A Minor Project Report

On

Customer rating on an

Bachelor of Technology

In

Computer Science and Engineering (AIML)

2023-2027

By

Name – Kasmya Bhatia

Registration Number – 23FE10CAI00010

MANIPAL UNIVERSITY
JAIPUR

Under the supervision of

Mr. Sanjay Kumar Tehariya

School Of Computer Science And Engineering

Department Of Artificial Intelligence And Machine Learning

Manipal University Jaipur, Jaipur, Rajasthan, India

Introduction:

In today's era of digital transformation, customer feedback forms the backbone of success for any app or website. Customer ratings can be seen as a highly relevant parameter in judging user satisfaction, quality service provision, and winning the trust of potential users. A sound rating system will provide organizations with significant insights to make informed decisions and improve on their offerings based on data.

The purpose of the Customer Rating App mini-project is to simulate collecting and analyzing customer ratings for an app or website. Using Python, the project generates randomly distributed customer data with ratings, demographic details, and a unique identifier. The data will be structured in a format using a Pandas DataFrame, and stored as a long-term CSV file.

It also utilizes an object-oriented approach to programming for the encapsulation of customer information through class objects, making it manageable and scalable. This allows customers with high rating scores to be identified and highlighted, thus creating a sense of actionable insight into improving one's decisions.

In a nutshell, this project shares a basic understanding of customer feedback systems while demonstrating some key concepts in Python, such as generating random numbers, string manipulation, file handling, and data analysis with Pandas.

Project Workflow:

---

1. Data Generation

- **Input**: Define the number of customers to generate data for.
- **Process**:
    - Generate random customer data, including:
        - **Customer IDs**: Unique identifiers for each customer.
        - **Names**: Randomly generated customer names.

        - **Age**: Random age values within a defined range (e.g., 18–70 years).
        - **Mobile Numbers**: Randomly generated 10-digit mobile numbers.
        - **Ratings**: Random customer ratings on a scale of 1 to 5.
- **Output**: A list of customer data.

---

2. Data Storage

- **Process**:
    - Organize the generated data into a structured format using a **Pandas DataFrame**.
    - Export the DataFrame to a CSV file to enable permanent storage.
- **Output**: A CSV file (e.g., customer_data.csv) containing all the generated data.

---

3. Data Loading

- **Input**: Load data from the saved CSV file.
- **Process**:
    - Read the CSV file using Pandas.
    - Convert the data into a list of **Customer objects** (using object-oriented principles).
- **Output**: A list of Customer objects for further processing.

---

4. Data Filtering

- **Input**: The list of Customer objects.
- **Process**:
    - Apply filtering criteria (e.g., customers with a rating of 3.5 or higher).
    - Identify customers who meet the criteria and prepare the filtered list.
- **Output**: A filtered list of high-rated customers.

---

5. Insights and Actions

- **Input**: The filtered customer list.

- **Process**:
    - Simulate actionable insights based on customer ratings, such as:
        - Identifying high-value customers.
        - Highlighting trends in customer satisfaction.
        - Suggesting potential improvements to services.
- **Output**: Insights that can guide decision-making (e.g., a printed summary of high-rated customers).

---

6. Data Visualization (Optional)

- **Process**:
    - Use Python libraries (e.g., Matplotlib or Seaborn) to visualize customer data.
    - Create bar charts, pie charts, or histograms to display:
        - Rating distributions.
        - Demographic trends (age, ratings, etc.).
- **Output**: Graphical representations of customer data and insights.

---

7. Graphical User Interface

- **Process**:
    - Build a **Tkinter**-based interface to interact with the app.
    - Add features like:
        - Generating data.
        - Loading and displaying customer data.
        - Filtering data by rating.
    - Display the data in a table format for better user interaction.
- **Output**: An interactive GUI for non-programmers.

Code Implementation :

```
import pandas as pd
import string
import random

# Step 1: Generate random ratings between 1 and 5
def generate_rating():
    return round(random.uniform(1, 5), 1)

# Step 2: Generate customer names
def generate_customer_names(num_customers, name_length=50):
    base_string = ''.join(random.choices(string.ascii_letters, k=name_length))
    return [base_string[:random.randint(3, 10)] for _ in range(num_customers)]

# Step 3: Generate other customer details
def generate_customer_details(num_customers):
    customer_ids = [random.randint(1000, 9999) for _ in range(num_customers)]
    ages = [random.randint(18, 70) for _ in range(num_customers)]
    mobile_numbers = [random.randint(7000000000, 9999999999) for _ in
range(num_customers)]
    ratings = [generate_rating() for _ in range(num_customers)]
    return customer_ids, ages, mobile_numbers, ratings

# Step 4: Store data in a DataFrame and save it as a CSV file
def create_customer_data(num_customers=10):
    names = generate_customer_names(num_customers)
```

```
    ids, ages, mobiles, ratings = generate_customer_details(num_customers)
    data = pd.DataFrame({
        "Customer ID": ids,
        "Name": names,
        "Age": ages,
        "Mobile No.": mobiles,
        "Rating": ratings
    })
    data.to_csv("customer_data.csv", index=False)
    print("Customer data saved to 'customer_data.csv'")
    return data

# Step 5: Load data from the CSV into a list of objects (using classes)
class Customer:
    def __init__(self, customer_id, name, age, mobile, rating):  # Corrected method name
        self.customer_id = customer_id
        self.name = name
        self.age = age
        self.mobile = mobile
        self.rating = rating

def load_customer_data(filename="customer_data.csv"):
    data = pd.read_csv(filename)
    customers = [
        Customer(row["Customer ID"], row["Name"], row["Age"], row["Mobile No."],
row["Rating"])
        for _, row in data.iterrows()
    ]
```

```
    return customers

# Step 6: Filter customers with an average rating >= 3.5
def filter_customers_by_rating(customers, threshold=3.5):
    return [customer for customer in customers if customer.rating >= threshold]

# Main Execution
if __name__ == "__main__":  # Corrected condition
    num_customers = 15  # Specify the number of customers
    df = create_customer_data(num_customers)

    customers = load_customer_data()
    high_rating_customers = filter_customers_by_rating(customers)

    print(f"Customers with rating >= 3.5 ({len(high_rating_customers)}):")
    for customer in high_rating_customers:
        print(f"ID: {customer.customer_id}, Name: {customer.name}, Rating:
{customer.rating}")



import tkinter as tk
from tkinter import ttk, messagebox
import pandas as pd
import string
import random

# Step 1: Generate random ratings between 1 and 5
```

```
def generate_rating():
    return round(random.uniform(1, 5), 1)

# Step 2: Generate customer names
def generate_customer_names(num_customers, name_length=50):
    base_string = ''.join(random.choices(string.ascii_letters, k=name_length))
    return [base_string[:random.randint(3, 10)] for _ in range(num_customers)]

# Step 3: Generate other customer details
def generate_customer_details(num_customers):
    customer_ids = [random.randint(1000, 9999) for _ in range(num_customers)]
    ages = [random.randint(18, 70) for _ in range(num_customers)]
    mobile_numbers = [random.randint(7000000000, 9999999999) for _ in
range(num_customers)]
    ratings = [generate_rating() for _ in range(num_customers)]
    return customer_ids, ages, mobile_numbers, ratings

# Step 4: Store data in a DataFrame and save it as a CSV file
def create_customer_data(num_customers=10):
    names = generate_customer_names(num_customers)
    ids, ages, mobiles, ratings = generate_customer_details(num_customers)
    data = pd.DataFrame({
        "Customer ID": ids,
        "Name": names,
        "Age": ages,
        "Mobile No.": mobiles,
        "Rating": ratings
    })
```

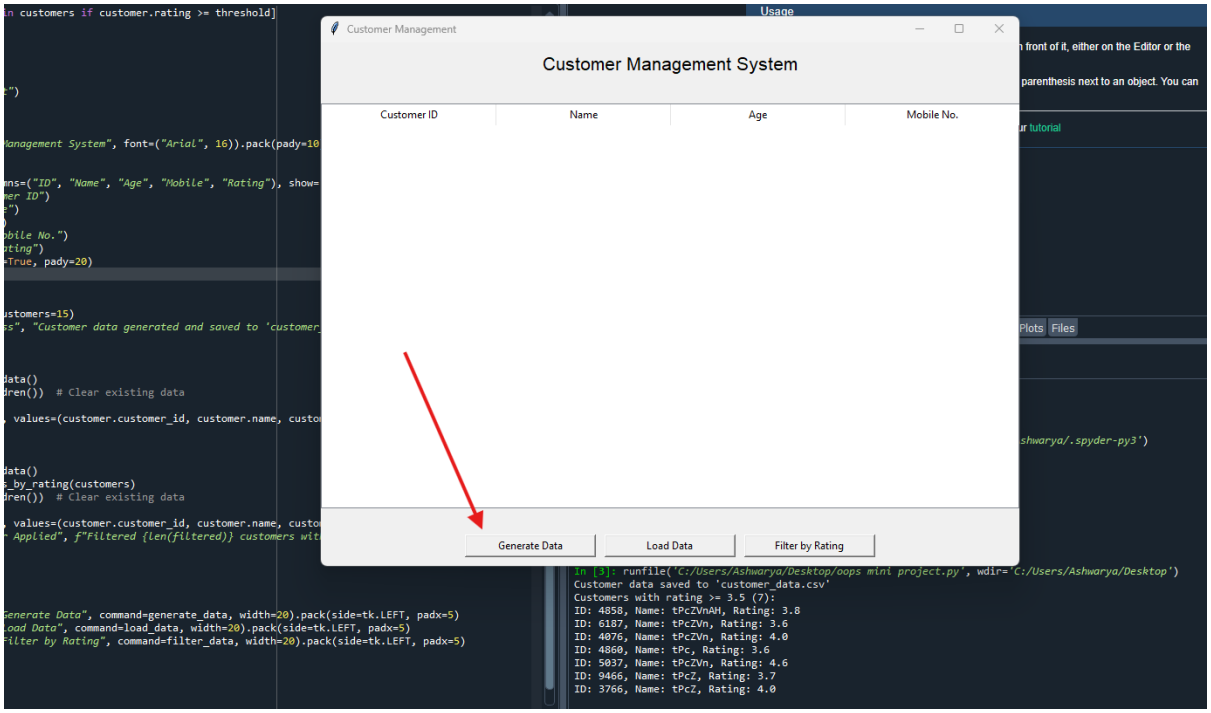
```
    data.to_csv("customer_data.csv", index=False)
    return data

# Step 5: Load data from the CSV into a list of objects (using classes)
class Customer:
    def __init__(self, customer_id, name, age, mobile, rating):
        self.customer_id = customer_id
        self.name = name
        self.age = age
        self.mobile = mobile
        self.rating = rating

def load_customer_data(filename="customer_data.csv"):
    try:
        data = pd.read_csv(filename)
        customers = [
            Customer(row["Customer ID"], row["Name"], row["Age"], row["Mobile No."],
row["Rating"])
            for _, row in data.iterrows()
        ]
        return customers
    except FileNotFoundError:
        messagebox.showerror("Error", f"{filename} not found!")
        return []

# Step 6: Filter customers with an average rating >= 3.5
def filter_customers_by_rating(customers, threshold=3.5):
    return [customer for customer in customers if customer.rating >= threshold]
```

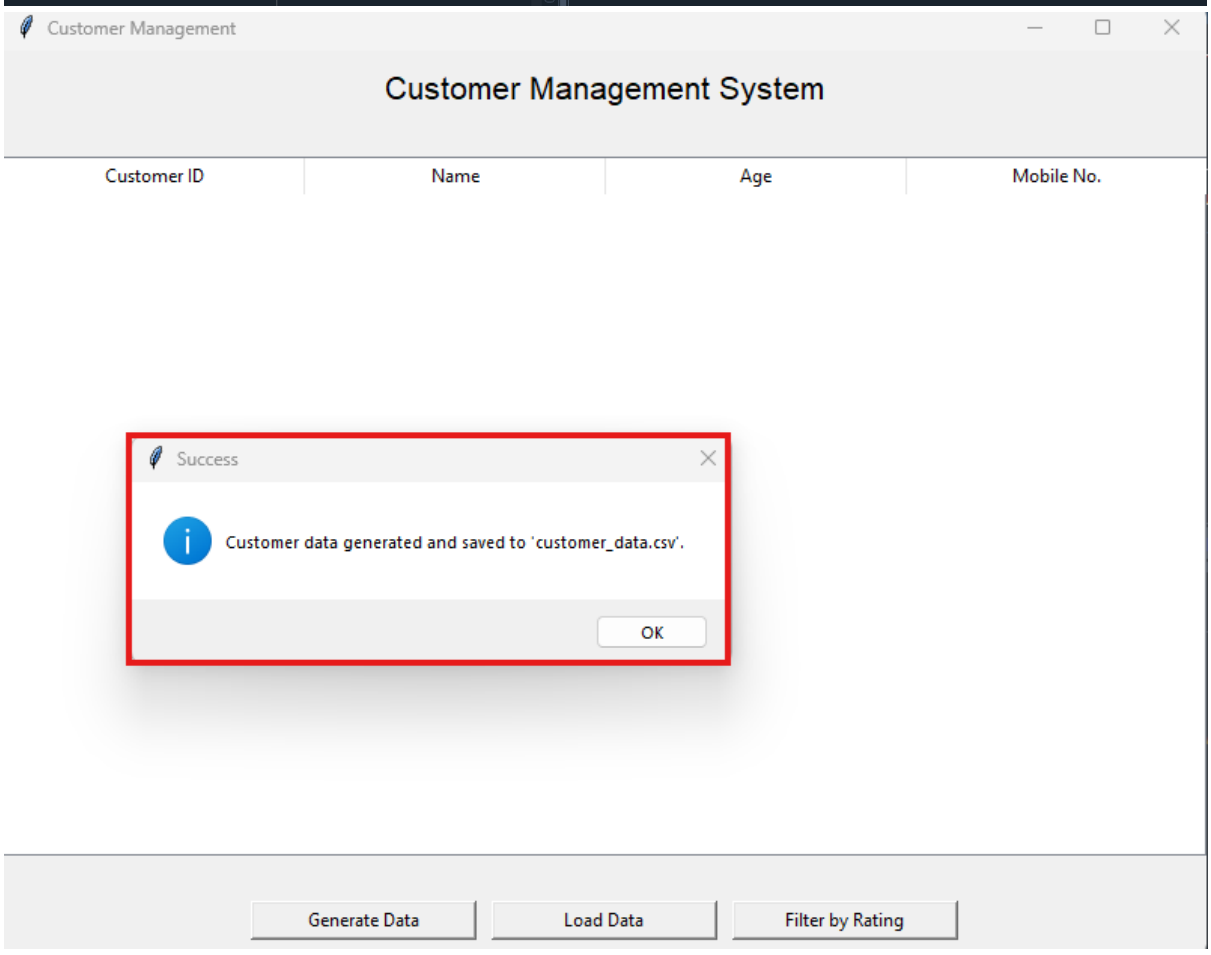
```
# Tkinter GUI Implementation
def create_interface():
    # Main window
    root = tk.Tk()
    root.title("Customer Management")
    root.geometry("800x600")

    # Label
    tk.Label(root, text="Customer Management System", font=("Arial", 16)).pack(pady=10)

    # Treeview to display data
    tree = ttk.Treeview(root, columns=("ID", "Name", "Age", "Mobile", "Rating"),
show="headings")
    tree.heading("ID", text="Customer ID")
    tree.heading("Name", text="Name")
    tree.heading("Age", text="Age")
    tree.heading("Mobile", text="Mobile No.")
    tree.heading("Rating", text="Rating")
    tree.pack(fill=tk.BOTH, expand=True, pady=20)

    # Generate customer data
    def generate_data():
        create_customer_data(num_customers=15)
        messagebox.showinfo("Success", "Customer data generated and saved to
'customer_data.csv'.")

    # Load customer data
    def load_data():
```

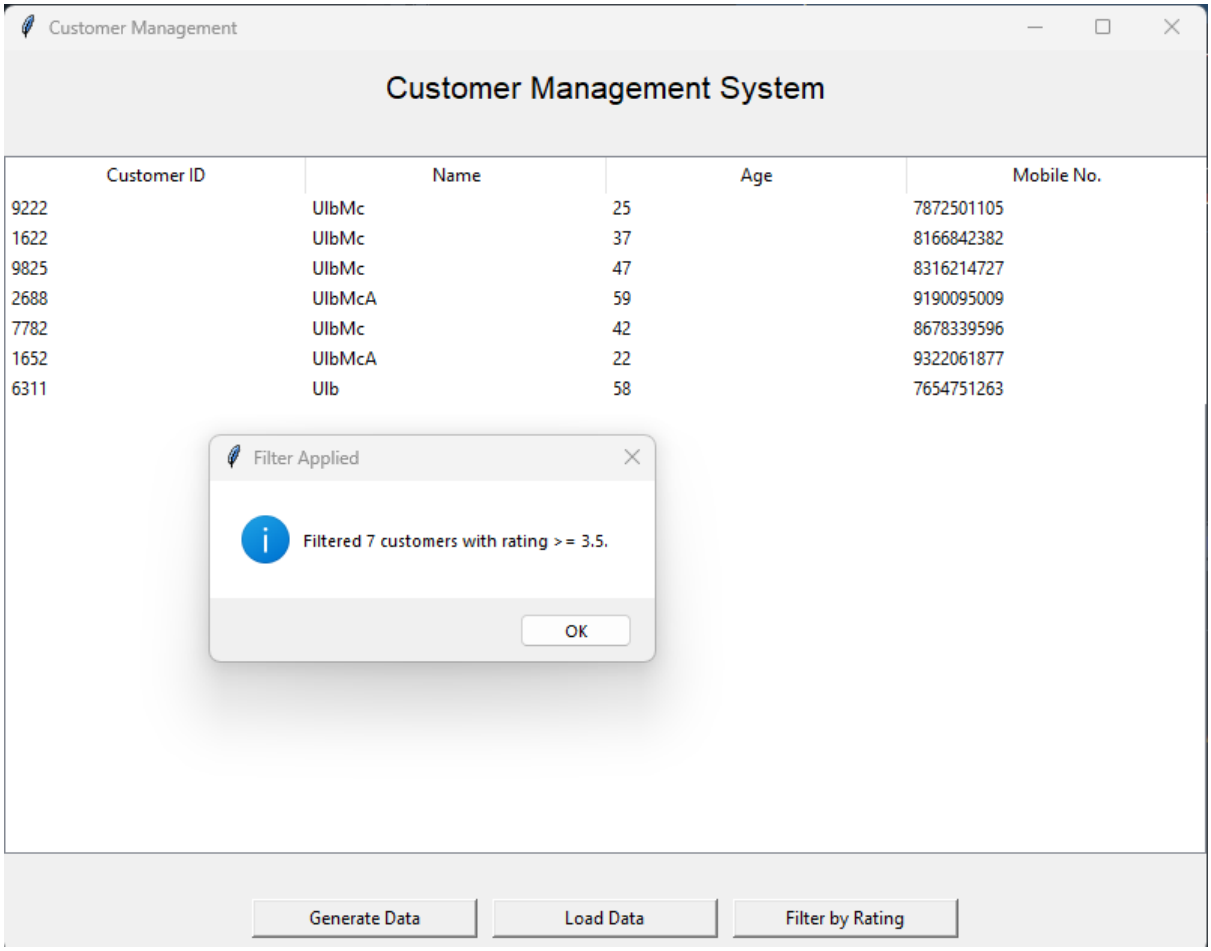
```
    customers = load_customer_data()
    tree.delete(*tree.get_children())  # Clear existing data
    for customer in customers:
        tree.insert("", tk.END, values=(customer.customer_id, customer.name, customer.age, customer.mobile, customer.rating))

# Filter high-rating customers
def filter_data():
    customers = load_customer_data()
    filtered = filter_customers_by_rating(customers)
    tree.delete(*tree.get_children())  # Clear existing data
    for customer in filtered:
        tree.insert("", tk.END, values=(customer.customer_id, customer.name, customer.age, customer.mobile, customer.rating))
    messagebox.showinfo("Filter Applied", f"Filtered {len(filtered)} customers with rating >= 3.5.")

# Buttons
button_frame = tk.Frame(root)
button_frame.pack(pady=10)

tk.Button(button_frame, text="Generate Data", command=generate_data, width=20).pack(side=tk.LEFT, padx=5)
tk.Button(button_frame, text="Load Data", command=load_data, width=20).pack(side=tk.LEFT, padx=5)
tk.Button(button_frame, text="Filter by Rating", command=filter_data, width=20).pack(side=tk.LEFT, padx=5)

root.mainloop()
```

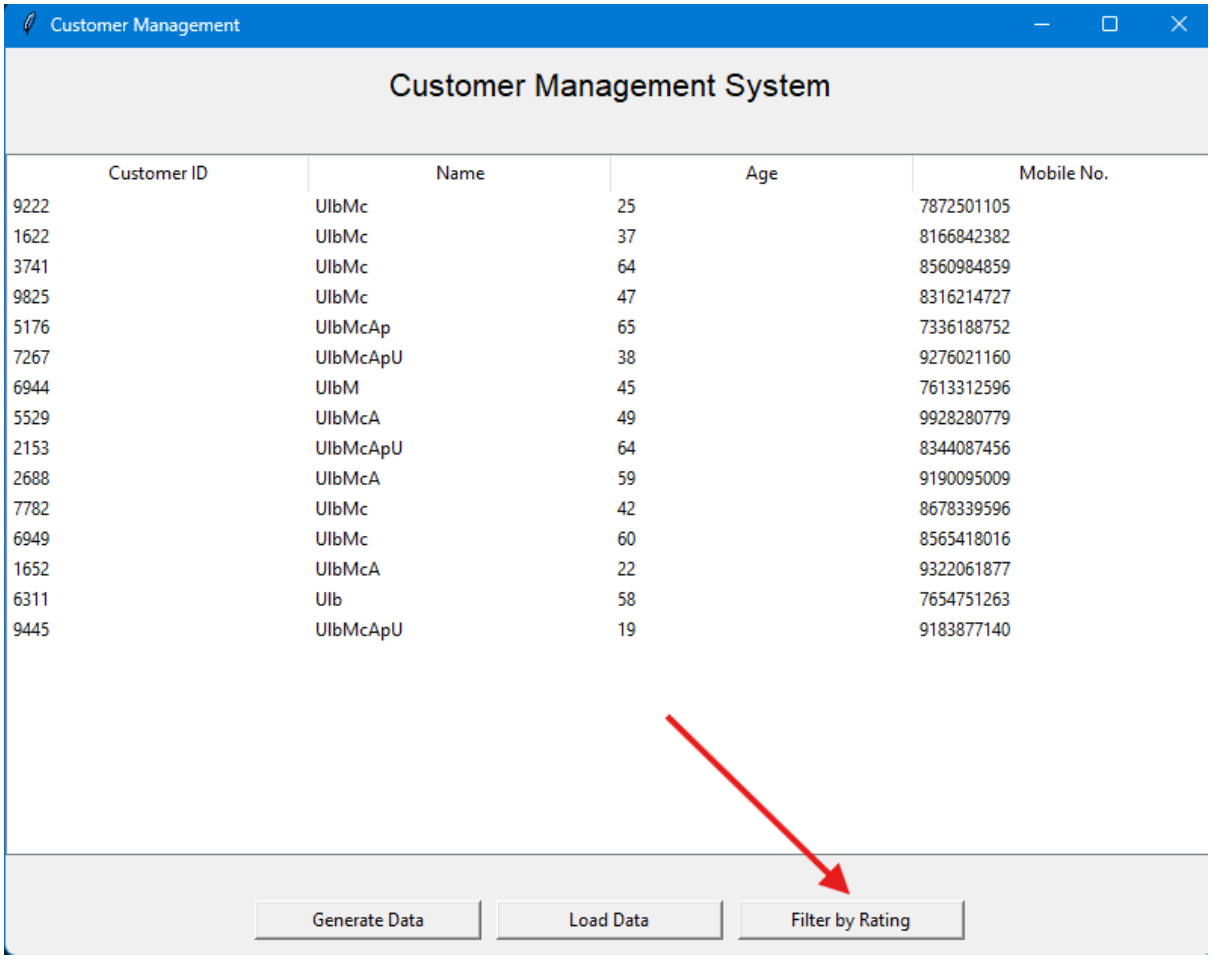
```
# Run the GUI
if __name__ == "__main__":
    create_interface()
```

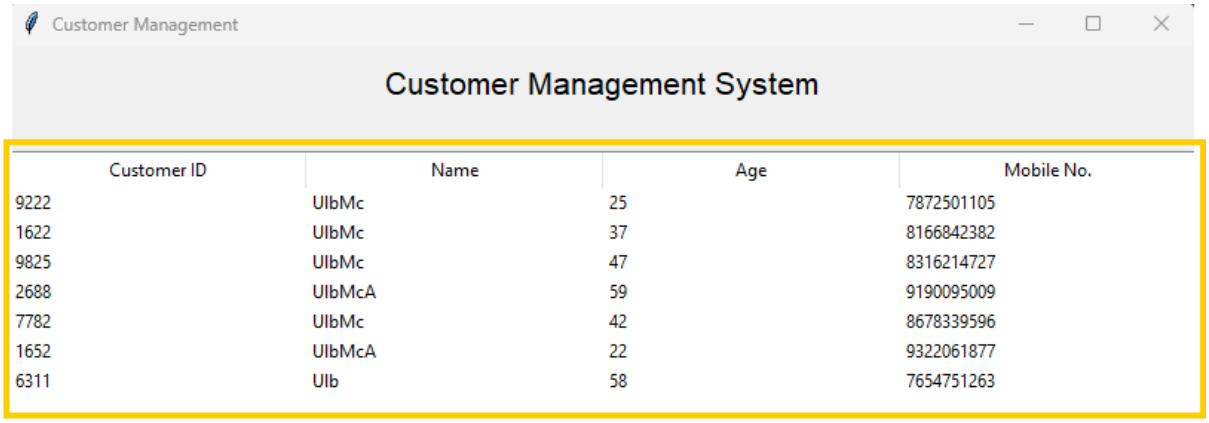
Output :

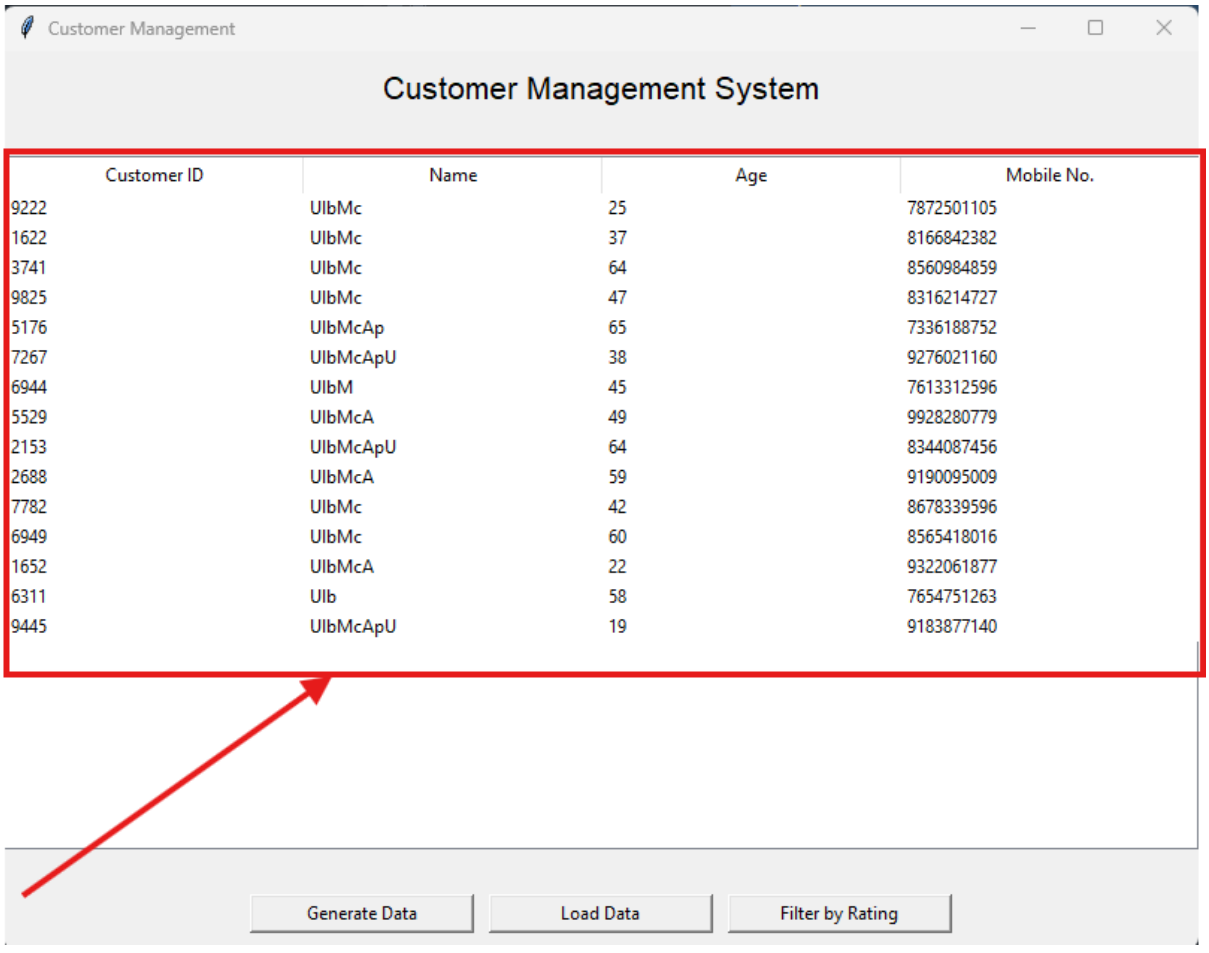

Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
Generate Data
Load Data
Filter by Rating
In [3]: runfile('C:/Users/Ashwarya/Desktop/oops mini project.py', wdir='C:/Users/Ashwarya/Desktop')
Customer data saved to 'customer_data.csv'
Customers with rating >= 3.5 (7):
ID: 4858, Name: tPcZVnAH, Rating: 3.8
ID: 6187, Name: tPcZVn, Rating: 3.6
ID: 4076, Name: tPcZVn, Rating: 4.0
ID: 4860, Name: tPc, Rating: 3.6
ID: 5037, Name: tPcZVn, Rating: 4.6
ID: 9466, Name: tPcZ, Rating: 3.7
ID: 3766, Name: tPcZ, Rating: 4.0


Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
Success
Customer data generated and saved to 'customer_data.csv'.
OK
Generate Data
Load Data
Filter by Rating


Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
9222 UlbMc 25 7872501105
1622 UlbMc 37 8166842382
9825 UlbMc 47 8316214727
2688 UlbMcA 59 9190095009
7782 UlbMc 42 8678339596
1652 UlbMcA 22 9322061877
6311 Ulb 58 7654751263
Filter Applied
Filtered 7 customers with rating >= 3.5.
OK
Generate Data
Load Data
Filter by Rating


Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
9222 UlbMc 25 7872501105
1622 UlbMc 37 8166842382
3741 UlbMc 64 8560984859
9825 UlbMc 47 8316214727
5176 UlbMcAp 65 7336188752
7267 UlbMcApU 38 9276021160
6944 UlbM 45 7613312596
5529 UlbMcA 49 9928280779
2153 UlbMcApU 64 8344087456
2688 UlbMcA 59 9190095009
7782 UlbMc 42 8678339596
6949 UlbMc 60 8565418016
1652 UlbMcA 22 9322061877
6311 Ulb 58 7654751263
9445 UlbMcApU 19 9183877140
Generate Data
Load Data
Filter by Rating


Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
9222 UlbMc 25 7872501105
1622 UlbMc 37 8166842382
9825 UlbMc 47 8316214727
2688 UlbMcA 59 9190095009
7782 UlbMc 42 8678339596
1652 UlbMcA 22 9322061877
6311 Ulb 58 7654751263
Generate Data
Load Data
Filter by Rating


Customer Management
Customer Management System
Customer ID
Name
Age
Mobile No.
9222 UlbMc 25 7872501105
1622 UlbMc 37 8166842382
3741 UlbMc 64 8560984859
9825 UlbMc 47 8316214727
5176 UlbMcAp 65 7336188752
7267 UlbMcApU 38 9276021160
6944 UlbM 45 7613312596
5529 UlbMcA 49 9928280779
2153 UlbMcApU 64 8344087456
2688 UlbMcA 59 9190095009
7782 UlbMc 42 8678339596
6949 UlbMc 60 8565418016
1652 UlbMcA 22 9322061877
6311 Ulb 58 7654751263
9445 UlbMcApU 19 9183877140
Generate Data
Load Data
Filter by Rating

Conclusion :

The Customer Rating App mini-project demonstrates the generation, storage, and analysis of customer feedback data using Python. The project simulates customer ratings, names, and demographic details to provide a practical framework to understand the dynamics of user feedback systems. Key concepts of Python programming incorporating random number generation, string manipulation, file handling, Pandas for data analysis, and object-oriented programming were effectively integrated.

The project developed was stored in a structured format to ensure smooth accessibility and manipulation as required in further analysis.