D.A.V. PUBLIC SCHOOL, AIROLI

COMPUTER SCIENCE PROJECT

ON

DaCoolStore General Store

SESSION – 2023-24

Name : **Akhil Tyagi**

Class : **XII - A**

Board Roll No : _____

CERTIFICATE

This is to certify that Master/Miss Akhil Tyagi of Std XII, D.A.V. Public School, Airoli , has successfully completed the project entitled DaCoolStore General Store using Python / Python & MySQL during the academic year 2023-24.

It is further certified that the project is the genuine work of the student and has been done sincerely and satisfactorily.

Internal Examiner's Signature	External Examiner's Signature	Principal's Signature

ACKNOWLEDGEMENT

I would like to express my heartfelt gratitude to the individuals who have played a significant role in shaping my academic and professional journey. Their unwavering support, guidance, and encouragement have been instrumental in my growth, and I am truly thankful for their contributions.

First and foremost, I extend my deepest appreciation to Mrs. Manisha Desai, who has been both my guide and teacher at D.A.V. Public School. Her wisdom, mentorship, and dedication to my education have been invaluable. Her insights and teachings have not only enriched my knowledge but also inspired me to strive for excellence.

I extend my sincere thanks to Mrs. Suman Pradhan, the Principal at D.A.V. Public School. Her leadership and commitment to academic excellence have created an environment conducive to learning and growth. Her support has been pivotal in my academic pursuits.

I would also like to acknowledge my colleagues who have been an essential part of my academic journey. Their camaraderie, collaboration, and collective pursuit of knowledge have made the learning experience at D.A.V. Public School enriching and enjoyable.

INDEX

INTRODUCTION

The Super Market General Store based Python coding project aims to develop a comprehensive and user-friendly system for managing a fictional store. The project integrates key functionalities such as customer registration, product management, order processing, and staff administration into a Graphical User Interface (GUI) using the PySimpleGUI library. The system is designed to enable efficient handling of customer orders, product additions, and updates, while providing staff members with a secure login system.

Customers can register, browse available products, and place orders seamlessly through an intuitive interface. The product management module allows staff members to add, modify, or remove products, ensuring an up-to-date inventory. The project also incorporates features for staff login, enhancing security and control over the system. Furthermore, the system integrates a database using MySQL to store and retrieve essential information, fostering data persistence and enabling seamless data management. The GUI components, designed with PySimpleGUI, provide a pleasing and interactive environment for users.

In addition to the primary functionalities, the project introduces advanced features such as updating the delivery status of orders and tracking customer orders. These enhancements contribute to an enhanced user experience and operational efficiency. Ultimately, this project serves as a scalable solution for a store management system, showcasing the versatility of Python in creating practical and user-centric applications.

FLOWCHART

SOURCE CODE

1)main_file.py

```
from functions import *

# Main function
if __name__ == "__main__":
    main()
```

2)functions.py

```
import mysql.connector
import PySimpleGUI as sg
from datetime import datetime

# Establishing variable for time
f = (datetime.now()).strftime('''Date: %d-%m-%y
Time: %H:%M:%S''')

# Function 1: Establish a connection to the MySQL database
def connect_to_database():
    try:
        db= mysql.connector.connect(
            host='sql12.freemysqlhosting.net',
            user='sql12663296',
            password='5tsIMMy6fP',
            database='sql12663296'
        )
        return db

    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None

# Function 2: Create a login window for staff
def staff_login_window(db):
    layout = [
        [sg.Text("Staff ID:", font=('helvetica', 20),text_color="black"), sg.InputText(key='-
ID-'),],
```

```
        [sg.Text(("Password:"),
font=('helvetica',20),text_color=("black")),sg.InputText(key='-PASSWORD-',
password_char='*')],
        [sg.Button("Login", bind_return_key=True,size=(20,1)), sg.Exit(size=(20,1)),
sg.Button(("Back"),size=(20,1))]
    ]
    window = sg.Window("Staff Login", layout, finalize=True,)
    cursor = db.cursor()

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, "Exit"):
            break
        if event == "Back":
           window.close()
           main()
        elif event == 'Login':
            window.hide()
            cid = values['-ID-']
            password = values['-PASSWORD-']
            db = connect_to_database()
            cursor = db.cursor()
            insert_query = "INSERT INTO attendance (staffid) VALUES (%s);"
            values = (cid,)
            cursor.execute(insert_query, values)
            db.commit()

            if cid == '1' and password in ('akhil1202','a','1202'):
                sg.popup('Admin functionality acquired. Please use with care \n\n',f)
                admin_window()
                break
            else:
                cursor.execute("SELECT * FROM staff WHERE Staff_ID=%s AND Password=%s;",
(cid, password))
                user = cursor.fetchone()
                if user:
                    sg.popup("Login Successful!", f"Welcome, {user[1]}. Have a nice day ^^
\n"
                    "\n",f)
                    window.close()
                    staff_options_window()
                else:
                    sg.popup_error("Login Failed", "Invalid username or password")

    db.close()
    window.close()
```

```
# Function 3: Special administrator window access only for some people
def admin_window():
    layout = [
        [sg.Text(("AUTHORISED ACCESS ONLY,"),font=('helvetica', 10),text_color="red")],
        [sg.Text(("DO NOT COMMIT ANY CHANGES WHEN NOT TOLD TO DO SO."),font=('helvetica',
10),text_color="red")],
        [sg.Button(('Employee Attendance'),size=(35,2)), sg.Button('Employee
Info',size=(35,2))],
        [sg.Button('Customer Orders',size=(35,2)), sg.Button('Customer
Details',size=(35,2))],
        [sg.Exit(size=(35,2)), sg.Button('Logout',size=(35,2))]
    ]
    window = sg.Window('Admin Window', layout, finalize=True,)

    while True:
        event, _ = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Employee Attendance':
            window.hide()
            display_attendance_tab(connect_to_database())
        elif event == 'Employee Info':
            window.hide()
            employee_info_tab(connect_to_database())
        elif event == 'Customer Orders':
            window.hide()
            display_all_orders_admin(connect_to_database())
        elif event == 'Customer Details':
            window.hide()
            fetch_all_and_displaywindow_admintab(connect_to_database())
        elif event == 'Logout':
            window.hide()
            staff_login_window(connect_to_database())

    window.close()

# Function 4: To display all customer orders for admin
def display_all_orders_admin(db):
    layout = [
        [sg.Text("All Orders", font=("Helvetica", 18))],
        [sg.Table(values=[], headings=["Cust. ID", "Phone No.", "Delivery Address", "Note",
"Time", "Status"],
                  auto_size_columns=False, col_widths=[6, 13, 25, 20, 15, 13],
display_row_numbers=False,
                  justification="center", num_rows=20, key='-TABLE-')],
        [sg.Exit(), sg.Button('Back')]
    ]
```

```
    window = sg.Window("All Orders", layout, finalize=True)
    try:
        cursor = db.cursor(dictionary=True)
        cursor.execute("SELECT * FROM orders JOIN customers ON orders.Cust_PhoneNumber =
customers.Cust_PhoneNumber;")
        orders = cursor.fetchall()
        table_data = []
        for order in orders:
            table_data.append([order["Cust_ID"], order["Cust_PhoneNumber"],
order["Cust_HomeAddress"], order["Note"], order["Time"], order["Delivery_Status"]])
        window['-TABLE-'].update(values=table_data)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, "Exit"):
            break
        if event == 'Back':
            window.close()
            admin_window()

    db.close()
    window.close()

# Function 5: To display customers tab for admin
def fetch_all_and_displaywindow_admintab(db):
    try:
        cursor = db.cursor()
        cursor.execute("SELECT * from customers")
        customer_data = cursor.fetchall()
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    layout = [
        [sg.TabGroup([
            [sg.Tab('Retrieve Customer Info', [
                [sg.Table(values=customer_data, headings=["CustID", "Name", "Gender",
"Email", "Password"], auto_size_columns=True, display_row_numbers=False,
justification="center", num_rows=10, key='-TABLE-')],
            ])],
            [sg.Tab('Delete Customer', [
                [sg.Text('Enter Customer ID:'), sg.InputText(key='-CUSTOMER_ID-')],
                [sg.Text('Enter Customer Name:'), sg.InputText(key='-CUSTOMER_NAME-')],
                [sg.Button('Delete')]
            ])],
            [sg.Tab('Update Customer Info', [
```

```
                [sg.Text('Enter Customer ID:'), sg.InputText(key='-CUSTOMER_ID2-')],
                [sg.Button('Submit')],
                [sg.Text(key='-CUSTOMER_INFO-')],
                [sg.Button('Change Customer Name'), sg.Button('Change Customer Gender'),
sg.Button('Change Email Address'), sg.Button('Change Password')]
            ])],
        ])],
        [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window('Customer Information', layout)

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Back':
            window.hide()
            admin_window()
        elif event == 'Delete':
            customer_id = values['-CUSTOMER_ID-']
            customer_name = values['-CUSTOMER_NAME-']
            if customer_id and customer_name:
                try:
                    cursor.execute('DELETE FROM customers WHERE Cust_ID=%s AND
Cust_Name=%s;', (customer_id, customer_name))
                    db.commit()
                    sg.popup('Customer Deleted Successfully!')
                except mysql.connector.Error as err:
                    sg.popup_error(f"Error: {err}")
        elif event == 'Submit':
            customer_id = values['-CUSTOMER_ID2-']
            cursor.execute('SELECT * FROM customers WHERE Cust_ID=%s', (customer_id,))
            customer_info = cursor.fetchone()
            print(customer_info)
            if customer_info:
                window['-CUSTOMER_INFO-'].update(f"Customer ID: {customer_info[0]}, Customer
Name: {customer_info[1]}, Gender: {customer_info[2]}\nEmail Address: {customer_info[3]},
Phone Number: {customer_info[4]}, Password: {customer_info[5]}")
        elif event == 'Change Customer Name':
            new_customer_name = sg.popup_get_text('Enter new customer name:')
            if new_customer_name:
                cursor = db.cursor()
                cursor.execute('UPDATE customers SET Cust_Name=%s WHERE Cust_ID=%s',
(new_customer_name, customer_id))
                db.commit()
                sg.popup('Customer Name changed successfully!')
        elif event == 'Change Customer Gender':
```

```
            new_customer_gender = sg.popup_get_text('Enter new customer gender:')
            if new_customer_gender:
                cursor = db.cursor()
                cursor.execute('UPDATE customers SET Cust_Gender=%s WHERE Cust_ID=%s',
(new_customer_gender, customer_id))
                db.commit()
                sg.popup('Customer Gender changed successfully!')
        elif event == 'Change Email Address':
            new_email_address = sg.popup_get_text('Enter new email address:')
            if new_email_address:
                cursor = db.cursor()
                cursor.execute('UPDATE customers SET Cust_EmailAddress=%s WHERE Cust_ID=%s',
(new_email_address, customer_id))
                db.commit()
                sg.popup('Email Address changed successfully!')
        elif event == 'Change Password':
            new_password = sg.popup_get_text('Enter new password:')
            if new_password:
                cursor = db.cursor()
                cursor.execute('UPDATE customers SET Cust_Password=%s WHERE Cust_ID=%s',
(new_password, customer_id))
                db.commit()
                sg.popup('Password changed successfully!')

    db.close()
    window.close()

# Function 6: To display staff attendance tab to admin
def display_attendance_tab(db):
    cursor = db.cursor()
    cursor.execute("SELECT * FROM attendance;")
    attendance_data = cursor.fetchall()
    layout = [
        [sg.TabGroup([
        [sg.Tab('Attendance',[
            [sg.Table(values=attendance_data, headings=['Attendance ID', 'Staff ID', 'Staff
Name', 'Timestamp'],
                      auto_size_columns=True, display_row_numbers=False,
justification='center',
                      key='-ATTENDANCE_TABLE-', enable_events=True)]
        ])],
        [sg.Tab('Check Staff Attendance', [
            [sg.Text("Enter Staff ID:", font=('Courier', 16)), sg.InputText(key='-STAFF_ID-
')],
            [sg.Button("Fetch Timestamps")],
            [sg.Output(size=(50, 10), key='-OUTPUT-')]
        ])],
```

```
    ])],
    [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window('Staff Attendance', layout)

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED,'Exit'):
            break
        elif event == 'Back':
            window.hide()
            admin_window()
        elif event == 'Fetch Timestamps':
            staff_id = values['-STAFF_ID-']
            if staff_id:
                cursor.execute("SELECT Timestamp FROM attendance WHERE StaffID = %s;",
(staff_id,))
                timestamps = cursor.fetchall()
                if timestamps:
                    timestamp_text = '\n'.join(str(timestamp[0]) for timestamp in timestamps)
                    window['-OUTPUT-'].update(timestamp_text)
                else:
                    window["-OUTPUT-"].update(f'No timestamps found for StaffID {staff_id}')
            else:
                sg.popup_error('Please enter Staff ID.')

    db.close()
    window.close()

# Function 7: To display employee info tab to admin
def employee_info_tab(db):
    try:
        cursor = db.cursor()
        cursor.execute('SELECT* FROM staff;')
        staff_info = cursor.fetchall()
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    layout = [
        [sg.TabGroup([
            [sg.Tab('Retrieve Staff Info', [
                [sg.Table(values=staff_info, headings=["StaffID", "Name", "Post", "Gender",
"Password"], auto_size_columns=False, display_row_numbers=False, justification="center",
num_rows=10, key='-TABLE-')],
            ])],
            [sg.Tab('Enter New Staff Info', [
                [sg.Text("Staff ID:", font=('Courier', 16)), sg.InputText(key='-STAFF_ID-')],
```

```
                [sg.Text("Staff Name:", font=('Courier', 16)), sg.InputText(key='-STAFF_NAME-
')],
                [sg.Text("Staff Post:", font=('Courier', 16)), sg.InputText(key='-STAFF_POST-
')],
                [sg.Text("Gender:", font=('Courier', 16)), sg.InputText(key='-GENDER-')],
                [sg.Text("Password:", font=('Courier', 16)), sg.InputText(key='-PASSWORD-',
password_char='*')],
                [sg.Button("Submit Now")],
            ])],
            [sg.Tab('Delete Staff Info', [
                [sg.Text('Enter Staff ID:'), sg.InputText(key='-ID-')],
                [sg.Text('Enter Staff Name:'), sg.InputText(key='-NAME-')],
                [sg.Button('Submit')]
            ])],
            [sg.Tab('Update Staff Info', [
                    [sg.Text('Enter Staff ID:'), sg.InputText(key='-STAFF_ID2-')],
                    [sg.Button('Submit now')],
                    [sg.Text(key='-STAFF_INFO-')],
                    [sg.Button('Change Staff Name'), sg.Button('Change Staff Post'),
sg.Button('Change Gender'), sg.Button('Change Password')],
                    [sg.Text('(Changes will be visible once you reload this window.)')]
            ])]
        ])],
        [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window('Employees', layout)

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Back':
            window.hide()
            admin_window()
        elif event == 'Submit':
            cid = values['-ID-']
            name = values['-NAME-']
            if cid==1 and name=='Akhil Tyagi':
                sg.popup('You cant delete that bro.')
                break
            elif not cid==1 and not name=='Akhil Tyagi':
                try:
                    cursor = db.cursor()
                    insert_query = 'DELETE FROM staff WHERE Staff_ID=%s and Staff_Name=%s;'
                    values = (cid, name)
                    cursor.execute(insert_query, values)
                    db.commit()
```

```
                    sg.popup('Successful')
                except mysql.connector.Error as err:
                    print(f"Error: {err}")
        elif event == 'Submit Now':
            staff_id = values['-STAFF_ID-']
            staff_name = values['-STAFF_NAME-']
            staff_post = values['-STAFF_POST-']
            gender = values['-GENDER-']
            password = values['-PASSWORD-']
            if staff_id and staff_name and staff_post and gender and password:
                try:
                    insert_query2 = "INSERT INTO staff (Staff_ID, Staff_Name, Staff_Post,
Gender, Password) VALUES (%s, %s, %s, %s, %s);"
                    values2 = (staff_id, staff_name, staff_post, gender, password)
                    cursor.execute(insert_query2, values2)
                    db.commit()
                    sg.popup('Staff Added Successfully!')
                except mysql.connector.Error as err:
                    sg.popup_error(f"Error: {err}")
            else:
                sg.popup_error("Please fill in all fields.")
        elif event == 'Submit now':
            staff_id2 = values['-STAFF_ID2-']
            cursor.execute('SELECT * FROM staff WHERE Staff_ID=%s', (staff_id2,))
            staff_info = cursor.fetchone()
            if staff_info:
                window['-STAFF_INFO-'].update(f"Staff ID: {staff_info[0]}, Staff Name:
{staff_info[1]}, Staff Post: {staff_info[2]}\nGender: {staff_info[3]}, Password:
{staff_info[4]}")
        elif event == 'Change Staff Name':
            new_staff_name = sg.popup_get_text('Enter new staff name:')
            if new_staff_name:
                cursor.execute('UPDATE staff SET Staff_Name=%s WHERE Staff_ID=%s',
(new_staff_name, staff_id2))
                db.commit()
                sg.popup('Staff Name changed successfully!')
        elif event == 'Change Staff Post':
            new_staff_post = sg.popup_get_text('Enter new staff post:')
            if new_staff_post:
                cursor = db.cursor()
                cursor.execute('UPDATE staff SET Staff_Post=%s WHERE Staff_ID=%s',
(new_staff_post, staff_id2))
                db.commit()
                sg.popup('Staff Post changed successfully!')
        elif event == 'Change Gender':
            new_gender = sg.popup_get_text('Enter new gender:')
            if new_gender:
```

```
                cursor = db.cursor()
                cursor.execute('UPDATE staff SET Gender=%s WHERE Staff_ID=%s', (new_gender,
staff_id2))
                db.commit()
                sg.popup('Gender changed successfully!')
        elif event == 'Change Password':
            new_password = sg.popup_get_text('Enter new password:')
            if new_password:
                cursor = db.cursor()
                cursor.execute('UPDATE staff SET Password=%s WHERE Staff_ID=%s',
(new_password, staff_id2))
                db.commit()
                sg.popup('Password changed successfully!')

    db.close()
    window.close()

# Function 8: Once the staff logins asking them what they want to do
def staff_options_window():
    layout = [
        [sg.Text("Staff Options", font=("Helvetica", 18),text_color="black")],
        [sg.Text(("What do you want to do now? "), font=("Helvetica",
15),text_color="black")],
        [sg.Button(("Customer Orders"),size=(30,2)), sg.Button(('Customer
Details'),size=(30,2))],
        [sg.Button("Products",size=(30,2)), sg.Button('Categories',size=(30,2))],
        [sg.Exit(size=(30,2)), sg.Button('Logout',size=(30,2))]
    ]
    window = sg.Window("Staff Options", layout, finalize=True)

    while True:
        event, _ = window.read()
        if event in ('Customer Orders','Customer Details','Products','Categories'):
            window.hide()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        if event == 'Logout':
            window.hide()
            staff_login_window(connect_to_database())
        elif event == "Customer Orders":
            display_all_orderstab(connect_to_database())
        elif event == "Customer Details":
            fetch_all_and_displaywindowtab(connect_to_database())
        elif event == "Products":
            add_product_to_dbtab(connect_to_database())
        elif event == "Categories":
            add_category_to_dbtab(connect_to_database())
```

```
    window.close()

# Function 9: To display all the orders
def display_all_orderstab(db):
    layout = [
        [sg.TabGroup([
            [sg.Tab('Display Orders', [
                [sg.Text("All Orders", font=("Helvetica", 18))],
                [sg.Table(values=[], headings=["Cust. ID", "OrderID", "Phone No.", "Delivery
Address", "Note", "Time", "Status"],
                          auto_size_columns=False, col_widths=[6, 8, 13, 25, 20, 15, 13],
display_row_numbers=False, justification="center", num_rows=20, key='-TABLE-')]
            ])],
            [sg.Tab('Update Orders', [
                [sg.Text('Enter Order ID:'), sg.InputText(key='-ORDER_ID-')],
                [sg.Text('Select Delivery Status:'), sg.Radio('Delivered', group_id='-
DELIVERY_STATUS-', key='-DELIVERED-', default=False), sg.Radio('Not Delivered', group_id='-
DELIVERY_STATUS-', key='-NOT_DELIVERED-', default=False)],
                [sg.Button('Update')]
            ])],
        ])],
        [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window('Customer Orders', layout, finalize=True)
    try:
        cursor = db.cursor(dictionary=True)
        cursor.execute("SELECT * FROM orders JOIN customers ON orders.Cust_PhoneNumber =
customers.Cust_PhoneNumber;")
        orders = cursor.fetchall()
        table_data = []
        for order in orders:
            table_data.append([order["Cust_ID"], order["OrderID"], order["Cust_PhoneNumber"],
order["Cust_HomeAddress"], order["Note"], order["Time"], order["Delivery_Status"]])
        window['-TABLE-'].update(values=table_data)
    except mysql.connector.Error as err:
        print(f"Error: {err}")

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Back':
            window.hide()
            staff_options_window()
        elif event == 'Update':
            order_id = values['-ORDER_ID-']
```

```
            delivered = values['-DELIVERED-']
            if order_id:
                try:
                    delivery_status = 'Delivered' if delivered else 'Not Delivered'
                    cursor.execute('UPDATE orders SET Delivery_Status=%s WHERE OrderID=%s;',
(delivery_status, order_id))
                    db.commit()
                    sg.popup('Delivery Status Updated Successfully!')
                except mysql.connector.Error as err:
                    sg.popup_error(f"Error: {err}")

    db.close()
    window.close()


# Function 10: To fetch all from the table customers and display it in a window
see_customer_data
def fetch_all_and_displaywindowtab(db):
    try:
        cursor = db.cursor()
        cursor.execute("SELECT * from customers")
        customer_data = cursor.fetchall()
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    layout = [
        [sg.Text("Customer Details", font=("Helvetica", 18))],
        [sg.Table(values=customer_data, headings=["CustID", "Name", "Gender", "Email",
"Password"], auto_size_columns=True,
                  display_row_numbers=False, justification="center", num_rows=10, key='-
TABLE-')],
        [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window("Customer Details", layout, finalize=True)

    while True:
        event, _ = window.read()
        if event in (sg.WIN_CLOSED, "Exit"):
            break
        if event == 'Back':
            window.hide()
            staff_options_window()

    db.close()
    window.close()

# Function 11: To take inputs from the staff for the new product and then insert the values
into the products table
```

```
def add_product_to_dbtab(db):
    layout = [
        [sg.TabGroup([
            [sg.Tab('Add Product', [
                [sg.Text("Add New Product", font=("Helvetica", 18))],
                [sg.Text("Product Name:"), sg.InputText(key='-PRODUCTNAME-')],
                [sg.Text("Price:"), sg.InputText(key='-PRICE-')],
                [sg.Text("Category ID:"), sg.InputText(key='-CATEGORYID-')],
                [sg.Button("Add Product", bind_return_key=True)]
            ])],
            [sg.Tab('Delete Product', [
                [sg.Text('Enter Product ID:'), sg.InputText(key='-PRODUCT_ID-')],
                [sg.Text('Enter Product Name:'), sg.InputText(key='-PRODUCT_NAME-')],
                [sg.Button('Delete')]
            ])],
        ])],
        [sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window('Products', layout)

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Back':
            window.hide()
        elif event == 'Add Product':
            window.hide()
            product_name = values['-PRODUCTNAME-']
            price = values['-PRICE-']
            category_id = values['-CATEGORYID-']
            try:
                cursor = db.cursor()
                insert_query = "INSERT INTO products (ProductName, Price, CategoryID) VALUES
(%s, %s, %s);"
                values = (product_name, price, category_id)
                cursor.execute(insert_query, values)
                db.commit()
                sg.popup(f"Task completed successfully!\n{product_name} was successfully
added as a product in the system!")
                staff_options_window()
            except mysql.connector.Error as err:
                sg.popup_error(f"Error occurred during product addition: {err}")
        elif event == 'Delete':
            product_id = values['-PRODUCT_ID-']
            product_name = values['-PRODUCT_NAME-']
            if product_id and product_name:
```

```
                try:
                    cursor = db.cursor()
                    cursor.execute('DELETE FROM products WHERE ProductID=%s AND
ProductName=%s;', (product_id, product_name))
                    db.commit()
                    sg.popup('Product Deleted Successfully!')
                except mysql.connector.Error as err:
                    sg.popup_error(f"Error: {err}")

    db.close()
    window.close()

# Function 12: To take inputs from the staff for the new category and then insert the values
into the table categories
def add_category_to_dbtab(db):
    layout=[
        [sg.TabGroup([
            [sg.Tab('Create Category', [
                [sg.Text("Add New Category", font=("Helvetica", 18))],
                [sg.Text("Category ID:"), sg.InputText(key='-CATEGORYID-')],
                [sg.Text("Category Name:"), sg.InputText(key='-CATEGORYNAME-')],
                [sg.Text("\nCategoryID should not be matching with any existing ones.")],
                [sg.Button("Add Category", bind_return_key=True)]
        ])],
        [sg.Tab('Delete Category', [
            [sg.Text('Enter Category ID:'), sg.InputText(key='-CATEGORY_ID-')],
            [sg.Text('Enter Category Name:'), sg.InputText(key='-CATEGORY_NAME-')],
            [sg.Text("\nNo Product should be existing in the deleting category.")],
            [sg.Button('Delete')]
        ])],
    ])],
    [sg.Exit(), sg.Button('Back')]
]
    window = sg.Window('Categories', layout)

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, 'Exit'):
            break
        elif event == 'Back':
            window.hide()
            staff_options_window()
        elif event == 'Add Category':
            window.hide()
            category_id = values['-CATEGORYID-']
            category_name = values['-CATEGORYNAME-']
            try:
```

```
                cursor = db.cursor()
                insert_query = "INSERT INTO categories (CategoryID, CategoryName) VALUES (%s,
%s);"
                values = (category_id, category_name)
                cursor.execute(insert_query, values)
                db.commit()
                sg.popup(f"Task completed successfully!\n{category_name} with
categoryid:{category_id} was  added as a product in the system!")
                staff_options_window()
            except mysql.connector.Error as err:
                sg.popup_error(f"Error occurred during category addition: {err}")
        elif event == 'Delete':
            category_id = values['-CATEGORY_ID-']
            category_name = values['-CATEGORY_NAME-']
            if category_id and category_name:
                try:
                    cursor = db.cursor()
                    cursor.execute('DELETE FROM categories WHERE CategoryID=%s AND
CategoryName=%s;', (category_id, category_name))
                    db.commit()
                    sg.popup('Category Deleted Successfully!')
                except mysql.connector.Error as err:
                    sg.popup_error(f"Error: {err}")

    db.close()
    window.close()

# Function 13: Create a customer registration window
def customer_registration_window():
    layout = [
        [sg.Text(("Enter full name:"),font=("helvetica",15),text_color="black"),
sg.Input(key="-NAME-", size=(40, 2))],
        [sg.Text(("Select gender:"),font=("helvetica",15),text_color="black"),
sg.Radio("Male", "GENDER", key="-MALE-", default=True), sg.Radio("Female", "GENDER", key="-
FEMALE-")],
        [sg.Text(("Enter your email address:"),font=("helvetica",15),text_color="black"),
sg.Input(key="-EMAIL-", size=(40, 2))],
        [sg.Text(("Enter your phone number:"),font=("helvetica",15),text_color="black"),
sg.Input(key="-PHONE-", size=(40, 2))],
        [sg.Text(("Enter your password"),text_color="black",font=("helvetica",15)),
sg.Input(key="-PASS-", password_char='*', size=(30,1),font=("helvetica",15))],
        [sg.Button("Register", bind_return_key=True,size=(20,2)), sg.Exit(size=(20,2)),
sg.Button('Back',size=(20,2))]
    ]
    window = sg.Window('Customer Registration', layout,size=(700,250))

    while True:
```

```
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Register":
            window.close()
            name = values["-NAME-"]
            gender = "Male" if values["-MALE-"] else "Female"
            email = values["-EMAIL-"]
            phone = values["-PHONE-"]
            password = values["-PASS-"]
            db = connect_to_database()
            if db:
                if insert_customer_data(db, name, gender, email, phone, password):
                    sg.popup(f"Welcome {name}! \n\nYour account has been successfully created
with the login details you have provided.\nPlease login to your account using your Phone
Number and Password. \n\n(You will be redirected to the main page when you click OK)")
                    customer_phone = phone
                    window.close()
                    main()
                else:
                    sg.popup_error("Error occurred during registration. Please try again
later.")
        elif event == 'Back':
            window.hide()
            main()

        db.close()
        window.close()

# Function 14: Insert customer data into the database
def insert_customer_data(db, name, gender, email, phone, password):
    try:
        cursor = db.cursor()
        insert_query = "INSERT INTO customers (Cust_Name, Cust_Gender, Cust_EmailAddress,
Cust_PhoneNumber, Cust_Password) VALUES (%s, %s, %s, %s, %s);"
        values = (name, gender, email, phone, password)
        cursor.execute(insert_query, values)
        db.commit()
        cursor.close()
        return True
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return False

# Function 15: Create a login window for customers
def customer_login_window(db):
    layout = [
```

```
        [sg.Text(("Phone Number:"),font=("helvetica",15),text_color="black"),
sg.InputText(key='-PHONE-')],
        [sg.Text(("Password:"),font=("helvetica",15),text_color="black"), sg.InputText(key='-
PASSWORD-', password_char='*')],
        [sg.Button("Login", bind_return_key=True,size=(20,1)), sg.Exit (size=(20,1)),
sg.Button('Back',size=(20,1))],
    ]
    window = sg.Window("Customer Login", layout, finalize=True)
    cursor = db.cursor()

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Login":
            window.hide()
            phone = values['-PHONE-']
            password = values['-PASSWORD-']
            cursor.execute("SELECT * FROM customers WHERE Cust_PhoneNumber=%s AND
Cust_Password=%s;", (phone, password))
            user = cursor.fetchone()
            if user:
                sg.popup("Login Successful", f"Welcome, {user[1]}. Have a lovely day :D
\n\n{f}")
                window.close()
                customer_order_window(db)
            else:
                sg.popup_error("Login Failed", "Invalid username or password")
        elif event == 'Back':
            window.hide()
            main()

    db.close()
    window.close()

# Function 16: Create a customer order window
def customer_order_window(db):
    layout = [
        [sg.Text("Select what you want to do:")],
        [sg.Button("Create New Order"), sg.Button("See Existing Orders"), sg.Exit(),
sg.Button('Logout')]
    ]
    window = sg.Window('Customer Options', layout)

    while True:
        event, values = window.read()
        if event in ('See Existing Orders','Create New Order','Logout'):
```

```
            window.hide()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Create New Order":
            window.close()
            create_order_window(db)
        elif event == "See Existing Orders":
            display_customer_orders(db)
        elif event == 'Logout':
            customer_login_window(db)

    db.close()
    window.close()

# Function 17: Create a window for order creation
def create_order_window(db):
    try:
        categories = []
        cursor = db.cursor()
        cursor.execute("SELECT DISTINCT CategoryName FROM categories;")
        for row in cursor.fetchall():
            categories.append(row[0])
    except mysql.connector.Error as err:
        print(f"Error: {err}")
    layout = [
        [sg.Text("Select Product Categories:")],
        [sg.Listbox(categories, key="-CATEGORIES-",
select_mode=sg.LISTBOX_SELECT_MODE_MULTIPLE, size=(40, 6), enable_events=True)],
        [sg.Text("Select Products:")],
        [sg.Listbox(values=[], key="-PRODUCTS-", select_mode=sg.LISTBOX_SELECT_MODE_MULTIPLE,
size=(40, 6))],
        [sg.Button("Confirm Selection", bind_return_key=True), sg.Exit(), sg.Button('Back')]
    ]
    window = sg.Window("Create New Order", layout)
    selected_product_names = []

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, "Exit",):
            break
        elif event == "-CATEGORIES-":
            selected_categories = values["-CATEGORIES-"]
            selected_products = []
            for category in selected_categories:
                selected_products.extend(fetch_products_by_category(db, category))
            window["-PRODUCTS-"].update(values=selected_products)
        elif event == "Confirm Selection":
```

```
            window.hide()
            selected_category = values["-CATEGORIES-"]
            selected_products = values["-PRODUCTS-"]
            if selected_products and selected_category:
                selected_product_names.extend(selected_products)
                window["-SELECTED-PRODUCTS-"].update(values=selected_product_names)
                product_info = fetch_product_info_by_names(db, selected_product_names)
                billing_window = display_billing_window(product_info)

                while True:
                    event, _ = billing_window.read()
                    if event == sg.WIN_CLOSED or event == "Exit":
                        break
                window.close()
                break
        if event == 'Back':
            window.close()
            customer_order_window(db)

    db.close()
    window.close()

# Function 18: Fetch products by category from the database
def fetch_products_by_category(db, category):
    try:
        products = []
        cursor = db.cursor()
        cursor.execute("SELECT ProductName FROM products WHERE CategoryID = (SELECT
CategoryID FROM categories WHERE CategoryName = %s);", (category,))
        for row in cursor.fetchall():
            products.append(row[0])
        return products
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return []
    db.close()

# Function 19: To fetch info of only the products in the list selected_product_names
def fetch_product_info_by_names(db, selected_product_names):
    try:
        cursor = db.cursor(dictionary=True)
        query = f"""
            SELECT c.CategoryName, p.ProductName, p.Price
            FROM products p
            JOIN categories c ON p.CategoryID = c.CategoryID
            WHERE p.ProductName IN ({', '.join(['%s' for _ in selected_product_names])});
        """
```

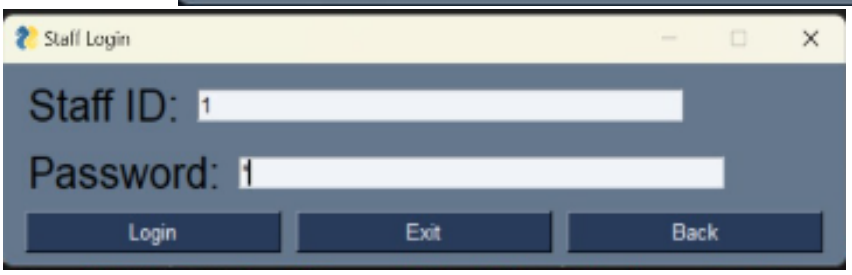
```
        cursor.execute(query, selected_product_names)
        product_info = cursor.fetchall()
        return product_info
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return []

# Function 20: To create the billing window
def display_billing_window(product_details):
    layout = [
    [sg.Text("Billing Details", font=("Helvetica", 18))],
    [sg.Table(values=[], headings=["Sr. No.", "Product Name", "Category Name", "Product
Price"], auto_size_columns=False,
              col_widths=[5, 20, 20, 15], justification="center", num_rows=10, key='-TABLE-
')],
    [sg.Text("Total Amount:", font=("Helvetica", 14)), sg.Text("", size=(10, 1),
font=("Helvetica", 14), key='-TOTAL-', justification="center")],
    [sg.Text("(Your order can not be edited once you select Confirm Order)")],
    [sg.Button("Confirm Order", bind_return_key=True), sg.Exit(), sg.Button('Change your
Order')]
    ]
    window = sg.Window('Billing Window', layout, finalize=True)

    serial_number = 1
    total_amount = 0.0
    table_data = []
    for order in product_details:
        category_name = order["CategoryName"]
        product_name = order["ProductName"]
        product_price = float(order["Price"])
        total_amount += product_price
        table_data.append([serial_number, product_name, category_name,
f"Rs.{product_price:.2f}"])
        serial_number += 1
    window['-TABLE-'].update(values=table_data)
    window['-TOTAL-'].update(f"Rs.{total_amount:.2f}")

    while True:
        event, values = window.read()
        if event in (sg.WIN_CLOSED, "Exit"):
            break
        elif event == "Confirm Order":
            window.hide()
            get_customer_details_and_put_it_in_orders(connect_to_database())
            window.close()
        elif event == 'Change your Order':
            window.hide()
```

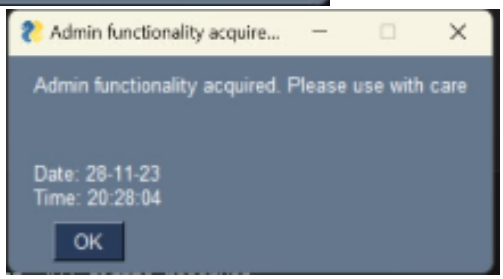
```
            create_order_window(connect_to_database())

    db.close()
    window.close()
    return window

# Function 21: To take inputs from the user about their order and inserts it into table
orders
def get_customer_details_and_put_it_in_orders(db):
    layout = [
        [sg.Text("Enter your contact details:")],
        [sg.Text("Billing Phone Number:"), sg.InputText(key="-PHONE-")],
        [sg.Text("Address:"), sg.InputText(key="-ADDRESS-")],
        [sg.Text("Extra Notes:"), sg.InputText(key="-EXTRA-NOTES-")],
        [sg.Button("Confirm", bind_return_key=True)]
    ]
    window = sg.Window("Customer Details", layout, finalize=True)

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED:
            break
        elif event == "Confirm":
            window.hide()
            phone_number = values["-PHONE-"]
            address = values["-ADDRESS-"]
            extra_notes = values["-EXTRA-NOTES-"]
            try:
                cursor = db.cursor()
                insert_query = "INSERT INTO orders (Cust_PhoneNumber, Cust_HomeAddress, Note)
VALUES (%s, %s, %s);"
                values = (phone_number, address, extra_notes)
                cursor.execute(insert_query, values)
                db.commit()
                cursor.close()
                sg.popup(f"Order successfully created! \nWill be delivered at
{address}.\n\n{f}")
                customer_order_window(db)
            except mysql.connector.Error as err:
                print(f"Error: {err}")
        elif event == 'Cancel':
            window.hide()
            customer_order_window(db)
            window.close()
            break

    db.close()
```

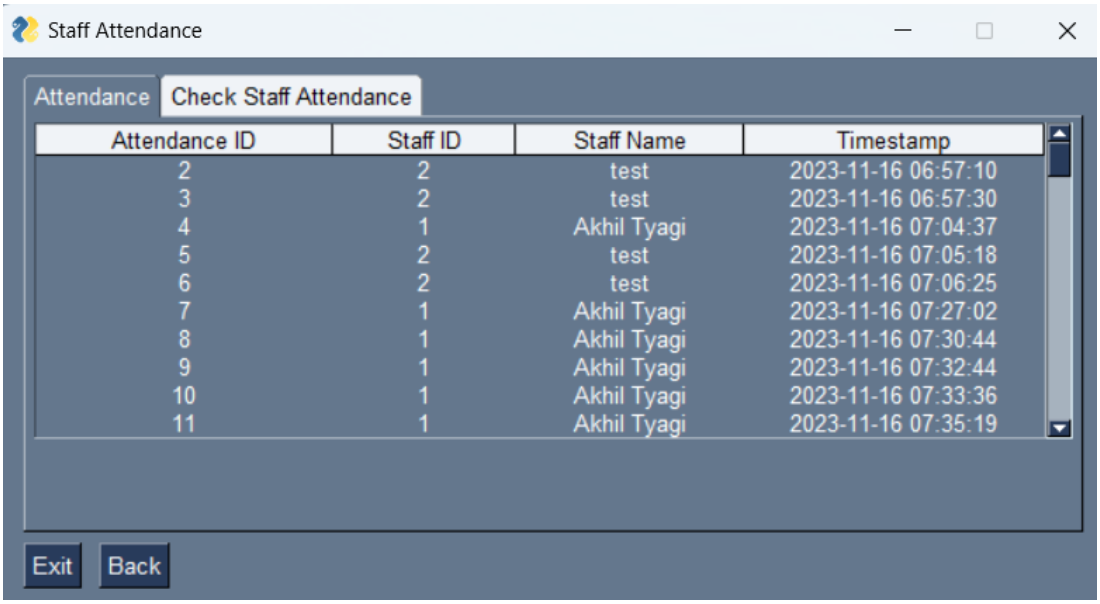
```
    window.close()

# Function 22: To display the customers order info if they select see existing orders, in a window ask for their phone number and then retrieve data from database
def display_customer_orders(db):
    layout = [
        [sg.Text("Enter Billing Phone Number:"), sg.InputText(key='-PHONE-')],
        [sg.Button("Show Orders"), sg.Exit(), sg.Button('Back')],
        [sg.Table(values=[], headings=["Order ID", "Address", "Notes", "Order Time", "Status"], auto_size_columns=False,
                  col_widths=[10, 25, 20, 15, 13], display_row_numbers=False, justification="center ", key='-TABLE-', row_height=35)]
    ]
    window = sg.Window("Your Existing Orders", layout, finalize=True,)

    while True:
        event, values = window.read()
        if event == sg.WIN_CLOSED or event == "Exit":
            break
        elif event == "Show Orders":
            phone_number = values["-PHONE-"]
            if phone_number:
                try:
                    cursor = db.cursor(dictionary=True)
                    cursor.execute("SELECT * FROM orders WHERE Cust_PhoneNumber = %s;", (phone_number,))
                    orders = cursor.fetchall()
                    if orders:
                        table_data = []
                        for order in orders:
                            table_data.append([order["OrderID"], order["Cust_HomeAddress"], order["Note"], order["Time"], order["Delivery_Status"]])
                        window['-TABLE-'].update(values=table_data)
                    else:
                        sg.popup("No orders found for the given phone number.")
                except mysql.connector.Error as err:
                    print(f"Error: {err}")
            else:
                sg.popup("Please enter a valid phone number.")
        elif event == 'Back':
            window.hide()
            customer_order_window(db)

    db.close()
    window.close()

# Function 23: Main function
```

```
def main():
    sg.theme('BrownBlue')
    layout_choice = [
        [sg.Image(filename="meow.png")],
        [sg.Text("Welcome to da cool store!", font=('Arial', 30),text_color="black")],
        [sg.Button("Staff",size=(20,3)), sg.Button("New User",size=(20,3)),
sg.Button("Existing User",size=(20,3)), sg.Exit(size=(20,3))]
    ]
    window_choice = sg.Window('Da Cool Store', layout_choice, element_justification="c",
size=(800, 450))

    while True:
        event_choice, _ = window_choice.read()
        if event_choice in ('Staff','New User','Existing User'):
            window_choice.close()
        if event_choice == sg.WIN_CLOSED or event_choice == "Exit":
            break
        elif event_choice == "Staff":
            staff_login_window(connect_to_database())
        elif event_choice == "New User":
            window_choice.close()
            customer_registration_window()
        elif event_choice == "Existing User":
            window_choice.close()
            customer_login_window(connect_to_database())

    window_choice.close()
```

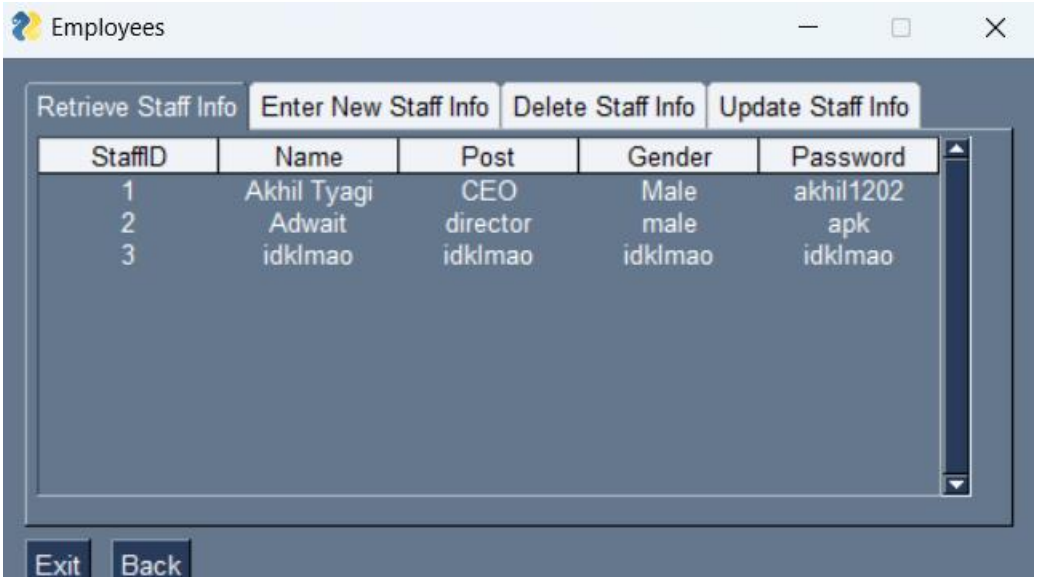
OUTPUT SCREENSHOTS

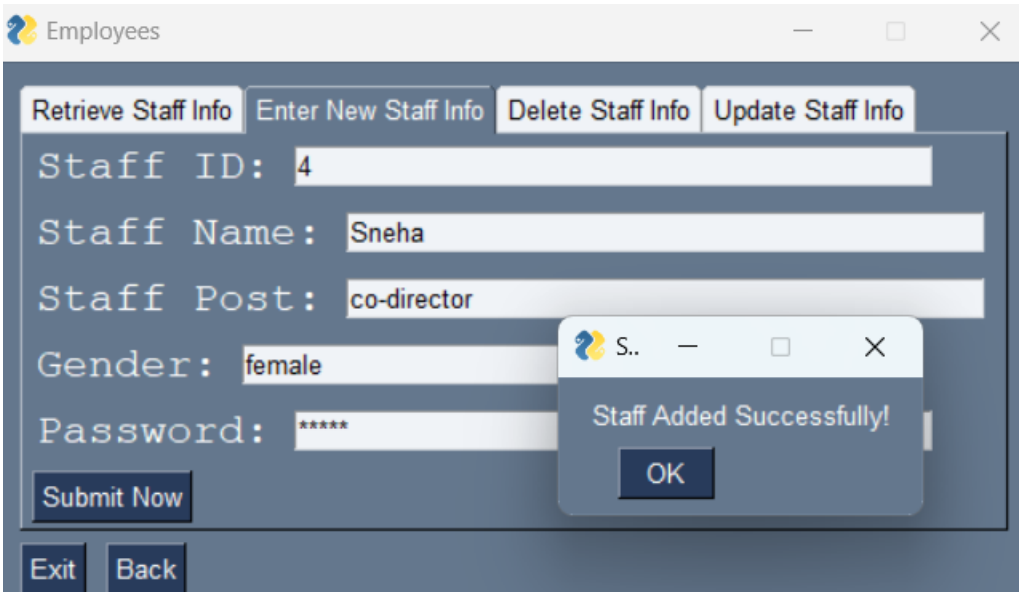


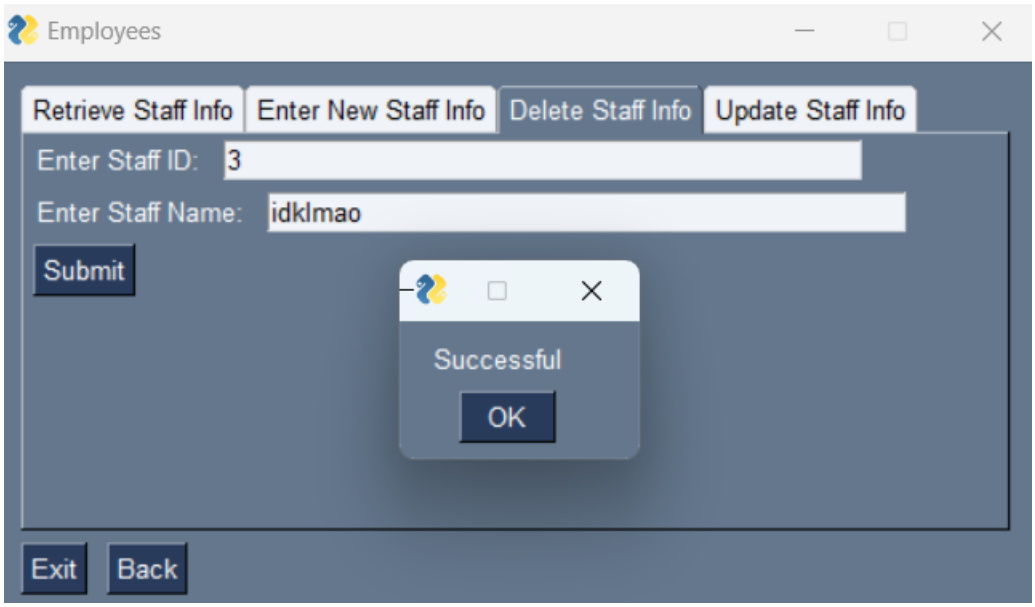


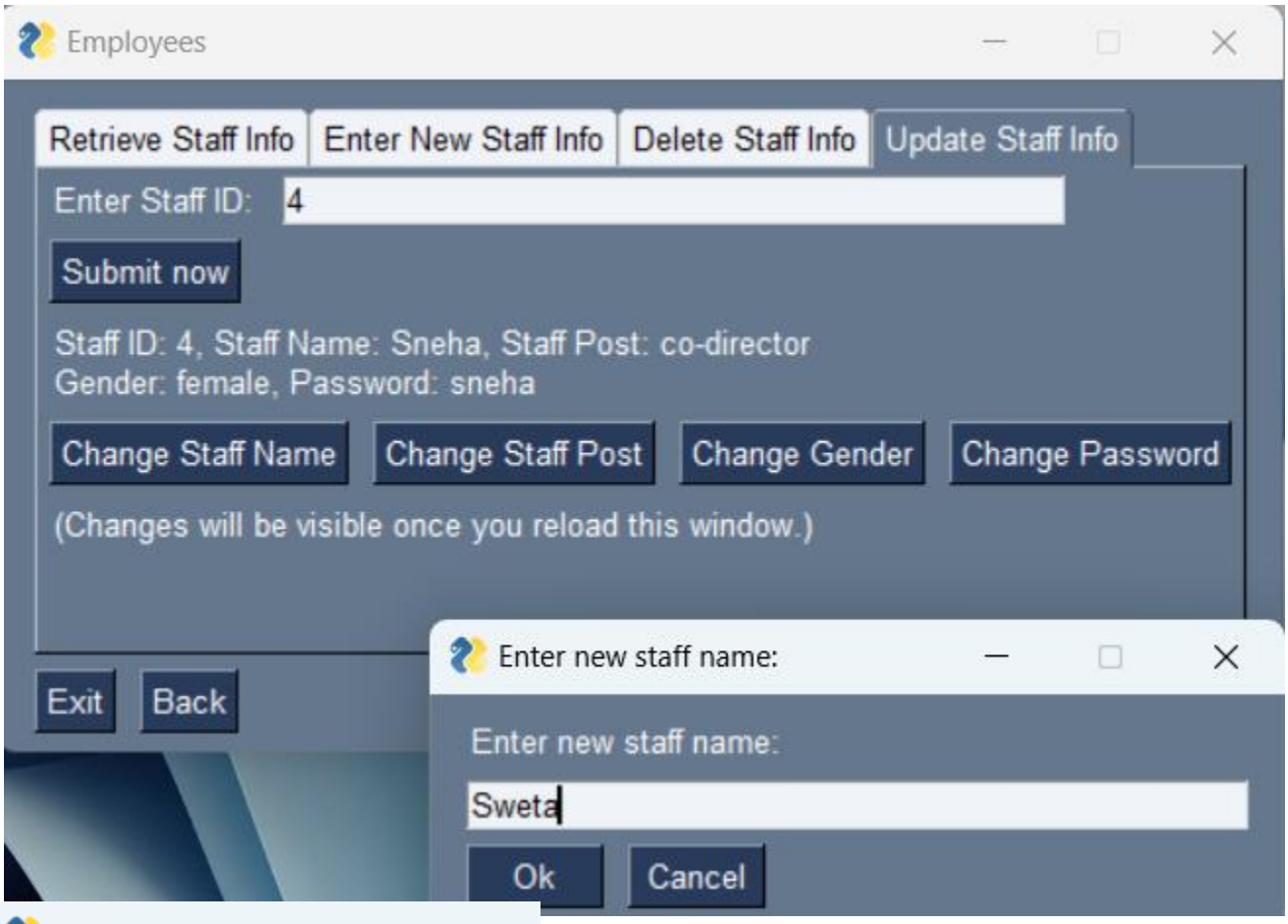

Staff Attendance
Attendance
Check Staff Attendance
Attendance ID
Staff ID
Staff Name
Timestamp
2 2 test 2023-11-16 06:57:10
3 2 test 2023-11-16 06:57:30
4 1 Akhil Tyagi 2023-11-16 07:04:37
5 2 test 2023-11-16 07:05:18
6 2 test 2023-11-16 07:06:25
7 1 Akhil Tyagi 2023-11-16 07:27:02
8 1 Akhil Tyagi 2023-11-16 07:30:44
9 1 Akhil Tyagi 2023-11-16 07:32:44
10 1 Akhil Tyagi 2023-11-16 07:33:36
11 1 Akhil Tyagi 2023-11-16 07:35:19
Exit
Back

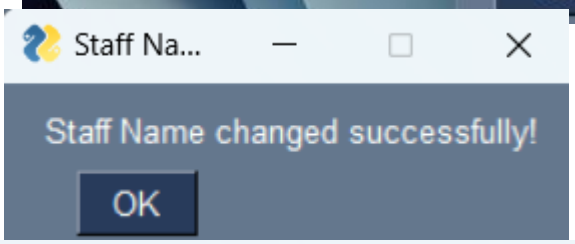

Staff Attendance
Attendance
Check Staff Attendance
Enter Staff ID:
1
Fetch Timestamps
2023-11-16 07:04:37
2023-11-16 07:27:02
2023-11-16 07:30:44
2023-11-16 07:32:44
2023-11-16 07:33:36
2023-11-16 07:35:19
2023-11-16 07:38:30
2023-11-16 07:38:54
2023-11-16 07:39:23
2023-11-16 07:43:08
Exit
Back

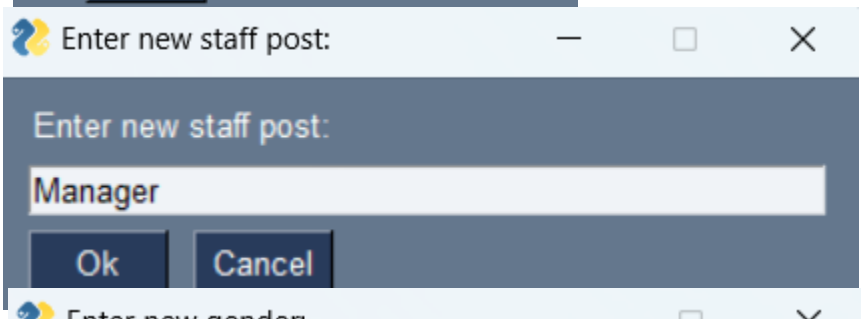

Employees
Retrieve Staff Info
Enter New Staff Info
Delete Staff Info
Update Staff Info
StaffID
Name
Post
Gender
Password
1 Akhil Tyagi CEO Male akhil1202
2 Adwait director male apk
3 idklmao idklmao idklmao idklmao
Exit
Back

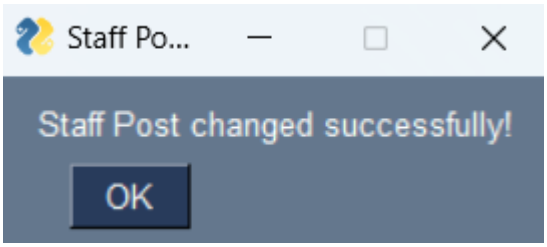

Employees
Retrieve Staff Info
Enter New Staff Info
Delete Staff Info
Update Staff Info
Staff ID: 4
Staff Name: Sneha
Staff Post: co-director
Gender: female
Password: *****
Submit Now
Exit
Back
S..
Staff Added Successfully!
OK

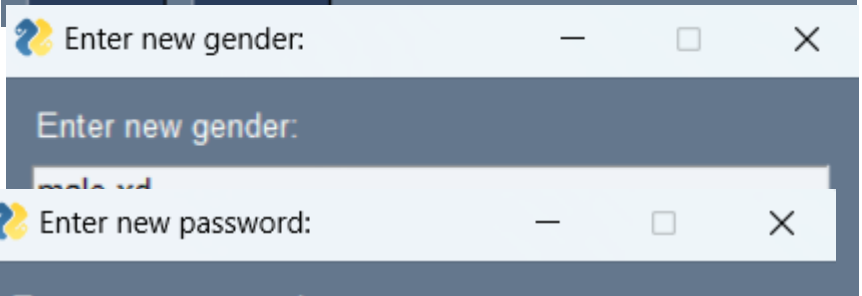

Employees
Retrieve Staff Info
Enter New Staff Info
Delete Staff Info
Update Staff Info
Enter Staff ID: 3
Enter Staff Name: idklmao
Submit
Successful
OK
Exit
Back

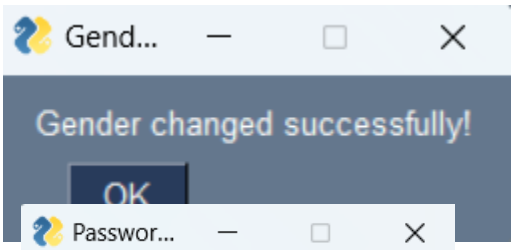


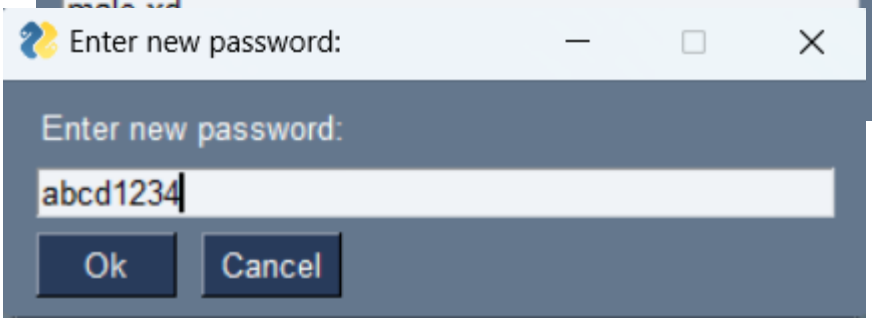


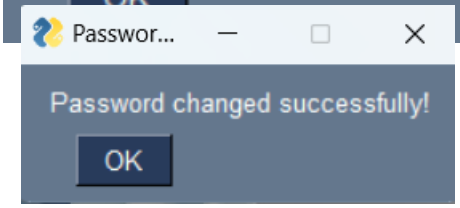




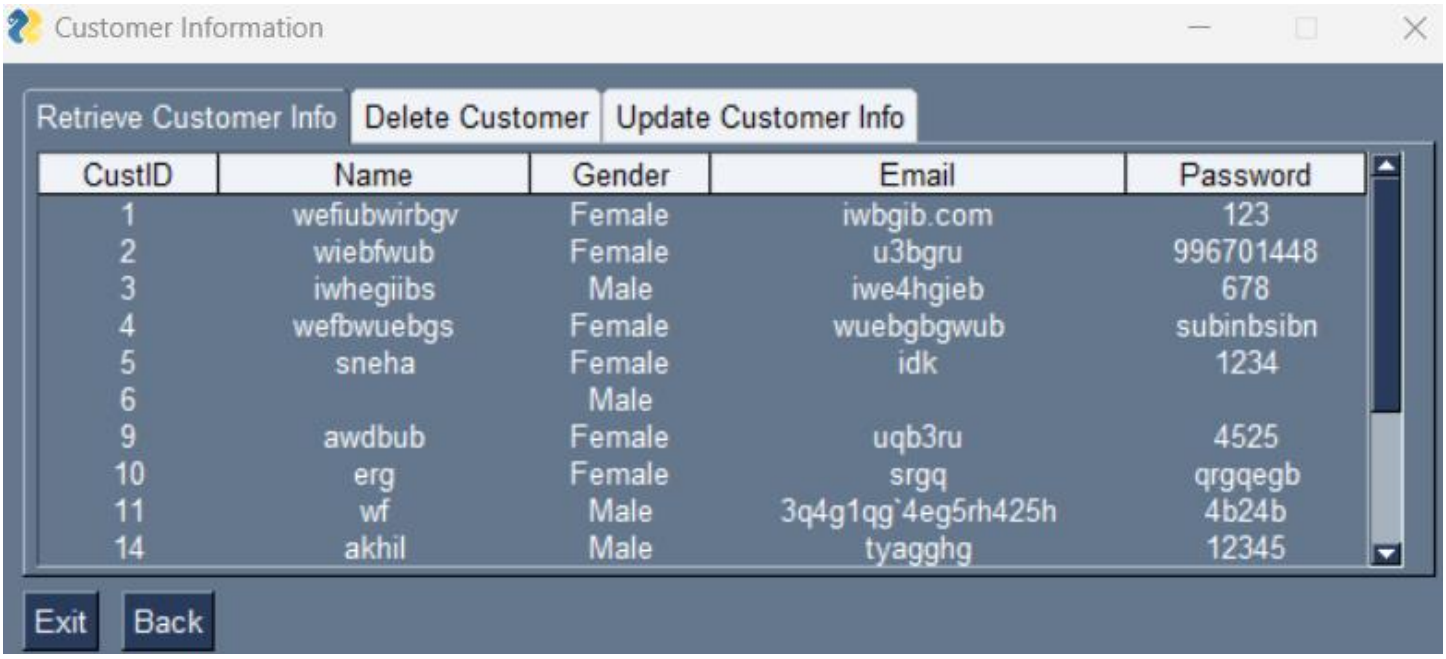


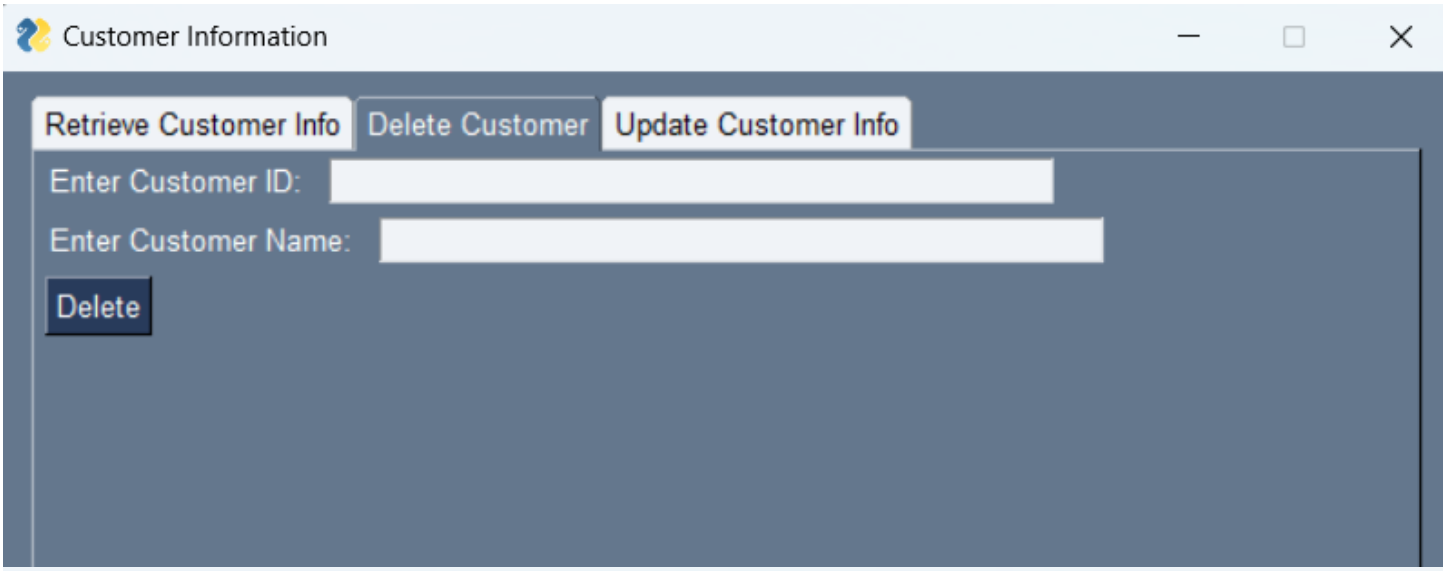


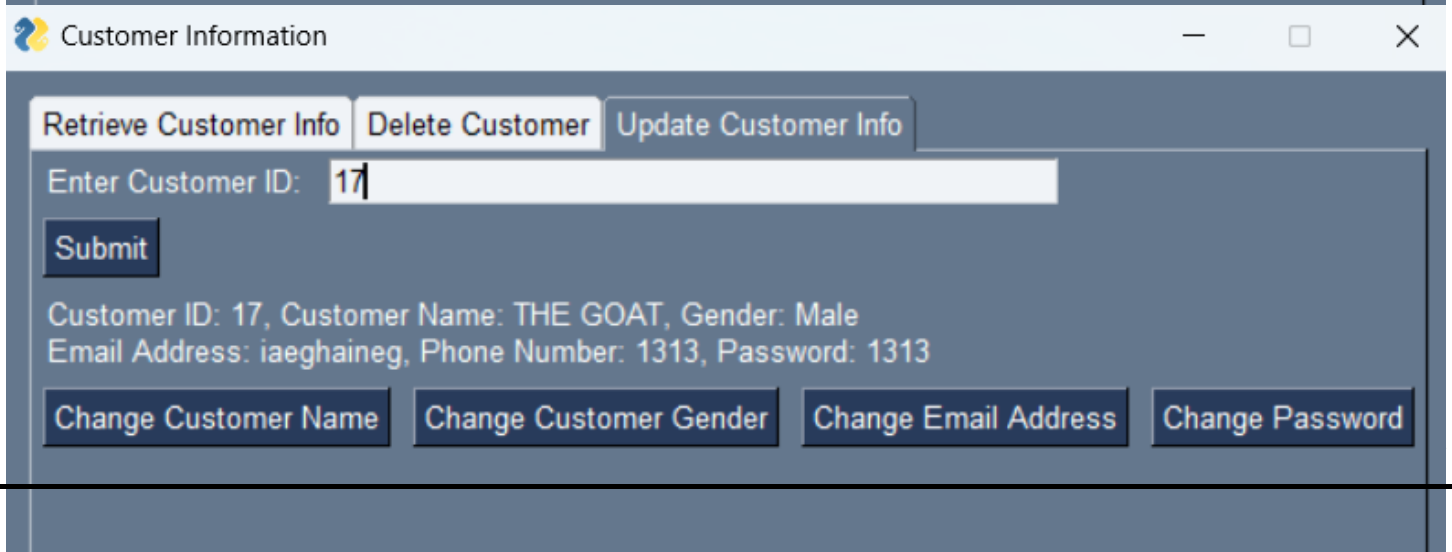


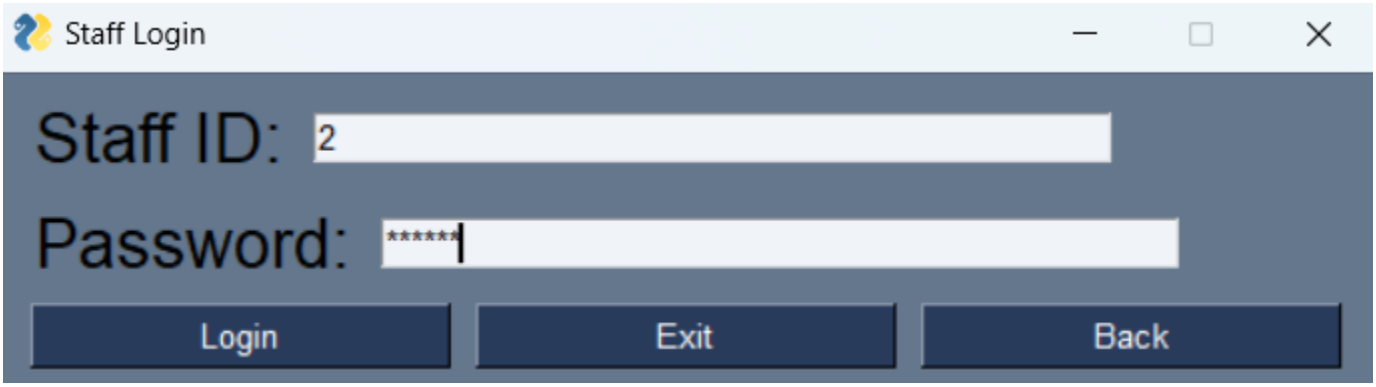


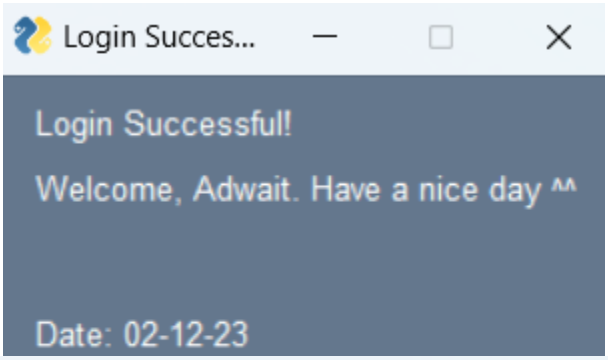
Akhil Tyagi XII-A

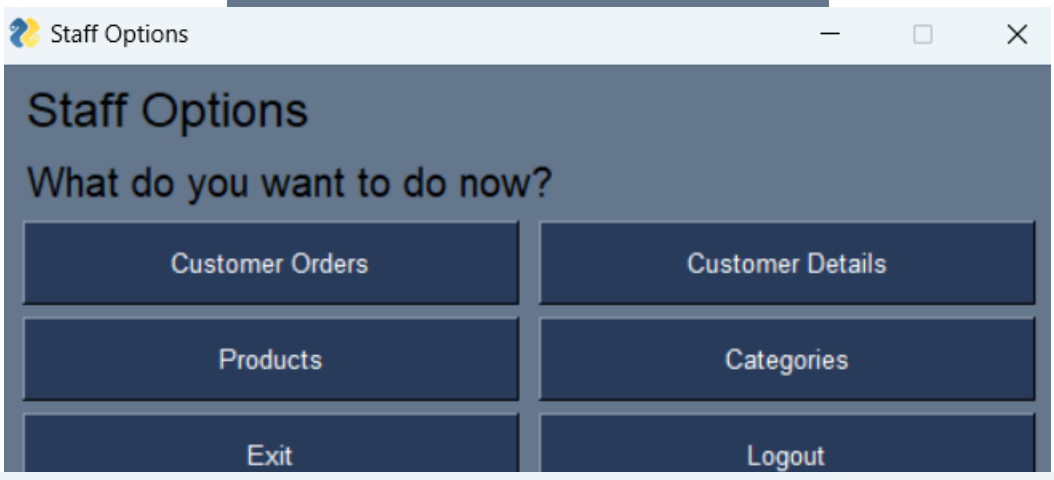

Customer Information
Retrieve Customer Info
Delete Customer
Update Customer Info
CustID Name Gender Email Password
1 wefiubwirbgv Female iwbgib.com 123
2 wiebfwub Female u3bgru 996701448
3 iwhegiibs Male iwe4hgieb 678
4 wefbwuebgs Female wuebgbgwub subinbsibn
5 sneha Female idk 1234
6 Male
9 awdbub Female uqb3ru 4525
10 erg Female srgq qrgqegb
11 wf Male 3q4g1qg`4eg5rh425h 4b24b
14 akhil Male tyagghg 12345
Exit
Back

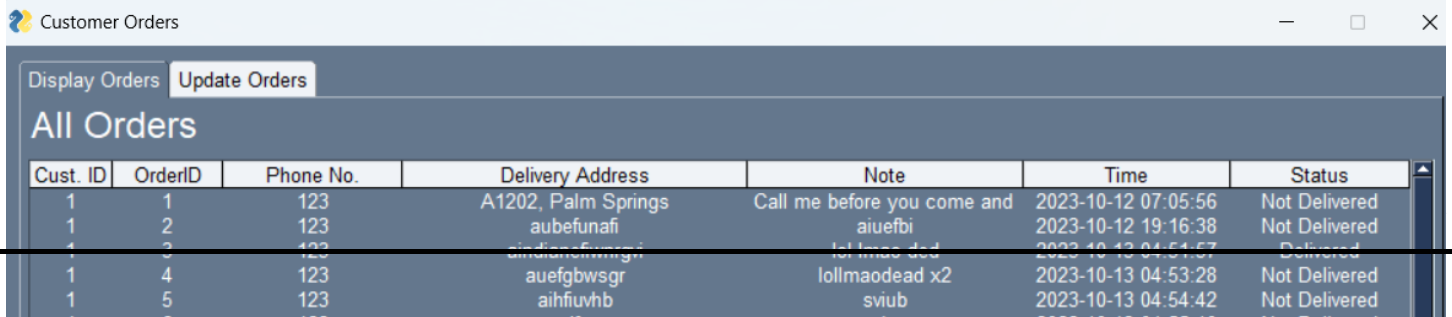

Customer Information
Retrieve Customer Info
Delete Customer
Update Customer Info
Enter Customer ID:
Enter Customer Name:
Delete

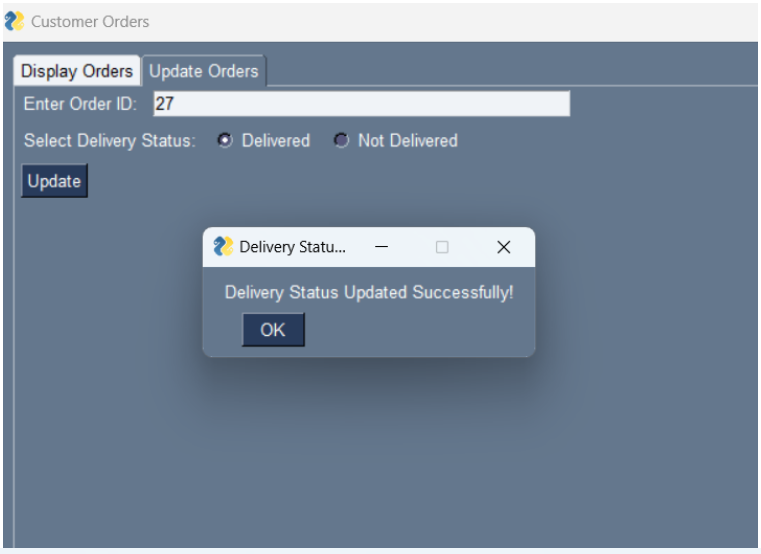

Customer Information
Retrieve Customer Info
Delete Customer
Update Customer Info
Enter Customer ID: 17
Submit
Customer ID: 17, Customer Name: THE GOAT, Gender: Male
Email Address: iaeghaineg, Phone Number: 1313, Password: 1313
Change Customer Name
Change Customer Gender
Change Email Address
Change Password

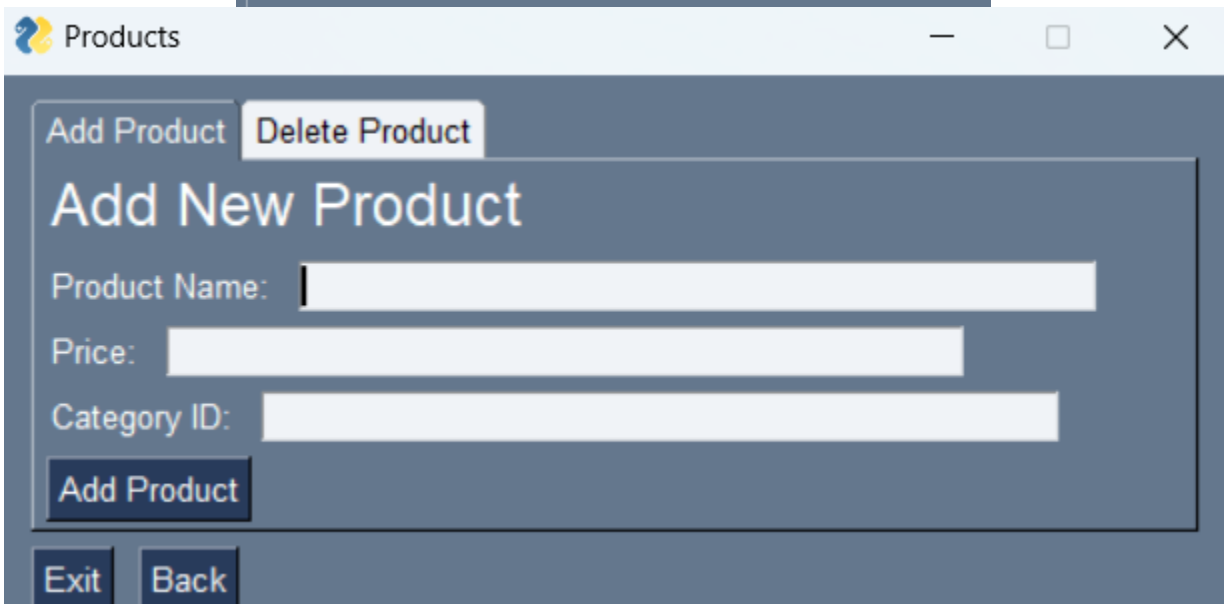

Staff Login
Staff ID:
2
Password:
******
Login
Exit
Back

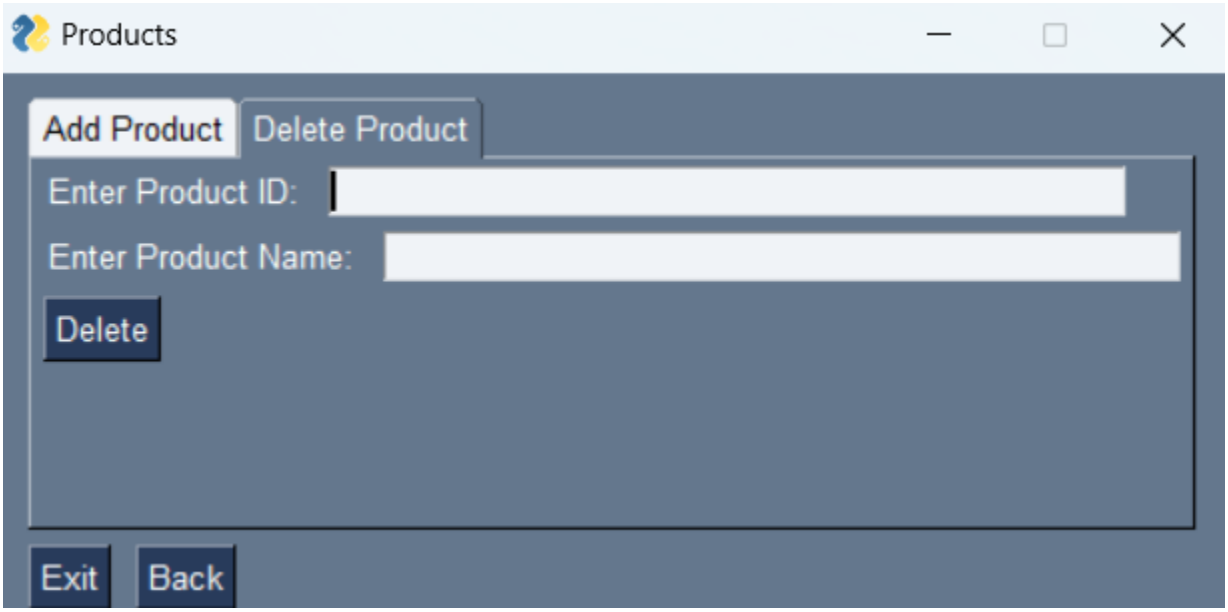

Login Succes...
Login Successful!
Welcome, Adwait. Have a nice day ^^
Date: 02-12-23

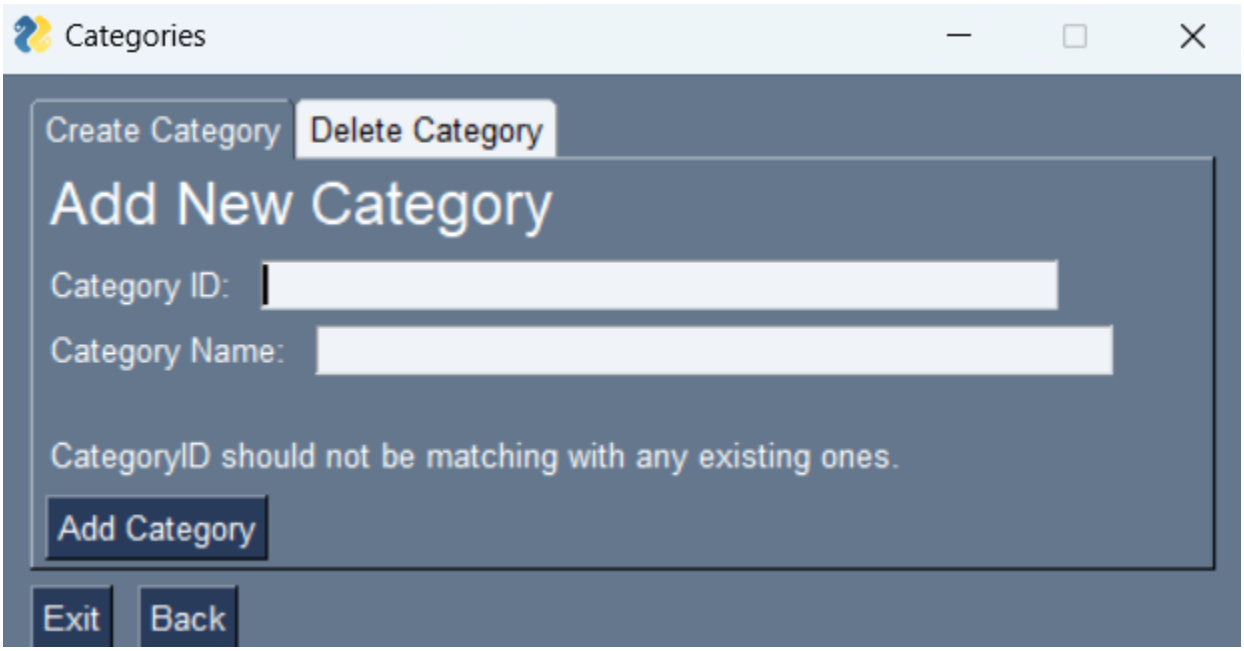

Staff Options
Staff Options
What do you want to do now?
Customer Orders
Customer Details
Products
Categories
Exit
Logout

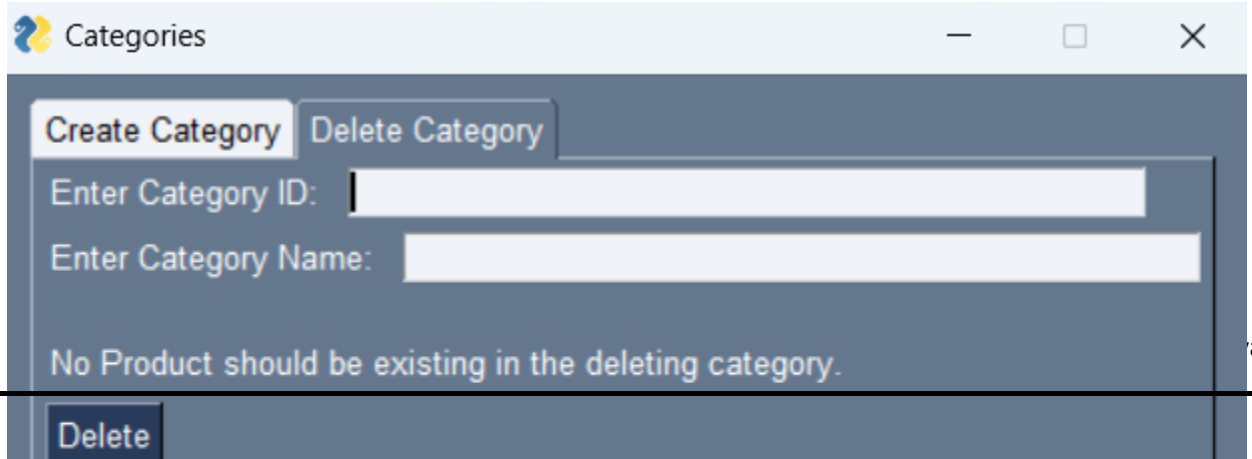

Customer Orders
Display Orders
Update Orders
All Orders
Cust. ID
OrderID
Phone No.
Delivery Address
Note
Time
Status
1 1 123 A1202, Palm Springs Call me before you come and 2023-10-12 07:05:56 Not Delivered
1 2 123 aubefunafi aiuefbi 2023-10-12 19:16:38 Not Delivered
1 4 123 auefgbwsgr lollmaodead x2 2023-10-13 04:53:28 Not Delivered
1 5 123 aihfiuvhb sviub 2023-10-13 04:54:42 Not Delivered

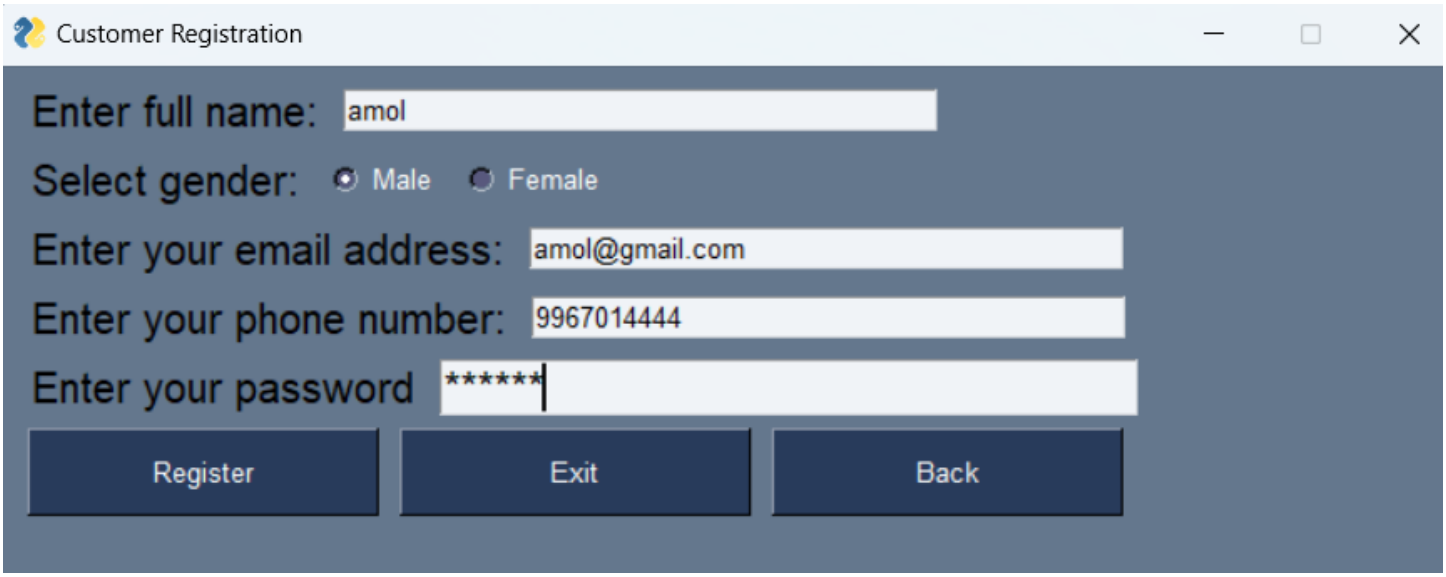

Customer Orders
Display Orders
Update Orders
Enter Order ID:
27
Select Delivery Status:
Delivered
Not Delivered
Update
Delivery Statu...
Delivery Status Updated Successfully!
OK

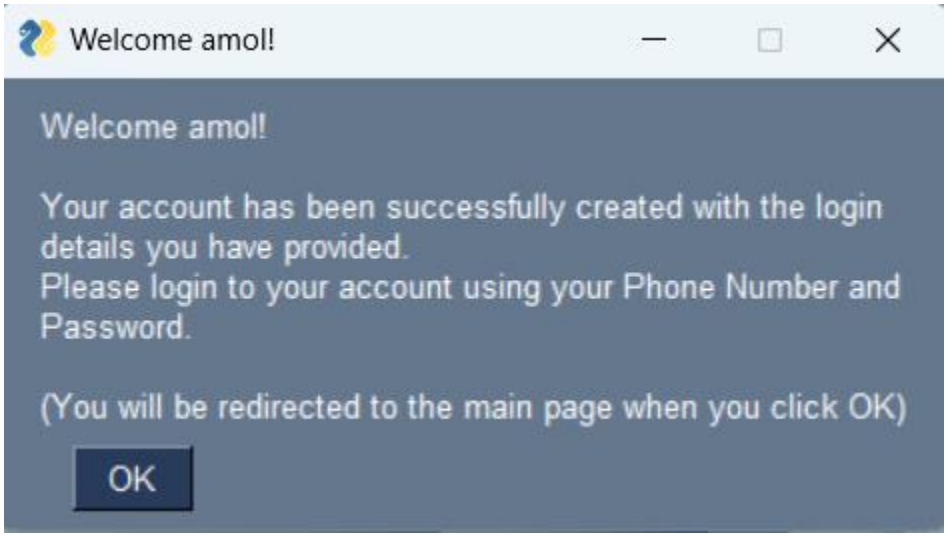

Products
Add Product
Delete Product
Add New Product
Product Name:
Price:
Category ID:
Add Product
Exit
Back

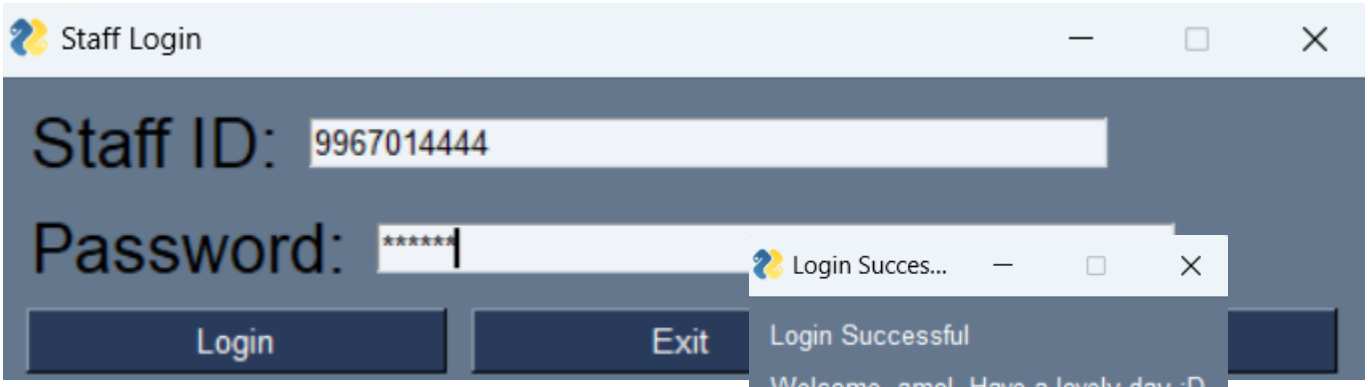

Products
Add Product
Delete Product
Enter Product ID:
Enter Product Name:
Delete
Exit
Back

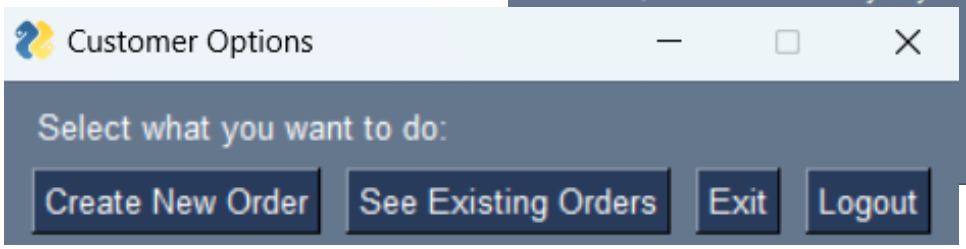

Categories
Create Category
Delete Category
Add New Category
Category ID:
Category Name:
CategoryID should not be matching with any existing ones.
Add Category
Exit
Back

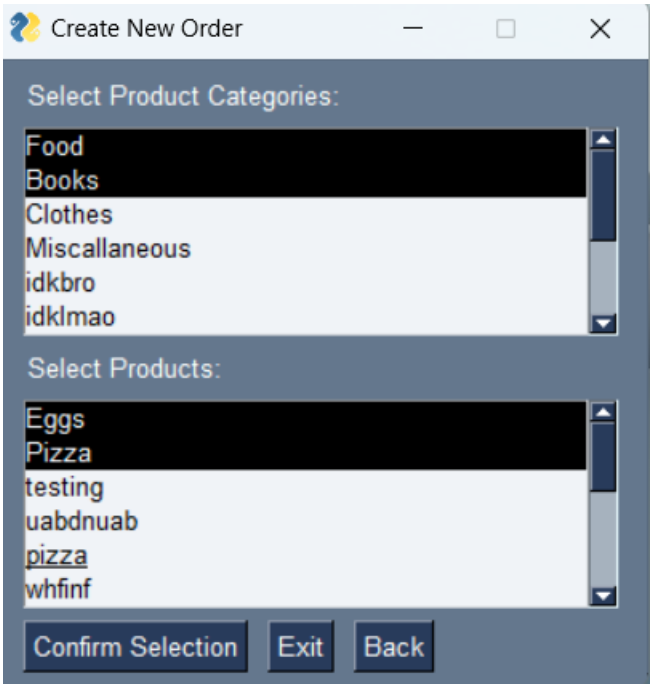

Categories
Create Category
Delete Category
Enter Category ID:
Enter Category Name:
No Product should be existing in the deleting category.
Delete

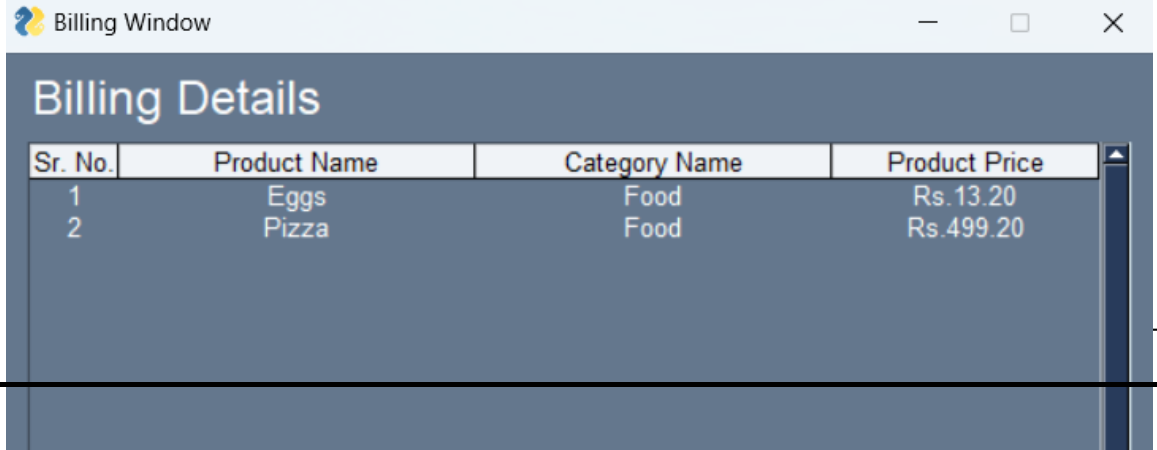

Customer Registration
Enter full name:
amol
Select gender:
Male
Female
Enter your email address:
amol@gmail.com
Enter your phone number:
9967014444
Enter your password
******
Register
Exit
Back


Welcome amol!
Welcome amol!
Your account has been successfully created with the login details you have provided.
Please login to your account using your Phone Number and Password.
(You will be redirected to the main page when you click OK)
OK

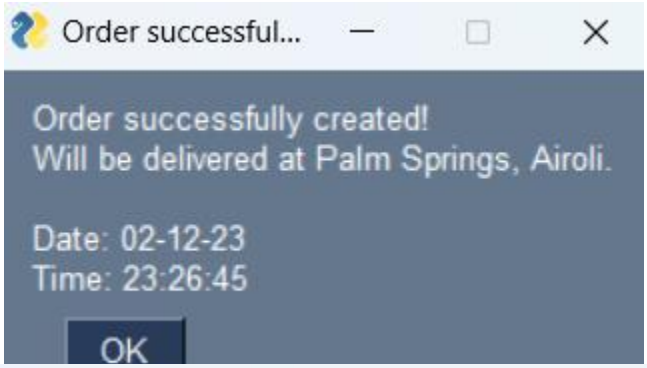

Staff Login
Staff ID:
9967014444
Password:
******
Login
Exit
Login Succes...
Login Successful
Welcome, amol. Have a lovely day :D

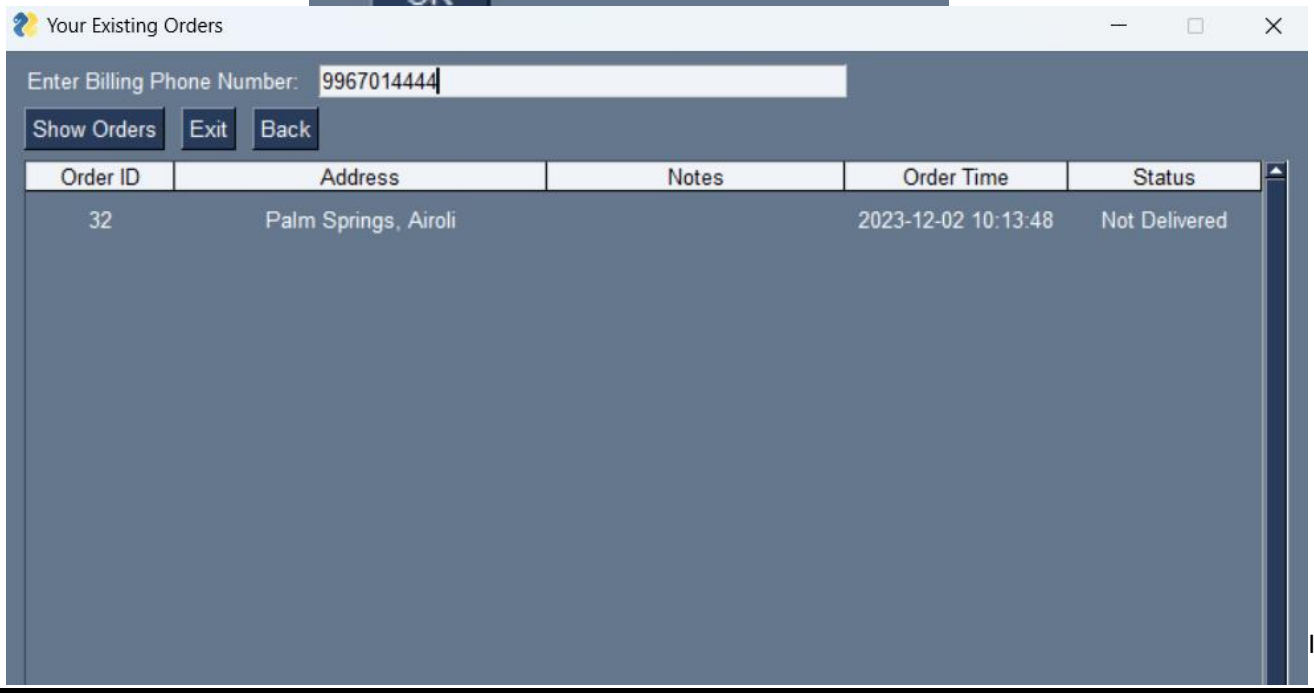

Customer Options
Select what you want to do:
Create New Order
See Existing Orders
Exit
Logout


Create New Order
Select Product Categories:
Food
Books
Clothes
Miscallaneous
idkbro
idklmao
Select Products:
Eggs
Pizza
testing
uabdnuab
pizza
whfinf
Confirm Selection
Exit
Back


Billing Window
Billing Details
Sr. No.
Product Name
Category Name
Product Price
1
Eggs
Food
Rs.13.20
2
Pizza
Food
Rs.499.20


Customer Details
Enter your contact details:
Billing Phone Number:
9967014444
Address:
Palm Springs, Airoli
Extra Notes:
Confirm


Order successful...
Order successfully created!
Will be delivered at Palm Springs, Airoli.
Date: 02-12-23
Time: 23:26:45
OK


Your Existing Orders
Enter Billing Phone Number:
9967014444
Show Orders
Exit
Back
Order ID
Address
Notes
Order Time
Status
32
Palm Springs, Airoli
2023-12-02 10:13:48
Not Delivered

FUTURE-SCOPE

- One of the key future scopes of a project on grocery store is the development of an integrated system for managing various aspects of the store, such as inventory, sales, and customer data. This system could include features such as real-time inventory tracking, automated sales reporting, and customer relationship management tools. By leveraging the capabilities of Python and PySimpleGUI, we can create a seamless and efficient management system that streamlines the operations of the store and improves overall productivity.

- Furthermore, the project can also incorporate features such as loyalty programs, promotional offers, and customer engagement tools, which can help in building a loyal customer base and driving more revenue for the store. With

the integration of online ordering and delivery services, the project can also offer convenience and accessibility to the customers, catering to the growing demand for e-commerce solutions in the grocery retail industry.

- In addition to these core functionalities, the future scope of the project can also encompass the integration of advanced analytics and reporting capabilities. By harnessing Python's data processing libraries and PySimpleGUI's visualization tools, the project can generate insights into sales trends, customer behavior, and operational performance. This can empower grocery store owners to make data-driven decisions, identify growth opportunities, and stay ahead of the competition.

- In conclusion, the future scope of a grocery store project made using Python and PySimpleGUI is immense. By leveraging the capabilities of these technologies, the project can offer a comprehensive solution for grocery store owners to streamline their operations, improve customer satisfaction, and drive business growth.

BIBLIOGRAPHY

- **geeksforgeeks.org**
- **TheCSClassroom**
- **Pysimplegui Docs**

Thank You.