COIMBATORE INSTITUTE OF TECHNOLOGY

(GOVERNMENT AIDED AUTONOMOUS INSTITUTION)

2023-2024 -SEMESTER-II-B.E.CSE

RESTAURANT MANAGEMENT SYSYTEM - USING C++

Submitted BY

Tech Challengers

*71762305040

*71762305061

ABSTRACT OF THE PROJECT

- This C++ program is an intricately designed Restaurant Management System, meticulously crafted to facilitate seamless operations within a dining establishment.
- Leveraging essential libraries such as iostream, string, and iomanip, it orchestrates diverse functionalities ranging from customer management to order processing with finesse and precision.
- This system offers a comprehensive suite of features, including customer registration, menu presentation, order placement, billing, reservation handling, and even a loyalty program to engage patrons effectively.
- Its modular architecture, driven by a user-friendly interface, ensures operational efficiency and superior customer service.
- Through meticulous attention to detail and robust implementation, this software empowers restaurant operators to optimize their workflows and elevate the overall dining experience.

Header Files

#include <iostream>
#include <string>
#include <iomanip>

#include <iostream>:

This header file is part of the C++ Standard Library and is used for input-output operations. It allows the program to read from the standard input (cin) and write to the standard output (cout).

#include <string>:
This header file allows the use of the std::string class, which is used to represent and manipulate sequences of characters.

#include <iomanip>:
This header file is used for input and output manipulators. In this code, it is used to format the output, such as setting the width of the output fields using setw.

Constants and Structs

```
//code
const int MAX_CUSTOMERS = 100;
const int MAX_MENU_ITEMS = 50;
const int MAX_ORDERS = 200;

struct MenuItem {
    int id;
    std::string name;
    float price;
};
```

```
struct Customer {
    int tableNo;
    string name;
    string phone;
    float billAmount;
    string orders[MAX_ORDERS];
    float orderPrices[MAX_ORDERS];
    int orderCount;
    bool reserved;
    int loyaltyPoints;
};
```

Constants:

- MAX_CUSTOMERS: Maximum number of customers the system can handle.
- MAX_MENU_ITEMS: Maximum number of menu items the system can store.
- MAX_ORDERS: Maximum number of orders a single customer can make.

Structs:

- MenuItem: Represents a menu item with an ID, name, and price.
- Customer: Represents a customer with various attributes such as table number, name, phone number, bill amount, orders, reservation status, and loyalty points.

Class Declaration

//code

```
class Restaurant {
private:
   Customer customers[MAX_CUSTOMERS];
   int customerCount;
   MenuItem menu[MAX_MENU_ITEMS];
   int menuCount;

public:
   Restaurant();
   void addCustomer();
   void displayCustomers();
   void displayMenu();
   void orderFood();
   void displayBillingDetails();
   void modifyCustomerDetails();
   void splitBill();
   void makeReservation();
   void cancelReservation();
   void addLoyaltyPoints(int tableNo, float amount);
   void displayLoyaltyPoints(int tableNo);
};
```

Class Restaurant:

Private Members:

- customers: Array of Customer objects.
- customerCount: Tracks the number of customers.
- menu: Array of MenuItem objects.

- menuCount: Tracks the number of menu items.

Public Members:
- Constructor: Initializes the restaurant with default values.
- Various member functions to manage customers, menu, orders, billing, reservations, and loyalty points.

Constructor

//code

```
Restaurant::Restaurant() : customerCount(0), menuCount(0) {
    menu[menuCount++] = {1, "Idli", 30.0};
    menu[menuCount++] = {2, "Dosa", 50.0};
    menu[menuCount++] = {3, "Vada", 20.0};
    menu[menuCount++] = {4, "Uttapam", 60.0};
    menu[menuCount++] = {5, "Sambar", 25.0};
    menu[menuCount++] = {6, "Rasam", 25.0};
    menu[menuCount++] = {7, "Pongal", 40.0};
    menu[menuCount++] = {8, "Bisi Bele Bath", 70.0};
    menu[menuCount++] = {9, "Upma", 35.0};
    menu[menuCount++] = {10, "Masala Dosa", 70.0};
    menu[menuCount++] = {11, "Filter Coffee", 15.0};
    menu[menuCount++] = {12, "Spaghetti Carbonara", 120.0};
    menu[menuCount++] = {13, "Margherita Pizza", 150.0};
    menu[menuCount++] = {14, "Caesar Salad", 80.0};
    menu[menuCount++] = {15, "Grilled Chicken", 200.0};
    menu[menuCount++] = {16, "Beef Steak", 300.0};
    menu[menuCount++] = {17, "French Fries", 50.0};
    menu[menuCount++] = {18, "Cheeseburger", 100.0};
    menu[menuCount++] = {19, "Tiramisu", 70.0};
    menu[menuCount++] = {20, "Apple Pie", 60.0};
    menu[menuCount++] = {21, "Pancakes", 90.0};
```

```
    menu[menuCount++] = {22, "Milkshake", 40.0};
    menu[menuCount++] = {23, "Chicken Nuggets", 80.0};
    menu[menuCount++] = {24, "Fish and Chips", 140.0};
}
```

- The constructor initializes the restaurant with a predefined list of menu items and sets the initial customer and menu counts to 0.

Member Functions

addCustomer

//code

```
void Restaurant::addCustomer() {
    if (customerCount < MAX_CUSTOMERS) {
        Customer& c = customers[customerCount];
        cout << "Enter table number: ";
        cin >> c.tableNo;
        cin.ignore();

        cout << "Enter name: ";
        getline(cin, c.name);

        cout << "Enter phone number: ";
        cin >> c.phone;

        c.billAmount = 0.0;
        c.orderCount = 0;
        c.reserved = false;
        c.loyaltyPoints = 0;
```

```
        ++customerCount;
        cout << "Customer added successfully!\n";
    } else {
        cout << "Customer limit reached!\n";
    }
}
```

- Prompts the user to enter customer details and adds a new customer to the array if the maximum limit is not reached.

displayCustomers

//code

```
void Restaurant::displayCustomers() {
    if (customerCount == 0) {
        cout << "No customers to display.\n";
        return;
    }

    cout << left << setw(10) << "Table No" << setw(20) << "Name" << setw(15) << "Phone" << setw(10) << "Bill" << setw(10) << "Reserved" << setw(10) << "Loyalty Points" << "\n";
    for (int i = 0; i < customerCount; ++i) {
        const Customer& c = customers[i];
        cout << left << setw(10) << c.tableNo << setw(20) << c.name << setw(15) << c.phone << setw(10) << c.billAmount << setw(10) << (c.reserved ? "Yes" : "No") << setw(10) << c.loyaltyPoints << "\n";
    }
}
```

- Displays the list of customers with their details formatted in a table.

displayMenu

//code

```
void Restaurant::displayMenu() {
    if (menuCount == 0) {
        cout << "No menu items to display.\n";
        return;
    }

    cout << left << setw(5) << "ID" << setw(25) << "Name" << setw(10)
<< "Price" << "\n";
    for (int i = 0; i < menuCount; ++i) {
        const MenuItem& item = menu[i];
        cout << left << setw(5) << item.id << setw(25) << item.name <<
setw(10) << item.price << "\n";
    }
}
```

- Displays the list of menu items with their details formatted in a table.

orderFood

//code

```
void Restaurant::orderFood() {
    int tableNo;
    cout << "Enter table number to order food: ";
    cin >> tableNo;

    for (int i = 0; i < customerCount; ++i) {
        if (customers[i].tableNo == tableNo) {
```

```
        Customer& c = customers[i];

        cout << "\nFood Menu:\n";
        displayMenu();

        while (true) {
            int menuId;
            cout << "Enter menu item ID (0 to stop ordering): ";
            cin >> menuId;

            if (menuId == 0) break;

            bool itemFound = false;
            for (int j = 0; j < menuCount; ++j) {
                if (menu[j].id == menuId) {
                    if (c.orderCount < MAX_ORDERS) {
                        c.orders[c.orderCount] = menu[j].name;
                        c.orderPrices[c.orderCount] = menu[j].price;
                        c.billAmount += menu[j].price;
                        ++c.orderCount;
                        addLoyaltyPoints(tableNo, menu[j].price);
                        cout << "Food ordered successfully!\n";
                        itemFound = true;
                        break;
                    } else {
                        cout << "Maximum orders reached

!\n";
                        return;
                    }
                }
            }
            if (!itemFound) {
```

```
                cout << "Invalid menu item ID.\n";
            }
        }
        return;
    }
  }
  cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Allows a customer to order food by specifying the table number and menu item IDs. The orders are added to the customer's record, and the bill amount is updated.

displayBillingDetails

//code

```
void Restaurant::displayBillingDetails() {
  int tableNo;
  cout << "Enter table number to display bill: ";
  cin >> tableNo;

  for (int i = 0; i < customerCount; ++i) {
    if (customers[i].tableNo == tableNo) {
      const Customer& c = customers[i];
      cout << "Bill Details for " << c.name << " (Table " << c.tableNo
<< "):\n";
      for (int j = 0; j < c.orderCount; ++j) {
        cout << left << setw(25) << c.orders[j] << setw(10) <<
c.orderPrices[j] << "\n";
      }
```

```
            cout << "Total Bill: $" << c.billAmount << "\n";
            return;
        }
    }
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Displays the billing details for a customer based on their table number.

modifyCustomerDetails

//code

```
void Restaurant::modifyCustomerDetails() {
    int tableNo;
    cout << "Enter table number to modify details: ";
    cin >> tableNo;

    for (int i = 0; i < customerCount; ++i) {
        if (customers[i].tableNo == tableNo) {
            Customer& c = customers[i];
            cout << "Enter new name: ";
            cin.ignore();
            getline(cin, c.name);

            cout << "Enter new phone number: ";
            cin >> c.phone;

            cout << "Customer details updated successfully!\n";
            return;
```

```
        }
    }
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Allows modification of customer details such as name and phone number.

splitBill

//code

```
void Restaurant::splitBill() {
    int tableNo;
    cout << "Enter table number to split bill: ";
    cin >> tableNo;

    for (int i = 0; i < customerCount; ++i) {
        if (customers[i].tableNo == tableNo) {
            Customer& c = customers[i];
            int numPayers;
            cout << "Enter the number of people splitting the bill: ";
            cin >> numPayers;

            if (numPayers <= 0) {
                cout << "Invalid number of payers.\n";
                return;
            }

            float splitAmount = c.billAmount / numPayers;
```

```
            cout << "Bill split successfully. Each person owes: $" <<
splitAmount << "\n";
            return;
        }
    }
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Splits the bill among a specified number of people.

makeReservation

//code

```
void Restaurant::makeReservation() {
    int tableNo;
    cout << "Enter table number for reservation: ";
    cin >> tableNo;

    for (int i = 0; i < customerCount; ++i) {
        if (customers[i].tableNo == tableNo) {
            if (customers[i].reserved) {
                cout << "Table already reserved.\n";
                return;
            }

            customers[i].reserved = true;
            cout << "Table " << tableNo << " reserved successfully!\n";
            return;
        }
    }
```

```
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Reserves a table for a customer.

cancelReservation

//code

```
void Restaurant::cancelReservation() {
    int tableNo;
    cout << "Enter table number to cancel reservation: ";
    cin >> tableNo;

    for (int i = 0; i < customerCount; ++i) {
        if (customers[i].tableNo == tableNo) {
            if (!customers[i].reserved) {
                cout << "Table is not reserved.\n";
                return;
            }

            customers[i].reserved = false;
            cout << "Reservation for table " << tableNo << " canceled
successfully!\n";
            return;
        }
    }
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Cancels a reservation for a table.

addLoyaltyPoints

//code

```
void Restaurant::addLoyaltyPoints(int tableNo, float amount) {
   for (int i = 0; i < customerCount; ++i) {
      if (customers[i].tableNo == tableNo) {
         customers[i].loyaltyPoints += static_cast<int>(amount / 10); //
1 point for every $10
         cout << "Loyalty points added successfully!\n";
         return;
      }
   }
   cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Adds loyalty points to a customer's account based on the amount spent.

displayLoyaltyPoints

//code

```
void Restaurant::displayLoyaltyPoints(int tableNo) {
   for (int i = 0; i < customerCount; ++i) {
      if (customers[i].tableNo == tableNo) {
         cout << "Customer " << customers[i].name << " has " <<
customers[i].loyaltyPoints << " loyalty points.\n";
         return;
      }
   }
```

```
    cout << "Customer with table number " << tableNo << " not
found.\n";
}
```

- Displays the loyalty points of a customer based on their table number.

Main Menu Display Function

//code

```
void displayMainMenu() {
    cout << "\nRestaurant Management System\n";
    cout << "1. Add Customer\n";
    cout << "2. Display Customers\n";
    cout << "3. Order Food\n";
    cout << "4. Display Bill\n";
    cout << "5. Make Reservation\n";
    cout << "6. Cancel Reservation\n";
    cout << "7. Modify Customer Details\n";
    cout << "8. Split Bill\n";
    cout << "9. Display Loyalty Points\n"; // New option
    cout << "10. Exit\n";
    cout << "Enter your choice: ";
}
```

- Displays the main menu options for the user to choose from.

Main Function

//code

```
int main() {
    Restaurant restaurant;
    int choice;

    while (true) {
        displayMainMenu();
        cin >> choice;

        switch (choice) {
        case 1:
            restaurant.addCustomer();
            break;
        case 2:
            restaurant.displayCustomers();
            break;
        case 3:
            restaurant.orderFood();
            break;
        case 4:
            restaurant.displayBillingDetails();
            break;
        case 5:
            restaurant.makeReservation();
            break;
        case 6:
            restaurant.cancelReservation();
            break;
        case 7:
            restaurant.modifyCustomerDetails();
```

```
            break;
        case 8:
            restaurant.splitBill();
            break;
        case 9:
            int tableNo;
            cout << "Enter table number to display loyalty points: ";
            cin >> tableNo;
            restaurant.displayLoyaltyPoints(tableNo);
            break;
        case 10:
            return 0;
        default:
            cout << "Invalid choice. Please try again.\n";
        }
    }

    return 0;
}
```

- The main function creates an instance of the Restaurant class and runs an infinite loop to display the main menu and handle user input. Based on the user's choice, the appropriate member function of the Restaurant class is called. The loop continues until the user chooses to exit by entering the choice 10, which causes the program to return 0, signaling normal termination.