UNIVERSITI TEKNOLOGI MARA,

MERBOK, KEDAH

SCHOOL OF INFORMATION SCIENCE,
COLLEGE OF COMPUTING, INFORMATICS, AND MATHEMATICS

PROGRAMMING FOR LIBRARIES

(IML208)

ASSIGNMENT II: GROUP PROJECT (**POTLEPAK RESTAURANT**)

PREPARED BY:

NAME	STUDENT ID
NUR HIDAYATI BINTI JAKARIA	2022843754
NURAMNI NADHIRAH BINTI MOHD NASIR	2022873844
NUR AYUNI BINTI GHAZALI	2022897352
WAN ALIAH FARISYA BINTI WAN NORUL AZAN	2022865804

CLASS: KCDIM1443E

PREPARED FOR:

SIR AIRUL SHAZWAN BIN NORSHAHIMI

SUBMISSION DATE:

16th JANUARY 2024

“POTLEPAK RESTAURANT”

NUR AYUNI BINTI GHAZALI
(2022897352)

NUR HIDAYATI BINTI JAKARIA
(2022843754)

NURAMNI NADHIRAH BINTI MOHD NASIR
(2022873844)

WAN ALIAH FARISYA BINTI WAN NORUL AZAN
(2022865804)

SCHOOL OF INFORMATION SCIENCE,
COLLEGE OF COMPUTING, INFORMATICS, AND MATHEMATICS,
UNIVERSITI TEKNOLOGI MARA,

MERBOK, KEDAH

16th JANUARY 2024

ACKNOWLEDGEMENT

Assalammualaikum w.b.t

First, we would like to praise ALLAH S.W.T. for giving us this opportunity to make this assignment go smoothly. Without His blessing, we could not succeed in solving this task.

Secondly, we would like to show our appreciation to our lecturer, Sir Airul Shazwan Bin Norshahimi for all the guidance and knowledge he shared with us throughout the process while doing this assignment. Without his guidance, we couldn't manage to complete this task. We also want to say thank you to him for teaching us in this course.

Finally, we would like to say thank you to our precious group members for giving cooperation while doing this assignment. Other than that, we like to give this appreciation to our family and friends because never give up giving us encouragement and prayers that have kept us going till now. We hope by doing this assignment, we can use the information for something beneficial to us in the future.

TABLE OF CONTENT

1.0 INTRODUCTION

In a time of rapid technological development and transformation, our group noticed that there is still a certain organization such as restaurants do not have upgraded technology in their systems. We're here today not just to solve problems, but also to start a strategic project that will lead to a future where productivity, connectivity, and unmatched user experiences are the norm. Especially in restaurant organization systems that leak in many aspects of technology. We confront the current reality of our operational landscape and highlight the crucial areas where using traditional methods has proven to be problematic. This difficulty, which was formerly perceived as a barrier, is now a chance for change.

As we unfold this narrative, imagine a narrative where manual processes give way to seamless digital experiences, where data becomes an asset rather than a liability, and where every interaction, from customer registration to employee management and table reservations, is characterized by efficiency and accuracy. This assignment serves as our compass, guiding us through the ins and outs of change and setting us on a path to a future where our organization not only adapts to the demands of the modern world but leads the way in setting new standards of excellence.

Moving into the specification that has been done in this project, our group has created systems that include create, read, update, and delete buttons for each attribute to give users more exciting experiences. This assignment is specifically about making systems for Potlepak Restaurant using a database interface, GUI, and coding. This assignment has also made us realize how crucial interfaces are to our day-to-day existence. This is due to its ability to expedite and simplify our daily tasks. As for database integration, it utilizes MySQL to store and manage reservation information. Regarding Tkinter GUI, it uses the Tkinter to generate a user-friendly graphical data entry interface. Aside from that, the deposit amount is dynamically determined by the number of guests attending the party. It improves accuracy and organization by capturing the reservation date and time. It also can help users to have a great experience by not wasting their time using traditional systems like having to write on paper to order some things that they want. Our group will prove that the systems we create are useable and user-friendly.

2.0 PROBLEM STATEMENT

1. **Customer Registration:**
    - The current membership registration process is paper-based, resulting in data entry errors and delays.
    - Lack of a centralized system to manage and update member information promptly.
    - Inability to offer a seamless online registration experience for potential members.

2. **Employee Details:**
    - Employee details, including personal and work-related information, are stored in disparate systems, leading to difficulties in data retrieval and updates.
    - Inefficient onboarding and offboarding processes for new and departing employees.
    - Limited accessibility to essential employee information for management and HR.

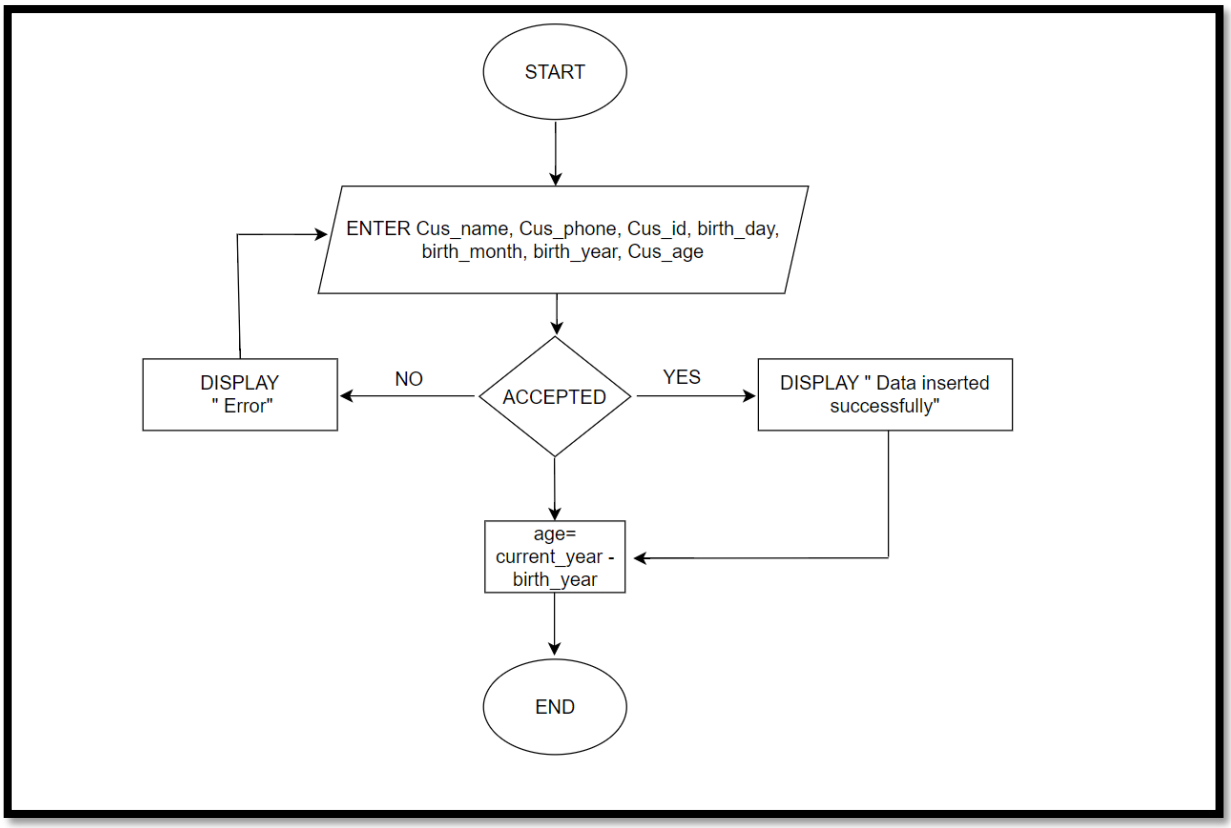
3. **Table Booking Information:**
    - Manual table booking processes contribute to errors in reservation details and inefficient use of seating capacity.
    - Inconsistent communication with customers regarding reservation confirmations and reminders.
    - Lack of an intuitive online interface for customers to easily book tables and manage reservations.

3.0 OBJECTIVES

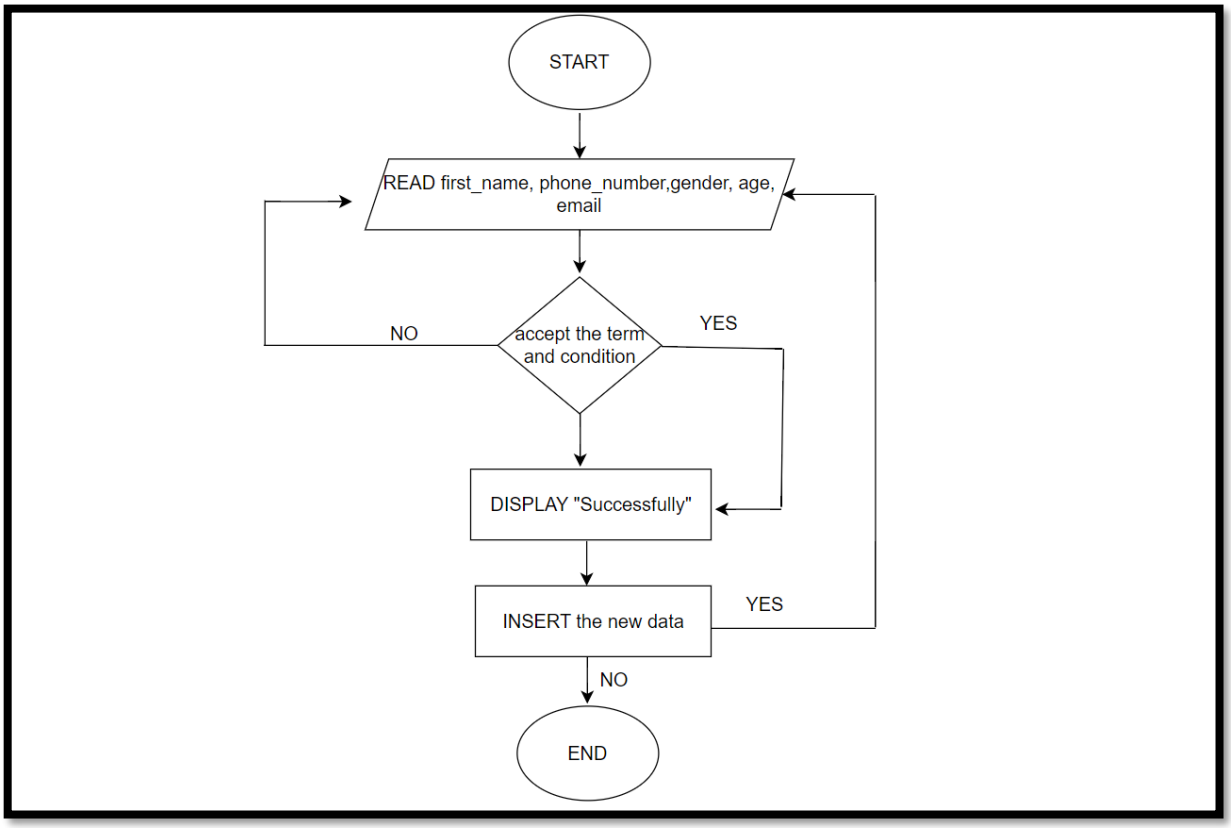
The improvement system we provide is customer registration, employee information, and table reservation information. These three systems are provided to ensure that all users can use a more systematic, high-tech system that simplifies and saves users' time without using traditional methods.

4.0 FLOWCHART

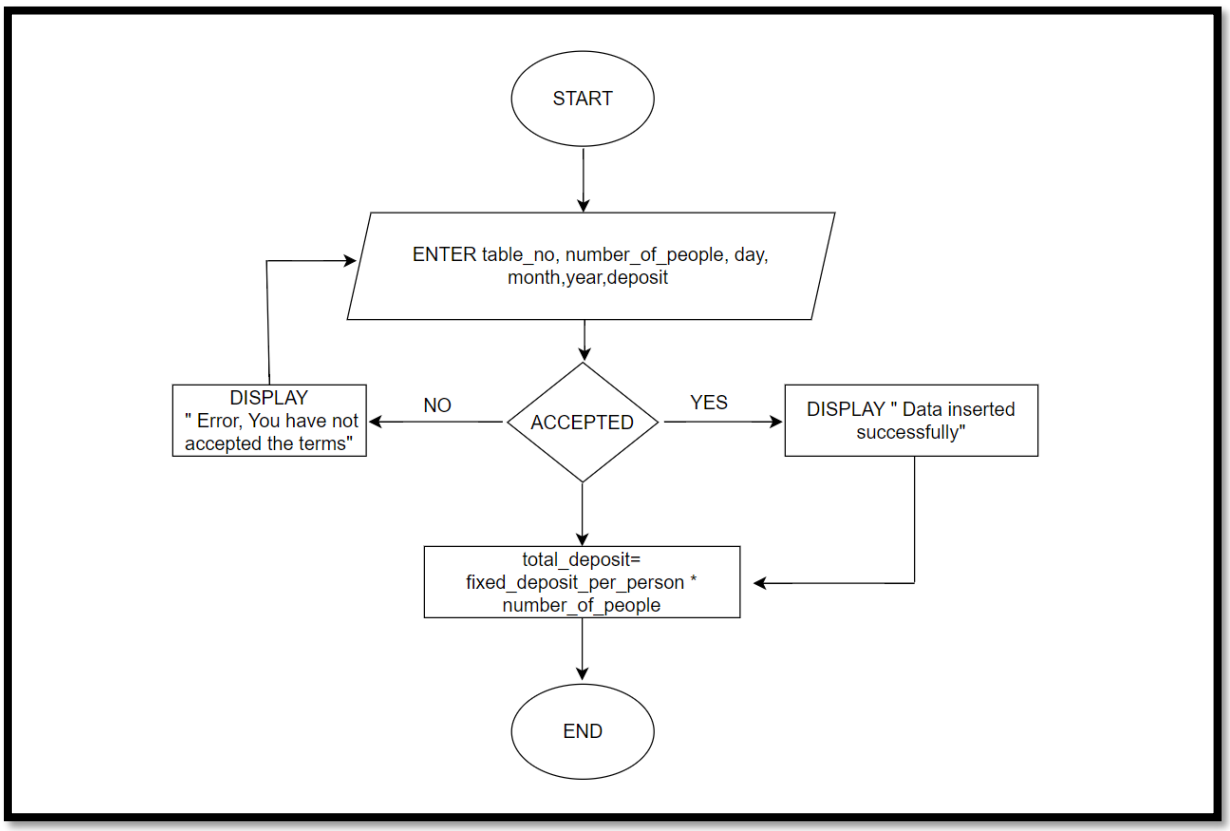
4.1 CUSTOMER REGISTRATION'S FLOWCHART



4.2 EMPLOYEE DETAIL'S FLOWCHART



4.3 TABLE RESERVATION INFORMATION'S FLOWCHART



5.0 SNAPSHOT OF CODE

5.1 CUSTOMER REGISTRATION CODE

```
import tkinter as tk
import mysql.connector
from tkinter import messagebox

mydb = mysql.connector.connect(
    host="localhost",
        user="root",
        password="",
        database="potlepak restaurant"
)

mycursor = mydb.cursor()

# Function to insert data into the table
def insert_data():
    Cus_Name = name_entry.get()
    Cus_Phone = phonenumber_entry.get()
    Cus_ID = id_entry.get()
    Birth_day = dayField.get()
    Birth_Month = monthField.get()
    Birth_year = yearField.get()
    Cus_Age = age_year_entry.get()


    # Corrected the INSERT INTO syntax and removed quotes from the data variable
    insert_query = "INSERT INTO customer (Cus_Name, Cus_Phone, Cus_ID, Birth_Day, Birth_Month, Birth_Year, Cus_Age) VALUES (%s, %s, %s, %s, %s, %s, %s)"
    data = (Cus_Name, Cus_Phone, Cus_ID, Birth_day, Birth_Month, Birth_year, Cus_Age)

    # Execute the query with the data
    mycursor.execute(insert_query, (Cus_Name, Cus_Phone, Cus_ID,Birth_day, Birth_Month, Birth_year, Cus_Age))

    # Commit the changes to the database
    mydb.commit()

    mycursor.close()
    mydb.close()
```

```
    messagebox.showinfo("Success", "Data inserted successfully!")


def calculate_age():
    # Extract values from the respective entry boxes
    birth_day = int(dayField.get())
    birth_month = int(monthField.get())
    birth_year = int(yearField.get())

    given_day = int(current_day.get())
    given_month = int(current_month.get())
    given_year = int(current_year.get())

    # If birth date is greater than given birth_month, adjust the values
    if birth_day > given_day:
        given_month -= 1
        given_day += 30  # Assuming a 30-day month to simplify

    if birth_month > given_month:
        given_year -= 1
        given_month += 12

    # Calculate day, month, year
    calculated_day = given_day - birth_day
    calculated_month = given_month - birth_month
    calculated_year = given_year - birth_year

    # Insert the calculated year into the entry box
    age_year_entry.insert(10, str(calculated_year))
```

```
def clear_entry_fields():
    name_entry.delete(0, tk.END)
    phonenumber_entry.delete(0, tk.END)
    id_entry.delete(0, tk.END)
    dayField.delete(0, tk.END)
    monthField.delete(0, tk.END)
    yearField.delete(0, tk.END)
    age_year_entry.delete(0, tk.END)

# Function to update data in the table
def update_data():
    Cus_Name = name_entry.get()
    Cus_Phone = phonenumber_entry.get()
    Cus_ID = id_entry.get()
    Birth_day = dayField.get()
    Birth_Month = monthField.get()
    Birth_year = yearField.get()
    Cus_Age = age_year_entry.get()

    if Cus_Name and Cus_ID:
        try:
            with mysql.connector.connect(
                host="localhost",
                user="root",
                password="",
                database="potlepak restaurant"
            ) as mydb:
                with mydb.cursor() as mycursor:
                    update_query = "UPDATE customer SET Cus_Name = %s, Cus_Phone = %s, Birth_Day = %s, Birth_Month = %s, Birth_Year = %s, Cus_Age = %s WHERE Cus_ID = %s"
                    mycursor.execute(update_query, (Cus_Name, Cus_Phone, Birth_day, Birth_Month, Birth_year, Cus_Age, Cus_ID))
                    mydb.commit()

            messagebox.showinfo("Success", "Data updated successfully")
        except mysql.connector.Error as err:
            tk.messagebox.showerror("Error", f"Error: {err}")
```

```
    else:
        tk.messagebox.showwarning(title="Error", message="Name and ID are required.")

# Function to delete data from the table
def delete_data():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="potlepak restaurant"
    )
    cursor = conn.cursor()

    delete_query = "DELETE FROM customer WHERE Cus_ID=%s"
    data = (id_entry.get(),)

    cursor.execute(delete_query, data)

    conn.commit()
    conn.close()
    messagebox.showinfo("Success", "Data deleted successfully")

# Tkinter GUI
root = tk.Tk()
root.title("Customer Registration")

label_name = tk.Label(root, text="Name:", font=('Times New Roman', 14, 'bold'))
label_name.grid(row=0, column=0)
name_entry = tk.Entry(root)
name_entry.grid(row=0, column=1)
```

```
label_phonenumber = tk.Label(root, text="Phone number:",font=('Times New Roman', 14, 'bold') )
label_phonenumber.grid(row=1, column=0)
phonenumber_entry = tk.Entry(root)
phonenumber_entry.grid(row=1, column=1)

label_id = tk.Label(root, text="Id:", font=('Times New Roman', 14, 'bold'))
label_id.grid(row=2, column=0)
id_entry = tk.Entry(root)
id_entry.grid(row=2, column=1)

# Date of birth
birth_date = tk.Label(root, text="Birth Day", font=('Times New Roman', 14, 'bold'))
birth_date.grid(row=3, column=0)
birth_month = tk.Label(root, text="Birth Month", font=('Times New Roman', 14, 'bold'))
birth_month.grid(row=3, column=1)
birth_year = tk.Label(root, text="Birth Year", font=('Times New Roman', 14, 'bold'))
birth_year.grid(row=3, column=2)

# Create a text entry box for filling or typing the information(dob).
dayField = tk.Entry(root)
dayField.grid(row=4, column=0)
monthField = tk.Entry(root)
monthField.grid(row=4, column=1)
yearField = tk.Entry(root)
yearField.grid(row=4, column=2)

# Current Year
curr_day = tk.Label(root, text= "Current Day", font=('Times New Roman',14, 'bold'))
curr_day.grid(row=5,column=0)
curr_month = tk.Label(root, text= "Current Month", font=('Times New Roman',14, 'bold'))
curr_month.grid(row=5, column=1)
curr_year = tk.Label(root, text= "Current Year", font=('Times New Roman',14, 'bold'))
curr_year.grid(row=5, column=2)
```

```
# Create a text entry box for filling or typing the information(current year).
current_day = tk.Entry(root)
current_day.grid(row=6, column=0)
current_month = tk.Entry(root)
current_month.grid(row=6, column=1)
current_year = tk.Entry(root)
current_year.grid(row=6, column=2)

# Age results
resultantAge = tk.Button(root, text = "Age", command = calculate_age, padx=25, pady=5)
resultantAge.grid(row=8, column=1, sticky= "news")

age_year = tk.Label(root, text= "Age:", font=('Times New Roman',14, 'bold'))
age_year.grid(row=7, column=0)
age_year_entry = tk.Entry(root)
age_year_entry.grid(row=7, column=1)

for widget in root.winfo_children():
    widget.grid(padx=10, pady=5)

insert_button = tk.Button(root, text="Insert Data", command=insert_data)
insert_button.grid(row=9, column=1, sticky="news")


# Update and Delete buttons
update_button = tk.Button(root, text="Update Data", command=update_data)
update_button.grid(row=10, column=0, sticky="news")

delete_button = tk.Button(root, text="Delete Data", command=delete_data)
delete_button.grid(row=10, column=2, sticky="news")

root.mainloop()
```

5.2 EMPLOYEE DETAIL CODE

```python
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import mysql.connector

def insert_data(employee_data, accept_var):
    accepted = accept_var.get()

    if accepted == "Accepted":
        first_name = employee_data.get("first_name")
        last_name = employee_data.get("last_name")
        number_phone = employee_data.get("number_phone")
        gender = employee_data.get("gender")
        age = employee_data.get("age")
        email = employee_data.get("email")

        #Connect to your MySQL database
        mydb = mysql.connector.connect(
            host = "localhost",
            user = "root",
            password = "",
            database = "potlepak restaurant"

        )

        #Create a cursor object to execute SQL queries
        mycursor = mydb.cursor()

        #SQL query to insert data into the table
        insert_query = "INSERT INTO employee (first_name, last_name, number_phone, gender, age, email) VALUES (%s, %s, %s, %s, %s, %s)"

        #Execute the query with the data
        mycursor.execute(insert_query, (first_name, last_name, number_phone, gender, age, email))
```

```python
        #Execute the query with the data
        mycursor.execute(insert_query, (first_name, last_name, number_phone, gender, age, email))

        #Commit the changes to the database
        mydb.commit()

        mycursor.close()
        mydb.close()

        messagebox.showinfo("Success", "Data inserted successfully")

    else :
        tk.messagebok.showwarning(title = "Error", message = "You have not accepted the terms")

def clear_entries():
    first_name_entry.delete(0, tk.END)
    last_name_entry.delete(0, tk.END)
    number_phone_entry.delete(0, tk.END)
    gender_entry.delete(0, tk.END)
    age_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)

def employee_registration():
    while True:
        root = tk.Tk()
        root.title("Employee Detail")
        root.geometry("400x500")

        employee_info = tk.LabelFrame(root, text="Employee Registration", font=('Times New Roman', 12))
        employee_info.grid(row=1, column=0, padx=20, pady=10)
```

```python
def update_data():
    accepted = accept_var.get()

    if accepted == "Accepted":
        # Employee Information
        first_name = first_name_entry.get()
        last_name = last_name_entry.get()
        age = age_entry.get()
        email = email_entry.get()

        if first_name and last_name:
            email = email_entry.get()

            # Connect to your MySQL database
            mydb = mysql.connector.connect(
                host="localhost",
                user="root",
                password="",
                database="potlepak restaurant"
            )

            # Create a cursor object to execute SQL queries
            mycursor = mydb.cursor()

            # SQL query to update data in the table
            update_query = "UPDATE employee SET age = %s, email = %s WHERE first_name = %s AND last_name = %s"

            # Execute the query with the data
            mycursor.execute(update_query, (age, email, first_name, last_name))

            # Commit the changes to the database
            mydb.commit()

            mycursor.close()
            mydb.close()
```

```
            messagebox.showinfo("Success", "Data updated successfully")
        else:
            tk.messagebox.showwarning(title="Error", message="First Name and Last Name are required")
    else:
        tk.messagebox.showwarning(title="Error", message="You have not accepted the terms")

def delete_data():
    email = email_entry.get()

    # Connect to your MySQL database
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="potlepak restaurant"
    )

    # Create a cursor object to execute SQL queries
    mycursor = mydb.cursor()

    # SQL query to delete data from the table
    delete_query = "DELETE FROM employee WHERE email = %s"

    # Execute the query with the data
    mycursor.execute(delete_query, (email,))

    # Commit the changes to the database
    mydb.commit()

    mycursor.close()
    mydb.close()

    messagebox.showinfo("Success", "Data deleted successfully")
```

```
    first_name_entry.delete(0, tk.END)
    last_name_entry.delete(0, tk.END)
    number_phone_entry.delete(0, tk.END)
    gender_entry.delete(0, tk.END)
    age_entry.delete(0, tk.END)
    email_entry.delete(0, tk.END)

root = tk.Tk()
root.title("Employee Detail")
root.geometry("400x500")

employee_info = tk.LabelFrame(root, text="Employee Registration", font=('Times New Roman', 12))
employee_info.grid(row=1, column=0, padx=20, pady=10)

first_name_label = tk.Label(employee_info, text="First Name :", font=('Times New Roman', 14))
first_name_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
first_name_label.grid(row=0, column=0)
first_name_entry.grid(row=0, column=1)

last_name_label = tk.Label(employee_info, text="Last Name :", font=('Times New Roman', 14))
last_name_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
last_name_label.grid(row=1, column=0)
last_name_entry.grid(row=1, column=1)

number_phone_label = tk.Label(employee_info, text="Phone Number :", font=('Times New Roman', 14))
number_phone_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
number_phone_label.grid(row=2, column=0)
number_phone_entry.grid(row=2, column=1)

gender_label = tk.Label(employee_info, text="Gender :", font=('Times New Roman', 14))
gender_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
gender_label.grid(row=3, column=0)
gender_entry.grid(row=3, column=1)
```

```
age_label = tk.Label(employee_info, text="Age :", font=('Times New Roman', 14))
age_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
age_label.grid(row=4, column=0)
age_entry.grid(row=4, column=1)

email_label = tk.Label(employee_info, text="Email :", font=('Times New Roman', 14))
email_entry = tk.Entry(employee_info, font=('Adlam Display', 12))
email_label.grid(row=5, column=0)
email_entry.grid(row=5, column=1)

for widget in employee_info.winfo_children():
    widget.grid_configure(padx = 10, pady = 5)

#Accept terms
terms_frame = tk.LabelFrame(root, text = "Terms and Conditions")
terms_frame.grid(row = 6, column = 0, sticky = "news", padx =20, pady = 20)

accept_var = tk.StringVar(value = "Not Value")
terms_check = tk.Checkbutton(terms_frame, text = "I accept the terms and conditions", variable = accept_var, onvalue = "Accepted", offvalue = "Not Accepted")
terms_check.grid(row = 0, column = 0)

# Insert Data
button = tk.Button(root, text="Insert Data", command=lambda: insert_data({
    "first_name": first_name_entry.get(),
    "last_name": last_name_entry.get(),
    "number_phone": number_phone_entry.get(),
    "gender": gender_entry.get(),
    "age": age_entry.get(),
    "email": email_entry.get()
}, accept_var))
button.grid(row=7, column=0, sticky="n", padx=5, pady=3)

#Update Button
update_button = tk.Button(root, text = "Update Data", command = update_data)
update_button.grid(row = 8, column = 0, sticky = "n", padx = 5, pady = 3)
```

```
#Delete Button
delete_button = tk.Button(root, text = "Delete Data", command = delete_data)
delete_button.grid(row = 10, column = 0, sticky = "n", padx = 5, pady = 3)

# Clear Entries Button
clear_button = tk.Button(root, text="Clear Entries", command=clear_entries)
clear_button.grid(row=9, column=0, sticky="n", padx=5, pady=3)

# Break the loop and exit the window when the user closes it
root.protocol("WM_DELETE_WINDOW", root.destroy)

root.mainloop()
```

5.3 TABLE RESERVATION INFORMATION CODE

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import mysql.connector

def insert_data():
    accepted = accept_var.get()

    if accepted == "Accepted":
        table_no = table_spinbox.get()
        number_of_people = no_people_spinbox.get()
        day = day_entry.get()
        month = month_entry.get()
        year = year_entry.get()
        deposit = deposit_entry.get()

        # Validate numeric input for number_of_people and deposit
        if not (number_of_people.isdigit() and deposit.replace('.', '', 1).isdigit()):
            tk.messagebox.showwarning(title="Error", message="Invalid input for number of people or deposit.")
            return

        # Connect to your MySQL database
        mydb = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="potlepak restaurant"
    )

        # Create a cursor object to execute SQL queries
        mycursor = mydb.cursor()
```

```
        # Create a cursor object to execute SQL queries
        mycursor = mydb.cursor()

        # SQL query to insert data into the table
        insert_query = "INSERT INTO `table reservation` (Table_Number, Number_of_Customer, Day, Month, Year, Deposit) VALUES (%s, %s, %s, %s, %s, %s)"

        # Execute the query with the data
        mycursor.execute(insert_query, (table_no, number_of_people, day, month, year, deposit))

        # Commit the changes to the database
        mydb.commit()

        mycursor.close()
        mydb.close()

    else:
        tk.messagebox.showwarning(title="Error", message="You have not accepted the terms.")

    messagebox.showinfo("Success", "Data inserted successfully!")

def update_data():
    accepted = accept_var.get()

    if accepted == "Accepted":
        # Customer Information
        table_number = table_spinbox.get()
        number_of_people = no_people_spinbox.get()

        if table_number and number_of_people:
            day = day_entry.get()
            month = month_entry.get()
            year = year_entry.get()
            deposit = deposit_entry.get()
```

```
            # Validate numeric input for number_of_people and deposit
            if not (number_of_people.isdigit() and deposit.replace('.', '', 1).isdigit()):
                tk.messagebox.showwarning(title="Error", message="Invalid input for number of people or deposit.")
                return

            # Connect to your MySQL database
            mydb = mysql.connector.connect(
                host="localhost",
                user="root",
                password="",
                database="potlepak restaurant"
            )

            # Create a cursor object to execute SQL queries
            mycursor = mydb.cursor()

            # SQL query to update data in the table
            update_query = "UPDATE `table reservation` SET Number_of_Customer=%s, Day=%s, Month=%s, Year=%s, Deposit=%s WHERE Table_Number=%s"

            # Execute the query with the data
            mycursor.execute(update_query, (number_of_people, day, month, year, deposit, table_number))

            # Commit the changes to the database
            mydb.commit()

            mycursor.close()
            mydb.close()

            messagebox.showinfo("Success", "Data updated successfully!")
        else:
            tk.messagebox.showwarning(title="Error", message="Table number and number of people are required.")
    else:
        tk.messagebox.showwarning(title="Error", message="You have not accepted the terms.")
```

```python
def delete_data():
    table_number = table_spinbox.get()

    # Validate numeric input for table_number
    if not table_number.isdigit():
        tk.messagebox.showwarning(title="Error", message="Invalid input for table number.")
        return

    # Connect to your MySQL database
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="potlepak restaurant"
    )

    # Create a cursor object to execute SQL queries
    mycursor = mydb.cursor()

    # SQL query to delete data from the table
    delete_query = "DELETE FROM `table reservation` WHERE Table_Number=%s"

    # Execute the query with the data
    mycursor.execute(delete_query, (table_number,))

    # Commit the changes to the database
    mydb.commit()

    mycursor.close()
    mydb.close()

    messagebox.showinfo("Success", "Data deleted successfully!")
```

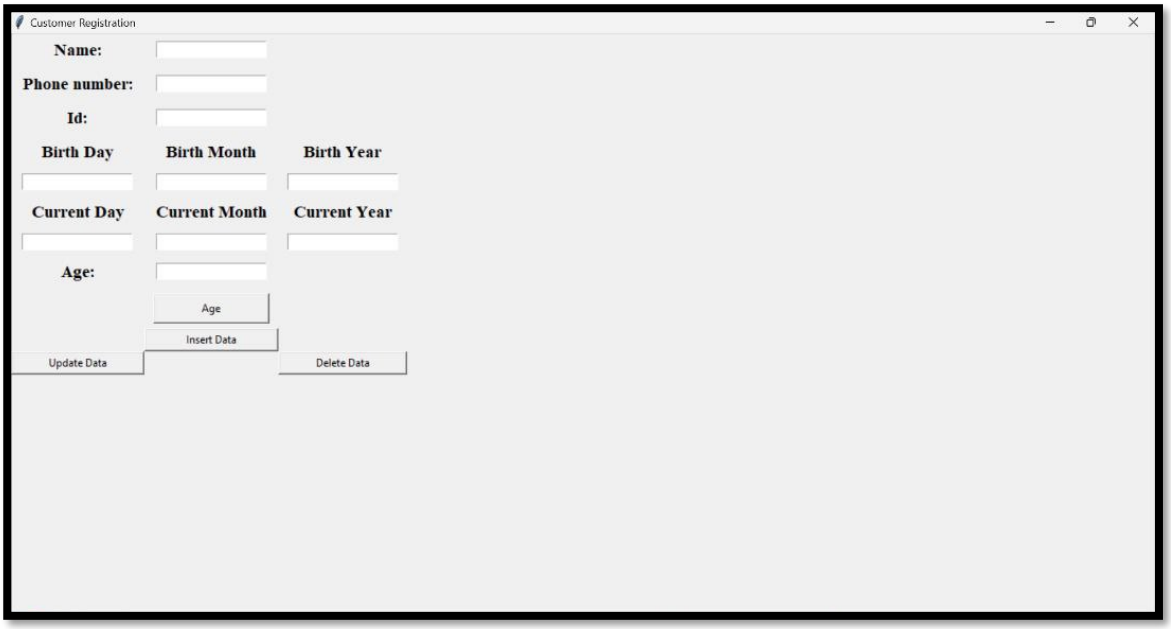
```python
# Fixed deposit amount per person
fixed_deposit_per_person = 5

def calculate_cost_per_person():
    try:
        number_of_people = int(no_people_spinbox.get())

        total_deposit = fixed_deposit_per_person * number_of_people

        # Update the deposit entry field with the calculated total deposit
        deposit_entry.delete(0, tk.END)  # Clear the current value
        deposit_entry.insert(0, str(total_deposit))  # Insert the calculated total deposit

        # Display the result in a messagebox
        result_message = f"Fixed Deposit per Person: {fixed_deposit_per_person}\nNumber of People: {number_of_people}\nTotal Deposit Amount: {total_deposit}"
        tk.messagebox.showinfo(title="Calculation Result", message=result_message)

    except ValueError:
        tk.messagebox.showwarning(title="Error", message="Invalid number of people.")


root = tk.Tk()
root.title("Table Reservation Form")

# Table Booking Information
table_info = tk.LabelFrame(root, text="Table Booking Information", font=('Arial', 16, 'bold'))
table_info.grid(row=1, column=0, padx=20, pady=20)

table_number_label = tk.Label(table_info, text="Table Number", font=('Arial', 14))
table_spinbox = ttk.Spinbox(table_info, from_=1, to=30, font=('Arial', 12))
table_number_label.grid(row=0, column=0)
table_spinbox.grid(row=1, column=0)
```

```
no_people_label = tk.Label(table_info, text="Number of People", font=('Arial', 14))
no_people_spinbox = ttk.Spinbox(table_info, from_=1, to=15, font=('Arial', 12))
no_people_label.grid(row=0, column=1)
no_people_spinbox.grid(row=1, column=1)

day_label = tk.Label(table_info, text="Day:", font=('Arial', 14))
month_label = tk.Label(table_info, text="Month:", font=('Arial', 14))
year_label = tk.Label(table_info, text="Year", font=('Arial', 14))
day_label.grid(row=2, column=0)
month_label.grid(row=3, column=0)
year_label.grid(row=4, column=0)

day_entry = tk.Entry(table_info, font=('Arial', 12))
month_entry = tk.Entry(table_info, font=('Arial', 12))
year_entry = tk.Entry(table_info, font=('Arial', 12))
day_entry.grid(row=2, column=1)
month_entry.grid(row=3, column=1)
year_entry.grid(row=4, column=1)

for widget in table_info.winfo_children():
    widget.grid_configure(padx=10, pady=5)

# Accept terms
terms_frame = tk.LabelFrame(root, text="Terms and Conditions")
terms_frame.grid(row=2, column=0, sticky="news", padx=20, pady=20)

accept_var = tk.StringVar(value="Not Accepted")
terms_check = tk.Checkbutton(terms_frame, text="I accept the terms and conditions", variable=accept_var, onvalue="Accepted", offvalue="Not Accepted")
terms_check.grid(row=0, column=0)

# Deposit customer paid
deposit_label = tk.Label(root, text="RM (deposit):", font=('Arial', 14))
deposit_entry = tk.Entry(root, font=('Arial', 12))
deposit_label.grid(row=3, column=0)
deposit_entry.grid(row=4, column=0)
```

```
# Calculate Button
calculate_button = tk.Button(root, text="Calculate Deposit", command=calculate_cost_per_person)
calculate_button.grid(row=5, column=0, sticky="n", padx=5, pady=3)

# Insert Data
button = tk.Button(root, text="Insert Data", command=insert_data)
button.grid(row=7, column=0, sticky="n", padx=5, pady=2)

# Update Button
update_button = tk.Button(root, text="Update Data", command=update_data)
update_button.grid(row=8, column=0, sticky="n", padx=5, pady=3)

# Delete Button
delete_button = tk.Button(root, text="Delete Data", command=delete_data)
delete_button.grid(row=9, column=0, sticky="n", padx=5, pady=3)

root.mainloop()
```

6.0 SNAPSHOT OF GUI

6.1 GUI OF CUSTOMER REGISTRATION

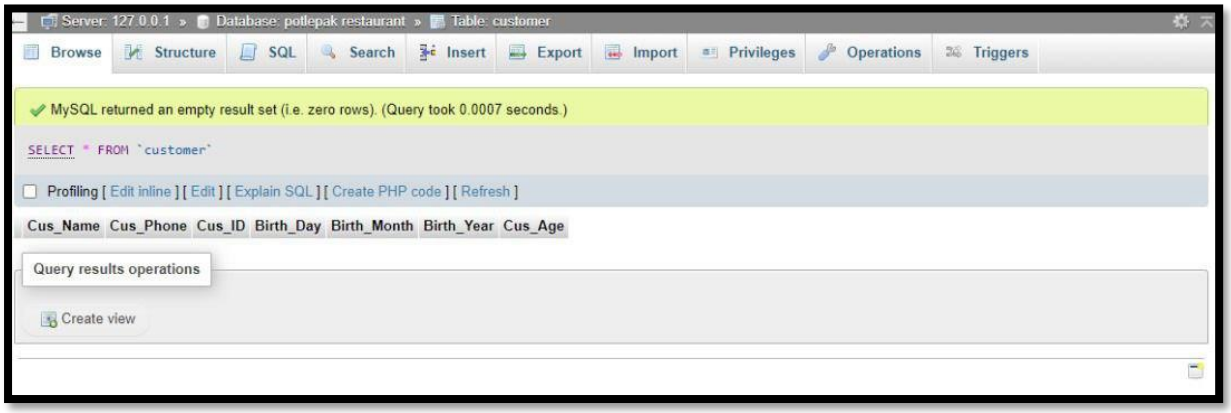


6.2 GUI OF EMPLOYEE DETAIL

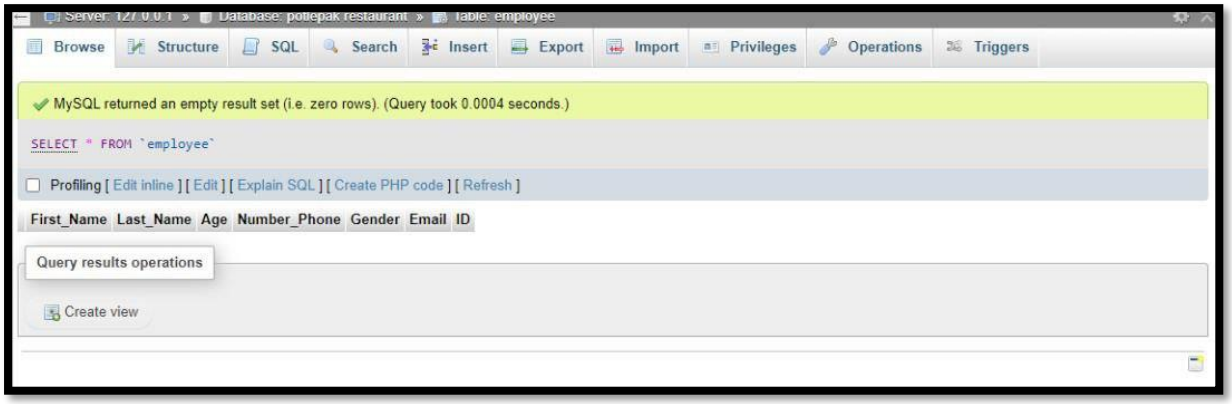
6.3 GUI OF TABLE RESERVATION INFORMATION

7.0 SNAPSHOT OF DATABASE

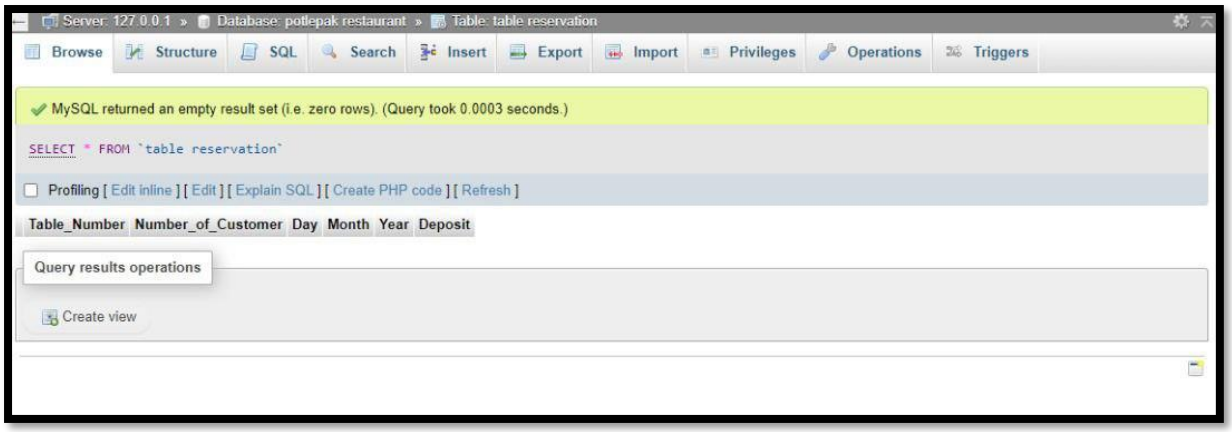
7.1 CUSTOMER REGISTRATION DATABASE



7.2 EMPLOYEE DETAIL DATABASE



7.3 TABLE RESERVATION INFORMATION DATABASE



8.0 CONCLUSION

In conclusion, our strategic project for Potlepak Restaurant represents a new step into a future where technology will fundamentally transform the restaurant industry, rather than just solving current problems. Our team envisions an easy digital experience where productivity, connectivity, and user satisfaction are prioritized, challenging the status current and reimagining every aspect of the conventional traditional procedures. We are committed to personalized design, which is demonstrated by the focus on creating, reading, updating, and deleting functionalities for every attribute. We also used a reliable database powered by MySQL and an intuitive Tkinter GUI. In addition to improving accuracy and organization, the rapid deposit calculation, reservation date, and time capture also help to create an unmatched experience for both customers and employees. This assignment shows how important it is to have upgraded technology in our lives. This assignment has also made us realize how crucial interfaces are to our day-to-day existence.