DESKTOP E-COMMERCE SOLUTION

A MINI-PROJECT REPORT

Submitted by

ANUPRIYA P 241901005

in partial fulfillment of the award of the degree

of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(CYBER SECURITY)

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project **"DESKTOP E-COMMERCE SOLUTION"** is the bonafide work of **" ANUPRIYA P"** who carried out the project work under my supervision.

SIGNATURE

MS.M. FOWZIA SIHANA

ASSISTANT PROFESSOR SS

Dept. of Computer Science and Engineering (Cyber Security)
Rajalakshmi Engineering College
Chennai

This mini project report is submitted for the viva voce examination to be held on

_____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

Abstract

This project presents a desktop-based sports item management system developed using CustomTkinter in Python. The application is designed to resemble a modern website while functioning as a standalone desktop app, offering a simple and attractive interface for managing sports products and related records. The system includes key features such as Add Product, View Product, Search Product, View Users, and View Orders, enabling efficient handling of product details, user information, and order data. The backend is connected to a MySQL database using Python's mysql.connector, allowing real-time data operations like insertion, retrieval, updating, and deletion. This direct connection ensures smooth performance without the need for a web server. By integrating CustomTkinter for the frontend and MySQL for the backend, the project achieves an effective blend of usability, functionality, and visual appeal. Overall, it demonstrates how Python can be used to create interactive, database-driven desktop applications for practical business purposes such as sports inventory management.

ACKNOWLEDGEMENT

I express my sincere thanks to our beloved and honorable chairman **MR. S. MEGANATHAN** and the chairperson **DR. M.THANGAM MEGANATHAN** for their timely support and encouragement.

I am greatly indebted to our respected and honorable principal **Dr. S.N. MURUGESAN** for his able support and guidance.

No words of gratitude will suffice for the unquestioning support extended by my Head Of The Department **Dr.BENEDICT JAYAPRAKASH NICHOLAS** for being ever supporting force during my project work .

I also extend my sincere and hearty thanks to my internal guide **MS.FOWZIA SIHANA,** for her valuable guidance and motivation during the completion of this project.

My sincere thanks to my family members, friends and other staff members of Computer Science and Engineering (Cyber Security).

ANUPRIYA P .

TABLE OF CONTENTS

CHAPTER NO.		TITLE	PAGE NO.
1.		ABSTRACT	3
2.		INTRODUCTION	6
	2.1	INTRODUCTION	6
	2.2	SCOPE OF THE WORK	6
	2.3	PROBLEM STATEMENT	6
	2.4	AIM AND OBJECTIVES OF THE PROJECT	6
3.		SYSTEM SPECIFICATIONS	7
	3.1	HARDWARE SPECIFICATIONS	7
	3.2	SOFTWARE SPECIFICATIONS	7
4.		MODULE DESCRIPTION	8
5.		CODING	10
6.		SCREENSHOTS	28
7.		CONCLUSION AND FUTURE ENHANCEMENT	34

INTRODUCTION:

E-commerce means buying and selling goods or services online. It allows customers to shop anytime and helps businesses reach more people with lower costs. This project focuses on creating an online shopping platform that is easy to use and secure for both buyers and sellers.

SCOPE OF THE WORK:

The project includes developing a website where users can register, browse products, add them to the cart, make payments, and track orders. An admin panel will help manage products, orders, and user data efficiently.

PROBLEM STATEMENT:

Traditional shopping is limited by location and time. Customers may not always find what they need, and businesses face challenges in reaching more customers. Hence, there is a need for an online platform that connects buyers and sellers easily and safely.

AIM:

To design and develop a secure and user-friendly e-commerce website for online shopping.

OBJECTIVES:

1. Create a simple interface for users to shop online.
2. Provide secure payment options.
3. Manage orders and inventory through an admin panel.
4. Protect user data and ensure privacy.
5. Allow users to track their orders easily

1. SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	INTEL i5
Memory Size	8GB RAM

2.2 SOFTWARE SPECIFICATIONS

Operating System	WINDOWS 11
Front-End	Python
Back-End	MySQL
Language	Python, SQL

3. MODULE DESCRIPTION

1. Login Module

The Login Module is the entry point of the system. It allows authorized users, such as administrators, to securely access the application by providing valid login credentials. The system verifies the username and password with stored records in the database. If the details match, the user is redirected to the Dashboard; otherwise, an error message is displayed. This module ensures that only authenticated users can manage products, users, and orders.

2. Dashboard Module

The Dashboard serves as the central control panel for navigating the system. After successful login, users are directed to this interface, where quick access buttons or menus are provided for different features such as adding products, viewing product lists, searching products, viewing users, and managing orders. It offers a clear overview and improves usability by organizing the system's main functionalities in one place.

3. Add Product Module

The Add Product Module allows the admin to insert new products into the system database. The page contains input fields for product details such as product name, category (e.g., bakery or sports items), price, description, and product image. Once submitted, the product is saved and becomes available for display on the View Products Page. This module enables efficient product management and updates.

4. View Products Module

This module displays the list of all products stored in the system. Products are shown in a structured tabular or card format for clarity. Each product entry typically includes

its name, category, price, and image. Additional options such as Edit or Delete (if included in your implementation) may also be available. This module helps the admin easily monitor and manage the inventory.

5. Search Product Module

The Search Module allows the admin or user to search for specific products based on keywords such as product name or category. The system filters the product list and displays only the matches, helping users find products quickly and efficiently. This improves user navigation and reduces browsing time.

6. View Users Module

The View Users Module displays the list of all registered users in the system. It allows the admin to oversee customer details such as username, contact information, or user activity (depending on stored data). This module supports user management and helps maintain effective administrative control over system access and customer records.

7. Orders Module

The Orders Module manages and displays customer orders. Each order may include details such as user information, product details, order quantity, total amount, and order status. This module enables the admin to track and process orders efficiently and ensures systematic order handling within the e-commerce system.

4. CODING

The above code denotes the full backend which is implemented in our Ecommerce desktop solutions project. This uses mysql and python to implement add product, search product, view orders, view users, update product and delete product.

BACKEND

```
import mysql.connector
#DATABASE CONNECTION
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="12345678",   # <-- use your own password
    database="sports_ecommerce"
)
cursor = conn.cursor()


# PRODUCT FUNCTIONS
def insert_product():
    try:
        name = input("Enter product name: ")
        category = input("Enter product category: ")
        price = float(input("Enter product price (without ₹): "))
        stock = int(input("Enter stock quantity: "))
        description = input("Enter product description: ")

        query = """
        INSERT INTO products (name, category, price, stock, description)
        VALUES (%s, %s, %s, %s, %s)
```

```
        """
        cursor.execute(query, (name, category, price, stock, description))
        conn.commit()
        print("✅ Product added successfully!")
    except Exception as e:
        print("❌ Error inserting product:", e)


def view_products():
    try:
        cursor.execute("SELECT product_id, name, category, price, stock, description
FROM products")
        results = cursor.fetchall()
        if results:
            print("\n------ 🏋️ Product List ------")
            for row in results:
                print(f"ID: {row[0]} | Name: {row[1]} | Category: {row[2]} | Price:
₹{row[3]} | Stock: {row[4]} | Desc: {row[5]}")
            print("--------------------------\n")
        else:
            print("No products found.")
    except Exception as e:
        print("❌ Error displaying products:", e)


def update_product():
    try:
        product_id = int(input("Enter Product ID to update: "))
```

```
    print("\nChoose the field to update:")
    print("1. Name")
    print("2. Category")
    print("3. Price")
    print("4. Stock")
    print("5. Description")

    choice = int(input("Enter your choice (1-5): "))
    fields = ["name", "category", "price", "stock", "description"]

    if 1 <= choice <= 5:
        new_value = input("Enter new value: ")
        if fields[choice - 1] == "price":
            new_value = float(new_value)
        elif fields[choice - 1] == "stock":
            new_value = int(new_value)

        query = f"UPDATE products SET {fields[choice - 1]} = %s WHERE
product_id = %s"
        cursor.execute(query, (new_value, product_id))
        conn.commit()

        if cursor.rowcount > 0:
            print("✅ Product updated successfully!")
        else:
            print("❌ No product found with that ID.")
    else:
        print("❌ Invalid choice.")
```

```
    except Exception as e:
        print("❌ Error updating product:", e)


def delete_product():
    try:
        product_id = int(input("Enter Product ID to delete: "))
        cursor.execute("DELETE FROM products WHERE product_id = %s",
(product_id,))
        conn.commit()

        if cursor.rowcount > 0:
            print("🗑️ Product deleted successfully!")
        else:
            print("❌ No product found with that ID.")
    except Exception as e:
        print("❌ Error deleting product:", e)


def search_product():
    try:
        keyword = input("Enter product name or category to search: ")
        query = "SELECT * FROM products WHERE name LIKE %s OR category
LIKE %s"
        cursor.execute(query, (f"%{keyword}%", f"%{keyword}%"))
        results = cursor.fetchall()

        if results:
```

```
        print("\n🔍 Search Results:")
        for row in results:
            print(f"ID: {row[0]} | Name: {row[1]} | Category: {row[2]} | Price: ₹{row[3]} | Stock: {row[4]} | Desc: {row[5]}")
    else:
        print("❌ No matching products found.")
  except Exception as e:
    print("❌ Error searching for product:", e)


# USER FUNCTIONS
def view_users():
  try:
    cursor.execute("SELECT user_id, name, email, phone, created_at FROM users")
    results = cursor.fetchall()
    if results:
        print("\n👤 USERS LIST:")
        for row in results:
            print(f"ID: {row[0]} | Name: {row[1]} | Email: {row[2]} | Phone: {row[3]} | Joined: {row[4]}")
    else:
        print("No users found.")
  except Exception as e:
    print("❌ Error fetching users:", e)
# ORDER FUNCTIONS
def view_orders():
  try:
```

```
        query = """
        SELECT o.order_id, u.name AS user_name, o.order_date, o.total_amount, o.status
        FROM orders o
        LEFT JOIN users u ON o.user_id = u.user_id
        ORDER BY o.order_date DESC
        """
        cursor.execute(query)
        results = cursor.fetchall()
        if results:
            print("\n📦 ORDERS LIST:")
            for row in results:
                print(f"Order ID: {row[0]} | Customer: {row[1]} | Date: {row[2]} | Amount: ₹{row[3]} | Status: {row[4]}")
        else:
            print("No orders found.")
    except Exception as e:
        print("❌ Error fetching orders:", e)

def view_order_items():
    try:
        order_id = int(input("Enter Order ID to view items: "))
        query = """
        SELECT p.name, oi.quantity, oi.unit_price
        FROM order_items oi
        JOIN products p ON oi.product_id = p.product_id
        WHERE oi.order_id = %s
        """
```

```
        cursor.execute(query, (order_id,))
        results = cursor.fetchall()
        if results:
            print(f"\n🛒 Items in Order {order_id}:")
            for row in results:
                print(f"- {row[0]} (x{row[1]}) - ₹{row[2]}")
        else:
            print("No items found for this order.")
    except Exception as e:
        print("❌ Error fetching order items:", e)
# MAIN MENU
def main():
    while True:
        print("\n===== 🏀 SPORTS E-COMMERCE BACKEND =====")
        print("1. Insert Product")
        print("2. View Products")
        print("3. Update Product")
        print("4. Delete Product")
        print("5. Search Product")
        print("6. View Users")
        print("7. View Orders")
        print("8. View Order Items")
        print("9. Exit")
       choice = input("Enter your choice (1-9): ")
        if choice == '1':
            insert_product()
        elif choice == '2':
            view_products()
```

```
        elif choice == '3':
            update_product()
        elif choice == '4':
            delete_product()
        elif choice == '5':
            search_product()
        elif choice == '6':
            view_users()
        elif choice == '7':
            view_orders()
        elif choice == '8':
            view_order_items()
        elif choice == '9':
            print("👋 Exiting... Goodbye!")
            break
        else:
            print("❌ Invalid choice. Please try again.")
# RUN PROGRAM
if __name__ == "__main__":
    main()
cursor.close()
conn.close()
```

FRONTEND

```
import mysql.connector
import time
import threading

# DATABASE CONNECTION
def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="12345678",   # change if needed
        database="sports_ecommerce"
    )

#APP SETUP
ctk.set_appearance_mode("light")
ctk.set_default_color_theme("blue")

app = ctk.CTk()
app.title("Sports E-Commerce 🏀")
app.geometry("1100x700")
app.resizable(False, False)

# HELPER FUNCTIONS
def clear_frame():
    for widget in main_frame.winfo_children():
        widget.destroy()

def back_to_home():
```

```
    clear_frame()
    create_dashboard()

def display_table(parent, headers, rows):
    frame = ctk.CTkScrollableFrame(parent, fg_color="white")
    frame.pack(pady=10, padx=20, fill="both", expand=True)

    for i, header in enumerate(headers):
        lbl = ctk.CTkLabel(frame, text=header, font=("Helvetica", 15, "bold"),
text_color="#005f73")
        lbl.grid(row=0, column=i, padx=15, pady=10)

    for r, row in enumerate(rows, start=1):
        for c, val in enumerate(row):
            lbl = ctk.CTkLabel(frame, text=val, font=("Helvetica", 13))
            lbl.grid(row=r, column=c, padx=15, pady=6)
```

#LOGIN PAGE

```
def login_page():
    clear_frame()
    ctk.CTkLabel(main_frame, text="🏀 SPORTS E-COMMERCE LOGIN",
font=("Helvetica", 26, "bold"), text_color="#005f73").pack(pady=30)

    frame = ctk.CTkFrame(main_frame, width=400, height=250, corner_radius=15)
    frame.pack(pady=20)

    username_entry = ctk.CTkEntry(frame, width=250,
placeholder_text="Username")
```

```
    password_entry = ctk.CTkEntry(frame, width=250, placeholder_text="Password",
show="*")
    username_entry.pack(pady=15)
    password_entry.pack(pady=10)

    def login_action():
        username = username_entry.get()
        password = password_entry.get()
        if username == "admin" and password == "admin123":
            messagebox.showinfo("Login Successful", "Welcome, Admin!")
            clear_frame()
            create_dashboard()
        else:
            messagebox.showerror("Invalid", "Incorrect username or password.")

    ctk.CTkButton(frame,          text="Login",          fg_color="#0a9396",
command=login_action).pack(pady=20)
```

#SPLASH SCREEN

```
def splash_screen():
    splash = ctk.CTkToplevel(app)
    splash.geometry("500x300+350+200")
    splash.title("Welcome 🏆 ")
    splash.configure(fg_color="white")
    splash.resizable(False, False)

    ctk.CTkLabel(splash, text="SPORTS E-COMMERCE", font=("Helvetica", 28,
"bold"), text_color="#0a9396").pack(pady=50)
```

```
    ctk.CTkLabel(splash, text="Loading your dashboard...", font=("Helvetica", 16)).pack(pady=20)

    progress = ctk.CTkProgressBar(splash, width=350, progress_color="#0a9396")
    progress.set(0)
    progress.pack(pady=20)

    def load():
        for i in range(101):
            time.sleep(0.03)
            progress.set(i / 100)
        splash.destroy()
        login_page()

    threading.Thread(target=load).start()

# PRODUCT FUNCTIONS
def add_product_page():
    clear_frame()
    ctk.CTkLabel(main_frame, text="➕ Add Product", font=("Helvetica", 22, "bold")).pack(pady=15)
    frame = ctk.CTkFrame(main_frame)
    frame.pack(pady=20)

    name = ctk.CTkEntry(frame, width=200, placeholder_text="Product Name")
    category = ctk.CTkEntry(frame, width=200, placeholder_text="Category")
    price = ctk.CTkEntry(frame, width=200, placeholder_text="Price (₹)")
    stock = ctk.CTkEntry(frame, width=200, placeholder_text="Stock Quantity")
    description = ctk.CTkEntry(frame, width=300, placeholder_text="Description")
```

```
    entries = [name, category, price, stock, description]
    for i, e in enumerate(entries):
        e.grid(row=i//2, column=i%2, padx=10, pady=10)

    def add_product():
        try:
            conn = connect_db()
            cur = conn.cursor()
            cur.execute("""
                INSERT INTO products (name, category, price, stock, description)
                VALUES (%s,%s,%s,%s,%s)
            """, (name.get(), category.get(), float(price.get()), int(stock.get()),
description.get()))
            conn.commit()
            conn.close()
            messagebox.showinfo("Success", "✅ Product added successfully!")
        except Exception as e:
            messagebox.showerror("Error", str(e))

    ctk.CTkButton(frame, text="Add Product", fg_color="#0a9396",
command=add_product).grid(row=3, column=0, columnspan=2, pady=20)
    ctk.CTkButton(main_frame, text="⬅ Back", fg_color="#0a9396",
command=back_to_home).pack(side="bottom", pady=15)

def view_products_page():
    clear_frame()
```

```
    ctk.CTkLabel(main_frame, text="📦 View Products", font=("Helvetica", 22, "bold")).pack(pady=15)
    try:
        conn = connect_db()
        cur = conn.cursor()
        cur.execute("SELECT product_id, name, category, price, stock FROM products")
        rows = cur.fetchall()
        headers = ["ID", "Name", "Category", "Price (₹)", "Stock"]
        display_table(main_frame, headers, rows)
        conn.close()
    except Exception as e:
        messagebox.showerror("Error", str(e))
    ctk.CTkButton(main_frame, text="⬅ Back", fg_color="#0a9396", command=back_to_home).pack(pady=20)

def search_product_page():
    clear_frame()
    ctk.CTkLabel(main_frame, text="🔍 Search Product", font=("Helvetica", 22, "bold")).pack(pady=15)
    frame = ctk.CTkFrame(main_frame)
    frame.pack(pady=10)

    entry = ctk.CTkEntry(frame, width=300, placeholder_text="Enter product name or category...")
    entry.grid(row=0, column=0, padx=10)

    result_frame = ctk.CTkFrame(main_frame)
```

```
    result_frame.pack(fill="both", expand=True, pady=10)

    def search_action():
        for w in result_frame.winfo_children():
            w.destroy()
        term = entry.get().strip()
        if not term:
            messagebox.showwarning("Empty", "Please enter a search term.")
            return
        conn = connect_db()
        cur = conn.cursor()
        cur.execute("SELECT * FROM products WHERE name LIKE %s OR category
LIKE %s", (f"%{term}%", f"%{term}%"))
        rows = cur.fetchall()
        conn.close()
        headers = ["ID", "Name", "Category", "Price", "Stock", "Description"]
        display_table(result_frame, headers, rows)

    ctk.CTkButton(frame, text="Search", fg_color="#0a9396",
command=search_action).grid(row=0, column=1, padx=10)
    ctk.CTkButton(main_frame, text="⬅ Back", fg_color="#0a9396",
command=back_to_home).pack(side="bottom", pady=15)
```

#USERS AND ORDERS

```
def view_users_page():
    clear_frame()
    ctk.CTkLabel(main_frame, text="👥 View Users", font=("Helvetica", 22,
"bold")).pack(pady=15)
```

```
    try:
        conn = connect_db()
        cur = conn.cursor()
        cur.execute("SELECT user_id, name, email, phone, created_at FROM users")
        rows = cur.fetchall()
        headers = ["User ID", "Name", "Email", "Phone", "Created At"]
        display_table(main_frame, headers, rows)
        conn.close()
    except Exception as e:
        messagebox.showerror("Error", str(e))
    ctk.CTkButton(main_frame, text="⬅ Back", fg_color="#0a9396",
command=back_to_home).pack(pady=15)

def view_orders_page():
    clear_frame()
    ctk.CTkLabel(main_frame, text="📜 Orders", font=("Helvetica", 22,
"bold")).pack(pady=15)
    try:
        conn = connect_db()
        cur = conn.cursor()
        cur.execute("""
            SELECT o.order_id, u.name AS user, o.total_amount, o.status, o.order_date
            FROM orders o
            JOIN users u ON o.user_id = u.user_id
        """)
        rows = cur.fetchall()
        headers = ["Order ID", "User", "Total (₹)", "Status", "Date"]
        display_table(main_frame, headers, rows)
```

```
        conn.close()
    except Exception as e:
        messagebox.showerror("Error", str(e))
    ctk.CTkButton(main_frame, text="⬅ Back", fg_color="#0a9396",
command=back_to_home).pack(pady=15)
```

 # DASHBOARD

```
def create_dashboard():
    ctk.CTkLabel(main_frame, text="🏀 SPORTS E-COMMERCE DASHBOARD",
font=("Helvetica", 26, "bold")).pack(pady=25)
    grid = ctk.CTkFrame(main_frame, fg_color="#edf6f9", corner_radius=15)
    grid.pack(pady=20)
    grid.grid_columnconfigure((0, 1, 2), weight=1)
    grid.grid_rowconfigure((0, 1), weight=1)

    buttons = [
        ("Add Product", "Add new sports product.", add_product_page),
        ("View Products", "See all available products.", view_products_page),
        ("Search Product", "Search products by name or category.",
search_product_page),
        ("View Users", "View registered users.", view_users_page),
        ("View Orders", "Check customer orders.", view_orders_page)
    ]

    r, c = 0, 0
    for title, desc, cmd in buttons:
        frame = ctk.CTkFrame(grid, corner_radius=15, fg_color="white", width=280,
height=160)
```

```
    frame.pack_propagate(False)
    frame.grid(row=r, column=c, padx=20, pady=20)
    ctk.CTkLabel(frame, text=title, font=("Helvetica", 18, "bold"), text_color="#005f73").pack(pady=10)
    ctk.CTkLabel(frame, text=desc, font=("Helvetica", 13), wraplength=220).pack(pady=5)
    ctk.CTkButton(frame, text="Open", fg_color="#0a9396", command=cmd).pack(pady=10)
    c += 1
    if c == 3:
        c = 0
        r += 1

# MAIN FRAME
main_frame = ctk.CTkFrame(app, fg_color="white")
main_frame.pack(fill="both", expand=True)

# Start with splash screen
splash_screen()
app.mainloop()
```

CHAPTER 5

SCREENSHOTS

LIST OF FIGURES

Figure No.	Title	Page No.
5.1	Login Page	26
5.2	Dashboard	26
5.3	Add Product Page	27
5.4	View Products Page	27
5.5	Search Product Page	28
5.6	View Users Page	28
5.7	Orders Page	29

5.1 LOGIN PAGE

5.2 DASHBOARD

5.3 ADD PRODUCTS

5.4 VIEW PRODUCTS

ID	Name	Category	Price (₹)	Stock
1	Football	Outdoor	999.00	20
2	Cricket Bat	Outdoor	1599.00	15
3	Tennis Racket	Indoor	1799.00	10
4	Yoga Mat	Fitness	499.00	30
5	Running Shoes	Footwear	2999.00	25
6	Badminton Shuttlecock	Indoor	299.00	50
7	Basketball	Outdoor	899.00	18
8	Skipping Rope	Fitness	249.00	40
9	Gym Gloves	Fitness	399.00	35
10	Dumbbell Set (5kg)	Fitness	999.00	25
11	Table Tennis Bat	Indoor	599.00	20
12	Table Tennis Balls	Indoor	199.00	100

5.5 SEARCH PRODUCTS

5.6 VIEW USERS

5.7 ORDERS

Sports E-Commerce

Orders

Order ID	User	Total (₹)	Status	Date
1	Athira	2598.00	Delivered	2025-10-01 00:00:00
2	Athira	2999.00	Pending	2025-10-05 00:00:00
11	Athira	899.00	Delivered	2025-10-26 00:00:00
3	Rahul	1799.00	Delivered	2025-10-07 00:00:00
4	Rahul	999.00	Shipped	2025-10-10 00:00:00
12	Rahul	1198.00	Delivered	2025-10-27 00:00:00
5	Priya	998.00	Delivered	2025-10-12 00:00:00
6	Priya	299.00	Delivered	2025-10-15 00:00:00
13	Priya	449.00	Shipped	2025-10-28 00:00:00
7	Vikram	2298.00	Shipped	2025-10-18 00:00:00
8	Vikram	1499.00	Delivered	2025-10-20 00:00:00
14	Vikram	598.00	Delivered	2025-10-29 00:00:00

Back

6. CONCLUSION AND FUTURE ENHANCEMENT

The E-Commerce project provides an efficient and secure way to manage online shopping activities such as user registration, product management, order processing, and payments. It ensures accurate data handling, quick access, and smooth interaction between customers and administrators. In the future, the system can be enhanced by adding AI-based product recommendations, real-time order tracking, secure online payment gateways with OTP verification, customer feedback and rating options, and a dedicated mobile application for better accessibility and user experience.