KENDRIYA VIDYALAYA NO. 2
F.C.I CAMPUS GORAKHPUR

तत् त्वं पूषन् अपावृणु
केन्द्रीय विद्यालय संगठन

COMPUTER SCIENCE PROJECT EXPLANATION

BAKERY MANAGEMENT SYSTEM

Presented by- Harsh Shukla

XII'A'

Board Roll No.-

SUBMITTED TO- MR. R. P. KUSHWAHA

SIGN OF INTERNAL

SIGN OF EXTERNAL

TABLE OF CONTENTS

CERTIFICATE

This is to certify that “Harsh Shukla”, “Baibhav Vishal Tripathi”, “Anuj Chandra”, “Avinash” student of class “XII (‘A’)”, has successfully prepared the report on the project entitled “Bakery Management ” under the guidance of Mr. R. P. Kushwaha (PGT Computer Science). The report is the result of his efforts & endeavors. The report is found worthy of acceptance as the final project report for the subject Computer Science of class XII (Sci.) 2024-25 .

Signature of Computer Science Teacher
(Mr. R. P. Kushwaha)

Signature of External Examiner

Signature of Principal
(Mr. A.K. Mishra)

Acknowledgement

I would like to express a deep sense of thanks and gratitude to my project guide Mr. R. P. Kushwaha for guiding me immensely through the course of the project. He always evinced keen interest in my project. His constructive advice & constant motivation have been responsible for the successful completion of his project.

My sincere thanks go to our principal sir for his coordination in extending every possible support for the completion of this project.

I must thank my classmates for their timely help and support for completion of this project.

Last but not the least, I would like to thank all those who had helped directly or indirectly towards the completion of this project.

HARSH SHUKLA

Class- XII("A")

Aim & Points to be covered

Aim

"EXPLANATION OF BAKARY MANAGEMENT"

Points To Be Covered

1. LANGUAGE AND LIBRARY USED.

2. BREIF EXPLANATION OF EVERY BLOCK OF CODE.

3. OUTPUT OF EVERY BLOCK OF PROGRAM.

4. LIMITATIONS OF PROGRAM.

LANGUAGES AND LIBRARY USED.

PYTHON 3.13.0 :-

Python is a set of instructions that we give in the form of a Program to our computer to perform any specific task. It is a Programming language having properties like it is interpreted, object-oriented and it is high-level too. Due to its beginner-friendly syntax, it became a clear choice for beginners to start their programming journey. The major focus behind creating it is making it easier for developers to read and understand, also reducing the lines of code.

- Python was created in "1980's" by "Guido van Rossum".

FEATURES OF PYTHON:-

1. Free and Open Source.
2. Easy to code.
3. Easy to Read.
5. GUI Programming Support.
6. High-Level Language.
7. Large Community Support.
8. Easy to Debug.
9. Python is a Portable language.
10. Python is an Integrated language.
11. Interpreted Language.
12. Large Standard Library.
13. Dynamically Typed Language.
14. Frontend and backend development.
15. Allocating Memory Dynamically.

LIBRARY USED

The provided code uses 3 libraries. Here's the list:

1. **mysql.connector**
    - Purpose: Used to connect to and interact with a MySQL database.
    - Functions:
        - Establishes a connection with the database.
        - Executes SQL queries and performs database operations.

2. **datetime**
    - Purpose: Provides classes to work with dates and times.
    - Functions:
        - Used for generating timestamps (datetime.now()) for order entries.

BREIF EXPLANATION OF PROGRAM

1. Database Connection and Setup:

```python
import mysql.connector
from datetime import datetime

# Connect to MySQL database
try:
    con = mysql.connector.connect(host="localhost", user="root", password="root")
    cur = con.cursor()
except mysql.connector.Error as e:
    print(f"Error connecting to database: {e}")
    exit()

# Create and use the database
cur.execute("CREATE DATABASE IF NOT EXISTS Bakery_Management")
cur.execute("USE Bakery_Management")
```

- **Imports:**
  - mysql.connector is used to connect to the MySQL database.
  - datetime is used to get the current timestamp for customer orders.
- **Database Connection:**
  - The mysql.connector.connect() method establishes a connection to the MySQL server using the provided credentials (localhost, root as the username, and root as the password).
  - If there's an error in connecting to the database, it exits the program.
- **Database Setup:**
  - CREATE DATABASE IF NOT EXISTS Bakery_Management: Creates the Bakery_Management database if it doesn't already exist.
  - USE Bakery_Management: Switches to using the Bakery_Management database.

2. Table Creation:

```
# Create tables if not already present
cur.execute("""
    CREATE TABLE IF NOT EXISTS items (
        serial_No INT PRIMARY KEY,
        products VARCHAR(20),
        quantity INT,
        cost INT
    )
""")

cur.execute("""
    CREATE TABLE IF NOT EXISTS flavours_cake (
        serial_No INT PRIMARY KEY,
        varieties VARCHAR(20),
        quantity INT,
        cost INT
    )
""")

cur.execute("""
    CREATE TABLE IF NOT EXISTS workers (
        serial_No INT PRIMARY KEY,
        name VARCHAR(20),
        salary FLOAT
    )
""")
```

- This block creates 3 tables (items, flavours_cake, workers) if they don't already exist:
    - items table contains columns for serial_No, products, quantity, and cost.
    - flavours_cake table holds cake flavours and their quantities with their cost.
    - workers table stores information about bakery workers (ID, name, salary).

3. Initial Data Population:

```python
# Insert initial data if tables are empty
def initialize_data():
    cur.execute("SELECT * FROM items")
    if not cur.fetchall():
        cur.executemany("INSERT INTO items (serial_No, products, quantity, cost) VALUES (%s, %s, %s, %s)", [
            (1, 'Pastry', 40, 20),
            (2, 'Milk', 60, 60),
            (3, 'Butter', 30, 25),
            (4, 'Cheese', 25, 50),
            (5, 'Whole Wheat Bread', 25, 50)
        ])
        con.commit()

    cur.execute("SELECT * FROM flavours_cake")
    if not cur.fetchall():
        cur.executemany("INSERT INTO flavours_cake (serial_No, varieties, quantity, cost) VALUES (%s, %s, %s, %s)", [
            (1, 'Vanilla_Cake', 4,200),
            (2, 'Chocolate_Cake', 6,800),
            (3, 'Strawberry_Cake', 5,400),
            (4, 'Butter_scotch_Cake', 5,600)
        ])
        con.commit()

    cur.execute("SELECT * FROM workers")
    if not cur.fetchall():
        cur.executemany("INSERT INTO workers (serial_No, name, salary) VALUES (%s, %s, %s)", [
            (1, 'Mukesh', 2500.00),
            (2, 'Ram', 2000.00),
            (3, 'Suresh', 6000.00),
            (4, 'Raju', 2500.00),
            (5, 'Amit', 5000.00)
        ])
        con.commit()

initialize_data()
```

- **initialize_data():**
    - This function checks if the items, flavours_cake, and workers tables are empty.
    - If any table is empty, it inserts default data into those tables. This ensures the database has pre-populated values for products, cake flavours, and workers.

5. Show Item, Worker, and Flavour Details:

```
# Functions to display
def show_items():
    print("Items in the shop:")
    print(f"{'Serial No.':<25}{'Product':<35}{'Quantity':<20}{'Cost':<10}")
    print("-" * 85)
    cur.execute("SELECT * FROM items")
    items = cur.fetchall()
    for serial_no, product, quantity, cost in items:
        print(f"{serial_no:<25}{product:<35}{quantity:<20}{cost:<10}")

def show_worker_details():
    print("Workers in the shop:")
    cur.execute("SELECT * FROM workers")
    workers = cur.fetchall()
    for e_id, name, salary in workers:
        print(f"{e_id}:\t{name}:\tSalary: {salary}")

def show_flavours():
    print("\nCake flavours:")
    print(f"{'Serial No.':<15}{'Variety':<20}{'Quantity':<10}{'Cost':<10}")
    print("-" * 65)
    cur.execute("SELECT * FROM flavours_cake")
    for serial_no, variety, quantity, cost in cur.fetchall():
        print(f"{serial_no:<15}{variety:<20}{quantity:<10}{cost:<10}")
```

- **show_items():** Displays all the items available in the bakery, including their serial number, product name, quantity, and cost.

- **show_worker_details():** Displays the details of all workers (ID, name, salary).

- **show_flavours():** Displays the available cake flavours along with their quantities and cost.

6. Admin Operations:

```python
# Admin operations
def add_item(serial_no, product, quantity, cost):
    cur.execute("INSERT INTO items (serial_No, products, quantity, cost) VALUES (%s, %s, %s, %s)", (serial_no, product, quantity, cost))
    con.commit()
    print("Item added successfully.")

def update_quantity(serial_no, new_quantity):
    cur.execute("SELECT quantity FROM items WHERE serial_No = %s", (serial_no,))
    result = cur.fetchone()
    if result:
        current_quantity = result[0]
        cur.execute("UPDATE items SET quantity = %s WHERE serial_No = %s", (current_quantity + new_quantity, serial_no))
        con.commit()
        print("Quantity updated successfully.")
    else:
        print("Item not found.")

def update_cost(serial_no, new_cost):
    cur.execute("UPDATE items SET cost = %s WHERE serial_No = %s", (new_cost, serial_no))
    con.commit()
    print("Cost updated successfully.")
```

1. add_item(serial_no, product, quantity, cost):

- Adds a new item to the items table with the specified serial number, product name, quantity, and cost.
- Commits the changes to the database and confirms success with a message.

2. update_quantity(serial_no, new_quantity):

- Updates the quantity of an existing item identified by its serial number in the items table.
- Adds the new quantity to the current quantity, commits the update, or displays an error if the item does not exist.

3. update_cost(serial_no, new_cost):

- Modifies the cost of an item in the items table based on its serial number.
- Commits the update and confirms the change with a message.

continue....

....continue

```python
def add_cake_flavour(serial_no, varieties, quantity, cost):
    cur.execute("INSERT INTO flavours_cake (serial_No, varieties, quantity, cost) VALUES (%s, %s, %s, %s)", (serial_no, varieties, quantity, cost))
    con.commit()
    print("Flavour added successfully.")

def update_quantity_flavours(serial_no, new_quantity):
    cur.execute("UPDATE flavours_cake SET quantity = %s WHERE serial_No = %s", (new_quantity, serial_no))
    con.commit()
    print("Cake quantity updated successfully.")

def update_cost_flavours(serial_no, new_cost):
    cur.execute("UPDATE flavours_cake SET cost = %s WHERE serial_No = %s", (new_cost, serial_no))
    con.commit()
    print("Cake cost updated successfully.")
```

1. add_cake_flavour:

- Inserts a new cake flavour into the database with details like serial number, variety, quantity, and cost.

2. update_quantity_flavours:

- Modifies the quantity of a specific cake flavour in the database using its serial number.

3. update_cost_flavours:

- Updates the price of a selected cake flavour in the database by referencing its serial number.

continue....

....continue

```
def delete_item(serial_no):
    cur.execute("SELECT * FROM items WHERE serial_No = %s", (serial_no,))
    if cur.fetchone():
        cur.execute("DELETE FROM items WHERE serial_No = %s", (serial_no,))
        con.commit()
        print(f"Item with Serial No. {serial_no} deleted successfully.")
    else:
        print(f"No item found with Serial No. {serial_no}.")
    show_items()

def add_worker(e_id, name, salary):
    cur.execute("INSERT INTO workers (serial_No, name, salary) VALUES (%s, %s, %s)", (e_id, name, salary))
    con.commit()
    print("Worker added successfully.")
    print('Workers at shop :')
    show_worker_details()

def fire_worker(employee_id):
    cur.execute("SELECT * FROM workers WHERE serial_No = %s", (employee_id,))
    if cur.fetchone():
        cur.execute("DELETE FROM workers WHERE serial_No = %s", (employee_id,))
        con.commit()
        print(f"Worker with Employee ID {employee_id} has been removed from the system.")
    else:
        print(f"No worker found with Employee ID {employee_id}.")
    show_worker_details()
```

7. delete_item(serial_no):

- Deletes an item from the items table using its serial number if it exists.
- Displays a success or error message and calls show_items to refresh the displayed inventory.

8. add_worker(e_id, name, salary):

- Inserts a new worker's details, including employee ID, name, and salary, into the workers table.
- Saves the change and confirms the worker's addition.

9. fire_worker(employee_id):

- Removes a worker from the workers table based on their employee ID if they exist.
- Displays success or error messages and refreshes the worker list using show_worker_details.

continue....

....continue

```
def remove_cake_flavour(serial_no):
    cur.execute("SELECT * FROM flavours_cake WHERE serial_No = %s", (serial_no,))
    if cur.fetchone():
        cur.execute("DELETE FROM flavours_cake WHERE serial_No = %s", (serial_no,))
        con.commit()
        print(f"Cake Flavour with Serial No. {serial_no} deleted successfully.")
    else:
        print(f"No cake flavour found with Serial No. {serial_no}.")
    show_flavours()
```

10. remove_cake_flavour(serial_no):

- Deletes a flavour of cake available in table flavours_cake using serial no. if it exists.
- Shows the available flavour in flavour_cake after removing.

7. Customer Operations :

```
def customer_order():
    orders = []
    name = input("Enter your name: ")
    phone = input("Enter your phone number: ")
    while True:
        print("\nWhat would you like to order?")
        print("1. Regular Items\n2. Cake Flavours\n3. Finish Order")
        choice = int(input("Enter your choice: "))
        if choice == 1:
            show_items()
            serial_no = int(input("Enter the serial number of the item: "))
            cur.execute("SELECT * FROM items WHERE serial_No = %s", (serial_no,))
            item = cur.fetchone()
            if item:
                quantity = int(input("Enter the quantity: "))
                if quantity > item[2]:
                    print(f"Only {item[2]} available.")
                else:
                    total_cost = quantity * item[3]
                    orders.append([item[1], quantity, item[3], total_cost])
                    cur.execute("UPDATE items SET quantity = quantity - %s WHERE serial_No = %s", (quantity, serial_no))
                    con.commit()
                    print(f"Added {quantity} {item[1]}(s) to your order.")
            else:
                print("Item not found.")
        elif choice == 2:
            show_flavours()
            serial_no = int(input("Enter the serial number of the cake flavour: "))
            cur.execute("SELECT * FROM flavours_cake WHERE serial_No = %s", (serial_no,))
            cake = cur.fetchone()
            if cake:
                quantity = int(input("Enter the quantity: "))
                if quantity > cake[2]:
                    print(f"Only {cake[2]} available.")
                else:
                    total_cost = quantity * cake[3]
                    orders.append([cake[1], quantity, cake[3], total_cost])
                    cur.execute("UPDATE flavours_cake SET quantity = quantity - %s WHERE serial_No = %s", (quantity, serial_no))
                    con.commit()
                    print(f"Added {quantity} {cake[1]} cake(s) to your order.")
            else:
                print("Cake flavour not found.")
        elif choice == 3:
            print("Finalizing your order...")
            break
        else:
            print("Invalid choice.")

    if orders:
        # Generate and display the bill
        print("\n===== BILL =====")
        print(f"Customer Name: {name}")
        print(f"Phone Number: {phone}")
        print("\nItems Ordered:")
        print(f"{'Item':<20}{'Quantity':<10}{'Cost per Unit':<15}{'Total Cost':<10}")
        print("-" * 65)
        total_amount = 0
        for item, qty, cost_per_unit, total_cost in orders:
            print(f"{item:<20}{qty:<10}{cost_per_unit:<15}{total_cost:<10}")
            total_amount += total_cost
        print("\nTotal Amount:", total_amount)
        print("================\n")

        # Save the bill to a file
        file_name = f"{name.replace(' ', '_')}_{datetime.now().strftime('%Y%m%d_%H%M%S')}_bill.txt"
        with open(file_name, "w") as file:
            file.write("===== BILL =====\n")
            file.write(f"Customer Name: {name}\n")
            file.write(f"Phone Number: {phone}\n\n")
            file.write("Items Ordered:\n")
            file.write(f"{'Item':<20}{'Quantity':<10}{'Cost per Unit':<15}{'Total Cost':<10}\n")
            file.write("-" * 65)
            for item, qty, cost_per_unit, total_cost in orders:
                file.write(f"{item:<20}{qty:<10}{cost_per_unit:<15}{total_cost:<10}\n")
            file.write(f"\nTotal Amount: {total_amount}\n")
            file.write("================\n")
        print(f"Bill saved as {file_name}.")
    else:
        print("No orders placed.")
```

continue....

STEP-BY-STEP EXPLANATION

1. Customer Details Input:

- Prompts the customer to enter their name and phone number.
- Stores this information for later use in the bill generation process.

2. Ordering Process:

- A loop runs until the customer chooses to finish the order.
- **Choice 1: Regular Items**
  - Displays available items using show_items().
  - Takes the item's serial number and fetches its details from the database.
  - Asks for the desired quantity and checks stock availability.
  - If enough stock is available, updates the database and adds the order details to the list orders.
- **Choice 2: Cake Flavours**
  - Displays cake flavours using show_flavours().
  - Works similarly to regular items: takes the serial number, checks availability, updates the database, and records the order.

continue....

continue....

- **Choice 3: Finish Order**
  - Ends the ordering process and moves to bill generation.

3. Bill Generation and Display:

- If the orders list contains any items:
  - Displays a well-formatted bill with item names, quantities, unit costs, and total amounts.
  - Calculates the total order amount.
- Saves the bill to a text file named using the customer's name and the current date-time.

4. No Orders Case:

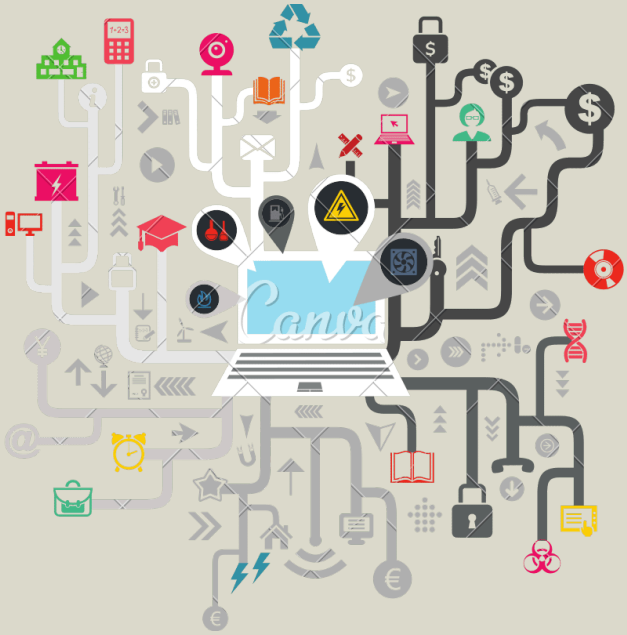
- If no orders were placed, prints "No orders placed."

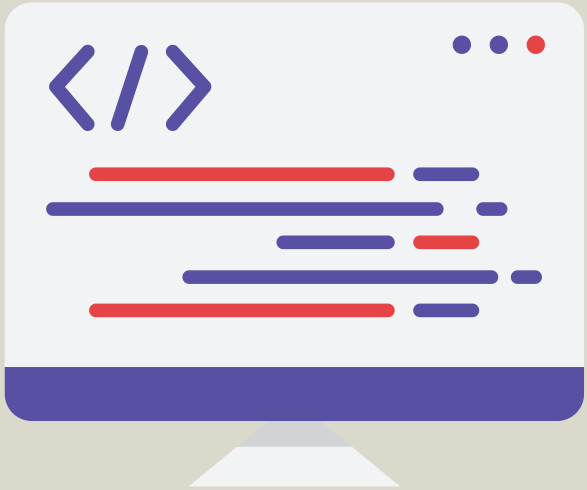
8. MAIN PROGRAM:

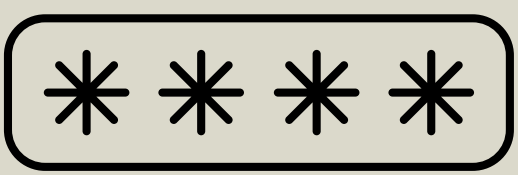
```
# Main Program
print("_____##Class 12th CS PROJECT(083)##_____")
print("_____")
print("| ..... @@@@@@@@@ WELCOME @@@@@@@@@ ..... |")
print("| .....  BAKERY MANAGEMENT SYSTEM  ..... |")
print("| .....          MADE BY:          ..... |")
print("| ..... #HARSH#, #BAIBHAV#, #ANUJ#, #AVINASH# ..... |")
print("| .....  SUBMITTED TO: R.P. KUSHWAHA SIR  ..... |")
print("| .....          SESSION: 2024-2025       ..... |")
print("_____")
```

This code prints a formatted welcome message for the Bakery Management System project. It displays:

- The project header for Class 12th CS Project.
- A welcoming message with the title Bakery Management System.
- The names of the creators (Harsh, Baibhav, Anuj, Avinash).
- The submission details to R.P. Kushwaha Sir.
- The academic session 2024-2025. The output is visually organized with decorative lines for better presentation.







continue....

....continue

```
while True:
    try:
        print("\nPlease choose:\n1. For Admin\n2. For Customer\n3. For Exit")
        choice = int(input("Enter your choice: "))

        if choice == 1:
            admin_name = input("Username: ")
            if admin_name.lower() in ['anuj', 'harsh', 'baibhav', 'avinash']:
                passkey = int(input("Enter Password: "))
                passwords = {'anuj': 8318, 'harsh': 9555, 'baibhav': 9919, 'avinash': 9511}
                if passwords.get(admin_name.lower()) == passkey:
                    print(f"Welcome, {admin_name.title()}!")
                    print("""01: Add item\n02: View items
03: Add cake flavour\n04: View cake flavours
05: Update Quantity of cake flavour\n06: Update cost of cake flavour
07: Update Quantity\n08: Update item cost
09: Delete item\n10: Delete cake flavour
11: Add worker\n12: Fire worker""")
                    admin_choice = int(input("Enter your choice: "))
                    if admin_choice == 1:
                        print("Current item:")
                        show_items()
                        add_item(int(input("Serial No: ")), input("Product: "), int(input("Quantity: ")), int(input("Cost: ")))
                    elif admin_choice == 2:
                        show_items()
                    elif admin_choice == 3:
                        print("Current Flavours:")
                        show_flavours()
                        add_cake_flavour(int(input("Serial No: ")), input("Variety: "), int(input("Quantity: ")), int(input("Cost: ")))
                    elif admin_choice == 4:
                        show_flavours()
                    elif admin_choice == 5:
                        update_quantity_flavours(int(input("Serial No: ")), int(input("New Quantity: ")))
                    elif admin_choice == 6:
                        update_cost_flavours(int(input("Serial No: ")), int(input("New Cost: ")))
                    elif admin_choice == 7:
                        update_quantity(int(input("Serial No: ")), int(input("New Quantity: ")))
                    elif admin_choice == 8:
                        update_cost(int(input("Serial No: ")), int(input("New Cost: ")))
                    elif admin_choice == 9:
                        print("Current item:")
                        show_items()
                        delete_item(int(input("Serial No: ")))
                    elif admin_choice == 10:
                        print("Current Flavours:")
                        show_flavours()
                        remove_cake_flavour(int(input("Serial No: ")))
                    elif admin_choice == 11:
                        add_worker(int(input("Employee ID: ")), input("Name: "), float(input("Salary: ")))
                    elif admin_choice == 12:
                        fire_worker(int(input("Employee ID: ")))
                    else:
                        print("Invalid choice.")
                else:
                    print("Incorrect password.")
            else:
                print("Access denied.")
        elif choice == 2:
            customer_order()
        elif choice == 3:
            print("Thanks for visiting!")
            break
        else:
            print("Invalid choice.")
    except ValueError:
        print("Invalid input. Please enter a number.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

con.close()
```

continue....

....continue

Main Menu Options:-

1. Admin Login and Operations:

- **Admin Authentication:**
  - Prompts for a username and checks if it matches predefined admins (anuj, harsh, baibhav, avinash).
  - Requests the corresponding password and verifies it.
- **Admin Options Menu:**
  - If authenticated, presents multiple admin tasks like:
    - Adding, viewing, updating, or deleting items and cake flavours.
    - Managing workers (hiring or firing).
  - Based on the chosen action, the corresponding function is called.
- **Error Handling:**
  - Displays messages for incorrect passwords, invalid choices, or unauthorized usernames.

2. Customer Orders:

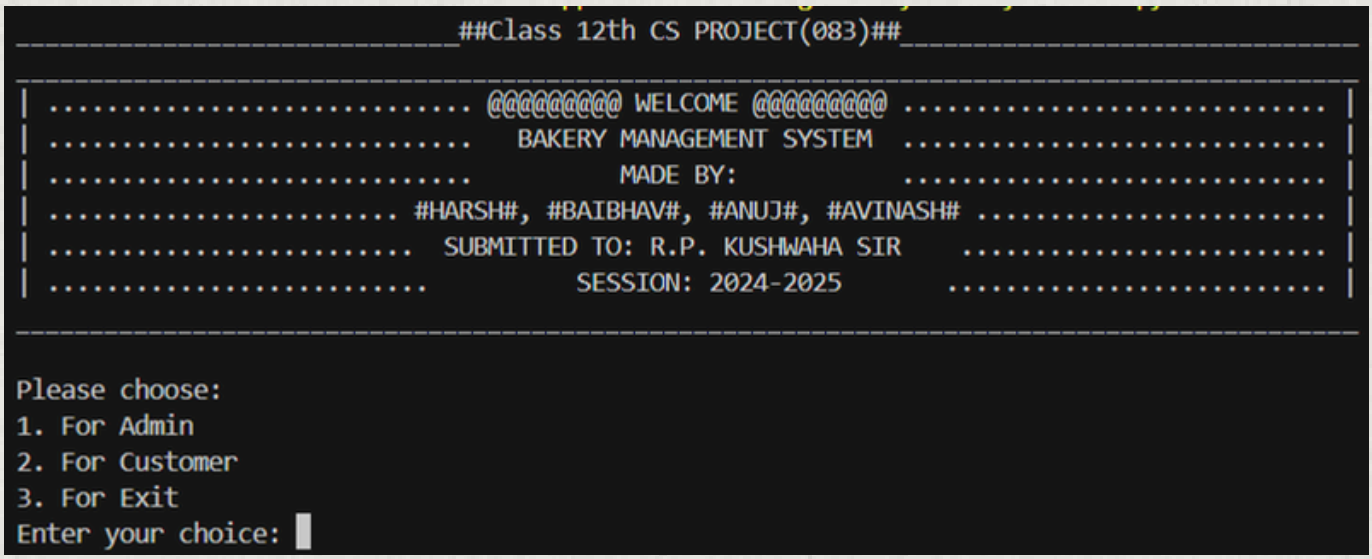
- Calls the customer_order() function to handle customer interactions and order placement.

3. Exit System:

- Displays a farewell message and breaks the loop, stopping the program.

OUTPUT OF CODE:-

OPENING PAGE:



ADMIN OPERATION OUTPUTS:

```
Enter your choice: 1
Username: HARSH
Enter Password: 9555
Welcome, Harsh!
01: Add item
02: View items
03: Add cake flavour
04: View cake flavours
05: Update Quantity of cake flavour
06: Update cost of cake flavour
07: Update Quantity
08: Update item cost
09: Delete item
10: Delete cake flavour
11: Add worker
12: Fire worker
Enter your choice:
```

STEPS:-

- LOGIN TO YOUR ADMIN ACCOUNT WITH CORRECT USERNAME AND PASSWORD.
- IF YOU ENTER CORRECT INFORMATION YOU WILL GET ACCESS OF AN ADMIN.
- YOU WILL GET ALL OPTION TO MAKE CHANGES.

1. ADDING A REGULAR ITEM

```
Enter your choice: 1
Current item:
Items in the shop:
Serial No.              Product                         Quantity          Cost
-------------------------------------------------------------------------------
1                       Pastry                          38                20
2                       Milk                            60                60
3                       Butter                          30                25
4                       Cheese                          25                50
5                       Whole Wheat Bread               25                50
Serial No: 6
Product: Cookie
Quantity: 45
Cost: 10
Item added successfully.
```

2. VIEW ITEMS

```
Enter your choice: 2
Items in the shop:
Serial No.              Product                         Quantity          Cost
-------------------------------------------------------------------------------
1                       Pastry                          38                20
2                       Milk                            60                60
3                       Butter                          30                25
4                       Cheese                          25                50
5                       Whole Wheat Bread               25                50
6                       Cookie                          45                10
```

3. ADDING A CAKE FLAVOUR

```
Enter your choice: 3
Current Flavours:

Cake flavours:
Serial No.      Variety             Quantity  Cost
-----------------------------------------------------------
1               Vanilla             4         200
2               Chocolate           6         800
3               Strawberry          5         400
4               Butter_scotch       5         600
Serial No: 5
Variety: Pineapple
Quantity: 3
Cost: 500
Flavour added successfully.
```

4. VIEWING FLAVOUR CAKE

```
Enter your choice: 4

Cake flavours:
Serial No.      Variety             Quantity  Cost
-----------------------------------------------------------------
1               Vanilla             4         200
2               Chocolate           6         800
3               Strawberry          5         400
4               Butter_scotch       5         600
5               Pineapple           3         500
```

5. UPDATE QUANTITY OF CAKE

```
Enter your choice: 5
Serial No: 5
New Quantity: 5
Cake quantity updated successfully.
```

6. UPDATE COST OF CAKE

```
Enter your choice: 6
Serial No: 4
New Cost: 300
Cake cost updated successfully.
```

7. UPDATE QUANTITY OF REGULAR ITEM

```
Enter your choice: 7
Serial No: 1
New Quantity: 35
Quantity updated successfully.
```

8. UPDATE COST OF REGULAR ITEM

```
Enter your choice: 8
Serial No: 1
New Cost: 30
Cost updated successfully.
```

9. DELETE A REGULAR ITEM

```
Enter your choice: 9
Current item:
Items in the shop:
Serial No.              Product                         Quantity        Cost
-------------------------------------------------------------------------------
1                       Pastry                          73              30
2                       Milk                            60              60
3                       Butter                          30              25
4                       Cheese                          25              50
5                       Whole Wheat Bread               25              50
6                       Cookie                          45              10
Serial No: 6
Item with Serial No. 6 deleted successfully.
Items in the shop:
Serial No.              Product                         Quantity        Cost
-------------------------------------------------------------------------------
1                       Pastry                          73              30
2                       Milk                            60              60
3                       Butter                          30              25
4                       Cheese                          25              50
5                       Whole Wheat Bread               25              50
```

10. DELETE A CAKE FLAVOUR

```
Enter your choice: 10
Current Flavours:

Cake flavours:
Serial No.      Variety             Quantity  Cost
-----------------------------------------------------------
1               Vanilla             4         200
2               Chocolate           6         800
3               Strawberry          5         400
4               Butter_scotch       5         300
5               Pineapple           5         500
Serial No: 5
Cake Flavour with Serial No. 5 deleted successfully.

Cake flavours:
Serial No.      Variety             Quantity  Cost
-----------------------------------------------------------
1               Vanilla             4         200
2               Chocolate           6         800
3               Strawberry          5         400
4               Butter_scotch       5         300
```

11. HIREING A NEW WORKER

```
Employee ID: 6
Name: Aditya
Salary: 3000.00
Worker added successfully.
Workers at shop :
Workers in the shop:
1:      Mukesh: Salary: 2500.0
2:      Ram:    Salary: 2000.0
3:      Suresh: Salary: 6000.0
4:      Raju:   Salary: 2500.0
5:      Amit:   Salary: 5000.0
6:      Aditya: Salary: 3000.0
```

12. FIREING A WORKER

```
Employee ID: 6
Worker with Employee ID 6 has been removed from the system.
Workers in the shop:
1:      Mukesh: Salary: 2500.0
2:      Ram:    Salary: 2000.0
3:      Suresh: Salary: 6000.0
4:      Raju:   Salary: 2500.0
5:      Amit:   Salary: 5000.0
```

CUSTOMER OPERATIONS OUTPUT:

```
Please choose:
1. For Admin
2. For Customer
3. For Exit
Enter your choice: 2
Enter your name: Ansh
Enter your phone number: 9912327684

What would you like to order?
1. Regular Items
2. Cake Flavours
3. Finish Order
Enter your choice:
```

STEPS:-

ENTER YOUR NAME AND 10 DIGIT MOBILE NUMBER TO CONTINUE.

WHEN YOU ENTER NAME AND NO. THE OPTIONS TO ORDER THINGS WILL BE DISPLAYED.

1. ORDER REGULAR ITEMS

```
Enter your choice: 1
Items in the shop:
Serial No.              Product                 Quantity        Cost
-------------------------------------------------------------------------------
1                       Pastry                  73              30
2                       Milk                    60              60
3                       Butter                  30              25
4                       Cheese                  25              50
5                       Whole Wheat Bread       25              50
Enter the serial number of the item: 1
Enter the quantity: 5
Added 5 Pastry(s) to your order.

What would you like to order?
1. Regular Items
2. Cake Flavours
3. Finish Order
Enter your choice:
```

2. ADDING CAKE IN ORDERS

```
Enter your choice: 2

Cake flavours:
Serial No.      Variety              Quantity  Cost
-----------------------------------------------------------------
1               Vanilla              4         200
2               Chocolate            6         800
3               Strawberry           5         400
4               Butter_scotch        5         300
Enter the serial number of the cake flavour: 2
Enter the quantity: 1
Added 1 Chocolate cake(s) to your order.

What would you like to order?
1. Regular Items
2. Cake Flavours
3. Finish Order
Enter your choice:
```

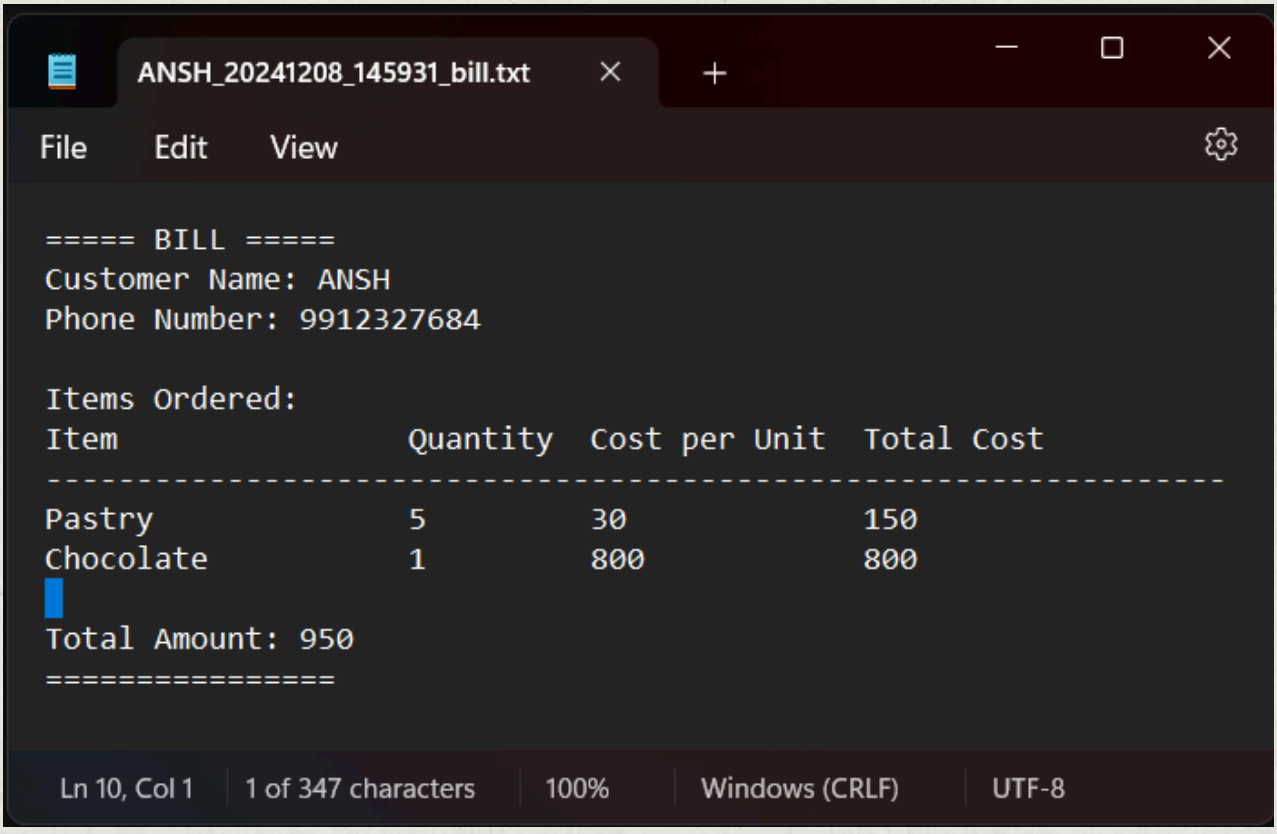
3. FINISHING THE ORDER

```
Enter your choice: 3
Finalizing your order...

===== BILL =====
Customer Name: ANSH
Phone Number: 9912327684

Items Ordered:
Item                 Quantity  Cost per Unit  Total Cost
-----------------------------------------------------------------
Pastry               5         30             150
Chocolate            1         800            800

Total Amount: 950
================

Bill saved as ANSH_20241208_145931_bill.txt.
```

EXITING THE PROGRAM:

```
Please choose:
1. For Admin
2. For Customer
3. For Exit
Enter your choice: 3
Thanks for visiting!
```

VIEWING THE ORDER IN TEXT FILE:



SQL TABLES:

DATABASE:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| bakery_management  |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (1.55 sec)
```

ITEMS TABLE:

```
mysql> select * from items;
+-----------+------------------+----------+------+
| serial_No | products         | quantity | cost |
+-----------+------------------+----------+------+
|         1 | Pastry           |       31 |   20 |
|         2 | Milk             |       60 |   60 |
|         3 | Butter           |       30 |   25 |
|         4 | Cheese           |       25 |   50 |
|         5 | Whole Wheat Bread |       25 |   50 |
+-----------+------------------+----------+------+
5 rows in set (0.09 sec)
```

WORKERS TABLE:

```
mysql> select * from workers;
+-----------+--------+--------+
| serial_No | name   | salary |
+-----------+--------+--------+
|         1 | Mukesh |   2500 |
|         2 | Ram    |   2000 |
|         3 | Suresh |   6000 |
|         4 | Raju   |   2500 |
|         5 | Amit   |   5000 |
+-----------+--------+--------+
5 rows in set (0.79 sec)
```

CAKE FLAVOUR TABLE:

```
mysql> select * from flavours_cake;
+-----------+-------------------+----------+------+
| serial_No | varieties         | quantity | cost |
+-----------+-------------------+----------+------+
|         1 | Vanilla_Cake      |        4 |  200 |
|         2 | Chocolate_Cake    |        4 |  800 |
|         3 | Strawberry_Cake   |        5 |  400 |
|         4 | Butter_scotch_Cake |        5 |  600 |
+-----------+-------------------+----------+------+
4 rows in set (0.01 sec)
```

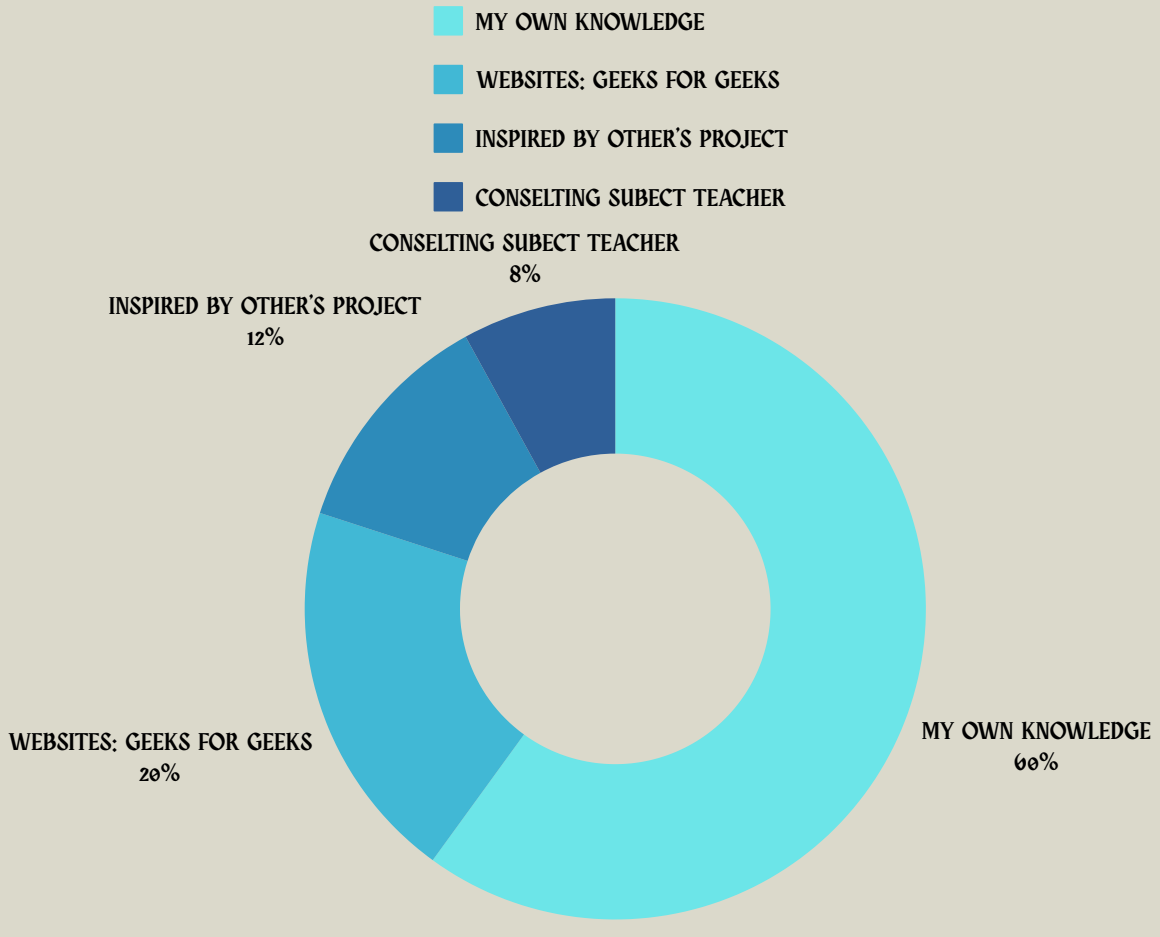
ALL TABLES IN DATABASE:

```
mysql> show tables;
+-----------------------------+
| Tables_in_bakery_management |
+-----------------------------+
| flavours_cake               |
| items                       |
| workers                     |
+-----------------------------+
3 rows in set (0.55 sec)
```

Hindrances:

- It cant calculate g.s.t and include it in bill.
- Storing passwords directly in the code is insecure.
- No input validation for products, quantities, or costs.

BIBLOGRAPHY:



THAT'S ALL

THANKYOU

VERY MUCH

HOPE YOU LIKE IT :)

"Together we dreamed, worked, and achieved. Signing off with gratitude and pride."-- HARSH SHUKLA

Until next time, stay inspired! 😊💫