FOOD ORDERING SYSTEM

Aditya Raj Mishra

Sap ID-590023764

December 2, 2025

ABSTRACT

This project titled "**Food Ordering System**" is a console-based application developed in the C programming language.

The main aim of this project is to provide a basic but functional system where users can view a food menu, place orders, calculate bills with automatic tax (GST) and discount application, and provide customer feedback. The project demonstrates the effective use of C programming concepts such as **structures, file handling, arrays, functions, and conditional statements**.

By automating the calculation of totals and taxes, and by storing order history and ratings in files (orders.txt and ratings.txt), this

system eliminates manual errors and ensures data persistence. It serves as a practical example for students to understand how real-life scenarios, like restaurant billing and order management, can be implemented using fundamental logic and programming skills.

INTRODUCTION

The **Food Ordering System** is a small C language project that helps in managing food orders and billing in an easy and organized way. In daily life, dining out and ordering food is one of the most common activities, and many people visit restaurants regularly. Managing orders manually often becomes confusing because it takes time to check prices, calculate taxes, apply discounts, and update records. To reduce this difficulty, a computer-based system is needed.

This project provides a simple menu-driven program where the user can view the menu, place an order, view previous orders, and rate the service. It uses basic C programming concepts like functions, structures, arrays, loops, conditional statements, and file handling. These features help the system store order data safely and allow the user to perform actions quickly.

When a user selects the ordering option, the system asks for details like the item code and quantity. It automatically calculates the subtotal, applies any valid discount codes (like "SAVE10" or "FLAT50"), adds GST, and generates a final bill. The system stores this billing information in a file (orders.txt) so that the data remains saved even after closing the program. The user can also check previous orders, which helps in keeping track of sales history. Additionally, the system captures customer feedback and stores it in a separate file (ratings.txt).

This project is helpful for beginners because it explains how real-life systems work in a computerized way. It shows how data can be stored, processed, and managed using simple C programming. Overall, the Food Ordering System is a useful and practical project that demonstrates the application of programming in solving billing and management problems.

1. Problem Definition

1.1. Overview

1. The Food Ordering System is designed to replace manual order taking and billing with a simple computerized system.
2. Manual billing often leads to problems like calculation errors in taxes and discounts, lost paper records, slow service, and difficulty in tracking previous sales.
3. A digital system makes order processing faster, billing more accurate, and record-keeping more organized.
4. The project uses C programming concepts such as functions, structures, arrays, loops, and file handling.
5. The system allows users to view the menu, place orders, generate bills with GST, and provide feedback easily.
6. File handling ensures that all order summaries and customer ratings remain saved even after closing the program.
7. This project shows how a basic real-world restaurant billing system can be created using simple logic and programming skills.

1.2. Objectives

1. To automate the food ordering and billing process.
2. To provide a simple menu-driven interface for easy operations.
3. To store order history and customer ratings safely using file handling.
4. To reduce manual calculation errors (specifically in tax and discounts) and increase accuracy.
5. To save time by offering fast order processing and instant bill generation.

6. To help students understand real-life applications of C programming.

2. System Design

2.1. System Architecture

The Food Ordering System follows a three-layer architecture. This structure helps divide the program into smaller parts so that each part has a clear role. The three layers are:

A. Presentation Layer

- This is the top layer of the system. It is responsible for interacting with the user.
- **Functions of Presentation Layer:**
    - Displays the main menu with options like View Menu, Place Order, View Previous Orders, View Ratings, and Exit .
    - Takes input from the user such as Item Code, Quantity, Discount Code, and Service Rating .
    - Shows the output on the screen (Menu table, Bill summary with Tax breakdown, and "Order saved" messages) .
    - Ensures that the program is easy to use and understand.
    - In this project, the presentation layer is built using printf() and scanf() statements in C.

B. Business Logic Layer

- This is the middle layer of the architecture. It contains the main logic and rules of the system.
- **Functions of Business Logic Layer:**
    - Processes the user input received from the presentation layer.

    - Contains functions like:
        - takeOrder()
        - displayMenu()
        - viewRatings()
    - **Verifies conditions such as:**
        - Whether the entered Item Code exists in the menu.
        - Whether the Quantity entered is valid (positive number).
        - Calculates the Line Total (Price * Qty).
        - Applies logic for Discount Codes (e.g., "SAVE10" or "FLAT50") .
        - Calculates 5% GST automatically on the subtotal.
    - Uses structures (struct Menu, struct Order) to manage food and order data in an organized way .

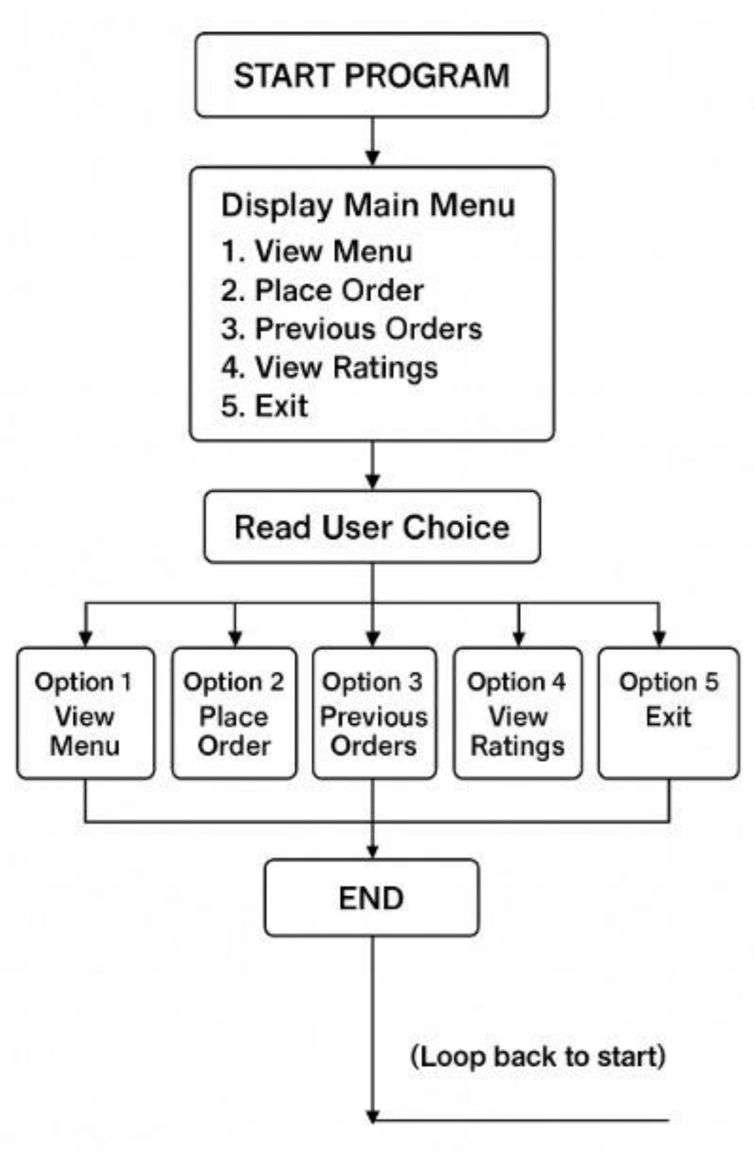
C. Data Access Layer

- This is the bottom layer of the architecture.
- It is responsible for storing and retrieving data.
- **Functions of Data Access Layer:**
    - Uses file handling (fopen, fprintf, fgetc, etc.) to store order details.
    - Saves bill summaries permanently in a file named "orders.txt".
    - Saves customer feedback and ratings in a file named "ratings.txt".
    - Fetches all stored details when the user wants to view previous orders or ratings .

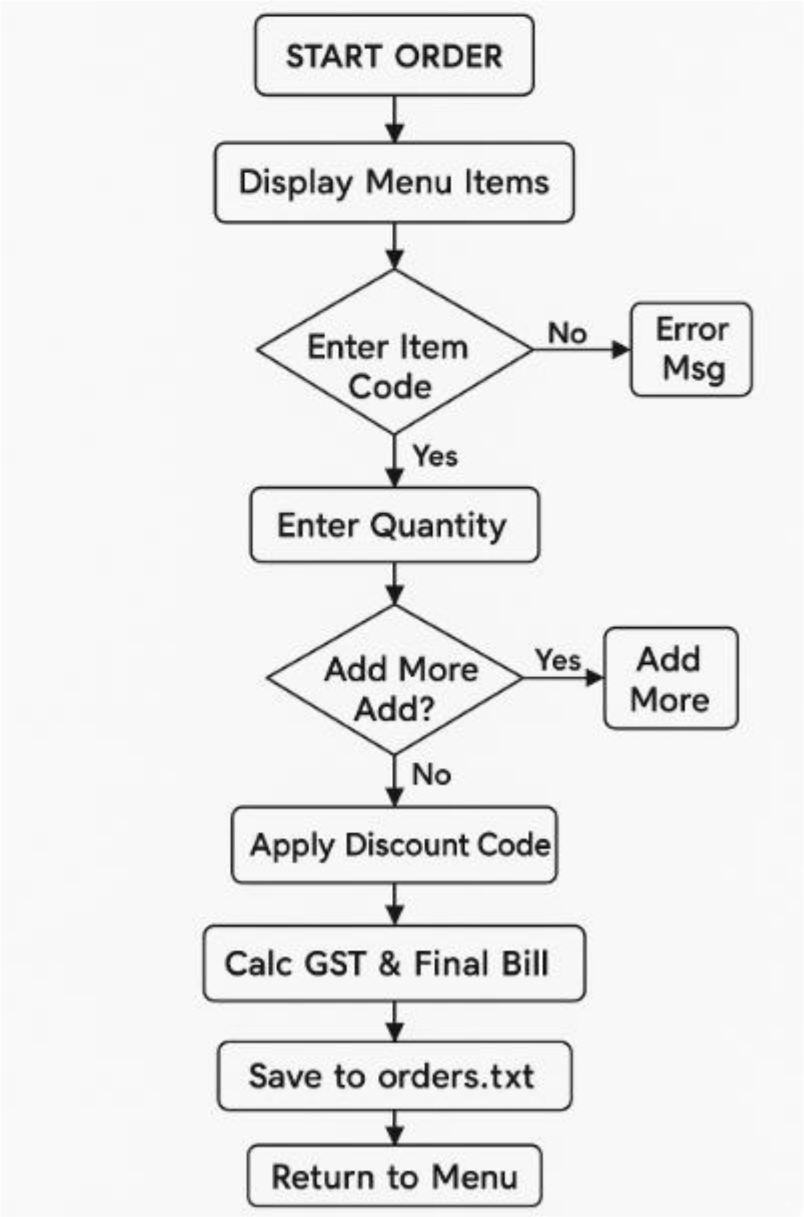
- Ensures data does not get lost even after closing the program.
- This layer makes the project work like a real system because it stores all records safely.

3. Flowchart

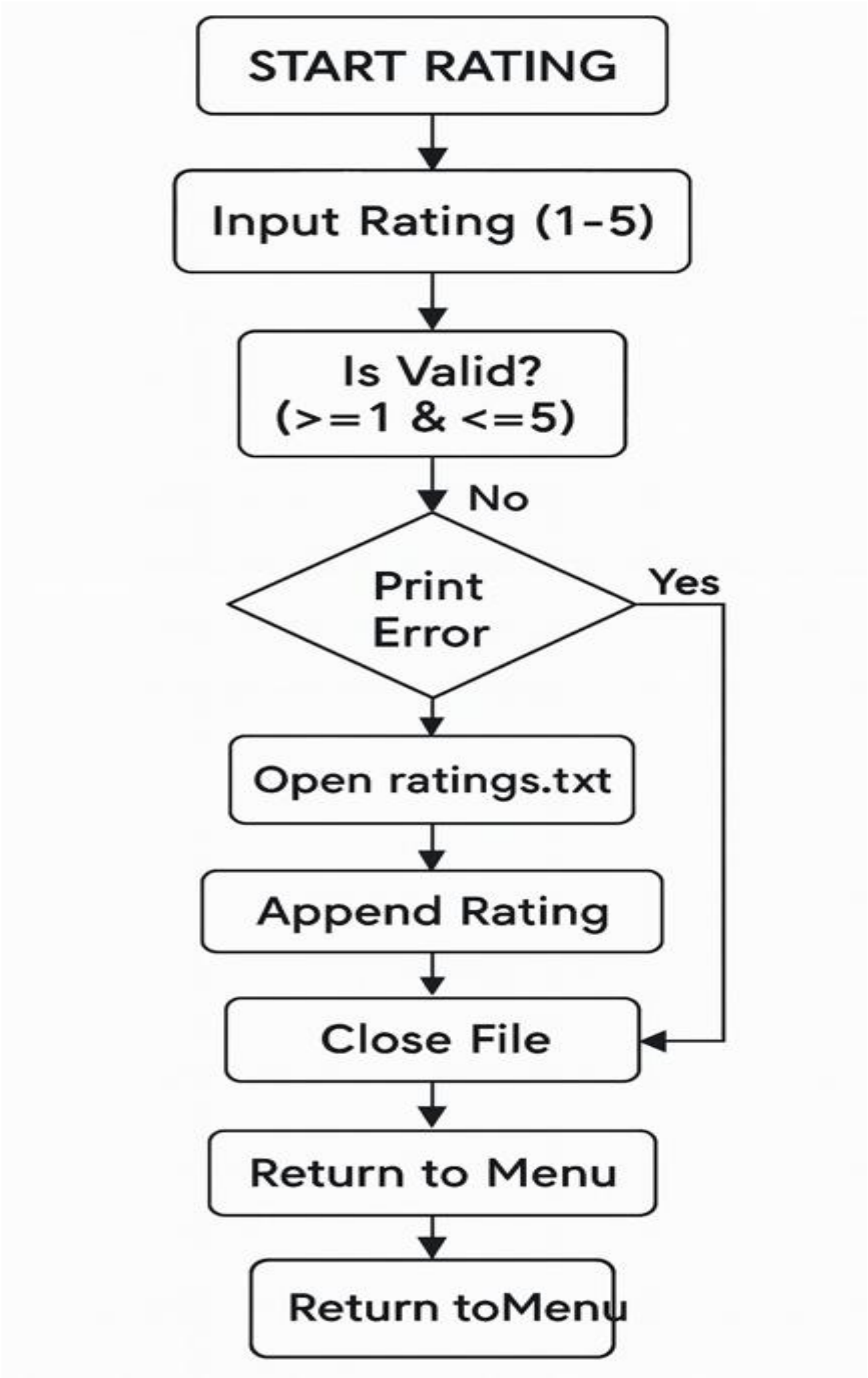
3.1. Main menu flowchart



3.2. Place Order Flowchart



3.3. Customer Rating Flowchart



4. Algorithms

1. Place Order

- Input **Item Code** and **Quantity**.
- **Check:** Is the item code valid? $\rightarrow$ If no, show error and ask again.

- **Process:**
    - Calculate Line Total = Price * Quantity.
    - Add to Grand Total.
    - Ask "Add another item?" $\rightarrow$ If yes, repeat loop.
- **Discount:** Input code. If "SAVE10" or "FLAT50", reduce total.
- **Finalize:** Calculate Tax (5%) and Final Bill. Save details to file.

2. View Previous Orders

- **Open File:** Open orders.txt in Read mode.
- **Check:** If file does not exist, show "No previous orders".
- **Display:** Read character by character until End of File (EOF) and print to screen.

3. Customer Rating

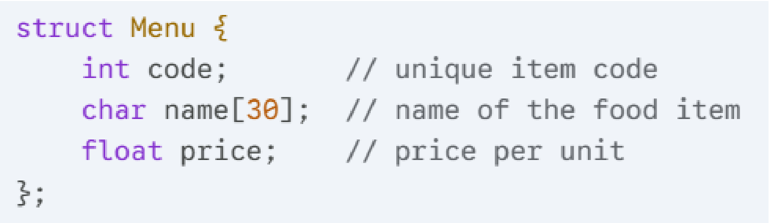
- Input **Rating** (1-5 stars).
- **Validate:** If rating < 1 or > 5 $\rightarrow$ Show error.
- **Save:** Open ratings.txt in Append mode $\rightarrow$ Write rating and bill amount.
- **Display:** Show "Thank you" message.

4. Tax Calculation

- **Logic:** Tax Amount = Subtotal * 0.05.
- **Total:** Final Payable = Subtotal + Tax Amount.

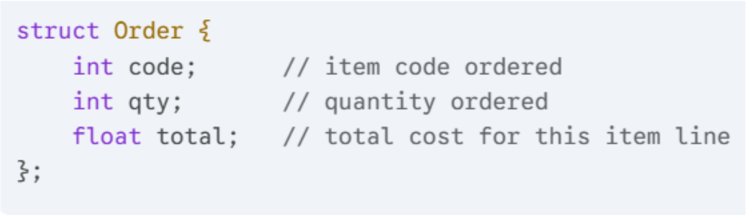
- ## 5. Data Structures
- In this project, data structures are used to store, organize, and manage the menu and order information in a proper way. The main data structure used in the Food Ordering System is the structure (struct) in C language.
- ### 5.1. User Defined Data Structures

- To store food items and order details efficiently, we use two structures named **Menu** and **Order**.
- **Structure Definition**
- A structure is a user-defined data type that allows us to group different types of information in one place. In this project, we use two primary structures:
- **1. struct Menu** Used to create the food menu with prices.



```
struct Menu {
    int code;         // unique item code
    char name[30];    // name of the food item
    float price;      // price per unit
};
```

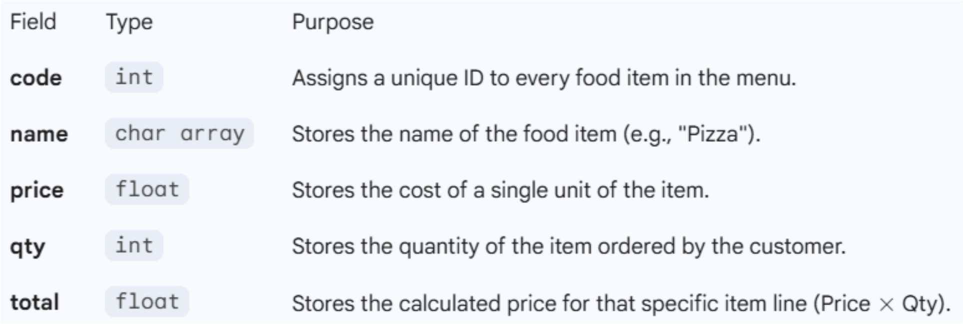
- **2. struct Order** Used to store details of items the user wants to buy



```
struct Order {
    int code;      // item code ordered
    int qty;       // quantity ordered
    float total;   // total cost for this item line
};
```

Purpose of each field:

The following table explains the fields used in these structures:



Field	Type	Purpose
code	`int`	Assigns a unique ID to every food item in the menu.
name	`char array`	Stores the name of the food item (e.g., "Pizza").
price	`float`	Stores the cost of a single unit of the item.
qty	`int`	Stores the quantity of the item ordered by the customer.
total	`float`	Stores the calculated price for that specific item line (Price × Qty).

6. Implementation Details

6.1. Key Features

1. Order Processing

- Takes item code and quantity from the user .
- Checks if the entered item code exists in the menu .
- Allows the user to add multiple items to a single order .
- Calculates the total cost dynamically as items are added .

2. View Previous Orders

- Displays all past order summaries stored in the file .
- Shows subtotal, GST, and final total bill amounts .
- Helps track sales history even after the program is closed.

3. Financial Calculations (Tax & Discounts)

- **Discounts:** Checks for valid promo codes (e.g., "SAVE10", "FLAT50") and applies deductions .
- **Taxation:** Automatically calculates 5% GST on the subtotal .
- **Final Bill:** specific logic ensures the grand total is never negative.

4. Customer Ratings

- Prompts the user to rate the service on a scale of 1 to 5 stars .
- Validates the input to ensure it is within the correct range.
- Saves the rating along with the order value to a file for feedback analysis .

5. Menu Management

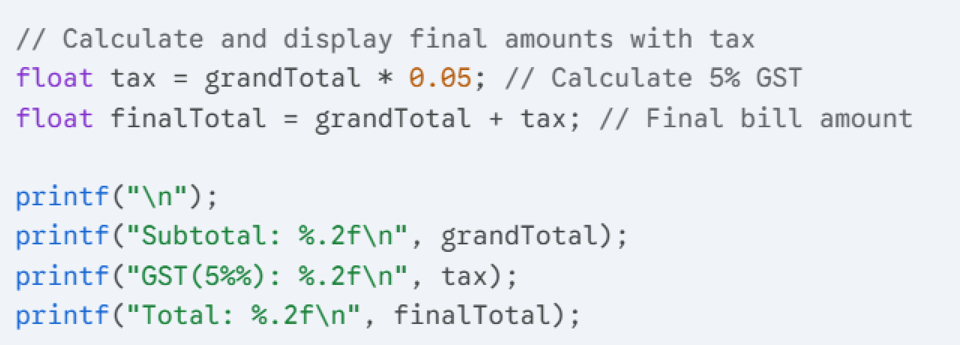
- Displays available food items in a neat tabular format .
- Shows Item Code, Name, and Price per unit to assist the user in ordering .

6. File Handling for Permanent Storage

- Uses orders.txt to store all generated bills .
- Uses ratings.txt to store customer feedback .
- Ensures that sales and feedback data are not lost when the application closes.

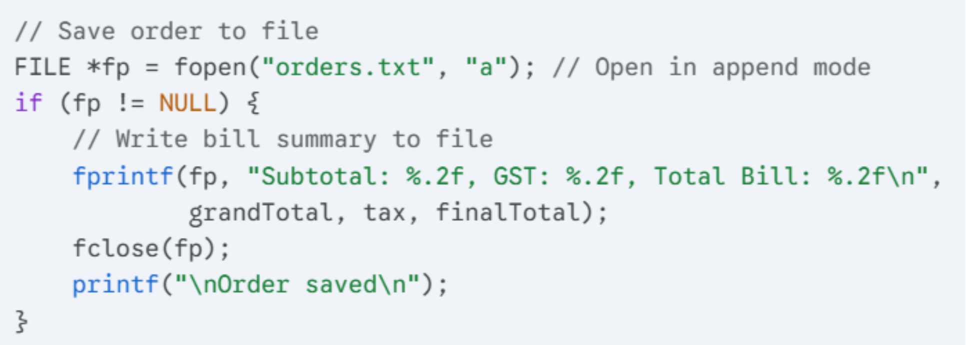
6.2. Important Code Snippets

A). Function to Calculate Bill and Tax



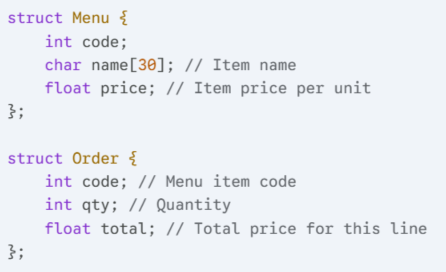
```c
// Calculate and display final amounts with tax
float tax = grandTotal * 0.05; // Calculate 5% GST
float finalTotal = grandTotal + tax; // Final bill amount

printf("\n");
printf("Subtotal: %.2f\n", grandTotal);
printf("GST(5%%): %.2f\n", tax);
printf("Total: %.2f\n", finalTotal);
```

B). File Handling for Orders



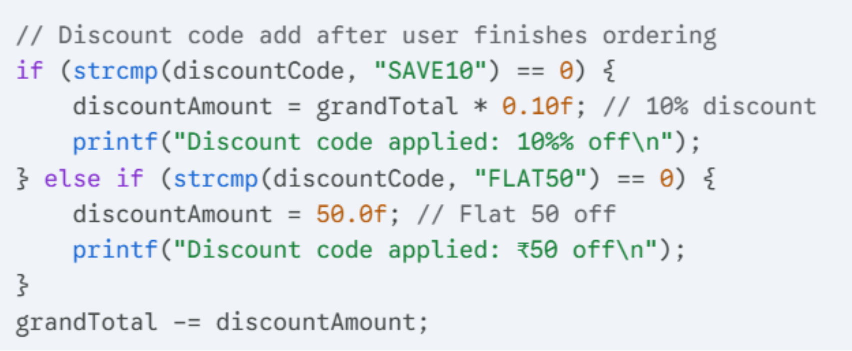
```c
// Save order to file
FILE *fp = fopen("orders.txt", "a"); // Open in append mode
if (fp != NULL) {
    // Write bill summary to file
    fprintf(fp, "Subtotal: %.2f, GST: %.2f, Total Bill: %.2f\n",
            grandTotal, tax, finalTotal);
    fclose(fp);
    printf("\nOrder saved\n");
}
```

C). Structures used for Menu and Order



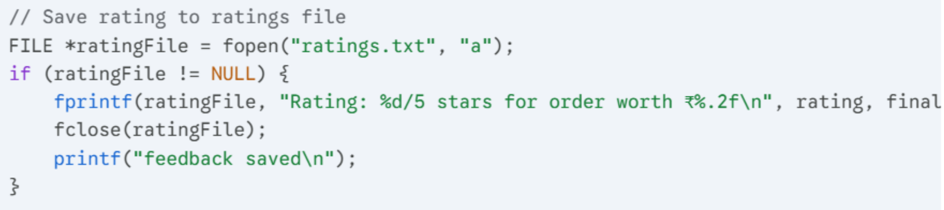
```c
struct Menu {
    int code;
    char name[30]; // Item name
    float price; // Item price per unit
};

struct Order {
    int code; // Menu item code
    int qty; // Quantity
    float total; // Total price for this line
};
```

D). Discount Logic



```
// Discount code add after user finishes ordering
if (strcmp(discountCode, "SAVE10") == 0) {
    discountAmount = grandTotal * 0.10f; // 10% discount
    printf("Discount code applied: 10%% off\n");
} else if (strcmp(discountCode, "FLAT50") == 0) {
    discountAmount = 50.0f; // Flat 50 off
    printf("Discount code applied: ₹50 off\n");
}
grandTotal -= discountAmount;
```

E). Saving Customer Ratings



```
// Save rating to ratings file
FILE *ratingFile = fopen("ratings.txt", "a");
if (ratingFile != NULL) {
    fprintf(ratingFile, "Rating: %d/5 stars for order worth ₹%.2f\n", rating, final
    fclose(ratingFile);
    printf("feedback saved\n");
}
```

7. Testing and Results

7.1. Test Cases

Test Case 1 – Placing a Valid Order

- **Input:**
  - **Item Code: 1 (Pizza)**
  - **Quantity: 2**
- **Expected Output:**
  - **Subtotal calculated ($199 \times 2 = 398$).**
  - **"Order saved" message displayed .**

Test Case 2 – Entering Invalid Item Code

- **Input:**
  - **Item Code: 99 (Not in menu)**
- **Expected Output:**
  - **"Invalid code" message.**
  - **User is prompted to enter code again.**

Test Case 3 – Viewing Previous Orders

- **Input:**
  - **Choose option 3 (View Previous Orders).**
- **Expected Output:**
  - **All past bill summaries stored in orders.txt should be displayed .**

Test Case 4 – Applying Discount Code

- **Input:**
  - **Discount Code: "SAVE10".**
- **Expected Output:**
  - **"Discount code applied: 10% off" message.**
  - **10% deduction reflected in the final total.**

Test Case 5 – Invalid Rating Input

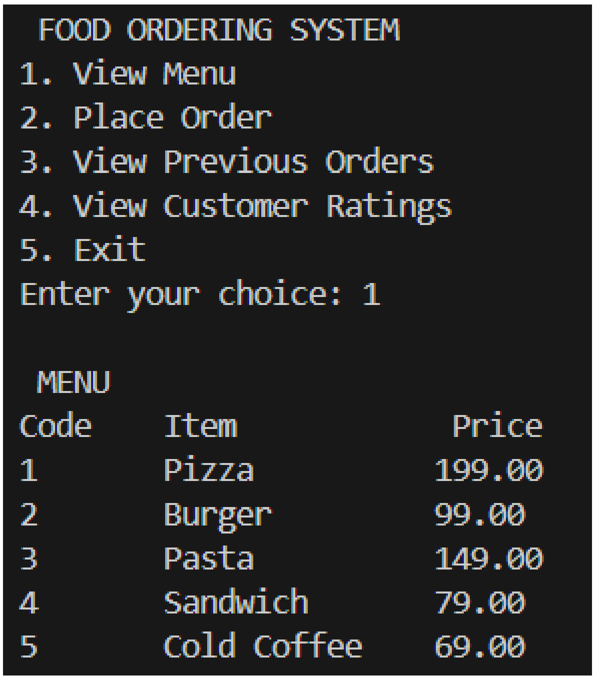
- **Input:**
  - **Rating: 6 (Range is 1-5).**
- **Expected Output:**
  - **"Invalid rating" message.**
  - **Rating is not saved to the file.**

Test Case 6 – Invalid Menu Input

- **Input:**
  - **Alphabet or symbol instead of number (e.g., 'a').**
- **Expected Output:**
  - **"Invalid input" message.**
  - **Program cleans input buffer and prompts for choice again; does not crash .**

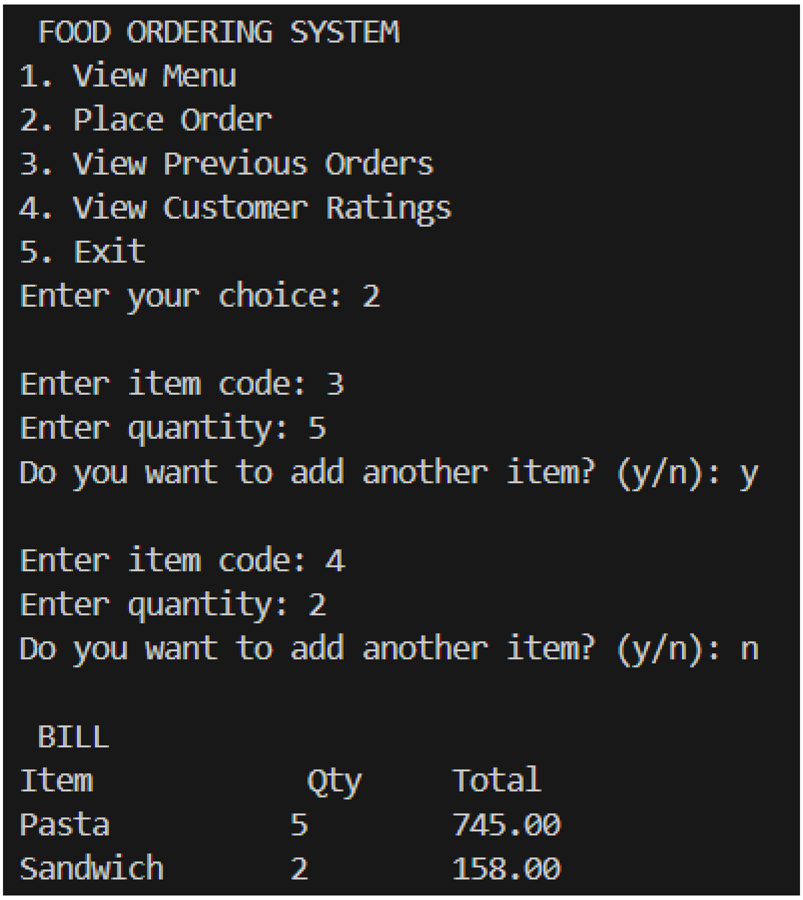
8. Outputs

Viewing menu



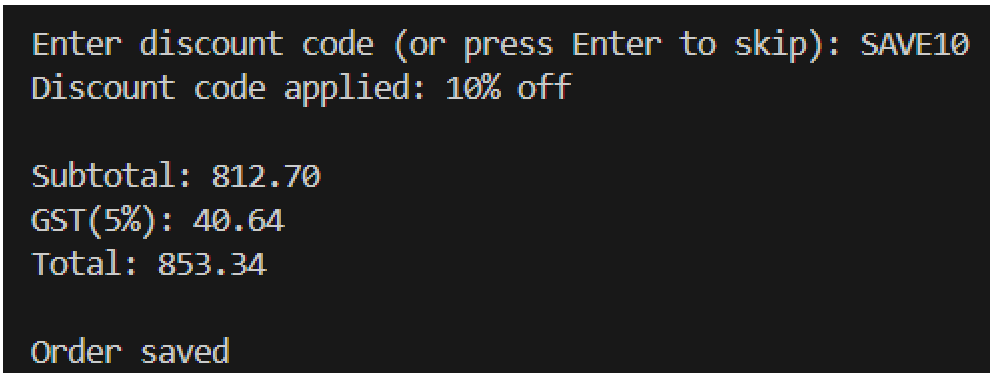
```
 FOOD ORDERING SYSTEM
1. View Menu
2. Place Order
3. View Previous Orders
4. View Customer Ratings
5. Exit
Enter your choice: 1

 MENU
Code    Item            Price
1       Pizza          199.00
2       Burger         99.00
3       Pasta          149.00
4       Sandwich       79.00
5       Cold Coffee    69.00
```

Ordering food



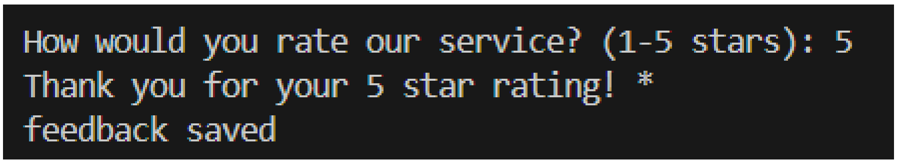
```
 FOOD ORDERING SYSTEM
1. View Menu
2. Place Order
3. View Previous Orders
4. View Customer Ratings
5. Exit
Enter your choice: 2

Enter item code: 3
Enter quantity: 5
Do you want to add another item? (y/n): y

Enter item code: 4
Enter quantity: 2
Do you want to add another item? (y/n): n

 BILL
Item            Qty     Total
Pasta          5        745.00
Sandwich       2        158.00
```

Discount code



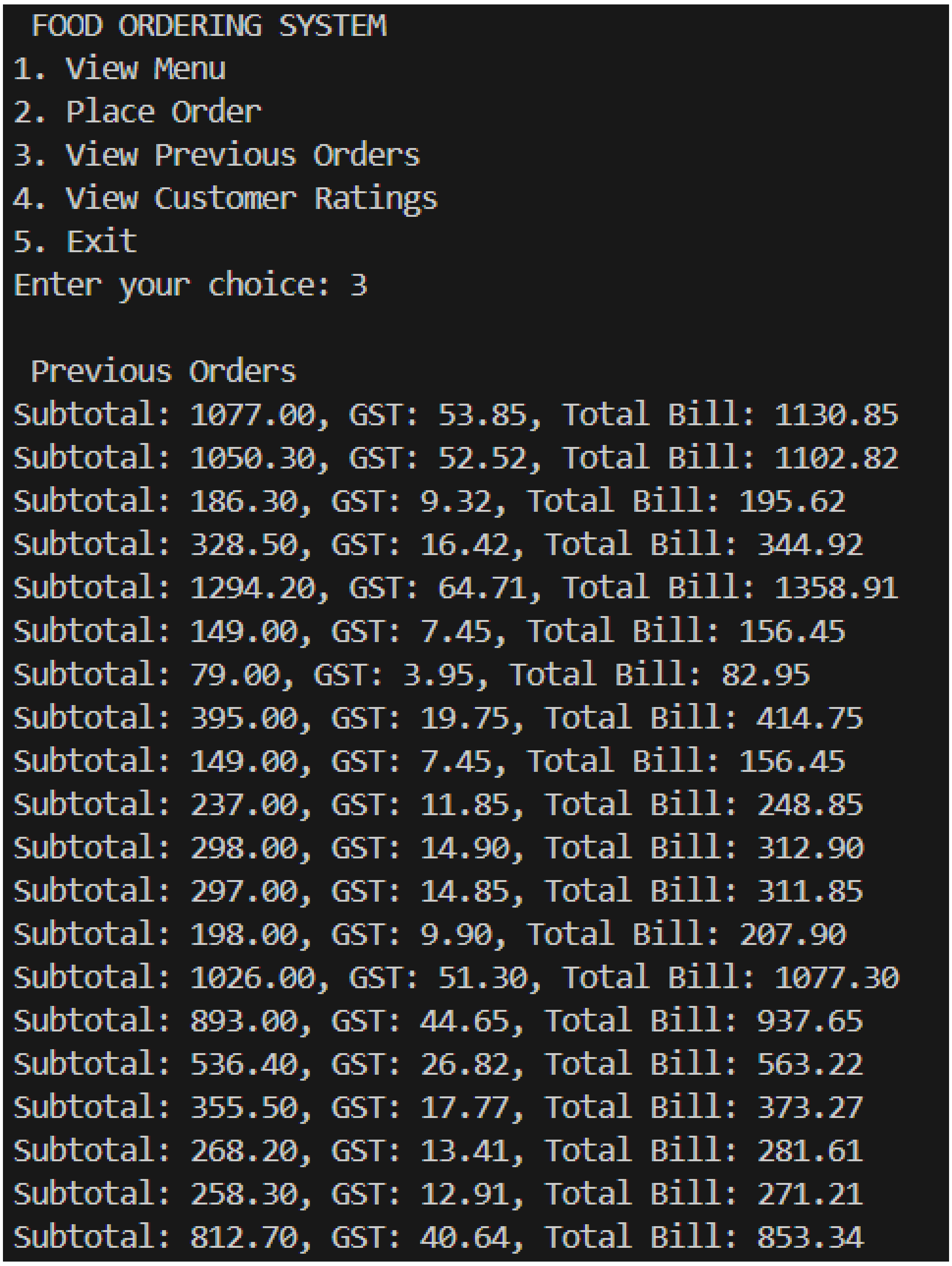
```
Enter discount code (or press Enter to skip): SAVE10
Discount code applied: 10% off

Subtotal: 812.70
GST(5%): 40.64
Total: 853.34

Order saved
```

Rating



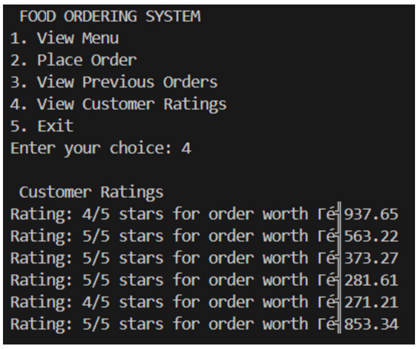
```
How would you rate our service? (1-5 stars): 5
Thank you for your 5 star rating! *
feedback saved
```

View previous orders



```
 FOOD ORDERING SYSTEM
1. View Menu
2. Place Order
3. View Previous Orders
4. View Customer Ratings
5. Exit
Enter your choice: 3

 Previous Orders
Subtotal: 1077.00, GST: 53.85, Total Bill: 1130.85
Subtotal: 1050.30, GST: 52.52, Total Bill: 1102.82
Subtotal: 186.30, GST: 9.32, Total Bill: 195.62
Subtotal: 328.50, GST: 16.42, Total Bill: 344.92
Subtotal: 1294.20, GST: 64.71, Total Bill: 1358.91
Subtotal: 149.00, GST: 7.45, Total Bill: 156.45
Subtotal: 79.00, GST: 3.95, Total Bill: 82.95
Subtotal: 395.00, GST: 19.75, Total Bill: 414.75
Subtotal: 149.00, GST: 7.45, Total Bill: 156.45
Subtotal: 237.00, GST: 11.85, Total Bill: 248.85
Subtotal: 298.00, GST: 14.90, Total Bill: 312.90
Subtotal: 297.00, GST: 14.85, Total Bill: 311.85
Subtotal: 198.00, GST: 9.90, Total Bill: 207.90
Subtotal: 1026.00, GST: 51.30, Total Bill: 1077.30
Subtotal: 893.00, GST: 44.65, Total Bill: 937.65
Subtotal: 536.40, GST: 26.82, Total Bill: 563.22
Subtotal: 355.50, GST: 17.77, Total Bill: 373.27
Subtotal: 268.20, GST: 13.41, Total Bill: 281.61
Subtotal: 258.30, GST: 12.91, Total Bill: 271.21
Subtotal: 812.70, GST: 40.64, Total Bill: 853.34
```

View customer ratings



```
 FOOD ORDERING SYSTEM
1. View Menu
2. Place Order
3. View Previous Orders
4. View Customer Ratings
5. Exit
Enter your choice: 4

 Customer Ratings
Rating: 4/5 stars for order worth Γé╣937.65
Rating: 5/5 stars for order worth Γé╣563.22
Rating: 5/5 stars for order worth Γé╣373.27
Rating: 5/5 stars for order worth Γé╣281.61
Rating: 4/5 stars for order worth Γé╣271.21
Rating: 5/5 stars for order worth Γé╣853.34
```

9. Conclusion and Future Work

9.1. Conclusion

The Food Ordering System successfully meets its goal of providing a simple and efficient way to place food orders, view history, and generate bills. It reduces manual calculation errors (such as GST and discounts) and makes the billing process faster and more organized. By using C programming concepts like structures, functions, and file handling, the project shows how real-life problems can be solved through basic programming logic. Overall, the system is easy to use, reliable for small-scale usage, and helps understand how software applications are designed and implemented.

9.2. Future Work

Graphical User Interface (GUI): Instead of a text-based console, a GUI can be created using GTK or any other C-based toolkit to make the system more attractive and user-friendly.

Online Ordering System: The project can be extended into a web-based application where users can order food online using internet connectivity.

Database Integration: Instead of storing data in text files (orders.txt and ratings.txt), a proper database like MySQL or SQLite can be used for better security, data integrity, and faster access.

User Login System: Admin and user login features can be added. The Admin can have access to modify the menu and prices, while users can only place orders.

Search Functionality & Categories: The system can be improved by adding food categories (Starters, Main Course, Drinks) and a search bar to find items by name instead of code.

Payment Integration: Online payment options (UPI, Credit Card) can be introduced to make the system more realistic.

Digital Receipt with QR Code: The system can generate a digital invoice with a QR code that can be scanned for payment or order verification.

10. References

1. **Balagurusamy, E.** – *Programming in ANSI C*, Tata McGraw-Hill.

2. **Byron Gottfried** – *Programming with C*, Tata McGraw-Hill Education.
3. **Yashavant Kanetkar** – *Let Us C*, BPB Publications.
4. **Online Tutorials** – GeeksforGeeks (C Programming Basics and File Handling).
5. **Online Documentation** – TutorialsPoint (C Language Concepts, Structures, and Functions).
6. Lecture notes provided by the faculty for C programming basics.
7. Sample reference codes from GitHub (Open-source C projects for learning only).

11. Appendix

A. File Structure Description

- **orders.txt : Stores all generated bill summaries and order history[2].**
- **ratings.txt : Stores customer feedback and ratings[3].**
- **main.c : Contains the complete source code (Menu, Order Logic, and File Handling functions).**

B. Tools Used

- **Language: C Programming**
- **Compiler: GCC**
- **Editor: VS Code / CodeBlocks / Turbo C (any used by student)**