**PROGRAMMING FUNDAMENTAL**

**(3+1 Credit Hours)**

**CS-351**

**“UBIT CANTEEN MANAGEMENT SYSTEM”**

**Project Report**



**Course Instructor: Sir Tauseef Mubeen**  
**Class Section:** **BSCS 1A**  
**Semester:** **1st**

**Group Members:**

|  |  |
| --- | --- |
| **Enrollment** | **Name** |
| EB24210006040 | Mishaal Fahim |
| EB24210006080 | Neha |

**DEPARTMENT OF COMPUTER SCIENCE**

UBIT , UNIVERSITY OF KARACHI , PAKISTAN

Table of Content:

[Instructions 3](#_Toc155457424)

[Introduction: 4](#_Toc155457425)

[ Background 4](#_Toc155457426)

[ Scope 4](#_Toc155457427)

[ Objectives 4](#_Toc155457428)

[Methodology: 4](#_Toc155457429)

[Implementation: 4](#_Toc155457430)

[ Inputs 4](#_Toc155457431)

[ Outputs 4](#_Toc155457432)

[ Code Snippets 4](#_Toc155457433)

[Conclusion: 4](#_Toc155457434)

[ Evaluation 4](#_Toc155457435)

[ Screenshots 4](#_Toc155457436)

**INTRODUCTION:**

Efficient management of a canteen is crucial in ensuring that it operates smoothly, meets the needs of its users, and remains financially viable. With the increasing demand for quick and reliable services, a well-organized canteen management system is essential. This project report presents the development and implementation of a comprehensive Canteen Management System (CMS), designed to streamline operations with a specific focus on simplicity and efficiency, catering exclusively to two key users: shopkeepers and customers.

Shopkeepers can update the menu, manage pending orders, and monitor finances, including actual costs and selling prices, ensuring smooth operations and profitability. Customers, after securely logging in with a system-generated password (usable only once), must change their password upon first sign-in. Once logged in, customers can browse the menu, add items to their cart, view their bill, and access their order history.

The report covers the design, development, testing, and deployment of the CMS, emphasizing its role in enhancing the efficiency and security of online canteen operations for shopkeepers and customers.

**Background:**

The canteen industry plays a vital role in various institutions, providing essential food services to students, employees, and visitors. However, traditional methods of managing menu items, processing orders, and overseeing daily operations often lead to inefficiencies, delays, and errors. These conventional practices can result in customer dissatisfaction and financial losses for the canteen. In an age where technology can greatly improve the efficiency and accuracy of service delivery, there is an urgent need for a comprehensive Canteen Management System.

**Problem Statement:**

The current canteen management practices suffer from several inefficiencies, including manual menu updates, cumbersome order processing, and inadequate systems for tracking finances and inventory. Additionally, there is no streamlined method for customers to securely manage their accounts, view order history, or provide feedback. These issues lead to operational delays, customer dissatisfaction, and potential revenue loss. To address these challenges, our project proposes the development of a comprehensive Canteen Management System. This system will automate menu updates, streamline order processing, enable secure customer account management, and provide detailed financial tracking, all aimed at improving operational efficiency and enhancing the customer experience.

**SCOPE:**

Our Canteen Management System is a comprehensive solution tailored to meet the needs of canteens in various institutions, such as schools, colleges, and corporate environments. This system is designed to streamline key aspects of canteen operations, including menu management, order processing, and financial oversight, ensuring efficient and customer-centric services.

The system includes advanced features such as secure user authentication, allowing both shopkeepers and customers to access their specific functionalities through password-protected accounts. Shopkeepers can efficiently update the menu, manage and mark pending orders as fulfilled, and track sales and finances, ensuring accurate financial reporting. Customers, on the other hand, can browse the menu, place orders, view billing details, track their order history, and securely change their passwords.

Our Canteen Management System enhances operational efficiency, improves customer satisfaction, and ensures data security by implementing strict access controls. This project not only modernizes canteen operations but also prioritizes user convenience, data accuracy, and the seamless flow of information, making it an essential tool for contemporary canteen management.

**OBJECTIVES:**

The Canteen Management System is designed to enhance the efficiency, accuracy, and user experience for both shopkeepers and students in canteen operations. This system provides essential tools for managing menu items, processing orders, handling payments, and maintaining financial and order history records. By automating these processes, the system reduces errors, saves time, and improves overall service quality, allowing both shopkeepers and students to benefit from a more streamlined and convenient canteen experience.

**Shopkeeper Features**

**Menu Management:**

**Objective:** Enable shopkeepers to efficiently manage and update the menu to reflect current offerings and pricing with its quantity.

**Implementation Steps:**

* Develop a user-friendly interface for shopkeepers to add, remove, or modify menu items.
* Implement real-time updates to ensure that any changes made by the shopkeeper are instantly visible to students.
* Allow for categorization of items (e.g., beverages, snacks, chocolates ,fast food) to improve organization and ease of access.

**Order Processing:**

**Objective:** Streamline the order fulfillment process to ensure timely service and accurate order handling.

**Implementation Steps:**

* Create a secure order management system where shopkeepers can view incoming orders, process them, and mark them as fulfilled.

**Financial Management:**

**Objective:** Enhance the accuracy and transparency of financial operations by enabling comprehensive tracking and reporting.

**Implementation Steps:**

* Develop a financial module that tracks sales, including detailed records of costs, revenue, and profitability.
* Ensure that financial data is securely stored and accessible only to authorized shopkeepers.

**Student Features**

**Menu Browsing:**

**Objective:** Provide students with a convenient way to view the canteen’s menu and make informed choices.

**Implementation Steps:**

* Create a user-friendly interface for students to browse the menu, with clear descriptions and prices for each item.
* Allow students to filter or search for items based on categories.
* Implement real-time updates to ensure students see the most current menu options.

**Order Placement and Billling:**

**Objective:** Simplify the ordering process for students, enabling them to place and get bill of orders with ease.

**Implementation Steps:**

* Develop a secure system where students can log in, select items, add them to their cart, and place orders.
* Provide a detailed billing summary before order confirmation to ensure transparency.

**Account Management:**

**Objective:** Allow students to securely manage their accounts and view order history.

**Implementation Steps:**

* Implement a secure login system where students can access their accounts using a system-generated password that requires updating upon first login.
* Develop a user interface for viewing past orders, managing saved payment methods, and updating account information.
* Ensure that all account and order data is stored securely and in compliance with privacy standards.

By incorporating these features, the Canteen Management System aims to deliver a seamless, efficient, and secure experience for both shopkeepers and students, ultimately enhancing the overall canteen operation.

**INPUTS AND OUTPUTS:**

1. **Shopkeeper I’d Login:**
   * Inputs:
     + Shopkeeper details (ID, password)
     + Checks it matches with the shopkeeper data
   * Outputs:
     + Updated menu , pending orders and finances
2. **Updated menu:**
   * Inputs:
     + Select category(Chocolates ,Snacks, Beverages, Fast food) then load items
     + Remove or add an item in the category
   * Outputs:
     + Menu is updated in database and students can see the updated menu
3. **Previous orders:**
   * Inputs:
     + Mark as fulfilled the order details (student id , item name, quantity, Total price)
   * Outputs:
     + Updated order details in the database
4. **Finances:**
   * Inputs:
     + By actual and selling values of the item products
   * Outputs:
     + Total investment for menu
     + Profit from menu by selling price
5. **Student login :**
   * Inputs:
     + Sign in or sign up
   * Outputs:
     + Menu (Place order and get bill)
     + Previous orders
6. **Menu:**
   * Inputs:
     + Add to cart
     + Place orders
   * Outputs:
     + Total bill of order

**Code Snippets:**

import tkinter as tk

from tkinter import messagebox, filedialog, ttk

from PIL import Image, ImageTk

from tkinter import font

import sqlite3

import random

import string

from io import BytesIO

import requests

# Database setup

conn = sqlite3.connect('canteen\_management.db')

cursor = conn.cursor()

cursor.execute('''

CREATE TABLE IF NOT EXISTS shopkeeper (

id TEXT PRIMARY KEY,

password TEXT NOT NULL

)

''')

def create\_menu\_table():

cursor.execute('''

CREATE TABLE IF NOT EXISTS menu (

id INTEGER PRIMARY KEY AUTOINCREMENT,

item TEXT NOT NULL,

category TEXT,

quantity INTEGER NOT NULL,

actual\_price REAL NOT NULL,

selling\_price REAL NOT NULL,

boycotted TEXT NOT NULL

)

''')

def add\_category\_column():

try:

cursor.execute('ALTER TABLE menu ADD COLUMN category TEXT')

except sqlite3.OperationalError:

# Column might already exist

pass

cursor.execute('''

CREATE TABLE IF NOT EXISTS student (

id TEXT PRIMARY KEY,

name TEXT NOT NULL,

cnic TEXT NOT NULL,

major TEXT NOT NULL,

photo BLOB,

password TEXT NOT NULL,

first\_login INTEGER NOT NULL DEFAULT 1

)

''')

cursor.execute('''

CREATE TABLE IF NOT EXISTS orders (

id INTEGER PRIMARY KEY AUTOINCREMENT,

student\_id TEXT NOT NULL,

items TEXT NOT NULL,

total\_price REAL NOT NULL,

fulfilled INTEGER NOT NULL DEFAULT 0,

FOREIGN KEY(student\_id) REFERENCES student(id)

)

''')

def add\_fulfilled\_column():

try:

cursor.execute('ALTER TABLE orders ADD COLUMN fulfilled INTEGER NOT NULL DEFAULT 0')

except sqlite3.OperationalError:

pass # Column might already exist

# Insert shopkeeper data

cursor.execute("INSERT OR IGNORE INTO shopkeeper (id, password) VALUES ('675409', 'ubit2468'), ('676377', 'ubit2468')")

# Insert default menu items

def insert\_sample\_data():

cursor.execute('''

INSERT OR IGNORE INTO menu (item, category, quantity, actual\_price, selling\_price, boycotted) VALUES

('Dairy milk', 'Chocolates', 100, 10.0, 12.0, 'yes'),

('Tempo', 'Snacks', 50, 20.0, 25.0, 'no'),

('Cola next', 'Beverages', 30, 35.0, 40.0, 'no'),

('Burger', 'Fast Food', 25, 50.0, 65.0, 'no')

''')

create\_menu\_table()

add\_category\_column()

add\_fulfilled\_column()

insert\_sample\_data()

conn.commit()

# Helper functions

def generate\_random\_password(length=8):

return ''.join(random.choice(string.ascii\_letters + string.digits) for \_ in range(length))

def save\_image(image\_path):

try:

with open(image\_path) as file:

return file.read()

except FileNotFoundError:

return None

except Exception as e:

print(f"Error reading image: {e}")

return None

def validate\_student\_data(student\_id, cnic):

if len(student\_id) != 6 or not student\_id.isdigit():

return False, "Student ID must be 6 digits."

if len(cnic) != 13 or not cnic.isdigit():

return False, "CNIC must be 13 digits."

return True, ""

def clear\_window():

for widget in root.winfo\_children():

widget.destroy()

# Shopkeeper Functions

def shopkeeper\_menu(shopkeeper\_id):

clear\_window()

# Create two frames to divide the window in half

left\_frame = tk.Frame(root, bg='MediumOrchid1', width=300, height=400)

left\_frame.pack(side="left", fill="both", expand=True)

right\_frame = tk.Frame(root, bg='white', width=300, height=400)

right\_frame.pack(side="right", fill="both", expand=True)

# Add Welcome Label to the left frame

welcome\_label = tk.Label(left\_frame, text=f"Welcome \n\n{shopkeeper\_id}!", font=("Helvetica", 40, "bold"), fg='white', bg='MediumOrchid1')

welcome\_label.pack(pady=250) # Adjust padding as necessary

# Custom style for buttons

button\_font = font.Font(family="Helvetica", size=18, weight="bold")

button\_style = {"font": button\_font, "fg": "white", "bg": "MediumOrchid1", "activebackground": "DarkOrchid2", "bd": 0, "relief": "flat", "width": 20}

tk.Frame(right\_frame, bg='white').pack(side="top", expand=True)

# Add buttons to the right frame

tk.Button(right\_frame, text="Update Menu", command=update\_menu, \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Pending Orders", command=show\_pending\_orders, \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Finance", command=show\_finance, \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Exit", command=main\_screen, \*\*button\_style).pack(pady=20)

tk.Frame(right\_frame, bg='white').pack(side="bottom", expand=True)

def update\_menu():

clear\_window()

# Create a main frame that will contain the two sections

main\_frame = tk.Frame(root, bg="white")

main\_frame.pack(fill="both", expand=True)

# Create the left frame for displaying items

left\_frame = tk.Frame(main\_frame, bg='brown')

left\_frame.grid(row=0, column=0, sticky="nsew")

# Create the right frame for adding new items

right\_frame = tk.Frame(main\_frame, bg='peach puff')

right\_frame.grid(row=0, column=1, sticky="nsew")

# Configure column weights to give more space to the left frame

main\_frame.columnconfigure(0, weight=3)

main\_frame.columnconfigure(1, weight=2)

main\_frame.rowconfigure(0, weight=1)

# Update Menu Label on the left

tk.Label(left\_frame, text="Update Menu", font=("Helvetica", 28,"bold"), fg="peach puff", bg='brown').pack(pady=30)

categories = ["Chocolates", "Snacks", "Beverages", "Fast Food"]

category\_var = tk.StringVar(root)

category\_var.set("Select Category")

style = ttk.Style()

style.theme\_use('default')

style.configure("TCombobox",

fieldbackground="white",

background="white",

selectbackground="white",

selectforeground="black",

arrowsize=20,

relief="flat")

category\_dropdown = ttk.Combobox(left\_frame, textvariable=category\_var, values=categories, state="readonly", width=20, font=("Helvetica", 15), style="TCombobox")

category\_dropdown.pack(pady=10)

items\_frame = tk.Frame(left\_frame, bg='brown')

items\_frame.pack(pady=10)

def load\_items():

for widget in items\_frame.winfo\_children():

widget.destroy()

selected\_category = category\_var.get()

cursor.execute("SELECT item, quantity, actual\_price, selling\_price, boycotted FROM menu WHERE category=?", (selected\_category,))

items = cursor.fetchall()

for item in items:

item\_frame = tk.Frame(items\_frame, bg='peach puff')

item\_frame.pack(pady=5, fill="x", padx=5)

item\_info = f"{item[0]} - Qty: {item[1]}, Actual Price: {item[2]}, Selling Price: {item[3]}, Boycotted: {item[4]}"

tk.Label(item\_frame, text=item\_info, bg='peach puff', font=("Helvetica", 12, "bold")).pack(side=tk.LEFT, padx=5)

button\_frame = tk.Frame(item\_frame, bg='brown')

button\_frame.pack(side=tk.RIGHT)

tk.Button(button\_frame, text="Remove", command=lambda i=item[0]: remove\_item(i), fg="black", bg='peach puff').pack(padx=5, pady=2)

def remove\_item(item\_name):

cursor.execute("DELETE FROM menu WHERE item=?", (item\_name,))

conn.commit()

load\_items()

tk.Button(left\_frame, text="Load Items", command=load\_items, bg='peach puff',font=("Helvetica", 10)).pack(pady=10)

# Add New Item section on the right

tk.Label(right\_frame, text="Add New Item", font=("Helvetica", 28, "bold"),fg="brown", bg='peach puff').pack(pady=30)

tk.Label(right\_frame, text="Item Name", bg='peach puff').pack()

item\_name = tk.Entry(right\_frame)

item\_name.pack(pady=5)

tk.Label(right\_frame, text="Quantity", bg='peach puff').pack()

quantity = tk.Entry(right\_frame)

quantity.pack(pady=5)

tk.Label(right\_frame, text="Actual Price", bg='peach puff').pack()

actual\_price = tk.Entry(right\_frame)

actual\_price.pack(pady=5)

tk.Label(right\_frame, text="Selling Price", bg='peach puff').pack()

selling\_price = tk.Entry(right\_frame)

selling\_price.pack(pady=5)

tk.Label(right\_frame, text="Boycotted (yes/no)", bg='peach puff').pack()

boycotted = tk.Entry(right\_frame)

boycotted.pack(pady=5)

def save\_menu\_item():

cursor.execute('''INSERT OR REPLACE INTO menu (item, category, quantity, actual\_price, selling\_price, boycotted)

VALUES (?, ?, ?, ?, ?, ?)''', (item\_name.get(), category\_var.get(), int(quantity.get()),

float(actual\_price.get()), float(selling\_price.get()), boycotted.get()))

conn.commit()

messagebox.showinfo("Success", "Menu item updated successfully!")

shopkeeper\_menu(current\_shopkeeper\_id)

tk.Button(right\_frame, text="Save", command=save\_menu\_item, bg='peach puff', font=("Helvetica", 10)).pack(pady=10)

tk.Button(right\_frame, text="Back", command=lambda: shopkeeper\_menu(current\_shopkeeper\_id), bg='peach puff',font=("Helvetica", 10)).pack(pady=10)

def show\_pending\_orders():

clear\_window()

response = requests.get("https://img.freepik.com/free-vector/flat-geometric-background\_23-2148967370.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="Pending Orders", font=("Helvetica", 30,"bold"), fg='brown').pack(pady=50)

orders\_frame = tk.Frame(root, bg='light pink')

orders\_frame.pack(pady=10)

cursor.execute("SELECT id, student\_id, items, total\_price FROM orders WHERE fulfilled=0")

orders = cursor.fetchall()

checkbuttons = []

order\_vars = []

for order in orders:

order\_frame = tk.Frame(orders\_frame, bg='light pink')

order\_frame.pack(pady=5)

order\_var = tk.IntVar()

order\_vars.append(order\_var)

order\_info = f"Order ID: {order[0]}, Student ID: {order[1]}, Items: {order[2]}, Total Price: {order[3]}"

tk.Checkbutton(order\_frame, text=order\_info, variable=order\_var, bg='light pink').pack(side=tk.LEFT)

checkbuttons.append(order\_var)

def mark\_orders\_fulfilled():

for i, order\_var in enumerate(order\_vars):

if order\_var.get() == 1:

cursor.execute("UPDATE orders SET fulfilled=1 WHERE id=?", (orders[i][0],))

conn.commit()

messagebox.showinfo("Success", "Selected orders have been fulfilled!")

show\_pending\_orders()

tk.Button(root, text="Mark as Fulfilled", command=mark\_orders\_fulfilled, font=("Helvetica", 15), bg='light pink').pack(pady=10)

tk.Button(root, text="Back", command=lambda: shopkeeper\_menu(current\_shopkeeper\_id), font=("Helvetica", 15), bg='light pink').pack(pady=10)

def show\_finance():

# Retrieve all fulfilled orders

cursor.execute("SELECT items, total\_price FROM orders WHERE fulfilled=1")

orders = cursor.fetchall()

total\_cost = 0

total\_profit = 0

for items, total\_price in orders:

items\_list = items.split(', ')

for item in items\_list:

try:

# Split the item into name and quantity

item\_parts = item.rsplit(' x', 1)

if len(item\_parts) == 2:

item\_name, quantity\_str = item\_parts

quantity = int(quantity\_str)

# Fetch the actual price and selling price for the item

cursor.execute("SELECT actual\_price, selling\_price FROM menu WHERE item=?", (item\_name,))

result = cursor.fetchone()

if result:

actual\_price, selling\_price = result

item\_cost = actual\_price \* quantity

item\_profit = (selling\_price - actual\_price) \* quantity

total\_cost += item\_cost

total\_profit += item\_profit

else:

print(f"Warning: Item '{item\_name}' not found in the menu.")

else:

print(f"Warning: Item format incorrect '{item}'.")

except ValueError as e:

print(f"Error processing item '{item}': {e}")

clear\_window()

response = requests.get("https://img.freepik.com/free-vector/flat-geometric-background\_23-2148967370.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="Finance", font=("Helvetica", 30,"bold"), fg='brown').pack(pady=50)

tk.Label(root, text=f"Total Cost: {total\_cost:.2f} pkr", font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

tk.Label(root, text=f"Total Profit: {total\_profit:.2f} pkr", font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Back", command=lambda: shopkeeper\_menu(current\_shopkeeper\_id), font=("Helvetica", 18), bg='light pink').pack(pady=20)

# Student Functions

def student\_menu():

clear\_window()

response = requests.get("https://img.freepik.com/free-vector/flat-design-abstract-background\_23-2149103999.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="Student Options", font=("Helvetica", 36,"bold"), fg="brown", bg='white').pack(pady=70)

# Load and resize the signup image

signup\_image = Image.open("signup-icon.png")

signup\_image = signup\_image.resize((50, 50))

signup\_image = ImageTk.PhotoImage(signup\_image)

# Load and resize the login image

login\_image = Image.open("logins-icon.png")

login\_image = login\_image.resize((50, 50))

login\_image = ImageTk.PhotoImage(login\_image)

# Load and resize the exit image

exit\_image = Image.open("exit-icon.png")

exit\_image = exit\_image.resize((50, 50))

exit\_image = ImageTk.PhotoImage(exit\_image)

# Create a frame to hold the signup image and button side by side

signup\_frame = tk.Frame(root, bg='white')

signup\_frame.pack(pady=20)

tk.Label(signup\_frame, image=signup\_image, bg='white').pack(side="left")

tk.Button(signup\_frame, text="Student Sign-Up", command=student\_signup, font=("Helvetica", 18), bg='white').pack(side="left", padx=20)

# Create a frame to hold the login image and button side by side

login\_frame = tk.Frame(root, bg='white')

login\_frame.pack(pady=20)

tk.Label(login\_frame, image=login\_image, bg='white').pack(side="left")

tk.Button(login\_frame, text="Student Login", command=student\_login, font=("Helvetica", 18), bg='white').pack(side="left", padx=20)

# Create a frame to hold the exit image and button side by side

exit\_frame = tk.Frame(root, bg='white')

exit\_frame.pack(pady=20)

tk.Label(exit\_frame, image=exit\_image, bg='white').pack(side="left")

tk.Button(exit\_frame, text="Exit", command=main\_screen, font=("Helvetica", 18), bg='white').pack(side="left", padx=20)

# Keep a reference to the images to prevent garbage collection

signup\_frame.image = signup\_image

login\_frame.image = login\_image

exit\_frame.image = exit\_image

def student\_portal(student\_id):

clear\_window()

# Create two frames to divide the window in half

left\_frame = tk.Frame(root, bg='MediumOrchid1', width=300, height=400)

left\_frame.pack(side="left", fill="both", expand=True)

right\_frame = tk.Frame(root, bg='white', width=300, height=400)

right\_frame.pack(side="right", fill="both", expand=True)

# Add Welcome Label to the left frame

welcome\_label = tk.Label(left\_frame, text=f"Welcome \n\n{student\_id}!", font=("Helvetica", 40, "bold"), fg='white', bg='MediumOrchid1')

welcome\_label.pack(pady=250) # Adjust padding as necessary

# Custom style for buttons

button\_font = font.Font(family="Helvetica", size=18, weight="bold")

button\_style = {"font": button\_font, "fg": "white", "bg": "MediumOrchid1", "activebackground": "DarkOrchid2", "bd": 0, "relief": "flat", "width": 20}

tk.Frame(right\_frame, bg='white').pack(side="top", expand=True)

# Add buttons to the right frame

tk.Button(right\_frame, text="Show Menu", command=lambda: show\_menu(student\_id), \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Previous Orders", command=lambda: previous\_orders(student\_id), \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Change Password", command=lambda: change\_password(student\_id), \*\*button\_style).pack(pady=20)

tk.Button(right\_frame, text="Exit", command=student\_menu, \*\*button\_style).pack(pady=20)

tk.Frame(right\_frame, bg='white').pack(side="bottom", expand=True)

def show\_menu(student\_id):

clear\_window()

root.configure(bg='peachpuff')

tk.Label(root, text="Menu", font=("Helvetica", 30, "bold"), fg='brown', bg='peachpuff').pack(pady=30)

categories = ["Chocolates", "Snacks", "Beverages", "Fast Food"]

category\_var = tk.StringVar(root)

category\_var.set("Select Category")

style = ttk.Style()

style.theme\_use('default')

style.configure("TCombobox",

fieldbackground="white",

background="white",

selectbackground="white",

selectforeground="black",

arrowsize=20,

relief="flat")

category\_dropdown = ttk.Combobox(root, textvariable=category\_var, values=categories, state="readonly", width=20, font=("Helvetica", 15), style="TCombobox")

category\_dropdown.pack(pady=10)

items\_frame = tk.Frame(root, bg='peachpuff')

items\_frame.pack(pady=10)

cart\_frame = tk.Frame(root, bg='peachpuff')

cart\_frame.pack(pady=10)

cart = {}

def update\_cart\_display():

for widget in cart\_frame.winfo\_children():

widget.destroy()

total\_price = 0

for item, details in cart.items():

item\_frame = tk.Frame(cart\_frame, bg='peachpuff')

item\_frame.pack(pady=5)

item\_info = f"{item.capitalize()} - Qty: {details['quantity']}, Price: {details['price'] \* details['quantity']}"

total\_price += details['price'] \* details['quantity']

tk.Label(item\_frame, text=item\_info, bg='peachpuff').pack(side=tk.LEFT)

tk.Label(cart\_frame, text=f"Total Price: {total\_price}", bg='peachpuff').pack(pady=10)

def add\_to\_cart(item, price):

if item not in cart:

cart[item] = {"quantity": 1, "price": price}

else:

cart[item]["quantity"] += 1

update\_cart\_display()

def remove\_from\_cart(item):

if item in cart:

cart[item]["quantity"] -= 1

if cart[item]["quantity"] <= 0:

del cart[item]

update\_cart\_display()

def load\_items():

for widget in items\_frame.winfo\_children():

widget.destroy()

cursor.execute("SELECT item, quantity, selling\_price FROM menu WHERE category=? AND boycotted='no'", (category\_var.get(),))

items = cursor.fetchall()

for item in items:

item\_frame = tk.Frame(items\_frame, bg='peachpuff')

item\_frame.pack(pady=5)

item\_label = tk.Label(item\_frame, text=f"{item[0]} - Qty: {item[1]}, Price: {item[2]}", font=("Helvetica", 12, "bold"), bg='peachpuff')

item\_label.pack(side=tk.LEFT)

add\_button = tk.Button(item\_frame, text="+", command=lambda i=item[0], p=item[2]: add\_to\_cart(i, p), bg='peachpuff')

add\_button.pack(side=tk.RIGHT)

remove\_button = tk.Button(item\_frame, text="-", command=lambda i=item[0]: remove\_from\_cart(i), bg='peachpuff')

remove\_button.pack(side=tk.RIGHT)

tk.Button(root, text="Load Items", command=load\_items, font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Go to Billing", command=lambda: billing(student\_id, cart), font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Back", command=lambda: student\_portal(student\_id), font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

def billing(student\_id, cart):

clear\_window()

root.configure(bg='peachpuff')

tk.Label(root, text="Billing", font=("Helvetica", 30, "bold"), fg='brown', bg='peachpuff').pack(pady=20)

# Display a narrow receipt

receipt\_frame = tk.Frame(root, bg='white', relief="solid", borderwidth=2, width=1000)

receipt\_frame.pack(pady=10, padx=10)

tk.Label(receipt\_frame, text="UBIT Canteen", font=("Helvetica", 24), bg='white').pack(pady=20, padx=30)

tk.Label(receipt\_frame, text=f"Invoice to: {student\_id}", font=("Helvetica", 12), bg='white').pack()

items\_frame = tk.Frame(receipt\_frame, bg='white')

items\_frame.pack(pady=5, padx=5)

tk.Label(items\_frame, text="Sl. | Item | Price | Qty | Total", bg='white', anchor="w", justify=tk.LEFT).pack()

total\_price = 0

for idx, (item, details) in enumerate(cart.items(), start=1):

item\_total = details['price'] \* details['quantity']

total\_price += item\_total

tk.Label(items\_frame, text=f"{idx}. | {item.capitalize()} | {details['price']} | {details['quantity']} | {item\_total}", bg='white', anchor="w", justify=tk.LEFT).pack()

tk.Label(receipt\_frame, text=f"Subtotal: {total\_price}", font=("Helvetica", 12), bg='white').pack(pady=5)

tk.Label(receipt\_frame, text=f"Total: {total\_price}", font=("Helvetica", 12, 'bold'), bg='white').pack(pady=20)

def place\_order():

items\_str = '\n'.join([f"{item} x {details['quantity']}" for item, details in cart.items()])

cursor.execute('''INSERT INTO orders (student\_id, items, total\_price) VALUES (?, ?, ?)''', (student\_id, items\_str, total\_price))

conn.commit()

messagebox.showinfo("Order Placed", "Your order has been placed successfully!")

student\_portal(student\_id)

tk.Button(root, text="Place Order", command=place\_order, font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Back", command=lambda: show\_menu(student\_id),font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

def previous\_orders(student\_id):

clear\_window()

response = requests.get("https://img.freepik.com/free-vector/flat-geometric-background\_23-2148967370.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="Previous Orders", font=("Helvetica", 30, "bold"), fg='brown').pack(pady=50)

orders\_frame = tk.Frame(root, bg='white')

orders\_frame.pack(pady=10)

cursor.execute("SELECT items, total\_price FROM orders WHERE student\_id=?", (student\_id,))

orders = cursor.fetchall()

for order in orders:

order\_frame = tk.Frame(orders\_frame)

order\_frame.pack(pady=5)

order\_info = f"Items: {order[0]}, Total Price: {order[1]}"

tk.Label(order\_frame, text=order\_info, font=("Helvetica", 18), bg='peach puff', fg='brown').pack(side=tk.LEFT, pady=5)

tk.Button(root, text="Back", command=lambda: student\_portal(student\_id), font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

def change\_password(student\_id):

clear\_window()

response = requests.get("https://img.freepik.com/free-vector/flat-geometric-background\_23-2148967370.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="Change Password", font=("Helvetica", 30,"bold"), fg='brown').pack(pady=50)

tk.Label(root, text="New Password", font=("Helvetica", 22), fg='brown').pack(pady=20)

new\_password = tk.Entry(root, show="\*", width=40)

new\_password.pack(pady=5)

def update\_password():

cursor.execute("UPDATE student SET password=? WHERE id=?", (new\_password.get(), student\_id))

conn.commit()

messagebox.showinfo("Success", "Password updated successfully!")

student\_portal(student\_id)

tk.Button(root, text="Change Password", command=update\_password, font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=30)

tk.Button(root, text="Back", command=lambda: student\_portal(student\_id), font=("Helvetica", 18), bg='peach puff', fg='brown').pack(pady=20)

def student\_signup():

clear\_window()

response = requests.get("https://images.rawpixel.com/image\_png\_800/czNmcy1wcml2YXRlL3Jhd3BpeGVsX2ltYWdlcy93ZWJzaXRlX2NvbnRlbnQvbHIvdjkxNS13aXQtMDAyLnBuZw.png")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

# Create a frame for the sign-up form

signup\_frame = tk.Frame(root, bg='white')

signup\_frame.pack(pady=50)

# Create a label for the sign-up title

tk.Label(signup\_frame, text="Student Sign-Up", font=("Helvetica", 24), bg='#FFFFFF', fg='brown').pack(pady=20)

# Create a frame for the input fields

input\_frame = tk.Frame(signup\_frame, bg='white')

input\_frame.pack(pady=20)

# Create labels and entries for student ID, name, CNIC, major, and photo

tk.Label(input\_frame, text="Student ID", bg='white', fg='black').grid(row=0, column=0, padx=10, pady=10)

student\_id = tk.Entry(input\_frame, width=30, bg='#F7F7F7', fg='#808080')

student\_id.grid(row=0, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="Name", bg='white', fg='black').grid(row=1, column=0, padx=10, pady=10)

name = tk.Entry(input\_frame, width=30, bg='#F7F7F7', fg='#808080')

name.grid(row=1, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="CNIC", bg='white', fg='black').grid(row=2, column=0, padx=10, pady=10)

cnic = tk.Entry(input\_frame, width=30, bg='#F7F7F7', fg='#808080')

cnic.grid(row=2, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="Major", bg='white', fg='black').grid(row=3, column=0, padx=10, pady=10)

major = tk.Entry(input\_frame, width=30, bg='#F7F7F7', fg='#808080')

major.grid(row=3, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="Photo", bg='white', fg='black').grid(row=4, column=0, padx=10, pady=10)

photo\_path = tk.Entry(input\_frame, width=30, bg='#F7F7F7', fg='#808080')

photo\_path.grid(row=4, column=1, padx=10, pady=10)

def browse\_photo():

file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.jpg;\*.jpeg;\*.png;\*.gif")])

file\_path = file\_path.encode('utf-8')

photo\_path.delete(0, tk.END)

photo\_path.insert(0, file\_path)

tk.Button(input\_frame, text="Browse", command=browse\_photo, bg='peach puff', fg='brown').grid(row=4, column=2, padx=10, pady=10)

def register\_student():

validation\_result, message = validate\_student\_data(student\_id.get(), cnic.get())

if not validation\_result:

messagebox.showerror("Error", message)

return

generated\_password = generate\_random\_password()

photo\_data = save\_image(photo\_path.get())

cursor.execute('''INSERT INTO student (id, name, cnic, major, photo, password)

VALUES (?, ?, ?, ?, ?, ?)''',

(student\_id.get(), name.get(), cnic.get(), major.get(), photo\_data, generated\_password))

conn.commit()

tk.Label(signup\_frame, text="Password Generated:", font=("Helvetica", 18), bg='#FFFFFF', fg='#808080').pack(pady=10)

password\_label = tk.Label(signup\_frame, text=generated\_password, font=("Helvetica", 18), bg='#FFFFFF', fg='#808080')

password\_label.pack(pady=10)

def copy\_password():

root.clipboard\_clear()

root.clipboard\_append(generated\_password)

root.update()

messagebox.showinfo("Copied", "Password copied to clipboard!")

tk.Button(signup\_frame, text="Copy Password", command=copy\_password, bg='peach puff', fg='brown').pack(pady=10)

tk.Button(signup\_frame, text="Sign In", command=student\_login, bg='peach puff', fg='brown').pack(pady=10)

tk.Button(input\_frame, text="Sign Up", command=register\_student, bg='peach puff', fg='brown').grid(row=5, column=1, padx=10, pady=10)

tk.Button(input\_frame, text="Back", command=student\_menu,bg='peach puff', fg='brown').grid(row=5, column=2, padx=10, pady=10)

def student\_login():

clear\_window()

response = requests.get("https://t4.ftcdn.net/jpg/02/36/19/51/360\_F\_236195116\_IWbsVnjfNpwfioW6eHgIXoIqlif9ZXei.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

# Create a frame for the login form

login\_frame = tk.Frame(root, bg='white')

login\_frame.pack(pady=180, padx=180)

# Create a label for the student login title

tk.Label(login\_frame, text="Student Login", font=("Helvetica", 24, "bold"), fg='brown', bg='white').pack(pady=20)

# Create a frame for the input fields

input\_frame = tk.Frame(login\_frame, bg='white')

input\_frame.pack(pady=20)

# Create labels and entries for student ID and password

tk.Label(input\_frame, text="Student ID", font=("Helvetica", 18), fg='black', bg='white').grid(row=0, column=0, padx=10, pady=10)

student\_id = tk.Entry(input\_frame, font=("Helvetica", 18), width=25)

student\_id.grid(row=0, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="Password", font=("Helvetica", 18), fg='black', bg='white').grid(row=1, column=0, padx=10, pady=10)

student\_password = tk.Entry(input\_frame, font=("Helvetica", 18), width=25, show="\*")

student\_password.grid(row=1, column=1, padx=10, pady=10)

def login\_student():

cursor.execute("SELECT password FROM student WHERE id=?", (student\_id.get(),))

result = cursor.fetchone()

if result and result[0] == student\_password.get():

student\_portal(student\_id.get())

else:

messagebox.showerror("Error", "Invalid student ID or password.")

tk.Button(input\_frame, text="Login", command=login\_student, font=("Helvetica", 18), bg='pink', fg='white', width=10).grid(row=2, column=1, padx=10, pady=20)

tk.Button(input\_frame, text="Back", command=student\_menu, font=("Helvetica", 18), bg='gray', fg='white', width=10).grid(row=2, column=0, padx=10, pady=20)

# Main Screen

def main\_screen():

clear\_window()

response = requests.get("https://i.ytimg.com/vi/j03xfrFeYhM/maxresdefault.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

tk.Label(root, text="UBIT Canteen Management System", font=("Helvetica", 44, "bold"), fg='brown').pack(pady=70)

tk.Button(root, text="Shopkeeper", command=shopkeeper\_login, font=("Helvetica", 28), activebackground='lavender', activeforeground='brown', bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Student", command=student\_menu, font=("Helvetica", 28), activebackground='lavender', activeforeground='brown', bg='peach puff', fg='brown').pack(pady=20)

tk.Button(root, text="Exit", command=root.quit, font=("Helvetica", 28), activebackground='lavender', activeforeground='brown', bg='peach puff', fg='brown').pack(pady=20)

def shopkeeper\_login():

clear\_window()

response = requests.get("https://t4.ftcdn.net/jpg/02/36/19/51/360\_F\_236195116\_IWbsVnjfNpwfioW6eHgIXoIqlif9ZXei.jpg")

img\_data = response.content

img = Image.open(BytesIO(img\_data))

img = img.resize((root.winfo\_screenwidth(), root.winfo\_screenheight()))

background\_image = ImageTk.PhotoImage(img)

background\_label = tk.Label(root, image=background\_image)

background\_label.image = background\_image

background\_label.place(relwidth=1, relheight=1)

# Create a frame for the login form

login\_frame = tk.Frame(root, bg='white')

login\_frame.pack(pady=180, padx=180)

# Create a label for the shopkeeper login title

tk.Label(login\_frame, text="Shopkeeper Login", font=("Helvetica", 24, "bold"), fg='brown', bg='white').pack(pady=20)

# Create a frame for the input fields

input\_frame = tk.Frame(login\_frame, bg='white')

input\_frame.pack(pady=20)

# Create labels and entries for shopkeeper ID and password

tk.Label(input\_frame, text="Shopkeeper ID", font=("Helvetica", 18), fg='black', bg='white').grid(row=0, column=0, padx=10, pady=10)

shopkeeper\_id = tk.Entry(input\_frame, font=("Helvetica", 18), width=25)

shopkeeper\_id.grid(row=0, column=1, padx=10, pady=10)

tk.Label(input\_frame, text="Password", font=("Helvetica", 18), fg='black', bg='white').grid(row=1, column=0, padx=10, pady=10)

shopkeeper\_password = tk.Entry(input\_frame, font=("Helvetica", 18), width=25, show="\*")

shopkeeper\_password.grid(row=1, column=1, padx=10, pady=10)

def login\_shopkeeper():

global current\_shopkeeper\_id

cursor.execute("SELECT password FROM shopkeeper WHERE id=?", (shopkeeper\_id.get(),))

result = cursor.fetchone()

if result and result[0] == shopkeeper\_password.get():

current\_shopkeeper\_id = shopkeeper\_id.get()

shopkeeper\_menu(current\_shopkeeper\_id)

else:

messagebox.showerror("Error", "Invalid shopkeeper ID or password.")

tk.Button(input\_frame, text="Login", command=login\_shopkeeper, font=("Helvetica", 18), bg='pink', fg='white', width=10).grid(row=2, column=1, padx=10, pady=20)

# Create a button for back

tk.Button(input\_frame, text="Back", command=main\_screen, font=("Helvetica", 18), bg='gray', fg='white', width=10).grid(row=2, column=0, padx=20, pady=20)

# Initialize the main application window

root = tk.Tk()

root.title("Cafeteria Management System")

root.geometry("600x400")

# Launch the main screen

main\_screen()

# Start the GUI event loop

root.mainloop()

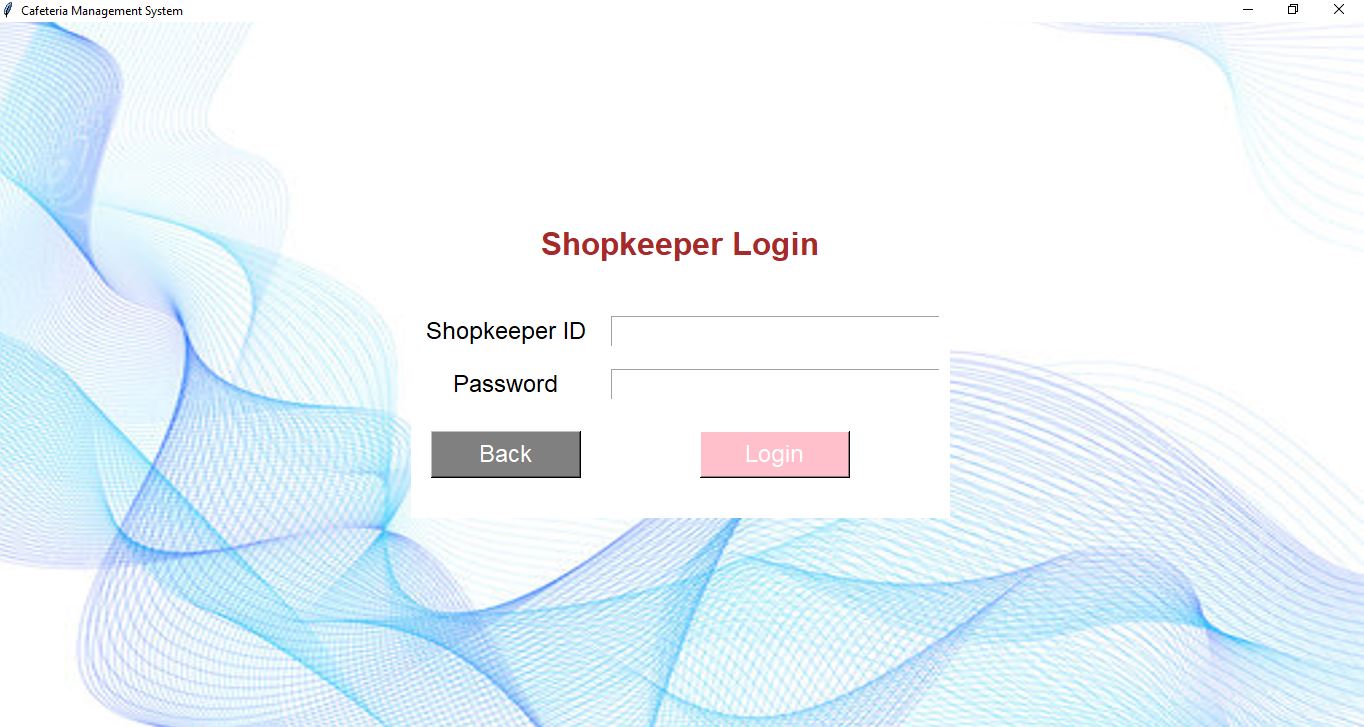
**Conclusion:**

1. **Screenshots:**

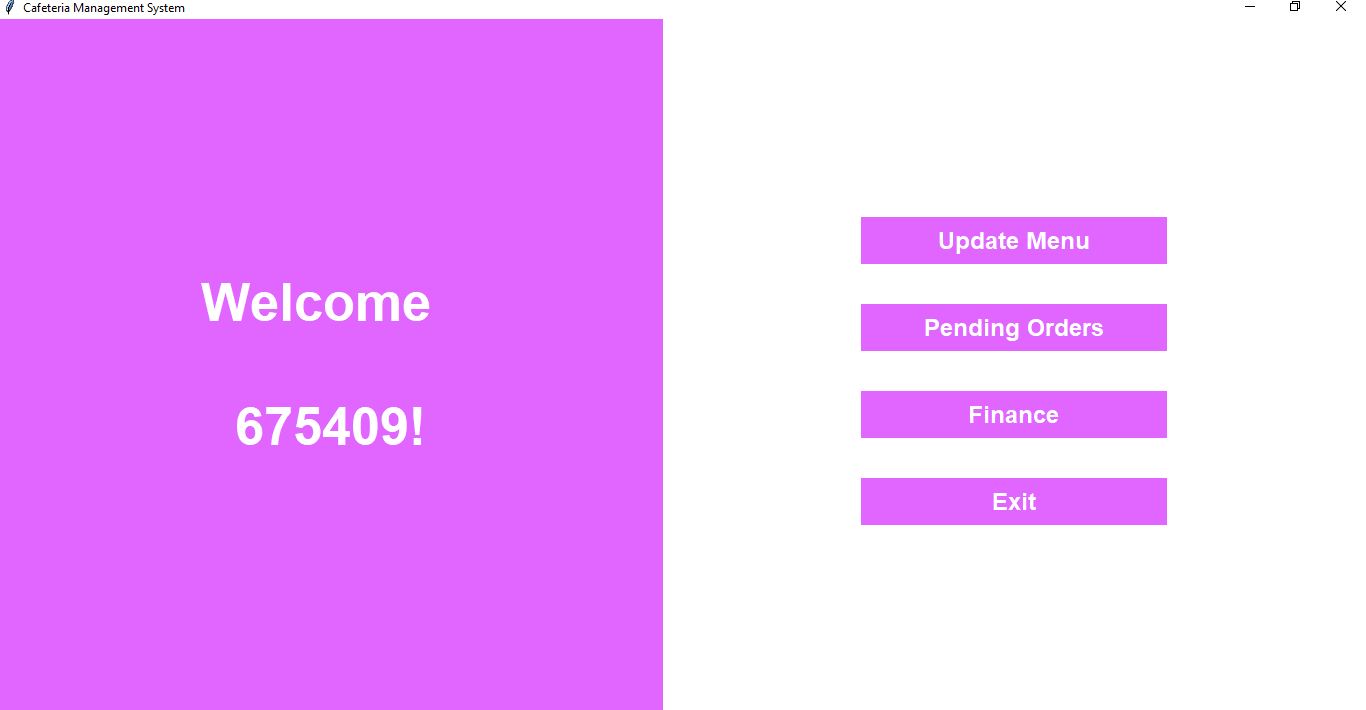
**HOME PAGE:**



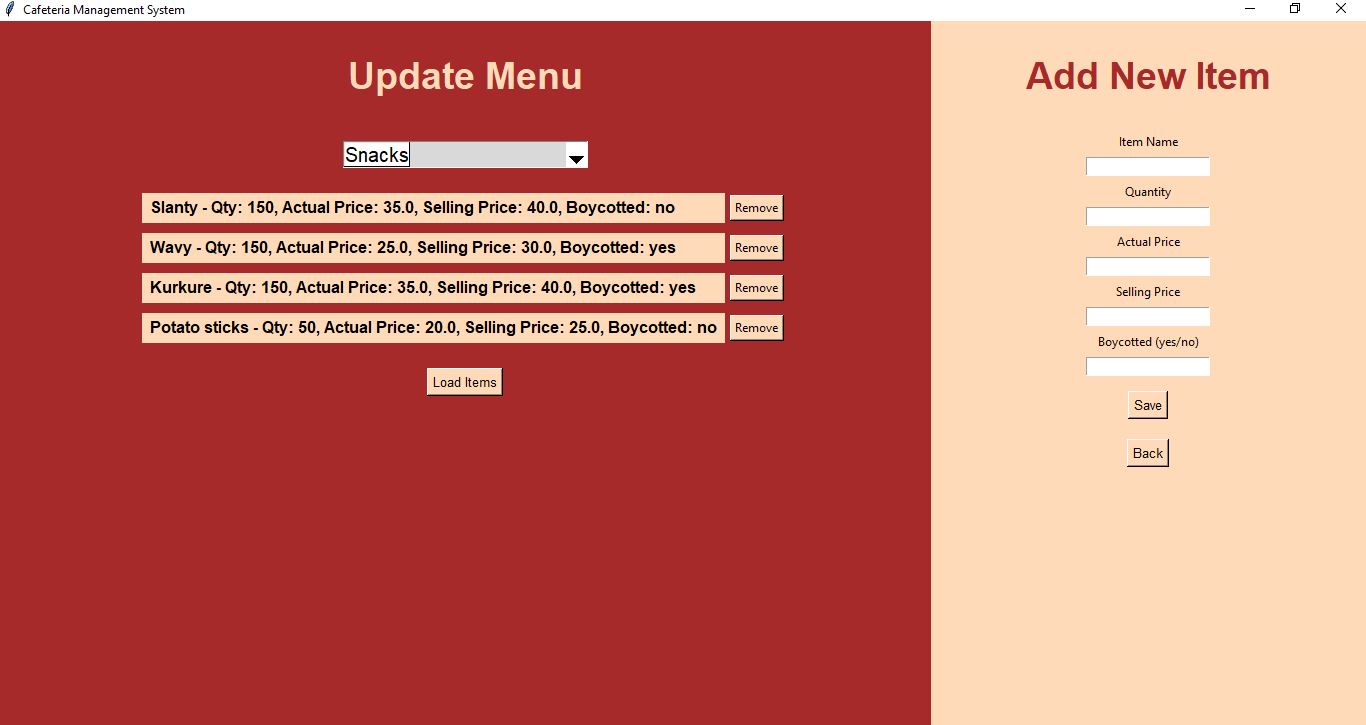
**Shopkeeper Login:**



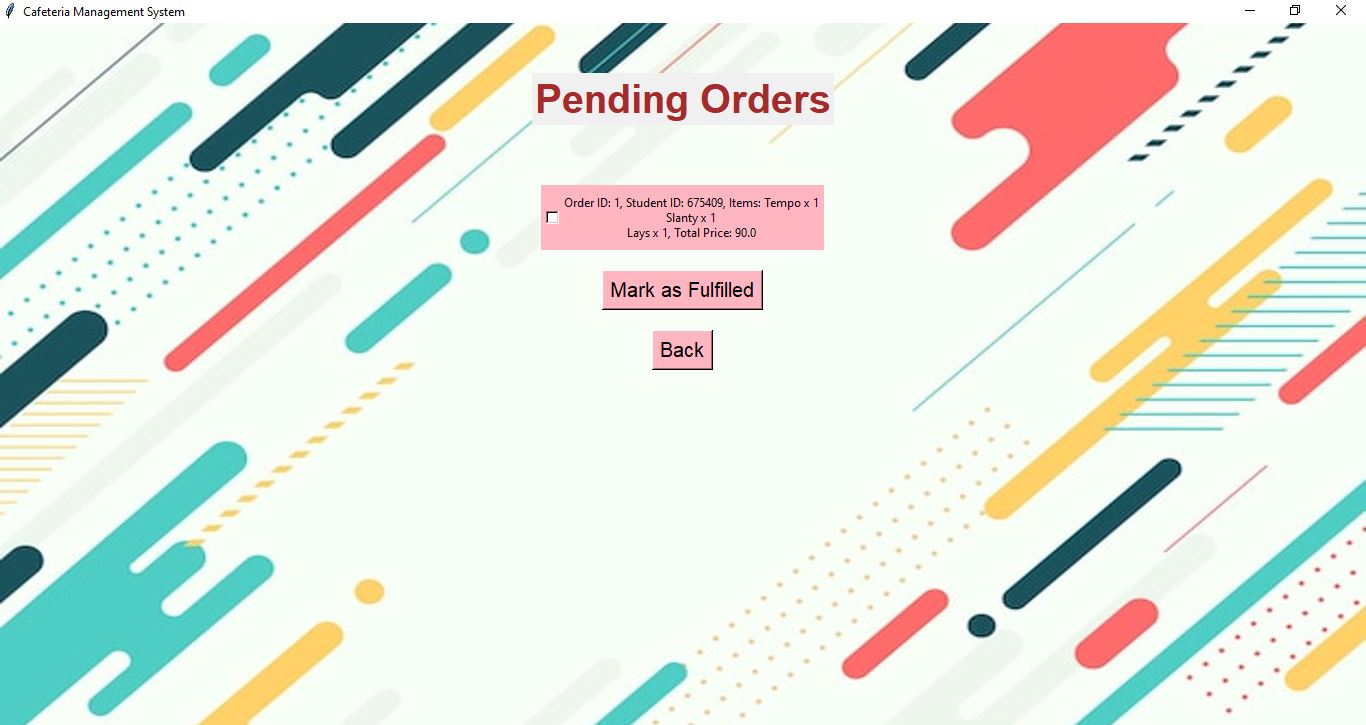
**Shopkeeper Homepage:**



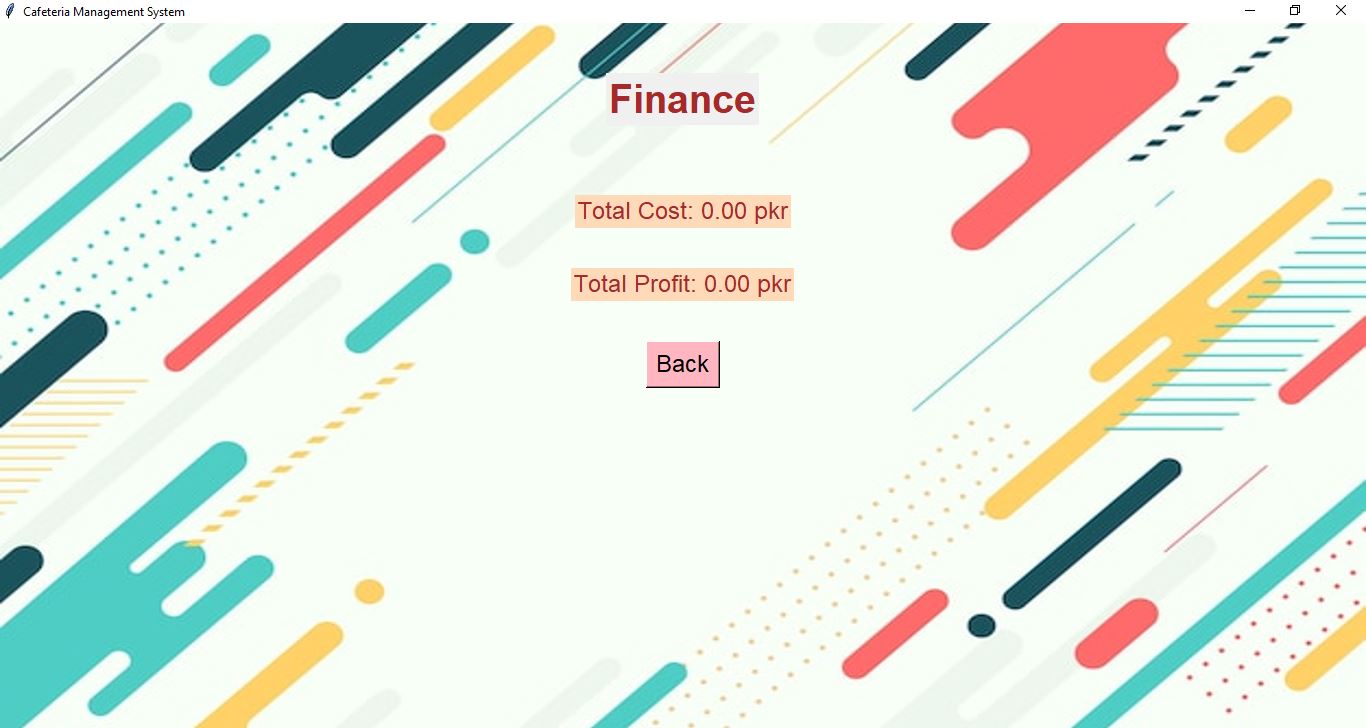
**Update Menu:**



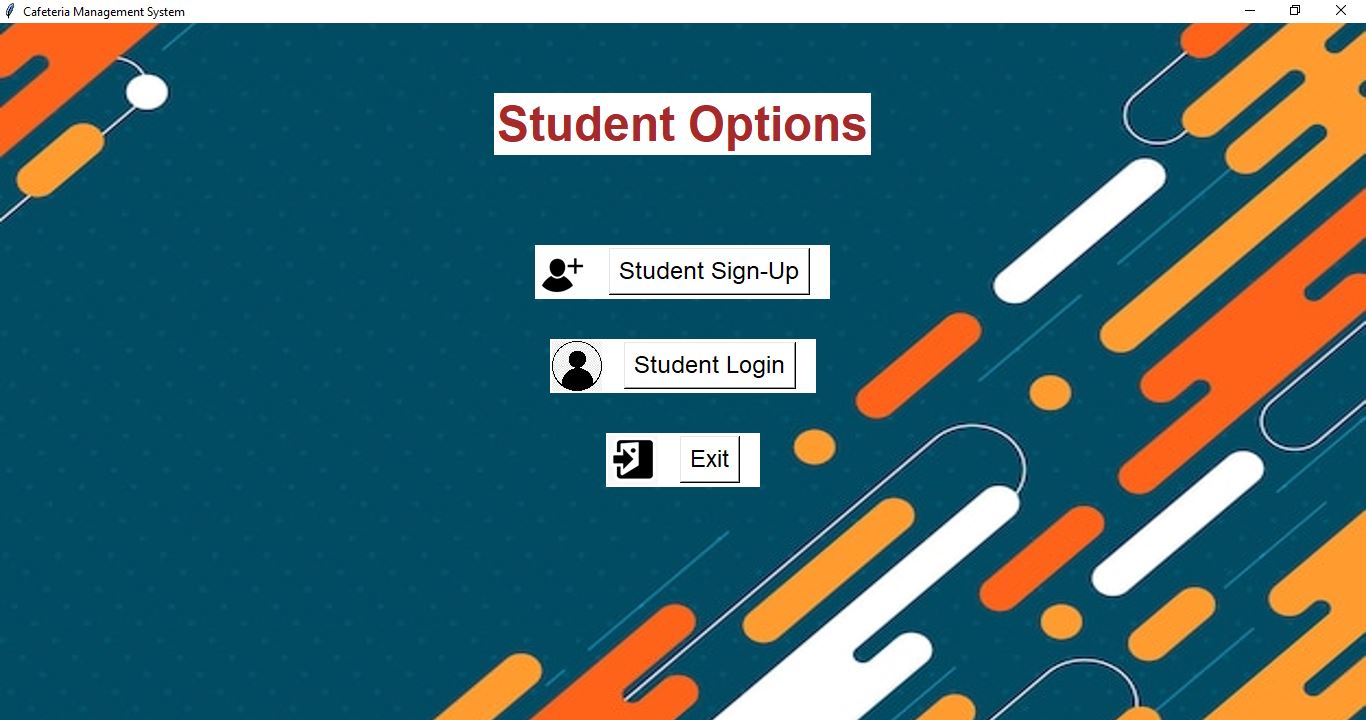
**Pending orders:**



**Finance:**



**Student Login page:**



**Sign up page:**