Project Report: Bakery Billing System

1. Introduction

The **Bakery Billing System** is a console-based application developed using the Python programming language. It is designed to automate the ordering and billing process for a small bakery. The system replaces manual calculations with a digital interface, allowing customers (or cashiers) to select items from a predefined menu, specify quantities, and generate an instant bill with a grand total.

2. Objectives

The primary objectives of this project are:

- **Automation:** To reduce the time taken to calculate bills manually.
- **Accuracy:** To eliminate human errors in arithmetic calculations during billing.
- **User Experience:** To provide a simple, interactive text-based interface for selecting items.
- **Learning Application:** To demonstrate the practical application of Python concepts such as **Dictionaries**, **Loops** (`while`, `for`), **Lists**, and **Conditional Statements**.

3. System Requirements

To run this project, the following environment is required:

- **Operating System:** Windows, macOS, or Linux.
- **Programming Language:** Python 3.x.
- **Hardware:** Standard PC or Laptop with a keyboard.

4. Data Structures Used

- **Dictionary (`items`):** Used as a database to store the menu.
  - *Key:* Item ID (Integer)
  - *Value:* List containing `[Item Name, Price]`.
- **List (`order_details`):** Used as a temporary "shopping cart" to store the current customer's selected items, quantities, and individual costs.
- **Variables:** Used for storing user choices, running totals, and loop control.

5. Algorithm / Logic Flow

1. **Start** the program.
2. Define the menu items in a dictionary.
3. **Enter Outer Loop (New Customer):**
   - Display the menu.

- Initialize `total = 0` and an empty list for the order.

4. **Enter Inner Loop (Ordering):**
   - Ask user for `Item Number`.
   - **If** input is `0`: Break the inner loop (finish ordering).
   - **If** input is valid:
     - Ask for `Quantity`.
     - Calculate `Cost = Price * Quantity`.
     - Add details to the order list and update the `total`.
   - **Else**: Display "Invalid item".
5. **Generate Bill:**
   - Iterate through the order list and print purchased items.
   - Print the **Grand Total**.
6. **Repeat or Exit:**
   - Ask the user if they want to order again.
   - If `No`, terminate the program. If `Yes`, return to Step 3.

6. Source Code

Python

```
# Bakery Billing System

# Default bakery items stored in a dictionary
items = {
    1: ["Cake", 120],
    2: ["Bread", 40],
    3: ["Donut", 30],
    4: ["Cookie", 20]
}

while True:
    print("\n===== BAKERY MENU =====")
    for key in items:
        print(f"{key}: {items[key][0]} - ₹{items[key][1]}")
    print("0. Finish & Show Bill")

    total = 0
    order_details = []

    while True:
        try:
            choice = int(input("\nEnter item number (0 to finish): "))
        except ValueError:
            print("Please enter a valid number.")
            continue

        if choice == 0:
            break

        if choice in items:
            qty = int(input(f"Enter quantity for {items[choice][0]}: 
"))
            name = items[choice][0]
            price = items[choice][1]
            cost = price * qty
            # Append [Name, Qty, Cost] to the order list
            order_details.append([name, qty, cost])
            total += cost

            print(f"Added {qty} {name}(s)!")
        else:
            print("Invalid item number!")

    # Billing Section
    print("\n----- BILL -----")
    if len(order_details) == 0:
        print("No items purchased.")
    else:
        print(f"{'Item':<10} {'Qty':<5} {'Cost'}")
        print("-" * 25)
        for x in order_details:
            print(f"{x[0]:<10} {x[1]:<5} ₹{x[2]}")
        print("-" * 25)

    print(f"TOTAL BILL = ₹{total}")

    again = input("\nOrder again? (y/n): ")
    if again.lower() != "y":
        print("Thank you for visiting!")
        break
```

7. Sample Output

Below is a demonstration of the program during execution:

Plaintext

```
===== BAKERY MENU =====
1 Cake - ₹ 120
2 Bread - ₹ 40
3 Donut - ₹ 30
4 Cookie - ₹ 20
0. Finish & Show Bill

Enter item number (0 to finish): 1
Enter quantity: 2
Added!

Enter item number (0 to finish): 3
Enter quantity: 4
Added!

Enter item number (0 to finish): 0

----- BILL -----
Cake       x 2    = ₹ 240
Donut      x 4    = ₹ 120
TOTAL BILL = ₹ 360

Order again? (y/n): n
Thank you!
```

8. Conclusion

The Bakery Billing System successfully demonstrates how Python can be used to handle day-to-day transactional logic. It efficiently handles menu display, user input, arithmetic calculations, and formatted printing. The use of dictionaries makes the system scalable; adding new items to the menu only requires updating the dictionary, not the logic code.

9. Future Scope

To further enhance this project, the following features could be added:

1. **File Handling:** Saving the bill transactions to a `.txt` or `.csv` file for record-keeping.
2. **Admin Mode:** Allowing a user to add or remove items from the menu while the program is running.
3. **Stock Management:** Tracking the inventory so that items can be marked as "Out of Stock" when supplies run out.