**PROJECT REPORT**

## Data Structure & Algorithms



**BS (CS)-3(B)**

**Project Title: INVENTORY MANAGEMENT SYSTEM**

*Group Members*

|  |  |
| --- | --- |
| **Name** | **Enrollment** |
| 1. ***Emad Tariq*** | 02-13221-068 |
| 2. ***Ammarah Ali*** | 02-134221-077 |
| 3. ***Ahsan Naeem*** | 02-134221-082 |
| 4. **Soha Ather** | 02-134221-088 |

# Submitted to:

# Miss TOOBA ZAHID

**BAHRIA UNIVERSITY KARACHI CAMPUS**

## Department of Computer Science

* **Contents:-**
* Abstract.
* Introduction
* Background
* Problem statement
* Proposed system
* Scope Of Project
* Modules In Project
* Flow diagram of the system
* Screen shots of module with explanation
* Code
* Future work
* Conclusion
* **Abstract:-**

According to our course Data Structures And Algorithms offered

by our institution, we student of Bahria University are able and given a chance to optimize the concept of Data Structures and implementation of it by performing it on Microsoft Visual Studio a C++ Compiler.

The proposed project is the console-based C++ implementation of a management system for a small-scaled store experiencing inward flow of stock/goods) as well as outward flow of stock/goods Apart from sales and purchases, the project respective personnel tasked to handle the inventory can keep track of it by using and accessing this simple inventory system when required. This means that they can keep count, mark, record items/goods, which been purchased or; how many items have been moved for storing in the inventory at the end of each day. The task is automated to some extent also helps in saving valuable man hours. Also increases efficiency and accuracy. Plus, it also reduces expenses made on staff wages.

This project aims to provide a solution to the common issues faced in managing such store before. For example in the past, the manual system employed by organizations usually involved the physical movement of their audit staff from office to office manually counting the assets residing in each office before entering them into the spreadsheet.. These methods wasted time, were prone to errors and were also very labor intensive. Through development of this C++ application, which keeps track of inventory in the “back of house” and updates it according to the daily purchases made and daily sales made, kind of issues, like the one discussed before will now be least of worries for many such businesses. Further, this store system can also be integrated and linked with a separate database system. And also linked with future A.I. technological systems as well as help in automating, digitalizing and making businesses smart and advanced.

* **Introduction:-**

The Inventory Management System helps us to record and organize the items in more efficient & convenient way. This project aimed to provide a solution to the common issues faced in managing Inventory and Finances by a business of such caliber. For this purpose, our C++ console-based application coupled with CRUD-operations (i.e. Create, Retrieve, Update and Delete operations) keeps track of inventory levels, keeps tabs on key financial parameters of business and promises to improve/ease the customer experience. The scope of our Inventory Management System can cover many needs that includes valuing the inventory, measuring the change in inventory, it also covers the finances that happened in front-end as well as back-end. The value of the Inventory at the end of each period provides a basis for financial reporting on the balance sheet. Measuring the change in inventory allows the company to determine the cost of inventory sold/purchased during a given period.

As we know that a Inventory Management System is used for keeping a detailed record of each new item as it enters the warehouse and the record of item as the item has been sold out. The user will input the detail of the items accordingly and when some item is sold out, the quantity of item that has been sold out will be deducted from inventory. Likewise for any exchange the business goes through when dealing with a customer. So basically, it can be said that the proposed system is an implementation of a Departmental Store Management System for a generic Store experiencing inward and outward flow of stock/goods/items/exchange.

* **Background:-**

In the past, while managing a Inventory Management System it was a lot of work to arrange records of everything like the inward and outward of an item from inventory as well as recording details on services provided to a customer manually. Because recording such inward and outward trade dealings manually usually resulted in human error, unnecessary waste of time and energy, hard labor work, etc. All these factors would damp the overall performance and would also often put it at high risk of incurring losses causing a business to struggle for survival or in many cases, liquidate or cease to exist.

* **Problem Statement:-**

The traditional method(s) employed by companies/businesses usually involve the physical staff/employee engagement in managing, recording and preparing document work for daily activities of the business as well as handling of customers. This method wasted time, was prone to errors and was very labor intensive.

* **Proposed System:-**

Our proposed Inventory Management System can cover many needs. As mentioned earlier like valuing of inventory, managing the inventory and planning for future upon inspection of live inventory levels. Also, the valuing the inventory at the end of each period (daily in our case) provides integral basis for financial reporting on the balance sheet. Further, managing sales to determine the cost of inventory sold during a given period. Moreover, due to convenient and quick, efficiency in providing customer service, our store is capable fast, precise and error free and the staff will also give the receipt to the customer and the store can easily keep record of their sale.

The proposed system also aims to fulfill various goals and objectives. One objective includes maximizing efficiency and accuracy in management of the finances and economics for the store via use of the programming.

The following objective discusses back-end inventory management of the store’s stock data. The idea is to maintain inventory at appropriate level to avoid excessive or shortage of inventory because both the cases are undesirable for business. Thus, to keep inventory at sufficiently high level to perform production and sales activities smoothly. And eventually when losses are minimized as well as investments are well-controlled, a business maximizes profitability which is essential for long-term and smooth-running of a business.

Stated below are multiple features of our proposed system:

* An inventory management system which is flexible and modular.
* Masked (password characters appear in asterisk) Password/Key-code for limited access to authorized personnel only.
* A user-friendly C++ console-based program with an interactive menu.
* User may add product/update product/search product/view details on products stored in inventory.
* User may also calculate some finances of the inventory.
* Details include unit cost, quantity, name and unique ID assigned to item/product/good.
* Respective personnel can view some other details on finances of the store as well.
* Option given to store important data regarding trading and commerce of our Generic Departmental Store in text files at back-end on a hard drive (i.e. inventory.txt, finances.txt).
* **Scope Of Project:-**

The scope of our project is to make our program user friendly and as the input of items is done by the user, it can be applicable to almost every kind of store not only to a small-scale and with minor changes can even be applied to other stores such as general stores, garments store or basically anywhere in which data entry of goods and stalk management is involved.

## The scope also includes covering many needs, including valuing the inventory and expenses, measuring the change in inventory and expenses and also for planning future inventory levels and expenditures. The value of the inventory at the end of each period and expenditure done provides a basis for financial reporting on the balance sheet. Measuring the change in inventory allows the company to determine the cost of inventory sold/purchased (depends on the nature of inventory) during a given period. This allows the company to plan for future inventory needs. Likewise for income and sales.

Another major scope is that the program has the capability to be modified, enhanced, and improved by adding some more features like delete/removal of items/increase storage of elements in the array depending on the nature, type and size of the business. Also, it can be linked with a separate database network system if present in the business and if we want to and a check-in system deployed in a warehouse. This program can also be assimilated with modern day A.I. technology by making suitable and required adjustments.

* **Modules In Project:-**
* struct node
* void insertItem()
* void updateItem(int iD)
* void deleteItem()
* void displayInventory()
* void fileInventory()
* void inventory()
* void addT()
* void updateT(int tID)
* void printTransactions()
* void fileTransactions()
* void fileFinances()
* void finance()
* bool takePasswdFromUser(char sp = '\*')
* int main()
* **Flowchart diagram:**



* **Screen shots of Module with Explanation:-**
* **Masked Password for Login and store control menu:**

Below shows input of password that is masked by asterisks. This ensures password safety/security at time of input and feature of login password narrows access to an authorized user only. On entry of correct password, program transfers control to the Store’ main control menu.

**Text

Description automatically generated**

* **Adding Items to Inventory:**

After choosing option 1 for inventory system, program takes us to another sub-menu where we are given further options to perform operations on the inventory of store. Here, we begin by adding products to our inventory.

Text

Description automatically generated

**Text

Description automatically generated**

* **Display live state of Inventory:**

After adding products to our inventory, program takes us back to our sub-menu where we select the next option we desire to try out. Here, we select display option 4 and print details of products stored in our inventory via use of single linked list in code.

* **Update item:**

**Text

Description automatically generated**After display, we chose to select update option 2. This operation allows us to make changes to details of information fed for any product held in inventory by use. This is useful if we make any mistake in data entry and we need to correct it.**Text

Description automatically generatedText

Description automatically generated**

* **Delete/Remove Item:**

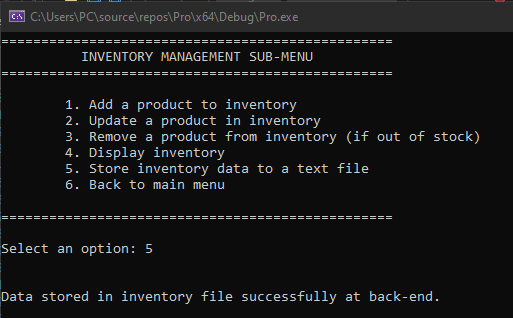
After update, we chose to select remove option 3. This operation allows us to check and remove products/items in our inventory whose quantity becomes 0 (i.e. is out of stock). This option adds special nuance to our inventory control system.

Text

Description automatically generated

* **Outlook of Inventory data stored in text file:**

After update, we chose to select option 5. This option creates a text file at back-end and enables storage of details and important data in a hard-drive.

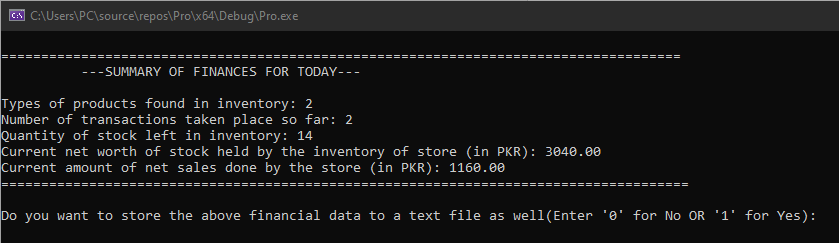


Text

Description automatically generated

* **Showing Finances of business:**

After this, program again takes us back to our sub-menu where we select the next option we desire to try out. Here, we choose to navigate back to the main menu and select option 3. This prints all the essential financial and commerce info that have taken so far in the running of the business.



* **Showing Finances of business stored in text file:**

Text

Description automatically generated

* **Code:-**

#include <iostream>

#include <conio.h>

#include <iomanip>

#include <string>

#include <fstream>

using namespace std;

struct node {

int ID, quantity;

double unitCost;

string name;

struct node\* next;

}\*start;

struct customerNode {

int transNo;

string name\_;

double bill;

struct customerNode\* fwd;

}\*top;

void insertItem() {

int id, quant;

double ucost;

string nme;

if (start == NULL)

{

node\* temp;

temp = new node();

cout << "\nEnter Product ID (Note: Unchangeable after entry so input carefully): "; cin >> id;

cin.ignore();

cout << "\nEnter Product Name: ";

getline(cin, nme);

cout << "\nEnter Product Quantity: "; cin >> quant;

cout << "\nEnter Product Unit Price (in PKR): "; cin >> ucost;

temp->ID = id;

temp->quantity = quant;

temp->unitCost = ucost;

temp->name.assign(nme);

temp->next = NULL;

start = temp;

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

}

else

{

cout << "\nEnter Product ID (Note: Unchangeable after entry so input carefully): "; cin >> id;

cin.ignore();

cout << "\nEnter Product Name: ";

getline(cin, nme);

cout << "\nEnter Product Quantity: "; cin >> quant;

cout << "\nEnter Product Unit Price (in PKR): "; cin >> ucost;

node\* ptr, \* tempnode;

ptr = start;

while (1)

{

if (ptr->next != NULL)

ptr = ptr->next;

else

break;

}

tempnode = new node();

tempnode->ID = id;

tempnode->quantity = quant;

tempnode->unitCost = ucost;

tempnode->name.assign(nme);

tempnode->next = NULL;

ptr->next = tempnode;

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

}

}

void updateItem(int iD) {

double cost;

int count = 0, check = 0, quan;

string newName;

node\* ptr = start;

while (ptr != NULL)

{

if (iD == ptr->ID) {

count++;

check = 0;

cout << "\nProduct ID # " << iD << " along with name '" << ptr->name << "' is located at " << ptr << " address in memory and occupies '" << count << "' position in the linked list.\n";

cout << "Set new product name as: ";

getline(cin, newName);

ptr->name.assign(newName);

cout << "\nPrevious product quantity: " << ptr->quantity << endl;

cout << "Set new quantity to be: "; cin >> quan;

ptr->quantity = quan;

cout << "\nPrevious product unit price (in PKR): " << ptr->unitCost << endl;

cout << "Set new unit price (in PKR) to be: "; cin >> cost;

ptr->unitCost = cost;

cout << endl;

break;

}

else {

check = 1;

count++;

}

ptr = ptr->next;

}

if (check == 1) {

cout << "\nNo such product with ID # " << iD << " present in the inventory.\n";

}

cout << endl;

system("pause");

}

void deleteItem() {

int count = 1;

if (start == NULL)

{

cout << "\nThe inventory is empty.\n\n";

system("pause");

return;

}

node\* temp1 = start;

while (temp1 != NULL)

{

if (count == 1 && temp1->quantity == 0)

{

cout << "\nProduct ID # " << temp1->ID << " is located at " << temp1 << " address in memory and occupies '" << count << "' position in the linked list.\n";

cout << "Deleting product record entirely as it is out of stock.\n";

temp1 = start;

start = start->next;

delete temp1;

cout << endl;

system("pause");

return;

}

else if (temp1->quantity == 0 && count > 1 && temp1->next == NULL)

{

node\* previous = new node;

temp1 = start;

while (temp1->next != NULL)

{

previous = temp1;

temp1 = temp1->next;

}

cout << "\nProduct ID # " << temp1->ID << " is located at " << temp1 << " address in memory and occupies '" << count << "' position in the linked list.\n";

cout << "Deleting product record entirely as it is out of stock.\n";

previous->next = NULL;

delete temp1;

system("pause");

return;

}

else if (temp1->quantity == 0 && count > 1 && temp1->next != NULL) {

int Id;

Id = temp1->ID;

cout << "\nProduct ID # " << temp1->ID << " is located at " << temp1 << " address in memory and occupies '" << count << "' position in the linked list.\n";

cout << "Deleting product record entirely as it is out of stock.\n";

node\* previous = new node;

temp1 = start;

for (int i = temp1->ID; i < Id; i++) {

previous = temp1;

temp1 = temp1->next;

}

previous->next = temp1->next;

system("pause");

return;

}

temp1 = temp1->next;

count++;

}

cout << "\nNo such product(s) found in inventory that is out of stock.\n";

system("pause");

return;

}

void displayInventory() {

node\* ptr = start;

if (ptr == NULL)

{

cout << "There is no element is the list" << endl;

system("pause");

return;

}

cout << "\nDisplaying details of all products stored in inventory:-\n\n";

while (ptr != NULL)

{

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

cout << "Product ID: " << ptr->ID << endl;

cout << "Product Name: " << ptr->name << endl;

cout << "Product Quantity: " << ptr->quantity << endl;

cout << "Product Per Unit Price (in PKR): " << ptr->unitCost << endl;

ptr = ptr->next;

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

}

cout << endl;

system("pause");

return;

}

void fileInventory() {

ofstream myfile3;

myfile3.open("Inventory.txt", ios::trunc);

node\* ptr = start;

myfile3 << "\n-----------DETAILS OF INVENTORY STATE-----------\n\n";

while (ptr != NULL)

{

myfile3 << "-------------------------------------------\n";

myfile3 << "Product ID: " << ptr->ID << endl;

myfile3 << "Product Name: " << ptr->name << endl;

myfile3 << "Product Quantity: " << ptr->quantity << endl;

myfile3 << "Product Per Unit Price (in PKR): " << ptr->unitCost << endl;

myfile3 << "\n-------------------------------------------\n";

ptr = ptr->next;

}

myfile3.close();

cout << "\nData stored in inventory file successfully at back-end.\n" << endl;

system("pause");

}

void inventory() {

int ch, n;

do

{

system("cls");

cout << "=================================================\n";

cout << " INVENTORY MANAGEMENT SUB-MENU \n";

cout << "=================================================\n\n";

cout << "\t1. Add a product to inventory\n\t2. Update a product in inventory\n\t3. Remove a product from inventory (if out of stock)\n";

cout << "\t4. Display inventory\n\t5. Store inventory data to a text file\n\t6. Back to main menu\n\n";

cout << "=================================================\n\n";

cout << "Select an option: ";

cin >> ch;

cin.ignore();

cout << endl;

switch (ch)

{

case 1:

cout << "\nEnter number of products you want to feed data for: ";

cin >> n;

cin.ignore();

for (int i = 1; i <= n; i++)

{

insertItem();

}

break;

case 2:

cout << "\nEnter the Product's ID for updating: ";

cin >> n;

cin.ignore();

updateItem(n);

break;

case 3:

deleteItem();

break;

case 4:

displayInventory();

break;

case 5:

fileInventory(); // same function but for writing to a text file.

break;

case 6:

break;

default:

cout << "ERROR: INVALID INPUT!\nPlease provide input (1 or 2 or 3...) as shown in menu.\n\n";

break;

}

} while (ch != 6);

cout << endl;

}

void addT() {

int iD, tno;

cout << "\nEnter ID of product being sold: "; cin >> iD;

cin.ignore();

double Bill;

int count = 0, check = 0, quan, check2;

string custName;

node\* TEMP1 = start;

while (TEMP1 != NULL)

{

if (iD == TEMP1->ID && top == NULL && TEMP1->quantity != 0) {

customerNode\* custTemp;

custTemp = new customerNode();

count++;

check = 0;

cout << "\nProduct ID # " << iD << " along with name '" << TEMP1->name << "' has been located\n";

cout << "\nEnter Customer transaction no. (Note: Unchangeable after entry so input carefully): ";

cin >> tno;

cin.ignore();

cout << "\nEnter Customer's Name: ";

getline(cin, custName);

cout << "\nEnter quantity of product being sold to customer: "; cin >> quan;

Bill = (TEMP1->unitCost) \* quan;

TEMP1->quantity = (TEMP1->quantity - quan);

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

cout << "Amount charged for transaction (in PKR): " << fixed << setprecision(2) << Bill << endl;

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

cout << endl;

custTemp->transNo = tno;

custTemp->bill = Bill;

custTemp->name\_.assign(custName);

custTemp->fwd = NULL;

top = custTemp;

break;

}

else if (iD == TEMP1->ID && top != NULL && TEMP1->quantity != 0) {

check = 0;

count++;

cout << "\nProduct ID # " << iD << " along with name '" << TEMP1->name << "' has been located\n";

cout << "\nEnter Customer transaction no. (Note: Unchangeable after entry so input carefully): ";

cin >> tno;

cin.ignore();

cout << "\nEnter Customer's Name: ";

getline(cin, custName);

cout << "\nEnter quantity of product being sold to customer: "; cin >> quan;

Bill = (TEMP1->unitCost) \* quan;

TEMP1->quantity = (TEMP1->quantity - quan);

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

cout << "Amount charged for transaction (in PKR): " << fixed << setprecision(2) << Bill << endl;

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

cout << endl;

customerNode\* Ptr, \* custTempnode;

Ptr = top;

while (1)

{

if (Ptr->fwd != NULL)

Ptr = Ptr->fwd;

else

break;

}

custTempnode = new customerNode();

custTempnode->transNo = tno;

custTempnode->name\_.assign(custName);

custTempnode->bill = Bill;

custTempnode->fwd = NULL;

Ptr->fwd = custTempnode;

break;

}

else {

if (TEMP1->quantity == 0 && TEMP1->next == NULL)

{

check2 = 1;

cout << "\nThe product is unfortunately out of stock. Sorry for convenience\n\n";

system("pause");

return;

}

check = 1;

count++;

}

TEMP1 = TEMP1->next;

}

if (check == 1) {

cout << "\nNo such product with ID # " << iD << " present in the inventory.\n";

}

cout << endl;

system("pause");

return;

}

void updateT(int tID) {

double BILL;

int count = 0, check;

double biLL;

string cName;

customerNode\* ptrrr = top;

while (ptrrr != NULL)

{

if (tID == ptrrr->transNo) {

count++;

check = 0;

cout << "\nTransaction # '" << tID << "' is located at " << ptrrr << " address in memory and occupies '" << count << "' position in the customer transaction records.\n";

cout << "\nOld Customer Name: " << ptrrr->name\_ << endl;

cout << "Change Customer name to (if u want to): ";

getline(cin, cName);

ptrrr->name\_.assign(cName);

cout << "\nOriginal transaction bill: " << fixed << setprecision(2) << ptrrr->bill << endl;

cout << "Set new bill to (in PKR) (if necessary then only): "; cin >> biLL;

ptrrr->bill = biLL;

break;

}

else {

check = 1;

count++;

}

ptrrr = ptrrr->fwd;

}

if (check == 1) {

cout << "\nNo transaction having no. " << tID << " and such nature ever took place.\n";

}

cout << endl;

system("pause");

return;

}

void printTransactions() {

cout << "\nDisplaying details of all transactions taken place so far:-\n";

customerNode\* ptr\_ = top;

while (ptr\_ != NULL)

{

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

cout << "Transaction No.: " << ptr\_->transNo << endl;

cout << "Customer Name: " << ptr\_->name\_ << endl;

cout << "===========================================\n";

cout << "Amount Charged (in PKR): " << fixed << setprecision(2) << ptr\_->bill << endl;

cout << "===========================================\n";

ptr\_ = ptr\_->fwd;

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

}

cout << endl;

system("pause");

return;

}

void fileTransactions() {

ofstream myfile2;

myfile2.open("Customers.txt", ios::trunc);

customerNode\* ptr\_ = top;

myfile2 << "--------------RECORD OF ALL TRANSACTIONS--------------\n";

while (ptr\_ != NULL)

{

myfile2 << "-------------------------------------------\n";

myfile2 << "Transaction No.: " << ptr\_->transNo << endl;

myfile2 << "Customer Name: " << ptr\_->name\_ << endl;

myfile2 << "========================================\n";

myfile2 << "Amount Charged (in PKR): " << fixed << setprecision(2) << ptr\_->bill << endl;

myfile2 << "========================================\n";

myfile2 << "\n-------------------------------------------\n";

ptr\_ = ptr\_->fwd;

}

myfile2.close();

cout << "\nData for all transactions stored in customer file successfully at back-end.\n" << endl;

system("pause");

return;

}

void customer() {

int cho, N;

do

{

system("cls");

cout << "===============================================\n";

cout << " CUSTOMER SERVICE SUB-MENU \n";

cout << "===============================================\n\n";

cout << "\t1. Add a transaction\n\t2. Update a transaction\n";

cout << "\t3. Display all transactions\n\t4. Store transactions data to a text file\n\t5. Back to main menu\n\n";

cout << "===============================================\n\n";

cout << "Select an option: ";

cin >> cho;

cin.ignore();

cout << endl;

switch (cho)

{

case 1:

cout << "\nEnter number of transactions you want to feed data for: ";

cin >> N;

cin.ignore();

for (int i = 1; i <= N; i++)

{

addT();

}

system("pasue");

break;

case 2:

cout << "\nEnter Transaction No. of transaction that need updating: ";

cin >> N;

cin.ignore();

updateT(N);

break;

case 3:

printTransactions();

break;

case 4:

fileTransactions(); // almost same function but for writing to a text file.

break;

case 5:

break;

default:

cout << "ERROR: INVALID INPUT!\nPlease provide input (1 or 2 or 3...) as shown in menu.\n\n";

break;

}

} while (cho != 5);

cout << endl;

}

void fileFinances() {

int tra = 0, prod = 0, Quantity = 0;

double invenValue = 0, sales = 0;

customerNode\* pTr\_ = top;

while (pTr\_ != NULL)

{

tra++;

sales = sales + pTr\_->bill;

pTr\_ = pTr\_->fwd;

}

node\* ptR\_ = start;

while (ptR\_ != NULL)

{

prod++;

Quantity = Quantity + ptR\_->quantity;

invenValue = invenValue + (ptR\_->quantity \* ptR\_->unitCost);

ptR\_ = ptR\_->next;

}

ofstream myfile;

myfile.open("Finances.txt", ios::trunc);

myfile << "\n=====================================================================================\n";

myfile << "\t ---SUMMARY OF FINANCES FOR TODAY--- \t\n";

myfile << "\nTypes of products found of inventory: " << fixed << setprecision(2) << prod;

myfile << "\nNumber of transactions taken place so far: " << fixed << setprecision(2) << tra;

myfile << "\nQuantity of stock left in inventory: " << fixed << setprecision(2) << Quantity;

myfile << "\nCurrent net worth of stock held by the inventory of store (in PKR): " << fixed << setprecision(2) << invenValue;

myfile << "\nCurrent amount of net sales done by the store (in PKR): " << fixed << setprecision(2) << sales;

myfile << "\n======================================================================================\n";

myfile.close();

cout << "\nData regarding stores' finances stored in 'finances' file successfully at back-end.\n" << endl;

system("pause");

}

void finance() {

int tra = 0, prod = 0, Quantity = 0;

double invenValue = 0, sales = 0;

customerNode\* pTr\_ = top;

while (pTr\_ != NULL)

{

tra++;

sales = sales + pTr\_->bill;

pTr\_ = pTr\_->fwd;

}

node\* ptR\_ = start;

while (ptR\_ != NULL)

{

prod++;

Quantity = Quantity + ptR\_->quantity;

invenValue = invenValue + (ptR\_->quantity \* ptR\_->unitCost);

ptR\_ = ptR\_->next;

}

cout << "\n=====================================================================================\n";

cout << "\t ---SUMMARY OF FINANCES FOR TODAY--- \t\n";

cout << "\nTypes of products found in inventory: " << fixed << setprecision(2) << prod;

cout << "\nNumber of transactions taken place so far: " << fixed << setprecision(2) << tra;

cout << "\nQuantity of stock left in inventory: " << fixed << setprecision(2) << Quantity;

cout << "\nCurrent net worth of stock held by the inventory of store (in PKR): " << fixed << setprecision(2) << invenValue;

cout << "\nCurrent amount of net sales done by the store (in PKR): " << fixed << setprecision(2) << sales;

cout << "\n======================================================================================\n";

cout << endl;

int f;

cout << "Do you want to store the above financial data to a text file as well(Enter '0' for No OR '1' for Yes): ";

cin >> f;

system("pause");

cin.ignore();

if (f == 1)

{

fileFinances();

}

else {

cout << "\n";

}

}

// Function that accepts the password

bool takePasswdFromUser(char sp = '\*')

{

char ch;

string pass = "", pwd = "12345678";

int i, chance = 0;

do

{

ch = \_getch(); // single character input without showing it on the screen

if (ch != 13) //!= 13 beacuse of enter

{

pass += ch;

cout << "\*"; // and intead of single character he is print \*

}

} while (ch != 13);

if (pass == pwd)

{

return true;

}

else

{

return false;

}

}

int main() {

int choice;

bool bypass = false;

do

{

cout << "\n================================================\n";

cout << " DEPARTMENTAL STORE MANAGEMENT SYSTEM \n";

cout << "================================================\n\n";

bool pwd;

cout << "\t ADMINISTRATOR LOGIN\t\n";

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

cout << "\nEnter login password: ";

// Function call

pwd = takePasswdFromUser();

cout << endl;

if (pwd == true)

{

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

bypass = true;

cout << "\n\t \*\*ACCESS GRANTED\*\*\t\n\nSwitching to control main menu....\n\n";

system("pause");

}

else {

cout << "\n\t\*\*ACCESS DENIED\*\*\t\n\nPlease re-enter the correct password.\n";

system("pause");

}

} while (bypass != true);

do {

system("cls");

cout << "==============================================\n"; // Interactive & user-friendly control menu for our Store Management System

cout << " STORE CONTROL MAIN MENU \n";

cout << "==============================================\n\n";

cout << "\t1. INVENTORY MANAGEMENT\n\t2. CUSTOMER SERVICE \n\t3. CHECK FINANCES OF STORE\n\t4. CLOSE MENU AND QUIT\n\n";

cout << "==============================================\n\n";

cout << "SELECT AN OPTION FROM MENU: ";

cin >> choice;

cin.ignore();

switch (choice)

{

case 1:

system("cls");

inventory();

cout << "==============================================\n\n";

break;

case 2:

system("cls");

customer();

cout << "==============================================\n\n";

break;

case 3:

system("cls");

finance();

cout << "==============================================\n\n";

break;

case 4:

break;

default:

cout << "ERROR: INVALID INPUT!\nPlease provide input (like 1 or 2 or 3...) as shown in menu.\n\n";

break;

}

} while (choice != 4);

cout << endl;

system("pause");

return 0;

}

* **Future work:-**

After successfully creating and working on the project, we realized that there are plenty of things that could have been done better and many more things that could have been made part of project if there were no outside factors involved that were out of our control (i.e. time constraints, still limited knowledge & experience of programming, personal reasons, etc. etc.). We definitely intent to deal with them along with these other provisions:-

1. Add a discount % feature in our proposed system.
2. Assign unique ID to products via use of a barcode scanner by forming it as part of our proposed system.
3. Use other return data types and more arguments in some methods in order to reduce source code size/length. Make them more intertwined along with each other. Also, involve better data structures or utilize more effective algorithms for searching, sorting, insertion, etc. to improve time and space complexity of the overall program.
4. Use of user-defined classes.
5. Connect an SQL database at back-end with our proposed system.
6. Make an online platform for our proposed Departmental Store system that alleviates our current standalone system.

* **Conclusion:-**

To conclude, this project is a very important aspect of any Departmental Store as basically every Store has some sort of Inventory & Billing management mechanism in place. If it’s paper-based, this can easily serve to be a good replacement and a viable investment for the general store in the long-run. However, there is always room for improvement and thus certainly there are a thing that can be done differently. Overall, we are satisfied with how it all turned out. We gained much needed experience and awareness about the subject, and we hope that we were able to display certain level of understanding and showcase necessary programming skills to the best of our abilities. And we also hope that we learnt many invaluable things along the way.