**PROJECT REPORT**

**Data Structure & Algorithms (CSL-221)**



**BS(CS)-3(A)**

**Project Title: Pharmacy Management**

**Submitted to: Miss.Maham**

**BAHRIA UNIVERSITY KARACHI CAMPUS**

**Department of Computer Science**

**Group Members**

|  |  |
| --- | --- |
| **Members Name** | **Enrollment Number** |
| * **Muhammad Awais Arshad** | **02-134201-038** |
| * **Sajjad Ali** | **02-134201-070** |
| * **Haroon** **Asif** | **02-134201-071** |

# ACKNOWLEDGEMENT:

We would first of all like to express all our gratitude to our teacher **Miss Maham** for the continuous support, motivation, enthusiasm and immense knowledge. Her guidance helped us all in the time of this project in every way. We could not have imagined having a better advisor and a perfect teacher for our lab course.

Also, this completion of the project could not have been accomplished without the support of us each members working together, for the sleepless nights we were working together so hard to work off this project and achieve good results.

# INTRODUCTION:

Pharmacy management system is a management system that is designed to improve accuracy and to enhance safety and efficiency in the pharmaceutical store. It is a computer-based system which helps the pharmacist to improve inventory, management, cost, and much more etc.

Can we believe that this is the 21st century and it’s 2021 and still in some of our hospitals and pharmaceutical stores still keep records of the medicines or whatever the work required for pharmacy stuff is still being manually hand written on a paper. In Pakistan medical works are based on the manual process, and each work is maintained in the paper. The details of purchasing drugs, audits, sell reports maintained on the paper while anyone can enter into the system and can make changes in these reports, so it is not a safe method to keep the information on the paper. The pharmacist faces problem in searching the products from the self as it is not an easy method to remember about the place of each medicine. There no system which can alert the pharmacist about the end of the drugs.

Well, just to get rid of that system and create something like a program software that will benefit everyone. The Pharmacy Management System gives service for users, managed by the pharmacist who gives implementation of function relatively in effective times as well as will design for removing time wasting, saving resources, easy data access of the medicine, security on data input and data access by removing almost manual based system. By removing all the manual-based system what can be achieved from the software program, here are some examples:

**Automated System:** Everything that is arranged, searched, updated and deleted is through human only. In general manual-based system is un-computerized system which is tedious in its data arrangement for efficient work.

**Management:** A system in which manage, organize, formulate data through a technical data structure arrangement.

**New way of Billing:** The way in which generating paper which store information about some specific data containing details explanation can be done easily using this software.

**User- Friendly:** The System is clear for using the created software interface for manipulating actions or tasks. In the other way the proposed system is designed for human likeable components in color, font and other related things.

# BACKGROUND:

Pharmacy management system is a system that consists of data entry, retrieval and monitoring stock, sale, customer records, debtor’s and management administrator’s records and determination of minimum quantity of each medicine. String searching technique also applied in this system. this system or technique refers by medicine names. Besides that, the system also provides two types of methods which are quantity and their prices.

This system enable administrator to control and monitor the drugs stock effectively. Due to the size and quantity service of the pharmacy. Pharmacy management deals with managing the medicine stock and selecting the suitable medicine needed by the customers.

# PROBLEM STATEMENT:

Improving performance and efficiency in pharmacy shops is a major goal of Pharmacy Management System. The transaction related to sales, maintaining of stocks records are maintained manually at present. These are to be automated and an application is required to relate of them relatively and logically so that the current system can be replaced and accepted without major changes and problems. The application will provide quick access to the records maintained and must reveal the important reviews about the business so that the growth can be easily compared and will provide with the various reports showing the related details that the important decisions could be taken easily. The following are among of the problems that lead to propose creation and development of Pharmacy Management System Software.

* There is no affective management of information.
* Hard and Time consuming on preparation of daily, weekly, monthly, and yearly reports.
* Discrepancies of stock items.
* It hard to determine stock balance.

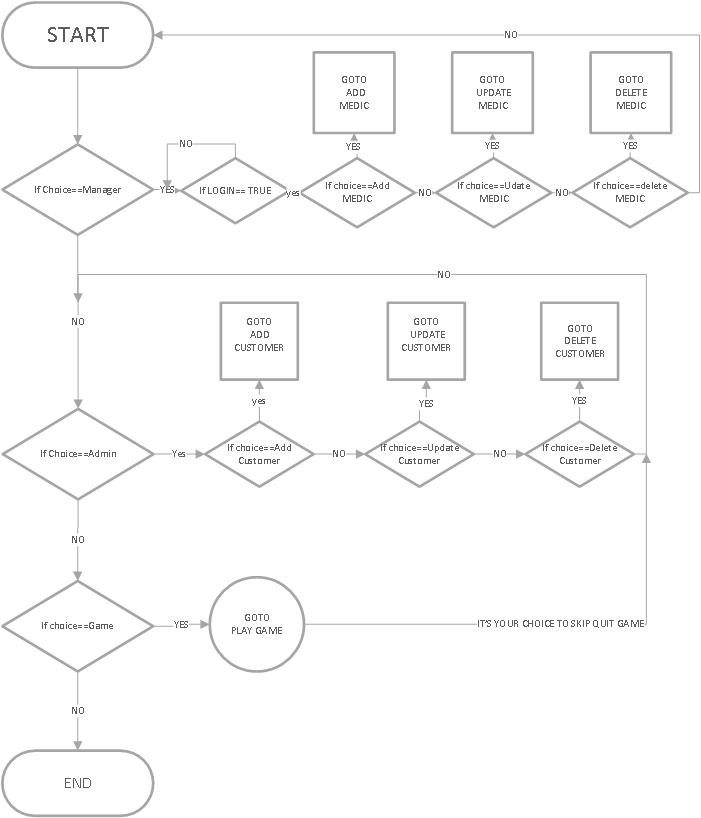
# PROPOSED SYSTEM:

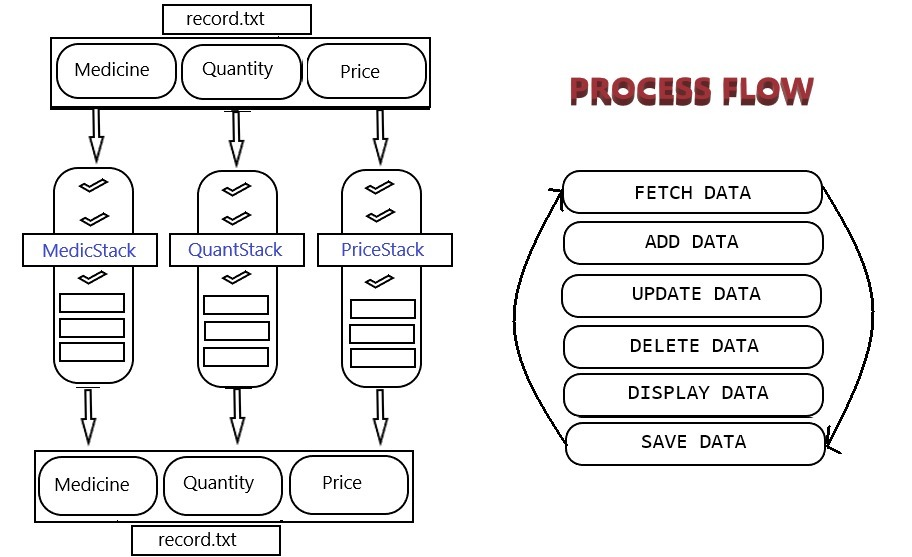
The pharmacy management system is designed based on computer science students in order to illuminate the problem of the current system which is provided by the system involved all the pharmaceutical of the pharmacy. The Pharmacy management system is based on the computer which will simplify the maintenance of the information, accessible and efficient. The system will provide the information about the of the drugs in the medical so that pharmacist can order them drugs before the end.

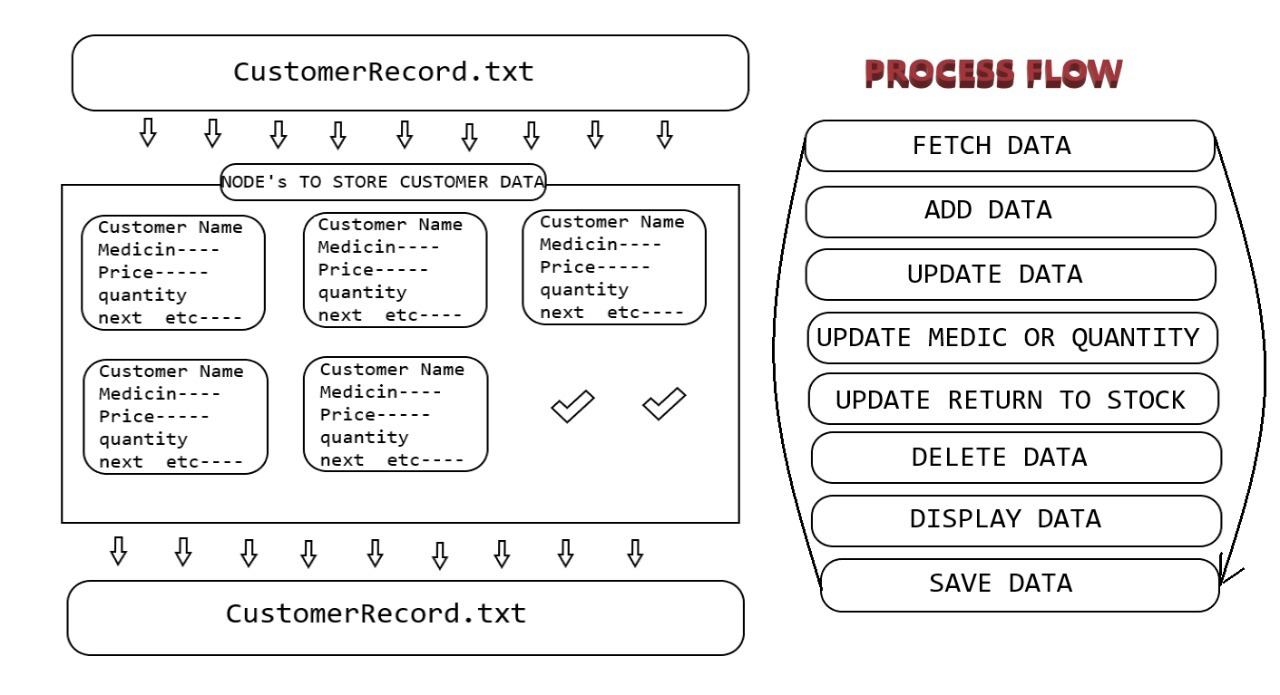
What’s new in our project is that the pharmacy management system that consists of data entry, retrieval and monitoring stock, sale, customer records, debtor’s and management administrator’s records and determination of minimum quantity of each medicine. String searching technique also applied in this system. this system or technique refers by medicine names. Besides that, the system also provides two types of methods which are quantity and their prices.

Compared to the current manual system, the implementation of pharmacy management system will reduce the time spent for paperwork, leading to concentration on improving pharmaceutical management system. This project is to improve the maintenance and manipulation of the drugs in the medicals. The pharmacy management system will be used to minimize the time and resource by maintaining the details of the drug systemically so that the data can be used in possible quickest time. While the resource which is minimized are workforce, money, papers, etc.

# FLOW DIAGRAM OF THE SYSTEM:







# SCREEN SHOTS OF MODULE WITH EXPLANATION AND CODE SNIPPET WITH ALGO APPLIED:

#include <iostream>

#include <string>

#include <fstream>

#include <cctype>

#include <sstream>

#include<windows.h>

#include <ctime>

# define Size 1000

using *namespace* std;

In this project we use three stacks to handle the data about Medicines. MedicStack which is used to store Medicine name , Price stack which is used to store Price of each Medic and the last one is Quantstack this one is used to store quantity of Medicine available in stock this all Stacks are Connected to each other by their Index they have a strong bond on the behalf of stack Index.

*int* recno = 0;

// Stacksuse to Store all fetched data;

string MedicStack[Size];

*int* PriceStack[Size];

*int* QuantStack[Size];

We Use nodes to store data about the customers and their whole data in which Id of customer and the date of brought Medicines and the top variable which representing the Number of medicines. we use stack to store a data about the customers Medicine and their quantity with total Prices.

*struct* node{

*int* id;

    string CustomerName;

    string date;

*int* top;

    string Medic[Size];

*int* Quant[Size];

*int* Price[Size];

*int* totalPrice;

    node\*prev;

    node\*next;

};

node\*head=NULL;

node\*tail=NULL;

Following are the function which are used to handle the program in pieces in this Declaration we show the description of line about the below functions

These all function are used for Manger Block who can handle only the Medicine stock.

// For Manager Block to Control All Information About Medicince like buy or Update Medic Stock impelement the Stack method here..

// for Welcome Designing

*void* Welcome();

// Manager have to required Login First to Move Further

*bool* Login();

// This Method Fetch All data About Medicines from record File

*void* FetchData();

// this method use to update every Possible Costimization

*void* UpdateData(*int* *code*,*bool* *check*);

// this method is to delete any Medicine

*void* DeleteData(*int* *code*,*bool* *confirm*);

// this method is used to Save Data in File

*void* SaveData();

// this method is used to Add medicine in Stock

*void* AddData();

// to Search about any Medicine

*void* SearchData();

// Display All information about Medicine

*void* DisplayData();

// Summary OF All Medicine

*void* MedicDisplay(*int* *id*);

// Expertize to Delete From Stack Using

*void* DeleteExpert(*int* *pos*);

// Using Bubble Sort to Sort the Medicines

*void* Sort();

// swap Medicine Name

*void* swappingName(*int* *i*,*int* *j*);

// swap a Quantity or Price as resspect to MEDICINE ORDERS(sorting)

*void* swappingValue(*int* *Quantity*[],*int* *i*,*int* *j*);

Below functions are the decleration of all those function which can are used for Admin block to handele all Details about Customers and their Orders.

// For Admin Block

// To Create a node to add Customers Data

*void* Create(*int* *num*);

// if the Insert Customer the the last node we will utilize this one Function

*void* InsertEnd();

// to use for insert Customer at the Begining of List

*void* InsertBegin();

// check the Customer List is Empty or not

*bool* checkEmpty();

// Delete Customer from Starting of List

*void* DeleteBegin();

// Delete Customer from End of List

*void* DeleteEnd();

// used to Delete Customer from required Position

*void* DeletePosition();

// Display All Data About Customer

*void* Display();

// this one Function Connected with Create(int n) function for Passing the n as Argument

*void* TakeOrder();

// This Function is used to Update any required data about Customer

*void* UpdateOrder();

// this is used to Search any customer to Update thier Record

*void* Searching();

// used to display All Customers Data

*void* AllOrdersDetail();

// Check the INPUT id is valid or not

*bool* CorrectID(*int* *id*);

// check the Correct Quantity is Valid or not

*bool* CorrectQuantity(*int* *id*,*int* *quant*);

// this one Funtion is Connect with Create function to Add the Medicine for Customers

*void* AddMedic(node \**temp*);

// this Function will Display the  individual Customer DATA

*void* DisplayNodeArray(*struct* node \**temp*);

// A funtion used to fetch data from File (customerRecord.txt)

*void* FetchCustomerData();

// A function used to Save all new or updated data to txt file

*void* SaveCustomerData();

// show Main menu

*void* MainMenu();

// show Admin menu

*void* AdminMenu();

// to show Manager Menu

*void* ManagerMenu();

// this one function is used to give ID to Updaate Order function

*void* TakeCustomerPosition();

// show the All Customer Summary for Some Places;

*void* CustomersSummary();

// this one give ID to Delete Position Function

*void* RemovePosition();

// this function return time Current

string GetTime();

## MAIN FUNCTION TO HANDLE ALL FUNCTIONS:

*int* main()

{

    system("CLS");

    system("Color 70");

    FetchData();

    FetchCustomerData();

*char* choiceLogin,choiceAdmin,choiceManager;

    do{

         system("CLS");

         Welcome();

         MainMenu();

    cout<<"\t\t\t\t\tSelect Require Login Option : ";  cin>>choiceLogin;

    choiceLogin=(*char*)tolower(choiceLogin);

        switch(choiceLogin){

        case 'm' :{

            if(Login()==true){

                system("CLS");

                ManagerMenu();

                do{

                    cout<<"\t\t\t\t\tSelect Require Option to Manage : ";  cin>>choiceManager;

                    choiceManager=(*char*)tolower(choiceManager);

                    switch(choiceManager){

                    case 'a':{

                        AddData();

                        break;

                    }

                    case 'u':{

                        UpdateData(0,false);

                        break;

                    }

                    case 'd':{

                        DeleteData(0,false);

                        break;

                    }

                    case 's':{

                        SearchData();

                        break;

                    }

                    case 'm':{

                        cout<<endl;

                        cout<<endl;

                        DisplayData();

*int* c=0;

*char* merzi;

*bool* check=false;

                        do{

                            c++;

                            if(c==1){

                                cout<<"\t\t\t\tIs you Want to Update Any Record(y/n) : "; cin>>merzi;

                            }

                            else{

                                cout<<"\t\t\t\tPlease Select \*valid Options\n ";

                                cout<<"\t\t\t\tIs you Want to Update Any Record(y/n) : "; cin>>merzi;

                            }

                            merzi=(*char*)tolower(merzi);

                            if(merzi=='y'||merzi=='n'){

                                check=true;

                            }

                            else{

                                check=false;

                            }

                        }while(check!=true);

                        if(merzi=='y'){

                            system("CLS");

                            UpdateData(0,false);

                        }

                        else{

                            system("CLS");

                            ManagerMenu();

                        }

                        break;

                    }

                    case 'e':{

                        cout<<"\t\t\t\t\tManager--Menu  Exited......\n";

                    }

                    default:{

                        cout<<"\n\t\t\t\tPlease Select Valid Option....thanks\n";

                        break;

                    }

                                        }

                }while (choiceManager!='e');

            }

            break;

        }

        case 'a' :{

            system("CLS");

            AdminMenu();

            do{

                cout<<"\t\t\t\tSelect Require Option as Admin : ";  cin>>choiceAdmin;

                choiceAdmin=(*char*)tolower(choiceAdmin);

                switch(choiceAdmin){

                    case 'a' :{

                        TakeOrder();

                        break;

                    }

                    case 'u' :{

                        TakeCustomerPosition();

                        break;

                    }

                    case 's' :{

                        Searching();

                        break;

                    }

                    case 'd' :{

                        RemovePosition();

                        break;

                    }

                    case 'o' :{

                        system("CLS");

                        Display();

*int* c=0;

*char* merzi;

*bool* check=false;

                        do{

                            c++;

                            if(c==1){

                                cout<<"\t\t\t\tIs you Want to Update Any Record(y/n) : "; cin>>merzi;

                            }

                            else{

                                cout<<"\t\t\t\tPlease Select \*valid Options\n ";

                                cout<<"\t\t\t\tIs you Want to Update Any Record(y/n) : "; cin>>merzi;

                            }

                            merzi=(*char*)tolower(merzi);

                            if(merzi=='y'||merzi=='n'){

                                check=true;

                            }

                            else{

                                check=false;

                            }

                        }while(check!=true);

                        if(merzi=='y'){

                            system("CLS");

                        TakeCustomerPosition();

                        }

                        else {

                            system("CLS");

                            AdminMenu();

                        }

                        break;

                    }

                    case 'e' :{

                        cout<<"\t\t\t\t\tAdmin--Menu  Exited......\n";

                        break;

                    }

                    default:{

                        cout<<"\n\t\t\t\tPlease Select Valid Option....thanks\n";

                    }

                }

            }while(choiceAdmin!='e');

            break;

        }

        case 'e':{

            cout<<"\t\t\t\t\tPharmacy Managment System Exited......\n";

            break;

        }

         case 'g':{

                        cout<<"\t\t\t\tDEAR USER : \n";

                        cout<<"\t\t\t\tAre you Getting Bored : it's Okay\n";

                        cout<<"\t\t\t\t         LETS PLAY GAME           \n";

                        cout<<"\t\t\t\t-----PRESS ANY KEY TO START-----\n";

                        system("pause");

                        system("cmd/c start https://guessanumber-32d75.web.app");

                        break;

                    }

        default:{

            cout<<"\n\t\t\tPlease Select Valid Option....thanks\n";

            break;

        }

    }

    }while(choiceLogin!='e');

    SaveData();

    SaveCustomerData();

    return 0;

}

FUNCTION TO GET CURRENT TIME WITH DATE (LOCAL):

string GetTime(){

                 time\_t ct = time(0);

*char*\* cc = ctime(&ct);

                string date="";

                for(*int* i=0;i<24;i++){

                // cout<<cc[i]<<" : "<<i<<endl;

                date+=cc[i];

                }

 return date;

}

FUNCTION TO USE LOGIN FOR MANAGER ONLY:

*bool* Login(){

    string userName;

    string userpassword;

*bool* login=false;

*int* count=0;

    while(login!=true){

        count++;

        if(count==5){

            login=false;

            cout<<"\t\t\tYou excced the limit.Please tryagain later.....\n";

            break;

        }

        if(count==1){

        cout<<"\t\t\t\tEnter UserName : "; cin>>userName;

        cout<<"\t\t\t\tEnter UserPassword : "; cin>>userpassword;

        }

        else{

        cout<<"\t\t\t\tEnter \*Valid UserName : "; cin>>userName;

        cout<<"\t\t\t\tEnter \*Valid UserPassword : "; cin>>userpassword;

        }

        if(userName=="admin" && userpassword=="123"){

            login=true;

        }

        }

    return login;

}

FUNCTION TO DISPLAY MEDICINE INDIVIDUAL:

*void* MedicDisplay(*int* *id*){

        cout<<"\t\t\t\t ------------------"<<MedicStack[*id*]<<"------------------\n";

        cout<<"\t\t\t\t           Product ID : "<<*id*<<"                  \n";

        cout<<"\t\t\t\t           Medic Name : "<<MedicStack[*id*]<<"      \n";

        cout<<"\t\t\t\t           Price      : "<<PriceStack[*id*]<<"      \n";

        cout<<"\t\t\t\t           Quantity   : "<<QuantStack[*id*]<<"      \n";

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

}

FUNCTION TO UPDATE MEDICINE DATA:

*void* UpdateData(*int* *code*,*bool* *check*){

*char* choice;

*int* recno2;

            if(*check*==true){

                recno2=*code*;

            }

            else{

                DisplayData();

                cout << "\t\t\t\tEnter Medicince(ID) which you like to Update : ";

                cin >> recno2; // get the user to pick a record

            }

            if (recno2 < 0 || recno2 >= recno) // validate their choice

            {

                cout << "\t\t\t\t\tInvalid choice of ID (id>0 && id<"<<recno<<") "<< endl;

            }

            else

            {

                do

                    {

                        system("CLS");

                        MedicDisplay(recno2);

                        cout << "\n\t\t\t\tYou have chosen to Update record " << recno2 << " : " <<MedicStack[recno2]<<endl;

                        cout<<"\t\t\t\t---------MEDICINE UPDATE MENU----------\n";

                        cout<<"\t\t\t\t     MEDICINE NAME     : PRESS N/n      \n";

                        cout<<"\t\t\t\t     MEDICINE PRICE    : PRESS P/p      \n";

                        cout<<"\t\t\t\t     MEDICINE QUANTITY : PRESS Q/q      \n";

                        cout<<"\t\t\t\t     MEDICINE FULL     : PRESS F/f      \n";

                        cout<<"\t\t\t\t     EXIT TO MANAGER   : PRESS E/e      \n";

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

                        cout<<"\t\t\t\t\tSelect Required Option : "; cin>>choice;

                        choice=(*char*)tolower(choice);

                        string nam;

*int* pr,quant;

                        switch (choice)

                        {

                        case 'p' :

                            {

                                cout<<"\t\t\t\t\tEnter Price of Medicince : "; *int* pr; cin>>pr;

                                PriceStack[recno2]=pr;

                                break;

                            }

                        case 'n' :

                            {

                                cout<<"\t\t\t\t\tEnter Name of Medicince : ";

                                cin.ignore();

                                getline(cin,nam);

                                MedicStack[recno2]=nam;

                                break;

                            }

                        case 'q' :

                            {

                                cout<<"\t\t\t\t\tEnter Quantity of Medicince : ";

                                cin>>quant;

                                QuantStack[recno2]=quant;

                                break;

                            }

                        case 'a' :

                            {

                                cout<<"\t\t\t\t\tEnter Name of Medicince : ";

                                cin.ignore();

                                getline(cin,nam);

                                MedicStack[recno2]=nam;

                                cout<<"\t\t\t\t\tEnter Price of Medicince : ";

                                cin>>pr;

                                PriceStack[recno2]=pr;

                                cout<<"\t\t\t\t\tEnter Quantity of Medicince : ";

                                cin>>quant;

                                QuantStack[recno2]=quant;

                                break;

                            }

                        case 'd' : {

                            DeleteData(recno2,true);

                        }

                        case 'e': {

                            system("CLS");

                            ManagerMenu();

                            break;

                        }

                        default:

                        cout<<"\t\t\t\t\tPlease Select Valid Option from Menu";

                            break;

                        }

                }while(choice!='e');

            }

    SaveData();

}

**FUNCTION TO DELETE MEDICINE from Any Position from Stack (CONNECTED WITH DELETEDATA(code,bool):**

*void* DeleteExpert(*int* *a*[],*int* *pos*,string *b*[],*bool* *integer*){

     if(*integer*==true){

            for(*int* i=*pos*; i<recno;i++)

                {

*a*[i]=*a*[i+1];

                }

        }

    else{

        for(*int* i=*pos*; i<recno;i++)

                {

*b*[i]=*b*[i+1];

                }

    }

}

**FUNCTION TO DELETE MEDICINE FROM STOCK:**

*void* DeleteData(*int* *code*,*bool* *confirm*){

*char* choice;

*int* recno1 = 0;

    do{

            if(*confirm*==true){

              recno1=*code*;

            }

            else{

            DisplayData();

            cout << "\t\t\t\tEnter Medicine(ID) you like to delete : ";

            cin >> recno1;

            }; // get the user to pick a record

            if (recno1 < 0 || recno1 >= recno) // validate their choice

            {

                cout << "\t\t\t\tInvalid choice" << endl;

            }

            else

            {

                cout << "\t\t\t\tYou have chosen to delete record " << recno1<<" : "<<MedicStack[recno1]<<endl;

                DeleteExpert(QuantStack,recno1,MedicStack,false);

                DeleteExpert(QuantStack,recno1,MedicStack,true);

                DeleteExpert(PriceStack,recno1,MedicStack,true);

                recno--; // blank out users choice

            }

            cout<<"\t\t\t\tMedicine "<<MedicStack[recno1]<<" Deleted Successfully....\n";

            SaveData();

            cout<<"\n\t\t\t\tDo you want to Delete More Data(y/n) : "; cin>>choice;

            choice=(*char*)tolower(choice);

*confirm*=false;

    }while(choice!='n');

    system("CLS");

    ManagerMenu();

}

FUNCTION TO ADD MEDICINE TO STOCK :

*void* AddData(){

*char* choice;

    string nam;

*int* pr,quant;

    do{

    cout<<"\t\t\t(A/a)Add Medicine(E/e) to Exit MENU...  : "; cin>>choice;

    choice=(*char*)tolower(choice);

        switch (choice)

            {

            case 'a' :

            {

                        cout<<"AT THIS RECORD"<<recno<<endl;

                        cout<<"\t\t\t\t\tEnter Name of Medicince : ";

                        cin.ignore();

                        getline(cin,nam);

                        MedicStack[recno]=nam;

                        cout<<"\t\t\t\t\tEnter Price of Medicince : ";

                        cin>>pr;

                        PriceStack[recno]=pr;

                        cout<<"\t\t\t\t\tEnter Quantity of Medicince : ";

                        cin>>quant;

                        QuantStack[recno]=quant;

                        recno++;

                        cout<<"Last ONE : "<<recno<<endl;

                        break;

            }

            case 'e' : {

                        system("CLS");

                        ManagerMenu();

                        break;

            }

            default:

                        {

                        cout<<"\t\t\t\tInvalid Choice\n";

                        break;}

            }

    }while(choice!='e');

    SaveData();

}

**FUNCTION TO DISPLAY THE WHOLE MEDICINE RECORD :**

*void* DisplayData(){

    cout<<"\t\t\t\t----------------MEDICINES DATA------------------\n";

      cout<<"\t\t\t\t  ID / MEDIC / PRICE(Rs) / QUANTITY\n";

        for (*int* i = 0; i < recno; i++)

            {

            cout<<"\t\t\t\t"<<i<<"\t"<<MedicStack[i]<<"\t"<<PriceStack[i]<<"\t"<<QuantStack[i]<<endl;

        }

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

    cout<<endl;

}

**FUNCTION TO SEARCH ANY MEDICINE :**

*void* SearchData(){

    string MediName;

*int* id=-1;

    cin.ignore();

    cout<<"\t\t\t\t\tEnter Medicine Name to Search : ";

    getline(cin,MediName);

    cout<<MediName;

    for(*int* i=0;i<recno;i++)

    {

        if(MediName==MedicStack[i]){

        id=i;

        break;

        }

    }

    if(id>-1){

        MedicDisplay(id);

        cout<<"\t\t\t\tYou Want to Edit this Record(y/n) : "; *char* merzi; cin>>merzi;

        merzi=(*char*)tolower(merzi);

        if(merzi=='y'){

            system("CLS");

            UpdateData(id,true);

        }

    }

    else{

        cout<<"\t\t\t\t Sorry "<<MediName<<" is Not Avaliable yet!.....\n";

    }}

**SAVE MEDICINE DATA TO FILE :**

*void* SaveData(){

    Sort();

    ofstream records\_output("records.txt"); // create a file stream and open the file for output

    for (*int* i = 0; i < recno; i++) // write out the new records to screen

    {

        if (!MedicStack[i].empty())

        {

            records\_output << MedicStack[i] << endl;// also overwrite the original file

            records\_output << QuantStack[i] << endl;// also overwrite the original file

            records\_output << PriceStack[i] << endl;// also overwrite the original file

        }

    }

    records\_output.close();

}

FETCH MEDICINE DATA FROM FILE :

*void* FetchData(){

    string line;

*int* num;

*int* countLine=0,medic=1,quant=2,price=3;

    ifstream records\_input("records.txt"); // create a file stream  for reading

    while (getline(records\_input, line)) // count all the lines from the file

    {

               countLine++; //count all the lines from the file

        if(countLine==medic){

            MedicStack[recno]=line;

                medic+=3;

        }

        else if(countLine==quant){

            istringstream ( line ) >> num;

            QuantStack[recno]=num;

            quant+=3;

        }

        else if(countLine==price){

            istringstream ( line ) >> num;

            PriceStack[recno]=num;

            price+=3;

            recno++;

        }

    }

    records\_input.close();

}

ADMIN BLOCK---------------

**FUNCTION TO GET NUMBER OF CUSTOMER CONNECT With CREATE(int n) to pass n arguments:**

*void* TakeOrder(){

*int* number;

*int* count=0;

    do{

        count++;

         if(count==1){   cout<<"\t\t\t\t\t\tEnter Numbers of Customers : "; cin>>number;}

         else{           cout<<"\t\t\t\t\t\tEnter Numbers of Customers(N>0) : "; cin>>number;}

    }while(number<=0);

    Create(number);

}

**This Function is used to Add required Medicines for Customers:**

*void* AddMedic(node\**temp*){

*char* choice;

    do{

*int* id;

*int* quant;

*bool* check=false;

                    while(check!=true)

                    {

                        cout<<"\t\t\t\tSelect Medic(ID) From TABLE to buy : ";  cin>>id;

                        check=CorrectID(id);

                    }

*temp*->Medic[*temp*->top]=MedicStack[id];

                    check=false;

                    while(check!=true)

                    {

                        cout<<"\t\t\t\tEnter Quantity of Order "<<MedicStack[id]<<" (x<"<<QuantStack[id]<<") : ";  cin>>quant;

                        check=CorrectQuantity(id,quant);

                    }

                QuantStack[id]=QuantStack[id]-quant;

*temp*->Quant[*temp*->top]=quant;

*temp*->Price[*temp*->top]=PriceStack[id]\*quant;

*temp*->top++;

                cout<<"\t\t\t\tIs You Want To Buy More Medics (y/n) : ";  cin>>choice;

                choice=(*char*)tolower(choice);

    }while(choice!='n');

}

**FUNTION TO CREATE A NODE TO SAVE CUSTOMER DETAIL:**

*void* Create(*int* *num*){

    node\* temp,\*start;

    // start=head;

    for(*int* i=1;i<=*num*;i++){

       cout<<"\t\t\t\t\tCUSTOMER # "<<i<<endl;

        if(head==NULL){

            cin.ignore();

            cout<<"\t\t\t\tName of Customer : "; string name; getline(cin,name);

            temp=new node();

            head=temp;

            temp->id=1;

            temp->CustomerName=name;

                      DisplayData();

                      AddMedic(temp);

*int* TotalPrice=0;

                    for(*int* i=0;i<temp->top;i++){

                        TotalPrice+=temp->Price[i];

                    }

            temp->date=GetTime();

            temp->totalPrice=TotalPrice;

            temp->next=NULL;

            temp->prev=NULL;

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

        }

        else{

            InsertEnd();

        }

    }

    SaveCustomerData();

    SaveData();

    system("CLS");

    AdminMenu();

}

**CONNECT WITH CREATE FUNCTION TO INSERT AT END OF NODE:**

*void* InsertEnd(){

    if(head==NULL){

        Create(1);

    }

    else{

    node \*temp,\*ptr;

    ptr=head;

    while(ptr->next!=NULL){

        ptr=ptr->next;

    }

    temp=new node();

        cin.ignore();

     cout<<"\t\t\t\tName of Customer : "; string name; getline(cin,name);

            temp->id=ptr->id+1;

            temp->CustomerName=name;

                    DisplayData();

                    AddMedic(temp);

*int* TotalPrice=0;

                    for(*int* i=0;i<temp->top;i++){

                        TotalPrice+=temp->Price[i];

                    }

            temp->date=GetTime();

            temp->totalPrice=TotalPrice;

            temp->next=NULL;

            temp->prev=ptr;

            ptr->next=temp;

            tail=temp;

    }

}

**FUNCTION TO TAKE CUSTOMER VALID ID TO UPDATE ORDER CONNECTED WITH CUSTOMER ORDER:**

*void* TakeCustomerPosition(){

*int* realID=0;

    node\*ptr;

    ptr=head;

*bool* checked=false;

*int* CountAttempt=0;

    system("CLS");

    CustomersSummary();

    do{

        CountAttempt++;

*int* id;

        if(CountAttempt==1){

        cout<<"\t\t\t\tEnter Customer ID to UPDATE DATA :  "; cin>>id;

        }

        else{

        cout<<"\t\t\t\tEnter \*Valid Customer ID to UPDATE DATA :  "; cin>>id;

        }

*int* count=0;

        while(ptr!=NULL){

            count++;

            if(ptr->id==id){

                cout<<ptr->id;

                realID=count;

            }

        if(id<=count&&id>0){

            checked=true;

        }

        else{

            checked==false;

        }

        ptr=ptr->next;

        }

    }while(checked!=true);

    if(checked==true){

        UpdateOrder(realID);

    }

    else{

        cout<<"\t\t\tYou put Invalid ID....\n";

    }

}

**FUNCTION TO UPDATE CUSTOMER DATA :**

*void* UpdateOrder(*int* *pos*){

*struct* node \*PrevNode;

*struct* node \*PosNode;

    PosNode=head;

*char* choice;

    string name;

    if(PosNode!=NULL){

         for(*int* i=1;i<*pos*;i++){

              PrevNode=PosNode;

              PosNode=PosNode->next;

                            }

         cout<<"\t\t\t\t ---------------EDITING MENU---------------\n";

         cout<<"\t\t\t\t|     CUSTOMER NAME     : PRESS N/n        | \n";

         cout<<"\t\t\t\t|     CUSTOMER MEDICINE : PRESS M/m        |\n";

         cout<<"\t\t\t\t|     GOTO ADMIN MENU   : PRESS E/e        |\n";

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

       do{

             cout<<"\t\t\t\tEnter require option to Edit : "; cin>>choice;

             choice=(*char*)tolower(choice);

         switch(choice){

             case 'n':

             {

                 cout<<PosNode->CustomerName<<endl;

                 cin.ignore();

                 cout<<"\t\t\t\t\tEnter Name of Custumer : "; getline(cin,name);

                 PosNode->CustomerName=name;

                 cout<<PosNode->CustomerName<<endl;

                 break;

             }

             case 'm' :

             {

                 cout<<"\t\t\t\t ---------------EDITING MENU---------------\n";

                 cout<<"\t\t\t\t|     UPDATE MEDICINE     : PRESS M/m        |\n";

                 cout<<"\t\t\t\t|     UPDATE QUANTITY     : PRESS Q/q        |\n";

                 cout<<"\t\t\t\t|     GOTO ADMIN MENU     : PRESS E/e        |\n";

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

*char* mchoice;

                 do{

                    cout<<"\n\t\t\t\tEnter Required Option : "; cin>>mchoice;

                    mchoice=(*char*)tolower(mchoice);

                    switch (mchoice)

                    {

                    case 'm':

                            {

                                DisplayNodeArray(PosNode);

*int* id;

*char* merzi;

                                do{

                                    // DisplayNode(\*\*PosNode);

                                    cout<<"\t\t\t(a)Add or (r)Remove Medicince (e)EXIT : "; cin>>merzi;

                                    if(merzi=='r')

                                    {

                                        cout<<PosNode->top<<"<-POSITION TOP\n";

*bool* correct=false;

                                        DisplayNodeArray(PosNode);

                                        do{

                                            cout<<"\t\t\t\tEnter id to remove Medicince : ";  cin>>id;

                                            if(id<=PosNode->top) correct=true;

                                            else id=false;

                                        }while(correct!=true);

                                        string Medicine=PosNode->Medic[id];

*int* realID=-1;

                                            for(*int* i=0;i<recno;i++){

                                                if(Medicine==MedicStack[i]){

                                                    realID=i;

                                                    break;

                                                }

                                            }

                                        if(realID!=-1){

                                            QuantStack[realID]+=PosNode->Quant[id];

                                            PosNode->totalPrice=PosNode->totalPrice-PosNode->Price[id];

                                        for(*int* i=id; i<PosNode->top;i++)

                                        {

                                            PosNode->Medic[i]=PosNode->Medic[i+1];

                                            PosNode->Quant[i]=PosNode->Quant[i+1];

                                            PosNode->Price[i]=PosNode->Price[i+1];

                                        }

                                        PosNode->top--;

                                        cout<<"\t\t\t\tNOTE :\n\t\t\t\t Medicince Removed Sucessfully\n\t\t\t\tMoney Transfer to your Account..\n";

                                        cout<<endl;

                                            }

                                            else{

                                                cout<<"\n\t\t\t\tNot Validate to return or Update this Medicince ......\n";

                                            }

                                    }

                                    else if(merzi=='a'){

*int* code;

*bool* check=false;

*int* countAttempt=0;

                                            while(check!=true){

                                                countAttempt++;

                                                if(countAttempt==1){

                                                cout<<"\t\t\t\tEnter Medicine code : "; cin>>code;

                                                }else{

                                                    cout<<"\t\t\t\tEnter \*Validate Medicine code : "; cin>>code;

                                                }

                                                check=CorrectID(code);

                                            }

                                            PosNode->Medic[PosNode->top]=MedicStack[code];

                                            check=false;

*int* q;

                                            while(check!=true){

                                                 cout<<"\t\t\t\tEnter Quantity of "<<MedicStack[code]<<" : "; cin>>q;

                                                 check=CorrectQuantity(code,q);

                                            }

                                                    PosNode->Quant[PosNode->top]=q;

                                                    QuantStack[code]=QuantStack[code]-q;

                                                    PosNode->Price[PosNode->top]=q\*PriceStack[code];

                                                    PosNode->totalPrice+=PosNode->Price[PosNode->top];

                                                    PosNode->top++;

                                         cout<<"\t\t\t\tAdded Successfully......\n";

                                    }

                                    else{

                                        cout<<"\t\t\tPlease press (a) for add, (r) for remove and (e) EXIT only.....\n";

                                    }

                            }while(merzi!='e');

                          break;

                        }

                        case 'q' :

                        {

*char* merzi;

                         do{

                            cout<<"\n\t\t\t(a)Add or (r)Remove Medicine Quantity (e)EXIT : "; cin>>merzi;

                            if(merzi=='r'){

*int* id;

*bool* correct=false;

                                DisplayNodeArray(PosNode);

                                do{

                                    cout<<"\t\t\t\t\t\t\tEnter id of Medicince to remove Quantity : ";  cin>>id;

                                    if(id<PosNode->top) correct=true;

                                    else id=false;

                                }while(correct!=true);

*int* items;

*bool* check=false;

                                do{

                                    cout<<"\t\t\t\tEnter Quantity of Medicine remove : "; cin>>items;

                                    if(items<=PosNode->Quant[id]){

                                        check=true;

                                    }

                                }while(check!=true);

*int* Remaining=PosNode->Quant[id]-items;

                                if(Remaining==0)

                                {

                                     string Medicine=PosNode->Medic[id];

*int* realID=-1;

                                            for(*int* i=0;i<recno;i++){

                                                if(Medicine==MedicStack[i]){

                                                    realID=i;

                                                    break;

                                                }

                                            }

                                    if(realID!=-1){

                                            QuantStack[realID]+=PosNode->Quant[id];

                                            PosNode->totalPrice=PosNode->totalPrice-PosNode->Price[id];

                                        for(*int* i=id; i<PosNode->top;i++)

                                        {

                                            PosNode->Medic[i]=PosNode->Medic[i+1];

                                            PosNode->Quant[i]=PosNode->Quant[i+1];

                                            PosNode->Price[i]=PosNode->Price[i+1];

                                        }

                                        }else

                                        {

                                             cout<<"\n\t\t\t\ttNot Validate for return or Update ......\n";

                                        }

                                }else{

                                PosNode->totalPrice=PosNode->totalPrice-(items\*(PosNode->Price[id]/PosNode->Quant[id]));

                                PosNode->Price[id]=PosNode->Price[id]-(items\*(PosNode->Price[id]/PosNode->Quant[id]));

                                PosNode->Quant[id]=Remaining;

                                }

                                DisplayNodeArray(PosNode);

                            }else if(merzi=='a'){

*int* id;

*bool* correct=false;

                                do{

                                    cout<<"\t\t\tEnter id of Medicince to Add Quantity : ";  cin>>id;

                                    if(id<PosNode->top) correct=true;

                                    else id=false;

                                }while(correct!=true);

*int* items;

*bool* check=false;

                                do{

                                    cout<<"\t\t\t\tEnter Quantity of Medicine to add : "; cin>>items;

                                    check=CorrectQuantity(id,items);

                                }while(check!=true);

                                string Medicine=PosNode->Medic[id];

*int* realID=-1;

                                            for(*int* i=0;i<recno;i++){

                                                if(Medicine==MedicStack[i]){

                                                    realID=i;

                                                    break;

                                                }

                                            }

                                if(realID!=-1){

                                PosNode->totalPrice=PosNode->totalPrice-(items\*PriceStack[realID]);

                                PosNode->Price[id]=PosNode->Price[id]-(items\*PriceStack[realID]);

                                PosNode->Quant[id]+=items;

                            }

                            else{

                               cout<<"\n\t\t\t\tNot Validate for return or Update ......\n";

                            }

                        }

                         }while (merzi!='e');

                          break;

                        }

                    case 'e' : {

                    }

                    default:

                        {cout<<"\t\t\t\tPlease select \*valid OPtion Only......\n";

                        break;}

                    }

                 }while(mchoice!='e');

             }

         }

        }while(choice!='e');

     }

    else{

        cout<<"\t\t\t\tList already is Empty\n";

}

 SaveData();

 SaveCustomerData();

  system("CLS");

  AdminMenu();

}

**FUNCTION TO GET ID TO REMOVE OR DELETE A CUSTOMER DATA:**

*void* RemovePosition(){

*int* realID=0;

    node\*ptr;

    ptr=head;

*bool* checked=false;

*int* CountAttempt=0;

    system("CLS");

    CustomersSummary();

    do{

        CountAttempt++;

*int* id;

        if(CountAttempt==1){

        cout<<"\t\t\t\tEnter Customer ID to DELETE DATA(-1/Exit) :  "; cin>>id;

        }

        else{

        cout<<"\t\t\t\tEnter \*Valid Customer ID to UPDATE DATA(-1/Exit) :  "; cin>>id;

        }

        if(id!=-1){

*int* count=0;

                 while(ptr!=NULL){

                    count++;

                    if(ptr->id==id){

                        cout<<ptr->id;

                        realID=count;

                    }

                if(id<=count&&id>0){

                    checked=true;

                }

                else{

                    checked==false;

                }

                ptr=ptr->next;

                }

        }else{

            break;

        }

    }while(checked!=true);

            if(checked==true){

                DeletePosition(realID);

            }

    system("CLS");

    AdminMenu();

}

**FUCTION TO DELETE A CUSTOMER DATA USING POSITION GETTING BY REMOVEPOSITION FUNCTION AND THIS ONE FUNCTION IS CONNECTED WITH DELETE BEGIN AND DELETE END:**

*void* DeletePosition(*int* *pos*){

    node\*PrevNode,\*PosNode;

    PosNode=head;

   if(PosNode!=NULL){

            if(*pos*==1){

                DeleteBegin();

            }

            else{

                for(*int* i=1;i<*pos*;i++){

                    PrevNode=PosNode;

                    PosNode=PosNode->next;

                }

                if(PosNode->next==NULL){

                    DeleteEnd();

                }

                else{

                    cout<<"\t\t\tSucessfully Deleted :"<<PosNode->CustomerName<<" \n";

                    PrevNode->next=PosNode->next;

                    PosNode->next->prev=PrevNode;

                    delete PosNode;

                }

            }

   }

   else{

       cout<<"\tList already is Empty\n";

   }

   SaveCustomerData();

}

**DELETE FROM BEGIN:**

*void* DeleteBegin(){

    node\* ptr=head;

    if(head==NULL){

        cout<<"\tAlready Empty...\n";

    }

    else{

        // For Additional Bug

        if(ptr->next==NULL){

            cout<<"\tA Sucessfully Deleted :"<<ptr->CustomerName<<" \n";

            delete ptr;

            head=NULL;

        }

        else{

            // Main to learn

             cout<<"\t Sucessfully Deleted :"<<ptr->CustomerName<<" \n";

            head=ptr->next;

            delete ptr;

            head->prev=NULL;

        }

    }

}

**DELETE FROM END:**

*void* DeleteEnd(){

    if(head==NULL){

        cout<<"\tAlready Empty...\n";

    }

    else{

        node\* ptr;

        ptr=head;

        // This if/Else Additional for Additional bug

        if(ptr->next==NULL){

            cout<<"\t\t\t\tSucessfully Deleted :"<<ptr->CustomerName<<" \n";

            delete ptr;

            head=NULL;

        }

        else

        {

        // Main to learn :

                while(ptr->next!=NULL){

                    ptr=ptr->next;

                }

                cout<<"\t\t\t\tSucessfully Deleted :"<<ptr->CustomerName<<" \n";

                ptr->prev->next=NULL;

                ptr->prev=NULL;

                delete ptr;

        }

    }

}

**DISPLAY CUSTOMER DATA SUMMARY:**

*void* CustomersSummary(){

    node\*ptr=head;

    if(ptr!=NULL){

        system("CLS");

        cout<<"\t\t\t\t------------CUSTOMERS SUMMARY-------------\n";

        cout<<"\t\t\t\t ID          NAME          TIMMING\n";

            while(ptr!=NULL){

        cout<<"\t\t\t\t  "<<ptr->id<<" -> "<<ptr->CustomerName<<" -> "<<ptr->date<<endl;

        ptr=ptr->next;

            }

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

    }

}

**DISPLAY FULL CUSTOMER DELTAIL FOR ALL :**

*void* Display(){

    node\* ptr=head;

    if(ptr!=NULL){

    cout<<"\t\t\t\t\t\_\_\_\_\_\_CUSTOMER DATA\_\_\_\_\_\_ \n";

    while (ptr!=NULL)

    {

        cout<<"\t\t\t\t\tID:"<<ptr->id<<") \n";

        cout<<"\t\t\t\t\t   NAME       : "<<ptr->CustomerName<<" \n";

        cout<<"\t\t\t\t\t   Date       : "<<ptr->date<<" \n";

        cout<<"\t\t\t\t\t   MEDIC/QUANTITY/PRICE->\n";

        for(*int* i=0;i<ptr->top;i++){

            cout<<"\t\t\t\t\t   "<<i<<")"<<ptr->Medic[i]<<"    "<<ptr->Quant[i]<<"    "<<ptr->Price[i]<<endl;

        }

        cout<<"\t\t\t\t\t  Total Price : "<<ptr->totalPrice<<" \n";

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

        ptr=ptr->next;

    }

    }

    else{

        cout<<"\t\t\t\t\t\tNo Customer Record Found....Sorry\n";

    }

}

DISPLAY FOR EACH CUSTOMER:

*void* DisplayNodeArray(*struct* node \**temp*){

    for(*int* i=0;i<*temp*->top;i++){

        cout<<"\t\t\t\t"<<i<<")   Medic : "<<*temp*->Medic[i]<<"   Price : "<<*temp*->Price[i]<<"   Quantity : "<<*temp*->Quant[i]<<endl;

    }

}

**SEARCHING THE CUSTOMER DATA FOR UPDATA OR DELETE OR TO VIEW ONLY :**

*void* Searching(){

*bool* check=false;

*int* count=0;

 do{

     count++;

     string name;

     if(count==1){

    cin.ignore();

    cout<<"\t\t\t\t\tEnter Customer Name : ";   getline(cin,name);

    }

    else{

        cout<<"\t\t\t\t\tEnter \*Valid Customer Name : ";  getline(cin,name);

    }

*int* check=-1;

    node\*ptr;

    ptr=head;

*int* count=0;

    while (ptr!=NULL)

    {

        count++;

        if(ptr->CustomerName==name){

            check=count;

            break;

        }

        ptr=ptr->next;

    }

    if(check!=-1){

                cout<<"\t\t\t\t\t--------------"<<ptr->CustomerName<<"--------------\n";

                cout<<"\t\t\t\t\t           Costumer ID     : "<<ptr->id<<"                  \n";

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

                cout<<"\t\t\t\t\t   MEDICINE / QUANTITY  / PRICE     \n";

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

                for(*int* i=0;i<ptr->top;i++){

                cout<<"\t\t\t\t\t"<<i+1<<")  "<<ptr->Medic[i]<<"    "<<ptr->Quant[i]<<"    "<<ptr->Price[i]<<endl;

                }

                cout<<"\t\t\t\t\t           Total  Price      : "<<ptr->totalPrice<<"      \n";

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

                cout<<"\t\t\t\tYou Want to Edit this Record(y/n) : "; *char* merzi; cin>>merzi;

                merzi=(*char*)tolower(merzi);

                if(merzi=='y'){

                    UpdateOrder(check);

                }

                else{

                    system("CLS");

                    ManagerMenu();

                    break;

                }

        check=true;

        }

        else{

        cout<<"\t\t\tCustomer is not found...\n";

        check=false;

        }

    }while(check!=true);

}

**GET CORRECT Quantity FROM USER:**

*bool* CorrectQuantity(*int* *id*,*int* *quant*){

*bool* check=false;

    if(*quant*>0&&*quant*<=QuantStack[*id*]){

        check=true;

    }

    else{

        check=false;

        cout<<"\nQuantity must less than "<<QuantStack[*id*]<<endl;

    }

    return check;

}

**GET CORRECT ID FROM USER :**

*bool* CorrectID(*int* *id*){

*bool* check=false;

        if(*id*>=0&&*id*<recno){

           check=true;

         }

        else{

           check=false;

           cout<<"\nValidate ID must be less than"<<recno<<endl;

         }

    return check;

}

**FETCH ALL CUSTOMER DATA FROM FILE:**

*oid* FetchCustomerData(){

    string line;

*int* num;

*int* countLine=0;

*int* countCustomer=1;

*int* countMedic=0;

*int* initialTop=0;

    ifstream records\_input("CustomerRecord.txt");

*int* medic=0,quant=0,price=0;

*int* top=0;

*int* lastLine=0;

*struct* node\*ptr;

    while (getline(records\_input, line)) // count all the lines from the file

    {

        countLine++;// this one count the lines from my file...

            if(countLine==1){

                medic=6;

                quant=7;

                price=8;

                ptr=new node();// cout<<"probelem"<<line<<endl;

                istringstream ( line ) >> num;

                ptr->id=num; // cout<<ptr->id<<endl;

            }

            else if(countLine==2){// cout<<"countLine : "<<countLine<<endl;

                string name=line; // cout<<line<<endl;

                ptr->CustomerName=name;

            }

            else if(countLine==3){

                ptr->date=line;

            }

            else if(countLine==4){

                istringstream ( line ) >> top;

                ptr->top=top;

            }

            else if(countLine==5){

            }

            else if(initialTop<top){

                            if(countLine==medic){

                                string medicname=line;

                                ptr->Medic[initialTop]=medicname;

                                medic=medic+3;

                            }

                            else if(countLine==quant){

                                istringstream ( line ) >> num;

                                ptr->Quant[initialTop]=num;

                                quant+=3;

                            }

                            else if(countLine==price){

                                istringstream ( line ) >> num;

                                ptr->Price[initialTop]=num;

                                price+=3;

                                initialTop++;

                            }

              }

              else  if(line=="Total Price : "){

              }

              else if(line=="--------------------"){

              }

              else{

*double* total=0;

                      istringstream ( line ) >> total;

                      ptr->totalPrice=total;

              }

        if(countCustomer==1&&line=="--------------------"){

            countLine=0;

            initialTop=0;

            countCustomer++;

            head=ptr;

            ptr->next=NULL;

            ptr->prev=NULL;

            tail=ptr;

            }

        else if(countCustomer>1&&line=="--------------------"){

            countLine=0;

            initialTop=0;

            countCustomer++;

            tail->next=ptr;

            ptr->prev=tail;

            ptr->next=NULL;

            tail=ptr;

        }

        }

 records\_input.close();

    }

**TO SAVE CUSTOMER DATA IN FILE :**

*void* SaveCustomerData(){

    ofstream records\_output("CustomerRecord.txt"); // create a file stream and open the file for output

*struct* node\*ptr;

    ptr=head;

    if(ptr==NULL){

        cout<<"\n\t\t\t\tNo Data to Store..........\n\n";

    }

    else{

        while(ptr!=NULL){

            records\_output<<ptr->id<<endl;

            records\_output<<ptr->CustomerName<<endl;

   records\_output<<ptr->date<<endl;

            records\_output<<(ptr->top)<<endl;

            records\_output<<"Medicince Detail : "<<endl;

            for(*int* i=0;i<ptr->top;i++){

                records\_output<<ptr->Medic[i]<<endl;

                records\_output<<ptr->Quant[i]<<endl;

                records\_output<<ptr->Price[i]<<endl;

            }

            records\_output<<"Total Price : "<<endl;

            records\_output<<ptr->totalPrice<<endl;

            records\_output<<"--------------------"<<endl;

            ptr=ptr->next;

        }

    }

    records\_output.close(); // close the file

}

# FUTURE WORK:

We tried our best to implement everything we could in our program. but this pharmacy management system program, no matter how much you think or code on this software something always lacks because of the new ideas in everyday life. What we think we could have added more but due to the shortage of time we couldn’t, in the future or as a future work we would like to add some more function into our program that would make the program more subtle and efficient. Some of those functions are written here:

## Expiry date: we can add expiry dates for the medicines on what date it would expire.

## Validation: we can also add validation for the prescription that doctors give to their patients.

## User interface: we could make the whole program user interface but because of the shortage of time we couldn’t.

# CONCLUSION:

Pharmacy management system is actually a software which handle the essential data and save the. This software helps in effectively management of the pharmaceutical store or shop. It provides the statistics about medicine or drugs which are in stocks which data can also be updated and edited. It works as per the requirement of the user and have options accordingly. It allows user to enter manufacturing as well as the expiry date of medicine placing in stock and for sales transaction. This software also has ability to print reports and receipts etc. There is other function available too. The main purpose is effectively and easily handling of pharmacy data and its management.