CHANDIGARH UNIVERSITY

Discover. Learn. Empower.

UNIVERSITY INSTITUTION OF COMPUTING

BRANCH:BACHELOR OF COMPUTER APPLICATION

Minor Project: Restaurant billing system

OBJECT ORIENTED PROGRAMMING

SUBMITTED BY-

NAME: Ritika Rastogi

UID: 24BCU10027

SECTION: 24BCU-1

NAME: Tamanna Sareen

UID: 24BCA20011

SECTION: 24BCV-1

SEMESTER: 3rd

SUBMITTED TO-

Mr.Jitendra Sir

Assistant Professor

INTRODUCTION

The Restaurant Billing System is a mini project developed in C++ using the fundamental concepts of Object-Oriented Programming
The main goal of this project is to automate the process of generating restaurant bills, making it quick, easy, and error-free.
Instead of calculating totals manually, this system allows users to select menu items, enter quantities, and view the total bill instantly through a console-based interface.

In today's fast-paced world, restaurants and cafes require efficient billing systems to handle multiple orders quickly and accurately. This project demonstrates how OOP concepts such as classes, objects, inheritance, constructors, destructors, and dynamic memory allocation can be used to design such a system effectively.

The system allows users to:

- View the available menu items,
- Select the food items and enter quantities,
- Automatically calculate and display the total amount, and
- Generate a final bill for the customer.

This project serves as a simple and practical example of how C++ can be used to develop real-world applications that are both structured and efficient.

ABSTRACT

The Restaurant Billing System is a console-based mini project created in C++ to demonstrate the practical use of Object-Oriented Programming (OOP) concepts in solving daily life problems.

The system helps restaurant staff or customers calculate bills easily by selecting menu items and quantities. Each item has a name, price, and quantity, and the program automatically computes the total amount due.

The project makes use of:

- Constructors to initialize menu items,
- Inheritance to differentiate between categories like *Drinks* and *Main Course*,
- Polymorphism to handle price calculations differently for various types of food,
- Destructors to clean up memory after billing is done, and
- Dynamic memory allocation to create and manage objects during runtime.

This project aims to help learners understand how to implement a simple and functional billing system using C++. It can be further expanded in the future with features like GST calculation, item removal, discounts, and digital receipts.

SYSTEM CONFIGURATION

The Restaurant Billing System is designed to be lightweight, portable, and compatible with multiple operating systems. It requires only a basic C++ environment for compilation and execution.

Hardware Requirements

Component	Specification
Processor	Intel Dual Core or above
RAM	Minimum 2 GB
Storage	500 MB free space
Display	1024×768 or higher

Software Requirements

Component	Specification
Operating System	Windows / Linux / macOS
Compiler	Turbo C++, GCC, or Code::Blocks
Editor/IDE	Dev C++, Visual Studio Code, or Code::Blocks
Language Used	C++ (Object-Oriented Programming)

<u>SYSTEM DESIGN AND WORKING</u>

System Flow Diagram (Textual Representation)

```
+---------------------------+
|  Start the Application    |
+---------------------------+
           |
           v
+---------------------------+
| Display Menu to Customer  |
+---------------------------+
           |
           v
+---------------------------+
| Enter Customer Details    |
+---------------------------+
           |
           v
+---------------------------+
| Choose Item and Quantity  |
+---------------------------+
           |
           v
```

```
+---------------------------+
| Calculate Total Bill (price × qty) |
+---------------------------+
          |
          v
+---------------------------+
| Display Final Bill Details |
+---------------------------+
          |
          v
+---------------------------+
|  Memory Released by Destructor  |
+---------------------------+
          |
          v
+---------------------------+
|         End          |
+---------------------------+
```

Workflow Explanation:

1. **The program begins by displaying the menu to the user.**
2. **The user enters their name, selects an item, and inputs the quantity.**
3. **The system uses object-oriented methods to fetch item details, calculate the cost, and display the bill.**

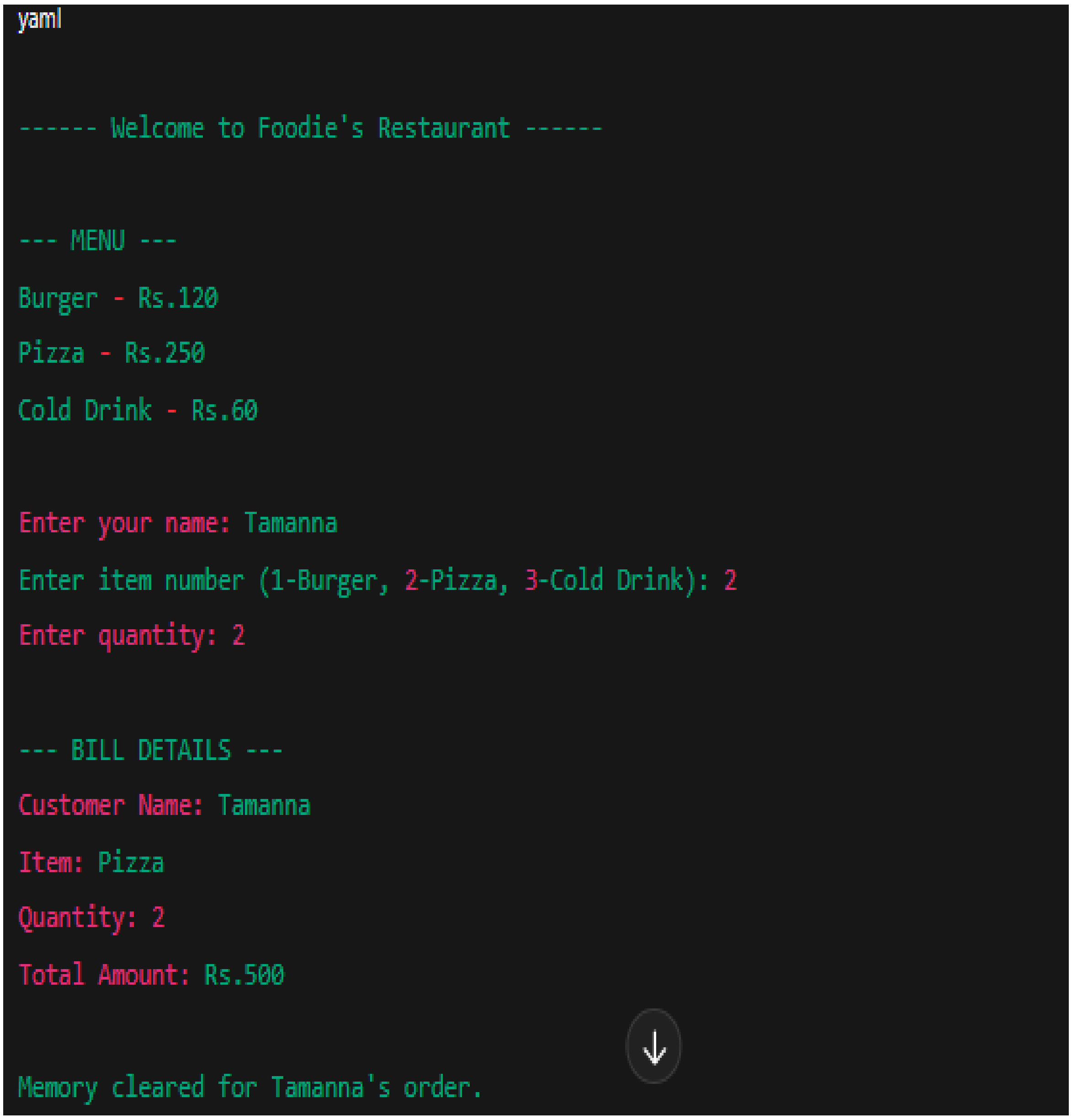
CODE IMPLEMENTATION

Below is the source code of the Restaurant Billing System, written using core C++ concepts.

```
#include <iostream>
#include <string>
using namespace std;

// Class representing a menu item
class MenuItem {
   string name;
   float price;

public:
   MenuItem(string n, float p) {
      name = n;
      price = p;
   }

   void display() {
      cout << name << " - Rs." << price << endl;
   }

   float getPrice() { return price; }
```

```
    string getName() { return name; }
};

// Class representing a customer order
class Order {
    string customerName;
    MenuItem *item;
    int quantity;

public:
    Order(string cname, MenuItem *i, int q) {
        customerName = cname;
        item = i;
        quantity = q;
    }

    void showBill() {
        cout << "\n--- BILL DETAILS ---\n";
        cout << "Customer Name: " << customerName << endl;
        cout << "Item: " << item->getName() << endl;
        cout << "Quantity: " << quantity << endl;
        cout << "Total Amount: Rs." << item->getPrice() * quantity <<
endl;
    }
    ~Order() {
        cout << "\nMemory cleared for " << customerName << "'s order.\n";
    }
};

// Main function
int main() {
    cout << "------ Welcome to Foodie's Restaurant ------\n";

    MenuItem m1("Burger", 120.0);
    MenuItem m2("Pizza", 250.0);
    MenuItem m3("Cold Drink", 60.0);

    cout << "\n--- MENU ---\n";
    m1.display();
    m2.display();
    m3.display();

    string name;
    int choice, qty;

    cout << "\nEnter your name: ";
    getline(cin, name);

    cout << "Enter item number (1-Burger, 2-Pizza, 3-Cold Drink): ";
    cin >> choice;

    cout << "Enter quantity: ";
    cin >> qty;

    MenuItem *selectedItem;
    if (choice == 1) selectedItem = &m1;
    else if (choice == 2) selectedItem = &m2;
    else selectedItem = &m3;

    Order *o1 = new Order(name, selectedItem, qty);
    o1->showBill();

    delete o1;
    return 0;
}
```

OUTPUT



yaml

```
------ Welcome to Foodie's Restaurant ------

--- MENU ---
Burger - Rs.120
Pizza - Rs.250
Cold Drink - Rs.60

Enter your name: Tamanna
Enter item number (1-Burger, 2-Pizza, 3-Cold Drink): 2
Enter quantity: 2

--- BILL DETAILS ---
Customer Name: Tamanna
Item: Pizza
Quantity: 2
Total Amount: Rs.500

Memory cleared for Tamanna's order.
```

ADVANTAGES OF THE PROJECT

- **Automation: Removes the need for manual calculations.**
- **User-Friendly: Simple and clean console interface.**
- **Error-Free: Reduces human errors in billing.**
- **Expandable: Can easily be upgraded with new features.**
- **Practical Learning: Strong demonstration of real-world OOP applications.**

FUTURE ENHANCEMENTS

To make the system more realistic and user-oriented, the following improvements can be added:

- **Graphical User Interface (GUI) using Qt or Tkinter.**
- **GST and Discount calculation for accurate billing.**
- **Database Integration (MySQL or SQLite) to save past orders.**
- **Multiple Item Ordering in one transaction.**
- **Online Payment Integration.**
- **Printable and Digital Receipts (PDF generation).**

CONCLUSION

The Restaurant Billing System project serves as an excellent example of how Object-Oriented Programming (OOP) concepts can be applied to solve real-world problems efficiently. The project demonstrates the transition from theoretical understanding to practical implementation of C++ programming skills.

Through the development of this system, we learned the significance of structuring programs into logical and manageable components using classes and objects. Each class in the program represents a real-world entity — such as menu items or customer orders — making the design both modular and easy to understand.

The use of constructors and destructors helped us realize the importance of automatic initialization and cleanup in a program's lifecycle. Encapsulation was used to keep important data, like item names and prices, secure and accessible only through class methods, thus maintaining data integrity. Abstraction allowed us to focus on what the system does (such as displaying bills) rather than how it internally processes each step.

The project also showcases dynamic memory allocation, which gives flexibility by allowing objects to be created during runtime as per user input. Additionally, by displaying the menu, taking user input, calculating the total, and generating the final bill, the system mimics a real-life restaurant billing process — all within a simple console-based environment.

Working on this project not only improved our technical coding skills but also enhanced our problem-solving abilities, logical thinking, and understanding of object-oriented design principles. It helped us comprehend the power of OOP in designing programs that are reusable, scalable, and easy to maintain.

In practical terms, this project lays the foundation for building more advanced systems in the future, such as restaurant management software, point-of-sale systems, and e-commerce billing modules. With further improvements — like integrating a database for record keeping, adding a graphical user interface (GUI), or including tax and discount features — this system could be transformed into a fully functional professional application.

Overall, the Restaurant Billing System project successfully fulfills its objectives: it simplifies the billing process, demonstrates key OOP principles, and provides a valuable hands-on learning experience for students of computer programming. This project not only deepened our knowledge of C++ and OOP concepts but also instilled confidence to design and develop real-world applications independently.

REFERENCES

1. *Programming in C++* by E. Balagurusamy
2. *Object-Oriented Programming with C++* by Robert Lafore
3. Official C++ Documentation — https://cplusplus.com
4. TutorialsPoint – https://www.tutorialspoint.com/cplusplus
5. GeeksforGeeks — https://www.geeksforgeeks.org/oops-concepts-in-cpp