اونيۏرسيتي مليسيا ڤهڠ

UNIVERSITI MALAYSIA PAHANG

PUSAT SAINS MATEMATIK

CENTRE FOR MATHEMATICAL SCIENCES

BSD2213 DATA SCIENCE PROGRAMMING I

SEMESTER I SESSION 2022/22

GROUP PROJECT

GRAPHICAL USER INTERFACE (GUI) USING PYTHON

Title Project: Electric Management System

GROUP MEMBERS:

NAME	STUDENT ID
NUR ADLINA ADILAH BINTI AZIMAN	SD21003
TEAN JIN HE	SD21063
NOR FARAWAHIDA BT ABDULLAH	SD21010
KAVVINA A/P SELVARAJU	SD21005

LECTURER : DR NORAZIAH BT ADZHAR

DATE OF SUBMISSION: 26. FEBRUARY. 2023

Table of Content

1.0 Introduction

Science and technology with all its fascinating advancements has been taking human life standards to the next level. The whole world will be literally jammed without these innovations. This project will show the user the amount of payment that the user needs to pay for that month.

Knowing your use of electricity can be crucial for you in determining your bill budget for each month. Our group came up with this project for the user to know by themselves their electric bill. This project allows users to keep up to date with their electricity billing every month by looking at their billing history. By knowing this, users can know if they had used a lot of electricity for that month or not. They can also know if they have reached their limit of budget or not. By that, they can manage the usage of electricity and save more power.

From this project, the user can also compare if their real electricity bill is correct or not. Nowadays, many people don't know how to calculate their own electricity bill which when needed, they just pay without knowing if it is true or not. So with this project, the user can know if they got cheated or the user usage is not comparable with the amount that they need to pay. If not, they can report it to the person in charge. Users can insert the amount of usage for that month and the system will automatically calculate the payment.

2.0 Why This Project?

The reason why we chose this project is that our group wants to highlight the meaning of the slogan save energy, save the environment. Consuming less energy helps the environment by conserving natural resources, reducing hazardous emissions from power plants, and safeguarding ecosystems from extinction. We can improve the health and happiness of everyone in the world by taking steps to limit our energy intake. So, with an app that helps us to calculate our usage of electricity, it can surely give a big involvement for saving our world.

Unlike other electrical management systems out there, our project has its own unique feature which is that the user can see their history in detail. For this feature, users can easily use this feature to take a look and manage their bill for every month even for the past few years. For example, the user can see the trend of the usage for the electricity based on the payment that they need to pay. If the payment is the highest for that month, maybe the user can detect which one of the devices is likely using the huge amount of electricity. So, they can reduce or limit the time of using the particular device.

The user still can see the history of their consumption and payment even after they have ended the system. If the user inserts new user details, it will automatically be saved in the system. So that every user that has inserted in their user details can see their name and their done or needs to be done payment. With a history system like this, the user can manage their budget on electricity besides saving our environment for a better future.

3.0 How Can This Project Be Extended?

An electricity bill management system built using Python can be extended in a number of ways to add more functionality and improve the user experience. We think there is still potential for improvement to make our product more beneficial based on the software we have developed. Implementing a real-time consumption tracking feature that enables customers to view their current usage and predicted cost in real-time will help us build better software. Real-time usage tracking allows customers to see exactly how much electricity they are using at any given time, which can help them to better understand their usage patterns and identify ways to reduce consumption and save money on their bills. Real-time consumption monitoring enables customers to make better educated choices about when to use energy-hungry appliances and when to practise energy conservation, which can lower their bills.

A feature that enables users to view and download their usage history and billing statements can also be added. Customers don't need to go to the utility company's office or wait for paper bills to be sent in order to examine and download their usage history and billing statements from the convenience of their home or while traveling. Customers can easily keep track of their usage history and billing statements, which can help them to identify patterns and trends, compare bills over time and make sure that they are paying the correct amount. By having easy access to their usage history and billing statements, customers can more effectively budget for their energy expenses.

Moreover, using data analytics and machine learning to forecast consumption trends and spot potential problems. Machine learning and data analytics can help to predict future usage patterns and identify potential issues, which can help customers to better plan for their energy expenses and avoid unexpected bills. In order to help customers avoid expensive repairs or replacements, machine learning can be used to discover trends in energy usage that point to prospective equipment failures or maintenance needs.

Furthermore,adding an automatic bill payment feature that allows customers to set up recurring payments or automatic payments from their bank account. Automatic bill payment

allows customers to set up recurring payments or automatic payments from their bank account, so they don't have to worry about remembering to pay their bills on time, or worry about late payment fees. Customers can budget more successfully by using automatic bill payment, which guarantees that their bills are paid on time and relieves them of the anxiety of unforeseen fees or late payments. By examining their bills and payments, consumers who use automatic bill payment can quickly and simply settle any billing problems or questions. By ensuring that payments are made to the right account and that payments are processed correctly, automatic bill payment features can help to lower the risk of fraud and mistakes.

Finally,implementing a feature that allows customers to view their energy consumption in terms of CO2 emissions and cost savings. Customers who can assess their energy use in terms of CO2 emissions are better equipped to comprehend their environmental impact and find strategies to lower their carbon footprint.

4.0 Source Code

```
import tkinter as tk
from csv import DictWriter
import os
import time
import pandas as pd
from tkinter import messagebox
from tkinter import *
import csv
from PIL import Image,ImageTk
from tkinter import ttk
import tkinter

def Bill_window():
    window = Toplevel()
    window.title("ELECTRIC BILL MANAGEMENT SYSTEM")
    window.geometry("1080x720")
    window.configure(background='#7ea7fe')

    Title = Label(window, text="ELECTRIC BILL", fg='blue', bd=10, bg='#7ea7fe',font=("Times New Roman", 20, "bold")).grid(row=0)

    #==============================Time======================================#
    localtime = time.asctime(time.localtime(time.time()))
    timeInfo = Label(window, font=('arial', 20, 'bold'), text=localtime, fg="blue", bg='#7ea7fe', bd=10, anchor='w')
    timeInfo.grid(row=1, column=0)
```

```
Label01 = Label(window, text="----------Details----------", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=2)
Label02 = Label(window, text="----------", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=3)
Label03 = Label(window, text="        Consumption block                    Rate(RM/kWh)", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=4)
Label04 = Label(window, text="----------", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=5)
Label05 = Label(window, text="      First 200 kWh (1-200 kWh)              0.218", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=6)
Label06 = Label(window, text="      Next 100 kWh (201-300 kWh)             0.334", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=7)
Label07 = Label(window, text="      Next 300 kWh (301-600 kWh)             0.516", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=8)
Label08 = Label(window, text="      Next 300 kWh (601-900 kWh)             0.546", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=9)
Label09 = Label(window, text="      Next 901 kWh onwards                   0.571", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=10)
Label10 = Label(window, text="----------", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=11)
Label11 = Label(window, text="More than 600 kWh need include service tax 6 percent", fg='blue', bg='#7ea7fe',font=("Times New Roman", 12)).grid(row=12)
Label12 = Label(window,
                text="ICPT help to assist the low income and vulnerable customers under this consumption band which is RM0.02per(kWh)",
                fg='blue', bg='#7ea7fe', font=("Times New Roman", 12)).grid(row=13)

rate = StringVar()
ICPT = StringVar()
electricprice = StringVar()
totalprice = StringVar()
```

```
    billdate = StringVar()
    servicetax = StringVar()
    electric = StringVar()
    month_value = StringVar()

    frame = Frame(window)
    frame.grid(row=15, column=1)

    Label(window, text="ENTER YOUR NAME:", fg='blue', bg='#7ea7fe', width=50, height=2,
          font=('Times New Romen', 12)).grid(row=14)
    username = Entry(window)
    username.grid(row=14, column=1, sticky=W)

        Label(window, text="ENTER YOUR BILL DATE", fg='blue', bg='#7ea7fe', width=30,
height=2,
          font=('Times New Romen', 12)).grid(row=15)
    month_list = ["JANUARY", "FEBRUARY", "MARCH", "APRIL",
                  "MAY", "JUNE", "JULY", "AUGUST", "SEPTEMBER",
                  "OCTOBER", "NOVEMBER", "DECEMBER"]

    month_value.set(month_list[0])
    month_menu = OptionMenu(frame, month_value, *month_list)
    month_menu.grid(row=15, column=1, sticky=W)
    billyear = Entry(window)
    billyear.grid(row=15, column=2, sticky=W)

         Label(window, text="ENTER YOUR TOTAL CONSUMPTION (kWh):", fg='blue',
bg='#7ea7fe', width=50, height=2,
          font=('Times New Romen', 12)).grid(row=16)
    electricbx = Entry(window, textvariable=electric)
    electricbx.grid(row=16, column=1, sticky=W)
```

```
def calculation():
    if (electric.get() == ""):
        electricprice = 0
    else:
        electricprice = float(electric.get())
        if (electricprice <= 200):
            electric_rate = electricprice * 0.218
            electric_ICPT = electricprice * 0.02
            electric_price = "RM", str('%.2f' % (electric_rate - electric_ICPT))
            totalprice.set(electric_price)
        elif (electricprice <= 300):
            electric_rate = (200 * 0.218) + ((electricprice - 200) * 0.334)
            electric_ICPT = electricprice * 0.02
            electric_price = "RM", str('%.2f' % (electric_rate - electric_ICPT))
            totalprice.set(electric_price)
        elif (electricprice <= 600):
            electric_rate = (200 * 0.218) + (100 * 0.334) + ((electricprice - 300) * 0.516)
            electric_ICPT = electricprice * 0.02
            electric_price = "RM", str('%.2f' % (electric_rate - electric_ICPT))
            totalprice.set(electric_price)
        elif (electricprice <= 900):
            electric_rate = (200 * 0.218) + (100 * 0.334) + (300 * 0.516) + ((electricprice - 600) *
0.546)
            electric_ICPT = electricprice * 0.02
            electric_servicetax = (((electricprice - 600) * 0.546) - ((electricprice - 600) * 0.02)) *
0.06
            electric_price = "RM", str('%.2f' % (electric_rate - electric_ICPT +
electric_servicetax))
            totalprice.set(electric_price)
        else:
```

```
        electric_rate = (200 * 0.218) + (100 * 0.334) + (300 * 0.516) + (300 * 0.546) + (
            (electricprice - 900) * 0.571)
        electric_ICPT = electricprice * 0.02
        electric_servicetax = ((300 * 0.546) + ((electricprice - 900) * 0.571) - (
            (electricprice - 600) * 0.02)) * 0.06
                    electric_price = "RM", str('%.2f' % (electric_rate - electric_ICPT +
electric_servicetax))
        totalprice.set(electric_price)


    Label(window, text="TOTAL PAYMENT :", fg='blue', bg='#7ea7fe', width=20, height=2,
        font=('Times New Romen', 12)).grid(row=17)
    totalpricebx = Entry(window, textvariable=totalprice)
    totalpricebx.grid(row=17, column=1)

        calcbtn = tk.Button(window, bd=16, fg="black", font=('arial', 16, 'bold'), width=10,
text="Total", bg="#F44336",command=calculation)
    calcbtn.grid(row=18, column=1)

    def submit():
        name = username.get()
        month = month_value.get()
        year = billyear.get()
        elec = electricbx.get()
        totalpayment = totalpricebx.get()
        with open("bill_records.csv", 'a', newline='') as f:
            dict_writer = DictWriter(f, fieldnames=['Name', 'Month', 'Year', 'Total Consumption',
'Total Payment'])

        if os.stat("bill_records.csv").st_size == 0:  # checks if file contains the header or not
            DictWriter.writeheader(dict_writer)
```

```
        dict_writer.writerow(
                {'Name': name, 'Month': month, 'Year': year, 'Total Consumption': elec, 'Total Payment': totalpayment})

        messagebox.showinfo('Message', 'Record added Sucessfully')  # creating message box

        Name = username.delete(0, tk.END)
        Month = month_value(0, tk.END)
        Year = billyear.delete(0, tk.END)
        Electric = electricbx.delete(0, tk.END)
        Total = totalpricebx.delete(0, tk.END)

    submit_btn = tk.Button(window, bd=16, fg="black", font=('arial', 16, 'bold'), width=10, bg="#F44336", text='Submit',command=submit)
    submit_btn.grid(row=18, column=2)

def User_window():
    Userwindow= Toplevel()
    Userwindow.title('USER DETAILS')
    Userwindow.geometry("550x550")
    Userwindow.configure(background="#7ea7fe")
            label=Label(Userwindow,text="ENTER USER DETAILS\n", fg="yellow", bg="#7ea7fe",font=("Times New Roman",17,"bold"))
    label.pack()

            image = Image.open('C:\\Users\\Dell\\Desktop\\STUDENT\\DATA SCIENCE PROGRAMMING I\\GROUP ASSIGMENT\\user.png')
    resize_image = image.resize((200, 150))
    img = ImageTk.PhotoImage(resize_image)
    Label3 = Label(Userwindow, image=img)
```

```
Label3.image = img
Label3.place(x=170, y=30)

name = Label(Userwindow, text="Name:", fg="white", bg="black",font=("Times New Roman",15,))
name.place(x=90, y=200)
email = Label(Userwindow, text="Email:", fg="white", bg="black",font=("Times New Roman",15,))
email.place(x=90, y=240)
icnumber = Label(Userwindow, text="IC Number:",fg="white", bg="black",font=("Times New Roman",15,))
icnumber.place(x=90, y=280)
address = Label(Userwindow, text="Address:",fg="white", bg="black",font=("Times New Roman",15,))
address.place(x=90, y=320)

name1 = Entry(Userwindow)
name1.place(x=200, y=200,width=240,height=20)
email1 = Entry(Userwindow)
email1.place(x=200, y=240,width=240,height=20)
icnumber1 = Entry(Userwindow)
icnumber1.place(x=200, y=280,width=240,height=20)
address1 = Entry(Userwindow)
address1.place(x=200, y=320,width=240,height=50)

def clear_text():
    text.delete(1.0, END)

def submit1():
    namex = name1.get()
    emailx = email1.get()
```

```
    icx = icnumber1.get()
    addressx = address1.get()
    with open("user_records.csv", 'a', newline='') as f:
        dict_writer = DictWriter(f, fieldnames=['Name', 'Email', 'IC Number','Address'])

        if os.stat("user_records.csv").st_size == 0:  # checks if file contains the header or not
            DictWriter.writeheader(dict_writer)

                dict_writer.writerow({'Name': namex, 'Email': emailx, 'IC Number': icx,'Address':
addressx})

        messagebox.showinfo('Message', 'Record added Sucessfully')  # creating message box

        name = name1.delete(0, tk.END)
        email = email1.delete(0,tk.END)
        ic = icnumber1.delete(0, tk.END)
        address = address1.delete(0, tk.END)

                sbmitbtn = Button(Userwindow, text="Submit", activebackground="pink",
activeforeground="red",bd=16, fg="black", font=('arial', 16, 'bold'), width=10,bg="#F44336",
command=submit1).place(x=230, y=390)

    text=Text(height=5)
                backtomain_window=Button(Userwindow, text="BACK TO MAIN MENU",
fg='black',bg='green',width=500,height=1,font=("Times New Roman",12),
command=Userwindow.destroy).pack(side=BOTTOM)

btnReset=Button(Userwindow,text="CLEAR",fg='black',bg='green',width=500,height=1,font=("
Times New Roman",12), command=clear_text).pack(side=BOTTOM)

def Report_window():
```

```
    Reportwindow=Toplevel()
    Reportwindow.geometry("500x400")
    Reportwindow.configure(background='#7ea7fe')
    Reportwindow.title("ELECTRICAL CALCULATOR TO COMFIRM YOUR BILLS")
        greeting=Label(Reportwindow,text="BILL  REPORT",fg='blue',bg='#7ea7fe',font=("Times
New Roman",20,"bold")).pack()

            image = Image.open('C:\\Users\\Dell\\Desktop\\STUDENT\\DATA SCIENCE
PROGRAMMING I\\GROUP ASSIGMENT\\Report.jpeg')
    resize_image = image.resize((250, 250))
    img = ImageTk.PhotoImage(resize_image)
    Label2 = Label(Reportwindow, image=img)
    Label2.image = img
    Label2.place(x=120, y=40)

    f1 = pd.read_csv("user_records.csv", index_col=0)
    f2 = pd.read_csv("bill_records.csv", index_col=0)
    combine_record = pd.merge(f1,f2, left_on="Name", right_on="Name", how="left")
    combine_record.to_csv("combine_record.csv")

    def getUser_FullStatement():
        window1 = Tk()
        window1.geometry("800x400")
        window1.configure(background='#7ea7fe')
        window1.title("ELECTRICAL CALCULATOR TO COMFIRM YOUR BILLS")
         Label_01 = Label(window1, text="FULL REPORT", fg='black', bg='#7ea7fe',font=("Times
New Roman", 12, "bold")).pack()

        csv.path = r".\combine_record.csv"
        def sumbit2():
            search_name = namebx.get()
```

```
        search_month = monthbx.get()
        search_year = yearbx.get()

        icbx.configure(state=tk.NORMAL)
        emailbx.configure(state=tk.NORMAL)
        addressbx.configure(state=tk.NORMAL)
        consumptionbx.configure(state=tk.NORMAL)
        totalbx.configure(state=tk.NORMAL)

        icbx.delete(0, 'end')
        emailbx.delete(0, 'end')
        addressbx.delete(0, 'end')
        consumptionbx.delete(0, 'end')
        totalbx.delete(0, 'end')

        df=pd.read_csv("combine_record.csv")
        row_count = df.shape[0]

        with open("combine_record.csv") as file:
            csv_read = csv.reader(file)
            df= pd.DataFrame([csv_read],index = None)
            n=1
            while n<=row_count:
                for val in list(df[n]):
                        if str(val[0]) == str(search_name) and str(val[4]) == str(search_month) and
str(val[5]) == str(search_year):
                    icbx.insert(0, val[2])
                    emailbx.insert(0, val[1])
                    addressbx.insert(0,val[3])
                    consumptionbx.insert(0, val[6])
                    totalbx.insert(0, val[7])
```

```
            icbx.configure(state=tk.DISABLED)
            emailbx.configure(state=tk.DISABLED)
            addressbx.configure(state=tk.DISABLED)
            consumptionbx.configure(state=tk.DISABLED)
            totalbx.configure(state=tk.DISABLED)
        n+=1

    Label_011 = Label(window1, text="Name          :", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=50)
    Label_012 = Label(window1, text="IC Number     :", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=70)
    Label_013 = Label(window1, text="Email         :", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=90)
    Label_014 = Label(window1, text="Month,Year    :", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=110)

    Label_015 = Label(window1, text="=========================================================", fg='black', bg='#7ea7fe', font=("Times New Roman", 12)).place(x=0, y=130)
    Label_016 = Label(window1, text="Summary of Electrical Bills", fg='black', bg='#7ea7fe',font=("Times New Roman", 12, "bold")).place(x=0, y=150)
    Label_017 = Label(window1, text="-------------------------------------------------------", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=170)

    Label_018 = Label(window1, text="Bill", fg='black', bg='#7ea7fe', font=("Times New Roman", 12)).place(x=0, y=190)
    Label_019 = Label(window1, text="-------------------------------------------------------", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=210)
    Label_020 = Label(window1, text="Address       :", fg='black', bg='#7ea7fe', font=("Times New Roman", 12)).place(x=0, y=230)
```

```
Label_021 = Label(window1, text="Consumption(kWh):", fg='black', bg='#7ea7fe', font=("Times New Roman", 12)).place(x=0, y=250)
Label_022 = Label(window1, text="Payment        :", fg='black', bg='#7ea7fe', font=("Times New Roman", 12)).place(x=0, y=270)
Label_023 = Label(window1, text="--------------------------------------------------", fg='black', bg='#7ea7fe',font=("Times New Roman", 12)).place(x=0, y=290)

namebx = Entry(window1)
namebx.place(x=150, y=50,width=260,height=20)
icbx = Entry(window1)
icbx.place(x=150, y=70,width=260,height=20)
emailbx = Entry(window1)
emailbx.place(x=150, y=90,width=260,height=20)
monthbx = Entry(window1)
monthbx.place(x=150, y=110,width=260,height=20)
yearbx = Entry(window1)
yearbx.place(x=250, y=110)
addressbx = Entry(window1,width=80)
addressbx.place(x=150, y=230)
consumptionbx = Entry(window1)
consumptionbx.place(x=150, y=250)
totalbx = Entry(window1)
totalbx.place(x=150, y=270)

Button_00 = Button(window1, text="Search", fg='black', bg='#7ea7fe', width=30, height=1, font=("Times New Roman", 12),command=sumbit2).pack(side=BOTTOM)
Button_01 = Button(window1, text="Back", fg='black', bg='#7ea7fe', width=30, height=1, font=("Times New Roman", 12),command=window1.destroy).pack(side=BOTTOM)
```

```
Statement=Button(Reportwindow, text="Full Statement", fg='black',bg='#7ea7fe',width=30,height=1,font=("Times New Roman",12), command=getUser_FullStatement).place(x=240,y=300)

def bill():
    window2=Tk()
    window2.geometry("1250x1000")
    window2.configure(background='#7ea7fe')
    window2.resizable(width=False, height=False)
    window2.title("ELECTRICAL BILLS")

    with open("combine_record.csv", "r", newline="") as passfile:
        reader = csv.reader(passfile)
        data = list(reader)

    entrieslist = []
    for i, row in enumerate(data, start=4):
        entrieslist.append(row[0])
        for col in range(0, 8):
            tk.Label(window2, text=row[col],fg='black',bg='#7ea7fe',height=1,font=("Times New Roman",12)).grid(row=i, column=col)

billP = Button(Reportwindow, text="All Bills", fg='black',bg='#7ea7fe',width=30,height=1,font=("Times New Roman",12), command=bill).place(x=-15, y=300)
Statement1 = Button(Reportwindow, text="End Calculated", fg='black',bg='#7ea7fe',width=500, height=1,font=("Times New Roman",12), command=Reportwindow.destroy).pack(side=BOTTOM)
```

```
                    backtomain_window=Button(Reportwindow, text="Add New Payment",
fg='black',bg='#7ea7fe',width=500,height=1,font=("Times New Roman",12),
command=Bill_window).pack(side=BOTTOM)

mainwindow= Tk()
mainwindow.title("Electric Bill Management System")
mainwindow.geometry("750x600")
mainwindow.configure(background="#7ea7fe")
greeting=Label(mainwindow,text="Welcome to the Electric Bill Management System!\n"
                    "\nMain Menu\n"
                    "\nPlease click one of the button below:", fg="black", bg="#7ea7fe",
                    font=("Times New Roman",15,"bold"))
greeting.pack()

image=Image.open('C:\\Users\\Dell\\Desktop\\STUDENT\\DATA SCIENCE PROGRAMMING
I\\GROUP ASSIGMENT\\main.png')
resize_image=image.resize((280,280))
img=ImageTk.PhotoImage(resize_image)
Label1=ttk.Label(mainwindow,image=img)
Label1.image=img
Label1.place(x=240,y=150)

window4 = Button(mainwindow, text="Cancel", fg='black', bg='#7ea7fe',width=500,
height=1,font=("Times New Roman", 12, "bold"), command=mainwindow.destroy)
window4.pack(side=BOTTOM)
window3 = Button(mainwindow, text="Bill Report", fg='black', bg='#7ea7fe',width=500,
height=1,font=("Times New Roman", 12, "bold"),command=Report_window)
window3.pack(side=BOTTOM)
window2 = Button(mainwindow, text="Bill", fg='black', bg='#7ea7fe',width=500,
height=1,font=("Times New Roman", 12, "bold"), command=Bill_window)
window2.pack(side=BOTTOM)
```

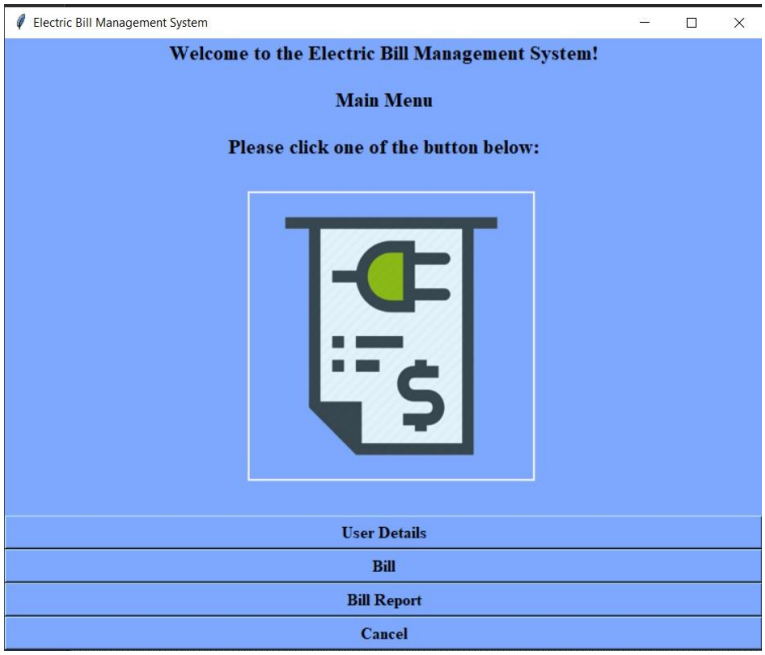
```
window1 = Button(mainwindow, text="User Details", fg='black', bg='#7ea7fe',width=500, height=1,font=("Times New Roman", 12, "bold"), command= User_window)
window1.pack(side=BOTTOM)

mainwindow.mainloop()
```

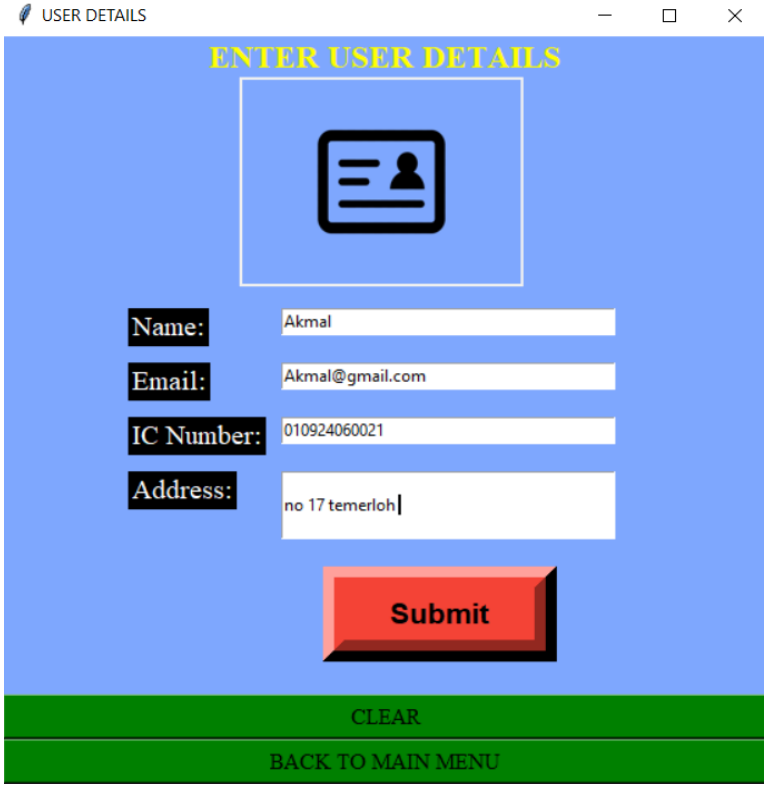
 # you can download the picture in the folder first before run the codes

https://drive.google.com/drive/folders/1AhGcmlg-p1rqKS3WL2pzFe_hHBCL-bkz?usp=share_link

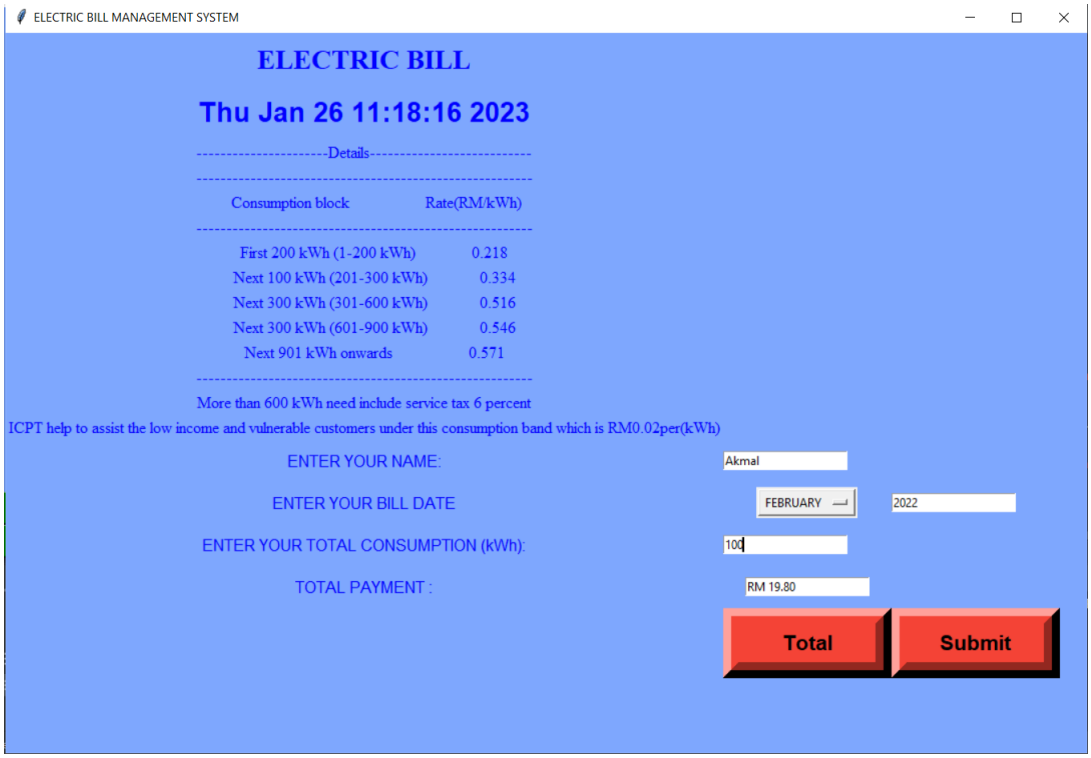
5.0 GUI Screenshot



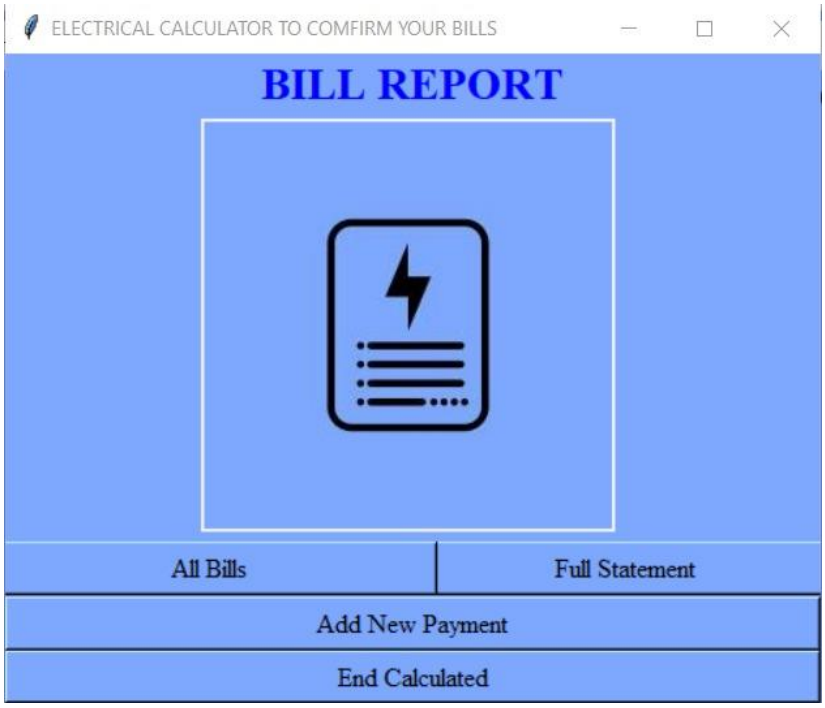
Main Menu



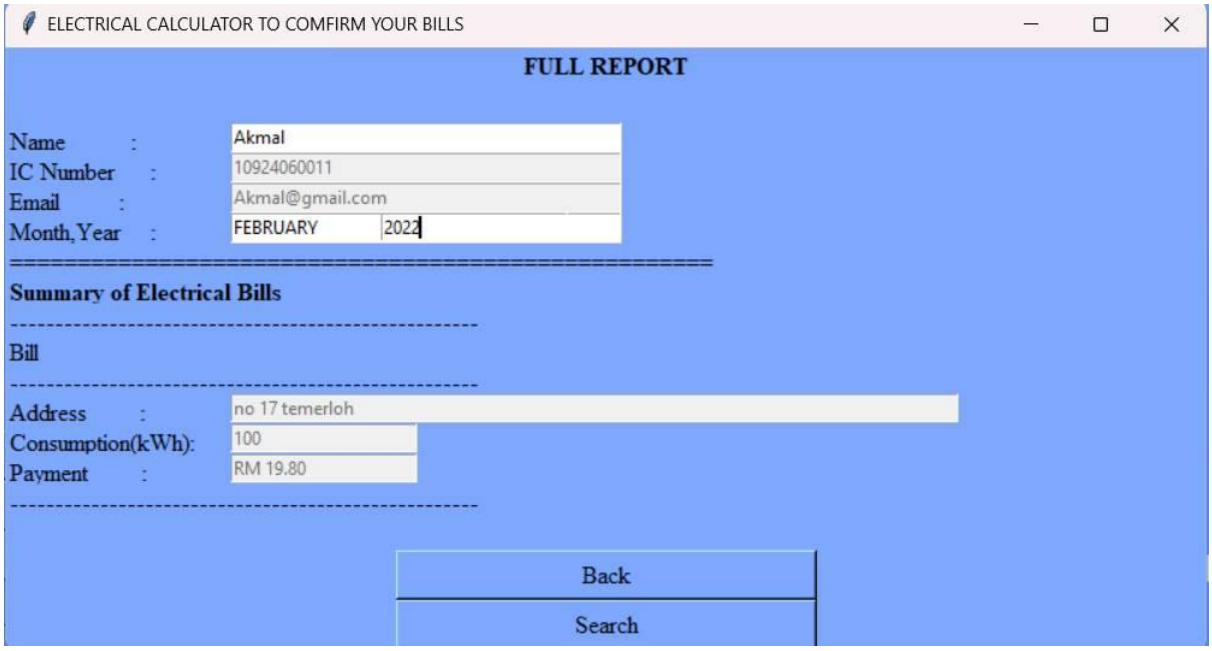
User Details



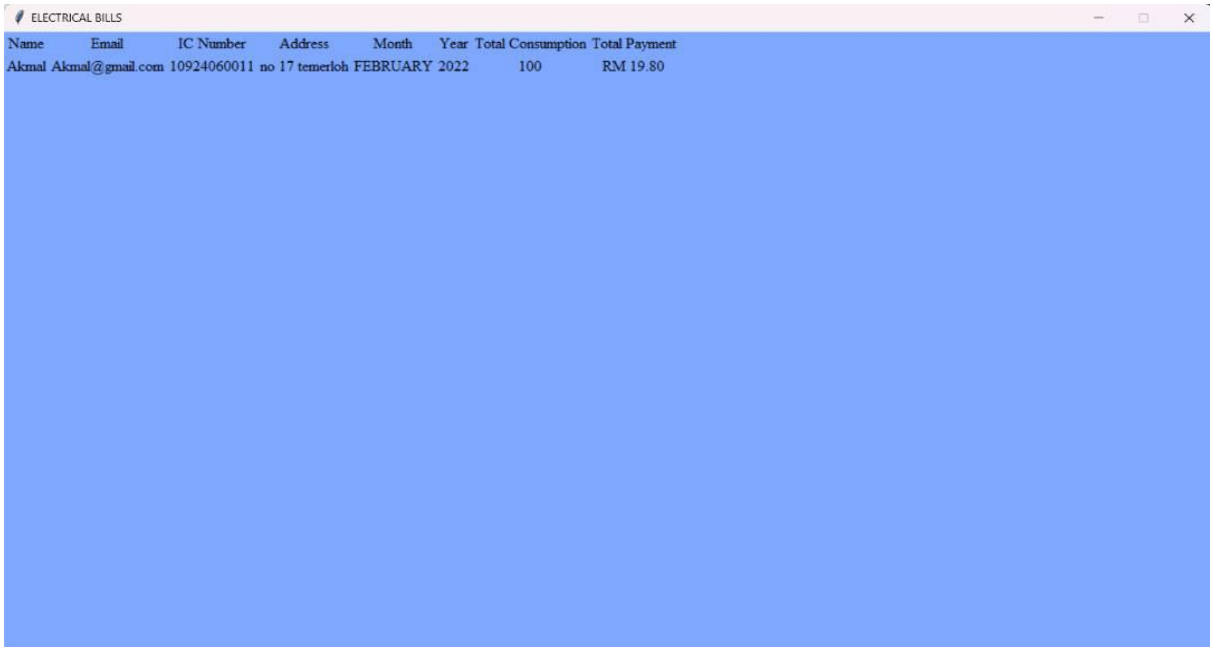
Bill



Bill Report



Full Statement



All Bills