ONLINE SHOPPING MANAGEMENT SYSTEM

A MINI PROJECT REPORT

Submitted by

KRISHNAVARTHINI K H 220701136

MADHUPREETHA A 220701152

In partial fulfillment for the award of the degree of

BACHELOR OF

ENGINEERING IN

COMPUTER SCIENCE

RAJALAKSHMI ENGINEERING COLLEGE (AUTONOMOUS)

THANDALAM

CHENNAI-602105

2023-24

BONAFIDE CERTIFICATE

Certified that this project report "**ONLINE SHOPPING MANAGEMENT SYSTEM**" is the bonafide work of **"KRISHNAVARTHINI K H(220701136), MADHU PREETHA A(220701152)"**who carried out the project work under my supervision.

Submitted for the Practical Examination held on _____

SIGNATURE
Dr.R.SABITHA
Professor and II Year Academic Head,
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai - 602 105

SIGNATURE
Ms. D. KALPANA
Assistant Professor (SG),
Computer Science and Engineering,
Rajalakshmi Engineering College,
Thandalam, Chennai-602 105

ABSTRACT

The Online Shopping is a web based application intended for online retailers. The main objective of this application is to make it interactive and its ease of use. It would make searching, viewing and selection of a product easier. It contains a sophisticated search engine for user's to search for products specific to their needs. The search engine provides an easy and convenient way to search for products where a user can Search for a product interactively and the search engine would refine the products available based on the user’s input. The user can then view the complete specification of each product. They can also view the product reviews and also write their own reviews. The application also provides a drag and drop feature so that a user can add a product to the shopping cart by dragging the item in to the shopping cart. The main emphasis lies in providing a user-friendly search engine for effectively showing the desired results and its drag and drop behavior.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 INTRODUCTION

Online shopping is the process whereby consumers directly buy goods or services from a seller in real-time, without an intermediary service, over the Internet. It is a form of electronic commerce. This project is an attempt to provide the advantages of online shopping to customers of a real shop. It helps buying the products in the shop anywhere through internet by using an android device. Thus the customer will get the service of online shopping and home delivery from his favorite shop

1.2 OBJECTIVES

The Online Shopping Management System project aims to develop an efficient, user-friendly platform that seamlessly integrates product and order management using a graphical user interface (GUI) built with Tkinter and a MySQL database for data storage and retrieval. The primary objectives include creating an intuitive interface that allows users to view products, add new products, and check their orders easily. The system is designed to perform CRUD operations effectively, ensuring data integrity and security during database interactions. Key functionalities such as displaying product details and managing orders are implemented to provide a smooth user experience. Additionally, the project emphasizes the importance of secure database connections, input validation, and error handling to prevent security vulnerabilities. Scalability and maintainability are crucial, with the architecture designed to accommodate future feature expansions and updates. The system undergoes thorough testing to ensure all functionalities meet the specified requirements, and comprehensive documentation is provided to support users and developers. Overall, the project aims to enhance the shopping experience for users while streamlining administrative tasks for businesses.

1.3 MODULES

Database Connection Module (db_connection.py):

- Handles database connection setup.

Product Management Module (product_management.py):

- Contains functions to fetch and add products.

Order Management Module (order_management.py):

- Contains functions to fetch orders.

GUI Module (gui.py):

- Handles all GUI-related operations and integrates product and order management functionalities.

Main Module (main.py):

- Entry point for the application, initiates the GUI.

By organizing the code into these modules, you ensure that each part of the system is logically separated, making the codebase more maintainable and scalable.

2. SURVEY OF TECHNOLOGIES

2.1 SOFTWARE DESCRIPTION

This system enables users to manage products, customers, and orders effectively through a user-friendly interface. It consists of three main components: product management, customer management, and order management.

Product Management:

1.Users can add new products with details like name, category, price, and quantity.

2.Existing products can be viewed.

3.Products are displayed in a Treeview widget for easy navigation.

Customer Management:

1.Customers can be added with their name, email, and phone number.

2.Existing customers can be viewed.

3.Customer information is stored for order processing and tracking.

Order Management:

1.Orders can be created by specifying the product ID,

customer ID, order date, and quantity.

2.Users can view all orders placed.

3.Order details are essential for tracking sales and managing inventory.

Graphical User Interface (GUI):

1.The GUI is divided into frames for organizing different sections (product, customer, order).

2.Entry widgets are used to input data for products, customers, and orders.

3.Buttons trigger actions such as adding, viewing, and closing operations.

4.Labels

provide context for the corresponding entry fields.

2.2 LANGUAGES

1) MYSQL

2)PYTHON

3.REQUIREMENTS AND ANALYSIS

3.1 Requirement Specification

1. Functional Requirements

User Interface:

1. The system must provide a graphical user interface (GUI) using Tkinter.

2.The main window must display buttons for viewing products, creating new products, and viewing orders.

Product Management:

1.The system must allow users to view all available products in a new window.

#Each product must display its name and price.

2.The system must allow users to add new products.

#Users must be able to enter the product name and price through input fields.

#The system must validate the inputs and add the new product to the database.

#A confirmation message must be displayed upon successful addition of a product.

Order Management:

1.The system must allow users to view all orders in a new window.

#Each order must display relevant details such as order name and price.

Database Operations:

1.The system must connect to a MySQL database named "shopping_system."

2.The database must have a "products" table with fields for product ID, name, and price.

3.The database must have an "orders" table with fields for order ID, name, and price.

4.The system must be able to execute SQL queries to fetch and insert data.

2. Non-Functional Requirements:

Performance:

1.The system must fetch and display data from the database within a reasonable time frame.

2.The system must handle multiple product and order entries efficiently.

Usability:

1.The user interface must be intuitive and easy to navigate.

2.The input fields must be clearly labeled and provide prompt feedback on user actions.

Reliability:

1.The system must ensure data integrity during database operations.

2.The system must handle errors gracefully and provide meaningful error messages.

Security:

1.The system must secure database connection credentials.

2.Input validation must be implemented to prevent SQL injection attacks.

Scalability:

1.The system architecture must support easy addition of new features and functionalities.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

Hardware Requirements:

Minimum Hardware:

1.Processor: 1 GHz or faster

2.RAM: 2 GB or more

3.Hard Disk: At least 500 MB of free space

Software Requirements:

Development Environment:

1.Python 3.8 or higher

2.Tkinter library (included with standard Python installations)

3.mysql-connector-python library

Database:

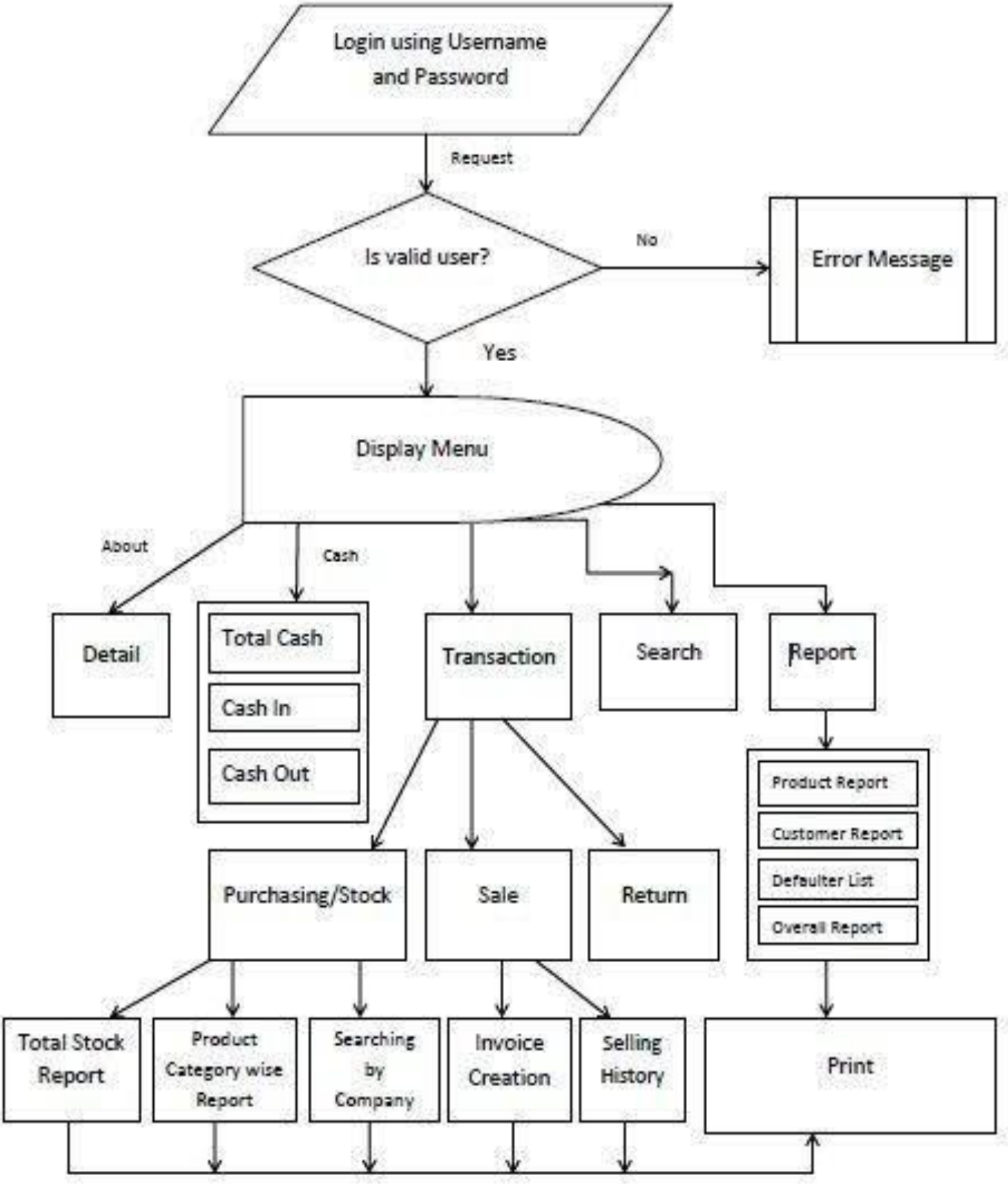
1.MySQL Server

2.MySQL Workbench (optional, for database management)

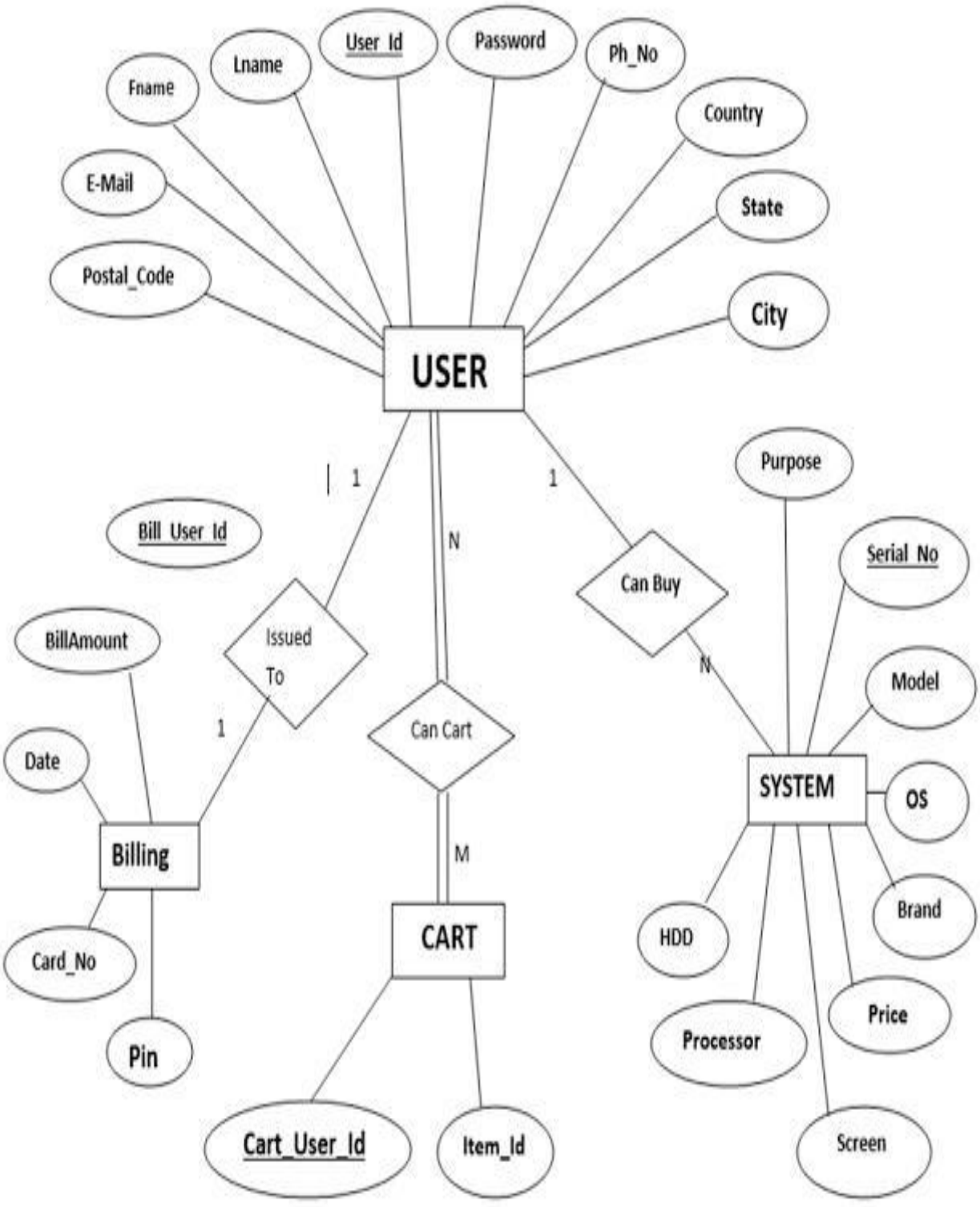
Operating System:

1.Windows, macOS, or Linux

3.3 ARCHITECTURE DIAGRAM



3.4 ER DIAGRAM



4. PROGRAM CODE

```
import mysql.connector

from tkinter import *

from tkinter import messagebox, ttk

from datetime import datetime  # Import datetime module


# Database operations

def connect():

    conn = mysql.connector.connect(

        host="localhost",

        user="root",  # replace with your MySQL username

        password="your_password",  # replace with your MySQL
password

        database="your_database"  # replace with your MySQL
database

    )

    cur = conn.cursor()

    cur.execute("""

        CREATE TABLE IF NOT EXISTS products (

            id INT AUTO_INCREMENT PRIMARY KEY,

            name VARCHAR(255),

            category VARCHAR(255),

            price DECIMAL(10, 2),

            quantity INT

        )

    """)

    cur.execute("""
```

```
        CREATE TABLE IF NOT EXISTS customers (
            id INT AUTO_INCREMENT PRIMARY KEY,
            name VARCHAR(255),
            email VARCHAR(255),
            phone VARCHAR(15)
        )
    """)
    cur.execute("""
        CREATE TABLE IF NOT EXISTS orders (
            id INT AUTO_INCREMENT PRIMARY KEY,
            product_id INT,
            customer_id INT,
            order_date DATE,
            quantity INT,
            FOREIGN KEY (product_id) REFERENCES products(id),
            FOREIGN KEY (customer_id) REFERENCES customers(id)
        )
    """)

    conn.commit()
    conn.close()

# Product operations
def insert_product(name, category, price, quantity):
    if not price.replace('.', '', 1).isdigit() or not quantity.isdigit():
        messagebox.showerror("Input Error", "Price must be a number
```

```
and Quantity must be an integer")
        return


    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password",
        database="your_database"
    )
    cur = conn.cursor()
    cur.execute("INSERT INTO products (name, category, price,
quantity) VALUES (%s, %s, %s, %s)", (name, category, price,
quantity))
    conn.commit()
    conn.close()
    view_command("products")


def view_products():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password",
        database="your_database"
    )
    cur = conn.cursor()
    cur.execute("SELECT * FROM products")
```

```
    rows = cur.fetchall()
    conn.close()
    return rows

# Customer operations
def insert_customer(name, email, phone):
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password",
        database="your_database"
    )
    cur = conn.cursor()
    cur.execute("INSERT INTO customers (name, email, phone)
VALUES (%s, %s, %s)", (name, email, phone))
    conn.commit()
    conn.close()
    view_command("customers")

def view_customers():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password”
        database="your_database"
    )
```

```
    cur = conn.cursor()
    cur.execute("SELECT * FROM customers")
    rows = cur.fetchall()
    conn.close()
    return rows

# Order operations
def insert_order(product_id, customer_id, order_date, quantity):
    if not quantity.isdigit():
        messagebox.showerror("Input Error", "Quantity must be an
integer")
        return

    try:
        # Validate the date format
        datetime.strptime(order_date, '%Y-%m-%d')
    except ValueError:
        messagebox.showerror("Input Error", "Order Date must be in
YYYY-MM-DD format")
        return

    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password",
        database="your_database"
```

```
    )
    cur = conn.cursor()
    cur.execute("INSERT INTO orders (product_id, customer_id, order_date, quantity) VALUES (%s, %s, %s, %s)", (product_id, customer_id, order_date, quantity))
    conn.commit()
    conn.close()
    view_command("orders")

def view_orders():
    conn = mysql.connector.connect(
        host="localhost",
        user="root",
        password="your_password",
        database="your_database"
    )
    cur = conn.cursor()
    cur.execute("SELECT * FROM orders")
    rows = cur.fetchall()
    conn.close()
    return rows

# GUI functions
def get_selected_row(event):
    global selected_tuple
    try:
```

```
        if list1.selection():
            item = list1.selection()[0]
            selected_tuple = list1.item(item, 'values')
            e1.delete(0, END)
            e1.insert(END, selected_tuple[1])
            e2.delete(0, END)
            e2.insert(END, selected_tuple[2])
            e3.delete(0, END)
            e3.insert(END, selected_tuple[3])
            e4.delete(0, END)
            e4.insert(END, selected_tuple[4])
    except IndexError:
        pass


def view_command(table):
    # Clear previous data
    list1.delete(*list1.get_children())


    # Insert attribute names as the first row based on the table
    if table == "products":
        list1.insert('', 'end', values=("ID", "NAME", "CATEGORY",
"PRICE", "QUANTITY"))
        for row in view_products():
            list1.insert('', 'end', values=row)
    elif table == "customers":
        list1.insert('', 'end', values=("ID", "NAME", "EMAIL",
```

```
"PHONE"))
        for row in view_customers():
            list1.insert('', 'end', values=row)
    elif table == "orders":
        list1.insert('', 'end', values=("ID", "PRODUCT_ID",
"CUSTOMER_ID", "ORDER_DATE", "QUANTITY"))
        for row in view_orders():
            list1.insert('', 'end', values=row)

# Adding and searching functionalities are similar for all tables

connect()

# Create window object
window = Tk()
window.wm_title("Online Shopping Management System")
window.configure(bg='lightblue')

# Create frames for better organization and color coding
frame1 = Frame(window, bg='lightblue')
frame1.grid(row=0, column=0, padx=10, pady=10)

frame2 = Frame(window, bg='lightgreen')
frame2.grid(row=0, column=1, padx=10, pady=10)

frame3 = Frame(window, bg='lightcoral')
```

```
frame3.grid(row=0, column=2, padx=10, pady=10)

# Labels for product table
l1 = Label(frame1, text="Product Name", bg='lightblue')
l1.grid(row=0, column=0)
l2 = Label(frame1, text="Category", bg='lightblue')
l2.grid(row=1, column=0)
l3 = Label(frame1, text="Price", bg='lightblue')
l3.grid(row=2, column=0)
l4 = Label(frame1, text="Quantity", bg='lightblue')
l4.grid(row=3, column=0)

# Entries for product table
name_text = StringVar()
e1 = Entry(frame1, textvariable=name_text)
e1.grid(row=0, column=1)
category_text = StringVar()
e2 = Entry(frame1, textvariable=category_text)
e2.grid(row=1, column=1)
price_text = StringVar()
e3 = Entry(frame1, textvariable=price_text)
e3.grid(row=2, column=1)
quantity_text = StringVar()
e4 = Entry(frame1, textvariable=quantity_text)
e4.grid(row=3, column=1)
```

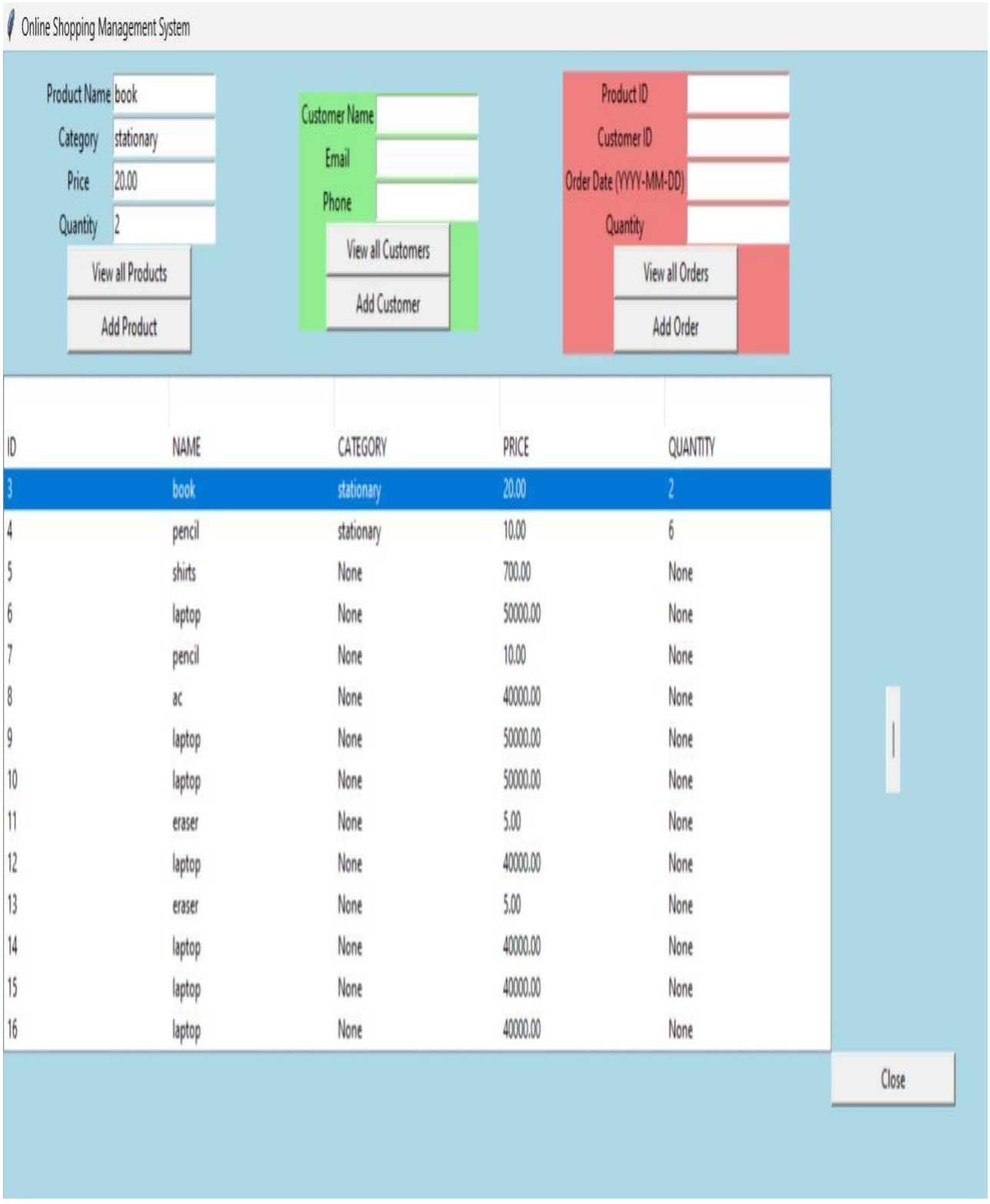
```
# Listbox -> Treeview for product table

list1 = ttk.Treeview(window, columns=("ID", "NAME",
"CATEGORY", "PRICE", "QUANTITY"), show="headings",
height=15)

list1.grid(row=1, column=0, columnspan=3)

# Scrollbar

sb1 = Scrollbar(window)

sb1.grid(row=1, column=3, rowspan=6)

list1.configure(yscrollcommand=sb1.set)

sb1.configure(command=list1.yview)


list1.bind('<<TreeviewSelect>>', get_selected_row)


# Buttons for product table

b1 = Button(frame1, text="View all Products", width=20,
command=lambda: view_command("products"))

b1.grid(row=4, column=0, columnspan=2)

b2 = Button(frame1, text="Add Product", width=20,
command=lambda: insert_product(name_text.get(),
category_text.get(), price_text.get(), quantity_text.get()))

b2.grid(row=5, column=0, columnspan=2)


# Labels for customer table

l5 = Label(frame2, text="Customer Name", bg='lightgreen')

l5.grid(row=0, column=0)

l6 = Label(frame2, text="Email", bg='lightgreen')

l6.grid(row=1, column=0)

l7 = Label(frame2, text="Phone", bg='lightgreen')
```

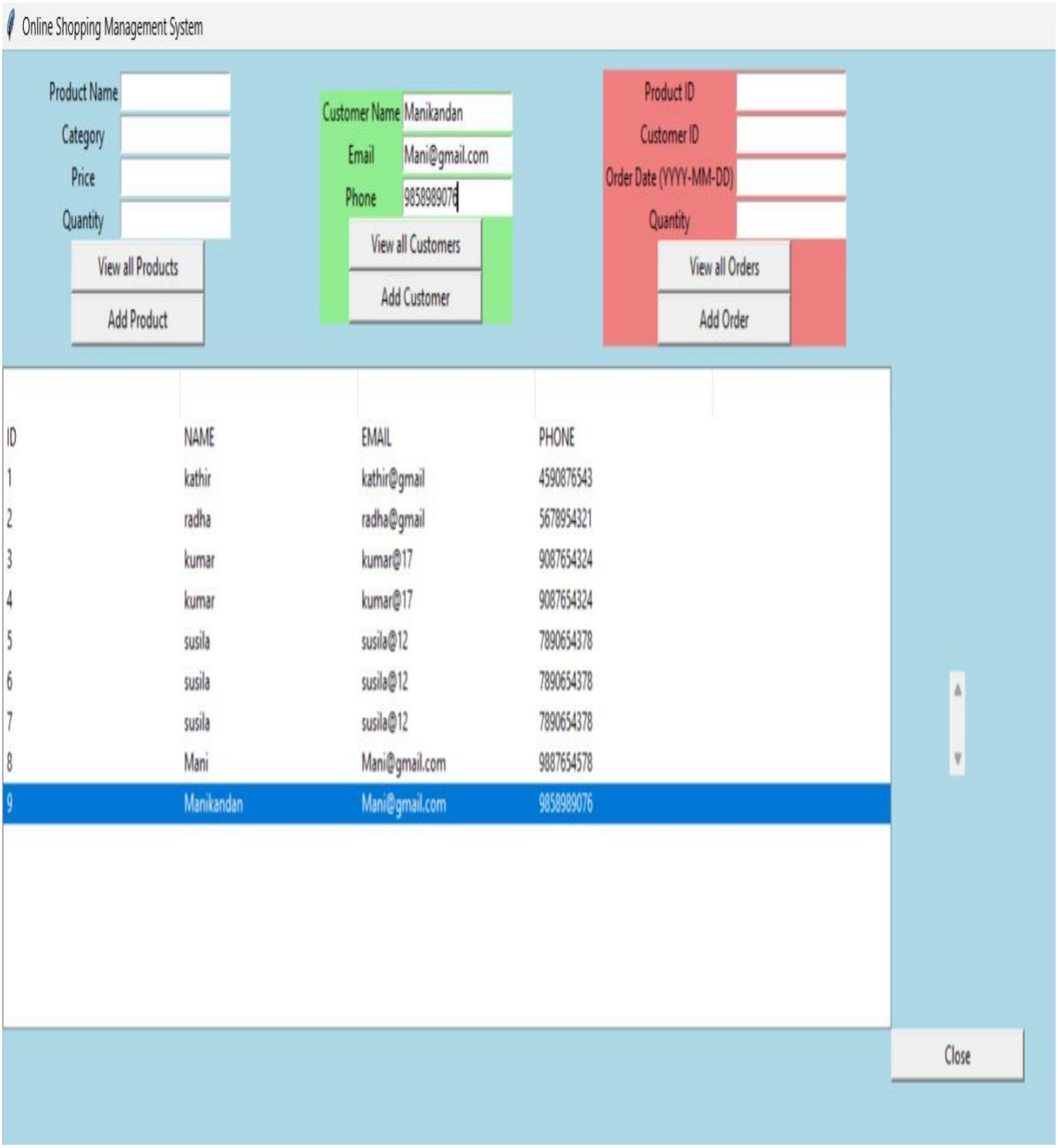
```
l7.grid(row=2, column=0)

# Entries for customer table
customer_name_text = StringVar()
e5 = Entry(frame2, textvariable=customer_name_text)
e5.grid(row=0, column=1)
email_text = StringVar()
e6 = Entry(frame2, textvariable=email_text)
e6.grid(row=1, column=1)
phone_text = StringVar()
e7 = Entry(frame2, textvariable=phone_text)
e7.grid(row=2, column=1)

# Buttons for customer table
b3 = Button(frame2, text="View all Customers", width=20,
command=lambda: view_command("customers"))
b3.grid(row=3, column=0, columnspan=2)
b4 = Button(frame2, text="Add Customer", width=20,
command=lambda: insert_customer(customer_name_text.get(),
email_text.get(), phone_text.get()))
b4.grid(row=4, column=0, columnspan=2)

# Labels for order table
l8 = Label(frame3, text="Product ID", bg='lightcoral')
l8.grid(row=0, column=0)
l9 = Label(frame3, text="Customer ID", bg='lightcoral')
l9.grid(row=1, column=0)
```

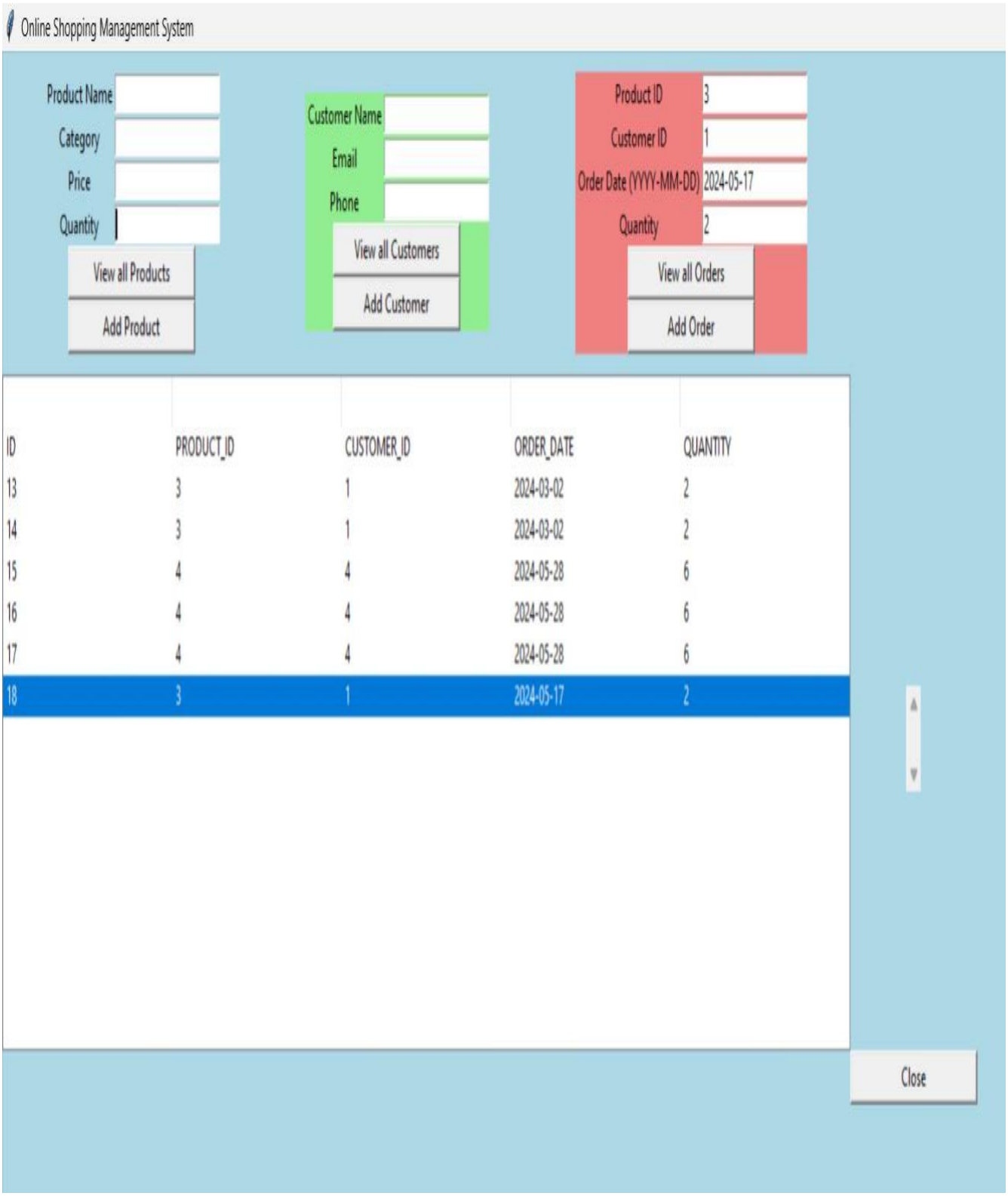
```
l10 = Label(frame3, text="Order Date (YYYY-MM-DD)",
bg='lightcoral')

l10.grid(row=2, column=0)

l11 = Label(frame3, text="Quantity", bg='lightcoral')

l11.grid(row=3, column=0)


# Entries for order table

product_id_text = StringVar()

e8 = Entry(frame3, textvariable=product_id_text)

e8.grid(row=0, column=1)

customer_id_text = StringVar()

e9 = Entry(frame3, textvariable=customer_id_text)

e9.grid(row=1, column=1)

order_date_text = StringVar()

e10 = Entry(frame3, textvariable=order_date_text)

e10.grid(row=2, column=1)

order_quantity_text = StringVar()

e11 = Entry(frame3, textvariable=order_quantity_text)

e11.grid(row=3, column=1)


# Buttons for order table

b5 = Button(frame3, text="View all Orders", width=20,
command=lambda: view_command("orders"))

b5.grid(row=4, column=0, columnspan=2)

b6 = Button(frame3, text="Add Order", width=20,
command=lambda: insert_order(product_id_text.get(),
customer_id_text.get(), order_date_text.get(),
order_quantity_text.get()))
```

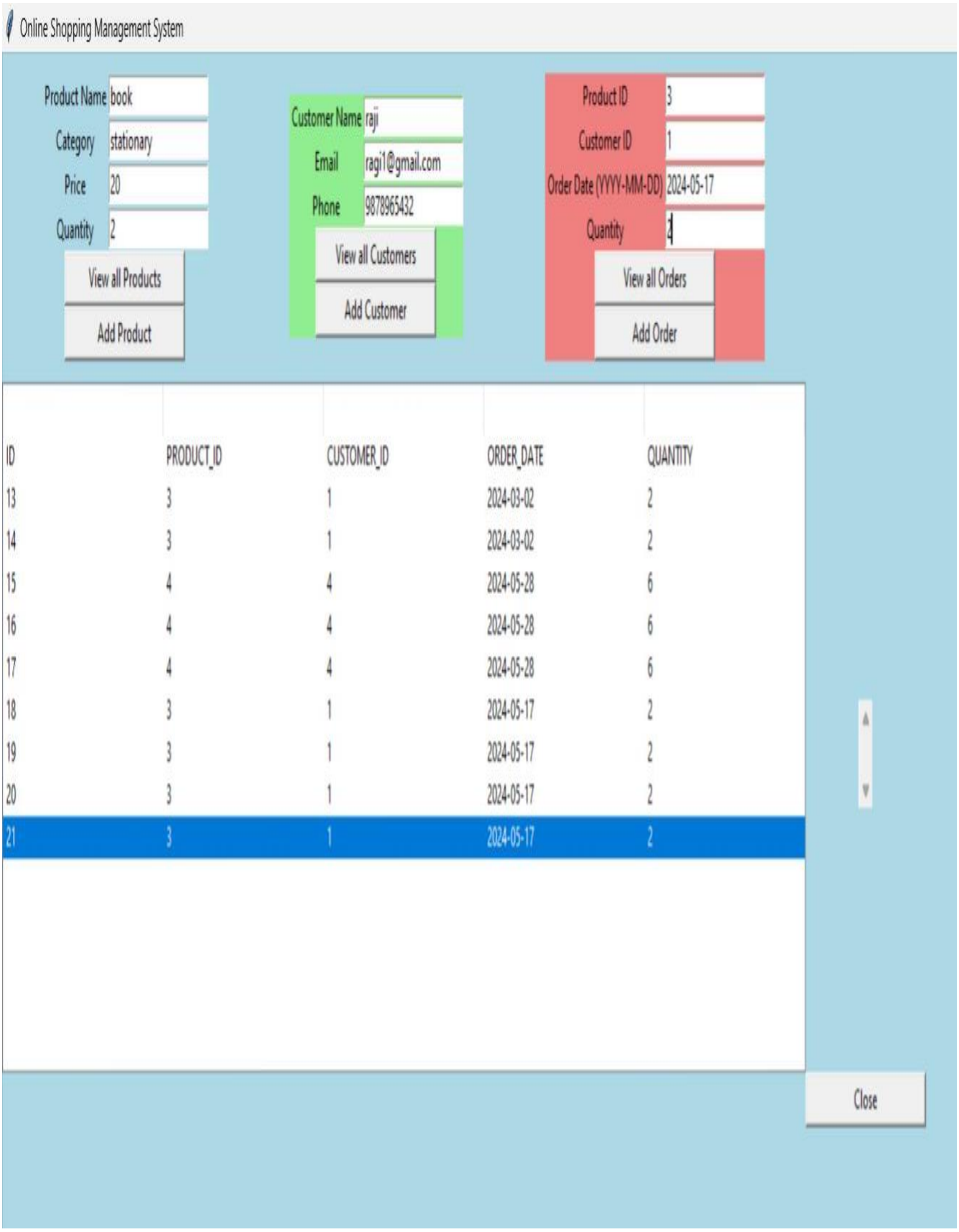
```
b6.grid(row=5, column=0, columnspan=2)


b7 = Button(window, text="Close", width=20,
command=window.destroy)

b7.grid(row=2, column=3)


window.mainloop()
```

5. RESULTS AND DISCUSSION


Online Shopping Management System
Product Name book
Category stationary
Price 20.00
Quantity 2
View all Products
Add Product
Customer Name
Email
Phone
View all Customers
Add Customer
Product ID
Customer ID
Order Date (YYYY-MM-DD)
Quantity
View all Orders
Add Order
ID NAME CATEGORY PRICE QUANTITY
3 book stationary 20.00 2
4 pencil stationary 10.00 6
5 shirts None 700.00 None
6 laptop None 50000.00 None
7 pencil None 10.00 None
8 ac None 40000.00 None
9 laptop None 50000.00 None
10 laptop None 50000.00 None
11 eraser None 5.00 None
12 laptop None 40000.00 None
13 eraser None 5.00 None
14 laptop None 40000.00 None
15 laptop None 40000.00 None
16 laptop None 40000.00 None
Close


Online Shopping Management System
Product Name
Category
Price
Quantity
View all Products
Add Product
Customer Name
Manikandan
Email
Mani@gmail.com
Phone
9858989076
View all Customers
Add Customer
Product ID
Customer ID
Order Date (YYYY-MM-DD)
Quantity
View all Orders
Add Order
ID NAME EMAIL PHONE
1 kathir kathir@gmail 4590876543
2 radha radha@gmail 5678954321
3 kumar kumar@17 9087654324
4 kumar kumar@17 9087654324
5 susila susila@12 7890654378
6 susila susila@12 7890654378
7 susila susila@12 7890654378
8 Mani Mani@gmail.com 9887654578
9 Manikandan Mani@gmail.com 9858989076
Close


Online Shopping Management System
Product Name
Category
Price
Quantity
View all Products
Add Product
Customer Name
Email
Phone
View all Customers
Add Customer
Product ID
3
Customer ID
1
Order Date (YYYY-MM-DD)
2024-05-17
Quantity
2
View all Orders
Add Order
ID
PRODUCT_ID
CUSTOMER_ID
ORDER_DATE
QUANTITY
13
3
1
2024-03-02
2
14
3
1
2024-03-02
2
15
4
4
2024-05-28
6
16
4
4
2024-05-28
6
17
4
4
2024-05-28
6
18
3
1
2024-05-17
2
Close


Online Shopping Management System
Product Name book
Category stationary
Price 20
Quantity 2
View all Products
Add Product
Customer Name raji
Email ragi1@gmail.com
Phone 9878965432
View all Customers
Add Customer
Product ID 3
Customer ID 1
Order Date (YYYY-MM-DD) 2024-05-17
Quantity 4
View all Orders
Add Order
ID PRODUCT_ID CUSTOMER_ID ORDER_DATE QUANTITY
13 3 1 2024-03-02 2
14 3 1 2024-03-02 2
15 4 4 2024-05-28 6
16 4 4 2024-05-28 6
17 4 4 2024-05-28 6
18 3 1 2024-05-17 2
19 3 1 2024-05-17 2
20 3 1 2024-05-17 2
21 3 1 2024-05-17 2
Close

6. CONCLUSION:

The implementation of an online shopping management system significantly enhances the efficiency and convenience of the shopping experience for both customers and retailers. By streamlining inventory management, order processing, and customer service, it reduces operational costs and errors, while increasing customer satisfaction through personalized and seamless interactions. Additionally, the system's data analytics capabilities provide valuable insights into consumer behavior and market trends, enabling businesses to make informed decisions and stay competitive. Overall, an online shopping management system is a crucial tool for modern retail operations, driving growth and improving the overall customer experience.

7. REFERENCES:

Books:

Turban, E., King, D., Lee, J. K., Liang, T. P., & Turban, D. C. (2015). Electronic Commerce: A Managerial and Social Networks Perspective. Springer.

Laudon, K. C., & Traver, C. G. (2020). E-commerce 2020: Business, Technology and Society. Pearson.

• Wikipedia for various diagrams & testing methods

http://www.wikipedia.org/

• Cool text for Images and Buttons

http://cooltext.com/

• K-State Research Exchange for samples in report writing

http://krex.k-state.edu/dspace/handle/2097/959

• Smart Draw for drawing all the Diagrams used in this report.

http://www.smartdraw.com/

• Sample Ecommerce Application

http://www.NewEgg.com