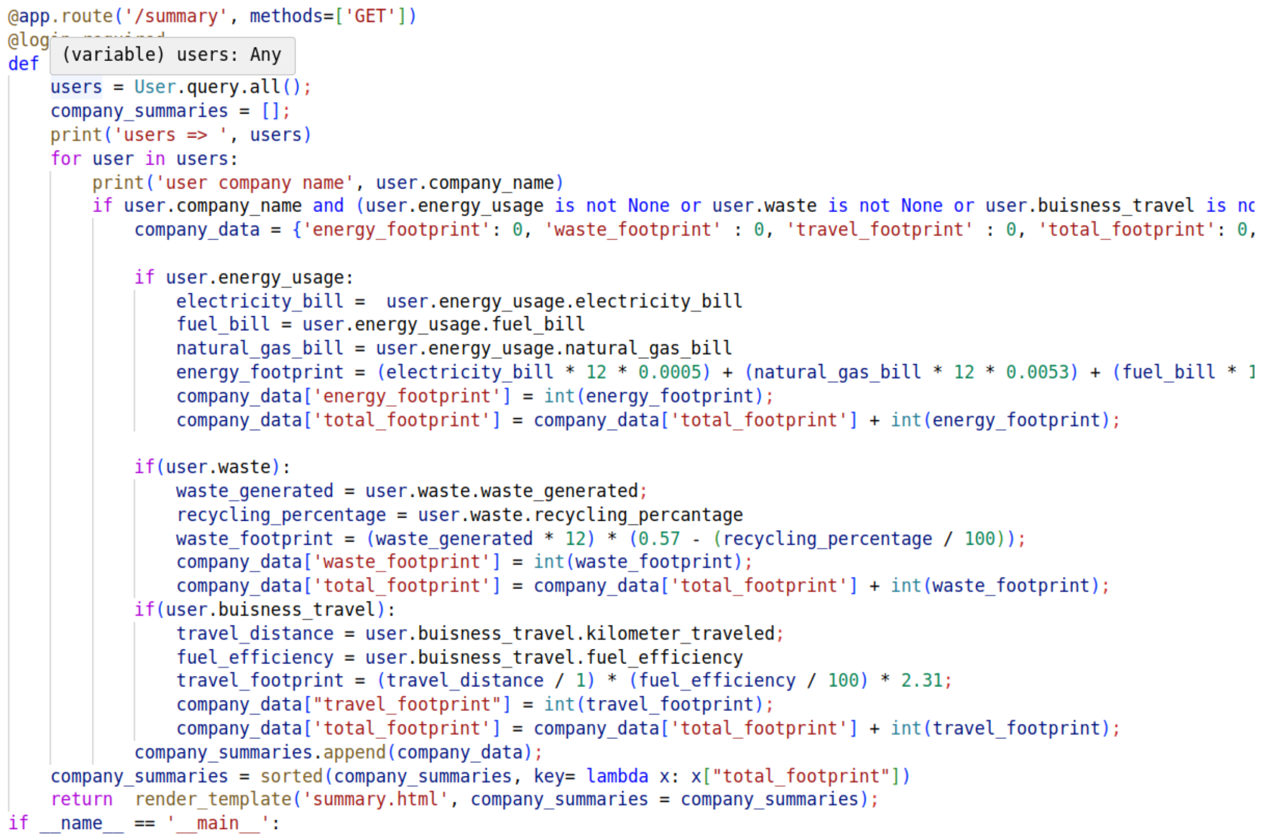
der of their Energy Footprint with maximum of five companies in every page

in terms of controlling Energy Footprint is PymTech

Company	Energy Footprint	Waste Footprint	Travel Footprint	Total Footprint
1. PymTech	27973	2219	5197	35389
2. StarkTech	55749	3239	2310	61298
3. energy tech	55773	3239	2310	61322
4. globalcorp	55837	3239	46200	105276

Pagination is another functionality which is added to handle the design in case companies registered go more than 5 then the data is sorted in ascending order and then top 5 are shown in first page and then remaining in other pages

Code snippet for summary



```
@app.route('/summary', methods=['GET'])
@log
def
    users = User.query.all();
    company_summaries = [];
    print('users => ', users)
    for user in users:
        print('user company name', user.company_name)
        if user.company_name and (user.energy_usage is not None or user.waste is not None or user.buisness_travel is nc
            company_data = {'energy_footprint': 0, 'waste_footprint' : 0, 'travel_footprint' : 0, 'total_footprint': 0,

            if user.energy_usage:
                electricity_bill =  user.energy_usage.electricity_bill
                fuel_bill = user.energy_usage.fuel_bill
                natural_gas_bill = user.energy_usage.natural_gas_bill
                energy_footprint = (electricity_bill * 12 * 0.0005) + (natural_gas_bill * 12 * 0.0053) + (fuel_bill * 1
                company_data['energy_footprint'] = int(energy_footprint);
                company_data['total_footprint'] = company_data['total_footprint'] + int(energy_footprint);

            if(user.waste):
                waste_generated = user.waste.waste_generated;
                recycling_percentage = user.waste.recycling_percantage
                waste_footprint = (waste_generated * 12) * (0.57 - (recycling_percentage / 100));
                company_data['waste_footprint'] = int(waste_footprint);
                company_data['total_footprint'] = company_data['total_footprint'] + int(waste_footprint);
            if(user.buisness_travel):
                travel_distance = user.buisness_travel.kilometer_traveled;
                fuel_efficiency = user.buisness_travel.fuel_efficiency
                travel_footprint = (travel_distance / 1) * (fuel_efficiency / 100) * 2.31;
                company_data["travel_footprint"] = int(travel_footprint);
                company_data['total_footprint'] = company_data['total_footprint'] + int(travel_footprint);
            company_summaries.append(company_data);
    company_summaries = sorted(company_summaries, key= lambda x: x["total_footprint"])
    return  render_template('summary.html', company_summaries = company_summaries);
if __name__ == '__main__':
```

This method retrieves the users from database and parses them to calculate the respective energy waste and travel footprint and then adds it to their individual total footprint score
Then the company summary dictionary is created using the data which is then sorted by keeping total foot print in ascending order which is then sent to frontend where this data is handled by javascript to create the dynamic charts

Tech Stack

Backend

Python 3.8
Flask-python framework
Flask-SqlAlchemy: For connecting to data based on ORM
SQLLite: Database for local and deployed environment
Flask-migrate for database migrations

Frontend

HTML5, CSS3, Bootstrap, Javascript
Chart.js for visualizing carbon footprint and summary
HTML2canvas for pdf

Suggestions

As the application grows in number its better to migrate to some more robust database like Postgresql or mysql. However, right now sqlite will works fine.

Conclusion

To conclude this project offers a robust software infrastructure which can help companies monitor their individual footprint as well as visualize other company's footprint. This can be beneficial for them in reducing the unnecessary usage of energy and recycling waste and hence contributing more towards a greener earth.

Personally, this application helped me learn about flask in great length. Also coming up with schema design was a challenge. I built various models and different schemas which didn't work. Later I came up with this db schema which was very useful in saving the data related to the user and showing them on a chart.