**C language program report: Amazon backend system**

| | |
|---|---|
| Mr. Supanut Sopha | ID : 67070503441 |
| Mr. Supapanya Yathip | ID : 67070503443 |
| Mr. Atip Infa-udom | ID : 67070503446 |
| Mr. Pawarit Wongdaeng | ID : 67070503466 |
| Mr. Piti Srisongkram | ID : 67070503467 |

This report is submitted as part of the coursework for CPE 100: Computer Programming for Engineering

Master of Engineering in Computer Engineering

Faculty of Engineering

King Mongkut's University of Technology Thonburi

Academic Year 2024

# Chapter 1 Introduction

## 1.1. Topic

Backend system for large e-commerce companies Amazon. This project focuses on developing a comprehensive backend system for a large e-commerce company, like Amazon. The system manages complex inventory management processes with multiple warehouses, tracks product movement between warehouses to maintain proper stock levels, and has a product search system with filters for writing such as quantity, price, or product name. In the customer-facing part, the system supports product sales to ensure that there is enough stock to fulfill customer orders. When a customer places an order, the system identifies the appropriate warehouse to ship the product.

## 1.2. Objectives

Objectives of the C Programming Project for the Backend System of an E-Commerce Company like Amazon:

1.2.1. To practice C programming and gain a deeper understanding of the language through project-based learning.

1.2.2. To develop an inventory management system that can track stock in multiple warehouses, with product quantities being updated whenever items are added, moved, or sold.

1.2.3. To enable customer-facing sales operations by creating a system that ensures product availability for customers making purchases.

1.2.4. To automate order fulfillment and shipping by coordinating the delivery of products from the most suitable warehouse to customers, reducing shipping time and cost by selecting the optimal warehouse for order dispatch.

1.2.5. To improve data management and file handling by reading and writing data from CSV files for products, orders, and users.

1.2.6. To support role-based access (admin and customer roles) by providing separate functionalities for admin users (product management) and customer users (browsing, searching, and purchasing products).

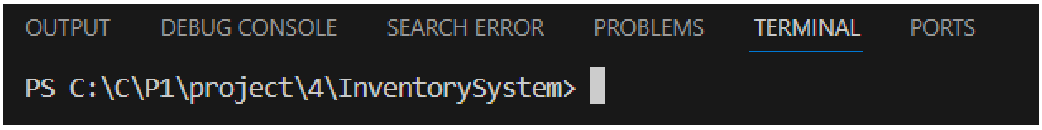
1.2.7. To promote real-time product availability to prevent customers from ordering out-of-stock items.

1.2.8. To provide a user-friendly interface and navigation by developing a simple and easy-to-use text-based interface for customers and admins to interact with the system.

### 1.3. How to make the program work

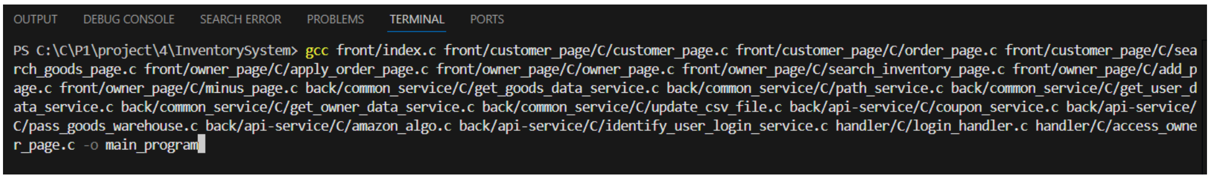
Link Repo Program: https://github.com/kmood-Sakura/InventorySystem/

1. After loading the file and opened file in the folder in vs code.
2. Open the terminal page and check if the location we are in is correct or not. Look at the folder in the outermost location of all files. If it is not at the outermost position, use the command “cd..” to exit to the largest folder

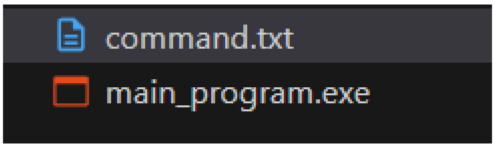


```
PS C:\C\P1\project\4\InventorySystem>
```

3. After that, use the command “gcc front/index.c front/customer_page/C/customer_page.c front/customer_page/C/order_page.c front/customer_page/C/search_goods_page.c front/owner_page/C/apply_order_page.c front /owner_page/C/owner_page.c front/owner_page/C/search_inventory_page.c front/owner_page/C/add_page.c front/owner_page/C/minus_page.c back/common_service/C/get_goods_data_service.c back/common_service/C/path_service.c back/common_service/C/get_user_data_service.c back/common_service/C /get_owner_data_service.c back/common_service/C/update_csv_file.c back/api-service/C/coupon_service.c back/api-service/C/pass_goods_warehouse.c back/api-service/C/amazon_algo.c back/api-service/C/identify_user_login_service.c handler/C/login_handler. c handler/C/access_owner_page.c -o main_program” to merge all files together and make them work.

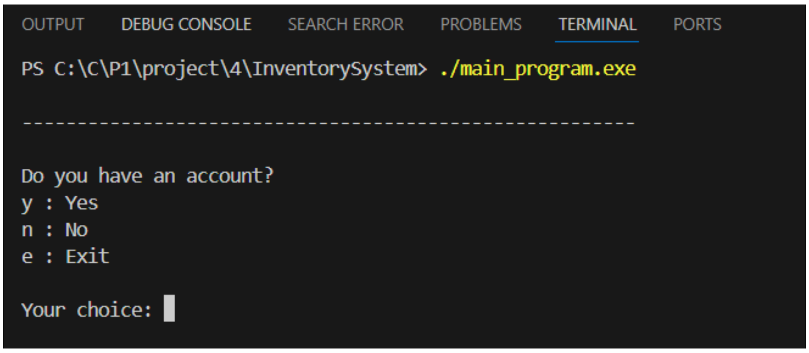


```
PS C:\C\P1\project\4\InventorySystem> gcc front/index.c front/customer_page/C/customer_page.c front/customer_page/C/order_page.c front/customer_page/C/sea
rch_goods_page.c front/owner_page/C/apply_order_page.c front/owner_page/C/owner_page.c front/owner_page/C/search_inventory_page.c front/owner_page/C/add_p
age.c front/owner_page/C/minus_page.c back/common_service/C/get_goods_data_service.c back/common_service/C/path_service.c back/common_service/C/get_user_d
ata_service.c back/common_service/C/get_owner_data_service.c back/common_service/C/update_csv_file.c back/api-service/C/coupon_service.c back/api-service/
C/pass_goods_warehouse.c back/api-service/C/amazon_algo.c back/api-service/C/identify_user_login_service.c handler/C/login_handler.c handler/C/access_owne
r_page.c -o main_program
```



command.txt
main_program.exe

4. Once we have created the file, use the command "./main_program.exe" is used to command the program to open. After that, the program will be able to run and can be used. When you want to exit the program, click "Ctrl + C" to exit the program.



```
PS C:\C\P1\project\4\InventorySystem> ./main_program.exe

----------------------------------------------------

Do you have an account?
y : Yes
n : No
e : Exit

Your choice: 
```

## Chapter 2 Data structure

The project will consist of 5 files used to store different data, which are divided based on the features as follows:

2.1 **inventory.csv** - Stores all product information in the system, such as product ID, product name, description, price, and available quantity. This information is used for viewing products and performing other related operations. In this file, there will be 5 files in total, divided into folders that will contain the user's main, secondary and main data.

2.2 **coupon.csv** - Stores information about coupons, such as coupon code, discount percentage, discount amount, and usage status. This data is used to apply discounts for customers when purchasing products.

2.3 **orders.csv** - This file records customer order information, including the product ID, good ID, user ID, product price, and quantity of the product. All this data is stored as order records, which also serve as a purchase history for users.

2.4 **user.csv** - This file user information such as user ID, name, email, phone number, and password, which are used for logging into the system. Users can browse and purchase products, and those with appropriate permissions can also access the backend system to manage inventory. (2 file)

2.5 **warehouse.csv** - Records information about warehouses, such as warehouse ID, warehouse name, and the products stored in the warehouse. This data is used to display products specific to each warehouse, allowing for inventory management within individual warehouses.

2.6 **owner.csv** – This file stores information such as ID, token, and name, which are used to access the backend system for managing products. It also allows users to view all products in the warehouse and is utilized for authentication during system login. (2 file)

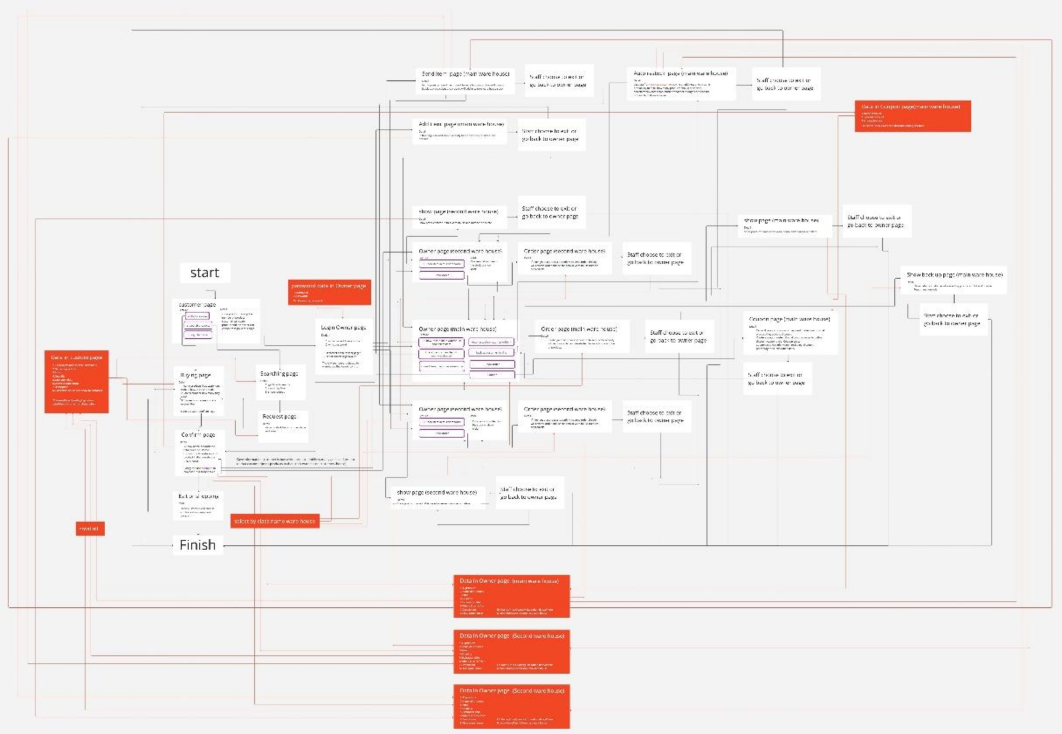
2.7 **remove-list.csv** – This file stores information such as the warehouse ID, warehouse name, product ID, and product quantity. It is used to track and facilitate the transfer of products from one warehouse to another.

2.8 **Transfer.csv** – This file stores the product ID and quantity of products to be moved to the selected warehouse. This information is used when the system automatically updates the inventory by reading the file. (2 file)

2.9 **Subwarehouse.csv** – This file will store the warehouse ID so that it can be retrieved when the warehouse ID is required.

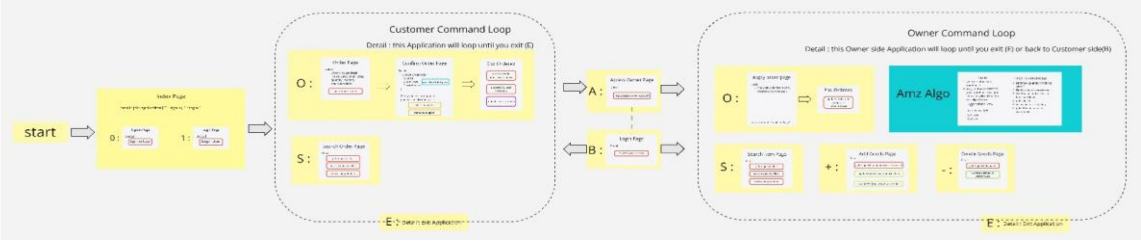
## Chapter 3 Flowchart and diagram

Flowchart C programming design that shows how the entire code works and how each thing works, as shown in the image below.



Link: https://miro.com/app/board/uXjVL2415g8=/?share_link_id=230614607013

Diagram C programming design:



Link: https://miro.com/app/board/uXjVL2415g8=/?share_link_id=230614607013

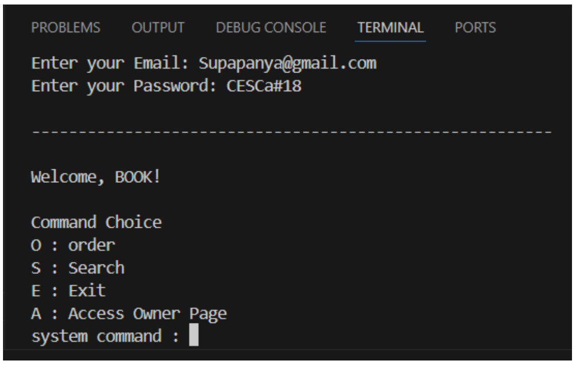
## Chapter 4 Feature explanation and responsibilities

### 4.1 Feature

In terms of coding or programming in C language, it is divided into 4 main parts. The programming features are as follows:

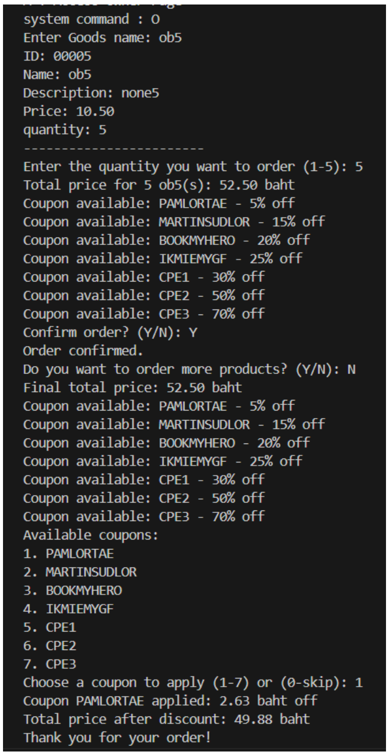
4.1.1 Front-end customer

I. The system selects the main functions of the first page, with options to order products, search for products, or exit the program and enter the back-end system.



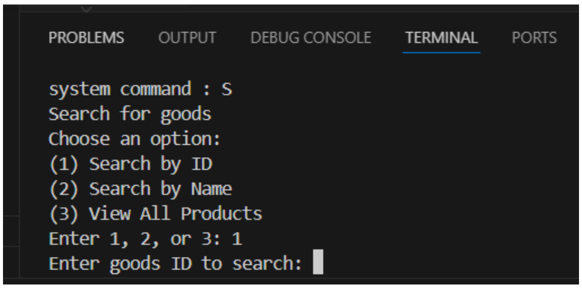
```
Enter your Email: Supapanya@gmail.com
Enter your Password: CESCa#18

-------------------------------------------------------

Welcome, BOOK!

Command Choice
O : order
S : Search
E : Exit
A : Access Owner Page
system command :
```

II. Coupon purchasing system and coupon system



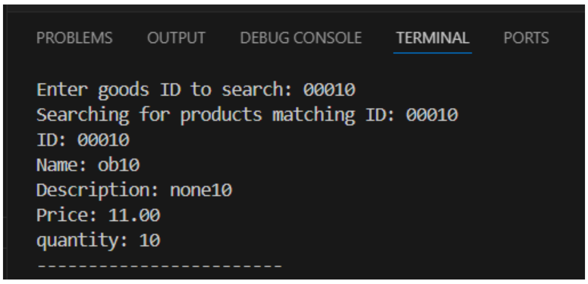
```
system command : O
Enter Goods name: ob5
ID: 00005
Name: ob5
Description: none5
Price: 10.50
quantity: 5
------------------------
Enter the quantity you want to order (1-5): 5
Total price for 5 ob5(s): 52.50 baht
Coupon available: PAMLORTAE - 5% off
Coupon available: MARTINSUDLOR - 15% off
Coupon available: BOOKMYHERO - 20% off
Coupon available: IKMIEMYGF - 25% off
Coupon available: CPE1 - 30% off
Coupon available: CPE2 - 50% off
Coupon available: CPE3 - 70% off
Confirm order? (Y/N): Y
Order confirmed.
Do you want to order more products? (Y/N): N
Final total price: 52.50 baht
Coupon available: PAMLORTAE - 5% off
Coupon available: MARTINSUDLOR - 15% off
Coupon available: BOOKMYHERO - 20% off
Coupon available: IKMIEMYGF - 25% off
Coupon available: CPE1 - 30% off
Coupon available: CPE2 - 50% off
Coupon available: CPE3 - 70% off
Available coupons:
1. PAMLORTAE
2. MARTINSUDLOR
3. BOOKMYHERO
4. IKMIEMYGF
5. CPE1
6. CPE2
7. CPE3
Choose a coupon to apply (1-7) or (0-skip): 1
Coupon PAMLORTAE applied: 2.63 baht off
Total price after discount: 49.88 baht
Thank you for your order!
```

III. Product search system divided into 3 functions



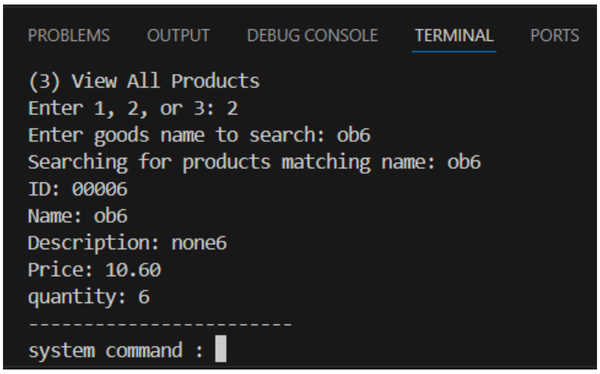
```
system command : S
Search for goods
Choose an option:
(1) Search by ID
(2) Search by Name
(3) View All Products
Enter 1, 2, or 3: 1
Enter goods ID to search: 
```

IV. Product search system using product ID



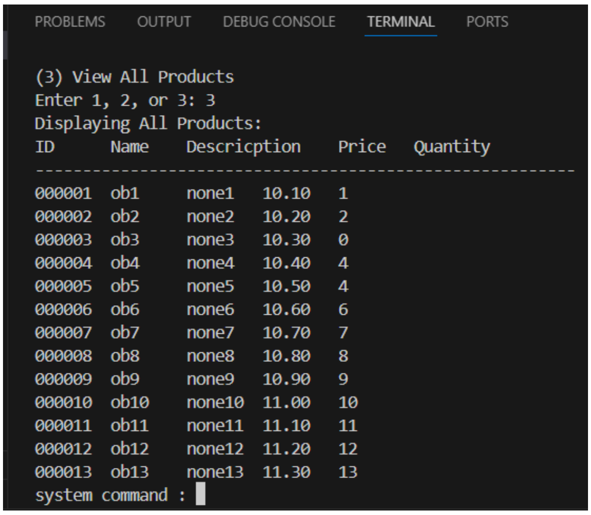
```
Enter goods ID to search: 00010
Searching for products matching ID: 00010
ID: 00010
Name: ob10
Description: none10
Price: 11.00
quantity: 10
------------------------
```

V. Product search system using product name



```
(3) View All Products
Enter 1, 2, or 3: 2
Enter goods name to search: ob6
Searching for products matching name: ob6
ID: 00006
Name: ob6
Description: none6
Price: 10.60
quantity: 6
------------------------
system command : 
```

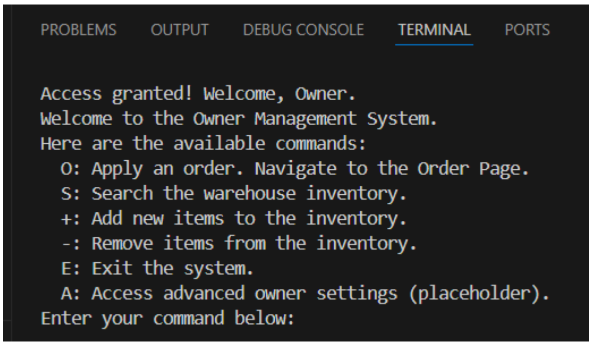
VI. All product display system



```
(3) View All Products
Enter 1, 2, or 3: 3
Displaying All Products:
ID      Name    Descricption    Price   Quantity
---------------------------------------------------------
000001  ob1     none1   10.10   1
000002  ob2     none2   10.20   2
000003  ob3     none3   10.30   0
000004  ob4     none4   10.40   4
000005  ob5     none5   10.50   4
000006  ob6     none6   10.60   6
000007  ob7     none7   10.70   7
000008  ob8     none8   10.80   8
000009  ob9     none9   10.90   9
000010  ob10    none10  11.00   10
000011  ob11    none11  11.10   11
000012  ob12    none12  11.20   12
000013  ob13    none13  11.30   13
system command : 
```

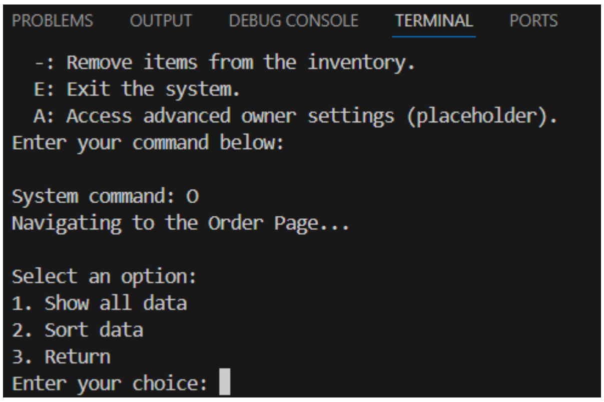
### 4.1.2 Front-end owner

I. The system displays a list of all the administrator's products.



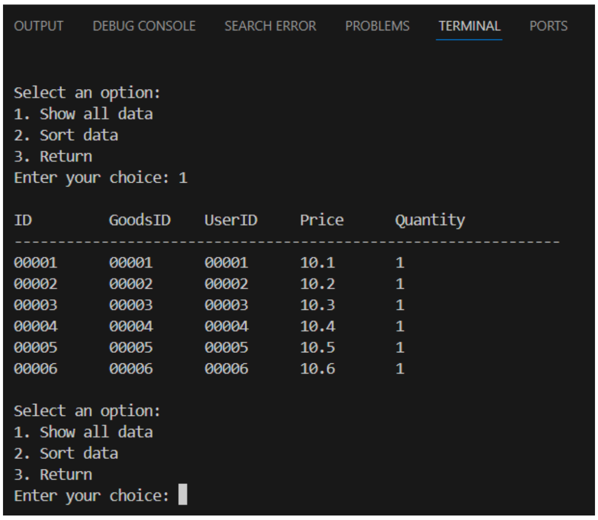
```
Access granted! Welcome, Owner.
Welcome to the Owner Management System.
Here are the available commands:
  O: Apply an order. Navigate to the Order Page.
  S: Search the warehouse inventory.
  +: Add new items to the inventory.
  -: Remove items from the inventory.
  E: Exit the system.
  A: Access advanced owner settings (placeholder).
Enter your command below:
```

II. The system displays the history of product orders, with the option to display all products or to sort products according to specified conditions, or to return to the main product management system.



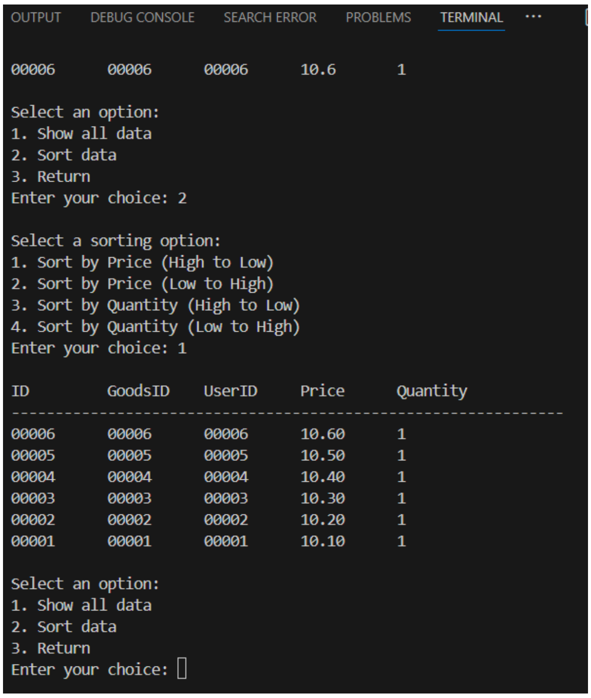
```
  -: Remove items from the inventory.
  E: Exit the system.
  A: Access advanced owner settings (placeholder).
Enter your command below:

System command: O
Navigating to the Order Page...

Select an option:
1. Show all data
2. Sort data
3. Return
Enter your choice: 
```

III. The system displays all information about the previously selected topic.



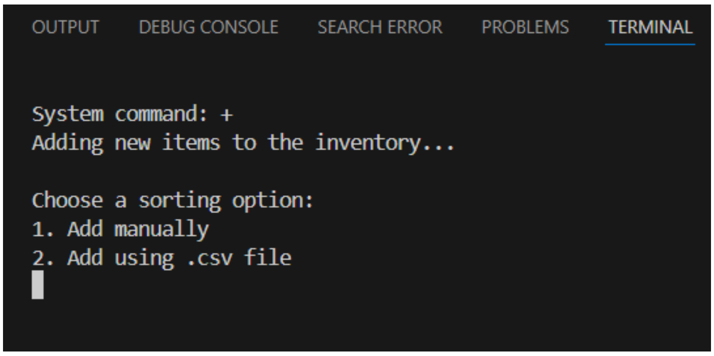
```
Select an option:
1. Show all data
2. Sort data
3. Return
Enter your choice: 1

ID          GoodsID     UserID      Price       Quantity
------------------------------------------------------------
00001       00001       00001       10.1        1
00002       00002       00002       10.2        1
00003       00003       00003       10.3        1
00004       00004       00004       10.4        1
00005       00005       00005       10.5        1
00006       00006       00006       10.6        1

Select an option:
1. Show all data
2. Sort data
3. Return
Enter your choice: 
```

IV. Product display system with filter selection to display products according to the previously selected topic.



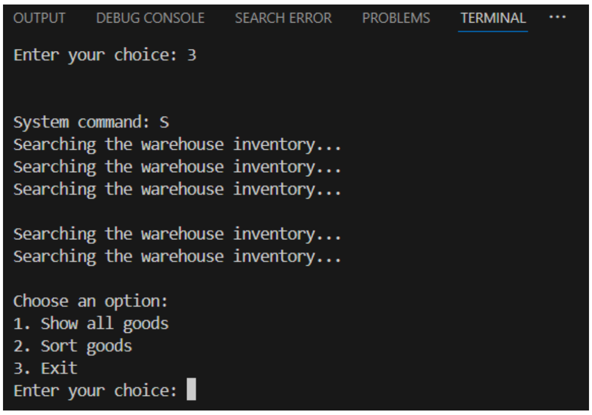
```
00006      00006      00006      10.6       1

Select an option:
1. Show all data
2. Sort data
3. Return
Enter your choice: 2

Select a sorting option:
1. Sort by Price (High to Low)
2. Sort by Price (Low to High)
3. Sort by Quantity (High to Low)
4. Sort by Quantity (Low to High)
Enter your choice: 1

ID         GoodsID    UserID     Price      Quantity
--------------------------------------------------------------
00006      00006      00006      10.60      1
00005      00005      00005      10.50      1
00004      00004      00004      10.40      1
00003      00003      00003      10.30      1
00002      00002      00002      10.20      1
00001      00001      00001      10.10      1

Select an option:
1. Show all data
2. Sort data
3. Return
Enter your choice:
```

V. The system for adding products has 2 types: 1. adding by entering products manually and 2. Adding by reading a .csv file.



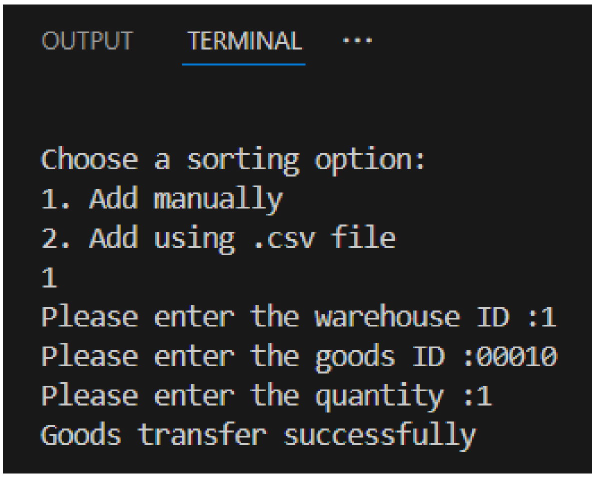
```
System command: +
Adding new items to the inventory...

Choose a sorting option:
1. Add manually
2. Add using .csv file
```

VI. Warehouse trading system.



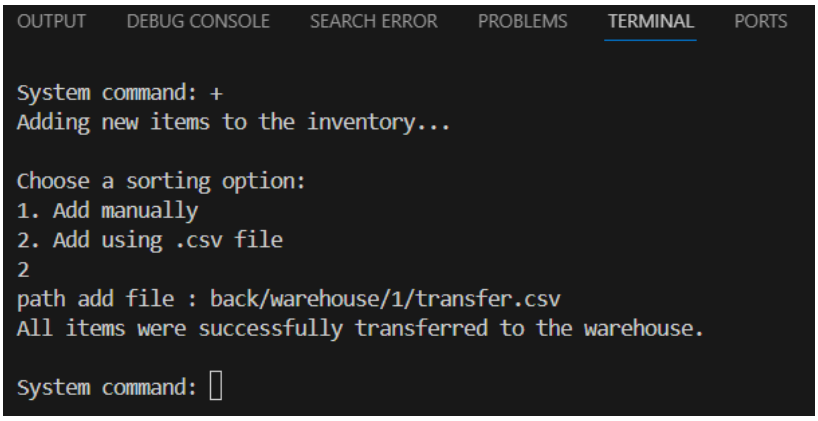
```
Enter your choice: 3


System command: S
Searching the warehouse inventory...
Searching the warehouse inventory...
Searching the warehouse inventory...

Searching the warehouse inventory...
Searching the warehouse inventory...

Choose an option:
1. Show all goods
2. Sort goods
3. Exit
Enter your choice:
```

VII. Manual product adding system, can select warehouse ID



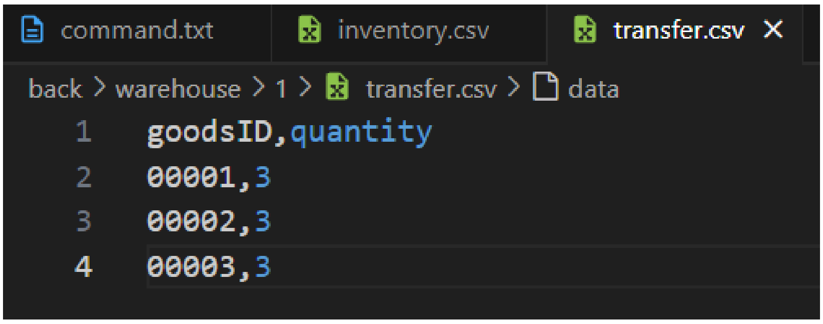
```
Choose a sorting option:
1. Add manually
2. Add using .csv file
1
Please enter the warehouse ID :1
Please enter the goods ID :00010
Please enter the quantity :1
Goods transfer successfully
```

VIII. The system adds products through a .csv file to the secondary warehouse.



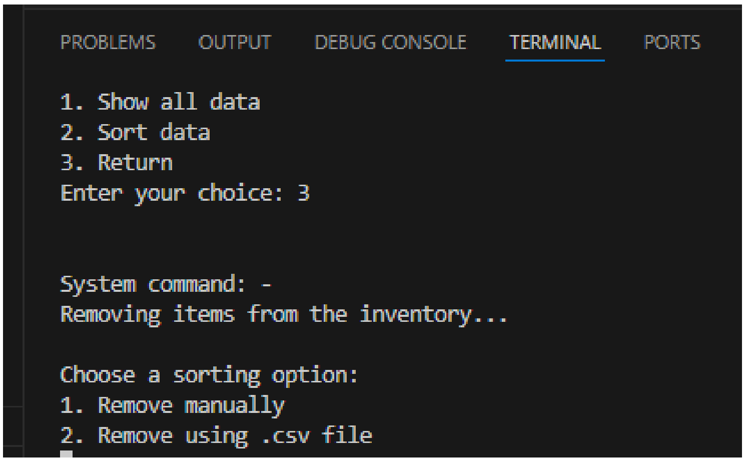
```
System command: +
Adding new items to the inventory...

Choose a sorting option:
1. Add manually
2. Add using .csv file
2
path add file : back/warehouse/1/transfer.csv
All items were successfully transferred to the warehouse.

System command: 
```

File sample



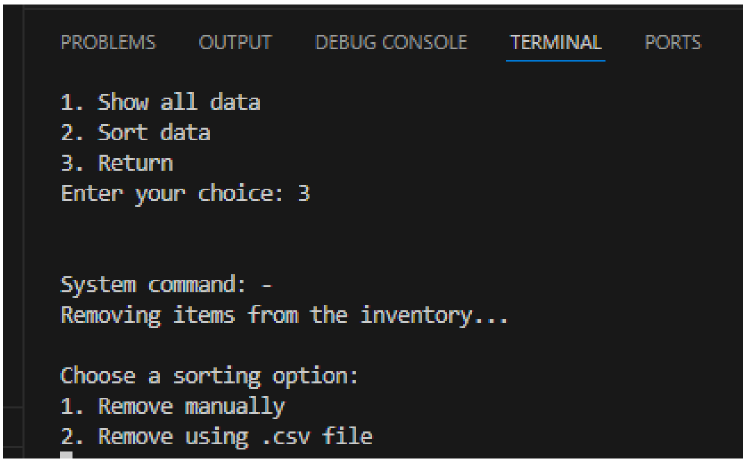
```
goodsID,quantity
00001,3
00002,3
00003,3
```

IX. Product deletion system



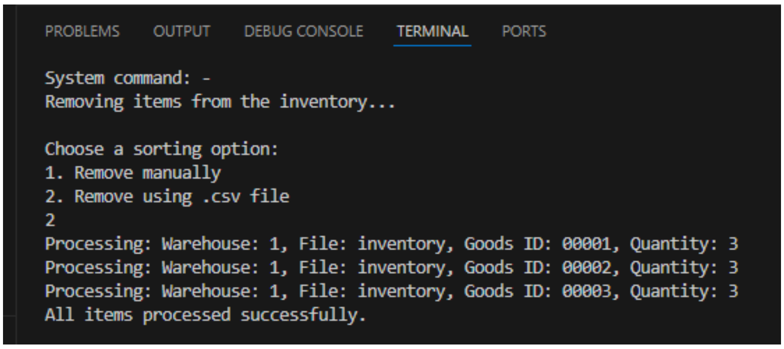
```
1. Show all data
2. Sort data
3. Return
Enter your choice: 3


System command: -
Removing items from the inventory...

Choose a sorting option:
1. Remove manually
2. Remove using .csv file
```

X. Manual or precise inventory removal system



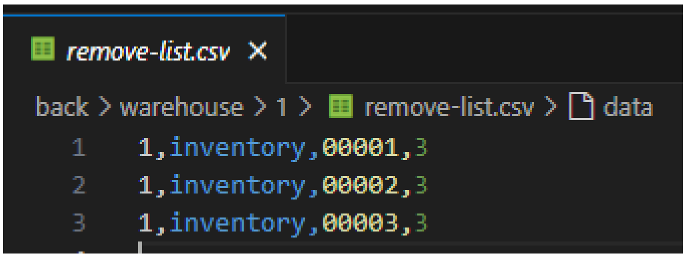
```
1. Show all data
2. Sort data
3. Return
Enter your choice: 3


System command: -
Removing items from the inventory...

Choose a sorting option:
1. Remove manually
2. Remove using .csv file
```

XI. Product deletion system by reading through .csv file



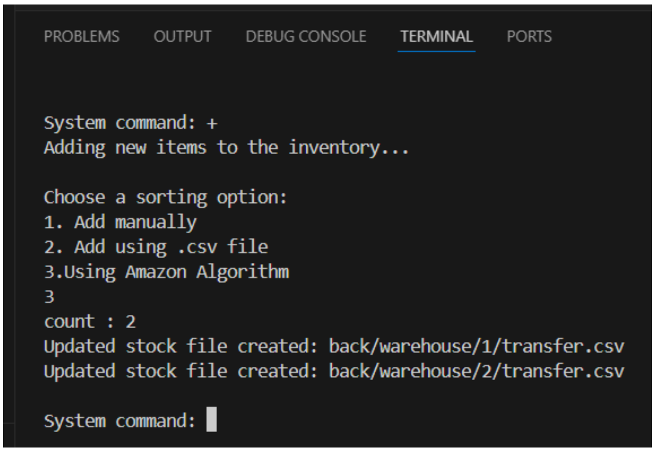
```
System command: -
Removing items from the inventory...

Choose a sorting option:
1. Remove manually
2. Remove using .csv file
2
Processing: Warehouse: 1, File: inventory, Goods ID: 00001, Quantity: 3
Processing: Warehouse: 1, File: inventory, Goods ID: 00002, Quantity: 3
Processing: Warehouse: 1, File: inventory, Goods ID: 00003, Quantity: 3
All items processed successfully.
```

File sample



```
1,inventory,00001,3
1,inventory,00002,3
1,inventory,00003,3
```

XII. Algorithmic system for adding products to the warehouse by calculating the products before writing the CSV file.



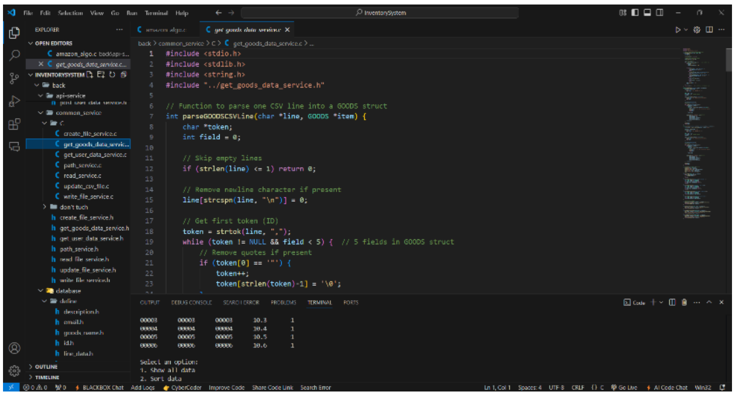
```
System command: +
Adding new items to the inventory...

Choose a sorting option:
1. Add manually
2. Add using .csv file
3.Using Amazon Algorithm
3
count : 2
Updated stock file created: back/warehouse/1/transfer.csv
Updated stock file created: back/warehouse/2/transfer.csv

System command: 
```

## 4.1.3 Back-end

### A. Common service

#### I. Get goods data

It can read product data in a CSV file and send values to the code that can be called in the form of a map of struct goods and can display products according to the CSV file.

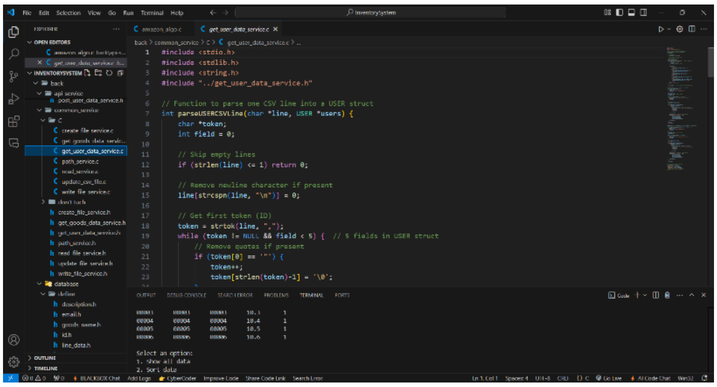


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../get_goods_data_service.h"

// Function to parse one CSV line into a GOODS struct
int parseGOODSCSVLine(char *line, GOODS *item) {
    char *token;
    int field = 0;

    // Skip empty lines
    if (strlen(line) <= 1) return 0;

    // Remove newline character if present
    line[strcspn(line, "\n")] = 0;

    // Get first token (ID)
    token = strtok(line, ",");
    while (token != NULL && field < 5) {  // 5 fields in GOODS struct
        // Remove quotes if present
        if (token[0] == '"') {
            token++;
            token[strlen(token)-1] = '\0';
```

```
00003      00003      00003      10.3      1
00004      00004      00004      10.4      1
00005      00005      00005      10.5      1
00006      00006      00006      10.6      1

Select an option:
1. Show all data
2. Sort data
```

II. Get user data

It can read user data in a csv file for login and send values to the callable code in the form of a map of struct user and can display user data according to the csv file

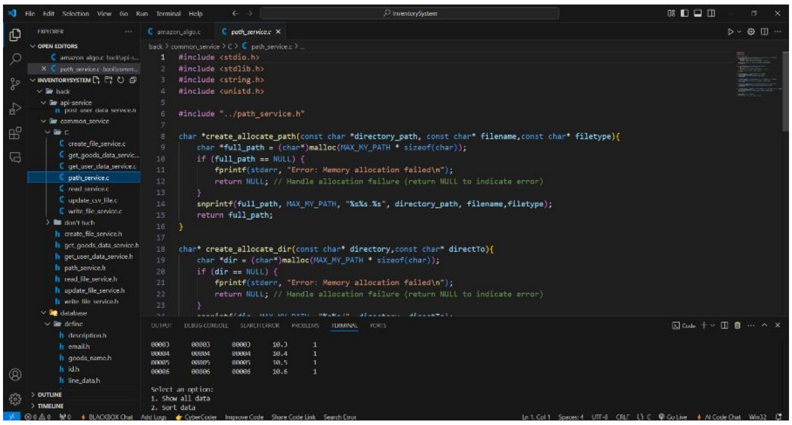


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../get_user_data_service.h"

// Function to parse one CSV line into a USER struct
int parseUSERCSVLine(char *line, USER *users) {
    char *token;
    int field = 0;

    // Skip empty lines
    if (strlen(line) <= 1) return 0;

    // Remove newline character if present
    line[strcspn(line, "\n")] = 0;

    // Get first token (ID)
    token = strtok(line, ",");
    while (token != NULL && field < 5) {  // 5 fields in USER struct
        // Remove quotes if present
        if (token[0] == '"') {
            token++;
            token[strlen(token)-1] = '\0';
```

```
00003     00003     00003     10.3      1
00004     00004     00004     10.4      1
00005     00005     00005     10.5      1
00006     00006     00006     10.6      1

Select an option:
1. Show all data
2. Sort data
```

III. Path service

It is used as a system to reduce the time required for paths and to make it easier to create systems that require paths to access data.

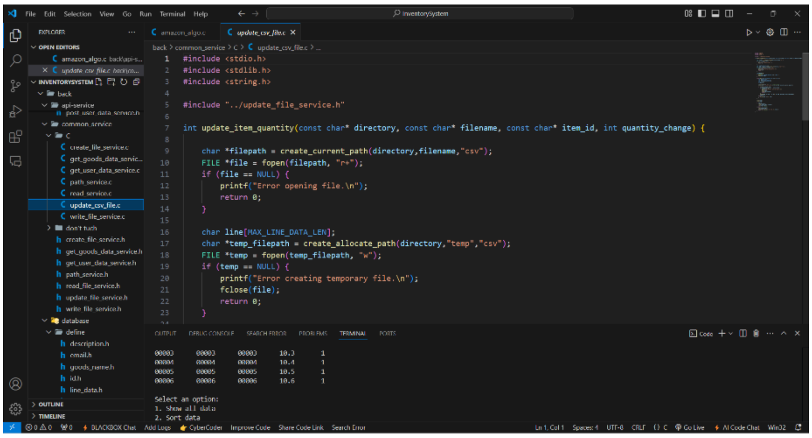


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include "../path_service.h"

char *create_allocate_path(const char *directory_path, const char* filename,const char* filetype){
    char *full_path = (char*)malloc(MAX_MY_PATH * sizeof(char));
    if (full_path == NULL) {
        fprintf(stderr, "Error: Memory allocation failed\n");
        return NULL; // Handle allocation failure (return NULL to indicate error)
    }
    snprintf(full_path, MAX_MY_PATH, "%s%s.%s", directory_path, filename,filetype);
    return full_path;
}

char* create_allocate_dir(const char* directory,const char* directTo){
    char *dir = (char*)malloc(MAX_MY_PATH * sizeof(char));
    if (dir == NULL) {
        fprintf(stderr, "Error: Memory allocation failed\n");
        return NULL; // Handle allocation failure (return NULL to indicate error)
    }
```

```
00003     00003     00003     10.3      1
00004     00004     00004     10.4      1
00005     00005     00005     10.5      1
00006     00006     00006     10.6      1

Select an option:
1. Show all data
2. Sort data
```

IV. Update file service

It is used to update data in various csv files that are called by using the data ID to confirm the location where the data needs to be changed according to the call of this feature.



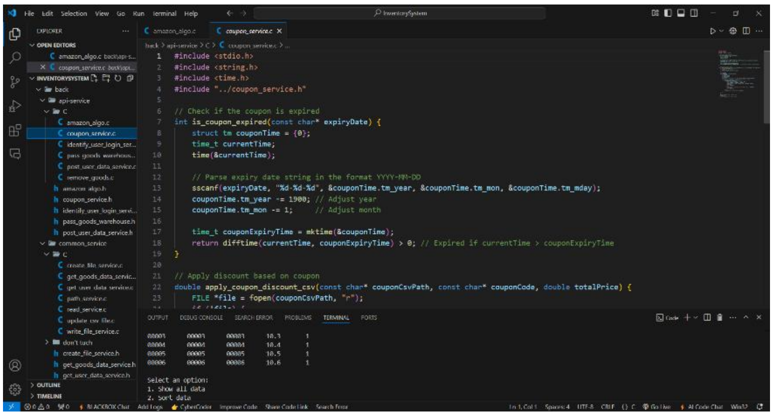
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "../update_file_service.h"

int update_item_quantity(const char* directory, const char* filename, const char* item_id, int quantity_change) {

    char *filepath = create_current_path(directory,filename,"csv");
    FILE *file = fopen(filepath, "r+");
    if (file == NULL) {
        printf("Error opening file.\n");
        return 0;
    }

    char line[MAX_LINE_DATA_LEN];
    char *temp_filepath = create_allocate_path(directory,"temp","csv");
    FILE *temp = fopen(temp_filepath, "w");
    if (temp == NULL) {
        printf("Error creating temporary file.\n");
        fclose(file);
        return 0;
    }
```

```
00003     00003     00003     10.3      1
00004     00004     00004     10.4      1
00005     00005     00005     10.5      1
00006     00006     00006     10.6      1

Select an option:
1. Show all data
2. Sort data
```

B. API service

I. Coupon sevice

It will manage the coupon system to check the expiration of the coupon or to edit the coupon usage status which will be called in the customer's ordering system.

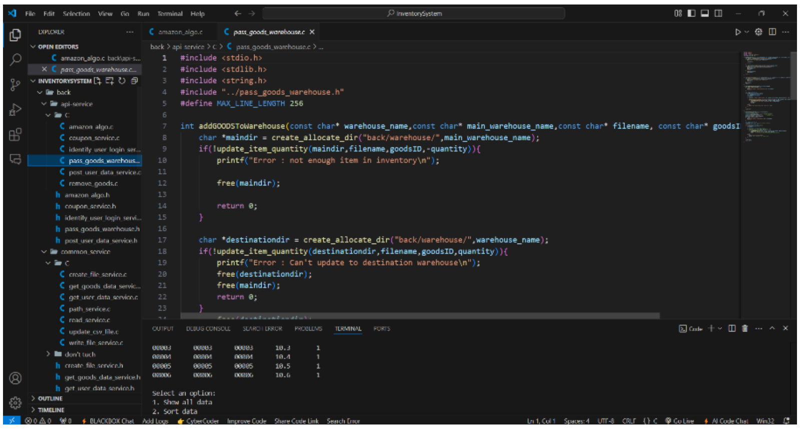


```c
#include <stdio.h>
#include <string.h>
#include <time.h>
#include "../coupon_service.h"

// Check if the coupon is expired
int is_coupon_expired(const char* expiryDate) {
    struct tm couponTime = {0};
    time_t currentTime;
    time(&currentTime);

    // Parse expiry date string in the format YYYY-MM-DD
    sscanf(expiryDate, "%d-%d-%d", &couponTime.tm_year, &couponTime.tm_mon, &couponTime.tm_mday);
    couponTime.tm_year -= 1900; // Adjust year
    couponTime.tm_mon -= 1;     // Adjust month

    time_t couponExpiryTime = mktime(&couponTime);
    return difftime(currentTime, couponExpiryTime) > 0; // Expired if currentTime > couponExpiryTime
}

// Apply discount based on coupon
double apply_coupon_discount_csv(const char* couponCsvPath, const char* couponCode, double totalPrice) {
    FILE *file = fopen(couponCsvPath, "r");
```

```
00003     00003     00003     10.3     1
00004     00004     00004     10.4     1
00005     00005     00005     10.5     1
00006     00006     00006     10.6     1

Select an option:
1. Show all data
2. Sort data
```

II. Pass goods warhouse

It will manage various data in the warehouse to send products to other warehouses or delete products from that warehouse.

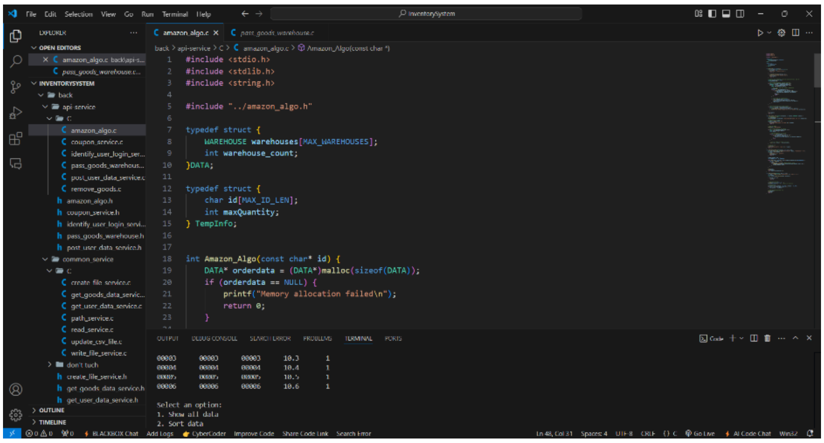


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "../pass_goods_warehouse.h"
#define MAX_LINE_LENGTH 256

int addGOODSToWarehouse(const char* warehouse_name,const char* main_warehouse_name,const char* filename, const char* goodsI
    char *maindir = create_allocate_dir("back/warehouse/",main_warehouse_name);
    if(!update_item_quantity(maindir,filename,goodsID,-quantity)){
        printf("Error : not enough item in inventory\n");

        free(maindir);

        return 0;
    }

    char *destinationdir = create_allocate_dir("back/warehouse/",warehouse_name);
    if(!update_item_quantity(destinationdir,filename,goodsID,quantity)){
        printf("Error : Can't update to destination warehouse\n");
        free(destinationdir);
        free(maindir);
        return 0;
    }
```

```
00003     00003     00003     10.3     1
00004     00004     00004     10.4     1
00005     00005     00005     10.5     1
00006     00006     00006     10.6     1

Select an option:
1. Show all data
2. Sort data
```

III. Amazon algo

It will be an automatic product replenishment system by calculating the products before shipping or replenishing the products. The calculation method will use the order quantity of every warehouse that has come in to find the ratio of product demand to be sent to the specified warehouses.

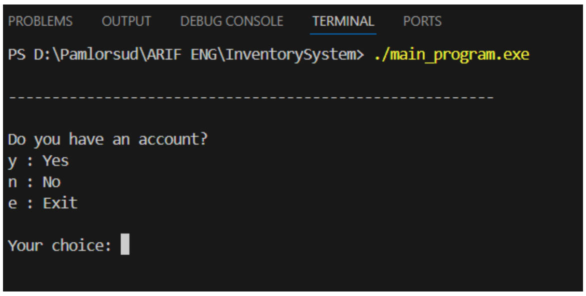


```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "../amazon_algo.h"

typedef struct {
    WAREHOUSE warehouses[MAX_WAREHOUSES];
    int warehouse_count;
}DATA;

typedef struct {
    char id[MAX_ID_LEN];
    int maxQuantity;
} TempInfo;


int Amazon_Algo(const char* id) {
    DATA* orderdata = (DATA*)malloc(sizeof(DATA));
    if (orderdata == NULL) {
        printf("Memory allocation failed\n");
        return 0;
    }
```

```
00003     00003     00003     10.3     1
00004     00004     00004     10.4     1
00005     00005     00005     10.5     1
00006     00006     00006     10.6     1

Select an option:
1. Show all data
2. Sort data
```

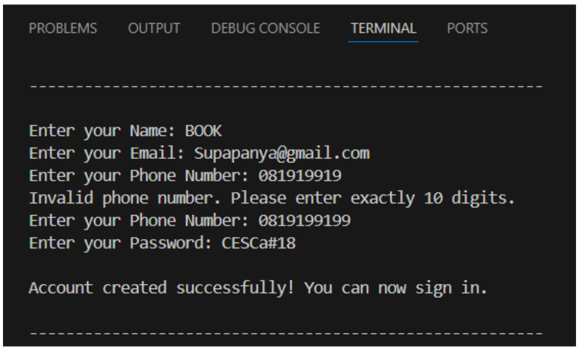
## 4.1.4 Logging system

I. The lock system has a username or if not, it will take you to the user creation page. If there is already one, you can continue using the program or choose to exit.



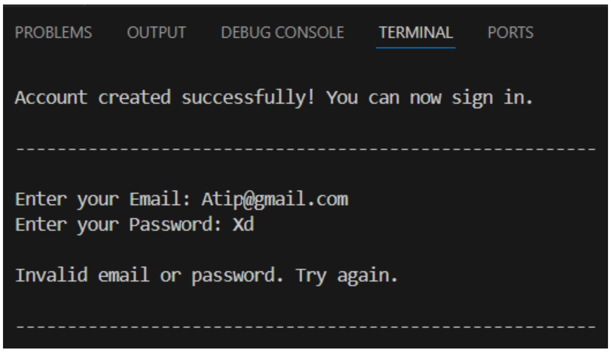
```
PS D:\Pamlorsud\ARIF ENG\InventorySystem> ./main_program.exe

---------------------------------------------------------

Do you have an account?
y : Yes
n : No
e : Exit

Your choice: 
```

II. The system generates a username by providing a name, email address, phone number, and password.



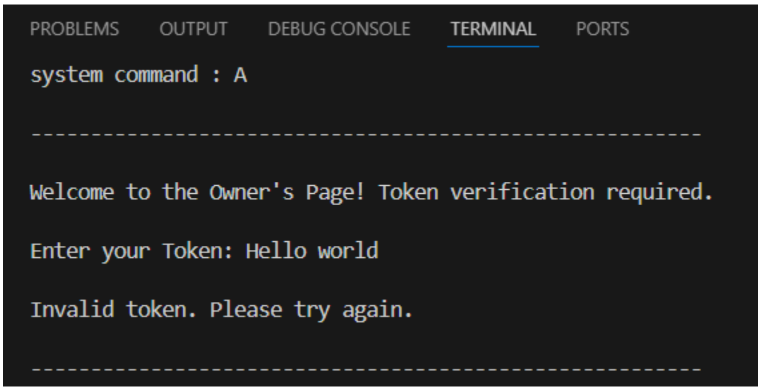
```
---------------------------------------------------------

Enter your Name: BOOK
Enter your Email: Supapanya@gmail.com
Enter your Phone Number: 081919919
Invalid phone number. Please enter exactly 10 digits.
Enter your Phone Number: 0819199199
Enter your Password: CESCa#18

Account created successfully! You can now sign in.

---------------------------------------------------------
```

III. Login system using email and password



```
Account created successfully! You can now sign in.

-----------------------------------------------------

Enter your Email: Atip@gmail.com
Enter your Password: Xd

Invalid email or password. Try again.

-----------------------------------------------------
```

IV. Administrator login system that provides a token to log into the administrator system.



```
system command : A

-----------------------------------------------------

Welcome to the Owner's Page! Token verification required.

Enter your Token: Hello world

Invalid token. Please try again.

-----------------------------------------------------
```

### 4.2 Responsibilities

The team responsibilities are assigned to each person to design different parts and then combined later, and the responsibilities of each person can be seen in this table.

| Summary table of project responsibilities | |
|---|---|
| **Name** | **Responsibilities** |
| Supanut Sopha | - System for moving goods between warehouses<br>- System for managing goods in selected warehouses<br>- System for searching warehouses<br>- System for filtering goods by price or quantity<br>- System for viewing goods |
| Supapanya Yathip | - Design Amazon Alchemy System<br>- Design the overall flow of the program<br>- Help organize various tasks within the team |
| Atip Infa-udom | - Purchase system |

| | |
|---|---|
| | - Product search system using product name and product ID<br>- Coupon system<br>- Advance order system using time |
| Pawarit Wongdaeng | - Login system for both users and owners to access the back office<br>- User registration system |
| Piti Srisongkram | - Design data within the program to listen to various results<br>- Create functions within the program<br>- Help to improve the code when there are additional questions |

## Chapter 5 Challenges and solutions

During the development of this project, we encountered several challenges, all of which were successfully resolved. The issues as follows:

### 5.1. File Management Challenge

The first issue we encountered was the challenge of managing a large number of files that were not properly organized. This led to confusion, making it difficult to access the correct files. When searching for specific functions or needing to modify code, we had to go through files one by one, which was very time-consuming. The solution was to reorganize the files by creating more detailed folders, such as separating .c and .h files, and giving the files more specific and descriptive names. This made it easier to identify and access the files, ensuring that when changes were needed, we knew exactly where to go and fix it.

### 5.2. File Opening Challenge

The second issue was a continuation of the first. After organizing the files into various folders, we encountered a new problem the inability to locate certain files when attempting to access them. This occurred because some files were nested within subfolders, leading to errors when trying to reference them. The solution was to thoroughly review the folder structure before attempting to access any files. We also implemented a habit of verifying the file paths before use. If this issue arose, we would first check the file path to ensure its accuracy before investigating other possible causes.

## Chapter 6 File path configuration

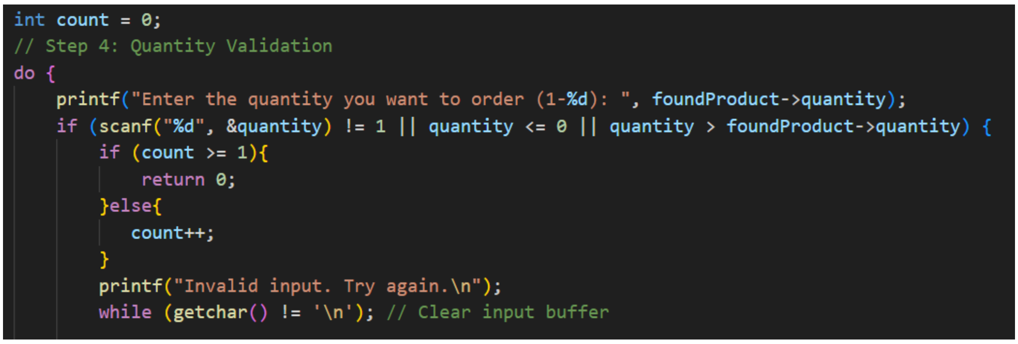
The file path that is in the program will be as follows that will be used:

- D:\HINEWLIFE\InventorySystem\back\database\inventory.csv
- D:\HI NEW LIFE\InventorySystem\back\database\owner.csv
- D:\HI NEW LIFE\InventorySystem\back\database\user.csv
- D:\HI NEW LIFE\InventorySystem\back\database\warehouse.csv
- D:\HI NEW LIFE\InventorySystem\back\user\1\coupon.csv
- D:\HI NEW LIFE\InventorySystem\back\user\1\inventory.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\1\inventory.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\1\order.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\1\remove-list.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\1\transfer.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\2\inventory.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\2\order.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\2\transfer.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\main\inventory.csv
- D:\HI NEW LIFE\InventorySystem\back\warehouse\main\subwarehouse.csv

# Improved from Feedback

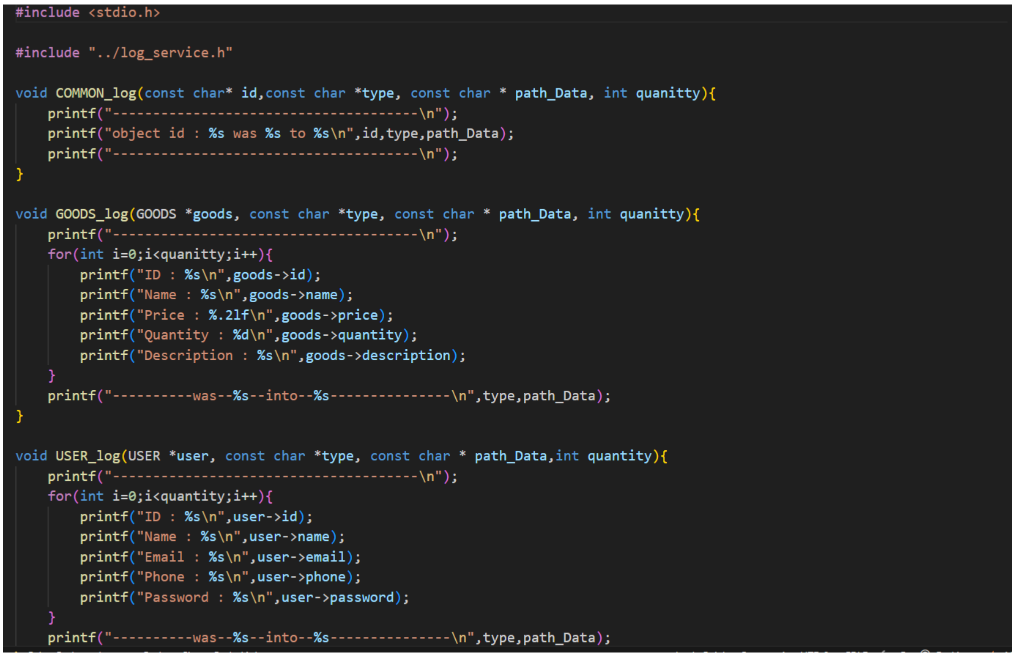
The feature or fixed we have improved will come from feedback from seniors and teachers who want us to adjust and add more features to make the program complete.

- Fixed the issue of entering the wrong amount more than 2 times, causing the program to repeatedly ask the user whether or not they want to buy the product or not.



```c
int count = 0;
// Step 4: Quantity Validation
do {
    printf("Enter the quantity you want to order (1-%d): ", foundProduct->quantity);
    if (scanf("%d", &quantity) != 1 || quantity <= 0 || quantity > foundProduct->quantity) {
        if (count >= 1){
            return 0;
        }else{
           count++;
        }
        printf("Invalid input. Try again.\n");
        while (getchar() != '\n'); // Clear input buffer
```

- The next thing that was improved was the addition of features such as adding a file log system that will keep a history of all the work done in the program, such as logging in, purchasing products, adding products to keep for viewing history.



```c
#include <stdio.h>

#include "../log_service.h"

void COMMON_log(const char* id,const char *type, const char * path_Data, int quanitty){
    printf("--------------------------------------\n");
    printf("object id : %s was %s to %s\n",id,type,path_Data);
    printf("--------------------------------------\n");
}

void GOODS_log(GOODS *goods, const char *type, const char * path_Data, int quanitty){
    printf("--------------------------------------\n");
    for(int i=0;i<quanitty;i++){
        printf("ID : %s\n",goods->id);
        printf("Name : %s\n",goods->name);
        printf("Price : %.2lf\n",goods->price);
        printf("Quantity : %d\n",goods->quantity);
        printf("Description : %s\n",goods->description);
    }
    printf("----------was--%s--into--%s---------------\n",type,path_Data);
}

void USER_log(USER *user, const char *type, const char * path_Data,int quantity){
    printf("--------------------------------------\n");
    for(int i=0;i<quantity;i++){
        printf("ID : %s\n",user->id);
        printf("Name : %s\n",user->name);
        printf("Email : %s\n",user->email);
        printf("Phone : %s\n",user->phone);
        printf("Password : %s\n",user->password);
    }
    printf("----------was--%s--into--%s---------------\n",type,path_Data);
```