Dustin Nguyen  
Professor Seol  
ITEC 4266  
2 May 2025

**Final Project**

**Executive summary**

This is a Coffee Shop Management System program that will help local coffee shop owners/employees manage their menu items, process customer orders, track inventory, and get daily sale reports.

**Project objectives**

The objective of this program will be to manage menu items, manage inventory, manage orders, create receipts, and create sale reports. All an employee has to do is open the program and select the choices from the following that are displayed.

**Progress overview**

I started this project planning on the base classes for the menu, order, inventory, and reports. I then set up the C++ environment and continued on in Visual Studios, then started creating failing test cases before implementation, then tried to fix bugs along the way as I code, implemented the base classes, added functions for menu updating, continued to the order system, continued to the inventory management, continued to the daily sale reports, then debugged the code and did a finalize user testing for the code as I create an entire menu, generate receipts, and update stock. Also, everything is operational within under a second and everything is 100% accurate with values.

**Key achievements**

Daily sale reports are timestamped, employees can use the program to efficiently place orders, generate receipts, and see real-time stock.

**Challenges and Solutions**

I saw the inventory manager could have duplicate inventories so I used a destructor to remove any duplicates. I also made sure, to further assist employees, to make a 0 stock item be in the menu, so employees can just update the specific item in case it gets stock back instead of re-entering or re-adding the item back in the menu.

**Cost and resources**

The resources are only C++, Visual Studios, and a computer while the costs are only time (11 hours) and system resources.

**Risks and any issues**

A big risk would be no actual data saving, meaning all data is lost upon closing the program, only simple error handling was implemented, and a good suggestion would be, in the future, to add actual savable data/databases/files and make this program not console based.

**Conclusion**

I feel the program meets the requirements of managing orders, menu, stock, and daily sale reports. I think it’s a good prototype to something bigger, and definitely a good start on learning how to make bigger projects later on.

**Source Code**

#include <iostream>

#include <vector>

#include <unordered\_map>

#include <iomanip>

#include <ctime>

using namespace std;

// Class for Menu Items

class MenuItem {

public:

string name;

double price;

int stock;

// Initialize menu item

MenuItem(string n, double p, int s) : name(n), price(p), stock(s) {}

};

// Class for the Inventory Manager

class InventoryManager {

unordered\_map<string, MenuItem\*> menu;

public:

// Have user add a menu item

void addItem(string name, double price, int stock) {

menu[name] = new MenuItem(name, price, stock);

cout << name << " added to menu.\n";

}

// Have user update a menu item

void updateItem(string name, double price, int stock) {

if (menu.count(name)) {

menu[name]->price = price;

menu[name]->stock = stock;

cout << name << " updated.\n";

} else {

cout << "Item not found.\n";

}

}

// Have user remove a menu item

void removeItem(string name) {

if (menu.count(name)) {

delete menu[name];

menu.erase(name);

cout << name << " removed from menu.\n";

} else {

cout << "Item not found.\n";

}

}

// Return item availability

bool isAvailable(string name, int quantity) {

return menu.count(name) && menu[name]->stock >= quantity;

}

// Return item price

double getPrice(string name) {

return menu[name]->price;

}

// Subtract stock after a sale

void deductStock(string name, int quantity) {

if (menu.count(name))

menu[name]->stock -= quantity;

}

// Output the menu, its price, and stock

void showMenu() {

cout << "\n--- Menu ---\n";

for (auto& pair : menu) {

cout << pair.first << " - $" << fixed << setprecision(2)

<< pair.second->price << " (" << pair.second->stock << " in stock)\n";

}

}

// Return full menu

unordered\_map<string, MenuItem\*> getMenu() {

return menu;

}

// Inventory Manager removing any duplicates

~InventoryManager() {

for (auto& pair : menu) {

delete pair.second;

}

}

};

// Employee making customer order

class Order {

vector<pair<string, int>> items; // List of items in the order

double total = 0.0; // Total cost of the order

public:

// Add item to order if available

void addItem(string name, int quantity, InventoryManager& inv) {

if (inv.isAvailable(name, quantity)) {

items.push\_back({name, quantity});

total += inv.getPrice(name) \* quantity;

inv.deductStock(name, quantity);

} else {

cout << "Item " << name << " is out of stock.\n";

}

}

// Return the total cost

double getTotal() {

return total;

}

// Output receipt for the order

void printReceipt() {

cout << "\n--- Receipt ---\n";

for (auto& item : items) {

cout << item.first << " x" << item.second << "\n";

}

cout << "Total: $" << fixed << setprecision(2) << total << "\n";

}

};

// Tracks total sales for the day and generates reports

class ReportGenerator {

double dailySales = 0.0;

public:

// Record a new sale

void recordSale(double amount) {

dailySales += amount;

}

// Generate and print daily sales report

void generateReport() {

time\_t now = time(0);

char\* dt = ctime(&now); // Get current time

cout << "\n--- Daily Sales Report ---\n";

cout << "Date: " << dt;

cout << "Total Sales: $" << fixed << setprecision(2) << dailySales << "\n";

}

};

// Entry point of the application

int main() {

InventoryManager inventory;

ReportGenerator report;

int choice;

// Loop Choices

do {

cout << "\n1. Show Menu\n2. Add Menu Item\n3. Update Menu Item\n4. Remove Menu Item\n5. Make Order\n6. Sales Report\n0. Exit\nChoice:";

cin >> choice;

cin.ignore();

// Else-if statements where 0 will end the loop

// This choice will show the menu

if (choice == 1) {

inventory.showMenu();

} //This choice will add items to the menu

else if (choice == 2) {

string name; double price; int stock;

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

cout << "Enter price: "; cin >> price;

cout << "Enter stock: "; cin >> stock;

inventory.addItem(name, price, stock);

} // This choice will update items in the menu

else if (choice == 3) {

string name; double price; int stock;

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

cout << "Enter new price: "; cin >> price;

cout << "Enter new stock: "; cin >> stock;

inventory.updateItem(name, price, stock);

} // This choice will remove items in the menu

else if (choice == 4) {

string name;

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

inventory.removeItem(name);

} // This choice will place orders, print receipt, and record the sale to the daily sales report

else if (choice == 5) {

Order order;

string item; int quantity;

char more;

do {

cout << "Enter item: "; getline(cin, item);

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

cin.ignore();

order.addItem(item, quantity, inventory);

cout << "Add more? (y/n): "; cin >> more;

cin.ignore();

} while (more == 'y');

order.printReceipt();

report.recordSale(order.getTotal());

} // This choice will display the daily sales report

else if (choice == 6) {

report.generateReport(); // Show daily sales

}

} while (choice != 0);

return 0;

}