CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF COMPUTING

PROJECT REPORT
ON

Restaurant Menu Ordering System

Program Name: BCA
Subject Name/Code: 23CAP-308

Submitted by:

Name: Karan Kulraj Singh

UID:23BCA10783

Section: 23BCA 2A

Submitted to:

Name: Rajat Kapoor

Designation: Assistant Professor

CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Project Report: Restaurant Menu Ordering System

Title: Restaurant Menu Ordering System Arrays, Loops, Conditional Statements

2. Aim of the Project

The aim of this project is to develop a console-based Restaurant Menu Ordering System that allows customers to browse food categories, select menu items, manage quantities, view orders in real-time, and generate detailed bills efficiently using advanced C++ programming concepts such as Object-Oriented Programming, STL containers (unordered_map, vector, pairs), control structures, and modular functions.

This project demonstrates the implementation of data structures and object-oriented principles to solve real-world restaurant management problems using procedural and object-oriented programming paradigms.

3. Objective

- To create an interactive console application for restaurant food ordering and management
- To implement category-wise menu browsing (Vegetarian/Non-Vegetarian) with proper segregation
- To manage cart operations dynamically using STL containers
- To provide real-time order preview and modification capabilities
- To implement quantity management and item removal functionality
- To calculate total bill automatically including tax calculations
- To design a robust menu-driven user interface with proper navigation
- To implement comprehensive input validation and error handling
- To apply C++ programming fundamentals in a practical project scenario

4. Tools and Technologies Used

Category	Details
Programming Language	C++
Compiler	GCC (GNU Compiler Collection) / MinGW
IDE	Visual Studio Code / Code::Blocks / Dev C++ / Turbo C++
Operating System	Windows / Linux / macOS
Core Concepts Used	Object-Oriented Programming, STL Containers (vector, unordered_map, pair), Functions, Control Structures, Input/Output Operations, Error Handling

5. System Requirements

Hardware Requirements

- Processor: Intel Pentium IV or higher
- RAM: 512 MB or more
- Hard Disk: 100 MB of free space
- Display: Standard monitor (800x600 resolution or higher)

So ware Requirements

- Operating System: Windows 7/8/10/11, Linux, or macOS
- Compiler: GCC, Turbo C, or any C compiler
- IDE/Text Editor: Visual Studio Code / Code::Blocks / Dev C++ / Turbo C++

6. Algorithm / Logic

Main Algorithm

1. Initialize restaurant menu data using structures and vectors

2. Display main menu with options:

- View Veg Menu
- View Non-Veg Menu
- View Current Order
- Generate Bill & Exit
- Exit Without Ordering

3. WHILE user does not choose Exit

Accept user's choice with validation

SWITCH (choice):

- Case 1: Display vegetarian menu and process orders
- Case 2: Display non-vegetarian menu and process orders
- Case 3: Display current order summary
- Case 4: Generate final bill and exit
- Case 5: Exit without ordering (with confirmation)

4. For category menus:

- Display available items with codes and prices
- Provide options: Add item, View order, Remove item, Back to main
- Validate all inputs and handle errors gracefully

5. Generate comprehensive bill with:

- Itemized list with quantities
- Subtotal calculation
- Tax computation (10%)
- Final total amount

6. Display farewell message

END

Main Menu
- View Veg Menu
- View Non-Veg Menu
- View Current Order
- Generate Bill & Exit
- Exit Without Ordering

Category Menu
- Add Item to Order
  - Validate Item Code
  - Get Quantity (1-9)
  - Update Order Cart
- View Current Order
  - Display All Items
  - Show Quantities
  - Calculate Subtotal
- Remove Item from Order
  - Select Item
  - Specify Quantity
  - Update Cart
- Back to Main Menu

Core Function Descriptions

1. initializeMenu(): Populates the menu with predefined items
2. getValidatedInput(): Ensures user input is within valid ranges
3. displayMenuByCategory(): Shows items filtered by category
4. addItemToOrder(): Adds/updates items in the order cart
5. removeItemFromOrder(): Removes or reduces item quantities
6. generateFinalBill(): Creates detailed bill with tax calculation
7. startOrdering(): Main program flow controller

7. Code Overview

```
#include <iostream>
#include <unordered_map>
#include <string>
#include <vector>
#include <iomanip>
#include <limits>

using namespace std;

class RestaurantOrderingSystem {
private:
    struct MenuItem {
        string name;
        int price;
        string category;
        string code;
    };

    vector<MenuItem> menu;
    unordered_map<string, pair<int, int>> orders; // item_name -> {price, quantity}

    void initializeMenu() {
        // Veg Items
        menu.push_back({"Matar Paneer", 150, "veg", "V1"});
        menu.push_back({"Aaloo Paratha", 25, "veg", "V2"});
        menu.push_back({"Butter Roti", 15, "veg", "V3"});
        menu.push_back({"Pulao", 120, "veg", "V4"});
        menu.push_back({"Paneer Roll", 100, "veg", "V5"});

        // Non-Veg Items
        menu.push_back({"Butter Chicken", 250, "nonveg", "NV1"});
        menu.push_back({"Boiled Egg", 25, "nonveg", "NV2"});
        menu.push_back({"Omelette(2 eggs)", 80, "nonveg", "NV3"});
        menu.push_back({"Chicken Briyani", 180, "nonveg", "NV4"});
        menu.push_back({"Egg Roll(2 eggs)", 100, "nonveg", "NV5"});
```

```
}


void clearInputBuffer() {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

int getValidatedInput(const string& prompt, int minVal, int maxVal) {
    int input;
    while (true) {
        cout << prompt;
        cin >> input;
        if (cin.fail() || input < minVal || input > maxVal) {
            cout << "Invalid input! Please enter a number between "
                 << minVal << " and " << maxVal << "." << endl;
            clearInputBuffer();
        } else {
            clearInputBuffer();
            return input;
        }
    }
}

void displayMenuByCategory(const string& category) {
    cout << "\n" << string(50, '=') << endl;
    cout << "         " << (category == "veg" ? "VEG ITEMS" : "NON-VEG ITEMS")
<< endl;
    cout << string(50, '=') << endl;
    cout << left << setw(8) << "Code" << setw(25) << "Item Name"
         << setw(15) << "Price (Rs.)" << endl;
    cout << string(50, '-') << endl;

    for (const auto& item : menu) {
        if (item.category == category) {
            cout << left << setw(8) << item.code
                 << setw(25) << item.name
                 << setw(15) << item.price << endl;
        }
    }
    cout << string(50, '=') << endl;
}
```

```
void displayCurrentOrders() {
    if (orders.empty()) {
        cout << "\nYour cart is empty!" << endl;
        return;
    }

    cout << "\n" << string(60, '=') << endl;
    cout << "                    CURRENT ORDER" << endl;
    cout << string(60, '=') << endl;
    cout << left << setw(25) << "Item" << setw(10) << "Quantity"
         << setw(15) << "Price Each" << setw(15) << "Total" << endl;
    cout << string(60, '-') << endl;

    int subtotal = 0;
    for (const auto& order : orders) {
        int itemTotal = order.second.first * order.second.second;
        cout << left << setw(25) << order.first
             << setw(10) << order.second.second
             << setw(15) << order.second.first
             << setw(15) << itemTotal << endl;
        subtotal += itemTotal;
    }

    cout << string(60, '-') << endl;
    cout << right << setw(50) << "Subtotal: Rs. " << subtotal << endl;
    cout << string(60, '=') << endl;
}

void addItemToOrder(const string& itemCode, int quantity) {
    for (const auto& item : menu) {
        if (item.code == itemCode) {
            if (orders.find(item.name) != orders.end()) {
                // Item already exists, update quantity
                orders[item.name].second += quantity;
            } else {
                // New item
                orders[item.name] = {item.price, quantity};
            }
```

```
            cout << "\n✓ Added " << quantity << " x " << item.name << " to your order."
<< endl;
            return;
        }
    }
    cout << "Item not found!" << endl;
}

void removeItemFromOrder() {
    if (orders.empty()) {
        cout << "No items to remove!" << endl;
        return;
    }

    displayCurrentOrders();
    cout << "\nEnter the item name to remove: ";
    string itemName;
    getline(cin, itemName);

    if (orders.find(itemName) != orders.end()) {
        cout << "Enter quantity to remove (0 to remove all): ";
        int quantity;
        cin >> quantity;
        clearInputBuffer();

        if (quantity <= 0 || quantity >= orders[itemName].second) {
            orders.erase(itemName);
            cout << "✓ Removed " << itemName << " from your order." << endl;
        } else {
            orders[itemName].second -= quantity;
            cout << "✓ Reduced quantity of " << itemName << " by " << quantity << "."
<< endl;
        }
    } else {
        cout << "Item not found in your order!" << endl;
    }
}

void processCategoryOrder(const string& category) {
    while (true) {
        displayMenuByCategory(category);
```

```
        cout << "\nOptions:" << endl;
        cout << "1. Add item to order" << endl;
        cout << "2. View current order" << endl;
        cout << "3. Remove item from order" << endl;
        cout << "4. Back to main menu" << endl;

        int choice = getValidatedInput("Enter your choice (1-4): ", 1, 4);

        if (choice == 1) {
            cout << "Enter item code: ";
            string itemCode;
            cin >> itemCode;
            clearInputBuffer();

            // Validate item code
            bool validCode = false;
            for (const auto& item : menu) {
                if (item.code == itemCode && item.category == category) {
                    validCode = true;
                    break;
                }
            }

            if (!validCode) {
                cout << "Invalid item code for this category!" << endl;
                continue;
            }

            int quantity = getValidatedInput("Enter quantity (1-9): ", 1, 9);
            addItemToOrder(itemCode, quantity);

        } else if (choice == 2) {
            displayCurrentOrders();
        } else if (choice == 3) {
            removeItemFromOrder();
        } else if (choice == 4) {
            break;
        }
    }
}
```

```
void generateFinalBill() {
    if (orders.empty()) {
        cout << "\nNo orders to generate bill!" << endl;
        return;
    }

    cout << "\n" << string(70, '=') << endl;
    cout << "                    FINAL BILL" << endl;
    cout << string(70, '=') << endl;
    cout << left << setw(25) << "Item" << setw(10) << "Quantity"
         << setw(15) << "Price Each" << setw(15) << "Total" << endl;
    cout << string(70, '-') << endl;

    int subtotal = 0;
    for (const auto& order : orders) {
        int itemTotal = order.second.first * order.second.second;
        cout << left << setw(25) << order.first
             << setw(10) << order.second.second
             << setw(15) << order.second.first
             << setw(15) << itemTotal << endl;
        subtotal += itemTotal;
    }

    double tax = subtotal * 0.10; // 10% tax
    double total = subtotal + tax;

    cout << string(70, '-') << endl;
    cout << right << setw(55) << "Subtotal: Rs. " << subtotal << endl;
    cout << right << setw(55) << "Tax (10%): Rs. " << fixed << setprecision(2) << tax
<< endl;
    cout << right << setw(55) << "Total: Rs. " << fixed << setprecision(2) << total <<
endl;
    cout << string(70, '=') << endl;
  }

public:
  RestaurantOrderingSystem() {
    initializeMenu();
```

```
}


void startOrdering() {
    cout << string(60, '=') << endl;
    cout << "       WELCOME TO RESTAURANT ORDERING SYSTEM" << endl;
    cout << string(60, '=') << endl;

    while (true) {
        cout << "\nMAIN MENU:" << endl;
        cout << "1. View Veg Menu" << endl;
        cout << "2. View Non-Veg Menu" << endl;
        cout << "3. View Current Order" << endl;
        cout << "4. Generate Bill & Exit" << endl;
        cout << "5. Exit Without Ordering" << endl;

        int choice = getValidatedInput("Enter your choice (1-5): ", 1, 5);

        switch (choice) {
            case 1:
                processCategoryOrder("veg");
                break;
            case 2:
                processCategoryOrder("nonveg");
                break;
            case 3:
                displayCurrentOrders();
                break;
            case 4:
                generateFinalBill();
                cout << "\nThank you for your order! Visit again!" << endl;
                return;
            case 5:
                if (!orders.empty()) {
                    cout << "You have items in your cart. Are you sure you want to exit?
(y/n): ";
                    char confirm;
                    cin >> confirm;
                    clearInputBuffer();
                    if (confirm == 'y' || confirm == 'Y') {
```

```
                cout << "Thank you for visiting! Hope to see you again!" << endl;
                return;
            }
        } else {
            cout << "Thank you for visiting! Hope to see you again!" << endl;
            return;
        }
        break;
    }
  }
 }
};

int main() {
    RestaurantOrderingSystem restaurant;
    restaurant.startOrdering();
    return 0;
}
```

8. Screenshots/ Output

```
============================================================
       WELCOME TO RESTAURANT ORDERING SYSTEM
============================================================

MAIN MENU:
1. View Veg Menu
2. View Non-Veg Menu
3. View Current Order
4. Generate Bill & Exit
5. Exit Without Ordering
Enter your choice (1-5): 1

==================================================
         VEG ITEMS
==================================================
Code    Item Name                Price (Rs.)
--------------------------------------------------
V1      Matar Paneer             150
V2      Aaloo Paratha            25
V3      Butter Roti              15
V4      Pulao                    120
V5      Paneer Roll              100
==================================================

Options:
1. Add item to order
2. View current order
3. Remove item from order
4. Back to main menu
Enter your choice (1-4):
```

```
Options:
1. Add item to order
2. View current order
3. Remove item from order
4. Back to main menu
Enter your choice (1-4): 2

============================================================
                    CURRENT ORDER
============================================================
Item                    Quantity  Price Each     Total
------------------------------------------------------------
Aaloo Paratha           3         25             75
------------------------------------------------------------
                                    Subtotal: Rs. 75
============================================================
```

```
MAIN MENU:
1. View Veg Menu
2. View Non-Veg Menu
3. View Current Order
4. Generate Bill & Exit
5. Exit Without Ordering
Enter your choice (1-5): 2

==================================================
         NON-VEG ITEMS
==================================================
Code    Item Name               Price (Rs.)
--------------------------------------------------
NV1     Butter Chicken          250
NV2     Boiled Egg              25
NV3     Omelette(2 eggs)        80
NV4     Chicken Briyani         180
NV5     Egg Roll(2 eggs)        100
==================================================

Options:
1. Add item to order
2. View current order
3. Remove item from order
4. Back to main menu
Enter your choice (1-4):
```

```
MAIN MENU:
1. View Veg Menu
2. View Non-Veg Menu
3. View Current Order
4. Generate Bill & Exit
5. Exit Without Ordering
Enter your choice (1-5): 4

========================================================================
                      FINAL BILL
========================================================================
Item                    Quantity  Price Each     Total
------------------------------------------------------------------------
Butter Chicken          4         250            1000
Aaloo Paratha           3         25             75
------------------------------------------------------------------------
                                        Subtotal: Rs. 1075
                                      Tax (10%): Rs. 107.50
                                          Total: Rs. 1182.50
========================================================================

Thank you for your order! Visit again!
```

9. Conclusion

The Restaurant Menu Ordering System was successfully designed and implemented using the C++ programming language, demonstrating the practical application of object-oriented programming principles, STL containers, and advanced control structures. The system efficiently manages menu listings, order processing, cart management, and bill generation through a professional, menu-driven console interface.

This project significantly improves upon traditional restaurant ordering by providing a digital solution that reduces errors, enhances customer experience, and streamlines the ordering process. The system ensures data consistency, input validation, and provides users with a smooth and interactive ordering experience.

By utilizing STL containers for data management, object-oriented design for code organization, and comprehensive input validation for reliability, the program achieves both robustness and user-friendliness. The system can be easily extended with additional features such as:

- Database integration for menu management
- Multiple payment method support
- Order history tracking
- Table reservation system
- Graphical user interface
- Inventory management integration
- Customer loyalty programs

Overall, this project serves as an excellent foundation for understanding modern point-of-sale system design and demonstrates the importance of modular programming, user experience design, and efficient algorithm implementation in software development.

10. Learning Outcomes

Through the completion of this project, the following key skills were gained:

Area	Learning Outcome
Object-Oriented Programming	Designed and implemented class-based architecture with proper encapsulation
STL Containers	Effectively used vectors, unordered_maps, and pairs for data management
Input/Output Operations	Implemented professional console I/O with formatting and validation
Error Handling	Developed comprehensive input validation and error recovery mechanisms
Control Structures	Applied complex conditional logic and loop structures for program flow
Functions	Created modular, reusable functions for better code organization
Problem Solving	Translated real-world restaurant operations into logical program design
User Interface Design	Developed intuitive, user-friendly console interface
Software Engineering	Understood complete software development lifecycle from design to implementation

11. References

Books

1. Stroustrup, B. (2013). *The C++ Programming Language (4th Ed.)*. Addison-Wesley Professional.
2. Lippman, S. B., Lajoie, J., & Moo, B. E. (2012). *C++ Primer (5th Ed.)*. Addison-Wesley Professional.
3. Schildt, H. (2017). *C++: The Complete Reference (4th Ed.)*. McGraw-Hill Education.

Online Resources

- GeeksforGeeks – C++ Programming Language
- CPlusPlus.com – The C++ Resources Network
- Stack Overflow – C++ Community Discussions
- C++ Reference Documentation

Documentation

- C++ Standard Library Reference – cppreference.com
- GNU C++ Library Documentation
- C++ Core Guidelines

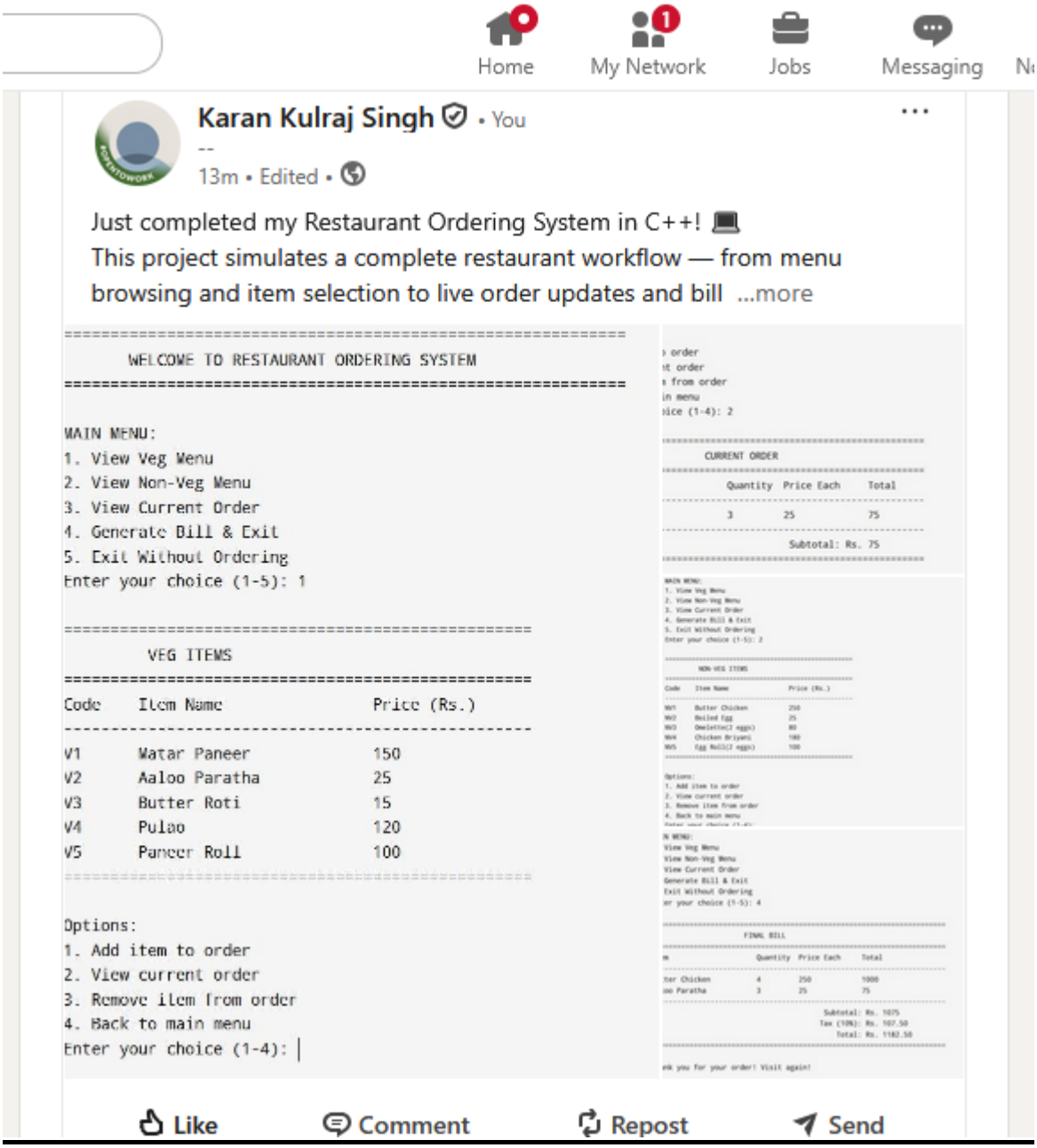
Video Tutorials

- *TheCherno – C++ Series*
- *freeCodeCamp – C++ Programming Course*
- *CodeBeauty – C++ Programming Tutorials*

Research Papers

- "Design Patterns in Object-Oriented Programming" – Gamma et al.
- "Efficient Data Structures for Commercial Applications" – ACM Journal
- "User Interface Design Principles for Console Applications" – HCI Research Papers

Linkedin: https://www.linkedin.com/posts/karan-kulraj-singh-34169b293_cplusplus-programming-projects-activity-7391379908234756096-7-I_?utm_source=share&utm_medium=member_desktop&rcm=ACoAAEcdXSEBPRyMtMdp47NxcgYol2EXXxFi6Zc



Github: https://github.com/karndhiman/Restaurant-Menu-Ordering-System