

# Restaurant Management System

## Using C++ OOP Principles

Muhammad Ahabb Sheraz  
2021327

Faculty of Computer Engineering  
Ghulam Ishaq Khan Institute  
[u2021327@giki.edu.pk](mailto:u2021327@giki.edu.pk)

Behram Khan  
2021127

Faculty of Computer Engineering  
Ghulam Ishaq Khan Institute  
[u2021127@giki.edu.pk](mailto:u2021127@giki.edu.pk)

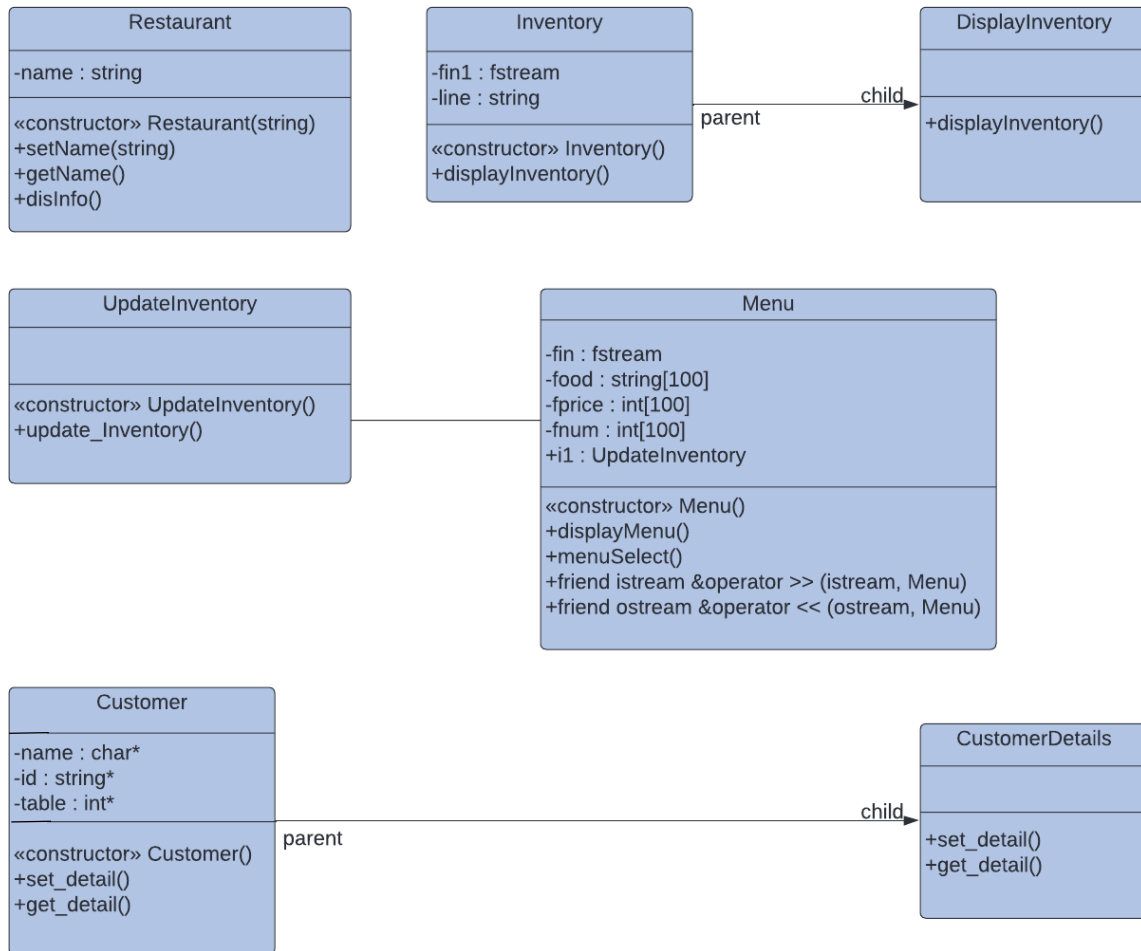
***Problem statement—*** The aim of our project was to develop a reliable restaurant management system on C++ using Object Oriented Programming principles. The system will display a menu to the customer. The system will also keep track of the inventory of the food items. At the end, bill will be displayed.

### INTRODUCTION:

Our system can be used for keeping record/track of the bills and the inventory. It can also display food menus and let users select the menu items that they want to order. The system using file handling and various OOP principles such as inheritance, operator overloading, friend classes, and polymorphism to achieve this purpose.

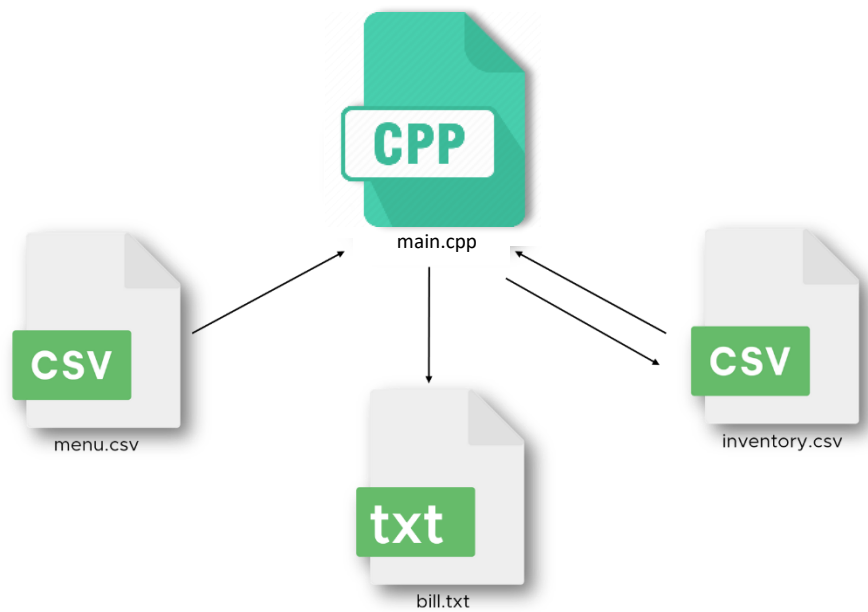
The system displays a menu to the customer. The system also keeps track of the inventory of the food items and bills.

## CLASS DIAGRAM (UML):



Unified Modeling Language Diagram.

## FUNCTIONAL REQUIREMENTS:



File hierarchy in our system.

The program consists of one .cpp file and two header files:

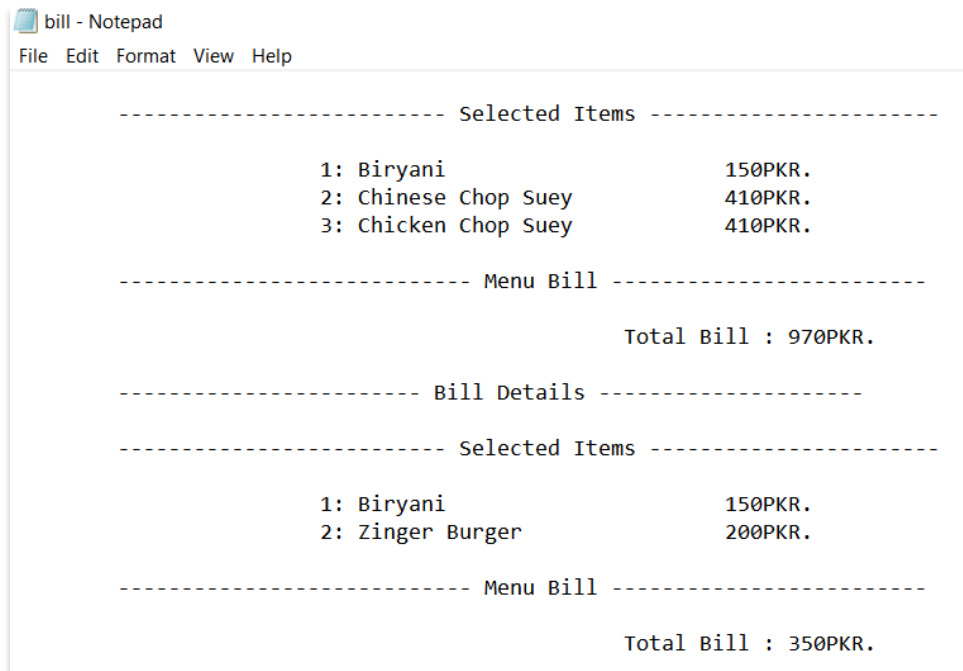
- main.cpp
- main.h (header file containing the Menu and Customer classes)
- inventory.h (header file containing the Inventory classes)

The program needs or utilizes 3 files to perform its functions:

**menu.csv:** Menu items and their cost are kept in this file. Records are stored in the following format: (S/no., item name, price).

```
*menu - Notepad
File Edit Format View Help
1,Biryani,150.00
2,Zinger Burger,200.00
3,Chinese Chop Suey,410.00
4,Chicken Chop Suey,410.00
5,American Chop Suey,440.00|
```

**bill.txt:** Generated bills are stored in this file.



```
bill - Notepad
File Edit Format View Help

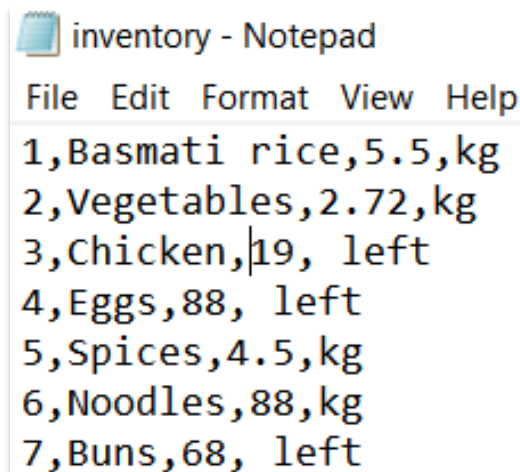
----- Selected Items -----
1: Biryani 150PKR.
2: Chinese Chop Suey 410PKR.
3: Chicken Chop Suey 410PKR.

----- Menu Bill -----
Total Bill : 970PKR.

----- Bill Details -----
----- Selected Items -----
1: Biryani 150PKR.
2: Zinger Burger 200PKR.

----- Menu Bill -----
Total Bill : 350PKR.
```

**inventory.csv:** Keeps track of the inventory of food items. Records are stored in the following format: (S/no., item name, quantity, unit).



```
inventory - Notepad
File Edit Format View Help
1,Basmati rice,5.5,kg
2,Vegetables,2.72,kg
3,Chicken,19, left
4,Eggs,88, left
5,Spices,4.5,kg
6,Noodles,88,kg
7,Buns,68, left
```

## Interfacing:

- 1) A menu will be displayed to the users. Users will then type the number of items that they want to order from the menu. After that, they can type the food item numbers to make a selection. A bill will then be generated. This functionality has been implemented in the Menu class of the main header file.
- 2) As the user selects the menu items, the inventory will be updated through UpdateInventory class implemented in the inventory header file.
- 3) After the bill is generated, users will be given the option to continue. If a user types 'y' or 'Y' then step 1 and 2 will be repeated again. If the user does not want to continue then they will be shown the inventory.

```
D:\C_With_Source_Code\restaurant-management-system\RMS\RMS.exe
*****
Bon Appetit
*****

*****
Menu Items
*****
1. Biryani          ...PKR150.00
2. Zinger Burger    ...PKR200.00
3. Chinese Chop Suey ...PKR410.00
4. Chicken Chop Suey ...PKR410.00
5. American Chop Suey ...PKR440.00

Enter the no. of items that you would love to select: 3
Enter item no.: 1
Enter item no.: 3
Enter item no.: 4

----- Bill Details -----
----- Selected Items -----

1: Biryani          150PKR.
2: Chinese Chop Suey 410PKR.
3: Chicken Chop Suey 410PKR.

----- Menu Bill -----

Total Bill : 970PKR.

Would you like to continue. (Y/N): n

*****
Inventory
*****
1. Basmati rice: 5kg
2. Vegetables: 1.72kg
3. Chicken: 18kg
4. Eggs: 86 left
5. Spices: 4kg
6. Noodles: 86kg
7. Buns: 68 left

-----
Process exited after 16.61 seconds with return value 0
Press any key to continue . . .
```

Program in action.

## THE CODE:

main.h:

```
1. #ifndef MAIN_H
2. #define MAIN_H
3.
4. #include <iostream>
5. #include <fstream>
6. #include <iomanip>
7. #include <string>
8. #include <sstream>
9. #include <vector>
10. #include <bits/stdc++.h>
11. #include "inventory.h"
12.
13.
14. using namespace std;
15.
16. class Restaurant{
17.     private:
18.         string name;
19.     public:
20.         Restaurant(string x){
21.             name = x;
22.         }
23.         void setName(string y){
24.             name = y;
25.         }
26.         string getName() const{
27.             return name;
28.         }
29.         void disInfo() const{
30.             cout << "\t ***** \t\n"
31.                 << "\t\t" << name << " \t\n"
32.                 << "\t ***** \t\n";
33.         }
34. };
35.
36. class Menu{
37.     private:
38.         fstream fin, fout;
39.         string food[100];
40.         int fprice[100];
41.         int fnum[100];
42.     public:
43.         friend class UpdateInventory; //gives class UpdateInventory access to private members
44.         of Menu class
45.         UpdateInventory i1; //UpdateInventory class object to update inventory in MenuSelect
46.         function
47.         Menu(){
48.             fin.open("menu.csv", ios::in);
49.             fout.open("bill.txt", ios::out | ios::app);
50.         }
51.         void displayMenu(){
52.             vector<string> row;
53.             string line, word;
54.             int i = 0;
```

```

54.         cout << "\t ***** \t\n"
55.         << "\t  Menu Items \t\n"
56.         << "\t ***** \t\n";
57.         while(!fin.eof()){
58.             row.clear();
59.
60.             // read an entire row and
61.             // store it in a string variable 'line'
62.             getline(fin, line);
63.
64.             // used for breaking words
65.             stringstream s(line);
66.             while (getline(s, word, ',')) {
67.                 // add all the column data
68.                 // of a row to a vector
69.                 row.push_back(word);
70.             }
71.             cout << "\t " << row[0] << ". " << row[1] << "\t" << " ...PKR" << row[2] <<
endl;
72.
73.             //storing file data in arrays for use within class
74.             food[i] = row[1];
75.             fprice[i] = stoi(row[2]);
76.             fnum[i] = stoi(row[0]);
77.             i++;
78.         }
79.         cout << endl;
80.     }
81.     void menuSelect(){
82.         int selectitems, rollfood, bill = 0;
83.         cout << "\nEnter the no. of items that you would love to select: ";
84.         cin >> selectitems;
85.         string line, word;
86.         int sfood[selectitems];
87.         //unique food id is typed while updating the inventory in the loop section
88.         for(int i = 0; i < selectitems; i++){
89.             cout << "Enter item no.: ";
90.             cin >> sfood[i];
91.             i1.update_Inventory(sfood[i]);
92.         }
93.
94.
95.         cout<<"\n\t----- Bill Details ----- \n\n";
96.         fout << "\n\t----- Bill Details ----- \n\n";
97.
98.         //cout<<"\t----- Taxt total is 400.99 $ ----- \n\n";
99.         //outfile<<"\t----- Taxt total is 400.99 $ -----
\n\n";
100.
101.         cout<<"\t----- Selected Items -----
\n\n";
102.         fout << "\t----- Selected Items -----
\n\n";
103.
104.         for(int i = 0; i < selectitems; i++){
105.             cout << endl;
106.             cout << "\t\t\t" << i + 1 << ": " << food[sfood[i] - 1] << "\t\t" <<
fprice[sfood[i] - 1] << "PKR." << endl;
107.             fout << "\t\t\t" << i + 1 << ": " << food[sfood[i] - 1] << "\t\t" <<
fprice[sfood[i] - 1] << "PKR." << endl;
108.             bill += fprice[sfood[i] - 1];

```

```

109.         }
110.
111.
112.         cout<<"\n\t----- Menu Bill -----
\n\n";
113.         fout <<"\n\t----- Menu Bill -----
-\n\n";
114.
115.         cout<<"\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
116.         fout<<"\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
117.     }
118.
119.     //operator overloading function definition
120.     friend istream &operator >> (istream &i, Menu &m);
121.     friend ostream &operator << (ostream &o, Menu &m);
122.
123.     ~Menu(){
124.         fout.close();
125.         fin.close();
126.     }
127. };
128.
129.     istream &operator >> (istream &i, Menu &m){
130.         int selectitems, rollfood, bill = 0;
131.         cout << "\nEnter the no. of items that you would love to select: ";
132.         cin >> selectitems;
133.         string line, word;
134.         int sfood[selectitems];
135.         for(int i = 0; i < selectitems; i++){
136.             cout << "Enter item no.: ";
137.             cin >> sfood[i];
138.             m.i1.update_Inventory(sfood[i]);
139.         }
140.
141.
142.         cout<<"\n\t----- Bill Details ----- \n\n";
143.         m.fout <<"\n\t----- Bill Details ----- \n\n";
144.
145.         //cout<<"\t----- Taxt total is 400.99 $ ----- \n\n";
146.         //outfile<<"\t----- Taxt total is 400.99 $ -----
\n\n";
147.
148.         cout<<"\t----- Selected Items ----- \n\n";
149.         m.fout <<"\t----- Selected Items ----- \n\n";
150.
151.         for(int i = 0; i < selectitems; i++){
152.             cout << "\t\t\t" << i + 1 << ": " << m.food[sfood[i] - 1] << "\t\t" <<
m.fprice[sfood[i] - 1] <<"PKR."<<endl;
153.             m.fout << "\t\t\t" << i + 1 << ": " << m.food[sfood[i] - 1] << "\t\t" <<
m.fprice[sfood[i] - 1] <<"PKR."<<endl;
154.             bill += m.fprice[sfood[i] - 1];
155.         }
156.
157.         cout<<"\n\t----- Menu Bill ----- \n\n";
158.         m.fout <<"\n\t----- Menu Bill -----
\n\n";
159.
160.         cout<<"\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
161.         m.fout<<"\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
162.     }
163.

```



```

164.     ostream &operator << (ostream &o, Menu &m){
165.         vector<string> row;
166.         string line, word;
167.         int i = 0;
168.         cout << "\\t ***** \\t\\n"
169.         << "\\t   Menu Items \\t\\n"
170.         << "\\t ***** \\t\\n";
171.         while(!m.fin.eof()){
172.             row.clear();
173.
174.             // read an entire row and
175.             // store it in a string variable 'line'
176.             getline(m.fin, line);
177.
178.             // used for breaking words
179.             stringstream s(line);
180.             while (getline(s, word, ',')) {
181.                 // add all the column data
182.                 // of a row to a vector
183.                 row.push_back(word);
184.             }
185.             cout << "\\t " << row[0] << ". " << row[1] << "\\t" << " ...PKR" << row[2] <<
endl;
186.             m.food[i] = row[1];
187.             m.fprice[i] = stoi(row[2]);
188.             m.fnum[i] = stoi(row[0]);
189.             i++;
190.         }
191.         cout << endl;
192.     }
193.
194.     class Customer{
195.     protected:
196.         char *name;
197.         string *id;
198.         int *table;
199.     public:
200.         Customer(){
201.             name=new char[100];
202.             id=new string;
203.             table = new int;
204.         }
205.         virtual void set_detail(){
206.         }
207.         virtual void get_detail() = 0;
208.     };
209.     class CustomerDetails : public Customer{
210.     public:
211.         CustomerDetails(){} //the Constructor...
212.         void set_detail(){
213.             ofstream out11;
214.             out11.open("bill.txt",ios::out | ios::app);
215.             cout<<"\\nEnter a your name : ";
216.             cin.get(name,100); // use charactor case
217.             cout<<"\\n\\nEnter a your id_card number : ";
218.             cin>>*id;
219.             cout<<"\\n\\nEnter a your table number : ";
220.             cin>>*table;
221.         }
222.         void gets_detail(){
223.             ofstream out12;

```

```

224.         out12.open("bill.txt",ios::out | ios::app);
225.         cout<<"\nName: "<<name<<endl;
226.         out12<<"\nName: "<<name<<endl;
227.         cout<<"\nID_Card: "<<*id<<endl;
228.         out12<<"ID_Card: "<<*id<<endl;
229.         cout<<"\nTable: "<<*id<<endl;
230.         out12<<"Table: "<<*table<<endl;
231.     }
232.     ~CustomerDetails(){
233.     }
234. };
235.
236. #endif

```

## inventory.h:

```

1. #include <iostream>
2. #include <fstream>
3. #include <iomanip>
4. #include <string>
5. #include <sstream>
6. #include <vector>
7. #include <bits/stdc++.h>
8.
9. #ifndef INVENTORY_H
10. #define INVENTORY_H
11.
12. using namespace std;
13.
14. class Inventory{
15.     protected:
16.         fstream fin1;
17.         string line;
18.     public:
19.         Inventory(){
20.             // Open an existing record
21.             fin1.open("inventory.csv", ios::in);
22.         }
23.         virtual void displayInventory() = 0;
24.         ~Inventory(){
25.             // Close an existing record
26.             fin1.close();
27.         }
28. };
29.
30. class DisplayInventory : public Inventory{
31.     public:
32.         void displayInventory() override {
33.             vector<string> row;
34.             string line, word;
35.             cout << "\t ***** \t\n"
36.             << "\t  Inventory \t\n"
37.             << "\t ***** \t\n";
38.
39.             while(!fin1.eof()){
40.                 row.clear();
41.

```

```

42.         getline(fin1, line);
43.
44.         // used for breaking words
45.         stringstream s(line);
46.         while (getline(s, word, ',')) {
47.             // add all the column data
48.             // of a row to a vector
49.             row.push_back(word);
50.         }
51.         cout << row[0] << " " << row[1] << ": " << row[2] << row[3] << endl;
52.     }
53. }
54. };
55.
56. class UpdateInventory{
57. public:
58.     UpdateInventory(){
59.
60.     }
61.     ~UpdateInventory(){
62.
63.     }
64.     void update_Inventory(int sfood){
65.         fstream fin1, fout1;
66.         // Open an existing record
67.         fin1.open("inventory.csv", ios::in);
68.         // Create a new file to store updated data
69.         fout1.open("inventorynew.csv", ios::out);
70.         int found = 0, i = 0;
71.         float used=0.0;
72.         char sub;
73.         int index, new_marks;
74.         string line, word, party;
75.         vector<string> row;
76.
77.         // Traverse the file
78.         while (!fin1.eof()) {
79.
80.             row.clear();
81.             //row1.clear();
82.
83.             getline(fin1, line);
84.             stringstream s(line);
85.
86.             //getline(fin, line1);
87.             //stringstream s1(line1);
88.
89.             while (getline(s, word, ',')) {
90.                 row.push_back(word);
91.             }
92.
93.             int row_size = row.size();
94.
95.             if (sfood == 1){
96.                 found = 1;
97.                 if(stoi(row[0]) == 1){
98.                     stringstream convert;
99.
100.                    // sending a number as a stream into output string and adding new votes
101.                    used = stof(row[2]) - 0.5;
102.                    convert << used;

```

```

103.
104.         // the str() converts number into string
105.         row[2] = convert.str();
106.     }else if(stoi(row[0]) == 5){
107.         stringstream convert;
108.
109.         // sending a number as a stream into output string and adding new votes
110.         used = stof(row[2]) - 0.1;
111.         convert << used;
112.
113.         // the str() converts number into string
114.         row[2] = convert.str();
115.     }else if(stoi(row[0]) == 3){
116.         stringstream convert;
117.
118.         // sending a number as a stream into output string and adding new votes
119.         used = stof(row[2]) - 1.0;
120.         convert << used;
121.
122.         // the str() converts number into string
123.         row[2] = convert.str();
124.     }
125.
126.     if (!fin1.eof()) {
127.         for (i = 0; i < row_size - 1; i++) {
128.
129.             // write the updated data
130.             // into a new file 'inventorynew.csv'
131.             // using fout
132.             fout1 << row[i] << ",";
133.         }
134.
135.         fout1 << row[row_size - 1] << endl;
136.     }
137. }
138. else if (sfood == 2){
139.     found = 1;
140.     if(stoi(row[0]) == 2){
141.         stringstream convert;
142.
143.         // sending a number as a stream into output string and adding new votes
144.         used = stof(row[2]) - 0.08;
145.         convert << used;
146.
147.         // the str() converts number into string
148.         row[2] = convert.str();
149.     }else if(stoi(row[0]) == 3){
150.         stringstream convert;
151.
152.         // sending a number as a stream into output string and adding new votes
153.         used = stof(row[2]) - 0.5;
154.         convert << used;
155.
156.         // the str() converts number into string
157.         row[2] = convert.str();
158.     }else if(stoi(row[0]) == 7){
159.         stringstream convert;
160.
161.         // sending a number as a stream into output string and adding new votes
162.         used = stof(row[2]) - 2.0;
163.         convert << used;

```

```

164.
165.         // the str() converts number into string
166.         row[2] = convert.str();
167.     }
168.     if (!fin1.eof()) {
169.         for (i = 0; i < row_size - 1; i++) {
170.
171.             // write the updated data
172.             // into a new file 'inventorynew.csv'
173.             // using fout
174.             fout1 << row[i] << ",";
175.         }
176.
177.         fout1 << row[row_size - 1] << endl;
178.     }
179. }
180. else if (sfood == 3 || sfood == 4 || sfood == 5){
181.     found = 1;
182.     if(stoi(row[0]) == 2){
183.         stringstream convert;
184.
185.         // sending a number as a stream into output string and adding new votes
186.         used = stof(row[2]) - 0.5;
187.         convert << used;
188.
189.         // the str() converts number into string
190.         row[2] = convert.str();
191.     }else if(stoi(row[0]) == 4){
192.         stringstream convert;
193.
194.         // sending a number as a stream into output string and adding new votes
195.         used = stof(row[2]) - 1.0;
196.         convert << used;
197.
198.         // the str() converts number into string
199.         row[2] = convert.str();
200.     }else if(stoi(row[0]) == 5){
201.         stringstream convert;
202.
203.         // sending a number as a stream into output string and adding new votes
204.         used = stof(row[2]) - 0.2;
205.         convert << used;
206.
207.         // the str() converts number into string
208.         row[2] = convert.str();
209.     }else if(stoi(row[0]) == 6){
210.         stringstream convert;
211.
212.         // sending a number as a stream into output string and adding new votes
213.         used = stof(row[2]) - 1.0;
214.         convert << used;
215.
216.         // the str() converts number into string
217.         row[2] = convert.str();
218.     }
219.     if (!fin1.eof()) {
220.         for (i = 0; i < row_size - 1; i++) {
221.
222.             // write the updated data
223.             // into a new file 'inventorynew.csv'
224.             // using fout

```

```

225.             fout1 << row[i] << ",";
226.         }
227.
228.         fout1 << row[row_size - 1] << endl;
229.     }
230. }
231. else {
232.     if (!fin1.eof()) {
233.         for (i = 0; i < row_size - 1; i++) {
234.
235.             // writing other existing records
236.             // into the new file using fout.
237.             fout1 << row[i] << ",";
238.         }
239.
240.         // the last column data ends with a '\n'
241.         fout1 << row[row_size - 1] << endl;
242.     }
243. }
244. if (fin1.eof())
245.     break;
246. }
247.
248. //Exception Handling
249. try{
250.     if (found == 0){
251.         throw found;
252.     }
253. }
254. catch (int found) {
255.     cerr << "record not found\n";
256. }
257.
258. fin1.close();
259. fout1.close();
260.
261. // removing the existing file
262. remove("inventory.csv");
263. // renaming the updated file with the existing file name
264. rename("inventorynew.csv", "inventory.csv");
265. }
266. };
267.
268. #endif

```

## main.cpp

```

1. #include <iostream>
2. #include "main.h"
3. #include "inventory.h"
4. #include <bits/stdc++.h>
5.
6. using namespace std;
7.
8. int main() {

```

```

9.     char op;
10.    Restaurant r("Bon Appetit"); //Restaurant Class Object
11.    Menu m1; //Menu Class Object
12.
13.    r.disInfo(); //displays Restaurant name
14.    cout << endl;
15.
16.    //m1.displayMenu();
17.    cout << m1; //loads displayMenu() function using operator overloading and displays Menu
18.
19.    do{
20.        //m1.selectMenu();
21.        cin >> m1; //loads selectMenu() function using operator overloading and let user
        select their cuisine
22.
23.        cout<<"\nWould you like to continue. (Y/N): ";
24.        cin >> op;
25.
26.    }while((op == 'y') || (op == 'Y'));
27.
28.    Inventory* i1; //Inventory Class Pointer Object
29.    DisplayInventory i1d; //DisplayInventory Derived Class Object
30.    i1 = &i1d; // case polymorphism
31.    cout << endl;
32.    i1 -> displayInventory(); // runtime use polymorphism
33.
34. }

```

## CONCUSION:

This project was not only challenging but also fun to build. It tested our knowledge of the CS112 course to the fullest. We had a great time developing this project.