

Introduction

The project titled “**Consumer Billing System**” is aimed at developing a software application to automate the billing process in a small café. In many cafés, billing is often handled manually, where the staff calculates the total cost of items ordered by customers using a calculator or by hand. This manual process can be time-consuming, prone to errors, and inefficient during peak hours when many orders are placed. To overcome these challenges, this system is designed to provide a fast, efficient, and accurate way to manage customer orders and generate bills.

The system allows café staff to input customer orders by selecting items from a predefined menu. The menu consists of popular café items such as coffee, tea, Maggie, pizza, burgers, soft drinks, pasta, fries, vada pav, momo, and ice cream. Once an order is placed, the system automatically calculates the total cost by multiplying the price of each item by its ordered quantity. The application then adds all item costs together, applying any taxes or additional charges, and generates a detailed invoice.

This system reduces the need for manual calculations, thus eliminating the chances of human error and speeding up the billing process. Additionally, it enhances customer satisfaction by providing quick and accurate billing, especially during busy times. The generated bill can be printed or displayed on a screen, making it easy for both the staff and customers to view and verify the order details.

The **Consumer Billing System** aims to improve operational efficiency in small cafés by automating a critical part of the business process. It is designed to be user-friendly, ensuring that café staff with basic computer skills can operate it without difficulty. Moreover, by streamlining the billing process, the café can offer a better overall customer experience, leading to increased customer satisfaction and repeat business.

Objective

The primary goal of this project is to design a simple, user-friendly system for generating bills in a café setting. The system should:

- Present the available menu items with their respective prices.
- Allow the user to input item numbers and quantities for each item they wish to order.
- Automatically calculate the total amount based on the selected items and their quantities.
- Print a formatted bill with itemized details and the total amount.

Features

- Displays a menu of 11 café items, including beverages, snacks, and food items, with their corresponding prices.
- Allows customers to order multiple items and specify the desired quantity for each.
- Provides the flexibility to modify the order (by re-entering a quantity for a specific item).
- Automatically calculates the total amount to be paid based on the items ordered.
- Generates a well-formatted bill displaying item names, quantities, prices, and the total amount.

Menu Items

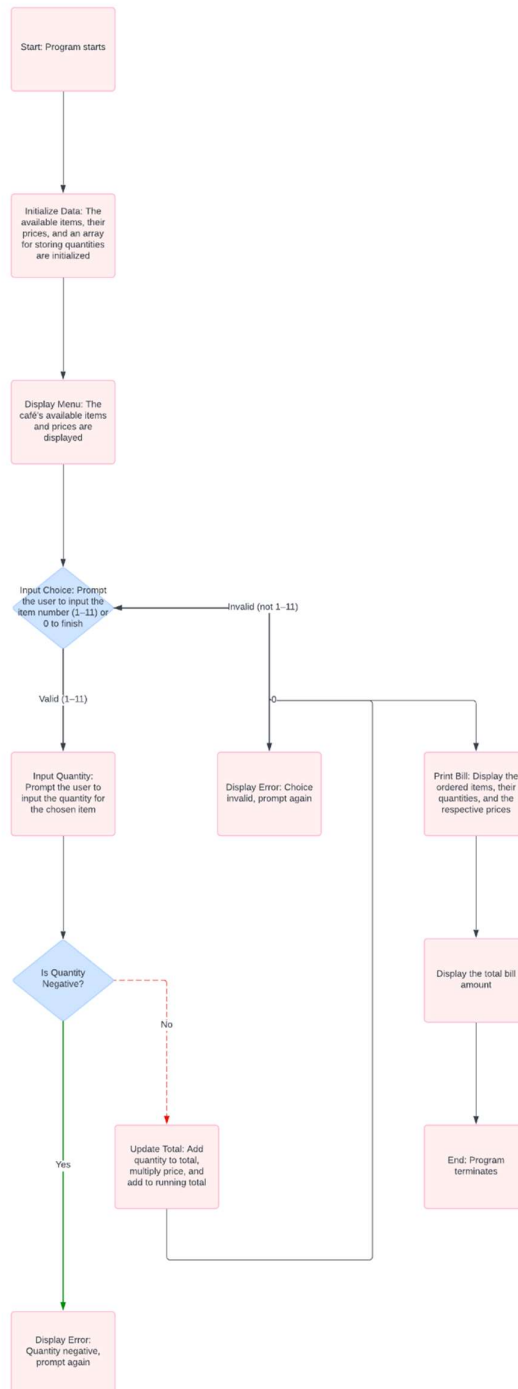
The café menu consists of the following items:

Item No.	Item Name	Price (INR)
1	Coffee	45
2	Tea	25
3	Maggie	50
4	Pizza	120
5	Burger	70
6	Soft Drink	60
7	Pasta	75
8	Fries	50
9	Vada Pav	25
10	Momo	80
11	Ice Cream	110

System Flow





1. **Menu Display:** The program starts by printing a list of available items, their prices, and item numbers.
2. **User Input:** The system allows the customer to select an item by entering its number and specify the desired quantity.
3. **Order Validation:** The system checks the validity of the selected item number and ensures that the quantity is not negative.
4. **Billing Process:** For each valid order, the system updates the running total by multiplying the item price by the quantity ordered.
5. **Bill Generation:** Once the customer finishes ordering (indicated by entering '0'), the system prints a formatted bill displaying each item ordered, its quantity, the total price for that item, and the overall amount.

Flow Chart



Code Explanation

The following code implements the consumer billing system in C:

```
main.c    Share  Run
```

```
1 #include <stdio.h>
2
3 int main() {
4     // Item names and their prices
5     const char* items[] = {
6         "Coffee", "Tea", "Maggie", "Pizza", "Burger",
7         "Soft Drink", "Pasta", "Fries", "Vada Pav", "Momo", "Ice
            Cream"
8     };
9     const int prices[] = {45, 25, 50, 120, 70, 60, 75, 50, 25, 80,
        110};
10    int quantities[11] = {0}; // To store quantity of each item
11    int choice;
12    int total = 0;
13
14    printf("\n");
15    printf("Welcome to the BackGate Cafe!");
16    printf("Available items:\n");
17    printf("-----\n");
18    printf("No.  Item          Price\n");
19    printf("-----\n");
20    for (int i = 0; i < 11; i++) {
21        printf("%d.  %-12s  %d\n", i + 1, items[i], prices[i]);
22    }
23    printf("-----\n");
24    printf("-----\n");
25    }
```

main.c



Share

Run

```
24
25 // Take orders
26 while (1) {
27     printf("Enter the item number to order (1-11), or 0 to
        finish: ");
28     scanf("%d", &choice);
29
30     if (choice == 0) {
31         break; // Exit the loop if the user is done ordering
32     } else if (choice < 1 || choice > 11) {
33         printf("Invalid choice, please try again.\n");
34         continue;
35     }
36
37     printf("Enter the quantity of %s: ", items[choice - 1]);
38     int quantity;
39     scanf("%d", &quantity);
40
41     if (quantity < 0) {
42         printf("Quantity cannot be negative. Please try again
            .\n");
43         continue;
44     }
45
46     quantities[choice - 1] += quantity; // Update quantity for
        the item
47     total += prices[choice - 1] * quantity; // Update total
        price
48
```

main.c

Share

Run

```
49
50 // Print the bill
51 printf("-----\n
    ");
52 printf("\nWelcome to the BackGate Cafe!\n");
53 printf("\nName of Customer:");
54 printf("\nBill:\n");
55 printf("-----\n
    );
56 printf("Item          Quantity      Price\n");
57 printf("-----\n
    );
58 for (int i = 0; i < 11; i++) {
59     if (quantities[i] > 0) {
60         printf("%-12s %d          %d\n", items[i],
                quantities[i], prices[i] * quantities[i]);
61     }
62 }
63 printf("-----\n
    );
64 printf("Total Amount: %d\n", total);
65 printf("Thank you for visiting! Please come again.\n");
66
67 return 0;
68 }
69
70
```

Output

Output

Welcome to the BackGate Cafe! Available items:

No. Item Price

1.	Coffee	45
2.	Tea	25
3.	Maggie	50
4.	Pizza	120
5.	Burger	70
6.	Soft Drink	60
7.	Pasta	75
8.	Fries	50
9.	Vada Pav	25
10.	Momo	80
11.	Ice Cream	110

Enter the item number to order (1-11), or 0 to finish: 1

Enter the quantity of Coffee: 2

Enter the item number to order (1-11), or 0 to finish: 4

Enter the quantity of Pizza: 1

Enter the item number to order (1-11), or 0 to finish: 11

Enter the quantity of Ice Cream: 3

Enter the item number to order (1-11), or 0 to finish: 0

Welcome to the BackGate Cafe!

Name of Customer:

Bill:

Item	Quantity	Price
Coffee	2	90
Pizza	1	120
Ice Cream	3	330

Total Amount: 540

Thank you for visiting! Please come again.

=== Code Execution Successful ===|

Code Breakdown

1. **Data Initialization:**
 - items[]: Stores the names of the items.
 - prices[]: Corresponding prices of the items.
 - quantities[]: Tracks the ordered quantity of each item.
2. **Menu Display:** A for loop prints the menu items along with their prices.
3. **Order Input:** The system uses a while loop to continuously ask for user input until the user chooses to finish ordering by entering 0.
4. **Validation:** Ensures that the chosen item number and quantity are valid.
5. **Billing Logic:** Updates the total price based on the ordered items and quantities.
6. **Bill Printing:** Displays the itemized bill along with the total amount due.

Conclusion

The "Consumer Billing System" project successfully demonstrates the automation of the billing process in a café setting. This system offers a simple and efficient solution to manage customer orders, compute the total bill, and generate an itemized receipt. The use of a command-line interface allows for straightforward interaction, where customers can easily select items from the café's menu, input quantities, and receive an accurate total bill. By eliminating the need for manual calculations, the system reduces human errors and ensures a smoother, faster checkout experience.

Key aspects of the system's effectiveness include:

1. **Efficiency and Speed:** The system enables quick data entry for orders, significantly reducing the time spent on billing. Customers can view the menu, choose items, and receive their total bill almost instantly, leading to a more efficient service.
2. **Error Minimization:** The automation of the billing process minimizes the likelihood of calculation mistakes. Manual billing can often result in errors, especially when there are many items, but this system ensures that prices and totals are calculated accurately, based on the input quantities.
3. **User-Friendly Interface:** Despite being a command-line application, the interface is intuitive. The program prompts users at each step, ensuring that even those with minimal technical knowledge can easily navigate the system. The system provides real-time feedback, correcting invalid inputs and guiding the user through the ordering process.
4. **Modularity and Expandability:** The current implementation serves as a foundation for future development. It can be expanded with additional features such as discounts, special offers, or loyalty programs to attract more customers. Advanced functionalities like generating detailed sales reports or integrating with inventory management systems can be added to help café owners streamline their operations further.
5. **Real-World Application:** This system is well-suited for small to medium-sized cafés or food outlets. It provides a practical, low-cost solution for managing orders and billing. In the real world, where customer experience is crucial, the billing system's speed and accuracy contribute significantly to customer satisfaction.
6. **Customization:** The menu items and prices are easily customizable, meaning that the system can be adapted to suit any café or small restaurant. This flexibility makes the system applicable across various types of food businesses without the need for significant changes to the underlying code.

Future Potential:

As the café business evolves, so too can this system. Integrating this system with a graphical user interface (GUI) would enhance the user experience, making it more visually appealing and interactive. A mobile or tablet-based version could allow staff to take orders directly from customers' tables, further increasing efficiency. The system can also be connected to a database, enabling the café to track sales trends, manage inventory, and perform customer analytics.

In conclusion, this project is a great example of how a simple software solution can bring immense value to everyday operations in a café. By improving the speed and accuracy of order management, the "Consumer Billing System" enhances both customer satisfaction and business efficiency. With additional enhancements, it can evolve into a comprehensive management tool that supports various aspects of running a small food business.

Future Improvements

- **Graphical User Interface (GUI):** A user-friendly GUI can be added to improve user interaction.
- **Database Integration:** The system could be linked to a database to keep track of inventory and sales records.
- **Discount Features:** Add functionality for discounts, offers, and loyalty points.