INDEX

PURPOSE OF THE PROJECT:

The primary goal of the online grocery project is to usher in a transformative era in the realm of traditional grocery shopping, fundamentally altering the way customers engage with the process. At its core, this initiative seeks to deliver a paradigm shift by providing a platform that is not just convenient and time-efficient, but also highly accessible for customers to procure essential items. By leveraging seamless digital interfaces, the project aims to create an experience that is intuitive and user-friendly, meeting the demands of an increasingly tech-savvy consumer base. Central to the project's objectives is the enhancement of user satisfaction. The digital platform is meticulously designed to cater to the preferences and expectations of modern consumers, offering a more personalized and efficient shopping experience. Through features such as personalized recommendations based on past purchases and preferences, the project aims to create a sense of convenience and satisfaction, fostering customer loyalty in the process.

Beyond the consumer-facing aspects, the online grocery project addresses critical logistical challenges inherent in traditional supply chain models. Through the optimization of inventory management, the project aims to strike a balance that prevents both stockouts and overstocking, contributing to operational efficiency and cost-effectiveness. This not only benefits the retailers but also ensures a more reliable and consistent service for the end consumer.

Furthermore, the project envisions a streamlined supply chain, leveraging digital tools to enhance the overall efficiency of the process. From order placement to last-mile delivery, every stage of the supply chain undergoes optimization to minimize delays and errors. This comprehensive approach not only aligns with the evolving preferences of consumers but also sets the stage for the broader integration of technology into the grocery retail landscape.

In essence, the online grocery project is not merely about digitizing the shopping experience; it is a strategic endeavour to revolutionize the entire ecosystem, offering a modern, responsive, and reliable solution that mirrors the changing expectations of consumers in today's fast-paced and technology-driven world.

MODULES AND FUNCTIONS

1.My SQL connector:

The MySQL. Connector module is a Python driver for connecting to MySQL databases. It provides a set of classes and methods to interact with MySQL databases, allowing you to perform various database operations using Python scripts.

2.TKINTER:

The tkinter module in Python is a standard GUI (Graphical User Interface) toolkit that provides tools for creating desktop applications with graphical interfaces. It's commonly used for building applications with buttons, labels, entry fields, and other GUI elements.

3. MESSAGE BOX:

The message Box module provides a simple way to create and display message boxes for showing information, asking questions, or displaying warnings and errors.
It contains functions like showinfo(), showwarning(), showerror(), askquestion(), askyesno(), etc., each serving a specific purpose.

4.PIL

The from PIL import Image, ImageTk statement is used to import the Image class and ImageTk module from the Python Imaging Library (PIL). These modules are commonly used for working with images in graphical applications, such as loading and manipulating images, as well as displaying them using Tkinter.

5.Listbox

The from tkinter import List box statement is used to import the List box class from the Tkinter module in Python. This class is employed to create a list Box widget, which is a box that can display a list of items that users can select.

6.REGULAR EXPRESSION

The import re statement is used to import the regular expression module, re, in Python. This module provides support for regular expressions, allowing you to perform pattern matching and manipulation of strings based on specified patterns.

7.PYGAME

The import pygame statement is used to import the Pygame library in Python. Pygame is a set of Python modules designed for writing video games.

8.PANDAS

The import pandas as pd statement are used to import the Pandas library in Python. Pandas is a powerful data manipulation and analysis library.

9.MATPLOTLIB

The import matplotlib.pyplot as plt statement is used to import the Matplotlib library's pyplot module in Python. Matplotlib is a plotting library that enables the creation of a wide variety of static, animated, and interactive plots.

15.RANDOM

The import random statement is used to import the random module in Python, allowing for the generation of random numbers.

10.NUMPY

The import NumPy as np statement is used to import the NumPy library in Python. NumPy is a powerful library for numerical computing, providing support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these arrays.

13.OS

The import os statement is used to import the os module in Python, providing a way to interact with the operating system.

14.MATH

The import math statement is used to import the math module in Python, providing a set of mathematical functions.

16.DATE TIME

The from datetime import datetime statement is used to import the datetime class from the datetime module in Python, allowing for working with date and time.

17.PARTIAL STATEMENT

The from functools import partial statement is used to import the partial function from the functools module in Python. The partial function is used to create partially applied functions, allowing you to fix certain arguments of a function and generate a new function with the remaining arguments

SQL TABLES

1.CREDENTIALS

credentials (5r × 4c)

username	PASSWORD	email	security_question
ALBENA	Kumar*123	test	ps senior
Chakradhar	Abcdefg?	cvb	mango
Dakshen	Dak!56789	dak12@gmail.com	orange
Hamdan	Iamgrat%*	greatji@gmail.com	ps senior
Kumar	Hi12345*	greatji@gmail.com	ps senior

COLUMNS (4r × 6c)

Field	Type	Null	Key
username	varchar(50)	NO	PRI
PASSWORD	varchar(40)	YES	
email	varchar(60)	YES	
security_question	varchar(170)	YES	

2.ITEMS

items (25r × 8c)

itemno	item_name	quantity	unit_price	expiry_date	sale_price	gst	net_price
1,000	bru	4	150	2025-11-15	200	5	210
1,001	Lipton green tea	3,450	100	2045-12-30	200	2	204
1,002	sugarfree	64	100	2025-05-05	110	7	117.7
1,003	Prime	15	500	2023-12-22	1,000	8	1,080
1,004	carrot	93	100	2025-01-05	110	7	117.7
1,005	AAVIN GHEE 1L	98	100	2025-05-05	110	6	116.6
1,006	orange	20	50	2023-12-30	57	2	59.85
1,007	AACHI GARAM MASALA	40	20	2025-08-08	30	5	31.5
1,008	lays	12	9	2024-11-23	10	5	10.5
1,009	A4 SHEET	3,440	0.5	2045-12-30	1	2	1.02
1,010	Harpic blue	100	100	2025-11-21	200	3	206
1,011	ajanata beetel nuts	20	0.75	2024-11-10	1	2	1.02
1,012	UNOMAX PAPERGLIDE	80	10	2026-11-20	20	1	20.2
1,013	PERK	100	5	2026-11-21	10	0	10
1,014	RR PONNI RICE	98	47	2027-11-29	65	1	65.65
1,015	VATIKA HAIR OIL	97	100	2026-11-23	120	2	122.4
1,016	IRRUTU KADAI HALWA	200	80	2021-07-20	100	4	104

COLUMNS (8r × 6c)

Field	Type	Null	Key
itemno	int(11)	NO	PRI
item_name	varchar(255)	YES	
quantity	int(11)	YES	
unit_price	float	YES	
expiry_date	date	YES	
sale_price	float	YES	
gst	float	YES	
net_price	float	YES	

3.BILLS

BILLS (16r × 13c)

bill_id	customer_name	mode_of_payment	item_no	item_name	unit_price	quantity	total_price	phone_number	address	purcha...	gst	net_price
1	CHAKRADHAR	card	1,024	apple	150	10	1,500	8144616142	MYLAPORE	100	2	1,530
2	ALBEN	card	1,012	UNOMAX PAPERGLIDE	20	10	200	8144616142	GUINDY	10	1	202
3	HAMDAN	card	1,020	PILLSBURRY ATTA	130	10	1,300	8144616142	SAIDAPET	100	4	1,352
4	KUMAR	card	1,021	MAGGI	20	10	200	8144616142	PORUR	15	4	208
5	DAKSHAN	card	1,023	FANATA 1L	35	1	35	8144616142	TEYNAMPET	30	3	36
6	RAHUL	card	1,013	PERK	10	10	100	8144616142	MINJUR	5	0	100
7	ABISHEK	card	1,014	RR PONNI RICE	65	10	650	8144616142	Guduvancerry	47	1	657
8	SATHAPPAN	card	1,015	VATIKA HAIR OIL	120	10	1,200	8144616142	Velacherry	100	2	1,224
9	SAJJAN	card	1,019	CELLO BUTTTERFLOW P...	10	10	100	8144616142	ADAYAR	7	4	104
10	Yuvan	cash	1,000	bru	200	2	400	9841224243	MYLAPORE	150	5	420
11	Shashank	cash	1,007	AACHI GARAM MASALA	30	5	150	9841224243	MYLAPORE	20	5	158
12	pranav rajesh	cash	1,021	MAGGI	20	5	100	9841224243	Mambalam	15	4	104
13	Aswin	cash	1,021	MAGGI	20	5	100	9841224243	T NAGAR	15	4	104
14	lakshana	cash	1,022	PONDS DREAM FLOWE...	10	5	50	9841224243	TAMBARAM	9	2	51
15	PAVISHIYA	cash	1,025	BRIL INK	25	5	125	9841224243	THARAMANI	20	2	128
16	Sanjana	cash	1,025	BRIL INK	25	5	125	9841224243	TRIPLICANE	20	2	128

COLUMNS (13r × 6c)

Field	Type	Null	Key	Default	Extra
bill_id	int(11)	NO	PRI	(NULL)	auto_increment
customer_name	varchar(255)	YES		(NULL)	
mode_of_payment	varchar(255)	YES		(NULL)	
item_no	int(11)	YES		(NULL)	
item_name	varchar(255)	YES		(NULL)	
unit_price	float	YES		(NULL)	
quantity	int(11)	YES		(NULL)	
total_price	float	YES		(NULL)	
phone_number	varchar(10)	YES		(NULL)	
address	varchar(800)	YES		(NULL)	
purchase_price	int(11)	YES		(NULL)	
gst	int(11)	YES		(NULL)	
net_price	int(11)	YES		(NULL)	

INTRODUCTION

Online grocery management is a dynamic and transformative approach to the traditional grocery shopping model, leveraging digital technologies to enhance every aspect of the customer experience. At its core, this entails the meticulous organization of inventory, orders, and deliveries through sophisticated digital platforms. Customers, in turn, benefit from the convenience of browsing, selecting, and purchasing groceries from the comfort of their homes, marking a significant departure from the conventional brick-and-mortar shopping experience. E-commerce platforms play a pivotal role in facilitating this seamless transition to online grocery shopping. These platforms serve as the virtual storefronts where customers can explore a diverse range of products, compare prices, and make informed choices. The accessibility and user-friendly interfaces of these platforms contribute to an overall efficient and enjoyable shopping experience.

Retailers, on the other hand, implement advanced management systems to ensure the optimization of stock levels and streamlined logistics. Utilizing data analytics, they gain valuable insights into customer preferences, enabling them to tailor offerings and promotions to individual needs. These management systems also play a crucial role in inventory control, reducing instances of stockouts or overstocking, thereby enhancing operational efficiency.

The emphasis on customer satisfaction is a cornerstone of online grocery management. By offering a convenient and efficient shopping experience, businesses aim to build loyalty and trust among their customer bases. Real-time order tracking, personalized recommendations, and responsive customer support further contribute to an enhanced overall service.

In essence, online grocery management represents a paradigm shift in the retail landscape, blending technological innovation with the practicality of daily life. As these systems continue to evolve, incorporating emerging technologies such as artificial intelligence and the Internet of Things, the online grocery experience is poised to become even more personalized, efficient, and integral to modern living

MAIN PROGRAM

```
import mysql.connector
import tkinter as tk
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
from tkinter import Listbox
from tkinter import *
import re
import pygame
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import os
import math
import random
from datetime import datetime
from functools import partial




conn = mysql.connector.connect(
    host='localhost',
```

```
    user='root',
    password='root',
    database='project'  # Use the 'project' database
)

# Create a cursor object
cursor = conn.cursor()

# Create the 'items' table if it doesn't exist
cursor.execute('''
    CREATE TABLE IF NOT EXISTS items (
        itemno INT AUTO_INCREMENT PRIMARY KEY,
        item_name VARCHAR(255),
        quantity INT,
        unit_price FLOAT,
        expiry_date DATE,
        sale_price FLOAT,
        gst FLOAT,
        net_price FLOAT
    )
''')

root = tk.Tk()
root.title("Grocery Store Management")
root.geometry("1024x1024")
```

```
# Load a background image (you should replace 'xyz.png' with your actual image path)
bg_image = tk.PhotoImage(file=r"C:\Users\babu\Desktop\project\intro.png")
bg_label = tk.Label(root, image=bg_image)
bg_label.place(relwidth=1, relheight=1)
pygame.init()

# Load the music file
pygame.mixer.music.load(r"C:\Users\babu\Desktop\project\welcome.mp3")

# Play the music
pygame.mixer.music.play()



# Define a common style for widgets
common_style = {"font": ("Arial", 25), "bg": "black","fg":"white", "padx": 10, "pady": 10}

frames = []

# Function to log in as an administrator
def login_administrator():
    username = username_entry.get()
    password = password_entry.get()
```

```
    if username == "admin" and password == "admin":
        messagebox.showinfo("Success", "Administrator logged in
successfully.")
        show_admin_menu()
    else:
        messagebox.showinfo("Error", "Invalid username or password.")
username_entry=None
password_entry=None




# Function to log in as a customer
customer_username_entry = None
customer_password_entry = None
customer_email_entry = None
security_question_entry = None

    # Add logic to authenticate customers here


# Function to sign up as a customer
def sign_up_customer():
    global customer_username_entry
    customer_username = customer_username_entry.get()

    customer_password = customer_password_entry.get()
    customer_email = customer_email_entry.get()
```

```
    sec_question = security_question_entry.get()


    # Check if the username meets the conditions
    if not re.search(r'[A-Z]', customer_username):
        messagebox.showinfo("Error", "Username must contain at least one capital letter.")
        return


    # Check if the password meets the conditions
    if not (re.search(r'[A-Z]', customer_password) and re.search(r'[!@#$%^&*?<>/.,=+_-]', customer_password) and len(customer_password) >= 8):
        messagebox.showinfo("Error", "Password must have at least one capital letter, one special character, and be at least 8 characters long.")
        return


    # Check if the email is not empty
    if not customer_email:
        messagebox.showinfo("Error", "Please fill in the email field.")
        return
    if not sec_question:
        messagebox.showinfo("Error", "Please fill in the security question field.")
        return




    # Check if the username is unique
```

```
    cursor.execute("SELECT * FROM credentials WHERE USERNAME =
%s", (customer_username,))

    existing_user = cursor.fetchone()

    if existing_user:

        messagebox.showinfo("Error", "Username already exists. Please
choose a different username.")

        return


    # If all conditions are met, insert customer data into the database

    cursor.execute("INSERT INTO credentials (USERNAME,
PASSWORD, EMAIL,security_question) VALUES (%s, %s, %s,%s)",

                   (customer_username, customer_password,
customer_email,sec_question))

    conn.commit()


    messagebox.showinfo("Success", "Account created successfully. You
can now log in as a customer.")

def login_customer():

    global customer_username

    global customer_password

    customer_username = customer_username_entry.get()

    customer_password = customer_password_entry.get()


    # Check the credentials against the SQL database

    cursor.execute("SELECT * FROM credentials WHERE USERNAME =
%s AND PASSWORD = %s", (customer_username,
customer_password))

    result = cursor.fetchone()


    if result:
```

```
        messagebox.showinfo("Success", "Customer logged in
successfully.")
        show_customer_menu()
    else:
        messagebox.showinfo("Error", "Invalid username or password.")
email_entry=None
new_password_entry=None




security1_question_entry=None
def forgot_password():
    def reset():
        customer_username = customer_username_entry.get()
        sec1_question = security1_question_entry.get()
        customer_password = new_password_entry.get()


        cursor.execute("SELECT username FROM credentials")
        result = cursor.fetchall()


        for i in result:
            if i[0] == customer_username:
                cursor.execute("SELECT security_question FROM credentials
WHERE username = %s", (customer_username,))
                result1 = cursor.fetchall()
                for j in result1:
                    if j[0] == sec1_question:
```

```
                    if len(customer_password) < 8 or not any(char.isupper() for char in customer_password) or not any(char.isdigit() for char in customer_password) or not any(char.isalnum() or char in '!@#$%^&*()_+' for char in customer_password):
                        messagebox.showinfo("Error", "Invalid password format. Password must have at least 8 characters with at least one uppercase letter, one digit, and one special character.")
                    else:
                        cursor.execute("UPDATE credentials SET password = %s WHERE username = %s", (customer_password, customer_username))
                        messagebox.showinfo("Success", "Password reset success. You can now log in as a customer.")
                        root2.destroy()
                        return
                else:
                    messagebox.showinfo("Error", "Incorrect security question answer.")
                    return
        messagebox.showinfo("Error", "Invalid username.")


    # GUI setup
    root2 = tk.Tk()
    root2.title("Password Reset")


    username_label = tk.Label(root2, text="Username")
    username_label.grid(row=0, column=0)
    customer_username_entry = tk.Entry(root2)
    customer_username_entry.grid(row=0, column=1)
```

```
    sec1_label = tk.Label(root2, text="enter your school studied or enter your favourite fruit")
    sec1_label.grid(row=1, column=0)
    security1_question_entry = tk.Entry(root2)
    security1_question_entry.grid(row=1, column=1)


    new_password_label = tk.Label(root2, text="New Password")
    new_password_label.grid(row=2, column=0)
    new_password_entry = tk.Entry(root2)
    new_password_entry.grid(row=2, column=1)


    reset_button = tk.Button(root2, text="Reset", command=reset)
    reset_button.grid(row=3, column=0, columnspan=2)

# You may want to start the Tkinter main loop




# You may want to start the Tkinter main loop
```

#

```
admin_login_frame=None
def admin_login():
    global username_entry
    global password_entry
    admin_login_frame = tk.Frame(root, bg="white", bd=5)
    admin_login_frame.place(relx=0.5, rely=0.5, relwidth=0.2,
relheight=0.2, anchor=tk.CENTER)



    username_label = tk.Label(admin_login_frame, text="Username:",
bg="white")
    username_label.grid(row=0, column=0)
    username_entry = tk.Entry(admin_login_frame)
    username_entry.grid(row=0, column=1)


    password_label = tk.Label(admin_login_frame, text="Password:",
bg="white")
    password_label.grid(row=1, column=0)
```

```
    password_entry = tk.Entry(admin_login_frame, show="*")
    password_entry.grid(row=1, column=1)

    login_button = tk.Button(admin_login_frame, text="Login as Admin", command=login_administrator)
    login_button.grid(row=2, columnspan=2)

# Create widgets for logging in as a customer
customer_login_frame=None
def customer_signin():
    global customer_login_frame, customer_username_entry, customer_password_entry  # Updated variable names
    customer_login_frame = tk.Frame(root, bg="blue", bd=10)
    customer_login_frame.place(relx=0.5, rely=0.2, relwidth=0.4, relheight=0.2, anchor="n")

    # Create the entry widgets
    customer_username_entry = tk.Entry(customer_login_frame)
    customer_username_entry.grid(row=0, column=1)

    customer_password_entry = tk.Entry(customer_login_frame, show="*")
    customer_password_entry.grid(row=1, column=1)

    customer_login_button = tk.Button(customer_login_frame, text="Login as Customer", command=login_customer)
```

```
    customer_login_button.grid(row=2, columnspan=2)


    customer_username_label = tk.Label(customer_login_frame,
text="Username:", bg="white")
    customer_username_label.grid(row=0, column=0)


    customer_password_label = tk.Label(customer_login_frame,
text="Password:", bg="white")
    customer_password_label.grid(row=1, column=0)


    forgot_pswd = tk.Button(customer_login_frame, text="forgot
password", command=forgot_password)
    forgot_pswd.grid(row=3, columnspan=2)




# Define global variables for customer entry fields
new_gst_entry=None
def customer_signup():
    global customer_username_entry, customer_password_entry,
customer_email_entry, security_question_entry  # Updated variable
names
    customer_sign_up_frame = tk.Frame(root, bg="white", bd=5)
    customer_sign_up_frame.place(relx=0.5, rely=0.8, relwidth=0.4,
relheight=0.2, anchor="n")


    customer_signup_username_label =
tk.Label(customer_sign_up_frame, text="Username:", bg="white")
    customer_signup_username_label.grid(row=0, column=0)
    customer_username_entry = tk.Entry(customer_sign_up_frame)
```

```
    customer_username_entry.grid(row=0, column=1)


    customer_signup_password_label =
tk.Label(customer_sign_up_frame, text="Password:", bg="white")
    customer_signup_password_label.grid(row=1, column=0)
    customer_password_entry = tk.Entry(customer_sign_up_frame,
show="*")
    customer_password_entry.grid(row=1, column=1)


    customer_email_label = tk.Label(customer_sign_up_frame,
text="Email:", bg="white")
    customer_email_label.grid(row=2, column=0)
    customer_email_entry = tk.Entry(customer_sign_up_frame)
    customer_email_entry.grid(row=2, column=1)


    security_question_label = tk.Label(customer_sign_up_frame,
text="enter your school studied or enter your favourite fruit", bg="white")
    security_question_label.grid(row=3, column=0)
    security_question_entry = tk.Entry(customer_sign_up_frame)
    security_question_entry.grid(row=3, column=1)


    sign_up_button = tk.Button(customer_sign_up_frame, text="Sign Up
as Customer", command=sign_up_customer)
    sign_up_button.grid(row=4, columnspan=2)


    sign_up_button = tk.Button(customer_sign_up_frame, text="Sign Up
as Customer", command=sign_up_customer)
    sign_up_button.grid(row=4, columnspan=2)
```

```
login_frame = ttk.Frame(root)
login_frame.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

b1 = tk.Button(login_frame, text="Administrator Login",
command=admin_login, **common_style)
b2 = tk.Button(login_frame, text="Customer Login",
command=customer_signin, **common_style)
b3 = tk.Button(login_frame, text="Customer Signup",
command=customer_signup, **common_style)

b1.pack(pady=60)
b2.pack(pady=60)
b3.pack(pady=60)

gst_entry=None
# Create widgets for the administrator's menu
def show_admin_menu():



    # Initialize pygame
     pygame.init()

    # Load the music file
pygame.mixer.music.load(r"C:\Users\babu\Desktop\project\music.mp3")
```

```
    # Play the music
    pygame.mixer.music.play()

    global admin_login_frame
    if admin_login_frame is not None:
        admin_login_frame.destroy()

    login_frame.destroy() # Destroy the login frame
    notebook = ttk.Notebook(root)
    notebook.pack(fill='both', expand='yes')

    # Create widgets for adding items
    def add_item():

        item_name = item_name_entry.get()
        quantity = quantity_entry.get()
        unit_price = unit_price_entry.get()
        item_no = item_no_entry.get()
        exp_date = exp_date_entry.get()
        sale_price = sale_price_entry.get()
        gst=gst_entry.get()

        query = "INSERT INTO items (itemno, item_name, quantity,
unit_price, expiry_date, sale_price, gst) VALUES (%s, %s, %s, %s, %s,
%s, %s)"
        values = (item_no, item_name, quantity, unit_price, exp_date,
sale_price,gst)
```

```
        try:
            print("Query:", query)
            print("Values:", values)
            cursor.execute(query, values)
            conn.commit()
            cursor.execute("update items SET
net_price=sale_price+sale_price*gst/100")
            conn.commit()
            messagebox.showinfo("Success", "Item added successfully.")
        except mysql.connector.IntegrityError:
            messagebox.showinfo("Warning", "Item already exists. ")



    add_item_frame = ttk.Frame(root)
    notebook.add(add_item_frame, text='Add Item')


    image = PhotoImage(file=r"C:\users\babu\Desktop\project\add
entry.png")


# Add a label to display the image
    image_label = ttk.Label(add_item_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image


# Create a custom style for labels, entry widgets, and the button
    custom_style = ttk.Style()
```

```
    custom_style.configure("Custom.TLabel", font=("Bernard MT condensed", 30))

    custom_style.configure("Custom.TEntry", font=("Bernard MT condensed", 30))

    custom_style.configure("Custom.TButton", font=("Bernard MT condensed", 30), background="black",foreground="green")


# Create labels with the custom style

    item_name_label = ttk.Label(add_item_frame, text="Product Name:", style="Custom.TLabel")

    item_name_label.grid(row=0, column=0, padx=10, pady=5)

    item_name_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    item_name_entry.grid(row=0, column=1, padx=10, pady=5)


    quantity_label = ttk.Label(add_item_frame, text="Quantity:", style="Custom.TLabel")

    quantity_label.grid(row=1, column=0, padx=10, pady=5)

    quantity_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    quantity_entry.grid(row=1, column=1, padx=10, pady=5)


    unit_price_label = ttk.Label(add_item_frame, text="Unit Price:", style="Custom.TLabel")

    unit_price_label.grid(row=2, column=0, padx=10, pady=5)

    unit_price_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    unit_price_entry.grid(row=2, column=1, padx=10, pady=5)


    item_no_label = ttk.Label(add_item_frame, text="Item No:", style="Custom.TLabel")
```

```
    item_no_label.grid(row=3, column=0, padx=10, pady=5)

    item_no_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    item_no_entry.grid(row=3, column=1, padx=10, pady=5)



    exp_date_label = ttk.Label(add_item_frame, text="Expiry Date (yy/mm/dd):", style="Custom.TLabel")

    exp_date_label.grid(row=4, column=0, padx=10, pady=5)

    exp_date_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    exp_date_entry.grid(row=4, column=1, padx=10, pady=5)



    sale_price_label = ttk.Label(add_item_frame, text="Sale Price:", style="Custom.TLabel")

    sale_price_label.grid(row=5, column=0, padx=10, pady=5)

    sale_price_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    sale_price_entry.grid(row=5, column=1, padx=10, pady=5)



    gst_label = ttk.Label(add_item_frame, text="GST:", style="Custom.TLabel")

    gst_label.grid(row=6, column=0, padx=10, pady=5)

    gst_entry = ttk.Entry(add_item_frame, style="Custom.TEntry", width=30)  # Increase entry width

    gst_entry.grid(row=6, column=1, padx=10, pady=5)




# Create and configure the "Add Item" button

    add_button = ttk.Button(add_item_frame, text="Add Item", command=add_item, style="Custom.TButton")
```

```
    add_button.grid(row=7, columnspan=2, pady=10)


    # Create widgets for showing stock
    stock_treeview = ttk.Treeview(notebook, columns=("itemno",
"item_name", "quantity", "unit_price", "expiry_date",
"sale_price","gst","net_price"))
    notebook.add(stock_treeview, text='Show Stock')
    button_frame = ttk.Frame(stock_treeview)
    button_frame.grid(row=0, column=0, pady=10)



    # Define columns for the table
    stock_treeview.heading("#1", text="Item No")
    stock_treeview.heading("#2", text="Item Name")
    stock_treeview.heading("#3", text="Quantity")
    stock_treeview.heading("#4", text="Unit Price")
    stock_treeview.heading("#5", text="Expiry Date")
    stock_treeview.heading("#6", text="Sale Price")
    stock_treeview.heading("#7", text="gst")
    stock_treeview.heading("#8", text="net_Price")



    # Set column widths
    stock_treeview.column("#1", width=80)
    stock_treeview.column("#2", width=100)
    stock_treeview.column("#3", width=100)
    stock_treeview.column("#4", width=100)
```

```
    stock_treeview.column("#5", width=50)
    stock_treeview.column("#6", width=50)
    stock_treeview.column("#7", width=100)
    stock_treeview.column("#8", width=100)

    # Function to populate the table with stock data
    def show_stock():

        stock_treeview.delete(*stock_treeview.get_children())  # Clear the table
        cursor.execute("SELECT * FROM items")
        result = cursor.fetchall()
        for row in result:
            stock_treeview.insert("", "end", values=row)




    show_stock_button = ttk.Button(button_frame, text="Show Stock", command=show_stock)
    show_stock_button.grid(row=0, column=0, pady=10)

    style = ttk.Style()
    style.configure("Treeview.Heading", font=("Helvetica", 6))  # Increase the font size (12) as needed

# Increase the font size for cell values
      # Increase the font size (10) as needed
```

```
# Set the background color for the Treeview
    style.configure("Treeview", background="#D9E4FF")  # Change the color code to the color you want




     # Function to remove items from the database
    style_options = {
        'font': ('Bernard MT Bold', 30),
        'background': 'white'}
    bill_treeview = ttk.Treeview(notebook, columns=("bill_id", "customer name","mode of payment", "item_no", "item_name", "unit_price", "unit_price", "sale_price", "phone_number", "address","purchase_price","gst","net_price"))
    notebook.add(bill_treeview, text='Show billed items')
    button_frame = ttk.Frame(bill_treeview)
    button_frame.grid(row=0, column=0, pady=10)


    bill_treeview.heading("#1", text="Bill No")
    bill_treeview.heading("#2", text="customer name")
    bill_treeview.heading("#3", text="mode of payment")
    bill_treeview.heading("#4", text="Item No")
    bill_treeview.heading("#5", text="Item Name")
    bill_treeview.heading("#6", text="unit_price")


    bill_treeview.heading("#7", text="quantity")
```

```
bill_treeview.heading("#8", text="Sale Price")
bill_treeview.heading("#9", text="Phone Number")
bill_treeview.heading("#10", text="Address")
bill_treeview.heading('#11',text='purchase_price')
bill_treeview.heading('#12',text='gst')
bill_treeview.heading('#13',text='net_price')



# Set column widths
bill_treeview.column("#1", width=50)
bill_treeview.column("#2", width=150)
bill_treeview.column("#3", width=80)
bill_treeview.column("#4", width=150)
bill_treeview.column("#5", width=60)
bill_treeview.column("#6", width=80)
bill_treeview.column("#7", width=100)
bill_treeview.column("#8", width=80)
bill_treeview.column("#9", width=100)
bill_treeview.column("#10", width=150)
bill_treeview.column("#11", width=30)
bill_treeview.column("#12", width=100)
bill_treeview.column("#13", width=100)



def show_bills():
    bill_treeview.delete(*bill_treeview.get_children())  # Clear the table
```

```
        cursor.execute("SELECT * FROM bills")
        result = cursor.fetchall()

        for row in result:
            bill_treeview.insert("", "end", values=row)

    show_stock_button2 = ttk.Button(button_frame, text="Show Bills",
command=show_bills)
    show_stock_button2.grid(row=1, column=0, pady=10)




    def remove_item():



        item_no = int(item_no_remove_entry.get())
        qty = int(quantity_remove_entry.get())

        cursor.execute("SELECT quantity FROM items WHERE itemno =
%s", (item_no,))
        current_qty = cursor.fetchone()

        if current_qty and current_qty[0] >= qty:
            cursor.execute("UPDATE items SET quantity = quantity - %s
WHERE itemno = %s", (qty, item_no))
            conn.commit()
            messagebox.showinfo("Success", "Item removed successfully.")
        else:
```

```
            messagebox.showinfo("Warning", "Item does not exist or
quantity is insufficient.")


    remove_item_frame = ttk.Frame(root)
    notebook.add(remove_item_frame, text='Remove Item')



    image =
PhotoImage(file=r"C:\users\babu\Desktop\project\remove.png")


# Add a label to display the image
    image_label = ttk.Label(remove_item_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image


    item_no_remove_label = ttk.Label(remove_item_frame, text="Item No
to Remove", **style_options)
    item_no_remove_label.grid(row=0, column=0, padx=10, pady=5)
    item_no_remove_entry = ttk.Entry(remove_item_frame,
**style_options)
    item_no_remove_entry.grid(row=0, column=1, padx=10, pady=5)


    quantity_remove_label = ttk.Label(remove_item_frame, text="Quantity
to Remove", **style_options)
    quantity_remove_label.grid(row=1, column=0, padx=10, pady=5)
    quantity_remove_entry = ttk.Entry(remove_item_frame,
**style_options)
    quantity_remove_entry.grid(row=1, column=1, padx=10, pady=5)
```

```
    remove_button = ttk.Button(remove_item_frame, text="Remove Item",
command=remove_item, style="Custom.TButton")
    remove_button.grid(row=2, columnspan=2, pady=10)
    # Load the image


    def modify_item():
        item_no = item_no_modify_entry.get()
        new_quantity = new_quantity_entry.get()
        new_unit_price = new_unit_price_entry.get()
        new_exp_date = new_exp_date_entry.get()
        new_sale_price = new_sale_price_entry.get()
        new_gst=new_gst_entry.get()


        query = "UPDATE items SET quantity = %s, unit_price = %s,
expiry_date = %s, sale_price=%s,gst=%s WHERE itemno = %s"
        values = (new_quantity, new_unit_price, new_exp_date,
new_sale_price,new_gst, item_no)


        cursor.execute(query, values)
        conn.commit()
        messagebox.showinfo("Success", f"Item {item_no} modified
successfully.")


# Create widgets for modifying items
    modify_item_frame = ttk.Frame(root)
    notebook.add(modify_item_frame, text='Modify Item')
    xyz = ttk.Style()
    xyz.configure("custom.TLabel", font=("Bernard MT condensed", 30))
```

```
    xyz.configure("custom.TEntry", font=("Bernard MT condensed", 30))

    xyz.configure("custom.TButton", font=("Bernard MT condensed", 30),
background="black",foreground="blue")

    image =
PhotoImage(file=r"C:\users\babu\Desktop\project\modify.png")


# Add a label to display the image

    image_label = ttk.Label(modify_item_frame, image=image)

    image_label.place(x=0, y=0, relwidth=1, relheight=1)

    image_label.image = image


    item_no_modify_label = ttk.Label(modify_item_frame, text="Item No
to Modify:",style="Custom.TLabel")

    item_no_modify_label.grid(row=0, column=0, padx=10, pady=5)

    item_no_modify_entry =
ttk.Entry(modify_item_frame,style="custom.TEntry")

    item_no_modify_entry.grid(row=0, column=1, padx=10, pady=5)


    new_quantity_label = ttk.Label(modify_item_frame, text="New
Quantity:",style="Custom.TLabel")

    new_quantity_label.grid(row=1, column=0, padx=10, pady=5)

    new_quantity_entry =
ttk.Entry(modify_item_frame,style="custom.TEntry")

    new_quantity_entry.grid(row=1, column=1, padx=10, pady=5)


    new_unit_price_label = ttk.Label(modify_item_frame, text="New Unit
Price:",style="Custom.TLabel")

    new_unit_price_label.grid(row=2, column=0, padx=10, pady=5)

    new_unit_price_entry =
ttk.Entry(modify_item_frame,style="custom.TEntry")
```

```
    new_unit_price_entry.grid(row=2, column=1, padx=10, pady=5)


    new_exp_date_label = ttk.Label(modify_item_frame, text="New Expiry
Date (yy/mm/dd):",style="Custom.TLabel")
    new_exp_date_label.grid(row=3, column=0, padx=10, pady=5)
    new_exp_date_entry =
ttk.Entry(modify_item_frame,style="custom.TEntry")
    new_exp_date_entry.grid(row=3, column=1, padx=10, pady=5)


    new_sale_price_label = ttk.Label(modify_item_frame, text="New Sale
Price:",style="Custom.TLabel")
    new_sale_price_label.grid(row=4, column=0, padx=10, pady=5)
    new_sale_price_entry =
ttk.Entry(modify_item_frame,style="custom.TEntry")
    new_sale_price_entry.grid(row=4, column=1, padx=10, pady=5)


    new_gst_label = ttk.Label(modify_item_frame, text="New
gst:",style="Custom.TLabel")
    new_gst_label.grid(row=5, column=0, padx=10, pady=5)
    new_gst_entry = ttk.Entry(modify_item_frame,style="custom.TEntry")
    new_gst_entry.grid(row=5, column=1, padx=10, pady=5)


    modify_button = ttk.Button(modify_item_frame, text="Modify Item",
command=modify_item,style="custom.TButton")
    modify_button.grid(row=7, columnspan=2, pady=10)




        # Create widgets for graphing data
```

```
    def profit_earned():
        cursor
        cursor.execute("select sum(total_price)-
sum(purchase_price*quantity) from bills")
        result = cursor.fetchall()
        k=result[0][0]
        cursor.execute("select sum(unit_price*quantity )from items where
expiry_date<=curdate() ")
        result1 = cursor.fetchall()
        if result1 and result1[0][0] is not None:
           l = result1[0][0]
          else:
           l = 0



        profit=k-l

profit_display=ttk.Label(profit_frame,text=profit,style="custom.TLabel")
        profit_display.grid(row=4, column=0, padx=100, pady=100)




    profit_frame=ttk.Frame(root)
    notebook.add(profit_frame, text='profit earned')
    image = PhotoImage(file=r"C:\users\babu\Desktop\project\profit.png")

# Add a label to display the image
```

```
    image_label = ttk.Label(profit_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image
    ss=ttk.Button(profit_frame,text="show
profit",command=profit_earned,style="custom.TButton")
    ss.grid(row=5,pady=10,columnspan=2)



    xyz = ttk.Style()
    xyz.configure("custom.TLabel", font=("Bernard MT condensed", 30))
    xyz.configure("custom.TEntry", font=("Bernard MT condensed", 30))


    xyz.configure("custom.TButton", font=("Bernard MT condensed", 30),
background="black",)
    cursor.execute("SELECT SUM(quantity),item_name FROM bills
GROUP BY item_name ORDER BY item_name")
    result7 = cursor.fetchall()
    cursor.execute("SELECT DISTINCT item_name FROM bills ORDER
BY item_name")
    result8 = cursor.fetchall()
    sales = [item[0] for item in result7]
    items = [item[0] for item in result8]
    def graphs1():
        fig = plt.figure(figsize = (7,5))
        axes = fig.add_subplot(1,1,1)
        axes.set_ylim(0, 300)
        palette = ['blue', 'red', 'green',
           'darkorange', 'maroon', 'black']
```

```
    plt.bar(items,sales,width=0.3,color=palette)
    plt.title('Profit Earned')

    plt.xlabel("ITEMS")
    plt.ylabel("SALES")




    plt.show()



def graphs2():
    plt.plot(items,sales)

    plt.xlabel("ITEMS")
    plt.ylabel("SALES")
    plt.show()
def graphs3():
    plt.pie(sales, labels=items)
    plt.title('Sales')

    plt.show()
```

```
    graph_frame = ttk.Frame(root)
    notebook.add(graph_frame, text='Graph Data')
    image = PhotoImage(file=r"C:\users\babu\Desktop\project\graph.png")

# Add a label to display the image
    image_label = ttk.Label(graph_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image

    graph_button1=ttk.Button(graph_frame,text="Generate
bargraph",command=graphs1,style="custom.TButton")
    graph_button1.grid(row=0, columnspan=1, padx=150, pady=10)
    graph_button2=ttk.Button(graph_frame,text="Generate
linechart",command=graphs2,style="custom.TButton")
    graph_button2.grid(row=4,columnspan=2,padx=150,pady=70)
    graph_button3=ttk.Button(graph_frame,text="Generate
piechart",command=graphs3,style="custom.TButton")
    graph_button3.grid(row=8,columnspan=3,padx=150,pady=90)

    user_treeview = ttk.Treeview(notebook, columns=("username",
"password","email", "security question"))
    notebook.add(user_treeview, text='Show users')
    user_treeview.heading("#1", text="username")
    user_treeview.heading("#2", text="password")
```

```
    user_treeview.heading("#3", text="email")
    user_treeview.heading("#4", text="security_question")

    # Set column widths
    user_treeview.column("#1", width=80)
    user_treeview.column("#2", width=200)
    user_treeview.column("#3", width=100)
    user_treeview.column("#4", width=100)



    # Function to populate the table with stock data
    def show_users():
        user_treeview.delete(*user_treeview.get_children())  # Clear the table
        cursor.execute("SELECT * FROM credentials")
        result = cursor.fetchall()
        for row in result:
            user_treeview.insert("", "end", values=row)


    user_frame = ttk.Frame(user_treeview)
    user_frame.grid(row=0, column=0, pady=10)
    show_user_button = ttk.Button(user_frame, text="Show users", command=show_users)
    show_user_button.grid(row=1, column=0, pady=10)


    style = ttk.Style()
    style.configure("Treeview.Heading", font=("Helvetica", 18))
```

```
    # Increase the font size for cell values
    # Increase the font size (10) as needed

    # Set the background color for the Treeview
    style.configure("Treeview", background="#D9E4FF")  # Change the color code to the color you want

    delete_username_entry = None

    del_frame = ttk.Frame(root)
    notebook.add(del_frame, text='delete Users')
    xyzp= ttk.Style()
    xyzp.configure("custom.TLabel", font=("Bernard MT condensed", 30))
    xyzp.configure("custom.TEntry", font=("Bernard MT condensed", 30))

    xyzp.configure("custom.TButton", font=("Bernard MT condensed", 30), background="black",)



    def del1():
        delete_username = delete_username_entry.get()


        cursor.execute("DELETE FROM credentials WHERE username=%s", (delete_username,))
        conn.commit()
        messagebox.showinfo("deleted successfully")
    image = PhotoImage(file=r"C:\users\babu\Desktop\project\users.png")
```

```
# Add a label to display the image
    image_label = ttk.Label(del_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image


    delete_label = ttk.Label(del_frame, text="enter userid to be deleted", style="custom.TLabel")
    delete_username_entry = ttk.Entry(del_frame, style="custom.TEntry")
    delete_button = ttk.Button(del_frame, text="delete", command=del1, style="custom.TButton")
    delete_label.grid(row=1, column=1,padx=150,pady=50)
    delete_username_entry.grid(row=4, column=1,pady=100,padx=100)
    delete_button.grid(row=5, column=1,pady=150,padx=150)
```

```
    frames.append(admin_login_frame)
      # Add the admin menu frame to the frames list

# Create widgets for the customer portal
customer_menu_frame = None
login_entry = None
customer_phone_number_entry=None
customer_address_entry=None



def show_customer_menu():

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS bills (
        bill_id INT AUTO_INCREMENT PRIMARY KEY,
        customer_name VARCHAR(255),
        mode_of_payment VARCHAR(255),
        item_no INT,
        item_name VARCHAR(255),
        unit_price FLOAT,
        quantity INT,
        total_price FLOAT,
        phone_number varchar(10),
        address varchar(800),
        purchase_price int,
        gst int,
```

```
        net_price int
    )
''')


    current_bill = []
    global customer_login_frame, frames, login_entry
    notebook = ttk.Notebook(root)
    notebook.pack(fill='both', expand='yes')




    if customer_login_frame is not None:
        customer_login_frame.destroy()
    login_frame.destroy()  # Destroy the login frame

    customer_cart = {}
    def order_groceries():
            current_bill = []

# Initialize the Tkinter window

    billing_frame = ttk.Frame(root)
    notebook.add(billing_frame, text='Billing')
    image = PhotoImage(file=r"C:\users\babu\Desktop\project\bill.PNG")
```

```
# Add a label to display the image
    image_label = ttk.Label(billing_frame, image=image)
    image_label.place(x=0, y=0, relwidth=1, relheight=1)
    image_label.image = image




# Function to display the current bill



# Function to update the displayed bill


    def display_customer_info():
        bill_text.config(state=tk.NORMAL)
        bill_text.delete("1.0", tk.END)
        bill_text.insert(tk.END, "*" * 100 + "\n")
        bill_text.insert(tk.END, "Customer Name: {}\tMode of Payment:
{}\n".format(customer_name_entry.get(), mode_of_payment_entry.get()))
        bill_text.insert(tk.END, "*" * 100 + "\n")
        bill_text.config(state=tk.DISABLED)


# Function to display items in the bill
    def display_items():
        bill_text.config(state=tk.NORMAL)
        bill_text.insert(tk.END, "Item No\t\tItem Name\t\t\tUnit
Price\t\tQuantity\t\tTotal Price\t\tgst\tnet_price\n")
```

```
        bill_text.insert(tk.END, "*" * 105 + "\n")
        for item in current_bill:
            item_no, item_name, unit_price, quantity,
total_price,gst,net_price= item
            bill_text.insert(tk.END,
f"{item_no}\t\t{item_name}\t\t\t{unit_price}\t\t{quantity}\t\t{total_price}\t\t{g
st}\t{net_price}\n")
            bill_text.insert(tk.END, "*" * 105 + "\n")
        total_amount = sum(item[6] for item in current_bill)
        total_amount_label.config(text=f"Total Amount: {total_amount:.2f}")
        bill_text.config(state=tk.DISABLED)



# Function to update the displayed bill
    def update_bill():
        display_customer_info()
        display_items()


# Function to bill an item
  # Function to bill an item
    finalize_stock_update = False
    def bill_item():
        item_no = item_no_bill_entry.get()
        quantity = quantity_bill_entry.get()
        customer_name = customer_name_entry.get()
        mode_of_payment = mode_of_payment_entry.get()


        if not item_no or not quantity:
```

```
            messagebox.showinfo("Error", "Item No and Quantity are
required.")
            return


        try:
            item_no = int(item_no)
            quantity = int(quantity)


            cursor.execute("SELECT item_name, sale_price, expiry_date,
gst FROM items WHERE itemno = %s", (item_no,))
            result = cursor.fetchone()


            if result:
                item_name, unit_price, expiry_date, gst = result
                cursor.execute("SELECT quantity FROM items WHERE
itemno = %s", (item_no,))
                current_stock = cursor.fetchone()


                if current_stock:
                    current_stock = current_stock[0]
                    total_quantity_in_bill = sum(item[3] for item in current_bill if
item[0] == item_no)
                    if current_stock >= (quantity + total_quantity_in_bill):
                        if expiry_date > datetime.now().date():  # Check if the
product is not expired
                            total_price = unit_price * quantity
                            net_price = total_price * gst / 100 + total_price
                            current_bill.append((item_no, item_name, unit_price,
quantity, total_price, gst, net_price))
```

```
                    update_bill()
                else:
                    messagebox.showerror('Expired', 'This product is
Expired')
            else:
                messagebox.showinfo("Error", "Insufficient stock for this
item.")
        else:
            messagebox.showinfo("Error", "Item not found.")
    else:
        messagebox.showinfo("Error", "Item not found.")


except ValueError:
    messagebox.showinfo("Error", "Invalid quantity. Please enter a
valid number.")



# Function to finalize the bill
def finalize_bill():
    global finalize_stock_update
    customer_name = customer_name_entry.get()
    mode_of_payment = mode_of_payment_entry.get()


    customer_phone_number = customer_phone_number_entry.get()
    customer_address = customer_address_entry.get()
    if not customer_phone_number or not customer_address:
        messagebox.showinfo("Error", "phone number address need to
be filled .")
```

```
        else:

            pygame.init()

            # Load the music file
pygame.mixer.music.load(r"C:\Users\babu\Desktop\project\confirm.mp3"
)

            # Play the music
            pygame.mixer.music.play()

            for item in current_bill:
                item_no, item_name, unit_price, quantity, total_price, gst,
net_price = item
                cursor.execute("select unit_price from items where
itemno=%s", (item_no,))
                price = cursor.fetchone()
                m = price[0]
                cursor.execute("select gst from items where itemno=%s",
(item_no,))
                tax = cursor.fetchone()
                t = tax[0]
                np = total_price + (total_price * t / 100)

                pur_price = unit_price * quantity

                cursor.execute(
```

```
            "INSERT INTO bills (customer_name, mode_of_payment,
item_no, item_name, unit_price, quantity,
total_price,phone_number,address,purchase_price,gst,net_price)
VALUES (%s, %s, %s, %s, %s, %s, %s,%s,%s,%s,%s,%s)",

            (customer_name, mode_of_payment, item_no, item_name,
unit_price, quantity, total_price,

            customer_phone_number, customer_address, m, t, np,))

            conn.commit()


        finalize_stock_update = True  # Set the flag to True to update
stock

        current_bill.clear()

        update_bill()

        finalize_stock_update = False

    xyz = ttk.Style()

    xyz.configure("custom.TLabel", font=("Bernard MT condensed",
15),foreground="red")

    xyz.configure("custom.TEntry", font=("Bernard MT condensed", 30))

    xyz.configure("custom.TButton", font=("Bernard MT condensed", 15),
background="black",foreground="green")


# Create UI elements


    customer_name_label = ttk.Label(billing_frame, text="Customer
Name:",style="custom.TLabel")

    customer_name_label.grid(row=0, column=0, padx=10, pady=5,
sticky="w")

    customer_name_entry =
ttk.Entry(billing_frame,style="custom.TEntry")

    customer_name_entry.grid(row=0, column=1, padx=10, pady=5,
sticky="w")
```

```
    mode_of_payment_label = ttk.Label(billing_frame, text="Mode of
Payment:",style="custom.TLabel")

    mode_of_payment_label.grid(row=1, column=0, padx=10, pady=5,
sticky="w")

    mode_of_payment_entry =
ttk.Entry(billing_frame,style="custom.TEntry")

    mode_of_payment_entry.grid(row=1, column=1, padx=10, pady=5,
sticky="w")



    item_no_bill_label = ttk.Label(billing_frame, text="Item
No:",style="custom.TLabel")

    item_no_bill_label.grid(row=2, column=0, padx=10, pady=5,
sticky="w")

    item_no_bill_entry = ttk.Entry(billing_frame)

    item_no_bill_entry.grid(row=2, column=1, padx=10, pady=5,
sticky="w")



    quantity_bill_label = ttk.Label(billing_frame,
text="Quantity:",style="custom.TLabel")

    quantity_bill_label.grid(row=3, column=0, padx=10, pady=5,
sticky="w")

    quantity_bill_entry = ttk.Entry(billing_frame)

    quantity_bill_entry.grid(row=3, column=1, padx=10, pady=5,
sticky="w")



    bill_button = ttk.Button(billing_frame, text="Bill Item",
command=bill_item,style="custom.TButton")

    bill_button.grid(row=4, columnspan=2, pady=10)



    bill_text = tk.Text(billing_frame, height=20, width=110)

    bill_text.grid(row=5, columnspan=2, pady=10,sticky="nsew")

    bill_text.config(state=tk.DISABLED)
```

```
    total_amount_label = ttk.Label(billing_frame, text="Total Amount: 0.00",style="custom.TLabel")

    total_amount_label.grid(row=6, columnspan=2, pady=5)


    finalize_button = ttk.Button(billing_frame, text="Finalize Bill", command=finalize_bill,style="custom.TButton")

    finalize_button.grid(row=8, columnspan=2, pady=10)


    customer_phone_number_label = ttk.Label(billing_frame, text="Phone Number:",style="custom.TLabel")

    customer_phone_number_label.grid(row=4, column=2, padx=40, pady=5)

    customer_phone_number_entry = ttk.Entry(billing_frame)

    customer_phone_number_entry.grid(row=4, column=3, padx=45, pady=5)


    customer_address_label = ttk.Label(billing_frame, text="Address:",style="custom.TLabel")

    customer_address_label.grid(row=5, column=2, padx=40, pady=5,)

    customer_address_entry = ttk.Entry(billing_frame)

    customer_address_entry.grid(row=5, column=3, padx=45, pady=5)


    stock_treeview = ttk.Treeview(notebook, columns=("itemno", "item_name", "quantity",  "expiry_date", "sale_price","gst","net_price"))

    notebook.add(stock_treeview, text='Show Stock')

    button_frame = ttk.Frame(stock_treeview)

    button_frame.grid(row=0, column=0, pady=10)
```

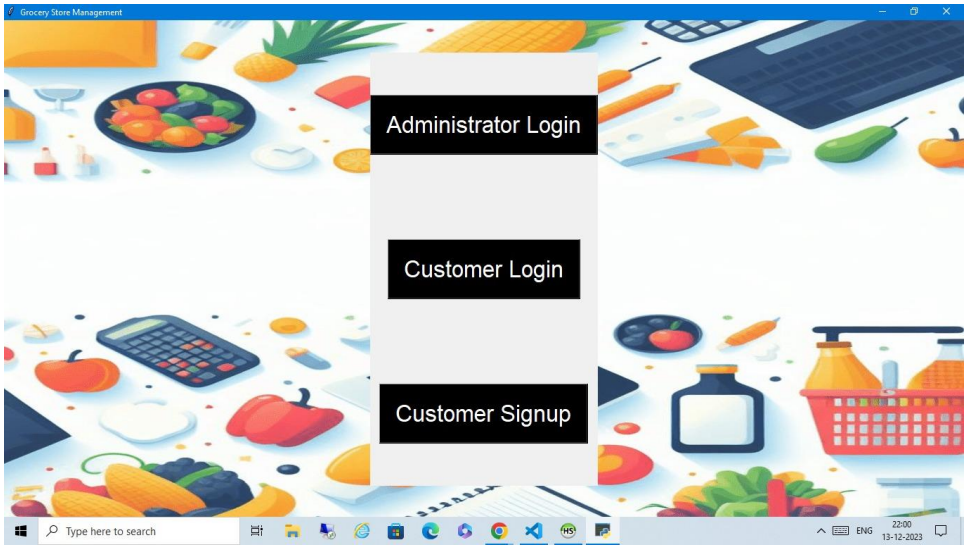
```
# Define columns for the table
stock_treeview.heading("#1", text="Item No")
stock_treeview.heading("#2", text="Item Name")
stock_treeview.heading("#3", text="Quantity")

stock_treeview.heading("#4", text="Expiry Date")
stock_treeview.heading("#5", text="Sale Price")
stock_treeview.heading("#6", text="gst")
stock_treeview.heading("#7", text="net price")


# Set column widths
stock_treeview.column("#1", width=100)
stock_treeview.column("#2", width=200)
stock_treeview.column("#3", width=100)

stock_treeview.column("#4", width=150)
stock_treeview.column("#5", width=100)
stock_treeview.column("#6", width=100)
stock_treeview.column("#7", width=100)




# Function to populate the table with stock data
def show_stock():
```

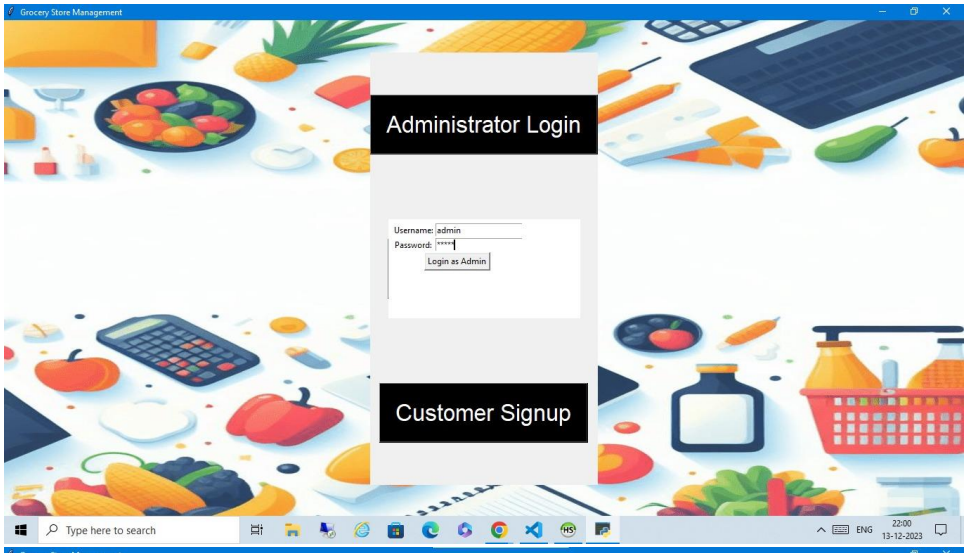
```
        stock_treeview.delete(*stock_treeview.get_children())  # Clear the table

        cursor.execute("SELECT itemno,item_name,quantity,expiry_date,sale_price,gst,net_price FROM items")

        result = cursor.fetchall()

        for row in result:

            stock_treeview.insert("", "end", values=row)




    show_stock_button = ttk.Button(button_frame, text="Show Stock", command=show_stock)

    show_stock_button.grid(row=0, column=0, pady=10)



    style = ttk.Style()

    style.configure("Treeview.Heading", font=("Helvetica", 18))  # Increase the font size (12) as needed



# Increase the font size for cell values

      # Increase the font size (10) as needed



# Set the background color for the Treeview

    style.configure("Treeview", background="#D9E4FF")  # Change the color code to the color you want

    pygame.init()



# Load the music file


pygame.mixer.music.load(r"C:\Users\babu\Desktop\project\success.mp3")
```

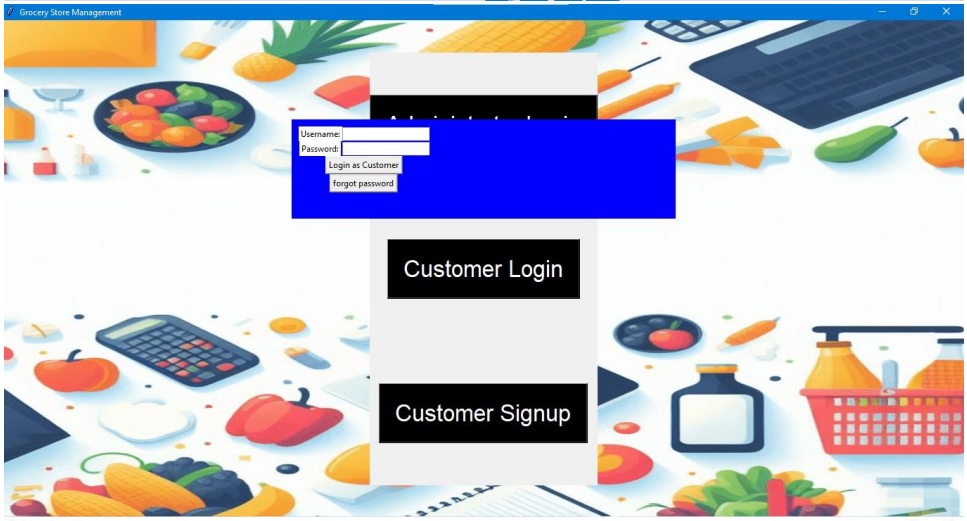
```
# Play the music
    pygame.mixer.music.play()

root.mainloop()
```

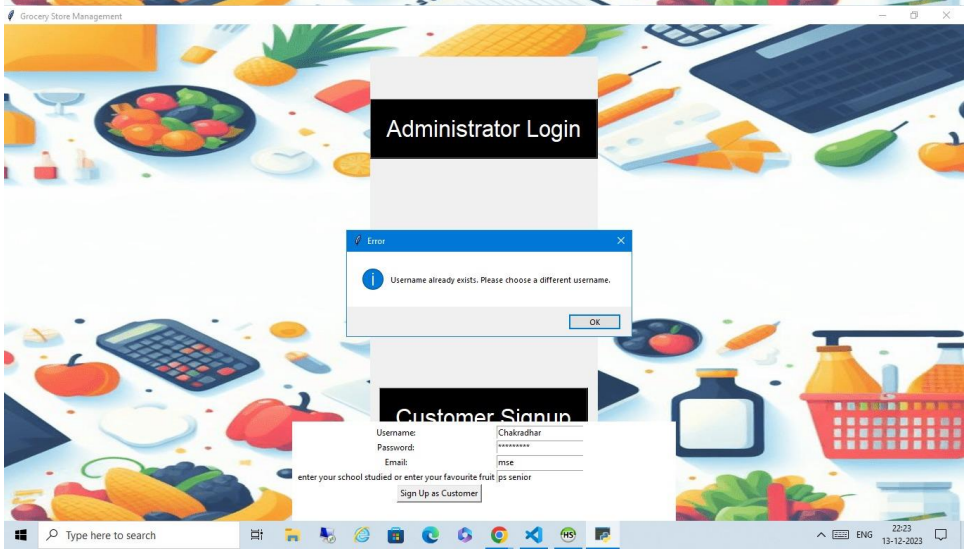
OUTPUT

LOGIN SCREEN:

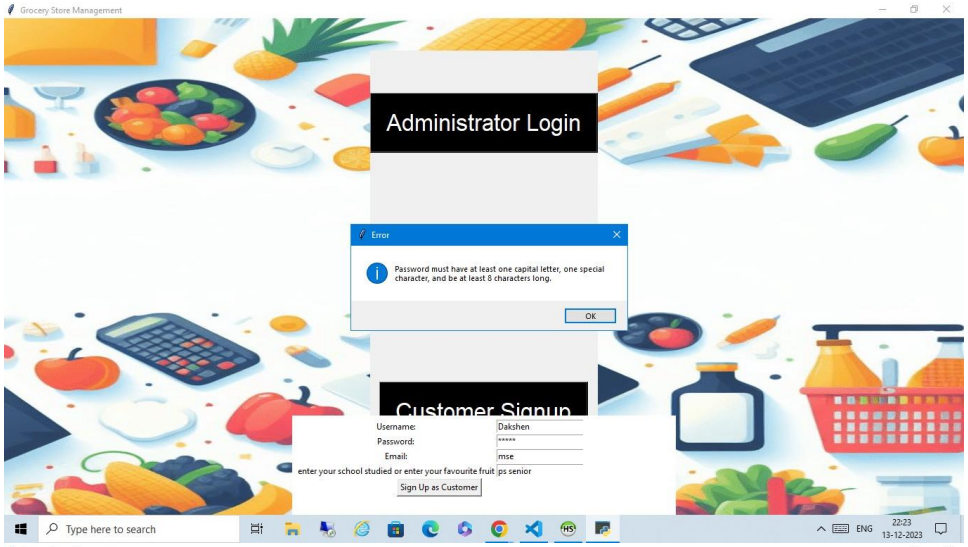


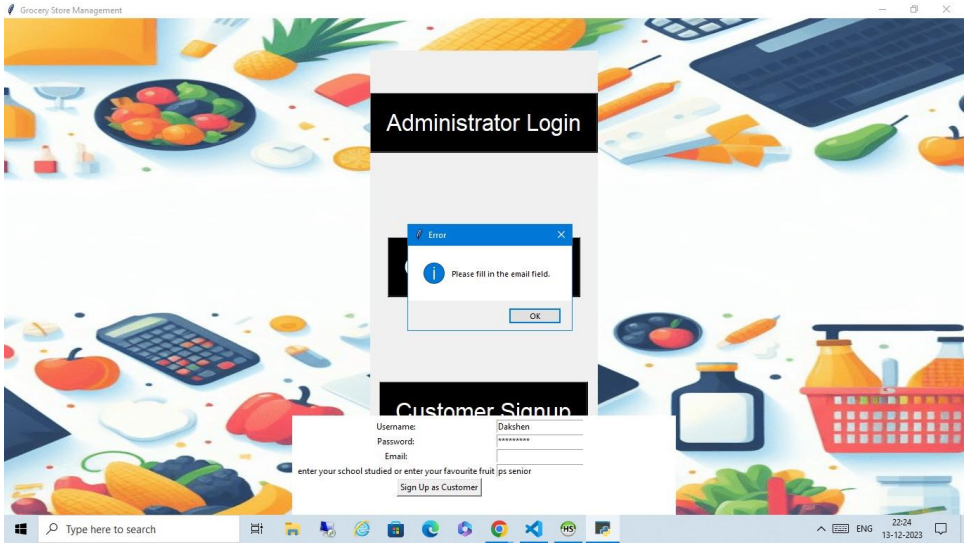

Grocery Store Management
Administrator Login
Username: admin
Password: *****
Login as Admin
Customer Signup
Type here to search
ENG
22:00
13-12-2023

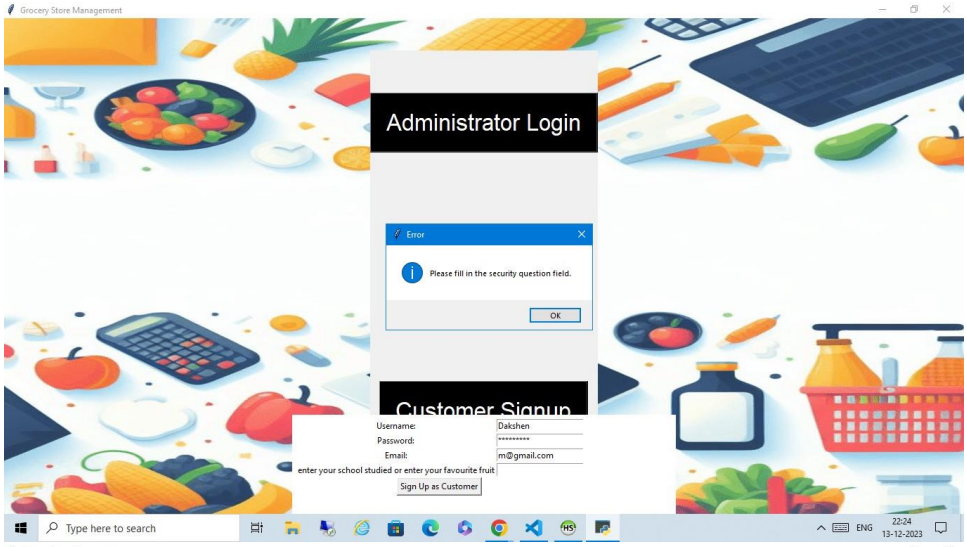

Grocery Store Management
Username:
Password:
Login as Customer
forgot password
Customer Login
Customer Signup

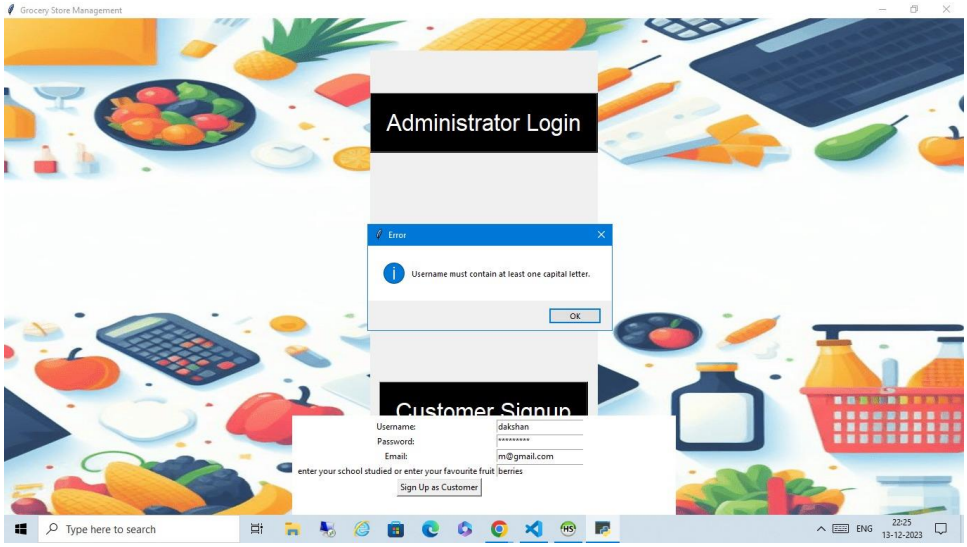

Grocery Store Management
Administrator Login
Error
Username already exists. Please choose a different username.
OK
Customer Signup
Username: Chakradhar
Password: *********
Email: mse
enter your school studied or enter your favourite fruit ps senior
Sign Up as Customer
Type here to search
ENG
22:23
13-12-2023


Password Reset
Username
enter your school studied or enter your favourite fruit
New Password
Reset


Grocery Store Management
Administrator Login
Error
Password must have at least one capital letter, one special character, and be at least 8 characters long.
OK
Customer Signup
Username:
Dakshen
Password:
*****
Email:
mse
enter your school studied or enter your favourite fruit
ps senior
Sign Up as Customer
Type here to search
ENG
22:23
13-12-2023


Grocery Store Management
Administrator Login
Error
Please fill in the email field.
OK
Customer Signup
Username:
Dakshen
Password:
****
Email:
enter your school studied or enter your favourite fruit
ps senior
Sign Up as Customer
Type here to search
ENG
22:24
13-12-2023



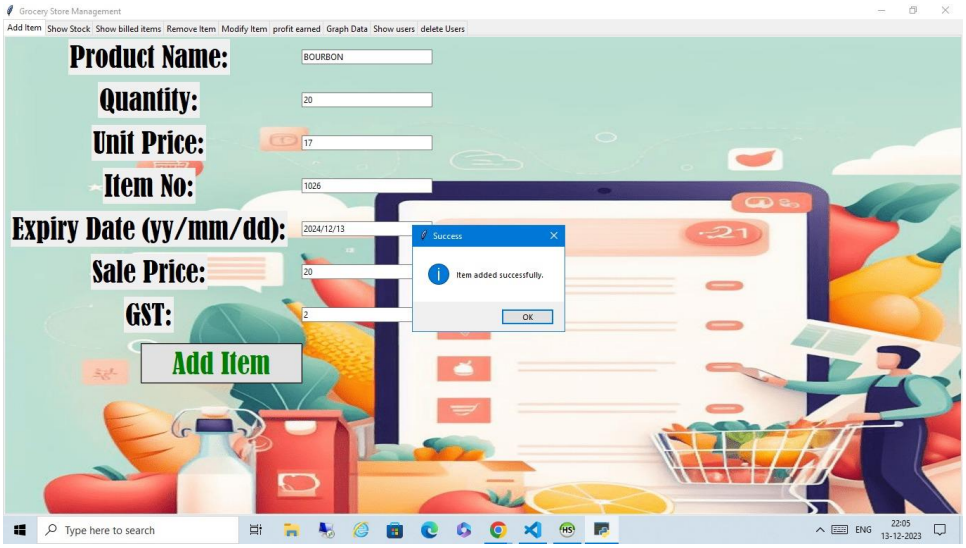


ADMINISTRATOR PORTAL

In this portal a soothing music is played in background. The portal is designed with notebook module to make functions easier to access

PURCHASE ENTRY:

In this the admin can add purchased items from wholesaler/manufacturer



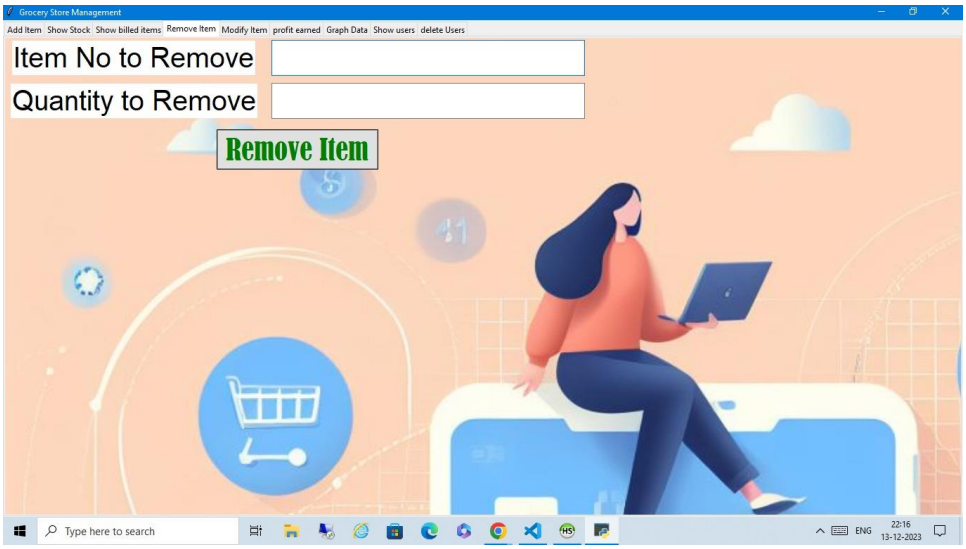
SHOW STOCK

SHOW BILLS

Show Bills

Bill No	customer nam	mode	Item No	Item	unit_pr	quantity	Sale P	Phone N	Address	pu	gst	net_price
1	CHAKRADHAR	card	1024	apple	150.0	10	1500.0	8144616142	MYLAPORE	100	2	1530
2	ALBEN	card	1012	UNOMAX	20.0	10	200.0	8144616142	GUINDY	10	1	202
3	HAMDAN	card	1020	PILLSBURI	130.0	10	1300.0	8144616142	SAIDAPET	100	4	1352
4	KUMAR	card	1021	MAGGI	20.0	10	200.0	8144616142	PORUR	15	4	208
5	DAKSHAN	card	1023	FANATA 1	35.0	1	35.0	8144616142	TEYNAMPET	30	3	36
6	RAHUL	card	1013	PERK	10.0	10	100.0	8144616142	MINJUR	5	0	100
7	ABISHEK	card	1014	RR PONN	65.0	10	650.0	8144616142	Guduvancerry	47	1	657
8	SATHAPPAN	card	1015	VATIKA H	120.0	10	1200.0	8144616142	Velacherry	100	2	1224
9	SAJJAN	card	1019	CELLO BU	10.0	10	100.0	8144616142	ADAYAR	7	4	104
10	Yuvan	cash	1000	bru	200.0	2	400.0	9841224243	MYLAPORE	150	5	420
11	Shashank	cash	1007	AACHI GA	30.0	5	150.0	9841224243	MYLAPORE	20	5	158
12	pranav rajesh	cash	1021	MAGGI	20.0	5	100.0	9841224243	Mambalam	15	4	104
13	Aswin	cash	1021	MAGGI	20.0	5	100.0	9841224243	T NAGAR	15	4	104
14	lakshana	cash	1022	PONDS D	10.0	5	50.0	9841224243	TAMBARAM	9	2	51
15	PAVISHIYA	cash	1025	BRIL INK	25.0	5	125.0	9841224243	THARAMANI	20	2	128
16	Sanjana	cash	1025	BRIL INK	25.0	5	125.0	9841224243	TRIPLICANE	20	2	128

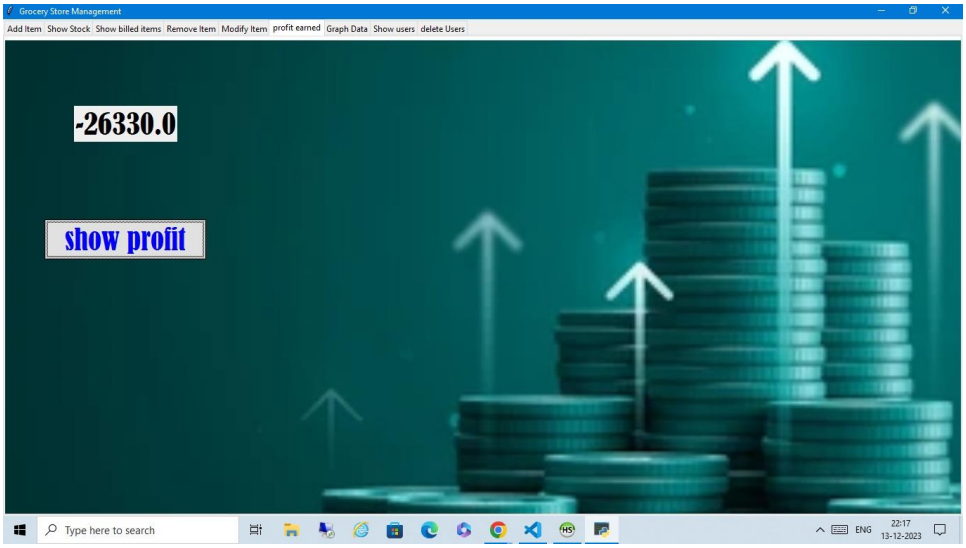
REMOVE ITEM



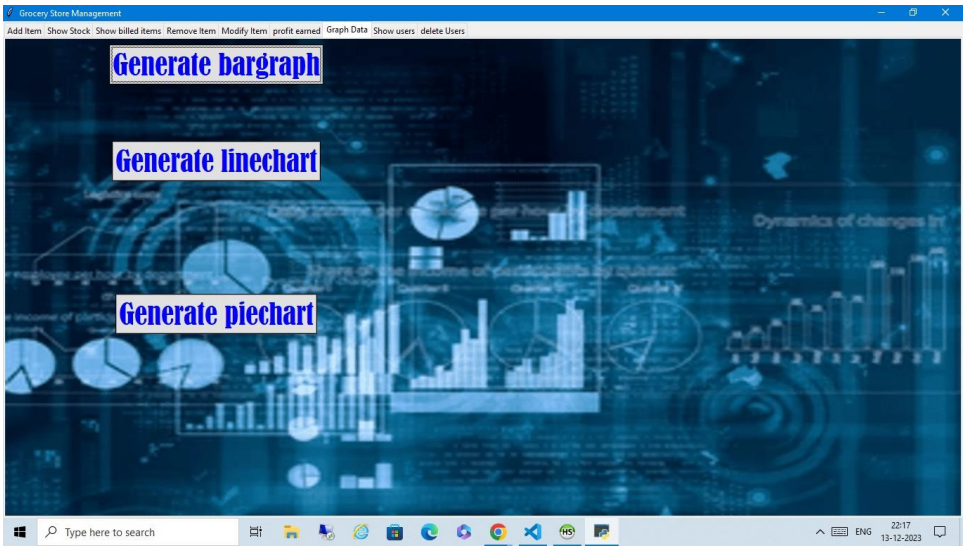
MODIFY ITEM


Grocery Store Management
Add Item
Show Stock
Show billed items
Remove Item
Modify Item
profit earned
Graph Data
Show users
delete Users
Item No to Modify:
New Quantity:
New Unit Price:
New Expiry Date (yy/mm/dd):
New Sale Price:
New gst:
Modify Item
MADDGIERTY
Appraisal
Type here to search
ENG
22:17
13-12-2023

SHOW PROFIT


Grocery Store Management
Add Item
Show Stock
Show billed items
Remove Item
Modify Item
profit earned
Graph Data
Show users
delete Users
-26330.0
show profit
Type here to search
ENG
22:17
13-12-2023

SHOW GRAPHS


Grocery Store Management
Add Item
Show Stock
Show billed items
Remove Item
Modify Item
profit earned
Graph Data
Show users
delete Users
Generate bargraph
Generate linechart
Generate piechart
Type here to search
ENG
22:17
13-12-2023

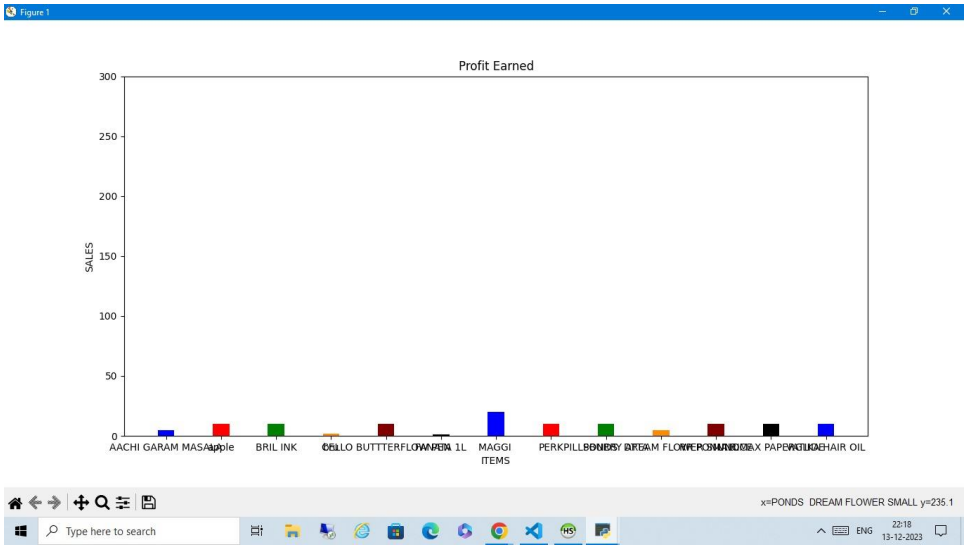


Figure 1
Profit Earned
SALES
300
250
200
150
100
50
0
AACHI GARAM MASA
Apple
BRIL INK
MAGGI
ITEMS
PERK
x=PONDS DREAM FLOWER SMALL y=235.1
Type here to search
ENG
22:18
13-12-2023

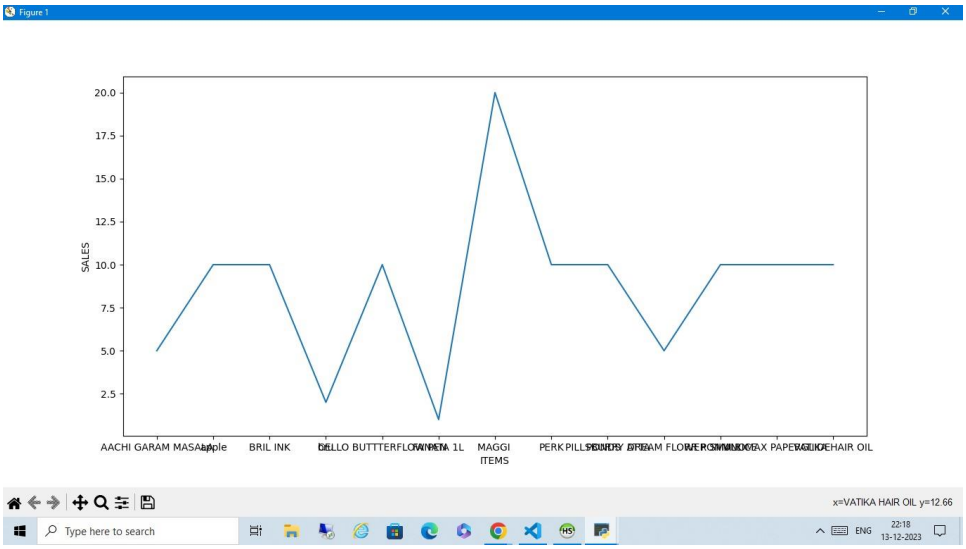


Figure 1
SALES
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
AACHI GARAM MASA
Apple
BRIL INK
MAGGI
ITEMS
PERK
x=VATIKA HAIR OIL y=12.66
Type here to search
ENG
22:18
13-12-2023

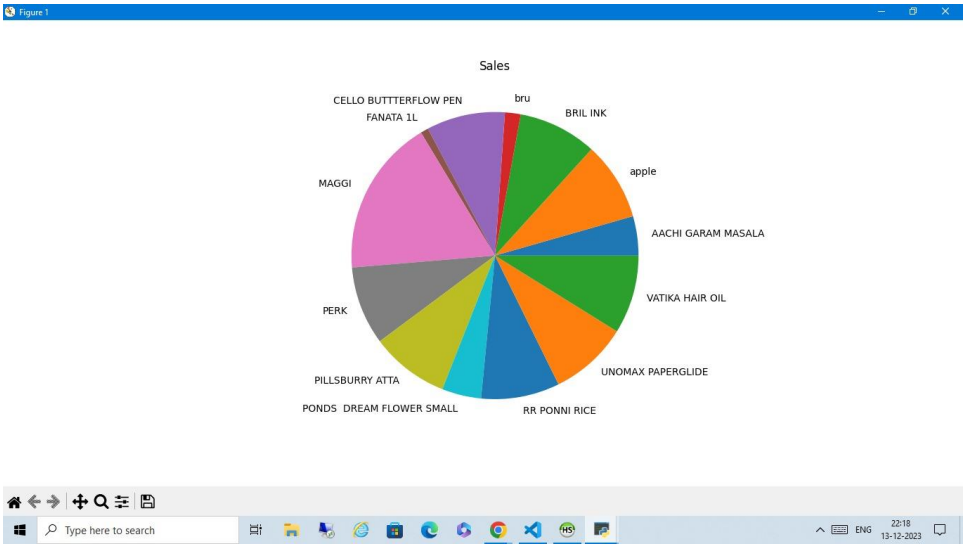
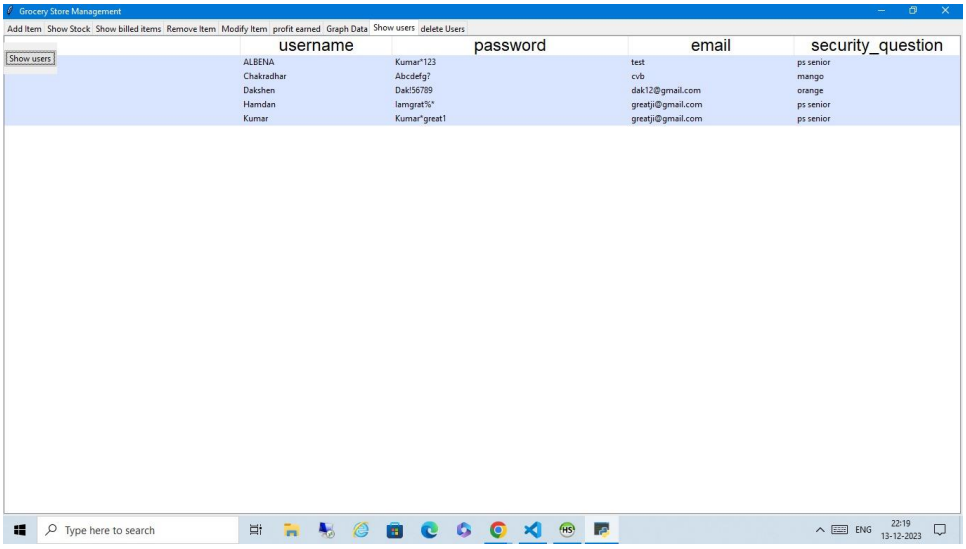
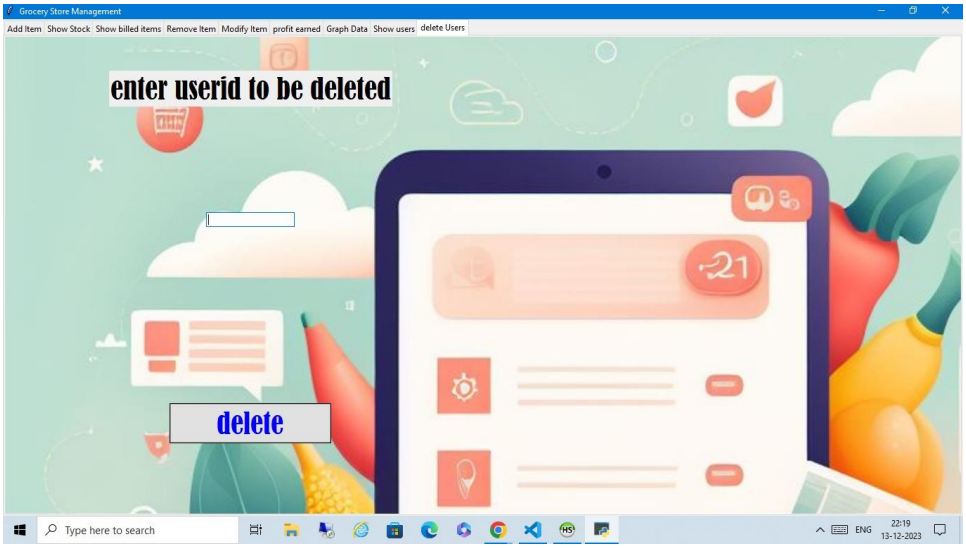


Figure 1
Sales
CELLO BUTTTERFLOW PEN
FANATA 1L
bru
BRIL INK
apple
AACHI GARAM MASALA
VATIKA HAIR OIL
UNOMAX PAPERGLIDE
RR PONNI RICE
PONDS DREAM FLOWER SMALL
PILLSBURRY ATTA
PERK
MAGGI
Type here to search
ENG
22:18
13-12-2023

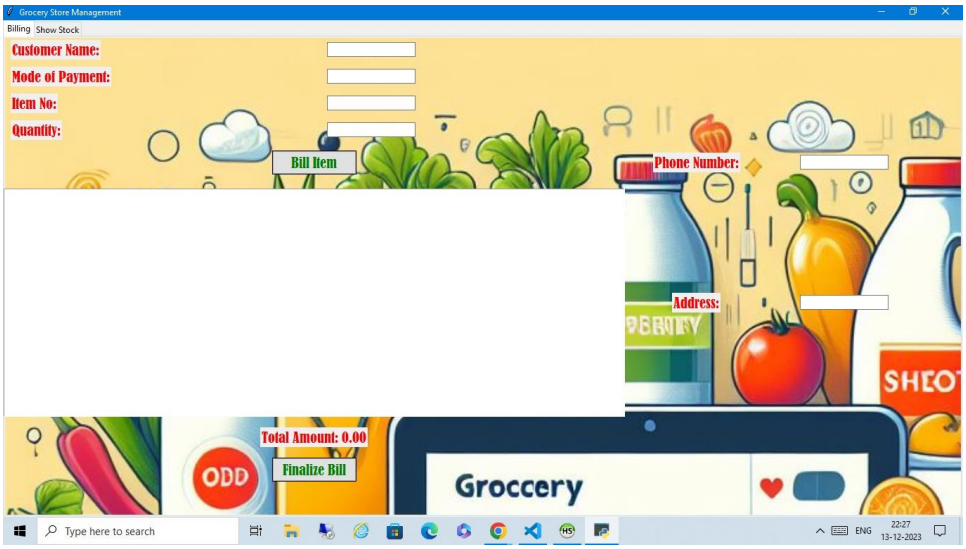
SHOW USERS


Grocery Store Management
Add Item
Show Stock
Show billed items
Remove Item
Modify Item
profit earned
Graph Data
Show users
delete Users
Show users
username
password
email
security_question
ALBENA
Kumar^123
test
ps senior
Chakradhar
Abcdefg?
cvb
mango
Dakshen
Dak!56789
dak12@gmail.com
orange
Hamdan
!amgrat%^
greatji@gmail.com
ps senior
Kumar
Kumar^great1
greatji@gmail.com
ps senior
Type here to search
ENG
22:19
13-12-2023

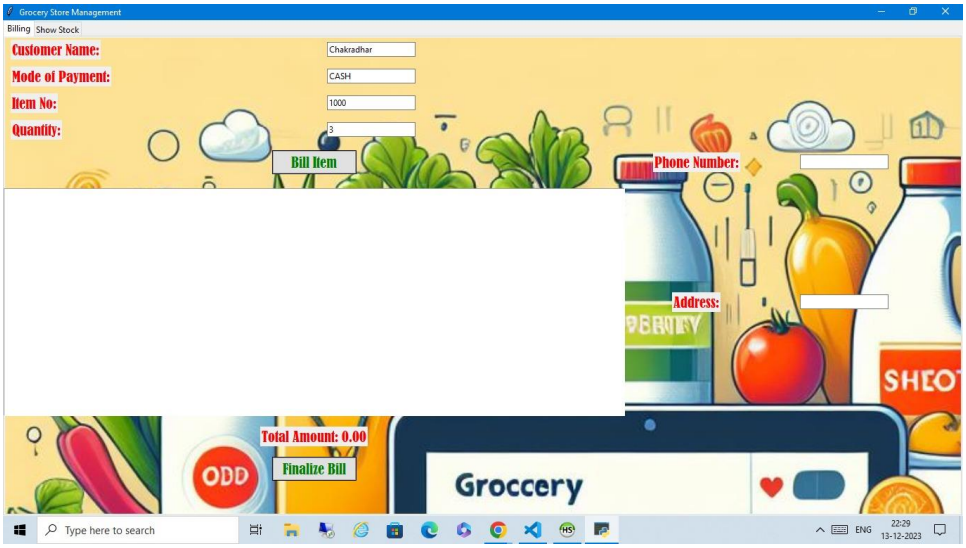
DELETE USERS

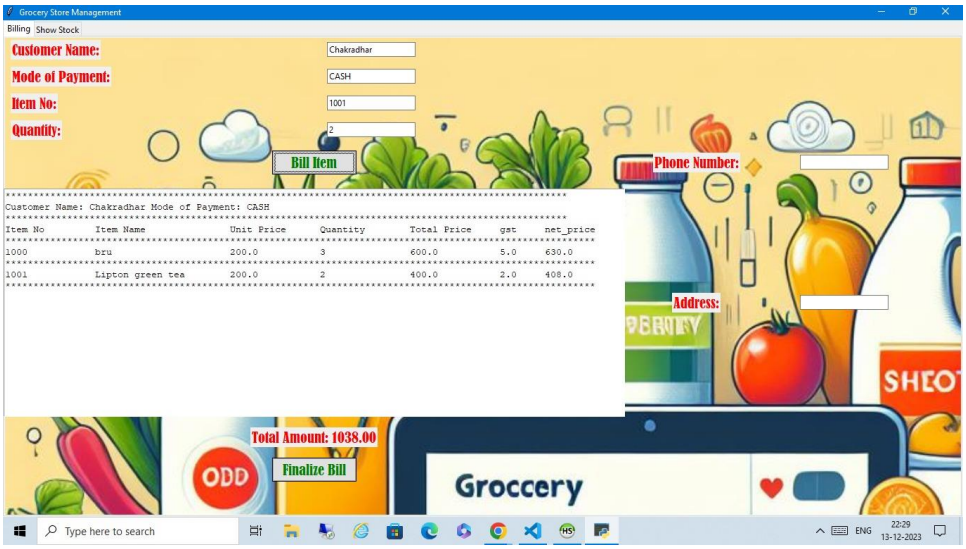


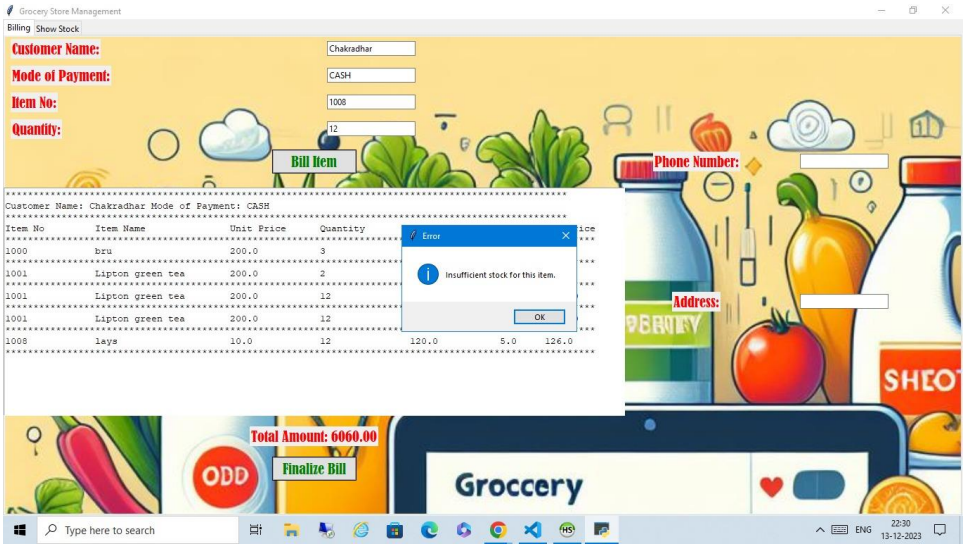
CUSTOMER PORTAL

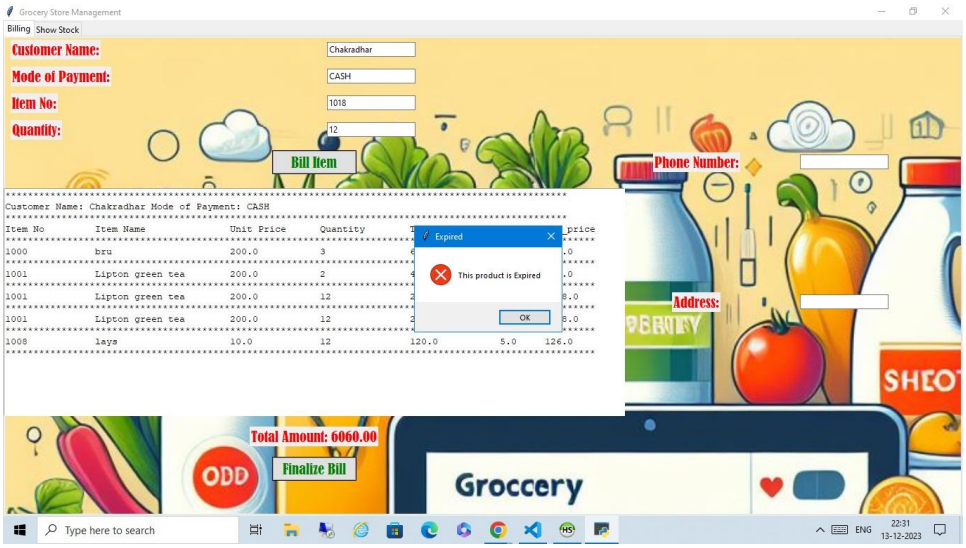


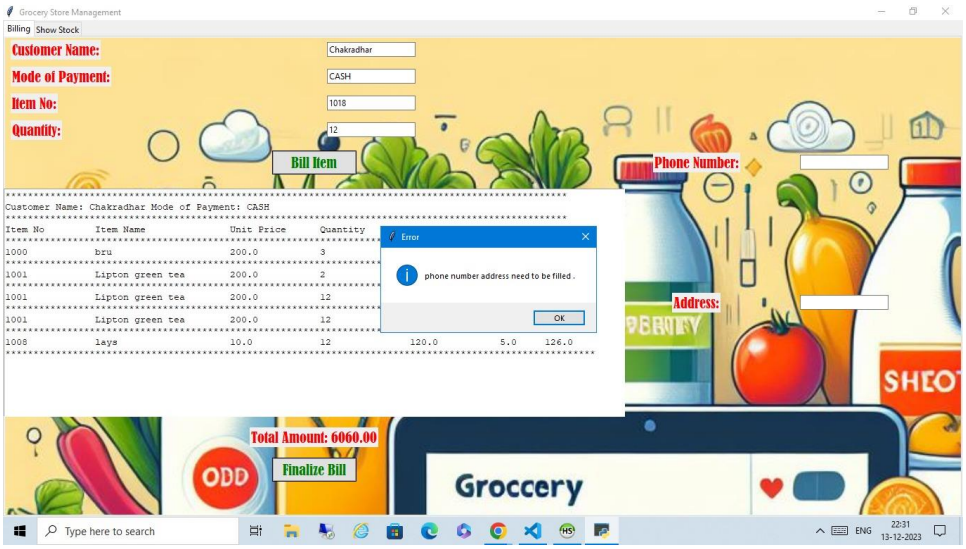
Item No	Item Name	Quantity	Expiry Date	Sale Price	gst	net price
1000	bru	4	2025-11-15	200.0	5.0	210.0
1001	Lipton green tea	3450	2045-12-30	200.0	2.0	204.0
1002	sugarfree	64	2025-05-05	110.0	7.0	117.7
1003	Prime	15	2023-12-22	1000.0	8.0	1080.0
1004	carrot	93	2025-01-05	110.0	7.0	117.7
1005	AAVIN GHEE 1L	98	2025-05-05	110.0	6.0	116.6
1006	orange	20	2023-12-30	57.0	2.0	58.14
1007	AACHI GARAM MASALA	40	2025-08-08	30.0	5.0	31.5
1008	lays	12	2024-11-23	10.0	5.0	10.5
1009	A4 SHEET	3440	2045-12-30	1.0	2.0	1.02
1010	Harpic blue	100	2025-11-21	200.0	3.0	206.0
1011	ajanata beetel nuts	20	2024-11-10	1.0	2.0	1.02
1012	UNOMAX PAPERGLIDE	80	2026-11-20	20.0	1.0	20.2
1013	PERK	100	2026-11-21	10.0	0.0	10.0
1014	RR PONNI RICE	98	2027-11-29	65.0	1.0	65.65
1015	VATIKA HAIR OIL	97	2026-11-23	120.0	2.0	122.4
1016	IRRUTU KADAI HALWA	200	2021-07-20	100.0	4.0	104.0
1017	gold winner palm oil	30	2025-10-23	150.0	2.0	105.0
1018	fortune atta	75	2021-07-20	100.0	5.0	105.0
1019	CELLO BUTTTERFLOW PEN	21	2025-12-11	10.0	4.0	10.4
1020	PILLSBURRY ATTA	136	2024-12-10	130.0	4.0	135.2
1021	MAGGI	97	2025-10-10	20.0	4.0	20.8
1022	PONDS DREAM FLOWER SMALL	18	2026-12-07	10.0	2.0	10.2
1023	FANATA 1L	26	2024-04-21	35.0	3.0	36.05
1024	apple	20	2023-12-30	150.0	2.0	153.0
1025	BRIL INK	40	2027-10-23	25.0	2.0	25.5
1026	BOURBON	20	2024-12-13	20.0	2.0	20.4


Grocery Store Management
Billing
Show Stock
Customer Name:
Chakradhar
Mode of Payment:
CASH
Item No:
1000
Quantity:
3
Bill Item
Phone Number:
Address:
Total Amount: 0.00
Finalize Bill
Groccery
Type here to search
ENG
22:29
13-12-2023


Grocery Store Management
Billing
Show Stock
Customer Name:
Chakradhar
Mode of Payment:
CASH
Item No:
1001
Quantity:
2
Bill Item
Phone Number:
Address:
Customer Name: Chakradhar Mode of Payment: CASH
Item No Item Name Unit Price Quantity Total Price gst net_price
1000 bru 200.0 3 600.0 5.0 630.0
1001 Lipton green tea 200.0 2 400.0 2.0 408.0
Total Amount: 1038.00
Finalize Bill
Groccery
Type here to search
ENG
22:29
13-12-2023


Grocery Store Management
Billing
Show Stock
Customer Name:
Chakradhar
Mode of Payment:
CASH
Item No:
1008
Quantity:
12
Bill Item
Phone Number:
Address:
Customer Name: Chakradhar Mode of Payment: CASH
Item No Item Name Unit Price Quantity
1000 bru 200.0 3
1001 Lipton green tea 200.0 2
1001 Lipton green tea 200.0 12
1001 Lipton green tea 200.0 12
1008 lays 10.0 12 120.0 5.0 126.0
Error
Insufficient stock for this item.
OK
Total Amount: 6060.00
Finalize Bill
Groccery
Type here to search
ENG
22:30
13-12-2023


Grocery Store Management
Billing
Show Stock
Customer Name:
Chakradhar
Mode of Payment:
CASH
Item No:
1018
Quantity:
12
Bill Item
Phone Number:
Address:
Customer Name: Chakradhar Mode of Payment: CASH
Item No
Item Name
Unit Price
Quantity
price
1000
bru
200.0
3
1001
Lipton green tea
200.0
2
1001
Lipton green tea
200.0
12
1001
Lipton green tea
200.0
12
1008
lays
10.0
12
120.0
5.0
126.0
Expired
This product is Expired
OK
Total Amount: 6060.00
Finalize Bill
Groccery
Type here to search
ENG
22:31
13-12-2023


Grocery Store Management
Billing
Show Stock
Customer Name:
Chakradhar
Mode of Payment:
CASH
Item No:
1018
Quantity:
12
Bill Item
Phone Number:
Address:
Customer Name: Chakradhar Mode of Payment: CASH
Item No
Item Name
Unit Price
Quantity
1000
bru
200.0
3
1001
Lipton green tea
200.0
2
1001
Lipton green tea
200.0
12
1001
Lipton green tea
200.0
12
1008
lays
10.0
12
120.0
5.0
126.0
Error
phone number address need to be filled .
OK
Total Amount: 6060.00
Finalize Bill
Groccery
Type here to search
ENG
22:31
13-12-2023

LIMITATIONS

1.The user portal cannot be linked with payment gateway as it is not feasible

2.The forgot password cannot be able to do one time password-based verification as it requires payment charges charged by service provider like amazon aws , using smtp and sending email is illegal

3.The bill id is generated each item instead of generating for one bill

4.the sign up process gets lagged sometimes due to connection between SQL table is not fast

5.the users cannot be able to modify the details

CONCLUSION

The online grocery management system has emerged as a transformative force in the retail landscape, offering unprecedented convenience and efficiency for both consumers and retailers. In conclusion, the digitalization of grocery shopping has proven to be a boon, streamlining the entire process from selection to delivery. One of the primary benefits is the convenience afforded to consumers. Online platforms provide a vast array of products at the fingertips of users, allowing them to browse and purchase from the comfort of their homes. This convenience is further enhanced by features such as personalized recommendations, user-friendly interfaces, and flexible delivery options, catering to the diverse needs of modern consumers.

Retailers, on the other hand, have experienced a paradigm shift in their operations. The adoption of online grocery management has optimized inventory management, reduced overhead costs associated with physical stores, and opened new avenues for targeted marketing and customer engagement. The data generated through online transactions enables retailers to analyze consumer behavior, preferences, and trends, facilitating more informed decision-making and strategic planning.

However, challenges such as logistics, last-mile delivery, and ensuring the quality of perishable goods persist. Overcoming these challenges requires continuous innovation and investment in technology. Moreover, issues related to data security and privacy necessitate robust systems to build and maintain trust among consumers.

In conclusion, the online grocery management system has redefined the retail experience, offering unparalleled convenience and efficiency. The continued evolution of technology and proactive measures to address challenges will be pivotal in sustaining and enhancing the positive impact of online grocery management in the future. As the industry matures, collaboration between retailers, technology providers, and policymakers will play a crucial role in shaping a seamless and sustainable future for online grocery management.

BIBLIOGRAPHY

1)Preeti Arora

2)All in one

3)Sumita Arora

4)V K Pandey & D K Dey

5)Dr Archana Agarwa