

**NATIONAL INSTITUTE OF TECHNOLOGY**

**PUDUCHERRY KARAIKAL – 609 609**

**(An Institution of National Importance under MoE, Govt. of India)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**Roll Number:** CS21B1020 **Name:** K R VIKRAM

**Semester:** 4 **Class:** CSE

**Subject Code:** CS210 **Subject Name:** Python Programming Lab

**Date :** 01-05-2023

**PYTHON MINI-PROJECT REPORT**

**PetrolPump-Management-Python**

**Aim :**

Create a Software that can manage and record the working of a Petrol Pump using Python.

**Objective :**

The main objective of developing a petrol pump management system using Python is to create a software application that can efficiently manage and automate the various tasks and processes involved in operating a petrol pump.

The following are some specific objectives for a petrol pump management system project developed using Python:

* **Automating Fuel Dispensing:** The system should be able to automate the process of fuel dispensing and provide accurate readings of the amount of fuel dispensed for each transaction.
* **Inventory Management:** The system should be able to track inventory levels, manage orders and deliveries, and generate alerts when inventory levels fall below a certain threshold.
* **Sales Tracking and Reporting:** The system should be able to generate reports on sales data, including transaction volumes, revenue, and customer behaviour, allowing for effective analysis and decision-making.
* **Payment Processing:** The system should be able to process payments efficiently and securely and integrate with popular payment gateways to support multiple payment methods.
* **Customer Management:** The system should be able to manage customer information, track loyalty programs and rewards, and provide personalized services to customers.
* **Employee Management:** The system should be able to manage employee information, work schedules, and track employee performance.
* **Security:** The system should be designed with security in mind, protecting sensitive data and preventing unauthorized access.

**Overview of the Project :**

Petrol Pump Management System is a computer-based software solution that aims to simplify and automate the operations of a petrol pump. This project is developed using Python programming language and uses a graphical user interface (GUI) to facilitate the user's interaction with the system.

The software provides a range of features to manage the daily operations of a petrol pump, including:

* **Pump management :** The system allows the user to manage the pumps at the petrol station, enabling them to monitor the status of each pump and track fuel sales for each pump.
* **Inventory management :** The software helps the user keep track of fuel inventory, including the fuel type, quantity, and price. It provides alerts for low inventory levels, allowing the user to order new stock in advance.
* **Sales management :** The system allows the user to record fuel sales and generate transaction history.

Overall, the Petrol Pump Management System developed using Python is a user-friendly, feature-rich software solution that simplifies and automates the daily operations of a petrol pump. It helps the user to streamline their operations, save time and effort, and make informed decisions based on accurate data.

**Motivation :**

The main motivation of the Petrol Pump Management System project is to provide improved efficiency, accuracy, cost savings, customer service, compliance, and data analysis.

**Scope :**

The scope of a petrol pump management system project involves developing a software application that automates and streamlines various tasks involved in operating a petrol pump, including fuel dispensing, inventory management, sales tracking and reporting, payment processing, customer and employee management, security, user interface, and mobile support.

**Problem Statement :**

The petrol pump management system project aims to address the challenges faced by a Traditional petrol pump like fuel dispensing, inventory management, payment processing, customer management, and employee management, which can be time-consuming, error-prone, and inefficient by developing a software application that automates and streamlines various tasks involved in operating a petrol pump, improving efficiency, accuracy, cost savings, customer service, compliance, and data analysis.

**Requirements :**

**Functional Requirements :**

* **Fuel Dispensing:** The system should be able to automate the fuel dispensing process and provide accurate readings of the amount of fuel dispensed for each transaction.
* **Inventory Management:** The system should be able to manage inventory levels, track orders and deliveries, and generate alerts when inventory levels fall below a certain threshold.
* **Sales Tracking and Reporting:** The system should be able to generate reports on sales data, including transaction volumes, revenue, and customer behaviour, allowing for effective analysis and decision-making.
* **Payment Processing:** The system should be able to process payments efficiently and securely and integrate with popular payment gateways to support multiple payment methods.
* **Customer Management:** The system should be able to manage customer information, track loyalty programs and rewards, and provide personalized services to customers.
* **Employee Management:** The system should be able to manage employee information, work schedules, and track employee performance.
* **Security:** The system should be designed with security in mind, protecting sensitive data and preventing unauthorized access.
* User Interface: The system should have an intuitive and user-friendly interface that allows petrol pump attendants to perform their tasks quickly and easily.
* **Integration:** The system should be able to integrate with other systems such as accounting software, fuel monitoring systems, and loyalty program providers.
* **Alerts and Notifications:** The system should be able to generate alerts and notifications for low inventory levels and suspicious transactions.
* **Data Analytics:** The system should be able to provide valuable insights into sales patterns, customer behaviour, and inventory levels, allowing petrol pump owners to make data-driven decisions and improve their operations.

**Non - Functional Requirements :**

* **Performance:** The system should be able to manage a high volume of transactions and provide a fast and responsive user experience.
* **Reliability:** The system should be dependable, with minimal downtime and data loss.
* **Security:** The system should be secure, protecting sensitive data and preventing unauthorized access.
* **Compatibility:** The system should be compatible with various hardware and software configurations.
* **Usability:** The system should be user-friendly, with an intuitive interface and easy-to-understand instructions.
* **Maintainability:** The system should be maintainable, with easy-to-understand code and documentation that allows for efficient updates and modifications.

**Required Modules :**

* tkinter
* customtkinter
* PIL
* uuid
* pickle
* csv
* time

**Code implementation :**

**Splash Screen**

This function is used to show the splash screen of the application. It will show the application name and a loading message.

class splashScreen:

    def \_\_init\_\_(self):

        root=Tk()

        width\_of\_window = 450

        height\_of\_window = 260

        screen\_width = root.winfo\_screenwidth()

        screen\_height = root.winfo\_screenheight()

        x\_coordinate = (screen\_width/2)-(width\_of\_window/2)

        y\_coordinate = (screen\_height/2)-(height\_of\_window/2)

        root.geometry("%dx%d+%d+%d" %(screen\_width,screen\_height,x\_coordinate,y\_coordinate))

        root.overrideredirect(1)

        Frame(root, width=570, height=250, bg='#272727').place(x=0,y=0)

        label1=Label(root, text='PETROL PUMP MANAGEMENT', fg='white', bg='#272727')

        label1.configure(font=("Berlin Sans FB Demi", 22))

        label1.place(x=50,y=90)

        label2=Label(root, text='Loading...', fg='white', bg='#272727')

        label2.configure(font=("Calibri", 14, 'bold'))

        label2.place(x=10,y=215)

        image\_a=ImageTk.PhotoImage(file='assets/c2.png')

        image\_b=ImageTk.PhotoImage(file='assets/c1.png')

        for i in range(3):

            l1=Label(root, image=image\_a, border=0, relief=SUNKEN).place(x=240, y=145)

            l2=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=260, y=145)

            l3=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=280, y=145)

            l4=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=300, y=145)

            root.update\_idletasks()

            time.sleep(0.2)

            l1=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=240, y=145)

            l2=Label(root, image=image\_a, border=0, relief=SUNKEN).place(x=260, y=145)

            l3=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=280, y=145)

            l4=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=300, y=145)

            root.update\_idletasks()

            time.sleep(0.2)

            l1=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=240, y=145)

            l2=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=260, y=145)

            l3=Label(root, image=image\_a, border=0, relief=SUNKEN).place(x=280, y=145)

            l4=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=300, y=145)

            root.update\_idletasks()

            time.sleep(0.2)

            l1=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=240, y=145)

            l2=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=260, y=145)

            l3=Label(root, image=image\_b, border=0, relief=SUNKEN).place(x=280, y=145)

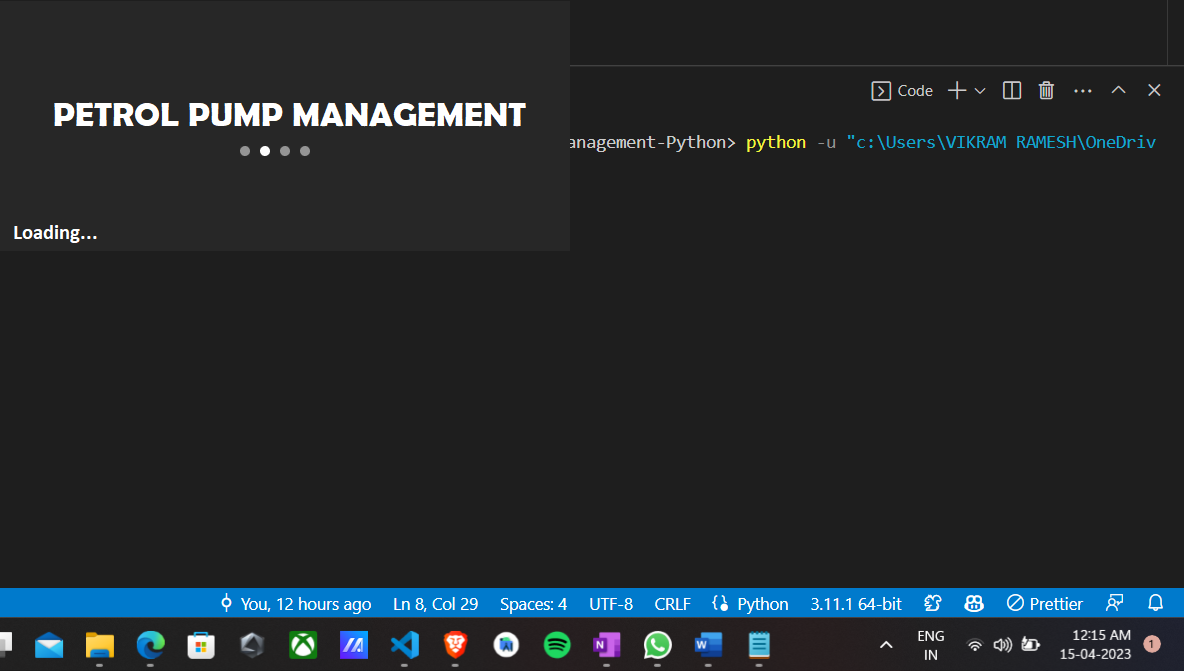
            l4=Label(root, image=image\_a, border=0, relief=SUNKEN).place(x=300, y=145)

            root.update\_idletasks()

            time.sleep(0.2)

        root.destroy()

        LoginPage()

        root.mainloop()

**Login Screen**

This function is used to create the login page of the application. Will show the username and password entry boxes. It will also show the login button.

def LoginPage():

    root1 = Tk()

    root1.withdraw()

    root = CTkToplevel()

    root.resizable(width= False, height= False)

    WW = 732

    WH = 450

    SW = root.winfo\_screenwidth()

    SH = root.winfo\_screenheight()

    x = SW/2 - WW/2

    y = SH/2 - WH/2

    root.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

    root.title('Petrol Pump Management')

    bg = ImageTk.PhotoImage(file='assets/background2\_small.jpeg')

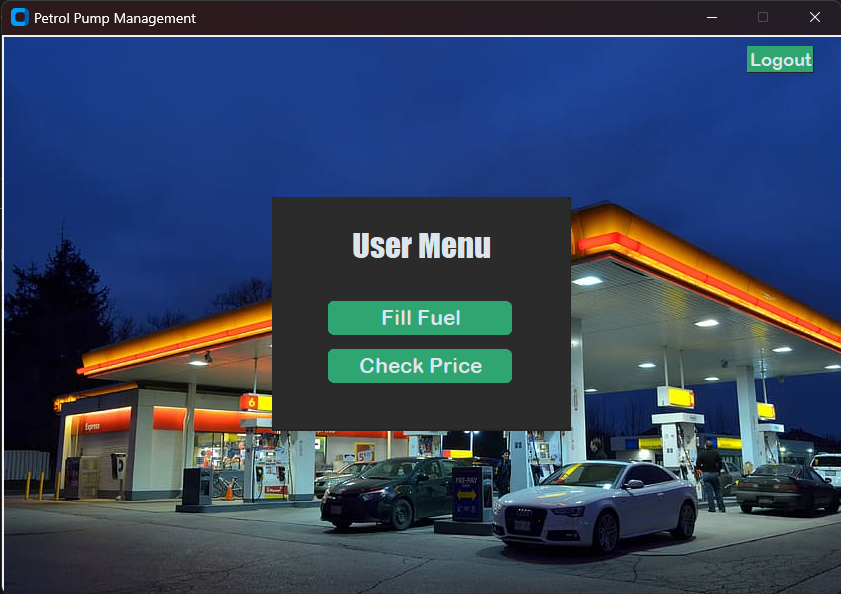
    label1 = Label(root, image = bg)

    label1.place(x = 0, y = 0)

    frame = CTkFrame(master = root, height=200, width=6000)

    frame.pack(pady=60, padx=60, expand=True)

    label = CTkLabel(master=frame, text='User Login', font=('Impact', 35,



**Admin Home Screen**

This function is used to show the admin menu. It will show the admin menu and the buttons to go to the other pages. It will also have a logout button to go back to the login page.

def admin():

    try:

        adminWindow = CTkToplevel()

        adminWindow.resizable(width= False, height= False)

        WW = 732

        WH = 450

        SW = adminWindow.winfo\_screenwidth()

        SH = adminWindow.winfo\_screenheight()

        x = SW/2 - WW/2

        y = SH/2 - WH/2

        adminWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

        adminWindow.title('Petrol Pump Management')

        bg = ImageTk.PhotoImage(file='assets/background.jpg')

        label1 = Label(adminWindow, image = bg)

        label1.place(x = 0, y = 0)

        frame = CTkFrame(master = adminWindow, height=200, width=6000)

        frame.pack(pady=60, padx=60, expand=True)

        label = CTkLabel(master=frame, text='Admin Menu', font=('Impact', 28))

        label.pack(pady=25, padx=70)

        filling = CTkButton(master=frame, text='Update  Price', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: updatePrice(adminWindow))

        filling.pack(pady=6, padx=10)

        filling = CTkButton(master=frame, text='Update  Quantity', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: updateQty(adminWindow))

        filling.pack(pady=6, padx=10)

        checkPrice = CTkButton(master=frame, text='Transactions', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: transactions(adminWindow))

        checkPrice.pack(pady=6, padx=10)

        label = CTkLabel(master=frame, text='', font=('Impact', 26))

        label.pack(pady=2, padx=40)

        logoutButton = CTkButton(master=adminWindow, text='Logout', font=('Arial Rounded MT Bold', 16), width=18, height=2, command= lambda: goBack(adminWindow))

        logoutButton.place(x= 650, y=10)

        adminWindow.mainloop()

    except Exception as e:

        messagebox.showerror(f'Python Error', 'Error: {e}')

Graphical user interface

Description automatically generated

**Customer Fuel Filling Page**

This class is used to create the fill page. It will show the fuel type options. It will also have a back button to go back to the homepage page.

def fillPage():

    fillWindow = CTkToplevel()

    fillWindow.resizable(width= False, height= False)

    WW = 732

    WH = 488

    SW = fillWindow.winfo\_screenwidth()

    SH = fillWindow.winfo\_screenheight()

    x = SW/2 - WW/2

    y = SH/2 - WH/2

    fillWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

    fillWindow.title('Petrol Pump Management')

    bg = ImageTk.PhotoImage(file='assets/background.jpg')

    label1 = Label(fillWindow, image = bg)

    label1.place(x = 0, y = 0)

    frame = CTkFrame(master = fillWindow, height=200, width=600)

    frame.pack(pady=60, padx=60, expand=True)

    label = CTkLabel(master=frame, text='Choose Fuel Type', font=('Impact', 28))

    label.pack(pady=25, padx=70)

    Petrol = CTkButton(master=frame, text='Petrol', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: petrol(fillWindow))

    Petrol.pack(pady=20, padx=30, side= LEFT)

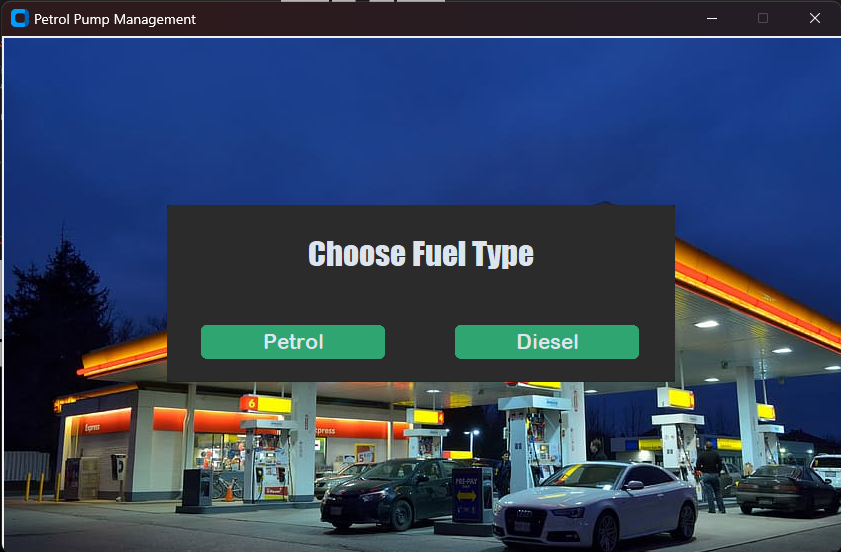
    Diesel = CTkButton(master=frame, text='Diesel', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: diesel(fillWindow))

    Diesel.pack(pady=20, padx=30, side= RIGHT)

    backButton = CTkButton(master=fillWindow, text='<- Back', font=('Arial Rounded MT Bold', 16), width=18, height=2, command= lambda: goBack(fillWindow))

    backButton.place(x= 5, y=5)

    fillWindow.mainloop()



**Customer Fuel Price Checking Page**

This function is used to show the current price of petrol and diesel. It will show the current price of petrol and diesel. It will also have a back button to go back to the home page. It will also have a button to update the price of petrol and diesel. It will also have an entry to enter the new price of petrol and diesel. It will also have a button to confirm the update. It will destroy the current window and open the update rate page.

def CheckPrice():

    try:

        fillWindow = CTkToplevel()

        fillWindow.resizable(width= False, height= False)

        WW = 732

        WH = 450

        SW = fillWindow.winfo\_screenwidth()

        SH = fillWindow.winfo\_screenheight()

        x = SW/2 - WW/2

        y = SH/2 - WH/2

        fillWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

        fillWindow.title('Petrol Pump Management')

        ic = Image.open('assets/background3.jpg')

        res\_img = ic.resize((910,605))

        bg = ImageTk.PhotoImage(res\_img)

        label1 = Label(fillWindow, image = bg)

        label1.place(x = 0, y = 0)

        frame = CTkFrame(master = fillWindow, height=200, width=6000)

        frame.pack(pady=60, padx=60, expand=True)

        currentPetrolPrice = dataHandling.getData(r'data\petrolPrice.pkl')

        currentDieselPrice = dataHandling.getData(r'data\dieselPrice.pkl')

        label1 = CTkLabel(master=frame, text='Current Price Details', font=('Arial Rounded MT Bold', 24), width=162, height=30)

        label1.pack(pady=8, padx=70)

        label2 = CTkLabel(master=frame, text=f'Petrol : {currentPetrolPrice}', font=('Arial Rounded MT Bold', 18), width=162, height=30)

        label2.pack(pady=5, padx=70)

        label3 = CTkLabel(master=frame, text=f'Diesel : {currentDieselPrice}', font=('Arial Rounded MT Bold', 18), width=162, height=30)

        label3.pack(pady=5, padx=70)

        sizeBox = CTkLabel(master=frame, text=f' ', font=('Arial Rounded MT Bold', 2), width=162, height=10)

        sizeBox.pack(pady=20, padx=10)

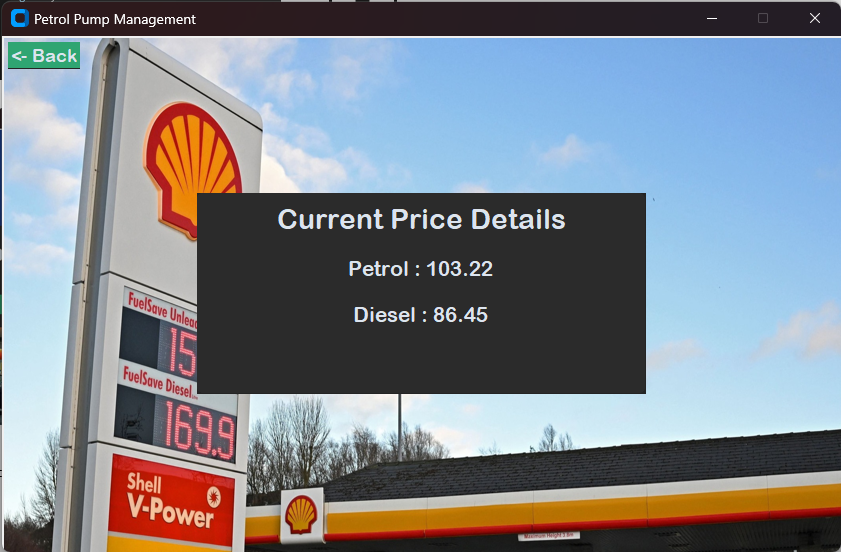
        backButton = CTkButton(master=fillWindow, text='<- Back', font=('Arial Rounded MT Bold', 16), width=18, height=2, command= lambda: goBack(fillWindow))

        backButton.place(x= 5, y=5)

        fillWindow.mainloop()

    except Exception as e:

        messagebox.showerror(f'Python Error', 'Error: {e}')



**Admin Fuel Price updating Page.**

This function is used to create the update rate page. It will show the current price of petrol and diesel. It will also have a back button to go back to the admin page. It will also have a button to update the price of petrol and diesel. It will also have an entry to enter the new price of petrol and diesel. It will also have a button to confirm the update. It will destroy the current window and open the update rate page.

def updateRate():

    try:

        costWindow = CTkToplevel()

        costWindow.resizable(width= False, height= False)

        WW = 732

        WH = 450

        SW = costWindow.winfo\_screenwidth()

        SH = costWindow.winfo\_screenheight()

        x = SW/2 - WW/2

        y = SH/2 - WH/2

        costWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

        costWindow.title('Petrol Pump Management')

        ic = Image.open('assets/background3.jpg')

        res\_img = ic.resize((910,605))

        bg = ImageTk.PhotoImage(res\_img)

        label1 = Label(costWindow, image = bg)

        label1.place(x = 0, y = 0)

        frame = CTkFrame(master = costWindow, height=500, width=60)

        frame.pack(pady=60, padx=60, expand=True)

        currentPetrolPrice = dataHandling.getData(r'data\petrolPrice.pkl')

        currentDieselPrice = dataHandling.getData(r'data\dieselPrice.pkl')

        label1 = CTkLabel(master=frame, text='Current Price Details', font=('Arial Rounded MT Bold', 24), width=162, height=30)

        label1.pack(pady=8, padx=70)

        label2 = CTkLabel(master=frame, text=f'Petrol : {currentPetrolPrice}', font=('Arial Rounded MT Bold', 18), width=162, height=30)

        label2.pack(pady=5, padx=70)

        label3 = CTkLabel(master=frame, text=f'Diesel : {currentDieselPrice}', font=('Arial Rounded MT Bold', 18), width=162, height=30)

        label3.pack(pady=5, padx=70)

        sizeBox = CTkLabel(master=frame, text=f' ', font=('Arial Rounded MT Bold', 50), width=162, height=30)

        sizeBox.pack(pady=20, padx=10)

        label4 = CTkLabel(master=frame, text=f'Enter New Petrol Rate', font=('Arial Rounded MT Bold', 15), width=100, height=30)

        label4.place(x=20, y=160)

        entry1 = CTkEntry(master=frame, placeholder\_text='Update Petrol Price')

        entry1.place(x=216, y=160)

        label5 = CTkLabel(master=frame, text=f'Enter New Diesel Rate', font=('Arial Rounded MT Bold', 15), width=162, height=30)

        label5.place(x=20, y=200)

        entry2 = CTkEntry(master=frame, placeholder\_text='Update Diesel Price')

        entry2.place(x=216, y=200)

        button = CTkButton(master=frame, text='Update Price', command= lambda: updatePrice(entry1, entry2, costWindow))

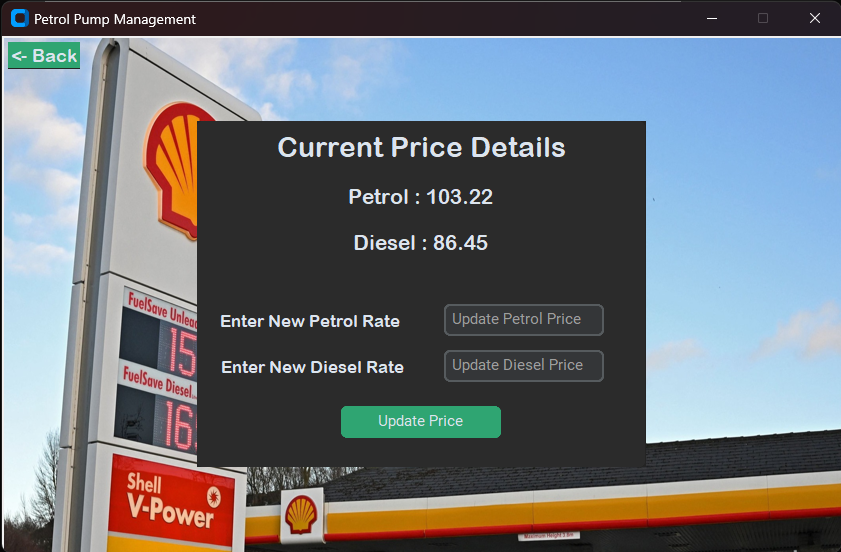
        button.pack(pady=25, padx=10)

        backButton = CTkButton(master=costWindow, text='<- Back', font=('Arial Rounded MT Bold', 16), width=18, height=2, command= lambda: goBack(costWindow))

        backButton.place(x= 5, y=5)

        costWindow.mainloop()

    except Exception as e:

        messagebox.showerror(f'Python Error', 'Error: {e}')

**Transaction History Page**

This function is used to show the transaction page. It will show the transaction page and the buttons to go to the other pages.

def transactionPage():

    transactionWindow = CTkToplevel()

    transactionWindow.resizable(width= False, height= False)

    WW = 732

    WH = 450

    SW = transactionWindow.winfo\_screenwidth()

    SH = transactionWindow.winfo\_screenheight()

    x = SW/2 - WW/2

    y = SH/2 - WH/2

    transactionWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

    transactionWindow.title('Petrol Pump Management')

    bg = ImageTk.PhotoImage(file='assets/background2\_small.jpeg')

    label1 = Label(transactionWindow, image = bg)

    label1.place(x = 0, y = 0)

    mainFrame = Frame(transactionWindow, bg='darkgray')

    mainFrame.pack(side=LEFT, padx=20)

    label = CTkLabel(master=mainFrame, text='Transaction History', font=('Impact', 35, 'bold'))

    label.pack(pady=10, padx=40)

    frm = Frame(mainFrame)

    frm.pack(side=LEFT, padx=20, pady=10)

    tv = ttk.Treeview(frm, columns=(1, 2, 3, 4), show='headings', height='20')

    tv.pack()

    backButton = CTkButton(master=transactionWindow, text='<- Back', font=('Arial Rounded MT Bold', 16), width=18, height=2, command= lambda: goBack(transactionWindow))

    backButton.place(x= 5, y=5)

    file = open(r'data\transactions.csv')

    csvreader = csv.reader(file)

    r\_set = [row for row in csvreader]

    tv.heading(1, text='ID', anchor=CENTER)

    tv.heading(2, text='Transaction Detail', anchor=W)

    tv.heading(3, text='Debit')

    tv.heading(4, text='Credit')

    for dat in r\_set:

        v = [r for r in dat]

        tv.insert('', 'end', iid=v[0], values=v)

    transactionWindow.mainloop()

Table

Description automatically generated

**Fuel Filling Page :**

This class is used to create the petrol page. It will show the current price and quantity of petrol. It will also have a back button to go back to the homepage page. It will also have a button to fill the petrol. It will also have an entry to enter the quantity of petrol to be filled.

def PetrolPage():

    crPrice = dataHandling.getData(r'data\petrolPrice.pkl')

    crQty = dataHandling.getData(r'data\petrolQty.pkl')

    # print(availPetrol)

    petrolWindow = CTkToplevel()

    petrolWindow.resizable(width= False, height= False)

    WW = 732

    WH = 450

    SW = petrolWindow.winfo\_screenwidth()

    SH = petrolWindow.winfo\_screenheight()

    x = SW/2 - WW/2

    y = SH/2 - WH/2

    petrolWindow.geometry('%dx%d+%d+%d' %(WW, WH, x, y))

    petrolWindow.title('Petrol Pump Management')

    bg = ImageTk.PhotoImage(file='assets/background.jpg')

    label1 = Label(petrolWindow, image = bg)

    label1.place(x = 0, y = 0)

    frame = CTkFrame(master = petrolWindow, height=200, width=600)

    frame.pack(pady=60, padx=60, expand=True)

    label = CTkLabel(master=frame, text='Petrol', font=('Impact', 28))

    label.pack(pady=9, padx=70)

    label = CTkLabel(master=frame, text=f'Current Petrol Price : {crPrice}', font=('Impact', 16))

    label.pack(pady=5, padx=70)

    label = CTkLabel(master=frame, text=f'Petrol Quantity : {crQty}', font=('Impact', 16))

    label.pack(pady=5, padx=70)

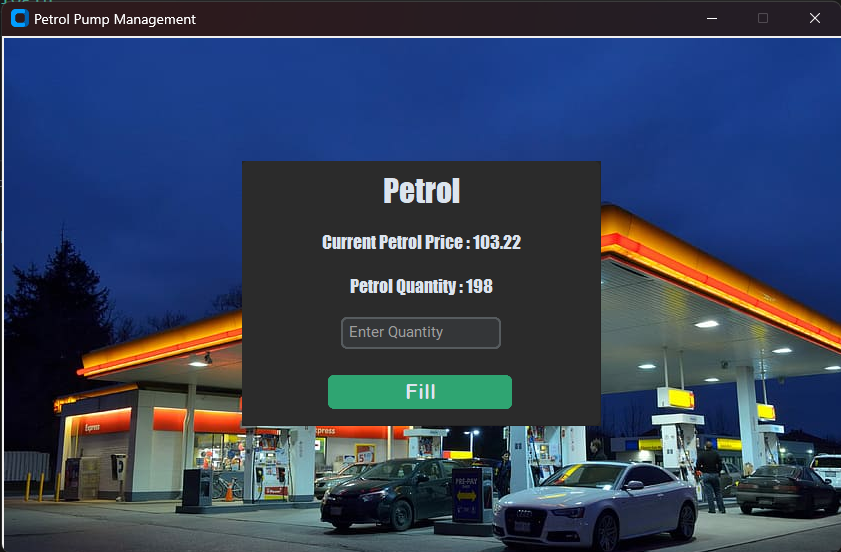
    entry1 = CTkEntry(master=frame, placeholder\_text='Enter Quantity')

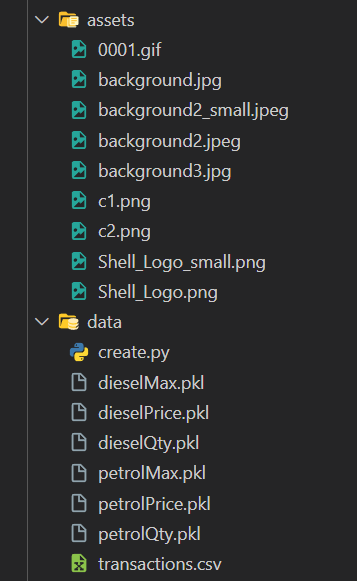
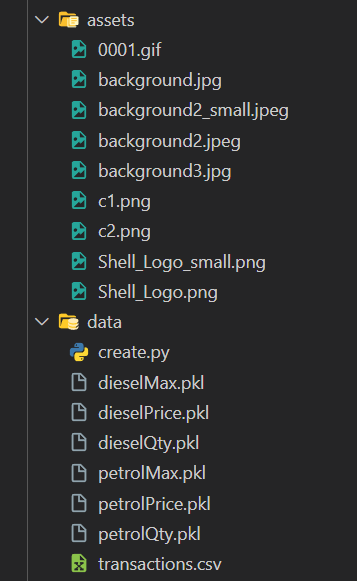
    entry1.pack(pady=8, padx=10)

    filling = CTkButton(master=frame, text='Fill', font=('Arial Rounded MT Bold', 18), width=162, height=30, command= lambda: fillLitreFunction(entry1, petrolWindow))

    filling.pack(pady=15, padx=30)

    petrolWindow.mainloop()



**Data Storage and Assets :**

**Test Cases :**

**Login :**

* Correct username and password : Authentication Successful
* Correct username and wrong Password : Incorrect Password
* Wrong username and any password : Invalid username

**Sell Fuel :**

* If required Quantity is less than available Quantity : Filling Successful
* If required Quantity is more than available Quantity : Insufficient Fuel

**Buy Fuel :**

* If adding Qty is less than Available Qty + adding Qty : Filling Successful
* If adding Qty is more than Available Qty + adding Qty : Storage Overloading

**Project Outcomes :**

The petrol pump management system project can lead to significant improvements in efficiency, customer service, data analysis, cost savings, security, employee performance, and competitive advantage, providing a comprehensive solution for petrol pump owners.

**Conclusion :**

In conclusion, a petrol pump management system project developed using Python can provide significant benefits to petrol pump owners by automating various tasks, improving operational efficiency, enhancing customer service, and providing valuable data analysis. The project's scope includes functional and non-functional requirements, such as fuel dispensing, inventory management, payment processing, security, performance, and scalability. The project's outcomes can lead to cost savings, increased security, improved employee performance, competitive advantage, and enhanced customer experience. With proper planning, design, and execution, a petrol pump management system project can be an asset for petrol pump owners, providing a modern and efficient solution for managing their operations.