Restaurant Billing System

(with menu categories and discounts)

SUBMITTED BY
Vatsal Chawla
SAP ID: 590027526

SUBMITTED TO
Dr. Prashant Trivedi

2. Abstract

- Manual billing on paper is very slow.
- It creates a mess in busy restaurants.
- A **digital program** was created to fix this.
- It categorizes the menu (Starters, Drinks).
- It calculates bills automatically.
- It gives a 10% discount for bills over 300 INR.

Placeholder: Illustration comparing manual paperwork to digital billing

3. Problem Definition

- • **Long Wait Times:** Customers wait too long for hand-written bills.
- • **Math Errors:** Mistakes happen when adding numbers manually.
- • **Lost Records:** Paper slips get lost or damaged easily.
- • **No Tracking:** It is hard to know total sales at the end of the day.

Placeholder: Illustration showing a stressed waiter or busy restaurant scenario

4. System Design

How the Program is Designed

- A plan was made before writing code to ensure structure.
- The system loads the menu into memory for quick access.
- It uses a simple loop to accept multiple orders from a single table.
- The core job is to check the total amount and apply a discount rule if needed.

Placeholder: Diagram showing system input process and output flow

5. Algorithm and Flowchart

Steps

- Start the program.
- Show the menu to the user.
- Ask for **Item Choice** and **Quantity**.
- Check if the choice is valid (1-12).
- Add the cost to the Total.
- If Total > 300, apply **10% Discount**.
- Print the Final Bill.

Placeholder: Flowchart showing the logic of a billing system including discount decision

6. Implementation: Data Definition (Structs)

Using 'Structs' for Items

- The C `struct` feature was used to group data.
- It holds the **Name**, **Price**, and **Category** together for each food item.
- This makes the code clean and easy to manage for 12 menu items.

```c
// Define the structure for a single menu item
struct menu {
    char itemname[50];
    float price;
    char category[20];
};

// Initializing the menu array with data
struct menu m[12] = {
    {"samosa", 40, "starter"},
    {"idli", 60, "starter"},
    {"coke", 70, "drinks"},
    // ... 9 more items ...
};
```

7. Implementation: Menu Display (The Printing Part)

Printing the Menu

- A simple `for` loop iterates through the 12 items stored in the `struct` array.
- The `printf` function displays the index number, item name, and price.
- This gives the user a clear, categorized list of available choices.

```c
// Loop to display all menu items to the user
printf("\n-- MENU --\n");
for (int i = 0; i < 12; i++) {
    printf("%2d. %-20s (Category: %-8s) - Price: %.2f\n",
        i + 1,
        m[i].itemname,
        m[i].category,
        m[i].price);
}
printf("--------------------------------\n");
```

8. Implementation: Input Loop & Validation

Input Management and Safety Check

- The `while(1)` loop runs until the user signals the end of the order (inputting '0').
- Input variables `ch` (choice) and `qty` (quantity) capture the order details.
- A crucial `if` check validates that the item choice is within the valid range (1-12) to prevent crashes.
- If input is invalid, `continue` forces the loop to restart without adding to the bill.

```c
// Continuous order input loop
while(1) {
    printf("Enter item choice (1-12, 0 to finish): ");
    scanf("%d", &ch);

    if (ch == 0) break;

    // Input Validation Check
    if (ch < 1 || ch > 12) {
        printf("Invalid choice. Please try again.\n");
        continue;
    }

    printf("Enter quantity: ");
    scanf("%d", &qty);

    // Accumulate total based on price and quantity
    total = total + (m[ch - 1].price * qty);
}
```

9. Implementation: Final Bill & Discount

Discount and Final Output

- Once the ordering loop breaks, the program executes the financial logic.
- The 10% discount is applied ONLY if the accumulated `total` is 300 INR or higher.
- The final payable amount is calculated as `total - discount`.
- A detailed receipt is printed using `printf`, showing the total, discount applied, and final amount.

```c
// Check if discount condition is met (Total >= 300)
float discount = 0.0;
if (total >= 300.0) {
    discount = total * 0.10;
    printf("\n-- Congratulations! 10%% Discount Applied --\n");
}

float final = total - discount;

// Print the final bill summary
printf("\n--- FINAL BILL ---\n");
printf("Subtotal:       %.2f INR\n", total);
printf("Discount:       %.2f INR\n", discount);
printf("----------------------\n");
printf("Final Amount:   %.2f INR\n", final);
```

10. Testing and Results

- **Input Validation:** The program successfully rejected non-menu item choices.
- **Discount Test 1 (Below Threshold):** Total 290 INR resulted in 0 INR discount. (Correct)
- **Discount Test 2 (Above Threshold):** Total 310 INR resulted in 31 INR discount. (Correct)
- **Accuracy:** All item quantities and totals were added precisely.

Placeholder: Screenshot of the output window showing a correct bill with discount.

11. Conclusion and Future Work

Summary

- Manual calculation errors are removed.
- Billing is now fast and accurate.
- Code structure is clean and organized.
- Discount logic works perfectly.

Future Scope

- Save bills to a file.
- Add a graphical interface (GUI).
- Add tax calculations.

12. References

- Class notes and course material
- Programming in C by Balagurusamy
- Online C programming tutorials (YouTube)
- Discussions with project guide