



INDIVIDUAL ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT038-3.5-2-ODJ

PROGRAMMING IN CPP

NAME: PEDRO FABIAN OWONO ONDO MANGUE (TP063251)

HAND-IN DATE: 8 JUNE 2023

Contents

INTRODUCTION	3
USE CASE DIAGRAM.....	4
CLASS DIAGRAM	5
IMPLEMENTATION OF OOP CONCEPTS	7
SCREENSHOTS OF THE OUTPUT OF THE PROGRAM	11
CONCLUSION.....	21
REFERENCES.....	22

INTRODUCTION

Nowadays, since we are getting involved in the digital world. Some companies and industries are getting worried about how they can expand their businesses through digitalisation methods, meaning that, they are thinking to innovate and bringing new ways or mechanisms to significantly increase their capital or economy. So basically, as far as this project is concerned, I am designing an inventory system or a purchase management system where there are three roles or main actors which are:

- Admin
- Purchase Manager
- Sales Manager

They perform different functionalities in the system and their functionalities in the system is as follows:

The first role is the Purchase Manager which provides the following features:

- It can check or view the list of items of the complete system.
- It can check or view the list of the present suppliers of the system.
- It can display or show the Requisition of the complete system.
- Now, the Purchase Manager can manipulate (add, save, drop, and edit) the data for the Purchase Order or generate a purchase order.
- It can finally view the list of Purchase Orders.

The second role is the Sales Manager which provides the following features:

The Sales Manager can manipulate (add, save, delete, and edit) the data for the following elements:

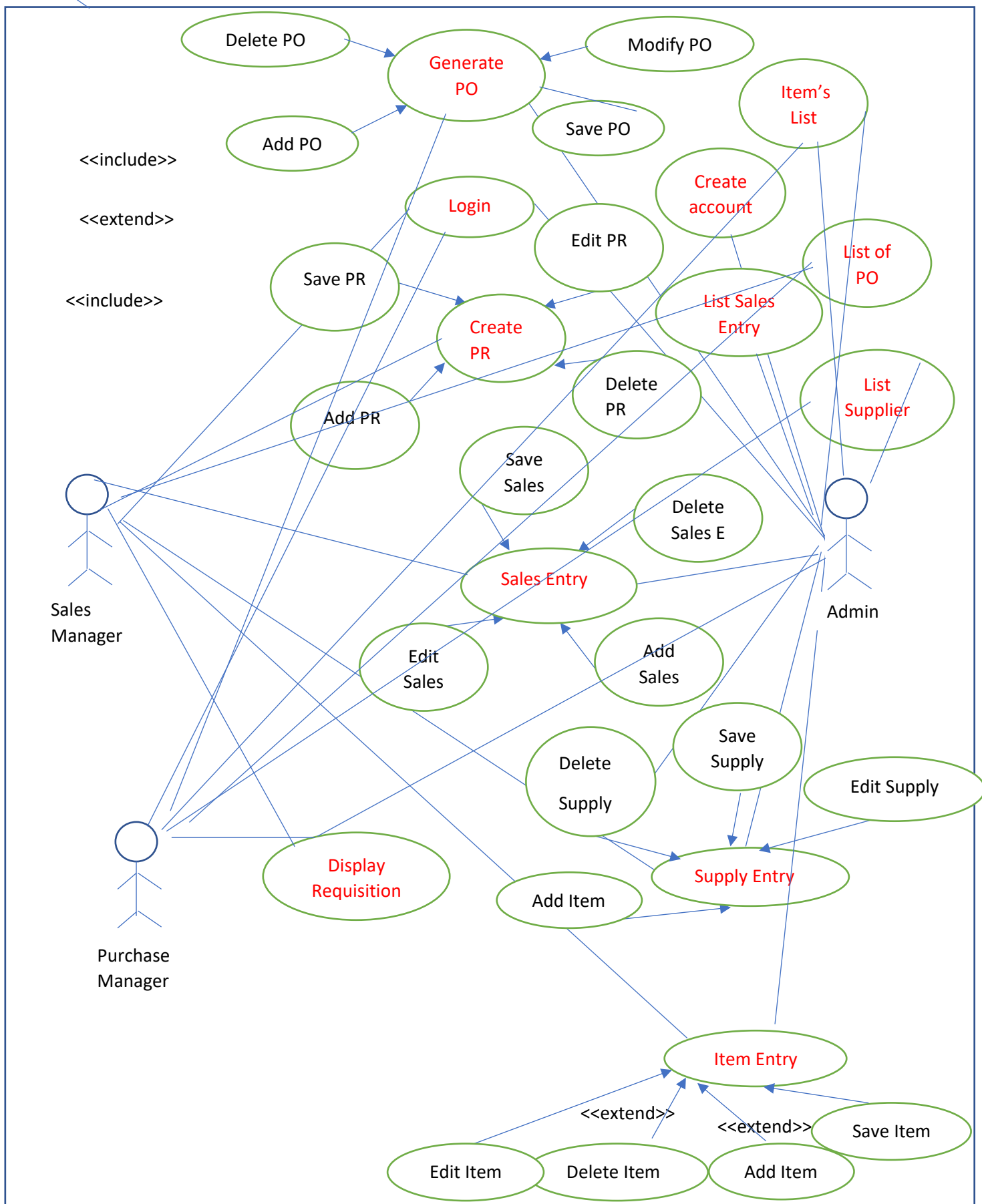
- Item Entry
- Supplier
- Daily Item-wise Sales Entry
- Create Purchase Requisition

Finally, it can view the information of the following data:

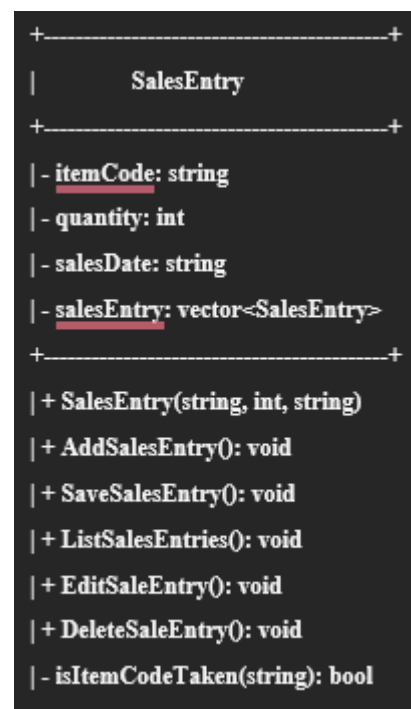
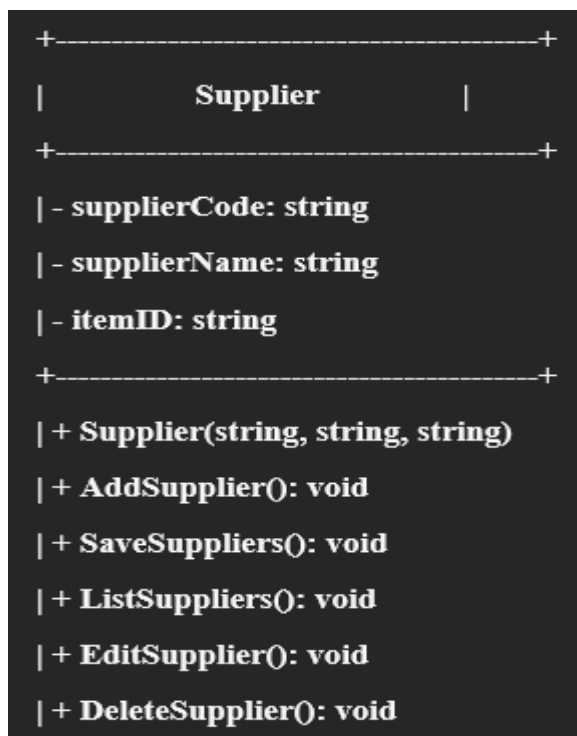
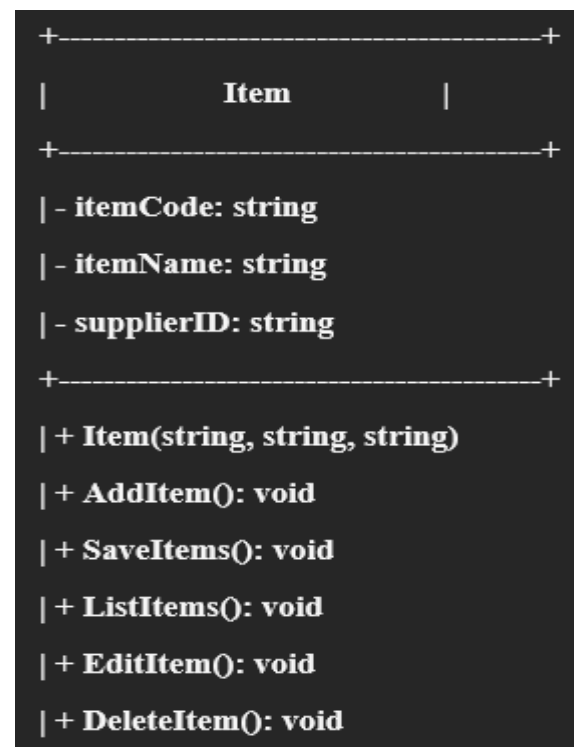
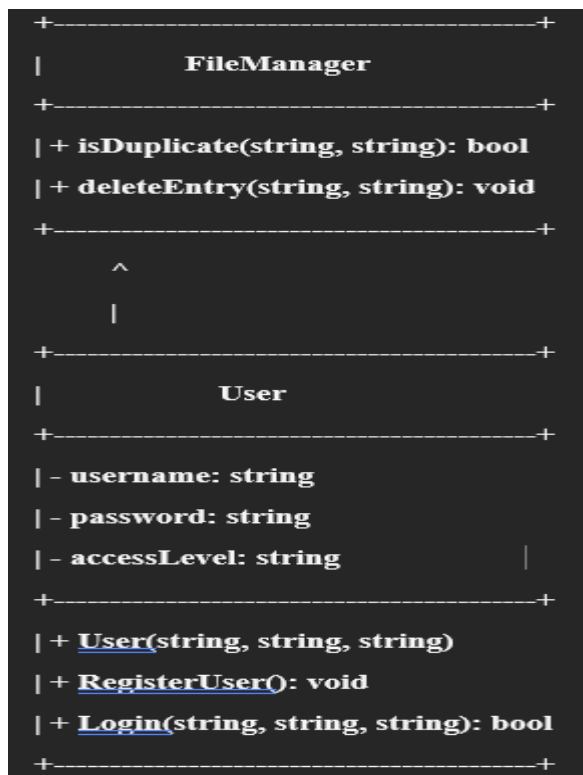
- Display Requisition
- List of Purchase Orders

Lastly, the final role is the admin who has access to all the functionalities as it can create unpracticed users in the system based on the three roles of the system.

USE CASE DIAGRAM



CLASS DIAGRAM



```

+-----+
|      PurchaseRequisition      |
+-----+
| - PRID: string                 |
| - itemCode: string             |
| - quantity: int                |
| - requiredDate: string         |
| - supplierCode: string         |
| - salesManager: string         |
| - PR: vector<PurchaseRequisition> |
+-----+
| + PurchaseRequisition(string, string, int, |
|   string, string, string)               |
| + CreatePR(): void                    |
| + SavePREntry(): void                 |
| + DisplayPRs(): void                 |
| + EditPR(): void                     |

```

```

+-----+
|      PurchaseOrder             |
+-----+
| - POID: string                 |
| - PRID: string                 |
| - purchaseManager: string       |
| - POs: vector<PurchaseOrder>    |
+-----+
| + PurchaseOrder(string, string, string) |
| + GeneratePO(): void              |
| + SavePOEntry(): void             |
| + ListPOs(): void                |
| + EditPO(): void                  |
| + DeletePO(): void                |
| - isPOIDTaken(string): bool       |
| - isPRIDValid(string): bool       |
| - isPurchaseManagerValid(string): bool |
+-----+

```

```

+-----+
|      Application                |
+-----+
| + DisplayMenu(string): void     |
| + HandleUserInput(string): void |
| + RegisterUser(): void          |
| + HandleItemMenu(): void        |
| + HandleSupplierMenu(): void    |
| + HandleSaleEntryMenu(): void   |
| + HandlePRMenu(): void          |
| + HandlePOMenu(): void          |

```

```

+-----+
|      FileManager                |
+-----+
| + isDuplicate(string, string): bool |
| + deleteEntry(string, string): void |

```

IMPLEMENTATION OF OOP CONCEPTS

There are several object-oriented programming concepts that I have implemented in my program:

- Classes and objects:

Considering or knowing that when we speak about OOP concepts there should be a presence of classes and objects as they are one of the essential characteristics of OOP concepts. As far as my program is concerned, my program is built in a way that I have created classes and objects such as Supplier, Item, SalesEntry, PurchaseRequisition, PurchaseOrder, Application, and some other relevant classes which object I have used to play or manipulate the behaviour of the attributes in some classes used in my program.

```
// User class
class User {
private:
    string username;
    string password;
    string accessLevel;

// Item class
class Item : public FileManager {
    static vector<Item> items;
protected:
    string itemCode;
    string itemName;
    string supplierID;

// Supplier class
class Supplier : public FileManager {
    static vector<Supplier> suppliers;
protected:
    string supplierCode;
    string supplierName;
    string itemID;
```

Illustration 1: examples of user, item, and supplier classes

- Inheritance:

Basically, to manipulate my data easily in the program I must implement inheritance for a better organization of my classes based on their information to save them in a text file through the use of vectors.

```
// Base Class
class FileManager {
public:
    // Check if a value already exists in a file
    bool isDuplicate(const string& value, const string& filename) {
        ifstream file(filename);
        string line;
        while (getline(file, line)) {
            istringstream iss(line);
            string fieldValue;
            iss >> fieldValue;
            if (value == fieldValue) {
                file.close();
                return true;
            }
        }
        file.close();
        return false;
    }
};
```

Illustration 2_0: Base class File Manager first part

```
// Item class
class Item : public FileManager {
    static vector<Item> items;
protected:
    string itemCode;
    string itemName;
    string supplierID;

public:
    Item(string itemCode, string itemName, string supplierID) {
        this->itemCode = itemCode;
        this->itemName = itemName;
        this->supplierID = supplierID;
    }
}
```

Illustration 2_1: class Item which inherits File Manager class.

```
// Supplier class
class Supplier : public FileManager {
    static vector<Supplier> suppliers;

protected:
    string supplierCode;
    string supplierName;
    string itemID;

public:
    Supplier(string supplierCode, string supplierName, string itemID) {
        this->supplierCode = supplierCode;
        this->supplierName = supplierName;
        this->itemID = itemID;
    }
}
```

Illustration 2_2: class Supplier inherits FileManager class.

Some other classes such as SalesEntry, PurchaseRequisition, and PurchaseOrder inherit from the base class which is FileManager. Based on the information discussed early we can say that the OOP concept of inheritance is clearly proved in the program.

- Encapsulation:

In my program, you can appreciate the implementation of an encapsulation. In my code, the data members and methods are encapsulated within classes. For example, in my code basically, the class named User is an example of encapsulation as its data member's username, password, & accessLevel are set as private and the only way that they can be accessed and changed is through the methods in this class such as RegisterUser. One of the case scenarios of encapsulation in my program is illustrated in the following illustration 3:


```
// User class
class User {
private:
    string username;
    string password;
    string accessLevel;

// Register a new user
void RegisterUser() {
    // Validate access level
    if (accessLevel != "SM" && accessLevel != "PM" && accessLevel != "Admin") {
        cout << "Error: Invalid access level. Access level should be SM/PM/Admin." << endl;
        return;
    }

    // Generate unique user ID
    string userId = generateUserId(accessLevel);

    // Save user details to file
    ofstream file("users.txt", ios::app);
    file << userId << " " << " " << password << " " << accessLevel << endl;
    file.close();
    cout << "User registration successful. User ID: " << userId << endl;
}
```

Illustration 3_0: User class with its method RegisterUser

- Data Abstraction:

In my program, I used one of the key points for OOP. Firstly, let's define what Data Abstraction is.

Firstly, if we start from abstraction, we can say that it is a way of showing only the significant or essential information of your program while hiding the details. So basically, I am using abstraction because I defined methods in my classes that perform specific tasks while hiding the implementation details from the user. Now an example of data abstraction in my code would be Item, Supplier, SalesEntry, PurchaseRequisition and User class because they mainly encapsulate the method and provide certain methods to interact with those data.

An example of data abstraction, for example, is the application class because it provides an interface to the user, hiding the implementation details. It could also be a good example of encapsulation.

```
class Application {
public:
    static void DisplayMenu(string accessLevel) {
        cout << "\n===== Main Menu =====" << endl;
        if (accessLevel == "Admin") {
            cout << "1. Item Entry (Add/Save/Edit/Delete)" << endl;
            cout << "2. Supplier Entry (Add/Save/Edit/Delete)" << endl;
            cout << "3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)" << endl;
            cout << "4. Create a Purchase Requisition (Add/Save/Edit/Delete)" << endl;
            cout << "5. Display Requisition (View)" << endl;
            cout << "6. List of Purchaser Orders (View)" << endl;
            cout << "7. Generate Purchase Order (Add/Save/Delete/Edit)" << endl;
            cout << "8. List of Items (View)" << endl;
            cout << "9. List of Suppliers (View)" << endl;
            cout << "10. List of Daily Item-wise Sales Entry (View)" << endl;
            cout << "11. User Registration" << endl;
        }
        else if (accessLevel == "SM") {
            cout << "1. Item Entry (Add/Save/Edit/Delete)" << endl;
            cout << "2. Supplier Entry (Add/Save/Edit/Delete)" << endl;
            cout << "3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)" << endl;
            cout << "4. Create a Purchase Requisition (Add/Save/Edit/Delete)" << endl;
            cout << "5. Display Requisition (View)" << endl;
            cout << "6. List of Purchaser Orders (View)" << endl;
        }
        else if (accessLevel == "PM") {
            cout << "1. List of Items (View)" << endl;
            cout << "2. List of Suppliers (View)" << endl;
            cout << "3. Display Requisition (View)" << endl;
            cout << "4. Generate Purchase Order (Add/Save/Delete/Edit)" << endl;
            cout << "5. List of Purchaser Orders (View)" << endl;
        }
    }
}
```

Illustration 4: Application class

- Polymorphism:

In my program, I have implemented a clear example of polymorphism which is called the `HandleUserInput` method inside the `Application` class. The `HandleInput` method is an example of polymorphism because it is used differently based on the role in this case the `accessLevel`, meaning if I access as an admin, I will have certain functionalities in the program while if I access for example as the sales manager then I will have different functionalities in the program. The next Illustration will reflect it:

```
static void HandleUserInput(string accessLevel) {
    string choice;
    while (true) {
        DisplayMenu(accessLevel);
        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        if (choice == "0") {
            cout << "Exiting to user access..." << endl;
            return;
        } else if (choice == "1") {
            if (accessLevel == "SM" || accessLevel == "Admin")
                HandleItemMenu();
            else if (accessLevel == "PM")
                listItems();
        } else if (choice == "2") {
            if (accessLevel == "SM" || accessLevel == "Admin")
                HandleSupplierMenu();
            else if (accessLevel == "PM")
                listSuppliers();
        } else if (choice == "3") {
            if (accessLevel == "SM" || accessLevel == "Admin")
                HandleSaleEntryMenu();
            else if (accessLevel == "PM")
                displayPRs();
        }
    }
}
```

Illustration 5: the `HandleInput` method

SCREENSHOTS OF THE OUTPUT OF THE PROGRAM

LOGIN ACCESS:

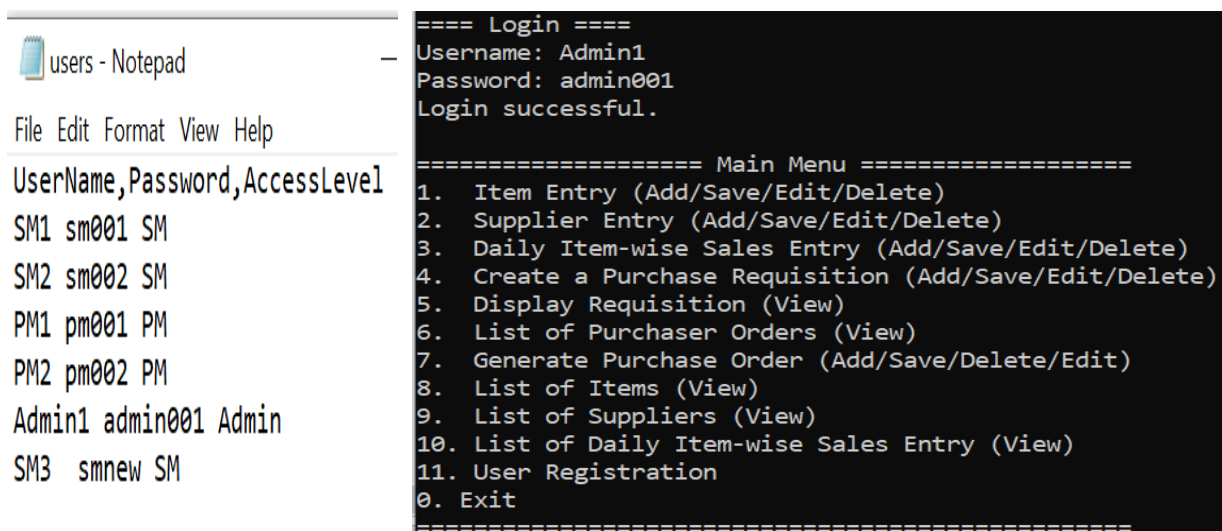
```

===== User Access =====
1. Admin
2. Sales Manager (SM)
3. Purchase Manager (PM)
0. Exit
=====
Enter your choice: _

```

Illustration 6: login page

Once I run the program the first output that will pop up in the screen is the user access with the three main roles or actor which are admin, sales manager, and purchase manager.



The screenshot shows a Notepad window titled 'users - Notepad' with a list of users and their access levels. The list is as follows:

UserName	Password	AccessLevel
SM1	sm001	SM
SM2	sm002	SM
PM1	pm001	PM
PM2	pm002	PM
Admin1	admin001	Admin
SM3	smnew	SM

Next to the Notepad window is a terminal window showing the login process for an admin user. The terminal output is as follows:

```

==== Login ====
Username: Admin1
Password: admin001
Login successful.

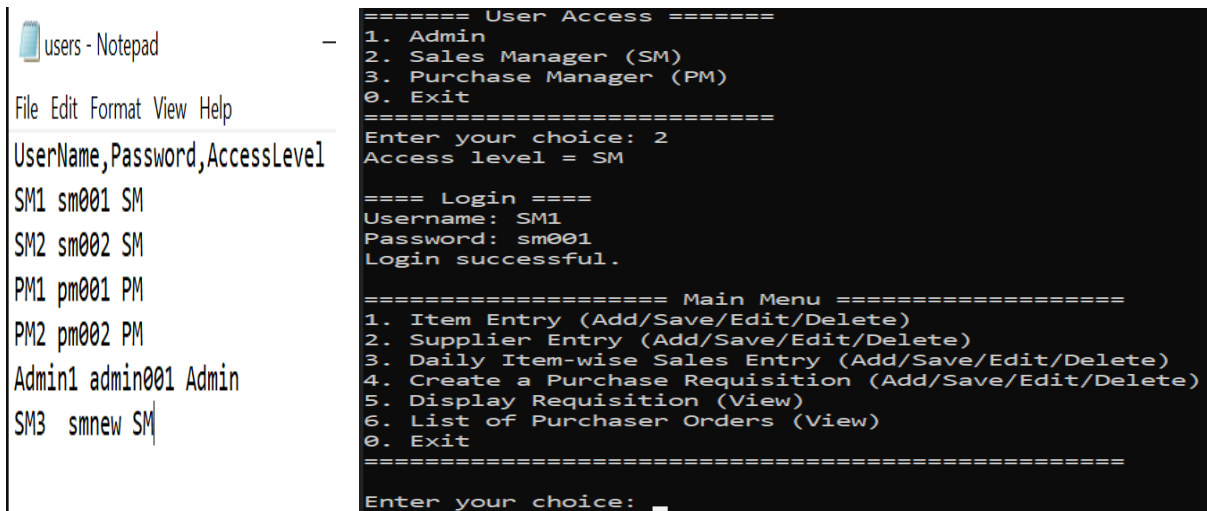
===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
7. Generate Purchase Order (Add/Save/Delete/Edit)
8. List of Items (View)
9. List of Suppliers (View)
10. List of Daily Item-wise Sales Entry (View)
11. User Registration
0. Exit
=====

```

Illustration 7: Login page for admin

So basically, I have logged in as admin and once I press the correct credential it is going to display to me all the functionalities for admin as well as user registration as required for the assignment and at the end, I have the option to exit the program as well.

Now, If I choose option 2 it is going to display the access level as sales manager, then it might ask me for the credentials and then after that, if the password and username are correct then the system will only show me the functionalities of sales manager as you can see in the illustration 8.



The image shows a Notepad window on the left with a list of users and a terminal window on the right showing the login process for a sales manager.

Notepad Window:

```

users - Notepad
File Edit Format View Help
UserName,Password,AccessLevel
SM1 sm001 SM
SM2 sm002 SM
PM1 pm001 PM
PM2 pm002 PM
Admin1 admin001 Admin
SM3 smnew SM

```

Terminal Window:

```

===== User Access =====
1. Admin
2. Sales Manager (SM)
3. Purchase Manager (PM)
0. Exit
=====
Enter your choice: 2
Access level = SM

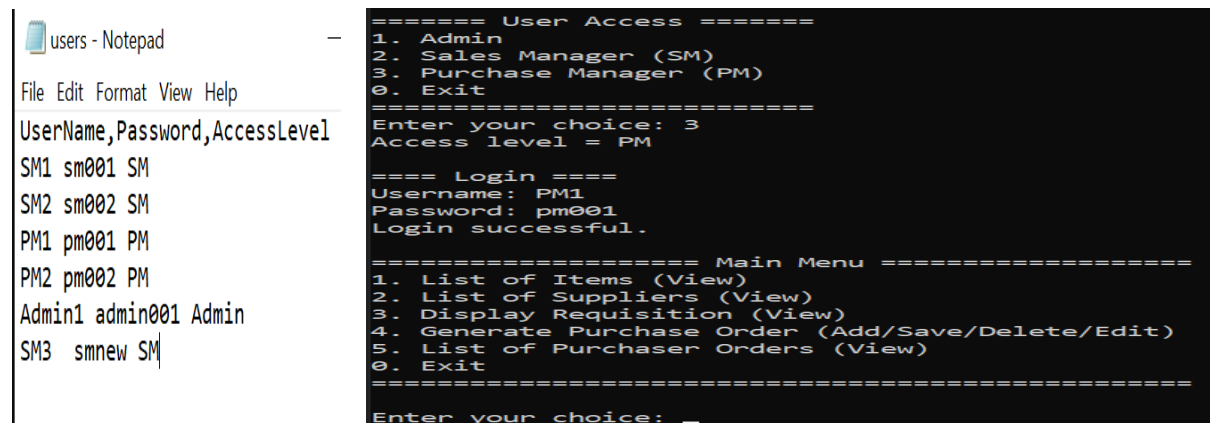
==== Login ====
Username: SM1
Password: sm001
Login successful.

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
0. Exit
=====
Enter your choice: _

```

Illustration 8: login for sales manager

In the same way, it will happen with the purchase manager access level or user, and you can see it in illustration 9:



The image shows a Notepad window on the left with a list of users and a terminal window on the right showing the login process for a purchase manager.

Notepad Window:

```

users - Notepad
File Edit Format View Help
UserName,Password,AccessLevel
SM1 sm001 SM
SM2 sm002 SM
PM1 pm001 PM
PM2 pm002 PM
Admin1 admin001 Admin
SM3 smnew SM

```

Terminal Window:

```

===== User Access =====
1. Admin
2. Sales Manager (SM)
3. Purchase Manager (PM)
0. Exit
=====
Enter your choice: 3
Access level = PM

==== Login ====
Username: PM1
Password: pm001
Login successful.

===== Main Menu =====
1. List of Items (View)
2. List of Suppliers (View)
3. Display Requisition (View)
4. Generate Purchase Order (Add/Save/Delete/Edit)
5. List of Purchaser Orders (View)
0. Exit
=====
Enter your choice: _

```

Illustration 9: login for purchase manager

USER REGISTRATION

For registration basically the administrator is the only user that have the functionality of creating new users in the system based on their access level, meaning whether the user is PM or SM.

```

===== Login =====
Username: Admin1
Password: admin001
Login successful.

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
7. Generate Purchase Order (Add/Save/Delete/Edit)
8. List of Items (View)
9. List of Suppliers (View)
10. List of Daily Item-wise Sales Entry (View)
11. User Registration
0. Exit

Enter your choice: 11
Enter access level (SM/PM/Admin): PM
Enter password: PM001
User registration successful. User ID: PM3

```

Illustration 10: registration for PM

The next illustration 11 is when the admin creates a new user as Sales Manager.

```

Enter your choice: 11
Enter access level (SM/PM/Admin): SM
Enter password: SM003
User registration successful. User ID: SM4

----- Main Menu -----

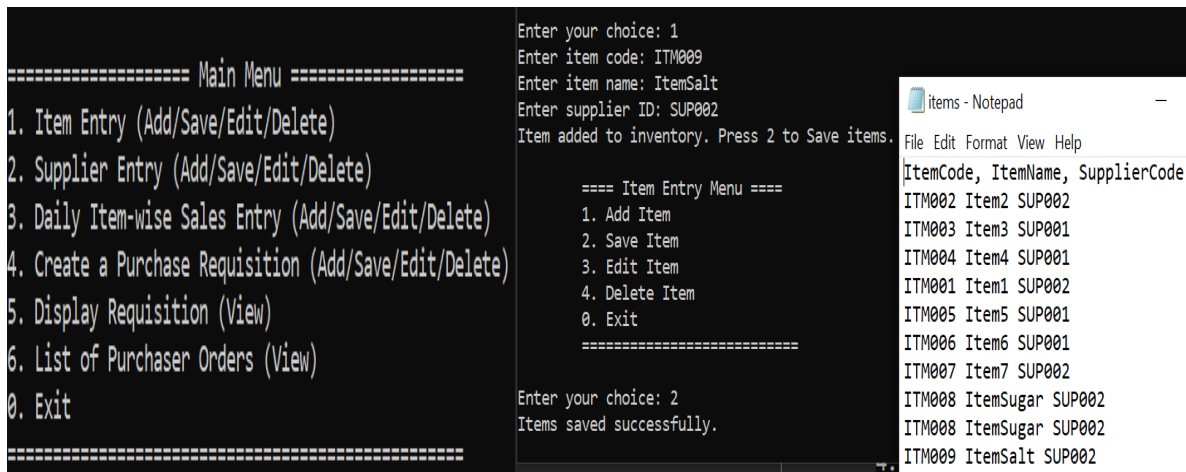
```

Illustration 12: registration for SM

Now, there is something that you must notice in the program. When the admin creates a new user, then it will automatically assign that user a unique ID which is going to be stored in a file called users. Basically, the unique ID for the users in the system is set as the username which is going to go through the first line of the file and then it will generate a unique ID by a sum or increment of 1 which is going to make the users have different username and a unique one. I set the userID feature in my program for username.

ITEM ENTRY AND SUPPLY ENTRY

For this section we must highlight that the users that got access to them are the three main users in the program. Let us try to access into it as the sales manager:



```

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
0. Exit

Enter your choice: 1
Enter item code: ITM009
Enter item name: ItemSalt
Enter supplier ID: SUP002
Item added to inventory. Press 2 to Save items.

===== Item Entry Menu =====
1. Add Item
2. Save Item
3. Edit Item
4. Delete Item
0. Exit

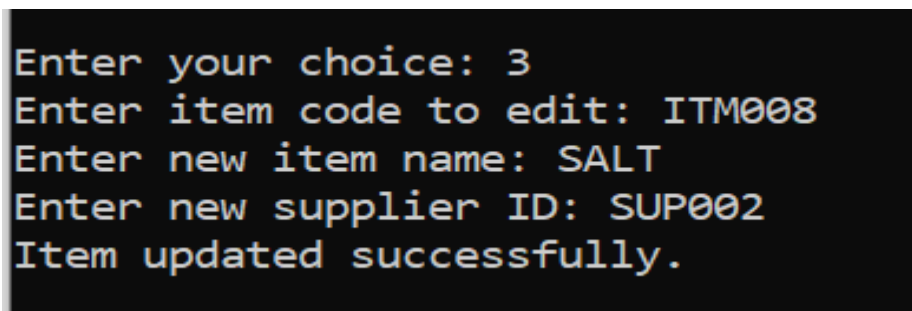
Enter your choice: 2
Items saved successfully.

```

items - Notepad

ItemCode	ItemName	SupplierCode
ITM002	Item2	SUP002
ITM003	Item3	SUP001
ITM004	Item4	SUP001
ITM001	Item1	SUP002
ITM005	Item5	SUP001
ITM006	Item6	SUP001
ITM007	Item7	SUP002
ITM008	ItemSugar	SUP002
ITM008	ItemSugar	SUP002
ITM009	ItemSalt	SUP002

Illustration 12: Functionalities of the Sales Manager, Item Entry (Add and Save)



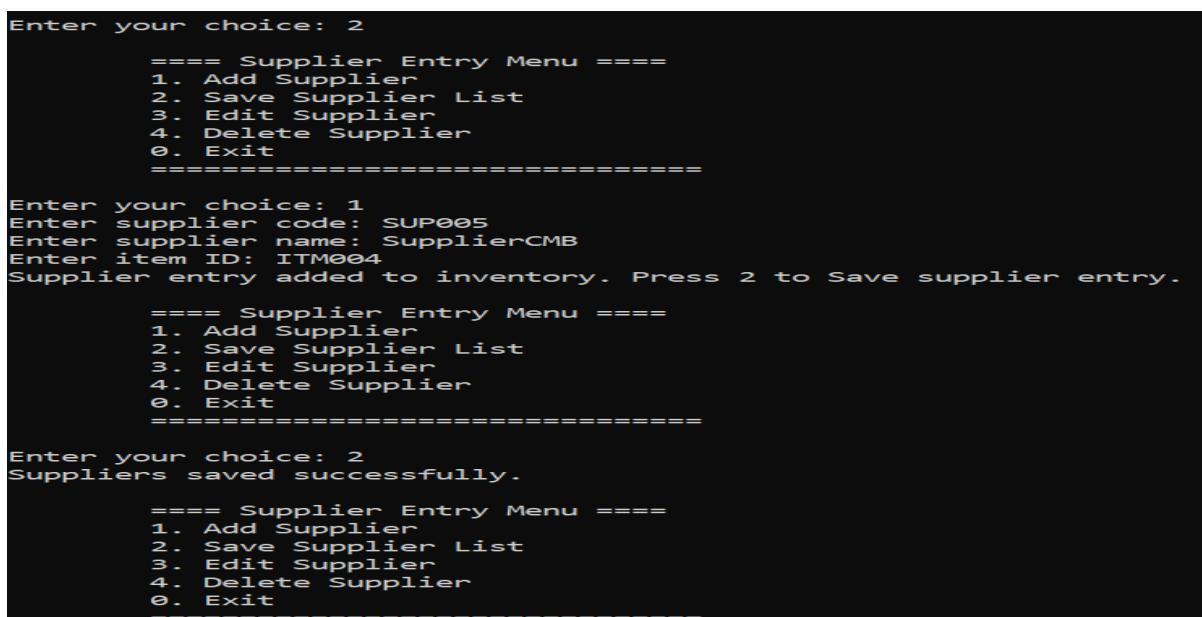
```

Enter your choice: 3
Enter item code to edit: ITM008
Enter new item name: SALT
Enter new supplier ID: SUP002
Item updated successfully.

```

Illustration 13: Functionalities of the Sales Manager, Item Entry (editing Item)

Here is the representation of supply entry for sales manager:



```

Enter your choice: 2

===== Supplier Entry Menu =====
1. Add Supplier
2. Save Supplier List
3. Edit Supplier
4. Delete Supplier
0. Exit

Enter your choice: 1
Enter supplier code: SUP005
Enter supplier name: SupplierCMB
Enter item ID: ITM004
Supplier entry added to inventory. Press 2 to Save supplier entry.

===== Supplier Entry Menu =====
1. Add Supplier
2. Save Supplier List
3. Edit Supplier
4. Delete Supplier
0. Exit

Enter your choice: 2
Suppliers saved successfully.

===== Supplier Entry Menu =====
1. Add Supplier
2. Save Supplier List
3. Edit Supplier
4. Delete Supplier
0. Exit

```

Illustration 14: Functionalities of the Sales Manager, Supply Entry

Now, I am going to show the illustration of the supply entry from Sales Manager

```

===== Sale Entry Menu =====
1. Add Sale Entry
2. Save Sale Entry
3. Edit Sale Entry
4. Delete Sale Entry
0. Exit
=====

Enter your choice: 1
Enter item code: ITM008
Enter quantity: 10
Enter sales date: 2023-06-09
Sales entry added to inventory. Press 2 to Save sales entry.

===== Sale Entry Menu =====
1. Add Sale Entry
2. Save Sale Entry
3. Edit Sale Entry
4. Delete Sale Entry
0. Exit
=====

Enter your choice: 2
Sales Entry saved successfully.

===== Sale Entry Menu =====
1. Add Sale Entry
2. Save Sale Entry
3. Edit Sale Entry
4. Delete Sale Entry
0. Exit
=====

Enter your choice: 3
Enter item code of the sale entry to edit: ITM004
Enter new sales date: 3
Enter new quantity:
5
Sale entry updated successfully.

```

Illustration 14: Operations as Sales Manager for Sale Entry

Now, the next functionality for the Sales Manager is create a purchase requisition which is going to be represented in the next illustration:

```

Enter your choice: 1
Enter PR ID: PR004
Enter item code: ITM006
Enter quantity: 17
Enter required date (YYYY-MM-DD): 2023-09-06
Enter supplier code: SUPP001
Enter sales manager: SM3
Purchase requisition added to inventory. Press 2 to Save purchase requisition.

===== Purchase Requisition Menu =====
1. Create Purchase Requisition
2. Save Purchase Requisitions
3. Edit Purchase Requisition
4. Delete Purchase Requisition
0. Exit
=====

Enter your choice: 2
Purchase requisitions saved successfully.

===== Purchase Requisition Menu =====
1. Create Purchase Requisition
2. Save Purchase Requisitions
3. Edit Purchase Requisition
4. Delete Purchase Requisition
0. Exit
=====

```

Illustration 15: Sales Manager operations for Purchase Requisition

```

===== Purchase Requisition Menu =====
1. Create Purchase Requisition
2. Save Purchase Requisitions
3. Edit Purchase Requisition
4. Delete Purchase Requisition
0. Exit
=====

Enter your choice: 3
Enter PRID of the purchase requisition to edit: PR002
Enter new item code: ITM004
Enter new quantity: 8
Enter new required date (YYYY-MM-DD): 2023-06-01
Enter new supplier code: SUP004
Enter new sales manager: SM4
Purchase requisition updated successfully.

===== Purchase Requisition Menu =====
1. Create Purchase Requisition
2. Save Purchase Requisitions
3. Edit Purchase Requisition
4. Delete Purchase Requisition
0. Exit
=====

Enter your choice: 4
Enter PRID of the purchase requisition to delete: PR002
Entry deleted successfully.

```

Illustration 16: Purchase Requisition for Sales Manager edit and delete.

Now I am going to show other functionalities for Sales Managers such as Display Purchase Requisition:

```

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
0. Exit
=====

Enter your choice: 5
PR003 ITM005 23 2023-32-22 SUP001 SM002
PR004 ITM006 17 2023-09-06 SUPP001 SM3

```

Illustration 17: Display Requisition for Sales Manager

Finally, the last session for the Sales Manager is to show the list of Purchase Orders available in the system.

```

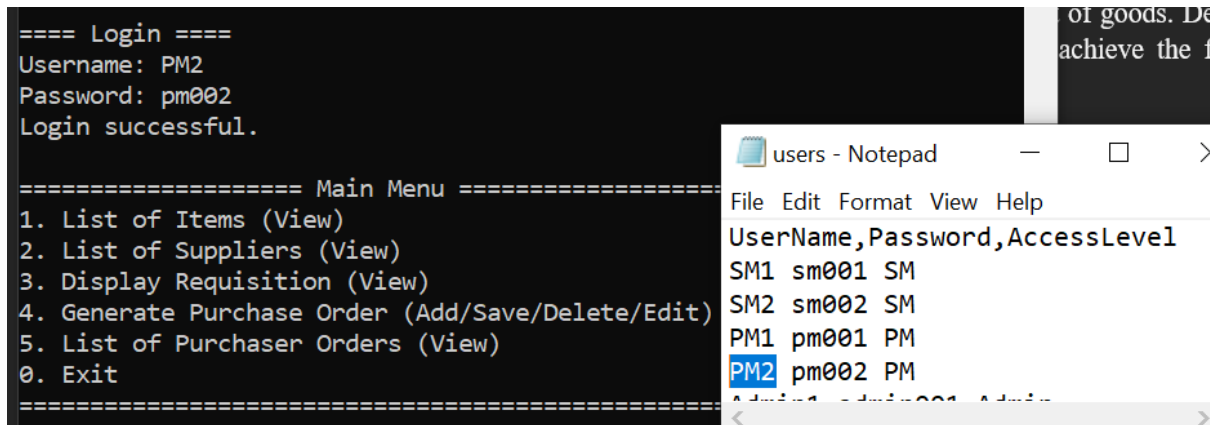
Enter your choice: 6
PO001 PR001 PM1
PO002 PR002 PM2
PO003 PR003 PM1
PO004 PR004 PM2
PO005 PR005 PM1

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
0. Exit

```

Illustration 18: Display list of Purchase Oder

Now, the functionalities of the Purchase Manager are as follows:



```

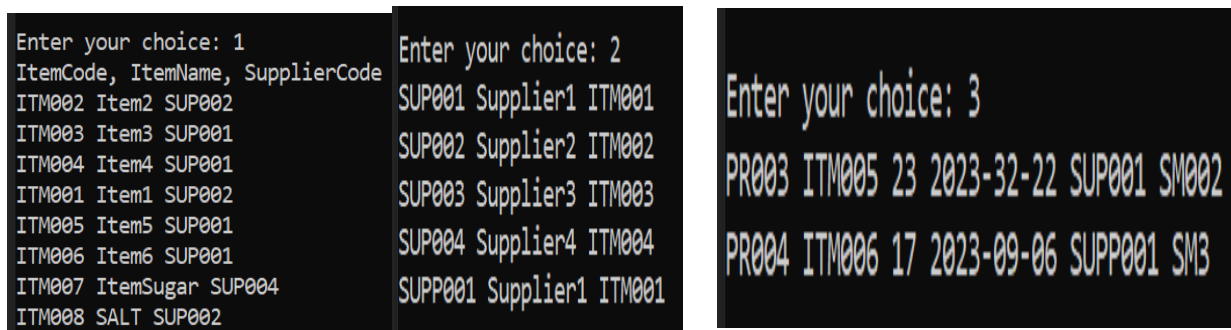
==== Login ====
Username: PM2
Password: pm002
Login successful.

===== Main Menu =====
1. List of Items (View)
2. List of Suppliers (View)
3. Display Requisition (View)
4. Generate Purchase Order (Add/Save/Delete/Edit)
5. List of Purchaser Orders (View)
0. Exit
=====

users - Notepad
File Edit Format View Help
UserName,Password,AccessLevel
SM1 sm001 SM
SM2 sm002 SM
PM1 pm001 PM
PM2 pm002 PM
Admin1 admin001 Admin

```

Illustration 19: Purchase Manager Menu



```

Enter your choice: 1
ItemCode, ItemName, SupplierCode
ITM002 Item2 SUP002
ITM003 Item3 SUP001
ITM004 Item4 SUP001
ITM001 Item1 SUP002
ITM005 Item5 SUP001
ITM006 Item6 SUP001
ITM007 ItemSugar SUP004
ITM008 SALT SUP002

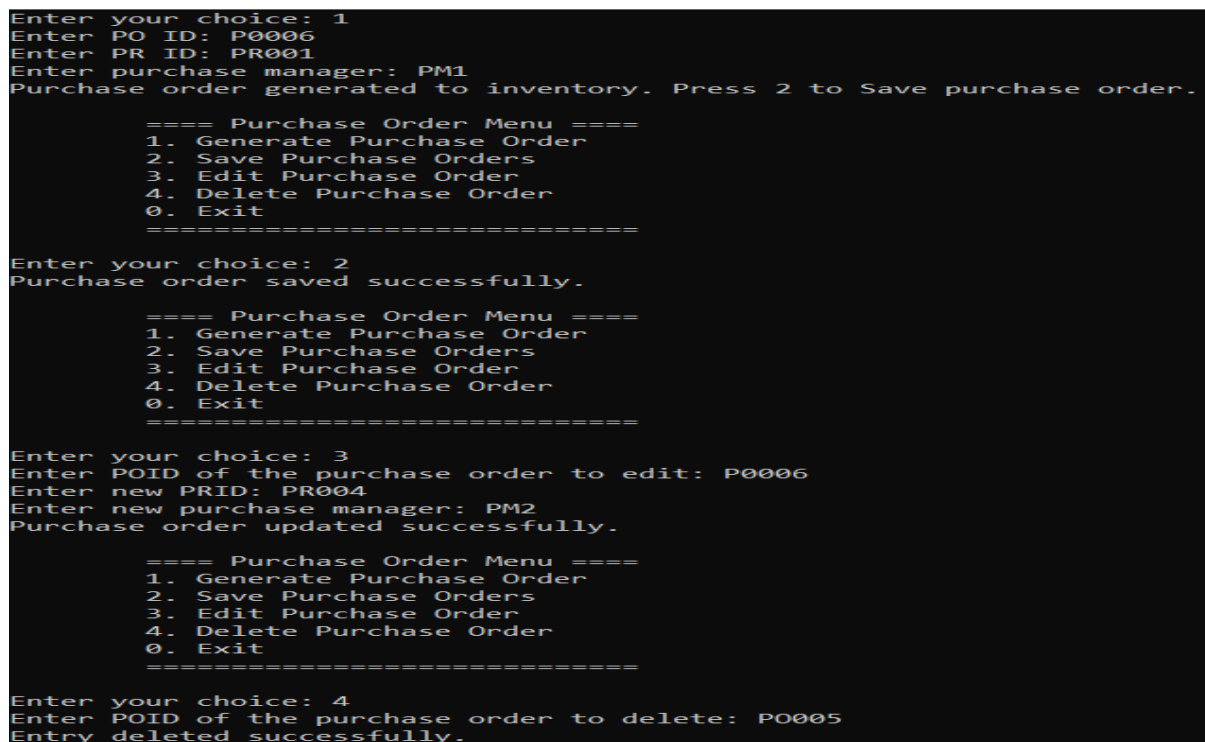
Enter your choice: 2
SUP001 Supplier1 ITM001
SUP002 Supplier2 ITM002
SUP003 Supplier3 ITM003
SUP004 Supplier4 ITM004
SUPP001 Supplier1 ITM001

Enter your choice: 3
PR003 ITM005 23 2023-32-22 SUP001 SM002
PR004 ITM006 17 2023-09-06 SUPP001 SM3

```

Illustration 20: Purchase Manager for list of items, suppliers, display requisition

Now for generating a purchase order, I have set in my program that the purchase manager is the one who is the one that is going to manipulate the data from it.



```

Enter your choice: 1
Enter PO ID: P0006
Enter PR ID: PR001
Enter purchase manager: PM1
Purchase order generated to inventory. Press 2 to Save purchase order.

===== Purchase Order Menu =====
1. Generate Purchase Order
2. Save Purchase Orders
3. Edit Purchase Order
4. Delete Purchase Order
0. Exit
=====

Enter your choice: 2
Purchase order saved successfully.

===== Purchase Order Menu =====
1. Generate Purchase Order
2. Save Purchase Orders
3. Edit Purchase Order
4. Delete Purchase Order
0. Exit
=====

Enter your choice: 3
Enter POID of the purchase order to edit: P0006
Enter new PRID: PR004
Enter new purchase manager: PM2
Purchase order updated successfully.

===== Purchase Order Menu =====
1. Generate Purchase Order
2. Save Purchase Orders
3. Edit Purchase Order
4. Delete Purchase Order
0. Exit
=====

Enter your choice: 4
Enter POID of the purchase order to delete: P0005
Entry deleted successfully.

```

Illustration 21: Purchase Manager functionality for generating a purchase order.

In the end, the purchase manager has the option to display the list of orders.

```

Enter your choice: 5
PO001 PR001 PM1
PO002 PR002 PM2
PO003 PR003 PM1
PO004 PR004 PM2
P0006 PR004 PM2

```

Illustration 22: Purchase Manager functionality for displaying the list of orders.

Finally, we have a presence of the admin in the program which got all the functionalities in the system and can do any task as well as create new users and add them to the system.

```

===== User Access =====
1. Admin
2. Sales Manager (SM)
3. Purchase Manager (PM)
0. Exit
=====
Enter your choice: 1
Access level = Admin

==== Login ====
Username: Admin1
Password: admin001
Login successful.

===== Main Menu =====
1. Item Entry (Add/Save/Edit/Delete)
2. Supplier Entry (Add/Save/Edit/Delete)
3. Daily Item-wise Sales Entry (Add/Save/Edit/Delete)
4. Create a Purchase Requisition (Add/Save/Edit/Delete)
5. Display Requisition (View)
6. List of Purchaser Orders (View)
7. Generate Purchase Order (Add/Save/Delete/Edit)
8. List of Items (View)
9. List of Suppliers (View)
10. List of Daily Item-wise Sales Entry (View)
11. User Registration
0. Exit
=====
Enter your choice:

```

users - Notepad

File Edit Format View Help

UserName,Password,AccessLevel

SM1	sm001	SM
SM2	sm002	SM
PM1	pm001	PM
PM2	pm002	PM
Admin1	admin001	Admin
SM3	smnew	SM
PM3	PM001	PM
SM4	SM003	SM

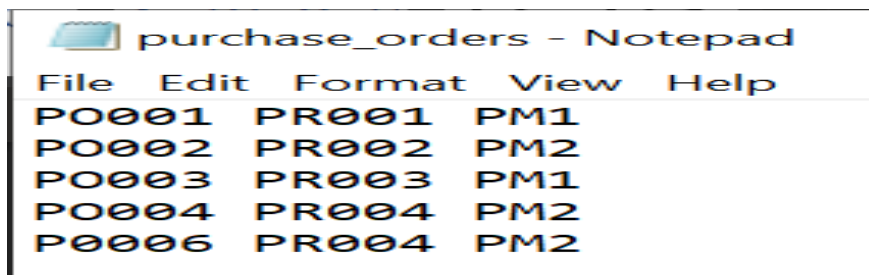
100% Windows (CRLF) UTF-8

Illustration: Administrator functionalities

PURCHASE MANAGEMENT FILES:

Five files have been used in this project and they are the following:

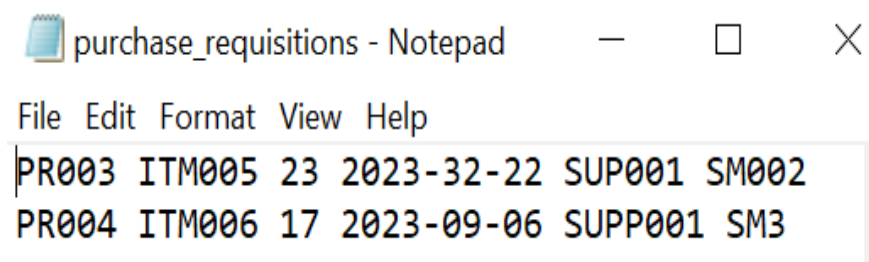
- Purchase requisition
- Sales
- Suppliers
- items
- Suppliers
- Users



```

purchase_orders - Notepad
File Edit Format View Help
PO001 PR001 PM1
PO002 PR002 PM2
PO003 PR003 PM1
PO004 PR004 PM2
P0006 PR004 PM2
  
```

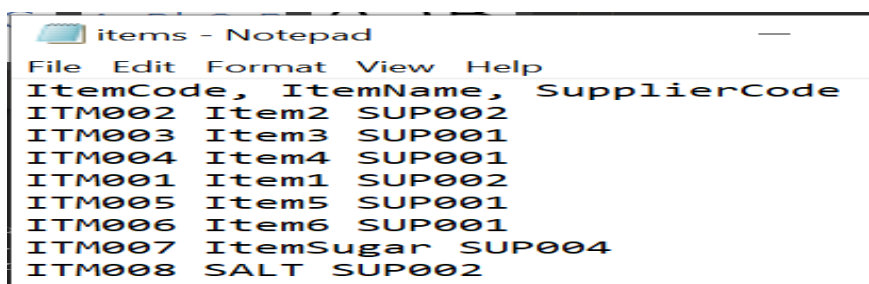
- Purchase orders



```

purchase_requisitions - Notepad
File Edit Format View Help
PR003 ITM005 23 2023-32-22 SUP001 SM002
PR004 ITM006 17 2023-09-06 SUPP001 SM3
  
```


- Purchase requisition



```


items - Notepad
File Edit Format View Help
ItemCode, ItemName, SupplierCode
ITM002 Item2 SUP002
ITM003 Item3 SUP001
ITM004 Item4 SUP001
ITM001 Item1 SUP002
ITM005 Item5 SUP001
ITM006 Item6 SUP001
ITM007 ItemSugar SUP004
ITM008 SALT SUP002
  
```

- items

 suppliers - Notepad

File	Edit	Format	View	Help
SUP001	Supplier1	ITM001		
SUP002	Supplier2	ITM002		
SUP003	Supplier3	ITM003		
SUP004	Supplier4	ITM004		
SUPP001	Supplier1	ITM001		

- Suppliers

 users - Notepad

File	Edit	Format	View	Help
UserName,Password,AccessLevel				
SM1	sm001	SM		
SM2	sm002	SM		
PM1	pm001	PM		
PM2	pm002	PM		
Admin1	admin001	Admin		
SM3	smnew	SM		
PM3	PM001	PM		
SM4	SM003	SM		

- Users

CONCLUSION

Finally, at this stage, I have completed the assignment and fulfilled all the criteria. I have implemented object-oriented programming (OOP) concepts such as polymorphism, data abstraction, inheritance, encapsulation and the use of classes and objects.

REFERENCES

- https://profile.w3schools.com/refresh-session?redirect_url=https%3A%2F%2Fwww.w3schools.com%2Fjava%2Fdefault.asp
W3Schools C++
- <https://www.geeksforgeeks.org/different-ways-reading-text-file-java/>
Different ways of reading text file C++, 20 December 2021
- <https://youtu.be/A74TOX803D0>
Java Tutorial, freeCodeCamp.org, YouTube