CS112B Semester Project

Restaurant Management System Using C++ OOP Principles

Muhammad Ahabb Sheraz
2021327
Faculty of Computer Engineering
Ghulam Ishaq Khan Institute
u2021327@giki.edu.pk

Behram Khan
2021127
Faculty of Computer Engineering
Ghulam Ishaq Khan Institute
u2021127@giki.edu.pk

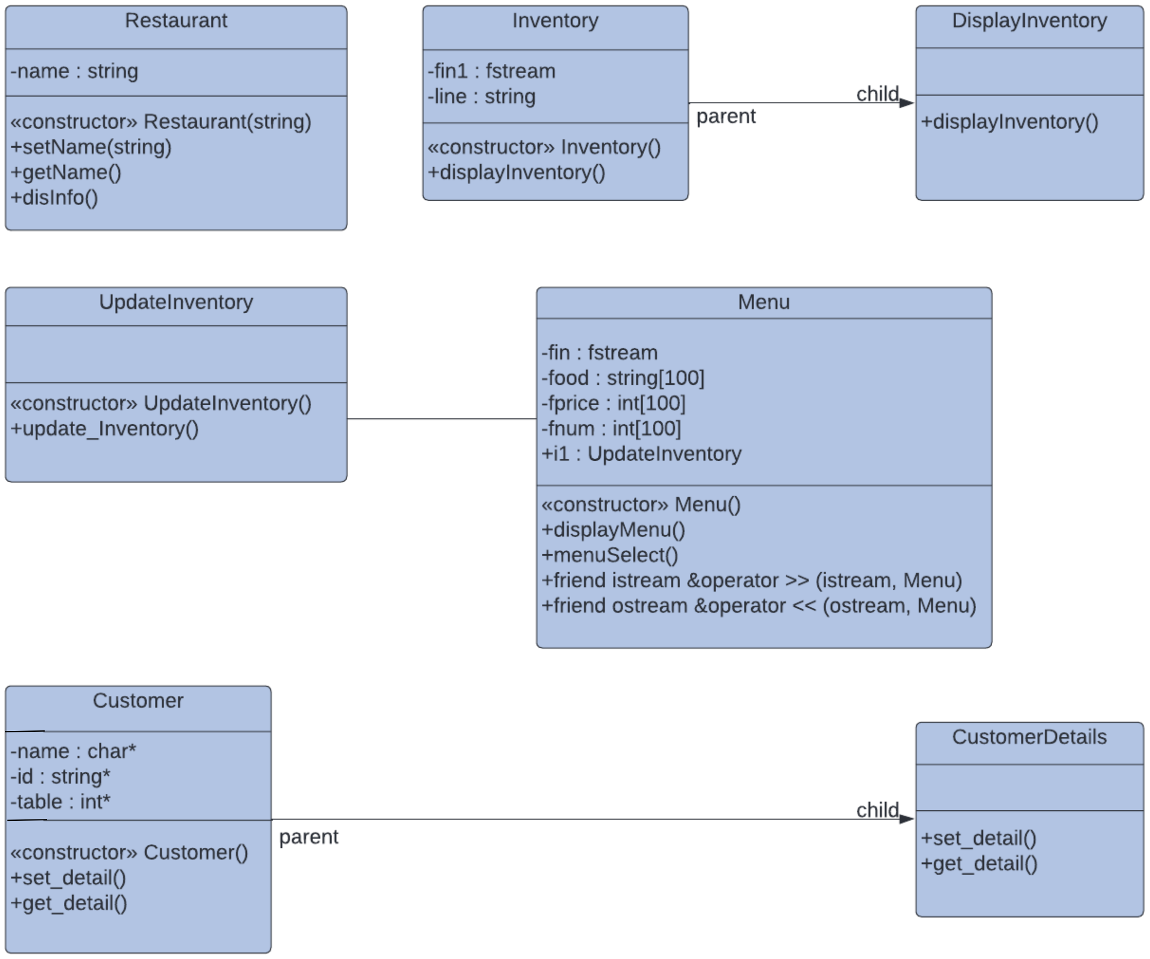
Problem statement— *The aim of our project was to develop a reliable restaurant management system on C++ using Object Oriented Programming principles. The system will display a menu to the customer. The system will also keep track of the inventory of the food items. At the end, bill will be displayed.*

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.

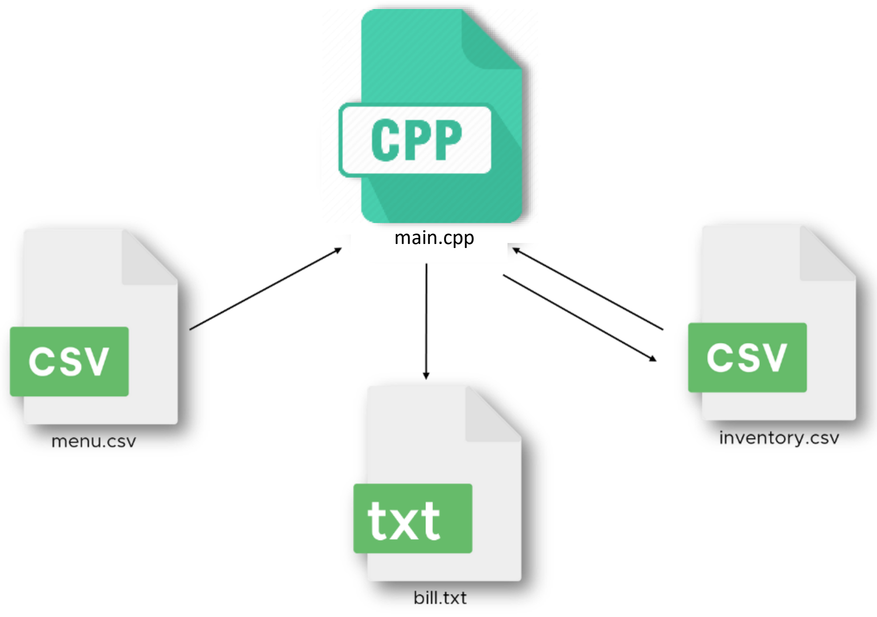
The system displays a menu to the customer. The system also keeps track of the inventory of the food items and bills.

CLASS DIAGRAM (UML):



Unified Modeling Language Diagram.

FUNCTIONAL REQUIREMENTS:



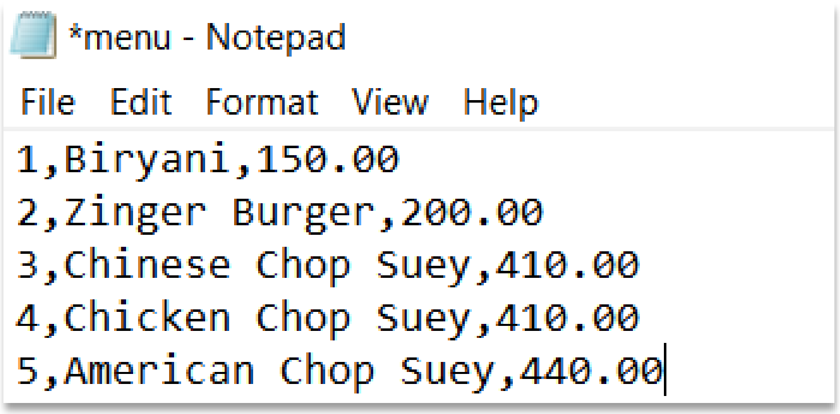
File hierarchy in our system.

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)

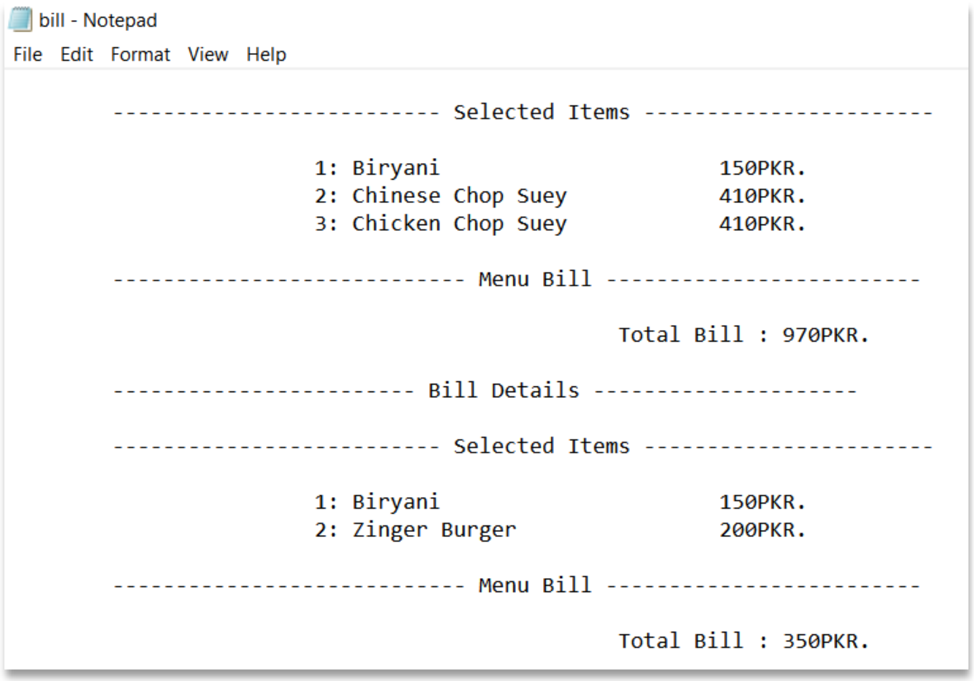
The program needs or utilizes 3 files to perform its functions:

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



```
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.

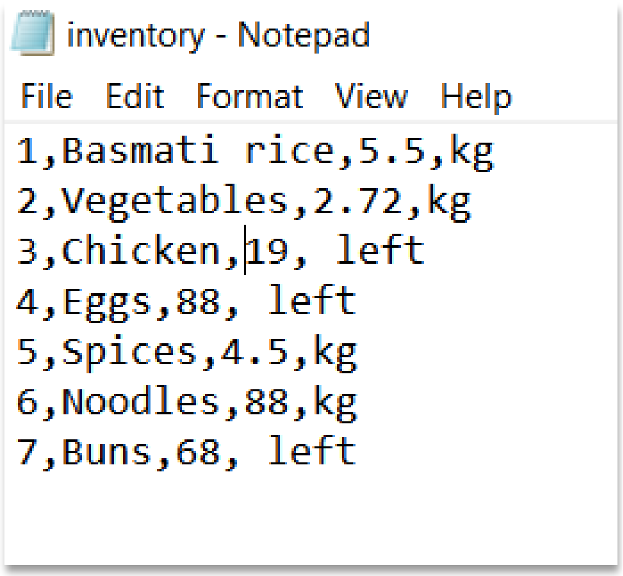


bill - Notepad

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).



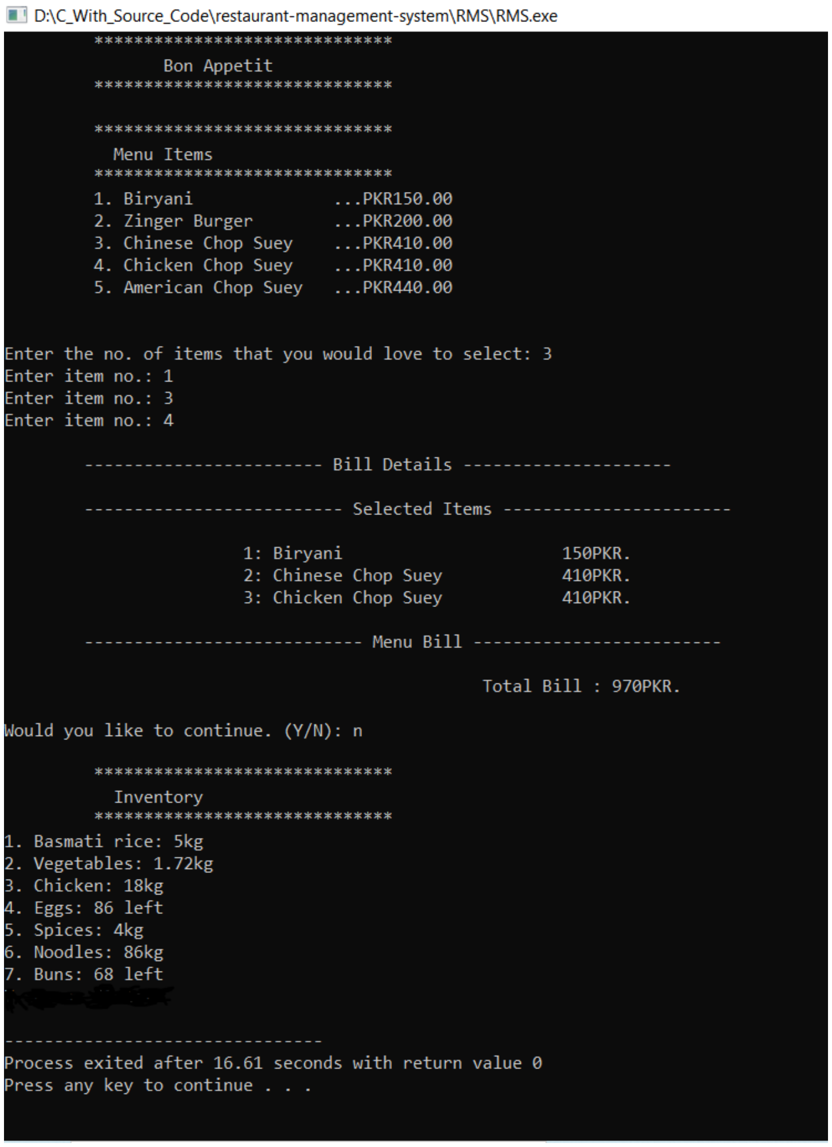
inventory - Notepad

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
[redacted]

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

Program in action.

THE CODE:

main.h:

```cpp
#ifndef MAIN_H
#define MAIN_H

#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <sstream>
#include <vector>
#include <bits/stdc++.h>
#include "inventory.h"


using namespace std;

class Restaurant{
    private:
        string name;
    public:
        Restaurant(string x){
            name = x;
        }
        void setName(string y){
            name = y;
        }
        string getName() const{
            return name;
        }
        void disInfo() const{
            cout << "\t ****************************** \t\n"
            << "\t\t" << name << " \t\n"
            << "\t ****************************** \t\n";
        }
};

class Menu{
    private:
        fstream fin, fout;
        string food[100];
        int fprice[100];
        int fnum[100];
    public:
        friend class UpdateInventory; //gives class UpdateInventory access to private members of Menu class
        UpdateInventory i1; //UpdateInventory class object to update inventory in MenuSelect function
        Menu(){
            fin.open("menu.csv", ios::in);
            fout.open("bill.txt",ios::out | ios::app);
        }

        void displayMenu(){
            vector<string> row;
            string line, word;
            int i = 0;
            cout << "\t ****************************** \t\n"
            << "\t   Menu Items \t\n"
            << "\t ****************************** \t\n";
            while(!fin.eof()){
                row.clear();

                // read an entire row and
                // store it in a string variable 'line'
                getline(fin, line);

                // used for breaking words
                stringstream s(line);
                while (getline(s, word, ',')) {
                    // add all the column data
                    // of a row to a vector
                    row.push_back(word);
                }
                cout << "\t " << row[0] << ". " << row[1] << "\t" << " ...PKR" << row[2] << endl;

                //storing file data in arrays for use within class
                food[i] = row[1];
                fprice[i] = stoi(row[2]);
                fnum[i] = stoi(row[0]);
                i++;
           }
           cout << endl;
        }
        void menuSelect(){
            int selectitems, rollfood, bill = 0;
            cout << "\nEnter the no. of items that you would love to select: ";
            cin >> selectitems;
            string line, word;
            int sfood[selectitems];
            //unique food id is typed while updating the inventory in the loop section
            for(int i = 0; i < selectitems; i++){
                cout << "Enter item no.: ";
                cin >> sfood[i];
                i1.update_Inventory(sfood[i]);
            }


            cout<<"\n\t------------------------ Bill Details ---------------------\n\n";
            fout <<"\n\t------------------------ Bill Details ---------------------\n\n";

            //cout<<"\t---------------------- Taxt total is 400.99 $ ------------------\n\n";
            //outfile<<"\t---------------------- Taxt total is 400.99 $ ------------------\n\n";

                   cout<<"\t------------------------- Selected Items ----------------------\n\n";
                   fout <<"\t------------------------- Selected Items ----------------------\n\n";

                   for(int i = 0; i < selectitems; i++){
                       cout << endl;
                       cout << "\t\t\t" << i + 1 << ": " << food[sfood[i] - 1] << "\t\t"<< fprice[sfood[i] - 1] <<"PKR."<<endl;
                       fout << "\t\t\t" << i + 1 << ": " << food[sfood[i] - 1] << "\t\t"<< fprice[sfood[i] - 1] <<"PKR."<<endl;
                       bill += fprice[sfood[i] - 1];
                   }


                   cout<<"\n\t---------------------------- Menu Bill -------------------------\n\n";
                   fout <<"\n\t---------------------------- Menu Bill -------------------------\n\n";

                   cout<<"\t\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
                   fout<<"\t\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
               }

           //operator overloading function definition
           friend istream &operator >> (istream &i, Menu &m);
           friend ostream &operator << (ostream &o, Menu &m);

           ~Menu(){
               fout.close();
               fin.close();
           }
       };

       istream &operator >> (istream &i, Menu &m){
           int selectitems, rollfood, bill = 0;
           cout << "\nEnter the no. of items that you would love to select: ";
           cin >> selectitems;
           string line, word;
           int sfood[selectitems];
           for(int i = 0; i < selectitems; i++){
               cout << "Enter item no.: ";
               cin >> sfood[i];
               m.i1.update_Inventory(sfood[i]);
           }


           cout<<"\n\t------------------------ Bill Details ---------------------\n\n";
           m.fout <<"\n\t------------------------ Bill Details ---------------------\n\n";

           //cout<<"\t------------------------ Taxt total is 400.99 $ ------------------\n\n";
           //outfile<<"\t------------------------ Taxt total is 400.99 $ ------------------\n\n";

           cout<<"\t-------------------------- Selected Items -----------------------\n\n";
           m.fout <<"\t-------------------------- Selected Items -----------------------\n\n";

           for(int i = 0; i < selectitems; i++){
               cout << "\t\t\t" << i + 1 << ": " << m.food[sfood[i] - 1] << "\t\t"<< m.fprice[sfood[i] - 1] <<"PKR."<<endl;
               m.fout << "\t\t\t" << i + 1 << ": " << m.food[sfood[i] - 1] << "\t\t"<< m.fprice[sfood[i] - 1] <<"PKR."<<endl;
               bill += m.fprice[sfood[i] - 1];
           }

           cout<<"\n\t---------------------------- Menu Bill -------------------------\n\n";
           m.fout <<"\n\t---------------------------- Menu Bill -------------------------\n\n";

           cout<<"\t\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
           m.fout<<"\t\t\t\t\t\tTotal Bill : "<<bill<<"PKR."<<endl;
       }

```

```cpp
    ostream &operator << (ostream &o, Menu &m){
        vector<string> row;
        string line, word;
        int i = 0;
        cout << "\t ****************************** \t\n"
        << "\t   Menu Items \t\n"
        << "\t ****************************** \t\n";
        while(!m.fin.eof()){
            row.clear();

            // read an entire row and
            // store it in a string variable 'line'
            getline(m.fin, line);

            // used for breaking words
            stringstream s(line);
            while (getline(s, word, ',')) {
                // add all the column data
                // of a row to a vector
                row.push_back(word);
            }
            cout << "\t " << row[0] << ". " << row[1] << "\t" << " ...PKR" << row[2] << endl;
            m.food[i] = row[1];
            m.fprice[i] = stoi(row[2]);
            m.fnum[i] = stoi(row[0]);
            i++;
            }
            cout << endl;
    }

    class Customer{
        protected:
            char *name;
            string *id;
            int *table;
        public:
            Customer(){
                name=new char[100];
                id=new string;
                table = new int;
            }
            virtual void set_detail(){
            }
            virtual void get_detail() = 0;
    };
    class CustomerDetails : public Customer{
        public:
            CustomerDetails(){} //the Constructor...
            void set_detail(){
                ofstream out11;
                out11.open("bill.txt",ios::out | ios::app);
                cout<<"\nEnter a your name : ";
                cin.get(name,100); // use charactor case
                cout<<"\n\nEnter a your id_card number : ";
                cin>>*id;
                cout<<"\n\nEnter a your table number : ";
                cin>>*table;
            }
            void gets_detail(){
                ofstream out12;
```

```
                    out12.open("bill.txt",ios::out | ios::app);
                    cout<<"\nName: "<<name<<endl;
                    out12<<"\nName: "<<name<<endl;
                    cout<<"\nID_Card: "<<*id<<endl;
                    out12<<"ID_Card: "<<*id<<endl;
                    cout<<"\nTable: "<<*id<<endl;
                    out12<<"Table: "<<*table<<endl;
                }
                ~CustomerDetails(){
                }
        };

        #endif
```

inventory.h:

```
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <sstream>
#include <vector>
#include <bits/stdc++.h>

#ifndef INVENTORY_H
#define INVENTORY_H

using namespace std;

class Inventory{
    protected:
        fstream fin1;
        string line;
    public:
        Inventory(){
            // Open an existing record
            fin1.open("inventory.csv", ios::in);
        }
        virtual void displayInventory() = 0;
        ~Inventory(){
            // Close an existing record
            fin1.close();
        }
};

class DisplayInventory : public Inventory{
    public:
        void displayInventory() override {
            vector<string> row;
            string line, word;
            cout << "\t ****************************** \t\n"
            << "\t   Inventory \t\n"
            << "\t ****************************** \t\n";

            while(!fin1.eof()){
                row.clear();

```

```
                    getline(fin1, line);

                    // used for breaking words
                    stringstream s(line);
                    while (getline(s, word, ',')) {
                        // add all the column data
                        // of a row to a vector
                        row.push_back(word);
                    }
                    cout << row[0] << ". " << row[1] << ": " << row[2] << row[3] << endl;
               }
            }
};

class UpdateInventory{
    public:
    UpdateInventory(){

    }
    ~UpdateInventory(){

    }
    void update_Inventory(int sfood){
    fstream fin1, fout1;
    // Open an existing record
    fin1.open("inventory.csv", ios::in);
    // Create a new file to store updated data
    fout1.open("inventorynew.csv", ios::out);
    int found = 0, i = 0;
    float used=0.0;
    char sub;
    int index, new_marks;
    string line, word, party;
    vector<string> row;

    // Traverse the file
    while (!fin1.eof()) {

        row.clear();
        //row1.clear();

        getline(fin1, line);
        stringstream s(line);

        //getline(fin, line1);
        //stringstream s1(line1);

        while (getline(s, word, ',')) {
            row.push_back(word);
        }

        int row_size = row.size();

        if (sfood == 1){
            found = 1;
            if(stoi(row[0]) == 1){
                stringstream convert;

                       // sending a number as a stream into output string and adding new votes
                       used = stof(row[2]) - 0.5;
                       convert << used;
```

```

                        // the str() converts number into string
                        row[2] = convert.str();
                    }else if(stoi(row[0]) == 5){
                        stringstream convert;

                        // sending a number as a stream into output string and adding new votes
                        used = stof(row[2]) - 0.1;
                        convert << used;

                        // the str() converts number into string
                        row[2] = convert.str();
                    }else if(stoi(row[0]) == 3){
                        stringstream convert;

                        // sending a number as a stream into output string and adding new votes
                        used = stof(row[2]) - 1.0;
                        convert << used;

                        // the str() converts number into string
                        row[2] = convert.str();
                    }

                    if (!fin1.eof()) {
                        for (i = 0; i < row_size - 1; i++) {

                            // write the updated data
                            // into a new file 'inventorynew.csv'
                            // using fout
                            fout1 << row[i] << ",";
                        }

                        fout1 << row[row_size - 1] << endl;
                    }
                }
                else if (sfood == 2){
                    found = 1;
                    if(stoi(row[0]) == 2){
                        stringstream convert;

                        // sending a number as a stream into output string and adding new votes
                        used = stof(row[2]) - 0.08;
                        convert << used;

                        // the str() converts number into string
                        row[2] = convert.str();
                    }else if(stoi(row[0]) == 3){
                        stringstream convert;

                        // sending a number as a stream into output string and adding new votes
                        used = stof(row[2]) - 0.5;
                        convert << used;

                        // the str() converts number into string
                        row[2] = convert.str();
                    }else if(stoi(row[0]) == 7){
                        stringstream convert;

                        // sending a number as a stream into output string and adding new votes
                        used = stof(row[2]) - 2.0;
                        convert << used;
```

```cpp

                    // the str() converts number into string
                    row[2] = convert.str();
                }
                if (!fin1.eof()) {
                    for (i = 0; i < row_size - 1; i++) {

                        // write the updated data
                        // into a new file 'inventorynew.csv'
                        // using fout
                        fout1 << row[i] << ",";
                    }

                    fout1 << row[row_size - 1] << endl;
                }
            }
            else if (sfood == 3 || sfood == 4 || sfood == 5){
                found = 1;
                if(stoi(row[0]) == 2){
                    stringstream convert;

                    // sending a number as a stream into output string and adding new votes
                    used = stof(row[2]) - 0.5;
                    convert << used;

                    // the str() converts number into string
                    row[2] = convert.str();
                }else if(stoi(row[0]) == 4){
                    stringstream convert;

                    // sending a number as a stream into output string and adding new votes
                    used = stof(row[2]) - 1.0;
                    convert << used;

                    // the str() converts number into string
                    row[2] = convert.str();
                }else if(stoi(row[0]) == 5){
                    stringstream convert;

                    // sending a number as a stream into output string and adding new votes
                    used = stof(row[2]) - 0.2;
                    convert << used;

                    // the str() converts number into string
                    row[2] = convert.str();
                }else if(stoi(row[0]) == 6){
                    stringstream convert;

                    // sending a number as a stream into output string and adding new votes
                    used = stof(row[2]) - 1.0;
                    convert << used;

                    // the str() converts number into string
                    row[2] = convert.str();
                }
                if (!fin1.eof()) {
                    for (i = 0; i < row_size - 1; i++) {

                        // write the updated data
                        // into a new file 'inventorynew.csv'
                        // using fout
                        fout1 << row[i] << ",";
                    }

                    fout1 << row[row_size - 1] << endl;
                }
            }
            else {
                if (!fin1.eof()) {
                    for (i = 0; i < row_size - 1; i++) {

                        // writing other existing records
                        // into the new file using fout.
                        fout1 << row[i] << ",";
                    }

                    // the last column data ends with a '\n'
                    fout1 << row[row_size - 1] << endl;
                }
            }
            if (fin1.eof())
                break;
        }

        //Exception Handling
        try{
            if (found == 0){
                throw found;
            }
        }
        catch (int found) {
           cerr << "record not found\n";
        }

        fin1.close();
        fout1.close();

        // removing the existing file
        remove("inventory.csv");
        // renaming the updated file with the existing file name
        rename("inventorynew.csv", "inventory.csv");
        }
    };

    #endif
```

main.cpp

```
#include <iostream>
#include "main.h"
#include "inventory.h"
#include <bits/stdc++.h>

using namespace std;

int main() {
    char op;
    Restaurant r("Bon Appetit"); //Restaurant Class Object
    Menu m1; //Menu Class Object

    r.disInfo(); //displays Restaurant name
    cout << endl;

    //m1.displayMenu();
    cout << m1; //loads displayMenu() function using operator overloading and displays Menu

    do{
        //m1.selectMenu();
        cin >> m1; //loads selectMenu() function using operator overloading and let user
select their cuisine

        cout<<"\nWould you like to continue. (Y/N): ";
        cin >> op;

    }while((op == 'y') || (op == 'Y'));

    Inventory* i1; //Inventory Class Pointer Object
    DisplayInventory i1d; //DisplayInventory Derived Class Object
    i1 = &i1d; // case polymorphism
    cout << endl;
    i1 -> displayInventory(); // runtime use polymorphism

}
```

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.