Online Food ordering system

Project report submitted in partial fulfillment of the Requirements for the Award of the Degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

Student's Names with Reg. Nos

VENKATA JAGADEESH 24KB1A05A7

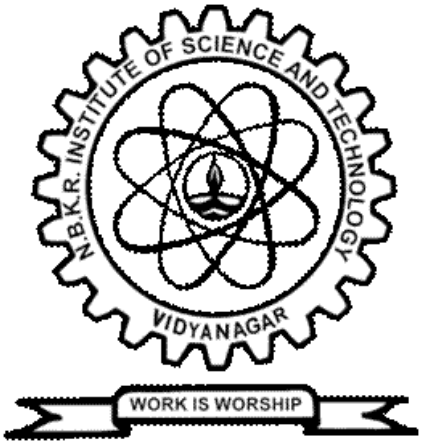
CHERALA GOWTHAM 24KB1A05A9

DUDALA AAKASH 24KB1A05E1

GANDU LOKA KEERTHAN 24KB1A05G9

Under the Guidance of

P.Suneetha Mam

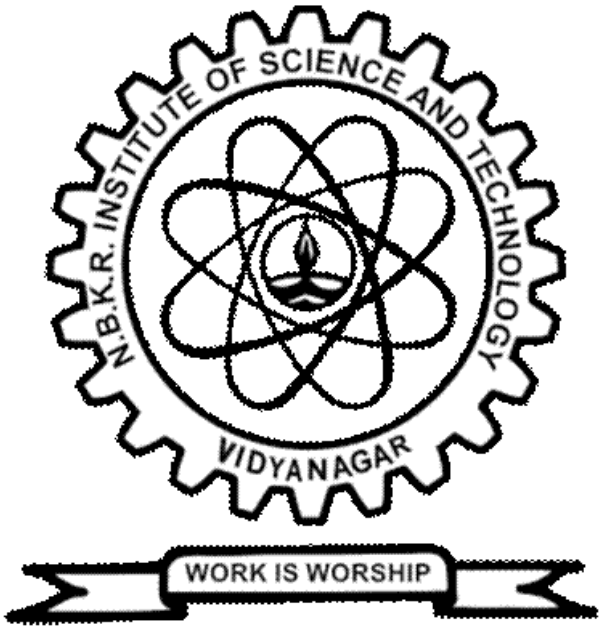


DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

This is to certify that the project report entitled YOUR PROJECT TITLE being submitted by

VENKATA JAGADEESH	24KB1A05A7
CHERALA GOWTHAM	24KB1A05A9
DUDALA AAKASH	24KB1A05E1
GANDU LOKA KEERTHAN	24KB1A05G9

in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science and Engineering to the **Jawaharlal Nehru Technological University, Anantapur** is a record of bonafied work carried out under my guidan

P.Suneetha Mam
Assistant professor

Dr. Ravindra reddy
M.Tech, Ph.D

Head of the Department

DECLARATION

I hereby declare that the dissertation entitled **Online Food ordering system**

submitted for the B.Tech Degree is my original work and the dissertation has not formed the basis for the award of any degree, associateship, fellowship or any other similar titles.

Place: Vidyanagar

Date:07/05/2025

CH.Jagadeesh

24KB1A05A7

ACKNOWLEDGEMENT :

This is a well-structured C program that simulates a basic online food ordering system for Indian dishes. Here's an acknowledgment and a quick analysis of what it does well:

1. **Structured Design:**
    - Clear use of `struct` to represent menu items and orders.
    - Modular functions like `addOrder()`, `displayMenu()`, and `displayBill()` improve code readability.
2. **Linked List Usage:**
    - Dynamic memory allocation through a linked list (`OrderNode`) allows flexible order entry without fixed limits.
3. **User Interaction:**
    - The looped menu and order input allow multiple entries until the user decides to stop**.**
4. **Accurate Billing:**
    - Calculates subtotal, applies a 10% tax, and prints a clean, formatted bill.
5. **Memory Management:**
    - Frees all dynamically allocated memory at the end, avoiding memory leaks.

Abstract:

This C program implements a **console-based Indian food ordering system** that allows users to select items from a predefined menu, specify quantities, and receive a detailed bill including tax. The system uses structured programming techniques with `struct` definitions for menu items and dynamic data structures (linked lists) to store customer orders. The menu consists of ten popular Indian dishes, each associated with a unique ID and price.

Users interact with the system by entering item numbers and quantities. These inputs are dynamically stored in a linked list representing the current order. Upon completion, the system generates a formatted bill showing each ordered item, quantity, individual cost, subtotal, 10% tax, and the total payable amount. Additionally, the program ensures efficient memory usage by freeing all dynamically allocated memory before termination. This program demonstrates fundamental concepts of **data structures, file-less transaction handling, and basic billing logic** in C, making it suitable for beginner-level projects in food service automation.

Introduction

The presented C program is a simple yet effective simulation of an **online Indian food ordering system** designed to operate via the console. It enables users to browse a fixed menu of popular Indian dishes, place orders by selecting item numbers and quantities, and receive a detailed bill that includes applicable taxes. The goal of this program is to demonstrate the application of core programming concepts such as **structures (`struct`), dynamic memory allocation using linked lists, user input handling, and formatted output**.

At the heart of the system is a menu array containing ten predefined dishes, each with a unique ID, name, and price. Orders are dynamically stored using a singly linked list, allowing flexibility in the number of items a customer can order. The program ensures that users can continue ordering until they choose to exit, after which a neatly formatted bill is displayed with a 10% tax applied.

This code serves as a foundational example for students and beginners exploring **data structures and procedural programming** in C. It models a real-world scenario, making it both educational and practical.

Project Overview

This project, titled **"Indian Food Online Ordering System"**, is a C-based console application that simulates a basic food ordering experience. It is designed to provide users with a simple interface to order food items from a fixed menu, calculate totals with tax, and display a formatted bill. The project showcases the integration of data structures, input/output operations, and memory management in a real-world context.

Key Features:

- **Menu Display**: Presents 10 popular Indian dishes with item IDs and prices.
- **Order Placement**: Allows users to choose multiple items and specify quantities interactively.
- **Dynamic Order Storage**: Uses a singly linked list to store each order entry, making it scalable and efficient.
- **Bill Generation**: Computes subtotals, applies a fixed 10% tax rate, and displays the final amount in a clean tabular format.
- **Memory Management**: Frees dynamically allocated memory before program termination to prevent memory leaks.

Objectives:

- To simulate a real-world food ordering process using C programming.
- To apply data structures (especially linked lists) in practical scenarios.
- To reinforce skills in structured programming and user interaction.

Applications:

- Educational tool for learning C programming.
- Prototype for a more advanced restaurant POS (Point of Sale) or online ordering system.
- Base for future expansion into categories, user login, file storage, or GUI-based systems.

Existing System

The existing system refers to traditional or basic approaches currently used for food ordering, especially in small-scale or non-digital environments. Before systems like the one developed in this project, food ordering was typically handled manually or with very limited automation.

Characteristics of the Existing System:

1. **Manual Order Taking**:
    - Orders are taken by waiters or over the phone and recorded on paper.
    - Prone to human errors (wrong entries, miscommunication).
2. **Limited Digital Tools**:
    - Some restaurants may use basic billing software, but these are often rigid and not tailored for dynamic or multiple item entries.
3. **No Centralized Order Management**:
    - Difficult to track, modify, or review previous orders.
    - Inventory and menu management are usually handled separately.

4. **Lack of Real-Time Billing**:
    - Calculations (subtotal, tax, discounts) are often done manually or via calculators.
    - No automatic tax addition or breakdown of costs.
5. **No User Interaction System**:
    - Customers cannot interact directly with the menu; they rely on physical menus or verbal information.

Limitations of the Existing System:

- **Error-prone** and lacks scalability.
- **Time-consuming** for both staff and customers.
- **Inefficient billing** and **no tax handling automation**.
- **Poor customer experience**, especially during rush hours.

Software Requirement Analysis

For the "Indian Food Online Ordering System" (C-based project)

1. Functional Requirements

These define the core operations that the software must perform.

- **Menu Display**

  The system must display a fixed list of Indian food items with item IDs, names, and prices.

- **Order Placement**

  Users must be able to select an item by ID and specify a quantity to add it to their order.

- **Order Storage**

  The system must store each order in memory using a linked list for dynamic handling.

- **Bill Generation**

  The system must calculate and display:

  - Item-wise subtotals
  - A cumulative subtotal
  - 10% tax

- Final total amount payable

- **Input Validation**

  The system should validate that user input is within the acceptable range (e.g., valid item numbers, positive quantities).

- **Memory Management**

  The system must properly free all dynamically allocated memory before exiting.

2. Non-Functional Requirements

These concern how the system performs its functions.

- **Performance**

  - The system must respond instantly to user inputs.

  - It should handle any number of orders within the runtime without delay.

- **Usability**

  - The interface should be simple, text-based, and user-friendly.

  - Instructions and prompts must be clear for first-time users.

- **Reliability**

  - The system must produce correct bills and calculations every time.

    - It should not crash for invalid inputs.

- **Portability**

    - Since it is written in standard C, it should compile and run on any system with a C compiler (Windows/Linux).

- **Maintainability**

    - The code should be modular and well-documented to allow future extensions, such as file handling, user authentication, or category-wise menu display.

3. Software and Hardware Requirements

Software:

- **Compiler**: GCC or any standard C compiler

- **Operating System**: Windows, Linux, or macOS

- **IDE (optional)**: Code::Blocks, Dev C++, Visual Studio Code, etc.

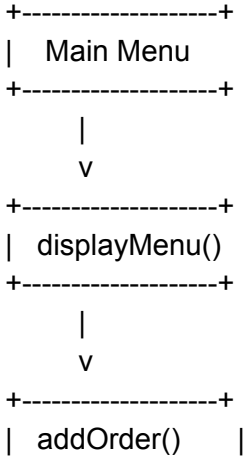
Hardware:

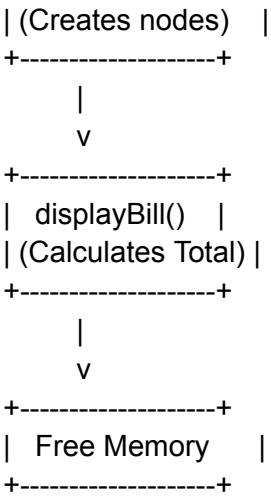
- **Processor**: Intel Pentium or above

- **RAM**: Minimum 512 MB

- **Disk Space**: Minimal (just to store source files and executables)

Software Design





Proposed System Features

Indian Food Online Ordering System (C Project)

The proposed system is designed to replace manual or semi-digital food ordering processes with a structured, interactive, and automated console application written in C. The key features of the proposed system are:

1. Interactive Menu Display

- Shows a list of 10 predefined Indian dishes.

- Displays each item's ID, name, and price in a clear format.
- Enables easy selection by number.

2. Dynamic Order Management

- Allows users to order multiple items in one session.
- Supports specifying quantity for each selected item.
- Orders are stored using a **singly linked list**, enabling flexibility and unlimited entries.

3. Real-time Bill Calculation

- Automatically calculates:
    - Per-item subtotal (`price × quantity`)
    - Total subtotal
    - Tax (fixed at 10%)
    - Final total amount
- Presents the bill in a tabular and readable format.

4. Input Validation

- Validates item number (must be between 1 and 10).
- Handles invalid entries gracefully by prompting the user to try again.

5. Efficient Memory Usage

- Uses dynamic memory allocation (`malloc`) to handle any number of orders.
- Frees all allocated memory before program termination to avoid memory leaks.

6. User-Friendly Console Interface

- Clear prompts and messages guide users through the ordering process.

- Easy to use even for beginners or non-technical users.

7. Extensibility

- The modular design allows easy enhancement in the future, such as:
    - Adding delivery address or customer name input
    - Introducing discounts or coupons
    - Categorizing items (veg, non-veg, beverages, etc.)
    - Exporting the bill to a file

Source code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MENU_SIZE 10
#define TAX_RATE 0.10  // 10% tax

// Menu item structure
typedef struct {
    int id;
    char name[30];
    float price;
} MenuItem;
```

```
// Order node structure
typedef struct OrderNode {
    int itemId;
    int quantity;
    struct OrderNode* next;
} OrderNode;

// Menu items (10 Indian dishes)
MenuItem menu[MENU_SIZE] = {
    {1, "Paneer Butter Masala", 150.00},
    {2, "Chicken Biryani", 180.00},
    {3, "Masala Dosa", 80.00},
    {4, "Aloo Paratha", 50.00},
    {5, "Gulab Jamun", 40.00},
    {6, "Chole Bhature", 90.00},
    {7, "Mutton Rogan Josh", 220.00},
    {8, "Vegetable Pulao", 100.00},
    {9, "Rasgulla", 40.00},
    {10, "Butter Naan", 25.00}
};

// Head of the linked list
OrderNode* orderHead = NULL;

// Function to add an order
void addOrder(int itemId, int quantity) {
    OrderNode* newNode = (OrderNode*)malloc(sizeof(OrderNode));
    newNode->itemId = itemId;
    newNode->quantity = quantity;
    newNode->next = NULL;

    if (orderHead == NULL) {
        orderHead = newNode;
    } else {
        OrderNode* temp = orderHead;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to display the menu
void displayMenu() {
    printf("\n--- MENU ---\n");
    for (int i = 0; i < MENU_SIZE; i++) {
        printf("%d. %s - Rs. %.2f\n", menu[i].id, menu[i].name, menu[i].price);
    }
}

// Function to display order summary and bill
void displayBill() {
```

```
    printf("\n--- BILL ---\n");
    printf("%-25s %-10s %-10s\n", "Item", "Qty", "Subtotal");
    printf("-------------------------------------------------\n");

    OrderNode* temp = orderHead;
    float subtotal = 0.0;

    while (temp != NULL) {
        MenuItem item = menu[temp->itemId - 1];
        float itemTotal = item.price * temp->quantity;
        printf("%-25s %-10d Rs. %-8.2f\n", item.name, temp->quantity, itemTotal);
        subtotal += itemTotal;
        temp = temp->next;
    }

    float tax = subtotal * TAX_RATE;
    float total = subtotal + tax;

    printf("-------------------------------------------------\n");
    printf("Subtotal:              Rs. %.2f\n", subtotal);
    printf("Tax (10%%):             Rs. %.2f\n", tax);
    printf("Total Amount:          Rs. %.2f\n", total);
}

// Main function
int main() {
    int choice, quantity;

    printf("Welcome to the Indian Food online Ordering System!\n");

    while (1) {
        displayMenu();
        printf("\nEnter item number to order (0 to finish): ");
        scanf("%d", &choice);

        if (choice == 0)
            break;

        if (choice < 1 || choice > MENU_SIZE) {
            printf("Invalid item number. Try again.\n");
            continue;
        }

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

        addOrder(choice, quantity);
        printf("Item added to order.\n");
    }

    if (orderHead == NULL) {
        printf("No items were ordered.\n");
```

```
    } else {
        displayBill();
    }

    // Free allocated memory
    OrderNode* temp;
    while (orderHead != NULL) {
        temp = orderHead;
        orderHead = orderHead->next;
        free(temp);
    }

    return 0;
}
```

OUTPUT :

```
Welcome to the Indian Food online Ordering System!

--- MENU ---
1. Paneer Butter Masala - Rs. 150.00
2. Chicken Biryani      - Rs. 180.00
3. Masala Dosa          - Rs. 80.00
4. Aloo Paratha         - Rs. 50.00
5. Gulab Jamun          - Rs. 40.00
6. Chole Bhature        - Rs. 90.00
7. Mutton Rogan Josh    - Rs. 220.00
8. Vegetable Pulao      - Rs. 100.00
9. Rasgulla             - Rs. 40.00
10. Butter Naan         - Rs. 25.00
```

```
Enter item number to order (0 to finish): 2
Enter quantity: 2
Item added to order.

Enter item number to order (0 to finish): 5
Enter quantity: 4
Item added to order.

Enter item number to order (0 to finish): 10
Enter quantity: 5
Item added to order.

Enter item number to order (0 to finish): 0
```

```
--- BILL ---
Item                          Qty         Subtotal
-------------------------------------------------------
Chicken Biryani               2           Rs. 360.00
Gulab Jamun                   4           Rs. 160.00
Butter Naan                   5           Rs. 125.00
-------------------------------------------------------
Subtotal:                     Rs. 645.00
Tax (10%):                    Rs. 64.50
Total Amount:                 Rs. 709.50
```

Conclusion

The Indian Food Online Ordering System, implemented in C, successfully demonstrates the practical application of core programming concepts such as structures, linked lists, dynamic memory management, and modular function design. It provides users with an interactive and efficient console-based interface to select food items, place orders, and generate detailed bills with tax calculations.

The system is user-friendly, scalable, and maintains data integrity through proper input validation and memory management. While it is currently a basic text-based prototype, it lays a strong foundation for future enhancements like category-based menus, discounts, file storage, and graphical interfaces.

In summary, the project not only simulates a real-world food ordering scenario but also serves as a strong educational tool for learning structured programming and data structure implementation in C.

References

1. **Mr.Sivanraj Sir, Bytexl**
   I would like to express my gratitude to Sivanraj Sir, our esteemed trainer from Bytexl, for his continuous support and guidance throughout the course. His expertise in C programming and dedication to teaching has greatly contributed to my understanding of various concepts, especially in developing an e-commerce cart system. Sir's valuable insights and feedback on the design and implementation have helped me refine the project and improve its overall functionality.

Thank you…