VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM - 590014

A DBMS Mini-Project Report
On

“Carbon Emission Calculator”

A Mini-project report submitted in partial fulfillment of the requirements for the award of the Bachelor ***of Artificial Intelligence and Machine Learning Engineering*** *of Visvesvaraya Technological University, Belgaum.*

Submitted by:
A S NAVYASHREE(1DT20AI001)
AND
DEEPTI HEGDE (1DT20AI018)

Under the Guidance of:
Dr.Shivaprasad A C(Asst.Prof.Dept of AIML)
and
Prof. Raghava M S (Asst. Prof. Dept of AIML)

Department of Artificial Intelligence and Machine Learning
DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT
Kanakapura Road, Udayapura, Bangalore 2019-2020
(Accredited by NBA, New Delhi for 3 years validity: 26-07-2018 to 30-06-2021)
DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT,
Kanakapura Road, Udayapura, Bangalore
DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

CERTIFICATE

This is to certify that the Mini-Project on Database Management System (DBMS) titled **“COUNT_THE_CARBON”** has been successfully carried out by **A S NAVYASHREE(1DT20AI001) and DEEPTI HEGDE(1DT20AI018),** bonafide students of **Dayananda Sagar Academy of Technology and Management** in partial fulfillment of the requirements for the award of degree in **Bachelor of Engineering in Artificial Intelligence and Machine Learning** of **Visvesvaraya Technological University, Belgaum** during academic year 2022-2023. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements with respect to the project work for said degree.

GUIDES:

1. **Prof. Raghava M.S**
2. **Dr. Shivaprasad A C**

Examiners: Signature with Date **Dr. Sandhya .N**

1.

2. **(HoD Artificial Intelligence and Machine Learning) Signature with Date**

ACKNOWLEDGEMENT

We present before you our project titled **"COUNT_THE_CARBON' USING PYTHON AND MYSQL.** We express our gratitude towards our institution, **DAYANANDA SAGAR ACADEMY OF TECHNOLOGY AND MANAGEMENT** for providing us with the knowledge and support required for completing the project.

We wish to express a sincere thanks to our respected Principal **Dr. M. Ravishankar** for his support.

We express our deepest gratitude and special thanks to **Dr. Sandhya .N, H.O.D, Dept. Of Artificial Intelligence and Machine Learning** for her guidance and encouragement.

We acknowledge the guidance and constant encouragement, and express our gratitude to our mini- project guides, **Prof. Raghava .M.S (Dept. of AIML) and Dr. Shivaprasad A C (Prof. Department of AIML)**

A S NAVYASHREE(1DT20AI001)
DEEPTI HEGDE(1DT20AI018)

ABSTRACT

‘COUNT THE CARBON” aims to calculate the carbon emission from various region types to generate awareness and keep track of carbon usage .

Carbon emissions refer to the release of carbon dioxide and other greenhouse gases into the atmosphere. These emissions are primarily caused by human activities such as the burning of fossil fuels for energy, deforestation, and industrial processes.

Carbon emissions have been linked to climate change, as the buildup of these gases in the atmosphere traps heat and leads to rising temperatures and changes in weather patterns. Reducing carbon emissions is essential to slowing the pace of climate change and mitigating its impacts. This can be achieved through a variety of means, such as increasing the use of renewable energy sources, implementing carbon pricing mechanisms, and investing in energy efficiency and conservation.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Background

‘Count The Carbon’ is a calculator application that calculates and keeps track of carbon emissions from various sources for pollution control. The calculator takes inputs for the quantity of sources used and gives the corresponding carbon emission in Kg of CO2 emitted.

1.2 Problem Definition

Carbon emissions are calculated to understand and measure the amount of greenhouse gasses that are being released into the atmosphere as a result of human activities. This information is important for several reasons:

Climate change: Carbon emissions are a major contributor to climate change, and by measuring them, we can understand the extent to which human activities are contributing to global warming and changes in weather patterns.

Policy and regulation: Governments and organizations use carbon emission data to develop policies and regulations aimed at reducing emissions and mitigating the impacts of climate change.

Corporate and individual responsibility: Carbon emissions are also used to track the environmental impact of companies, organizations, and individuals. This information can be used to hold them accountable for their actions and to encourage them to reduce their emissions.

Compliance with international agreements: Many countries have committed to reducing their carbon emissions as part of international agreements such as the Paris Agreement. Measuring emissions is necessary to track progress towards these targets and to report on emissions to international bodies.

Progress tracking: Measuring carbon emissions over time allows to track the progress made in reducing emissions and achieving goals set by governments, organizations, and individuals.

1.3 Motivation

Carbon emissions can have a range of negative impacts on the environment, including air pollution and acid rain, which can harm human health and damage ecosystems. Controlling emissions can help to protect the environment and preserve natural resources for future generations.

Knowing the sources for proper control of carbon emission can help to better preserve the environment.

1.4 Objective

Reduce greenhouse gas emissions: The primary goal of carbon emission control is to reduce the amount of greenhouse gasses that are released into the atmosphere. This can be achieved through a variety of means, such as increasing the use of renewable energy sources, implementing carbon pricing mechanisms, and investing in energy efficiency and conservation.

Slow the pace of climate change: By reducing carbon emissions, the aim is to slow down the pace of climate change and to mitigate its impacts on human societies and ecosystems.

Meet international commitments: Many countries have committed to reducing their carbon emissions as part of international agreements such as the Paris Agreement. Carbon emission control measures are put in place to meet these commitments.

Promote sustainable development: Carbon emission control can also promote sustainable development by reducing dependence on fossil fuels and promoting the use of renewable energy sources, which can have positive impacts on local economies, communities and the environment.

Increase energy security: by reducing carbon emissions, countries can increase their energy security by reducing dependence on fossil fuels, which are subject to price volatility, geopolitical tensions, and supply disruptions.

1.5 Scope of the project

The scope of a carbon calculator can vary depending on the specific tool and its intended use, but generally, it includes the ability to calculate the carbon emissions associated with different activities or product

CHAPTER 2

REQUIREMENTS

The requirements for the project are broken down into two major categories, namely hardware and software requirements.

The hardware requirements specifies the minimum hardware requirements for a system running our project. The software requirements specifies the essential software needed to build and run the project.

2.1 Hardware Requirements

The system is designed to run light and is capable of running on the most basic hardware.

- Processor - Intel® Pentium® Silver N5030 Processor or equivalent.
- Processor Speed - Base frequency 1.10 max frequency up to 3.10 GHz
- System Storage - 100 GB or greater
- RAM - 4GB or greater

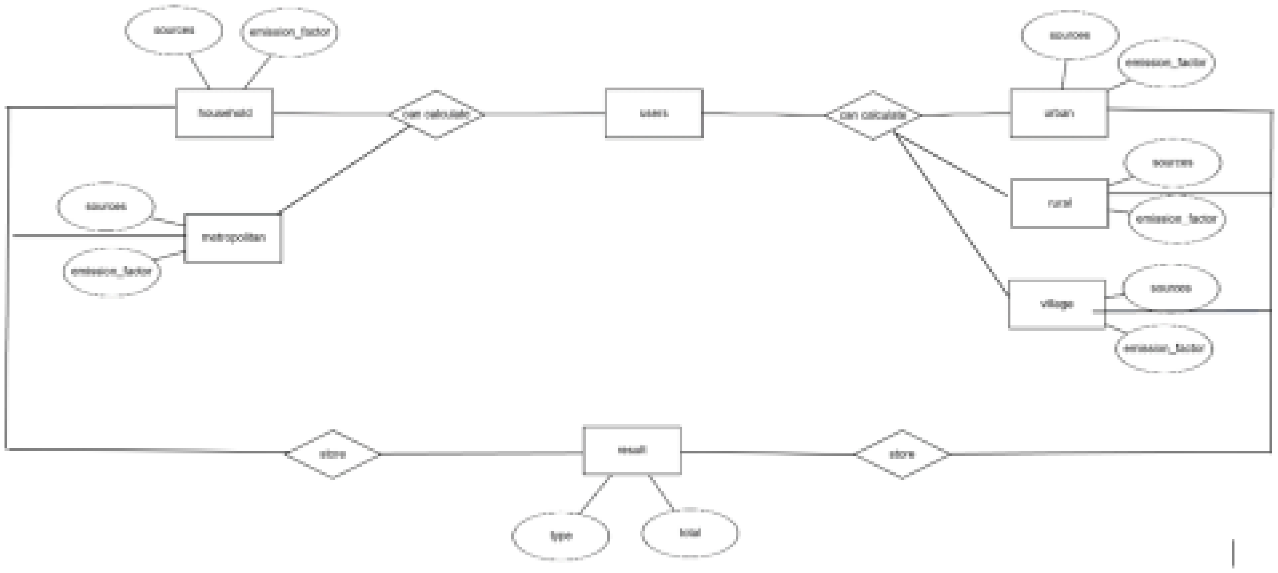
2.2 Software Requirements

- Operating System : Windows7 or later/IOS/Linux
- Language Used : Python
- Database : MySQL
- User Interface Design : Tkinter and CustomTkinter

CHAPTER 3

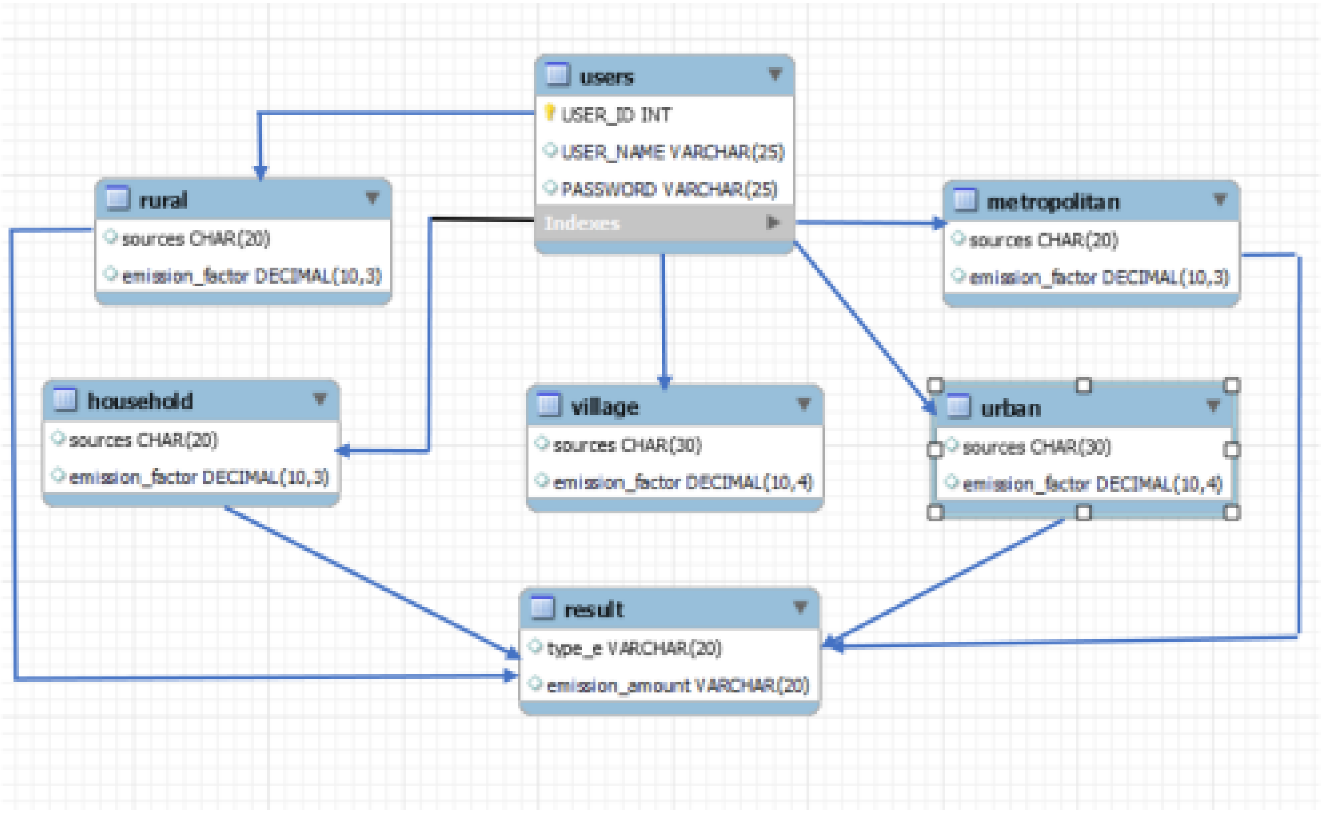
DATABASE DESIGN

3.1.1 E-R Diagram



3.1.2 Database Schema

Database:



users
USER_ID INT
USER_NAME VARCHAR(25)
PASSWORD VARCHAR(25)
Indexes

rural
sources CHAR(20)
emission_factor DECIMAL(10,3)

metropolitan
sources CHAR(20)
emission_factor DECIMAL(10,3)

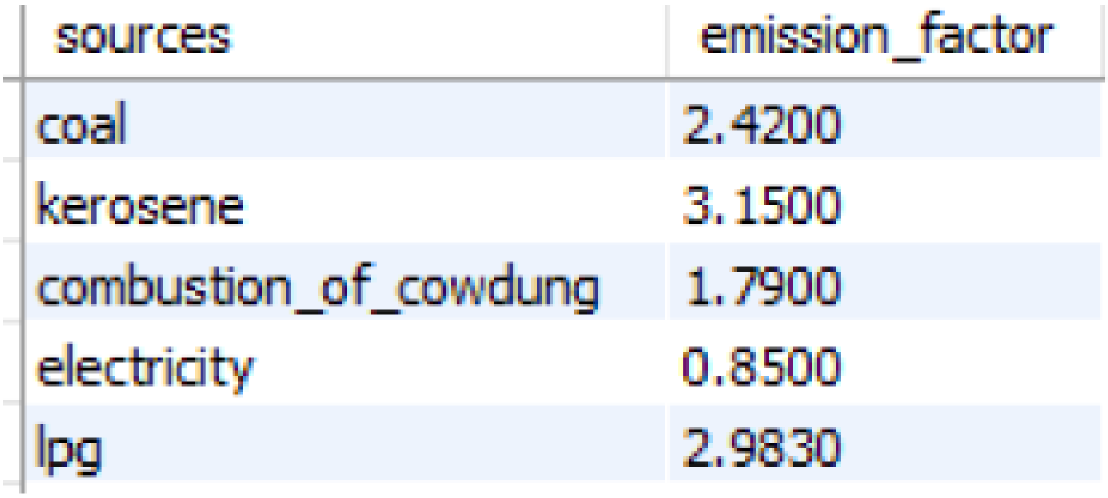
household
sources CHAR(20)
emission_factor DECIMAL(10,3)

village
sources CHAR(30)
emission_factor DECIMAL(10,4)

urban
sources CHAR(30)
emission_factor DECIMAL(10,4)

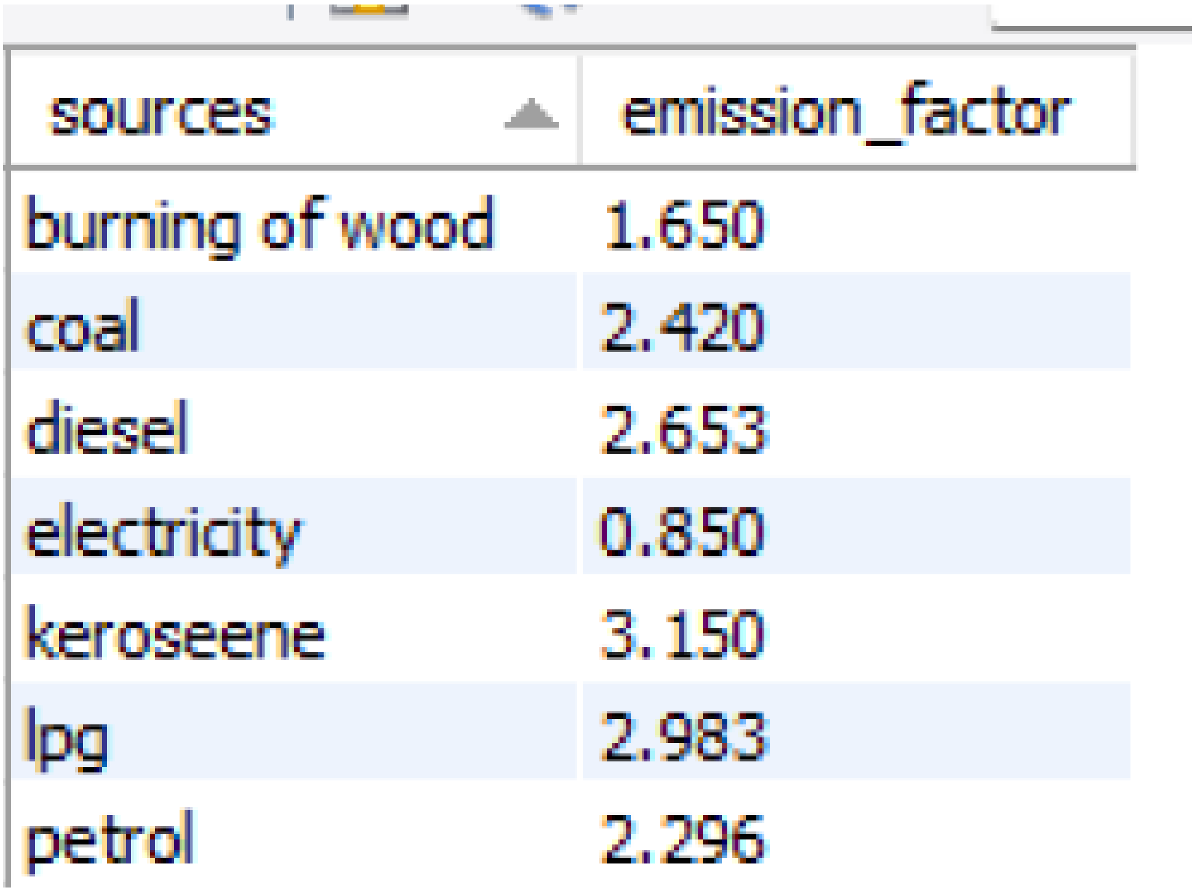
result
type_e VARCHAR(20)
emission_amount VARCHAR(20)

Table:household



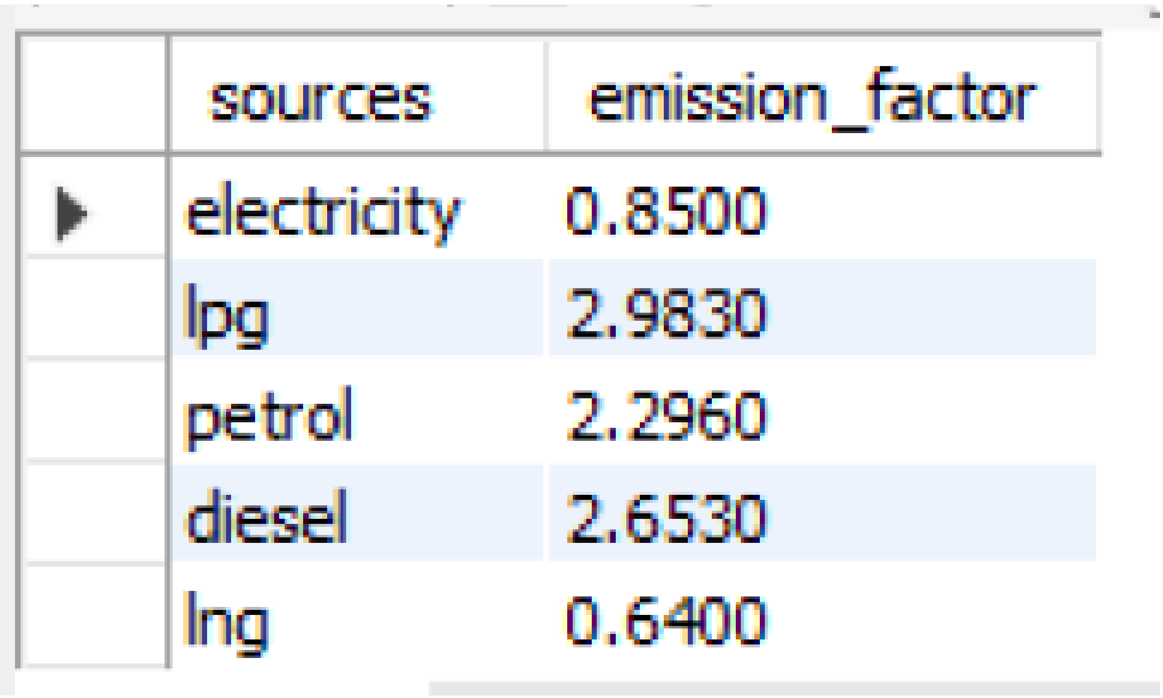
sources	emission_factor
coal	2.4200
kerosene	3.1500
combustion_of_cowdung	1.7900
electricity	0.8500
lpg	2.9830

Table:rural



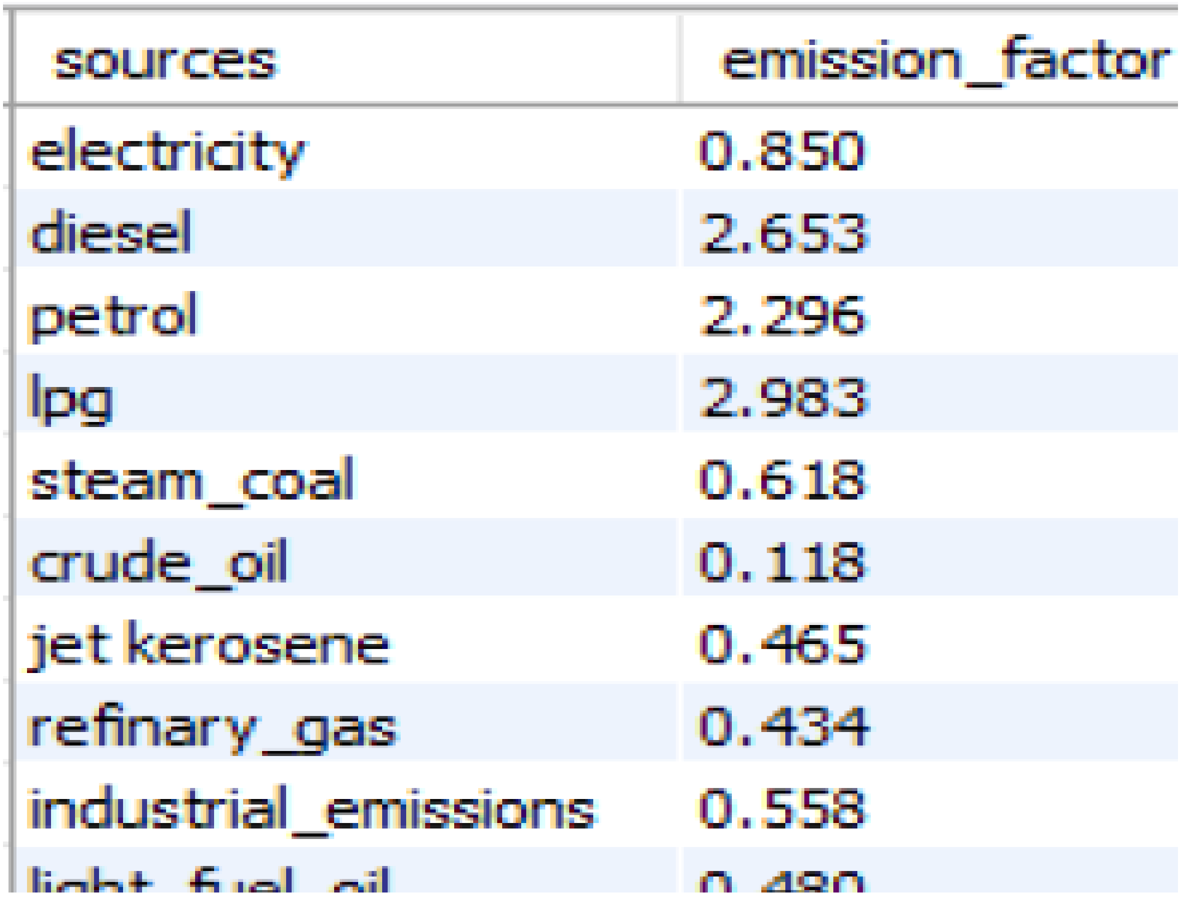
sources	emission_factor
burning of wood	1.650
coal	2.420
diesel	2.653
electricity	0.850
keroseene	3.150
lpg	2.983
petrol	2.296

Table:Urban

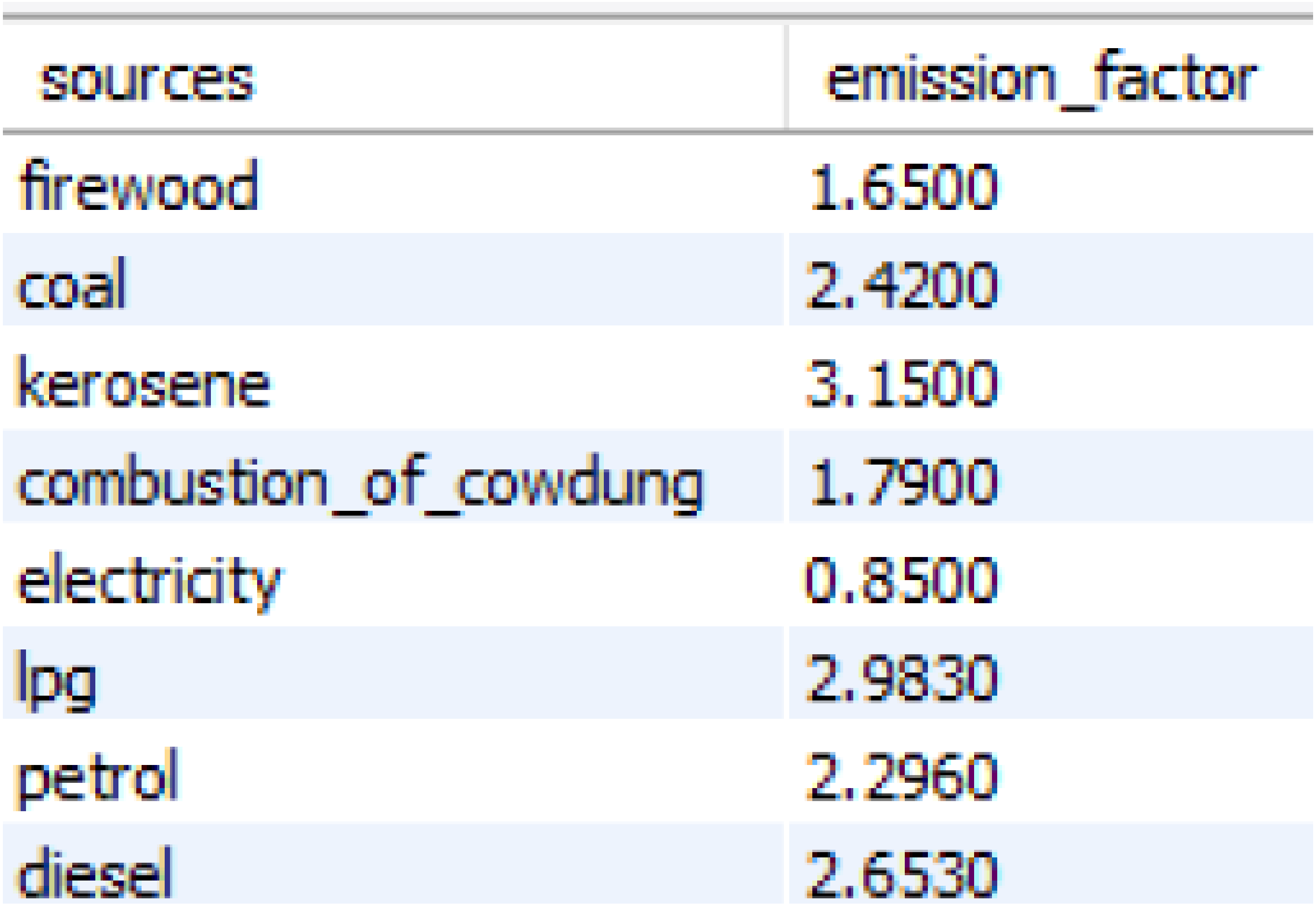


	sources	emission_factor
▶	electricity	0.8500
	lpg	2.9830
	petrol	2.2960
	diesel	2.6530
	lng	0.6400

Table :metropolitan



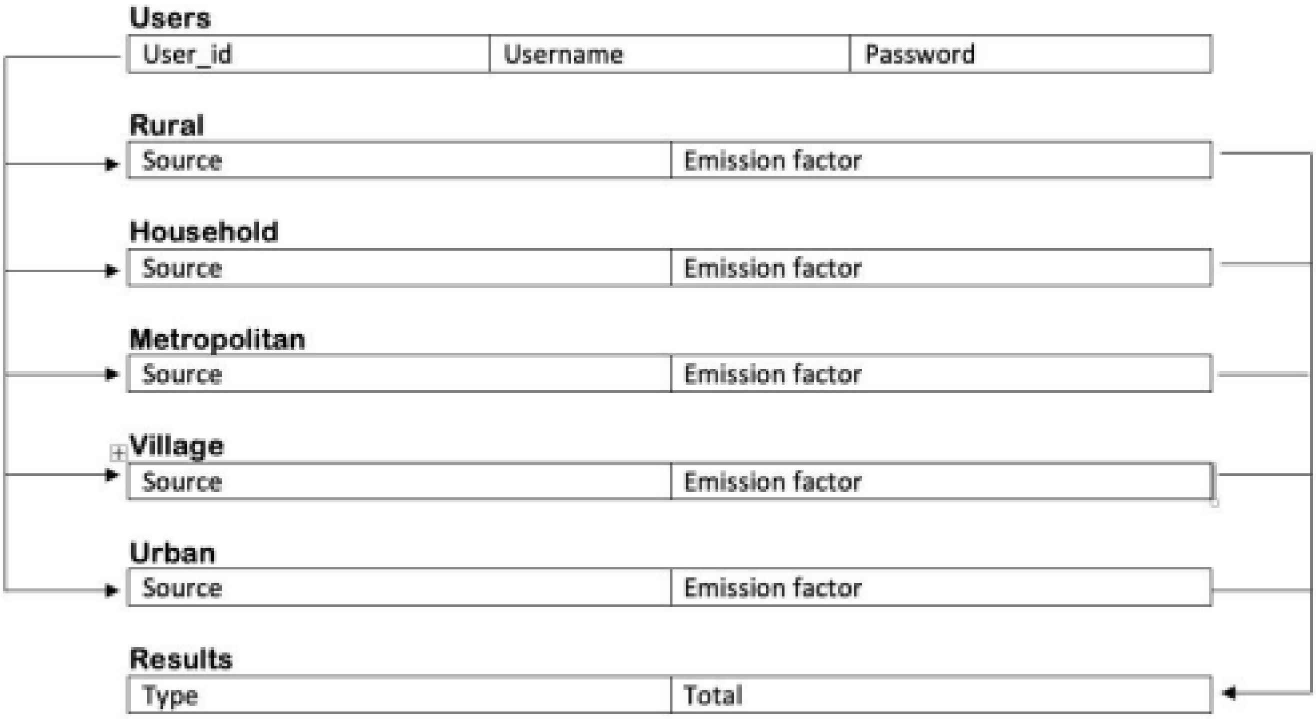
sources	emission_factor
electricity	0.850
diesel	2.653
petrol	2.296
lpg	2.983
steam_coal	0.618
crude_oil	0.118
jet kerosene	0.465
refinary_gas	0.434
industrial_emissions	0.558
light_fuel_oil	0.480



sources	emission_factor
firewood	1.6500
coal	2.4200
kerosene	3.1500
combustion_of_cowdung	1.7900
electricity	0.8500
lpg	2.9830
petrol	2.2960
diesel	2.6530

3.1.3 Relational Schema

Database:



Users

User_id	Username	Password

Rural

Source	Emission factor

Household

Source	Emission factor

Metropolitan

Source	Emission factor

Village

Source	Emission factor

Urban

Source	Emission factor

Results

Type	Total

3.2 Database Normalization

3.2.1 First Normal Form

All the Relations are designed in such a way that it has no repeating groups. Hence all tables are in 1st Normal Form.

3.2.2 Second Normal Form

A relation is said to be in second normal form if it is already in first normal form and it has no partial dependency. All the tables in the database are designed in such a way that there is no partial dependency. Hence all tables are in 2nd Normal Form.

3.2.3 Third Normal Form

A relation is said to be in third normal form if it is already in 1st and 2nd Normal Form and has no transitive dependency. All the tables in the database are designed in such a way that there is no transitive dependency. Hence all tables are in 3rd Normal Norm.

3.3 User Interface

The User Interface of the System is Open souce.

3.3.1 USER REGISTRATION MODULE

3.3.1.1 User Registration

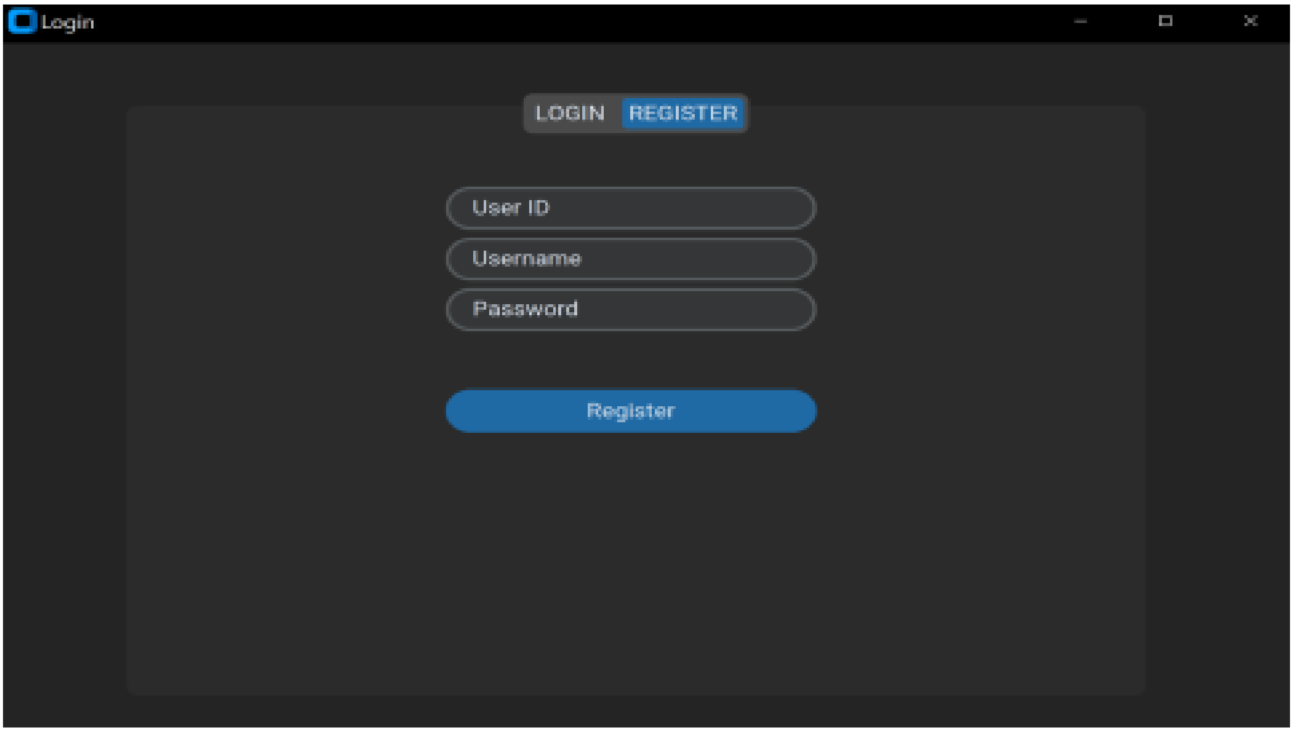


Figure 3.3.1.1: User registration module

3.3.1.2 User Login

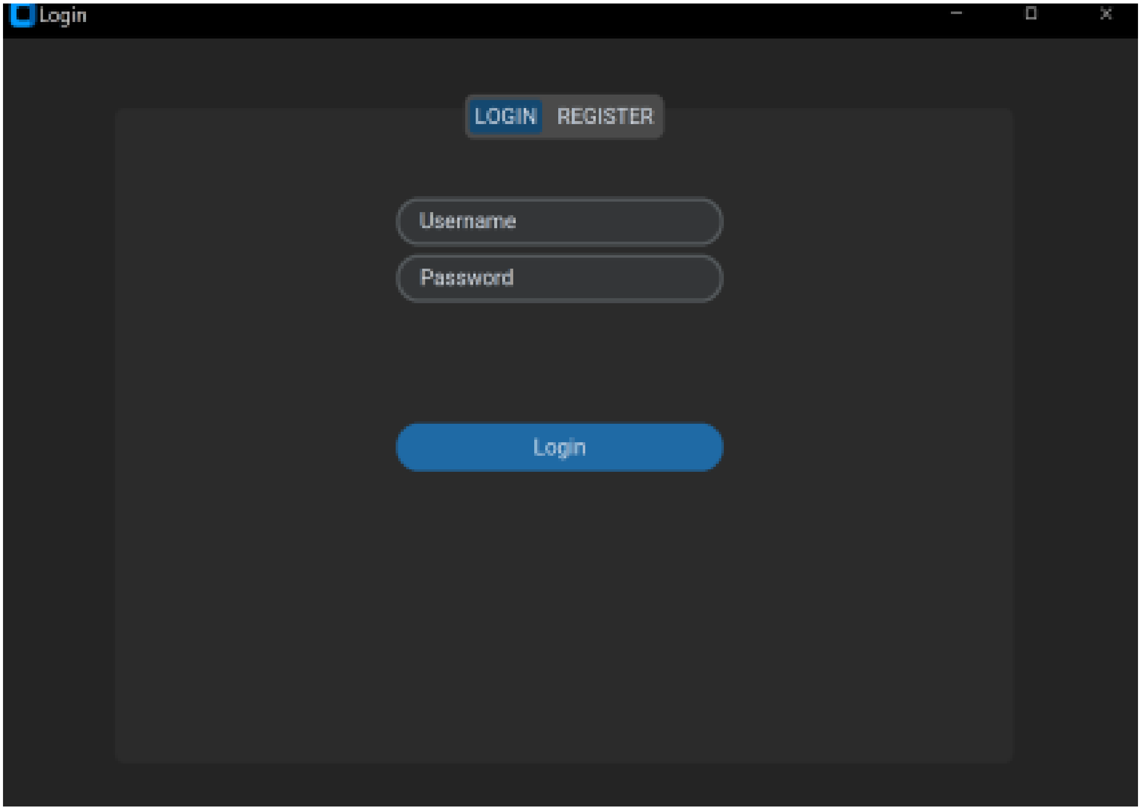
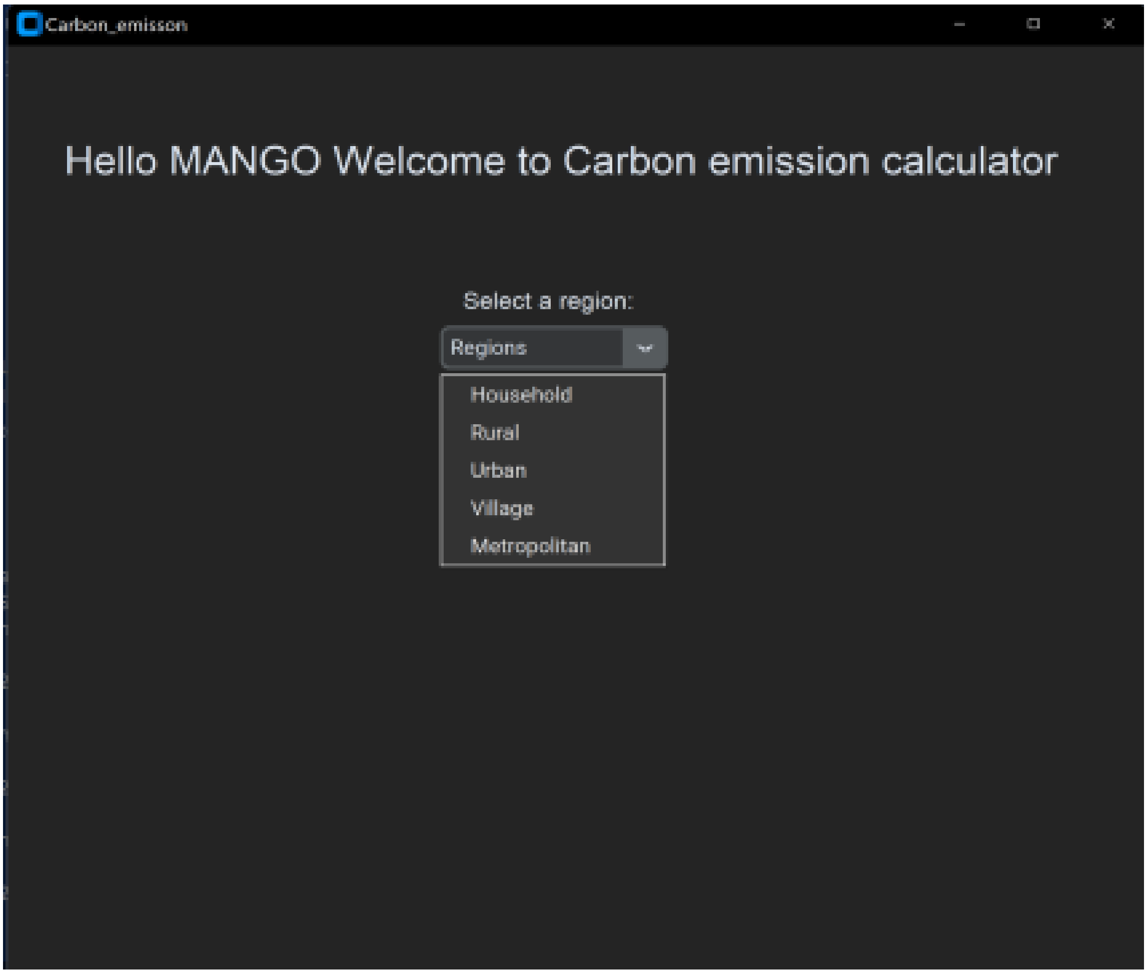


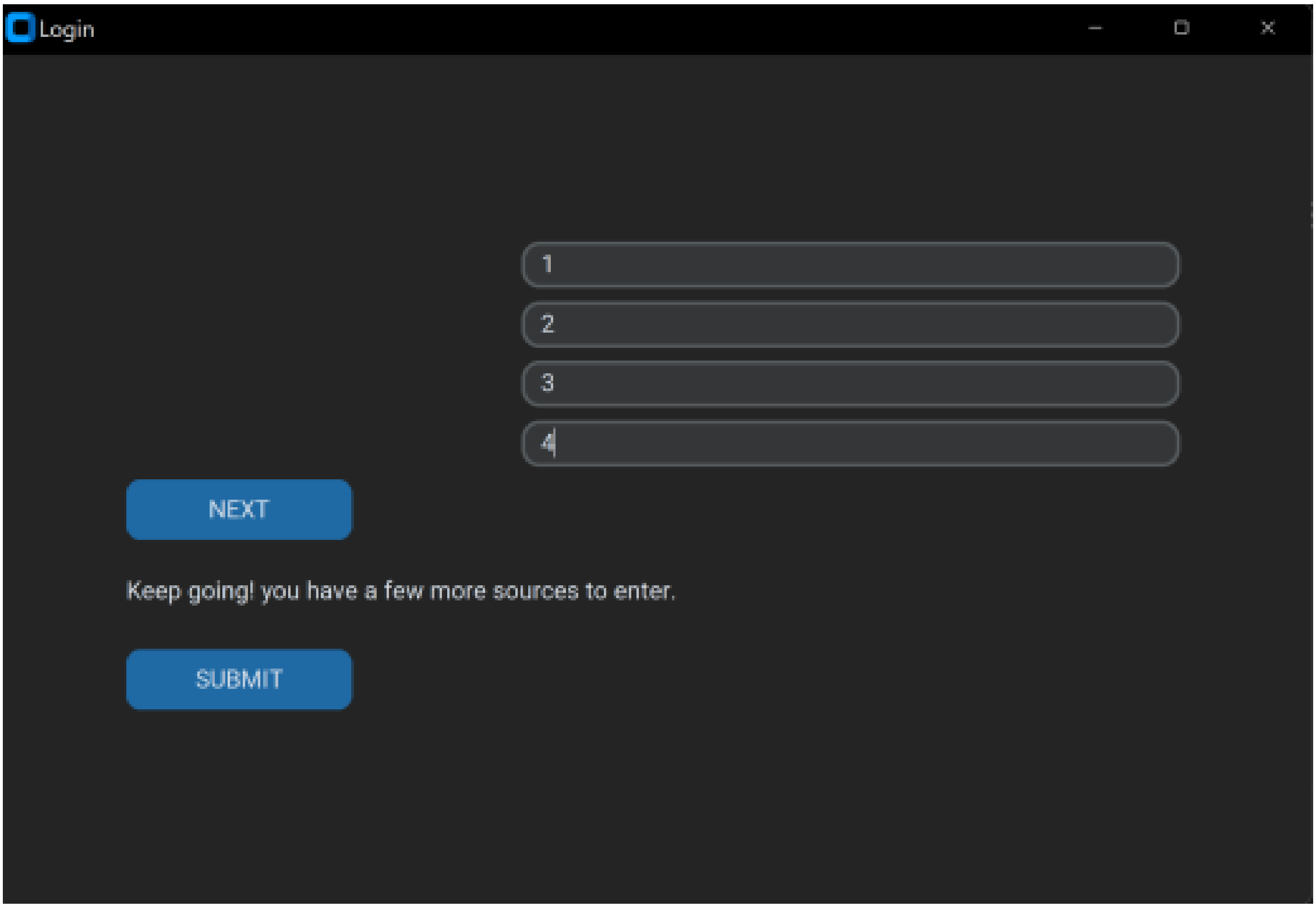
Figure 3.3.1.2: User Login

3.3.2 USER OPERATIONS MODULE



Hello MANGO Welcome to Carbon emission calculator

Figure 3.3.2 user operations module



Keep going! you have a few more sources to enter.

Figure 3.3.2.1 user operations module

3.3.2.3 user operations module

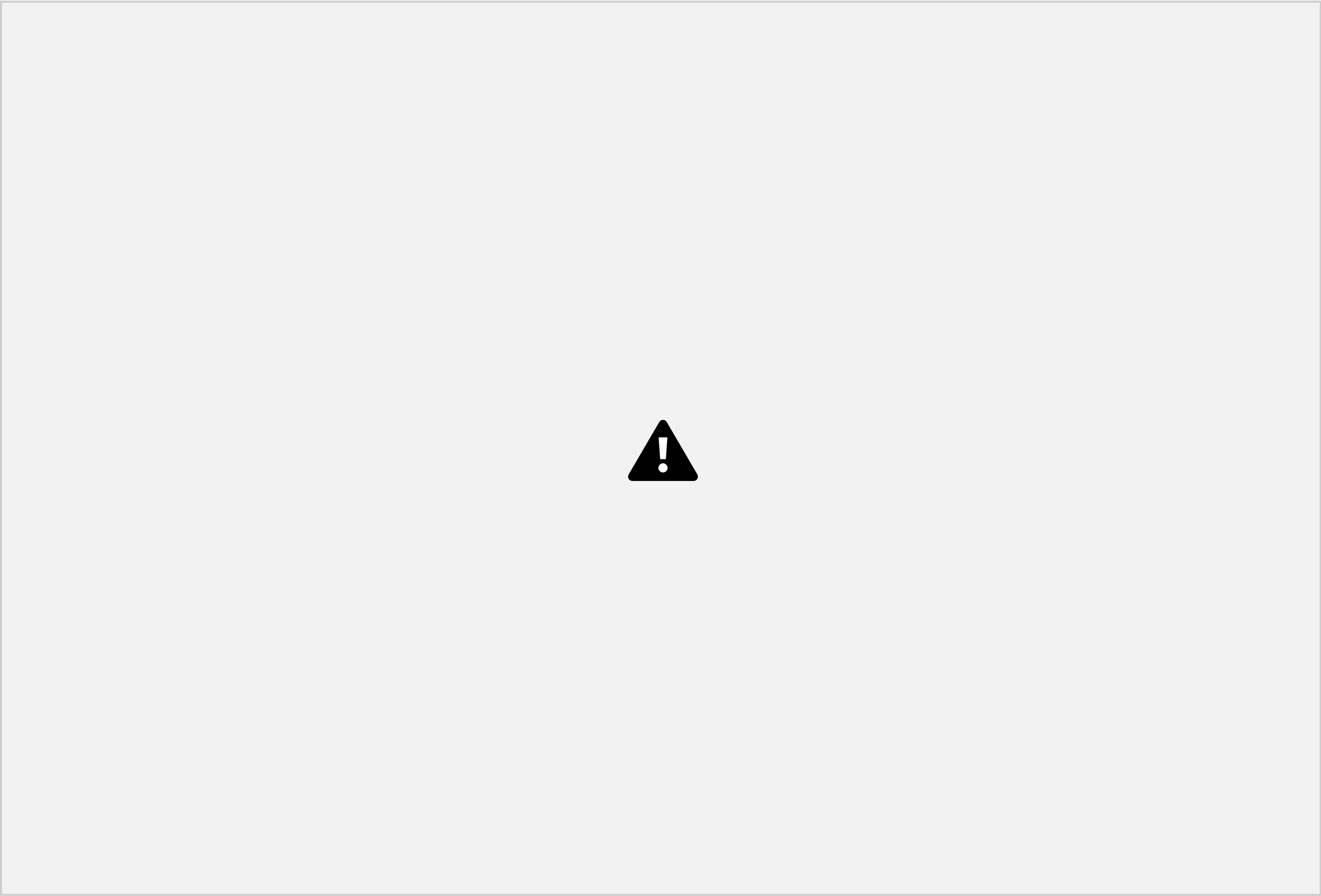
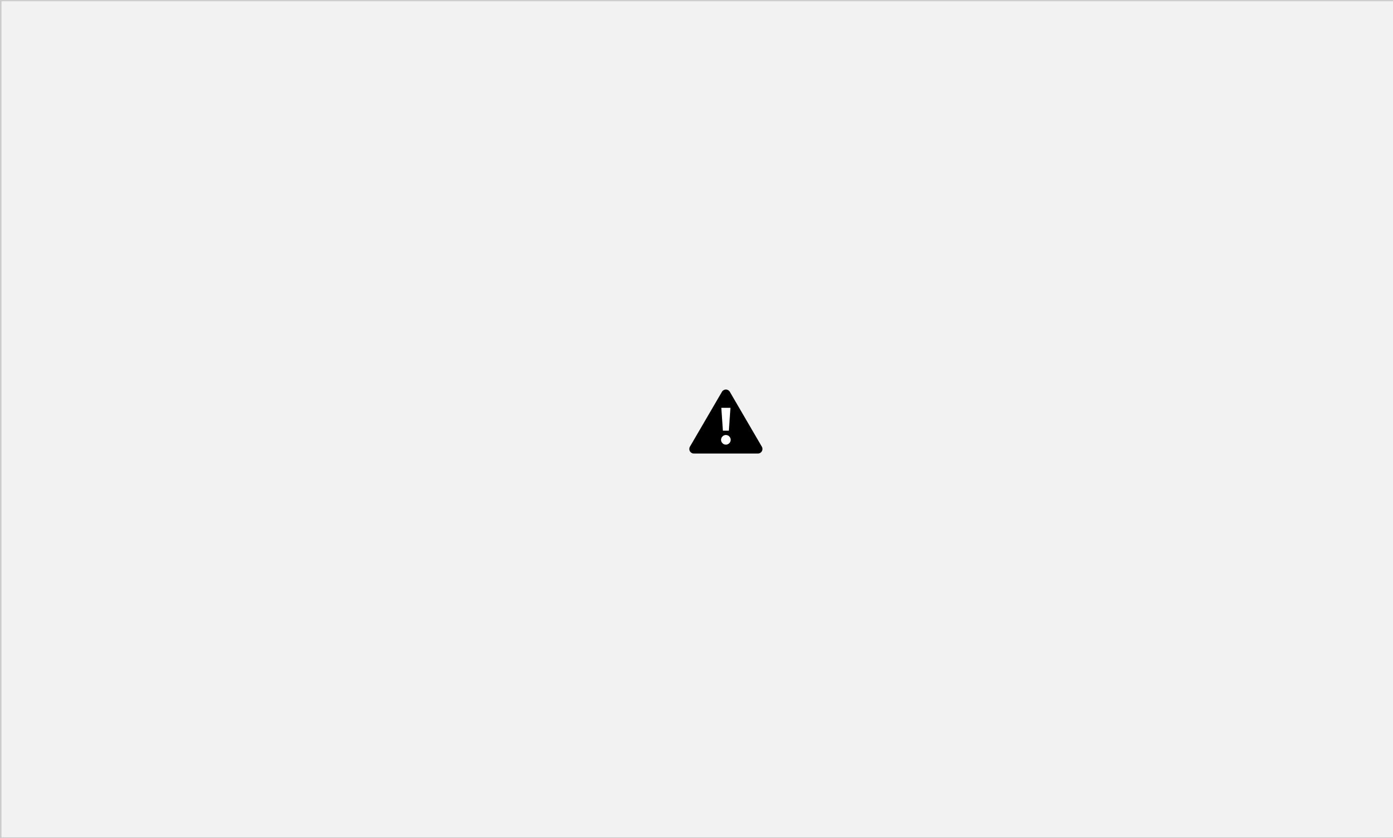


Figure 3.3.2.3User operations module

.3.2.4 carb



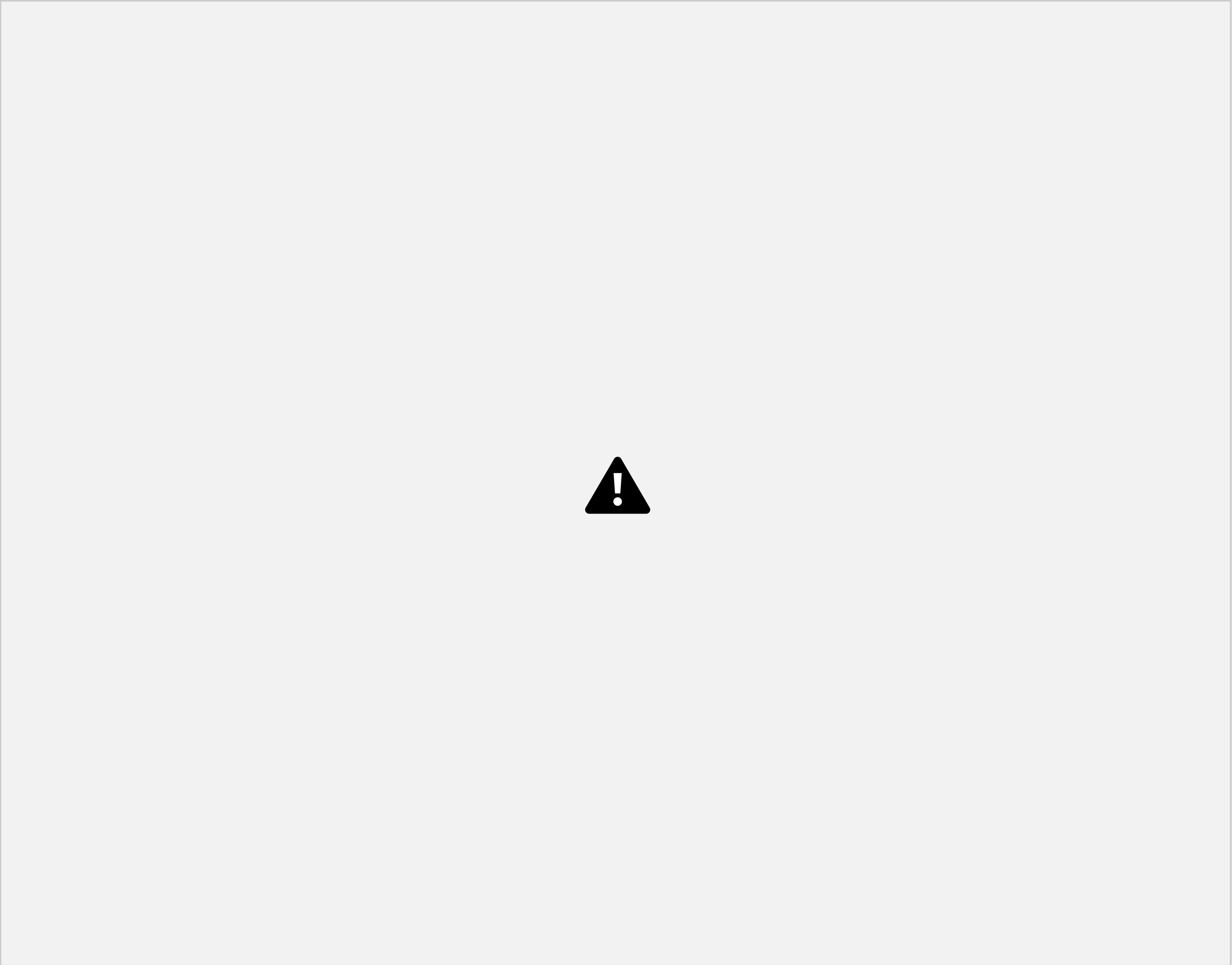


Figure 3.3.2.4: Carbon emission calculation

CHAPTER 4

IMPLEMENTATION

4.1.1 User Registration Module

Process Name	● User Registration
Process Number	● 1.1
Input	● User_id ● User_name ● User_password
Output	● User registered successfully
Error Condition	● User id already taken

4.1.2 User Login Module

Process Name	● User Login
Process Number	● 1.2
Input	● User_name ● User_password
Output	● Successfully Logged In
Error Condition	●Data not found

4.1.3 User operations module

Process Name	●carbon_emission
Process Number	● 2.1
Input	Choose the region ● Household ● Village ● Rural ● Urban ● Metropolitan

Error Condition	●—-----------------

4.1.4 User operations module

Process Name	● calculate the carbon emission
Process Number	● 2.2
Input	The amount of each source used in kg of co2
Output	● Displays the total carbon emission
Error Condition	●—--------------------------

4.2 SOURCE CODE

```
from custom tkinter import *
import mysql.connector
db = mysql.connector.connect(
    host='localhost',
    user='root',
    password='mango',
    database="count_the_carbon"
)

mc = db.cursor()

mc.execute("CREATE TABLE IF NOT EXISTS USERS(USER_ID INT PRIMARY KEY,USER_NAME
VARCHAR(25),PASSWORD VARCHAR(25))")
db.commit()

main = CTk()
main.title("Login")
main.geometry("700x450")

global login_user_name
login_user_name = StringVar(value="username")
global login_password
login_password = StringVar(value="password")
global register_user_id
register_user_id = StringVar(value="Enter UserId")
global register_user_name
register_user_name = StringVar(value="Enter desired Username")
global register_user_password
register_user_password = StringVar(value="Enter desired password")


def set_tab_default():
    login_user_name.set("")
    login_password.set("")
    register_user_id.set("User ID")
    register_user_name.set("Username")
    register_user_password.set("Password")


tabview = CTkTabview(main, width=550, height=400, command=set_tab_default)
tabview.place(relx=.10, rely=.050)
```

```
login = tabview.add("LOGIN")
register = tabview.add("REGISTER")
tabview.set("LOGIN")

label_login = CTkLabel(
    login, text="", underline=True)
label_login.place(relx=0.40, rely=0.7)

label_register = CTkLabel(
    register, text="", underline=True)
label_register.place(relx=0.40, rely=0.7)

lu = CTkEntry(
    login, corner_radius=20, width=200, textvariable=login_user_name)
lu.bind("<FocusIn>", lambda e: login_user_name.set(""))
lu.place(relx=.30, rely=.05)

lp = CTkEntry(
    login, corner_radius=20, width=200, textvariable=login_password)
lp.place(relx=.30, rely=.15)
lp.bind("<FocusIn>", lambda e: login_password.set(""))


rui = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_id)
rui.place(relx=.30, rely=.05)
rui.bind("<FocusIn>", lambda e: register_user_id.set(""))

ru = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_name)
ru.place(relx=.30, rely=.15)
ru.bind("<FocusIn>", lambda e: register_user_name.set(""))

rp = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_password)
rp.place(relx=.30, rely=.25)
rp.bind("<FocusIn>", lambda e: register_user_password.set(""))


def register_func(i, name, password):
    q = "INSERT INTO USERS(USER_ID,USER_NAME,PASSWORD) VALUES(%s,%s,%s)"
    val = (i, name, password)
    try:
        mc.execute(q, val)
    except mysql.connector.errors.IntegrityError:
            return "user id already taken"
        except:
            return "Something went wrong"
        else:
            db.commit()
            return "User registered successfully"


def login_func(name, password):
    p = ''
    mc.execute(f"SELECT PASSWORD FROM USERS WHERE USER_NAME='{name}'")
    for i in mc:
        p = i[0]
    if p == password:
        return True
    else:
        return False
def register_clicked():
    val = register_func(register_user_id.get(), register_user_name.get(
    ).upper(), register_user_password.get())
    label_register.configure(text=val)



def explore(un):

    Carbon_calculate = CTk()
    Carbon_calculate.geometry("700x600")
    Carbon_calculate.title("Carbon_emisson")
    CTkLabel(
   Carbon_calculate, text=f"Hello {un} Welcome to Carbon emission calculator",
      font=CTkFont(family="Sans-serif", size=25)).place(relx=.05, rely=.1)

    label = CTkLabel(master=Carbon_calculate, text="Select a region: ",
                  font=CTkFont(family="Sans-serif", size=15))
    label.place(relx=0.4, rely=0.25)

    optionmenu_var = StringVar(value="Regions")

    def optionmenu_callback(choice):
        Input_sources = CTk()
        Input_sources.title("Login")
        Input_sources.geometry("700x450")
        mc.execute(f"SELECT SOURCES FROM {choice}")
        global sources
        sources = []
        for i in mc:
            sources.append(i[0])
        global count
        count = 0
        global values
        values = {}
        global val
        val = StringVar(
            value=f"Enter amount of {sources[count]} consumed",
master=Input_sources)
        a = CTkEntry(master=Input_sources,
                     textvariable=val,
                     width=350,
                     height=25,
                     border_width=2,
                     corner_radius=10)
        a.bind("<FocusIn>", lambda e: val.set(""))
        a.place(relx=0.40, rely=0.22+(0.05*count))

        def next_button():
            global val
            global count
            values[sources[count]] = val.get()
            count += 1
            if count >= len(sources):
                label3 = CTkLabel(master=Input_sources, text="You have entered
all the sources, Please click on CALCULATE ")
                label3.place(relx=0.1, rely=0.6)
                pass
            else:
                label4 = CTkLabel(master=Input_sources, text="Keep going! you
have a few more sources to enter. ")
                label4.place(relx=0.1, rely=0.6)
                values[sources[count]] = val.get()
                val = StringVar(
                    value=f"Enter amount of {sources[count]} consumed",
master=Input_sources)
                a = CTkEntry(master=Input_sources,
                             textvariable=val,
                             width=350,
                             height=25,
                             border_width=2,
                             corner_radius=10)
                a.bind("<FocusIn>", lambda e: val.set(""))
                a.place(relx=0.40, rely=0.22+(0.07*count))
        next = CTkButton(master=Input_sources,
                         width=120,
                        height=32,
                        border_width=0,
                        corner_radius=8,
                        text="NEXT",
                        command=lambda: next_button())
        next.place(relx=0.1, rely=0.5)
        global result_label
        result_label = CTkLabel(master=Carbon_calculate,
                            text="",
                            width=120,
                            height=25,
                            corner_radius=8,
                            text_color="#33cc33",
                            font=CTkFont(family="Sans-serif", size=20))


        result_label.place(relx=.10, rely=.60)
        def calculate_button():
            Input_sources.destroy()
            mc.execute(f"SELECT sources,emission_factor from {choice}")
            original_data = {}
            for i in mc:
                original_data[i[0]] = float(i[1])
            total_carbon_emission = 0
            print(original_data)
            print(values)
            for i in original_data.keys():
                v = original_data[i] * float(values[i])
                total_carbon_emission += v
            if total_carbon_emission <= 50:
                result_label.configure(
                    text=f"The carbon emission of your {choice} is
{total_carbon_emission:.4f} kg of CO2")

            else:
                result_label.configure(text_color="#ff0000",
                    text=f"The carbon emission of your {choice} is
{total_carbon_emission:.4f} kg of CO2")
            try:
                mc.execute('create table result(type_e
varchar(20),emission_amount varchar(20))')
            except:
                pass
            sql='insert into result values(%s,%s)'
            val=(choice,total_carbon_emission)
            mc.execute(sql,val)
        calculate = CTkButton(master=Input_sources,
                              width=120,
                              height=32,
                              border_width=0,
                              corner_radius=8,
                              text="SUBMIT",
                              command=calculate_button)
        calculate.place(relx=0.1, rely=0.7)
        Input_sources.mainloop()
    combobox = CTkComboBox(master=Carbon_calculate,
                           values=["Household", "Rural", "Urban",
                                   "Village", "Metropolitan"],
                           command=optionmenu_callback,
                           variable=optionmenu_var)
    combobox.place(relx=0.38, rely=0.30)
    Carbon_calculate.mainloop()
    print(values)

def login_clicked():
    val = login_func(str(login_user_name.get()).upper(),
                     login_password.get())
    i = login_user_name.get().upper()
    if val:
        label_login.configure(text="Logged in successfully")
        main.destroy()
        explore(i)
    else:
        label_login.configure(text="Data not found")



CTkButton(
    register, text="Register", corner_radius=20, width=200,
command=register_clicked).place(relx=.30, rely=.45)
CTkButton(
    login, text="Login", corner_radius=20, width=200,
command=login_clicked).place(relx=.30, rely=.45)
main.mainloop()
```

```
from custom tkinter import *
import mysql.connector
db = mysql.connector.connect(
    host='localhost',
    user='root',
    password='mango',
    database="count_the_carbon"
)

mc = db.cursor()

mc.execute("CREATE TABLE IF NOT EXISTS USERS(USER_ID INT PRIMARY KEY,USER_NAME
VARCHAR(25),PASSWORD VARCHAR(25))")
db.commit()
main = CTk()
main.title("Login")
main.geometry("700x450")

global login_user_name
login_user_name = StringVar(value="username")
global login_password
login_password = StringVar(value="password")
global register_user_id
register_user_id = StringVar(value="Enter UserId")
global register_user_name
register_user_name = StringVar(value="Enter desired Username")
global register_user_password
register_user_password = StringVar(value="Enter desired password")



def set_tab_default():
    login_user_name.set("")
    login_password.set("")
    register_user_id.set("User ID")
    register_user_name.set("Username")
    register_user_password.set("Password")



tabview = CTkTabview(main, width=550, height=400, command=set_tab_default)
tabview.place(relx=.10, rely=.050)

login = tabview.add("LOGIN")
register = tabview.add("REGISTER")
tabview.set("LOGIN")

label_login = CTkLabel(
    login, text="", underline=True)
label_login.place(relx=0.40, rely=0.7)

label_register = CTkLabel(
    register, text="", underline=True)
label_register.place(relx=0.40, rely=0.7)
```

```
lu = CTkEntry(
    login, corner_radius=20, width=200, textvariable=login_user_name)
lu.bind("<FocusIn>", lambda e: login_user_name.set(""))
lu.place(relx=.30, rely=.05)
lp = CTkEntry(
    login, corner_radius=20, width=200, textvariable=login_password)
lp.place(relx=.30, rely=.15)
lp.bind("<FocusIn>", lambda e: login_password.set(""))



rui = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_id)
rui.place(relx=.30, rely=.05)
rui.bind("<FocusIn>", lambda e: register_user_id.set(""))

ru = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_name)
ru.place(relx=.30, rely=.15)
ru.bind("<FocusIn>", lambda e: register_user_name.set(""))

rp = CTkEntry(
    register, corner_radius=20, width=200, textvariable=register_user_password)
rp.place(relx=.30, rely=.25)
rp.bind("<FocusIn>", lambda e: register_user_password.set(""))



def register_func(i, name, password):
    q = "INSERT INTO USERS(USER_ID,USER_NAME,PASSWORD) VALUES(%s,%s,%s)"
    val = (i, name, password)
    try:
        mc.execute(q, val)
    except mysql.connector.errors.IntegrityError:
        return "user id already taken"
    except:
        return "Something went wrong"
    else:
        db.commit()
        return "User registered successfully"



def login_func(name, password):
    p = ''
    mc.execute(f"SELECT PASSWORD FROM USERS WHERE USER_NAME='{name}'")
    for i in mc:
        p = i[0]
    if p == password:
        return True
    else:
        return False



def register_clicked():
    val = register_func(register_user_id.get(), register_user_name.get(
    ).upper(), register_user_password.get())
    label_register.configure(text=val)



def explore(un):

    Carbon_calculate = CTk()
    Carbon_calculate.geometry("700x600")
    Carbon_calculate.title("Carbon_emisson")
    CTkLabel(
   Carbon_calculate, text=f"Hello {un} Welcome to Carbon emission calculator",
       font=CTkFont(family="Sans-serif", size=25)).place(relx=.05, rely=.1)

    label = CTkLabel(master=Carbon_calculate, text="Select a region: ",
                font=CTkFont(family="Sans-serif", size=15))
    label.place(relx=0.4, rely=0.25)

    optionmenu_var = StringVar(value="Regions")

    def optionmenu_callback(choice):
        Input_sources = CTk()
        Input_sources.title("Login")
        Input_sources.geometry("700x450")
        mc.execute(f"SELECT SOURCES FROM {choice}")
        global sources
        sources = []
        for i in mc:
            sources.append(i[0])
        global count
        count = 0
        global values
        values = {}
        global val
        val = StringVar(
            value=f"Enter amount of {sources[count]} consumed",
master=Input_sources)
        a = CTkEntry(master=Input_sources,
                     textvariable=val,
                     width=350,
                     height=25,
                     border_width=2,
                     corner_radius=10)
        a.bind("<FocusIn>", lambda e: val.set(""))
        a.place(relx=0.40, rely=0.22+(0.05*count))

        def next_button():
            global val
            global count
            values[sources[count]] = val.get()
            count += 1
            if count >= len(sources):
                label3 = CTkLabel(master=Input_sources, text="You have entered
all the sources, Please click on CALCULATE ")
                label3.place(relx=0.1, rely=0.6)
                pass
            else:
                label4 = CTkLabel(master=Input_sources, text="Keep going! you
have a few more sources to enter. ")
                label4.place(relx=0.1, rely=0.6)
                values[sources[count]] = val.get()
                val = StringVar(
                    value=f"Enter amount of {sources[count]} consumed",
master=Input_sources)
                a = CTkEntry(master=Input_sources,
                             textvariable=val,
                             width=350,
                             height=25,
                             border_width=2,
                             corner_radius=10)
                a.bind("<FocusIn>", lambda e: val.set(""))
                a.place(relx=0.40, rely=0.22+(0.07*count))
        next = CTkButton(master=Input_sources,
                         width=120,
                         height=32,
                         border_width=0,
                         corner_radius=8,
                         text="NEXT",
                         command=lambda: next_button())
        next.place(relx=0.1, rely=0.5)
        global result_label
        result_label = CTkLabel(master=Carbon_calculate,
                              text="",
                              width=120,
                              height=25,
                              corner_radius=8,
                              text_color="#33cc33",
                              font=CTkFont(family="Sans-serif", size=20))


        result_label.place(relx=.10, rely=.60)
        def calculate_button():
            Input_sources.destroy()
            mc.execute(f"SELECT sources,emission_factor from {choice}")
            original_data = {}
            for i in mc:
                original_data[i[0]] = float(i[1])
            total_carbon_emission = 0
            print(original_data)
            print(values)
            for i in original_data.keys():
                v = original_data[i] * float(values[i])
                total_carbon_emission += v
            if total_carbon_emission <= 50:
                result_label.configure(
                    text=f"The carbon emission of your {choice} is
{total_carbon_emission:.4f} kg of CO2")

            else:
                result_label.configure(text_color="#ff0000",
                    text=f"The carbon emission of your {choice} is
{total_carbon_emission:.4f} kg of CO2")
            try:
                mc.execute('create table result(type_e
varchar(20),emission_amount varchar(20))')
            except:
                pass
            sql='insert into result values(%s,%s)'
            val=(choice,total_carbon_emission)
            mc.execute(sql,val)
        calculate = CTkButton(master=Input_sources,
                              width=120,
                              height=32,
                              border_width=0,
                              corner_radius=8,
                              text="SUBMIT",
                              command=calculate_button)
        calculate.place(relx=0.1, rely=0.7)
        Input_sources.mainloop()
    combobox = CTkComboBox(master=Carbon_calculate,
                           values=["Household", "Rural", "Urban",
                                   "Village", "Metropolitan"],
                           command=optionmenu_callback,
                           variable=optionmenu_var)
    combobox.place(relx=0.38, rely=0.30)
    Carbon_calculate.mainloop()
    print(values)

def login_clicked():
    val = login_func(str(login_user_name.get()).upper(),
                    login_password.get())
    i = login_user_name.get().upper()
    if val:
        label_login.configure(text="Logged in
        succesfully") main.destroy()
        explore(i)
    else:
        label_login.configure(text="Data not found")



CTkButton(
    register, text="Register", corner_radius=20, width=200,
command=register_clicked).place(relx=.30, rely=.45)
CTkButton(
    login, text="Login", corner_radius=20, width=200,
command=login_clicked).place(relx=.30, rely=.45)
main.mainloop()
```

CONCLUSION

In conclusion, a carbon emission calculator project can play a crucial role in understanding and reducing carbon emissions. The calculator can provide valuable information on the carbon footprint of different activities, products, and events, and can help individuals, organizations, and companies to identify areas where they can reduce their emissions. By providing a clear and detailed picture of the emissions associated with different activities, the calculator can also help to support decision-making and the development of policies and regulations aimed at reducing emissions. Additionally, by providing recommendations for reducing emissions, the calculator can also support the transition towards a more sustainable and low-carbon future. Overall, a carbon emission calculator project can be a powerful tool for addressing climate change and promoting sustainable development.

BIBLIOGRAPHY

WEBSITE REFERENCES:

Tkinrer/CustomTkinter:

- https://docs.python.org/3/library/tkinter.html

MySQL/MySQL Connector:

- https://www.mysql.com

Contact Details :

Name : A S NAVYASHREE (1DT20AI001)
E-mail : 1dt20ai001@dsatm.edu.in

Name : DEEPTI HEGDE (1DT20AI018)

EMAIL : 1dt20ai018@dsatm.edu.in