**Pharmacy Management System**



Session: 2022 – 2026

**Submitted by:**

Muhammad Taha Saleem 2022-CS-139

**Submitted to:**

Dr. Awais Hassan.

Department of Computer Science

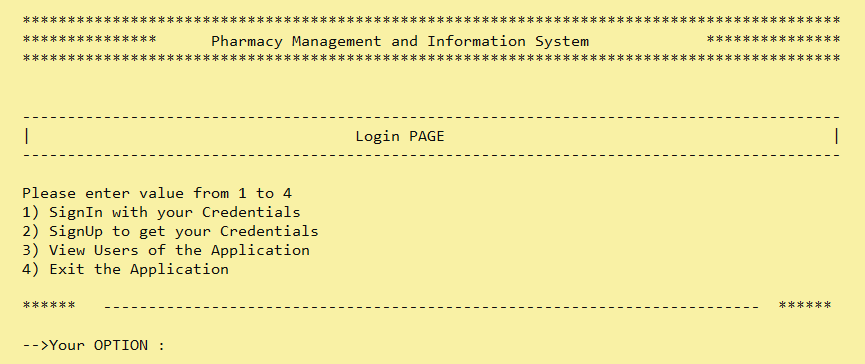
**University of Engineering and Technology**

**Lahore Pakistan**

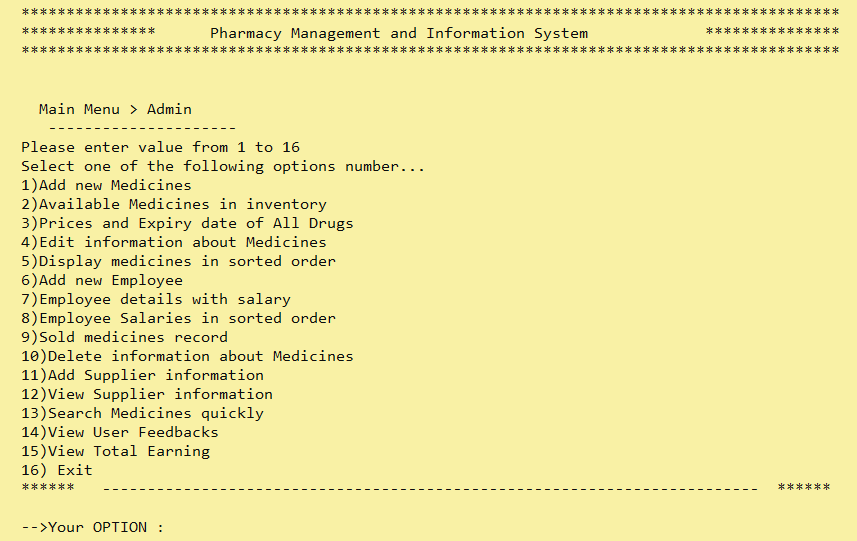
* **Short Description of project**
* This is actually a Pharmacy Management system in which the admin can add the Pharmacy Medicines and Staff etc. for his Medical Store. While Customers can view all medicines, buy medicines. Computer Science has made it possible that customers can easily buy medicines according their disease or other issues. By using C++ we can made Pharmacy medicine system application.
* **Users of Application** 
  + Basically there are there users of this application. Some common examples of project users include:
  1. Customers or clients
  2. Admin
* **Functional Requirements**

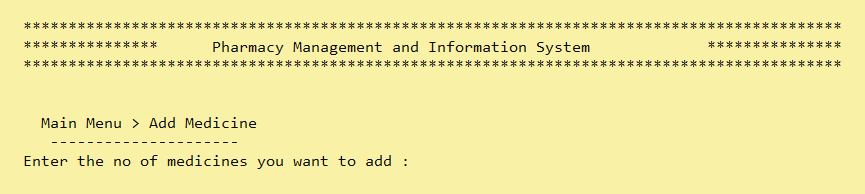
|  |  |  |  |
| --- | --- | --- | --- |
| User Story ID | *As a* | *I want to* | *So that I can* |
| **1** | Admin | Add medicine. | Keep them for my store in inventory |
| Show inventory. | So that I can show my all medicines. |
| Show prices and expiry. | To keep abreast of them. |
| Use edit Function. | Edit medicine data. |
| Sort. | Sort medicines according to their prices. |
| Add employee. | To add a employee for my Pharmacy to work as admin. |
| Show employee details | See employee information. |
| Sort salaries. | To view salaries in sorted order. |
| Add Achievements. | Show Accomplishments of the club to fans. |
| Set the ticket prices. | Change the price of tickets. |
| **2** | User | Schedule a training session. | Schedule a training session. |
| Check Team. | See the players and their attributes. |
| Change playing XI. | Show playing XI to other users. |
| Add a Player. | Add a player to the club. |
| Remove a Player. | Remove a player from the club. |
| Check Schedule. | Plan according to it. |
| Check the statistics of players. | Track the players performance. |
| Check Achievements. | See Accomplishments of the club. |
| Check Training Equipment. | Add/Remove the equipment. |
| **3** | Player | Check the statistics of players. | Track the players performance. |
| Check Achievements. | See Accomplishments of the club. |
| Check Schedule. | Plan according to it. |
| Check Playing XI | See playing XI |
| **4** | Fans | Check the statistics of players. | Check Players Performance. |
| Check Achievements. | Check Achievements. |
| Check Schedule. | Check future matches. |

* **Wireframes**

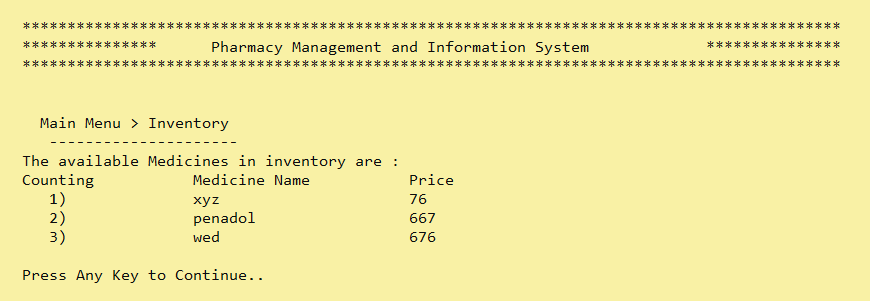


**Figure 1: Log in Screen**

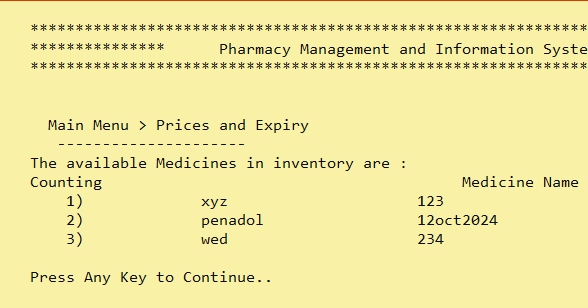
**Figure 2: Admin Main Menu Screen**



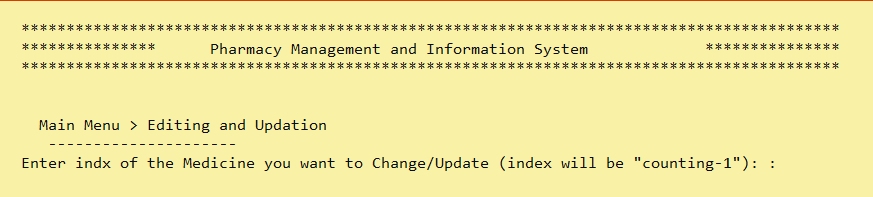
**Figure 3: Admin Option 1**

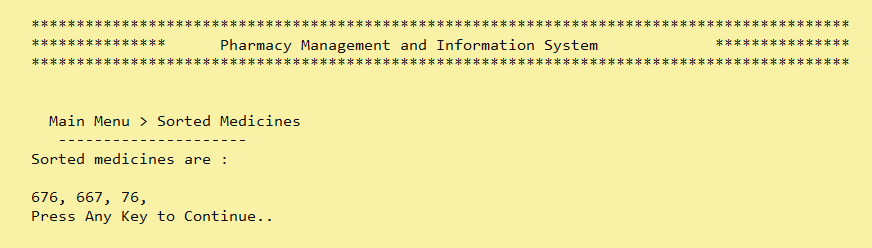


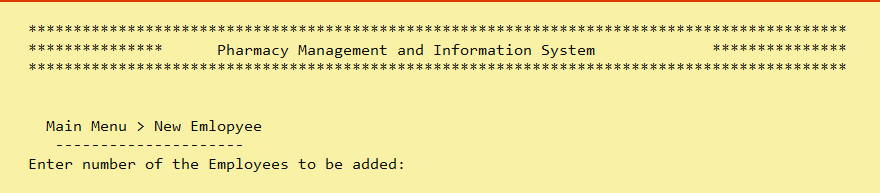
**Figure 4: Admin Option 2**



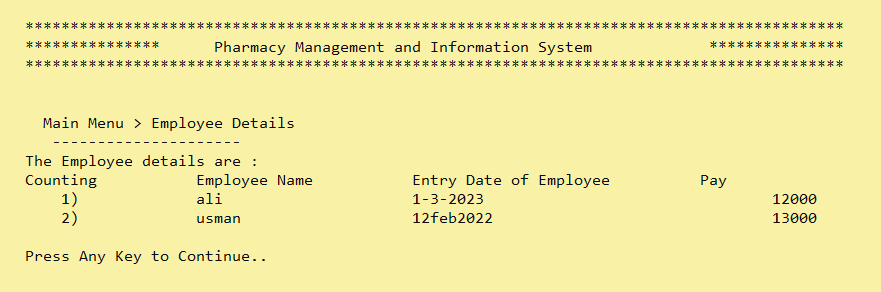
**Figure 5: Admin Option 3**

**Figure 6: Admin Option 4**

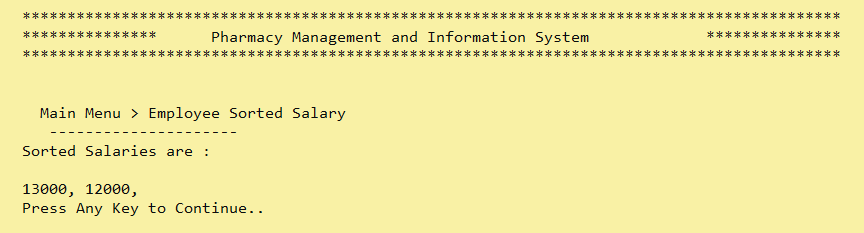
**Figure 7: Admin Option 5**



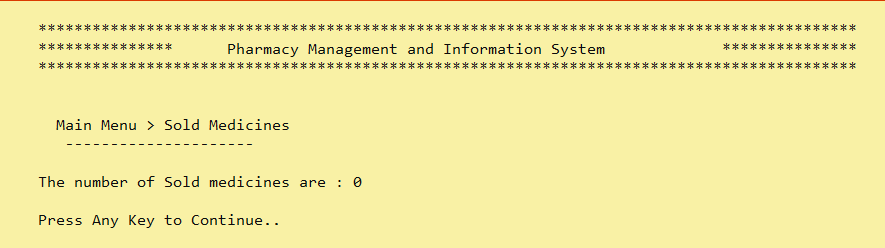
**Figure 8: Admin Option 6**



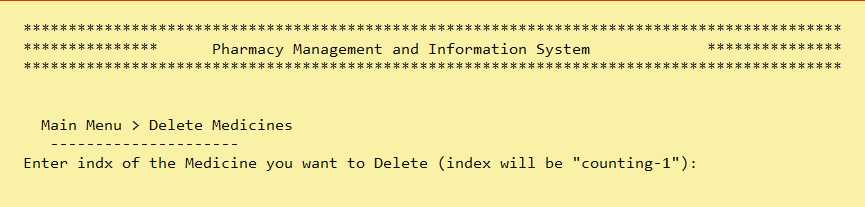
**Figure 9: Admin Option 7**



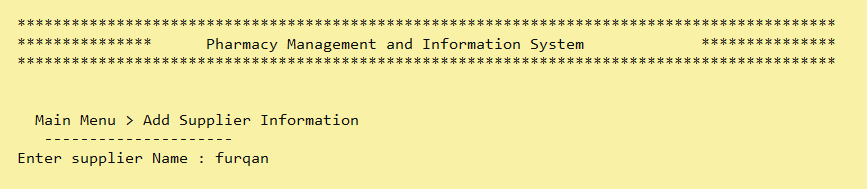
**Figure 10: Admin Option 8**

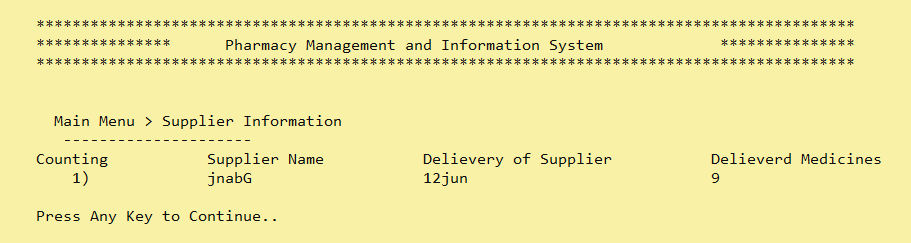


**Figure 11: Admin Option 9**

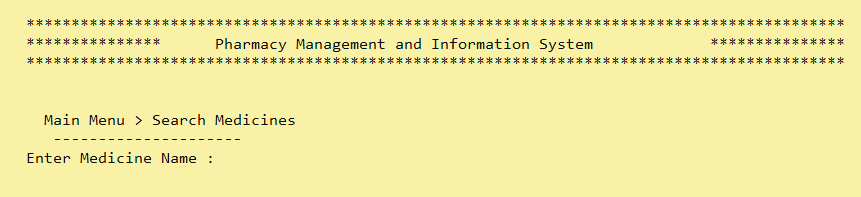


**Figure 12: Admin Option 10**

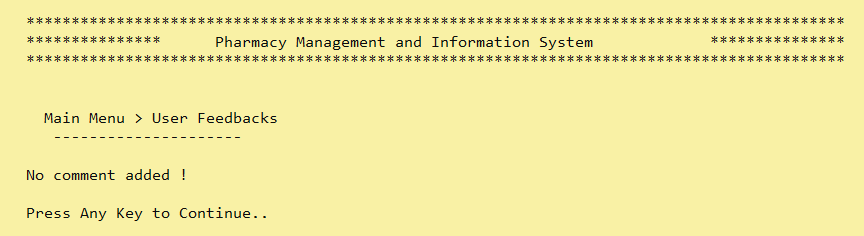
**Figure 13: Admin Option 11**

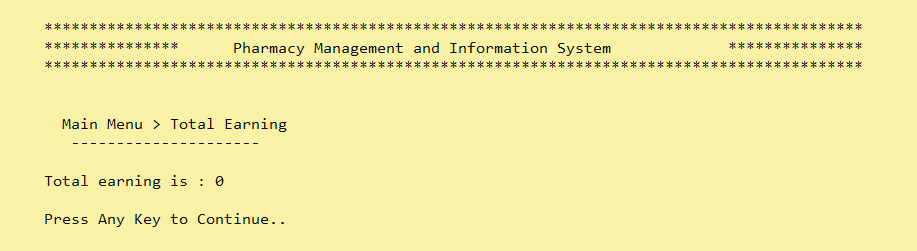


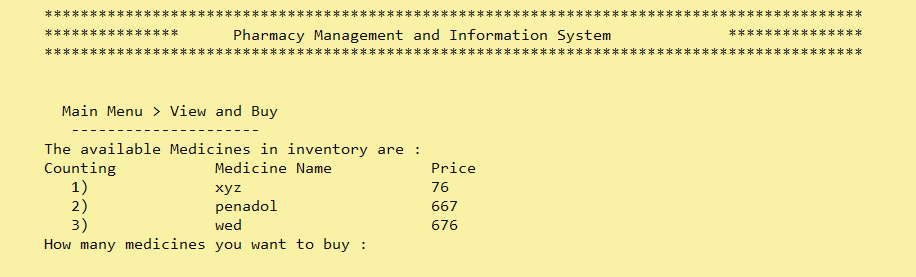
**Figure 14: Admin Option 12**



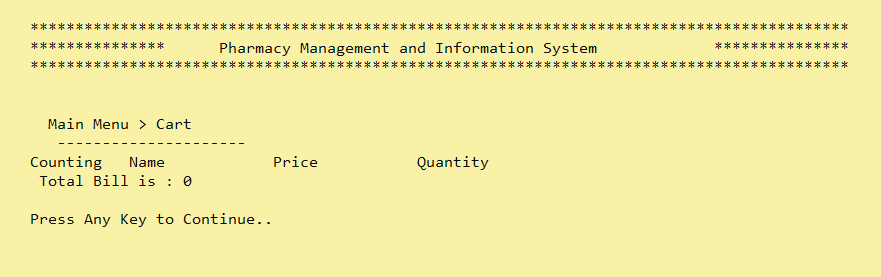
**Figure 15: Admin option 13**

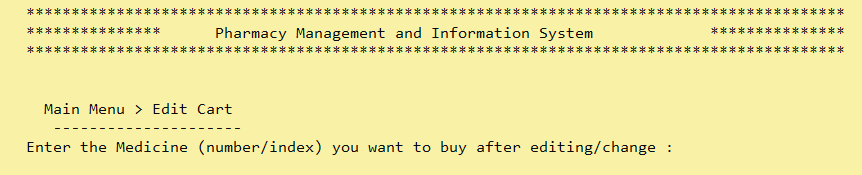
** Figure 16: Admin option 14**

** Figure 17: Admin option 15**

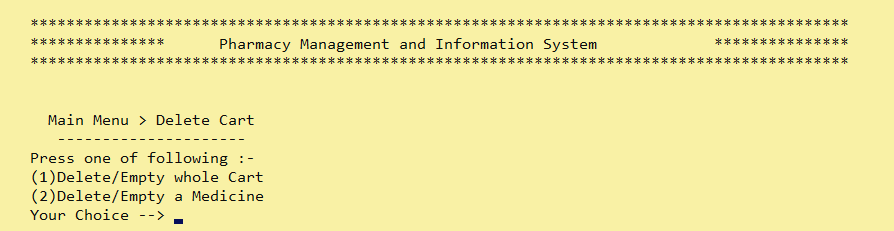


**Figure 18: User Option 1**

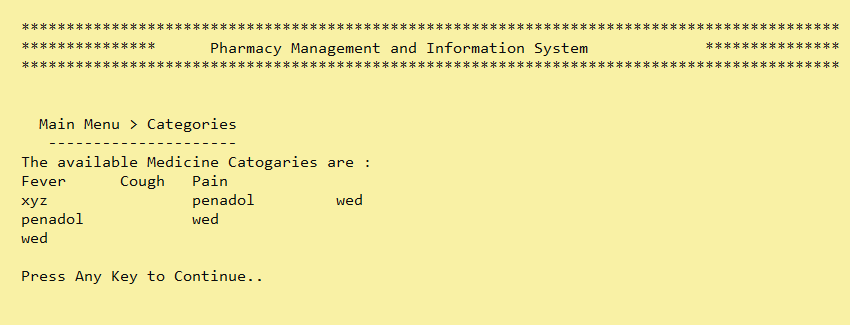
**Figure 19:User Option 2**

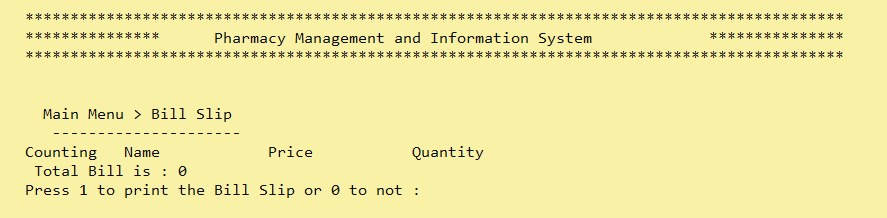


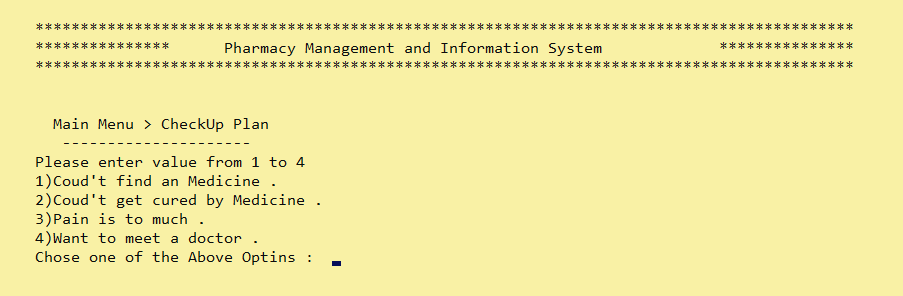
**Figure 20: User Option 3**



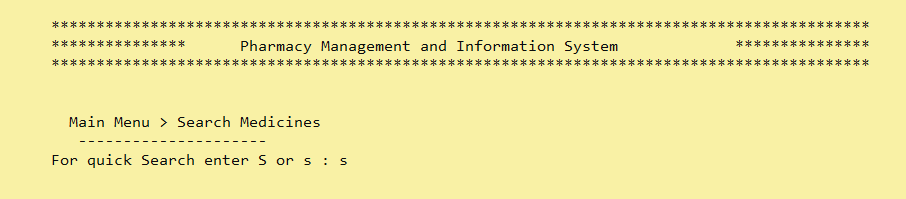
**Figure 21: User Option 4**

**Figure 22: User Option 5**

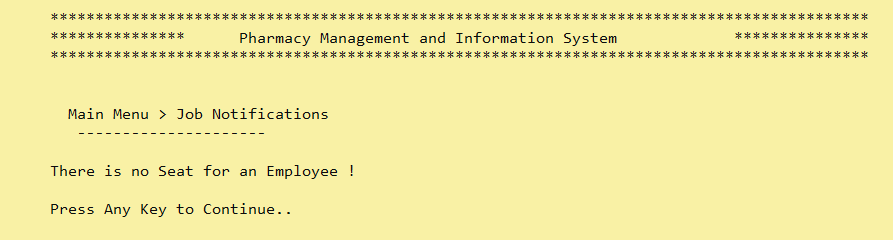
**Figure 23: User Option 6**



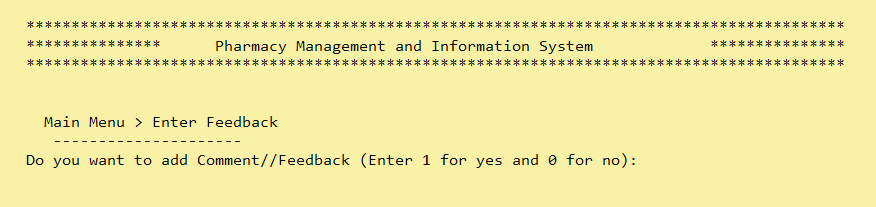
**Figure 24: User Option 7**

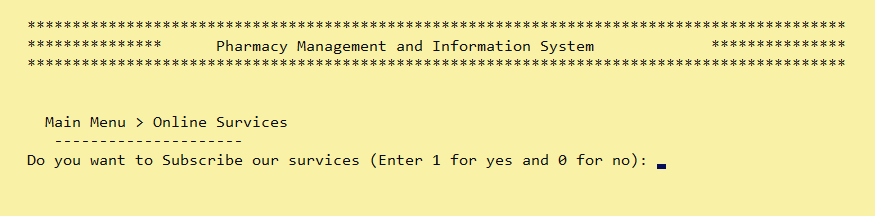


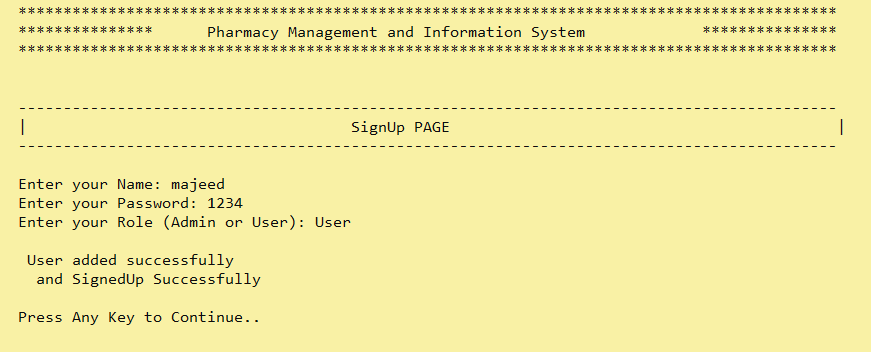
**Figure 25: User Option 8**

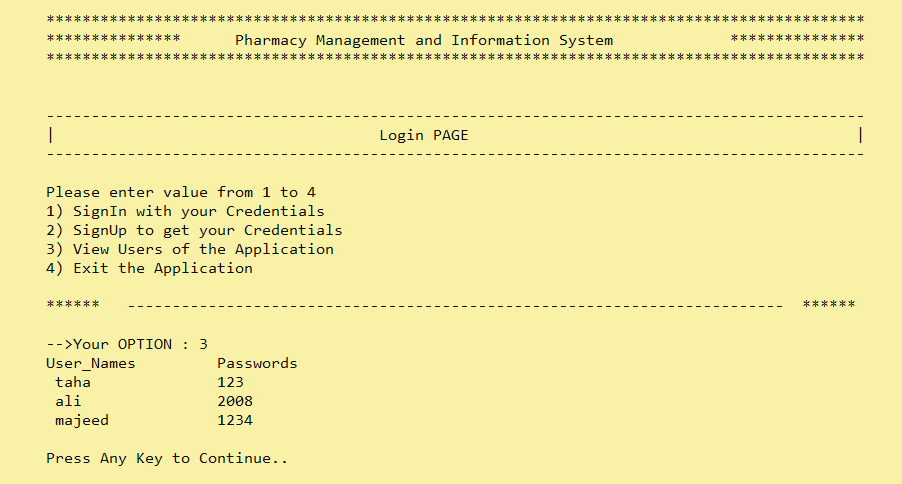


**Figure 26: User Option 9**

**Figure 27: User Option 10**

**Figure 28: User Option 11**

**Figure 29: Sign UP page**

**Figure 30: Users of App. Page**

* **Data Structures (Parallel Arrays)**
* • int medicineCount;
* • int usersCount = 0;
* • int employeeCount = 0;
* • int counting;
* • int soldMedicinesCount = 0;
* • int indx;
* • int totalBill = 0;
* • int earning;
* • const int userArrSize = 100;
* • const int size = 100;
* • const int totalFeedbacks = 100;
* • int feedbacksCount = 0;
* • int noOfMedicinestoBuy = 0;
* • int quantityOfSpecifiedMedicine[size];
* • int totalCash;
* • bool optiontoAddcomment = 0;
* • string Comment;
* • string Comments[totalFeedbacks];
* • int numOfFeedbacks = 1;
* • bool isStoreAble = true;
* • string users[userArrSize];
* • string passwords[userArrSize];
* • string roles[userArrSize];
* • int numberOfsupplier = 1;
* • string medicineName[size];
* • string expiryDate[size];
* • int prices[size];
* • string employeeName[size];
* • string entryDate[size];
* • int employeeSalary[size];
* • string supplierName[size];
* • string delieveryDate[size];
* • int quantityofMedicineSupplied[size];
* • int cart[size];
* • Function Prototypes
* • void topHeader();
* • void subMenuBeforeMainMenu(string submenu);
* • void subMenu(string submenu);
* • int loginMenu();
* • string signIn(string name, string password);
* • int adminMenu();
* • void adminInterface();
* • int userMenu();
* • void userInterface();
* • void clearScreen();
* • void optionLimit(int minValue, int maxValue);
* • int validateInteger();
* • void viewUsers();
* • bool isValid(string name);
* • void inputMedicine(int medicineCount);
* • void showInventory(int medicineCount);
* • void showPricesNExpiry(int medicineCount);
* • void updateInventory();
* • void addEmployee(int employeeCount);
* • void showEmployeeDetails(int employeeCount);
* • int calcSoldMedicines(int medicineCount);
* • void viewSoldMedicine(int soldGoods);
* • void deleteCartMediicine();
* • void deleteMedicines();
* • void addSupplierDetails();
* • void showSupplierDetails();
* • void searchMedicine(int medicineCount);
* • void showRemarks();
* • void showSortedEmployeeSalaries(int employeeCount);
* • int employeelargest(int s);
* • void showSortedmedicines(int medicineCount);
* • int medicinelargest(int s);
* • int calcEarning(int medicineCount);
* • void viewEarning(int totalEarning);
* • void viewNBuyMedicine();
* • void viewCartandTotal();
* • void editCart();
* • int deleteCart();
* • void differentDrugCategories();
* • void printBillSlip();
* • void checkUpPlan();
* • void UsearchMedicine();
* • void jobNotifications();
* • void addUserASEmployee();
* • void userFeedback();
* • void onlineSurvices();
* • bool load\_data(string name, string password, string role);
* • void load\_commebts\_inArr(string Comment);
* • void store\_comments();
* • void store\_suppliers();
* • void store\_employees();
* • void store\_medicines()
* • void store\_data();
* • void readComments();
* • void readSupplierData();
* • void readEmployeeData();
* • void readMedicinesData();
* • void readData();
* • string getField(string record, int field);
* **Functions Working Flow**
* Admin Interface
* User Interface
* **Complete Code of the Business Application**

#include <iostream>

#include <conio.h>

#include <fstream>

#include <string>

using namespace std;

int medicineCount;

int usersCount = 0;

int employeeCount = 0;

int counting;

int soldMedicinesCount = 0;

int indx;

int totalBill = 0;

int earning;

const int userArrSize = 100;

const int size = 100;

const int totalFeedbacks = 100;

int feedbacksCount = 0;

int noOfMedicinestoBuy = 0;

int quantityOfSpecifiedMedicine[size];

int totalCash;

bool optiontoAddcomment = 0;

string Comment;

string Comments[totalFeedbacks];

int numOfFeedbacks = 1;

bool isStoreAble = true;

string users[userArrSize];

string passwords[userArrSize];

string roles[userArrSize];

int numberOfsupplier = 1;

string medicineName[size];

string expiryDate[size];

int prices[size];

string employeeName[size];

string entryDate[size];

int employeeSalary[size];

string supplierName[size];

string delieveryDate[size];

int quantityofMedicineSupplied[size];

int cart[size];

string getField(string record, int field)

{

int commaCount = 1;

string item;

for (int idx = 0; idx < record.length(); idx++)

{

if (record[idx] == ',')

{

commaCount++;

}

else if (commaCount == field)

{

item = item + record[idx];

}

}

return item;

}

void readData()

{

string record;

fstream Credential;

Credential.open("Credentials.txt", ios::in);

while (getline(Credential, record))

{

users[usersCount] = getField(record, 1);

passwords[usersCount] = getField(record, 2);

roles[usersCount] = getField(record, 3);

usersCount++;

// cycleCount++;

}

Credential.close();

}

void readMedicinesData()

{

string record;

fstream file;

file.open("Medicine Details.txt", ios::in);

while (getline(file, record))

{

medicineName[medicineCount] = getField(record, 1);

expiryDate[medicineCount] = getField(record, 2);

try

{

prices[medicineCount] = stoi(getField(record, 3));

}

catch (const invalid\_argument &e)

{

// handle the error

}

medicineCount++;

}

file.close();

}

void readEmployeeData()

{

string record;

fstream file;

file.open("Employee Details.txt", ios::in);

while (getline(file, record))

{

employeeName[employeeCount] = getField(record, 1);

entryDate[employeeCount] = getField(record, 2);

try

{

employeeSalary[employeeCount] = stoi(getField(record, 3));

}

catch (const invalid\_argument &e)

{

// handle the error

}

employeeCount++;

}

file.close();

}

void readSupplierData()

{

string record;

fstream file;

file.open("Suppliers Details.txt", ios::in);

while (getline(file, record))

{

supplierName[numberOfsupplier] = getField(record, 1);

delieveryDate[numberOfsupplier] = getField(record, 2);

try

{

quantityofMedicineSupplied[numberOfsupplier] = stoi(getField(record, 3));

}

catch (const invalid\_argument &e)

{

// handle the error

}

}

file.close();

}

void readComments()

{

string record;

fstream Comment;

Comment.open("Comments.txt", ios::in);

while (getline(Comment, record))

{

Comments[feedbacksCount] = getField(record, 1);

feedbacksCount++;

}

Comment.close();

}

void store\_data()

{

fstream Credential;

Credential.open("Credentials.txt", ios::app);

Credential << users[usersCount] << "," << passwords[usersCount] << "," << roles[usersCount] << "\n";

Credential.close();

}

void store\_medicines()

{

fstream file;

file.open("Medicine Details.txt", ios::out);

for (int idx = 0; idx < medicineCount; idx++)

{

file << medicineName[idx] << "," << expiryDate[idx] << "," << prices[idx] << "\n";

}

file.close();

}

void store\_employees()

{

fstream file;

file.open("Employee Details.txt", ios::out);

for (int idx = 0; idx < employeeCount; idx++)

{

file << employeeName[idx] << "," << entryDate[idx] << "," << employeeSalary[idx] << "\n";

}

file.close();

}

void store\_suppliers()

{

fstream file;

file.open("Suppliers Details.txt", ios::out);

// for (int idx = 0; idx < numberOfsupplier; idx++)

// {

file << supplierName[numberOfsupplier] << "," << delieveryDate[numberOfsupplier] << "," << quantityofMedicineSupplied[numberOfsupplier] << "\n";

// }

file.close();

}

void store\_comments()

{

fstream file;

file.open("Comments.txt", ios::out);

for (int idx = 0; idx < feedbacksCount; idx++)

{

file << Comments[idx] << ",";

}

file.close();

}

void load\_commebts\_inArr(string Comment)

{

cin.clear();

cin.ignore();

if (feedbacksCount < totalFeedbacks)

{

Comments[feedbacksCount] = Comment;

store\_comments();

feedbacksCount++;

}

}

bool load\_data(string name, string password, string role)

{

cin.clear();

cin.ignore();

if (usersCount < userArrSize)

{

users[usersCount] = name;

passwords[usersCount] = password;

roles[usersCount] = role;

store\_data();

usersCount++;

}

else

{

isStoreAble = false;

}

return isStoreAble;

}

bool isValid(string name)

{

bool flag = true;

for (int idx = 0; idx < usersCount; idx++)

{

if (users[idx] == name)

{

flag = false;

break;

}

}

return flag;

}

void viewUsers()

{

if (usersCount == 0)

{

cout << " No User Added!" << endl;

}

else

{

cout << " User\_Names"

<< "\t\t"

<< "Passwords" << endl;

for (int idx = 0; idx < usersCount; idx++)

{

cout << " " << users[idx] << "\t\t" << passwords[idx] << endl;

}

}

}

// FOR INTEGER VALIDATION

int validateInteger()

{

int ID;

char c;

while (true)

{

cin >> ID;

if (cin.fail())

{

cout << " Invaid input !\n Enter Valid Input : ";

cin.clear();

cin.ignore(123, '\n');

continue;

}

bool isNumeric = true;

while (cin.get(c) && c != '\n')

{

if (!isdigit(c))

{

isNumeric = false;

break;

}

}

if (isNumeric && ID > 0)

{

break;

}

}

return ID;

}

void optionLimit(int minValue, int maxValue)

{

cout << " Please enter value from " << minValue << " to " << maxValue << endl;

}

void topHeader();

void subMenuBeforeMainMenu(string submenu);

void subMenu(string submenu);

int loginMenu();

string signIn(string name, string password);

int adminMenu();

void adminInterface();

int userMenu();

void userInterface();

void clearScreen();

// -----------------------------

void inputMedicine(int medicineCount);

void showInventory(int medicineCount);

void showPricesNExpiry(int medicineCount);

void updateInventory();

void addEmployee(int employeeCount);

void showEmployeeDetails(int employeeCount);

int calcSoldMedicines(int medicineCount);

void viewSoldMedicine(int soldGoods);

void deleteCartMediicine();

void deleteMedicines();

void addSupplierDetails();

void showSupplierDetails();

void searchMedicine(int medicineCount);

void showRemarks();

void showSortedEmployeeSalaries(int employeeCount);

int employeelargest(int s);

void showSortedmedicines(int medicineCount);

int medicinelargest(int s);

int calcEarning(int medicineCount);

void viewEarning(int totalEarning);

// ----------------------------------

void viewNBuyMedicine();

void viewCartandTotal();

void editCart();

int deleteCart();

void differentDrugCategories();

void printBillSlip();

void checkUpPlan();

void UsearchMedicine();

void jobNotifications();

void addUserASEmployee();

void userFeedback();

void onlineSurvices();

main()

{

readData();

readMedicinesData();

readEmployeeData();

readSupplierData();

readComments();

system("cls");

int loginOption = 0;

while (loginOption != 4)

{

topHeader();

subMenuBeforeMainMenu("Login");

loginOption = loginMenu();

if (loginOption == 1)

{

system("cls");

string name;

string password;

string role;

topHeader();

subMenuBeforeMainMenu("SignIn");

cout << " Enter your Name: ";

cin >> name;

cout << " Enter your Password: ";

cin >> password;

role = signIn(name, password);

if (role == "Admin")

{

system("cls");

// clearScreen();

adminInterface();

}

else if (role == "User")

{

system("cls");

// clearScreen();

userInterface();

}

else if (role == "undefined")

{

cout << " You Entered wrong Credentials" << endl;

clearScreen();

}

}

else if (loginOption == 2)

{

system("cls");

string name;

string password;

string role;

topHeader();

subMenuBeforeMainMenu("SignUp");

cout << " Enter your Name: ";

cin >> name;

cout << " Enter your Password: ";

cin >> password;

cout << " Enter your Role (Admin or User): ";

cin >> role;

// bool isValid = signUp(name, password, role);

bool decision = isValid(name);

if (decision == true)

{

isStoreAble = load\_data(name, password, role);

// isStoreAble = load\_data(name, password, role);

cout << endl

<< " User added successfully " << endl;

// clearScreen();

}

else

{

cout << " Username already exists ! " << endl

<< endl;

// clearScreen();

}

if (isStoreAble)

{

cout

<< " and SignedUp Successfully" << endl;

}

else if (!isStoreAble)

{

cout << " Users in the System have exceeded the Capacity" << endl;

clearScreen();

}

}

else if (loginOption == 3)

{

viewUsers();

}

clearScreen();

}

}

int loginMenu()

{

int option;

optionLimit(1, 4);

cout << " 1) SignIn with your Credentials" << endl;

cout << " 2) SignUp to get your Credentials" << endl;

cout << " 3) View Users of the Application" << endl;

cout << " 4) Exit the Application" << endl;

cout << endl;

cout << " \*\*\*\*\*\* ------------------------------------------------------------------------- \*\*\*\*\*\*" << endl

<< endl;

cout << " -->Your OPTION : ";

option = validateInteger();

return option;

}

string signIn(string name, string password)

{

readData();

for (int index = 0; index < usersCount; index++)

{

if (users[index] == name && passwords[index] == password)

{

cout << " Login SuccessFul ! ";

return roles[index];

}

}

return " undefined";

}

void topHeader()

{

cout << "" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Pharmacy Management and Information System \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << " \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* " << endl

<< endl

<< endl;

}

void subMenuBeforeMainMenu(string submenu)

{

cout << " -------------------------------------------------------------------------------------------" << endl;

cout << " | " << submenu << " PAGE | " << endl;

cout << " -------------------------------------------------------------------------------------------" << endl

<< endl;

}

void subMenu(string submenu)

{

string message = " Main Menu > " + submenu;

cout << message << endl;

cout << " ---------------------" << endl;

}

void clearScreen()

{

cout << endl

<< " Press Any Key to Continue.." << endl;

getch();

system("cls");

}

int adminMenu()

{

int option;

optionLimit(1, 16);

cout << " Select one of the following options number..." << endl;

cout << " 1)Add new Medicines" << endl;

cout << " 2)Available Medicines in inventory" << endl;

cout << " 3)Prices and Expiry date of All Drugs" << endl;

cout << " 4)Edit information about Medicines" << endl;

cout << " 5)Display medicines in sorted order" << endl;

cout << " 6)Add new Employee" << endl;

cout << " 7)Employee details with salary" << endl;

cout << " 8)Employee Salaries in sorted order" << endl;

cout << " 9)Sold medicines record" << endl;

cout << " 10)Delete information about Medicines" << endl;

cout << " 11)Add Supplier information" << endl;

cout << " 12)View Supplier information" << endl;

cout << " 13)Search Medicines quickly" << endl;

cout << " 14)View User Feedbacks" << endl;

cout << " 15)View Total Earning" << endl;

cout << " 16) Exit" << endl;

cout << " \*\*\*\*\*\* ------------------------------------------------------------------------- \*\*\*\*\*\*" << endl

<< endl;

cout << " -->Your OPTION : ";

// cin >> option;

option = validateInteger();

return option;

}

void adminInterface()

{

int adminOption = 0;

while (adminOption != 16)

{

topHeader();

subMenu("Admin");

adminOption = adminMenu();

if (adminOption == 1)

{

system("cls");

topHeader();

subMenu("Add Medicine");

cout << " Enter the no of medicines you want to add : ";

medicineCount = validateInteger();

inputMedicine(medicineCount);

// Implementing the Add Student Functionality

}

if (adminOption == 2)

{

system("cls");

topHeader();

subMenu("Inventory");

showInventory(medicineCount);

// Implementing the Add Student Functionality

}

if (adminOption == 3)

{

system("cls");

topHeader();

subMenu("Prices and Expiry");

showPricesNExpiry(medicineCount);

// Implementing the Add Student Functionality

}

if (adminOption == 4)

{

system("cls");

topHeader();

subMenu("Editing and Updation");

updateInventory();

// Implementing the Add Student Functionality

}

if (adminOption == 5)

{

system("cls");

topHeader();

subMenu("Sorted Medicines");

showSortedmedicines(medicineCount);

// Implementing the Add Student Functionality

}

if (adminOption == 6)

{

system("cls");

topHeader();

subMenu("New Emlopyee");

cout << " Enter number of the Employees to be added: ";

// cin >> employeeCount;

employeeCount = validateInteger();

addEmployee(employeeCount);

// Implementing the Add Student Functionality

}

if (adminOption == 7)

{

system("cls");

topHeader();

subMenu("Employee Details");

showEmployeeDetails(employeeCount);

// Implementing the Add Student Functionality

}

if (adminOption == 8)

{

system("cls");

topHeader();

subMenu("Employee Sorted Salary");

showSortedEmployeeSalaries(employeeCount);

// Implementing the Add Student Functionality

}

if (adminOption == 9)

{

int soldGoods;

system("cls");

topHeader();

subMenu("Sold Medicines");

soldGoods = calcSoldMedicines(medicineCount);

viewSoldMedicine(soldGoods);

// cout << endl

// << " The number of Sold medicines are : " << soldGoods << endl;

// Implementing the Add Student Functionality

}

if (adminOption == 10)

{

system("cls");

topHeader();

subMenu("Delete Medicines");

deleteMedicines();

// Implementing the Add Student Functionality

}

if (adminOption == 11)

{

system("cls");

topHeader();

subMenu("Add Supplier Information");

addSupplierDetails();

// Implementing the Add Student Functionality

}

if (adminOption == 12)

{

system("cls");

topHeader();

subMenu("Supplier Information");

showSupplierDetails();

// Implementing the Add Student Functionality

}

if (adminOption == 13)

{

system("cls");

topHeader();

subMenu("Search Medicines");

searchMedicine(medicineCount);

// Implementing the Add Student Functionality

}

if (adminOption == 14)

{

system("cls");

topHeader();

subMenu("User Feedbacks");

showRemarks();

// Implementing the Add Student Functionality

}

if (adminOption == 15)

{

system("cls");

topHeader();

subMenu("Total Earning");

int totalEarning = 0;

totalEarning = calcEarning(earning);

viewEarning(totalEarning);

// Implementing the Add Student Functionality

}

// Implementing the rest of the Admin Options

clearScreen();

}

}

int userMenu()

{

int option;

optionLimit(1, 12);

cout << " 1)View and Buy Medicines" << endl;

cout << " 2)View Cart" << endl;

cout << " 3)Edit Cart" << endl;

cout << " 4)Delete Cart" << endl;

cout << " 5)View different Drug Categories" << endl;

cout << " 6)View and Print Bill Slip" << endl;

cout << " 7)Not Cured with medicines Get Checked Up" << endl;

cout << " 8)Search Medicines" << endl;

cout << " 9)Subscribe Online Employee job Notifications" << endl;

cout << " 10)Enter Feedback" << endl;

cout << " 11)Subscribe Online Survices" << endl;

cout << " 12) Exit" << endl;

cout << " \*\*\*\*\*\* ------------------------------------------------------------------------- \*\*\*\*\*\*" << endl

<< endl;

cout << " -->Your OPTION : ";

// cin >> option;

option = validateInteger();

return option;

}

void userInterface()

{

int userOption = 0;

while (userOption != 12)

{

topHeader();

subMenu("User");

userOption = userMenu();

if (userOption == 1)

{

system("cls");

topHeader();

subMenu("View and Buy");

viewNBuyMedicine();

// Implementing the View Details Functionality

}

if (userOption == 2)

{

system("cls");

topHeader();

subMenu("Cart");

viewCartandTotal();

// totalCash = totalBill;

// cout << endl

// << " Total is :" << totalCash;

// Implementing the View Details Functionality

}

if (userOption == 3)

{

system("cls");

topHeader();

subMenu("Edit Cart");

editCart();

// Implementing the View Details Functionality

}

if (userOption == 4)

{

system("cls");

topHeader();

subMenu("Delete Cart");

int choice = 0;

cout << " Press one of following :-\n (1)Delete/Empty whole Cart \n (2)Delete/Empty a Medicine \n Your Choice --> ";

choice = validateInteger();

// cin>>choice;

if (choice == 1)

{

totalCash = deleteCart();

cout << endl

<< " Now Total is :" << totalCash;

}

else if (choice == 2)

{

deleteCartMediicine();

cout << " Deleted SuccessFully ! ";

}

// Implementing the View Details Functionality

}

if (userOption == 5)

{

system("cls");

topHeader();

subMenu("Categories");

differentDrugCategories();

// Implementing the View Details Functionality

}

if (userOption == 6)

{

system("cls");

topHeader();

subMenu("Bill Slip");

printBillSlip();

// Implementing the View Details Functionality

}

if (userOption == 7)

{

system("cls");

topHeader();

subMenu("CheckUp Plan");

checkUpPlan();

// Implementing the View Details Functionality

}

if (userOption == 8)

{

system("cls");

topHeader();

subMenu("Search Medicines");

UsearchMedicine();

// Implementing the View Details Functionality

}

if (userOption == 9)

{

system("cls");

topHeader();

subMenu("Job Notifications");

jobNotifications();

// Implementing the View Details Functionality

}

if (userOption == 10)

{

system("cls");

topHeader();

subMenu("Enter Feedback");

userFeedback();

// Implementing the View Details Functionality

}

if (userOption == 11)

{

system("cls");

topHeader();

subMenu("Online Survices");

onlineSurvices();

// Implementing the View Details Functionality

}

// Implementing the rest of the User Options

clearScreen();

}

}

// void footer()

// {

// cout << " ----------------------------" << endl;

// }

// Admin Fucntions

void inputMedicine(int medicineCount)

{

for (counting = 0; counting < medicineCount; counting++)

{

cout << " Enter medicine Name : ";

cin >> medicineName[counting];

cout << " Enter Expiry Date : ";

cin >> expiryDate[counting];

cout << " Enter Price : ";

cin >> prices[counting];

store\_medicines();

cout << endl

<< " Added Successfuly ! " << endl

<< endl;

}

}

void showInventory(int medicineCount)

{

cout << " The available Medicines in inventory are : " << endl;

cout << " Counting"

<< "\t\t"

<< "Medicine Name"

<< "\t\t"

<< "Price" << endl;

for (int count = 0; count < medicineCount; count++)

{

cout << " " << count + 1 << ")"

<< "\t\t" << medicineName[count] << "\t\t\t" << prices[count] << endl;

}

}

void showPricesNExpiry(int medicineCount)

{

cout << " The available Medicines in inventory are : " << endl;

cout << " Counting"

<< "\t\t"

<< " Medicine Name"

<< "\t\t"

<< " Expiry Date"

<< "\t\t"

<< " Price" << endl;

for (int count = 0; count < medicineCount; count++)

{

cout << " " << count + 1 << ")"

<< "\t\t" << medicineName[count] << "\t\t\t" << expiryDate[count] << "\t\t\t" << prices[count] << "\t\t" << endl;

}

}

void updateInventory()

{

int indx;

cout << " Enter indx of the Medicine you want to Change/Update (index will be \"counting-1\"): : ";

// cin >> indx;

indx = validateInteger();

string updatedName, updatedExpiry;

int updatedPrice;

// string medicineName[indx];

// string expiryDate[indx];

// int prices[indx];

cout << " Enter medicine Name again: ";

cin >> updatedName;

cout << " Enter Expiry Date again: ";

cin >> updatedExpiry;

cout << " Enter Price again: ";

// cin >> updatedPrice;

updatedPrice = validateInteger();

medicineName[indx] = updatedName;

expiryDate[indx] = updatedExpiry;

prices[indx] = updatedPrice;

store\_medicines();

cout << endl

<< " Updated Successfuly ! " << endl;

}

void addEmployee(int employeeCount)

{

for (int count = 0; count < employeeCount; count++)

{

cout << " Enter Employee Name : ";

cin >> employeeName[count];

cout << " Enter Entry Date of the new Employee : ";

cin >> entryDate[count];

cout << " Enter Employee Salary : ";

cin >> employeeSalary[count];

store\_employees();

cout << endl

<< " Added " << count + 1 << " employee Successfuly ! " << endl;

}

}

void showEmployeeDetails(int employeeCount)

{

cout << " The Employee details are : " << endl;

cout << " Counting"

<< "\t\t"

<< "Employee Name"

<< "\t\t"

<< "Entry Date of Employee"

<< "\t\t"

<< "Pay" << endl;

for (int count = 0; count < employeeCount; count++)

{

cout << " " << count + 1 << ")"

<< "\t\t" << employeeName[count] << "\t\t\t" << entryDate[count] << "\t\t\t\t" << employeeSalary[count] << "\t\t" << endl;

}

}

int calcSoldMedicines(int medicineCount)

{

int soldGoods = 0, soldCount;

for (int soldCount = 0; soldCount < soldMedicinesCount; soldCount++)

{

soldGoods = soldGoods + quantityOfSpecifiedMedicine[soldCount];

}

return soldGoods;

}

void viewSoldMedicine(int soldGoods)

{

cout << endl

<< " The number of Sold medicines are : " << soldGoods << endl;

}

int calcEarning(int earning)

{

int totalEarning = earning;

// for (int soldCount = 0; soldCount < noOfMedicinestoBuy; soldCount++)

// {

// totalEarning = totalEarning + totalCash;

// }

return totalEarning;

}

void viewEarning(int totalEarning)

{

cout << endl

<< " Total earning is : " << totalEarning << endl;

}

void deleteMedicines()

{

cout << " Enter indx of the Medicine you want to Delete (index will be \"counting-1\"): ";

// cin >> indx;

indx = validateInteger();

int deleteOption;

cout << " Enter zero to Delete : ";

cin >> deleteOption;

if (deleteOption == 0)

{

medicineName[indx] = "0";

expiryDate[indx] = "0";

prices[indx] = 0;

store\_medicines();

cout << endl

<< " Deleted Successfuly ! " << endl;

}

}

void addSupplierDetails()

{

// for (int count = 0; count < numberOfsupplier; count++)

// {

cout << " Enter supplier Name : ";

cin >> supplierName[numberOfsupplier];

cout << " Enter the Date : ";

cin >> delieveryDate[numberOfsupplier];

cout << " Enter number of Medicies he Delievered : ";

cin >> quantityofMedicineSupplied[numberOfsupplier];

store\_suppliers();

// }

}

void showSupplierDetails()

{

cout << " Counting"

<< "\t\t"

<< "Supplier Name"

<< "\t\t"

<< "Delievery of Supplier"

<< "\t\t"

<< "Delieverd Medicines" << endl;

// for (int count = 0; count < numberOfsupplier; count++)

// {

cout << " " << numberOfsupplier << ")"

<< "\t\t" << supplierName[numberOfsupplier] << "\t\t\t" << delieveryDate[numberOfsupplier] << "\t\t\t\t" << quantityofMedicineSupplied[numberOfsupplier] << "\t\t" << endl;

// }

}

void searchMedicine(int medicineCount)

{

string searchMedicine;

int counter;

bool flag = false;

cout << " Enter Medicine Name :";

cin >> searchMedicine;

for (counter = 0; counter < medicineCount; counter++)

{

if (searchMedicine == medicineName[counter])

{

flag = true;

break;

}

else

{

flag = false;

}

}

if (flag == true)

{

cout << " Medicie details are : " << endl;

cout << " Name:\t" << medicineName[counter] << "\n";

cout << " Expiry:\t" << expiryDate[counter] << "\n";

cout << " Price:\t" << prices[counter] << "\n"

<< endl;

}

else if (flag == false)

{

cout << endl

<< " No match medicine Found !";

}

}

void showRemarks()

{

if (optiontoAddcomment == 1)

{

for (int idx = 0; idx < feedbacksCount; idx++)

{

cout << " " << numOfFeedbacks << ")" << Comments[idx] << "\n";

numOfFeedbacks++;

}

}

else if (optiontoAddcomment == 0)

{

cout << endl

<< " No comment added ! " << endl;

}

}

// UserFuntions

void viewNBuyMedicine()

{

showInventory(medicineCount);

cout << " How many medicines you want to buy : ";

noOfMedicinestoBuy = validateInteger();

int cart[noOfMedicinestoBuy];

for (int counting = 0; counting < noOfMedicinestoBuy; counting++)

{

cout << " Enter the Medicine (number/index) you want to buy : ";

cart[counting] = validateInteger();

cout << " Enter quantity : ";

quantityOfSpecifiedMedicine[counting] = validateInteger();

soldMedicinesCount++;

}

}

void viewCartandTotal()

{

totalBill = 0;

cout << " Counting"

<< "\t"

<< "Name"

<< "\t\t"

<< "Price"

<< "\t\t"

<< "Quantity\t\t" << endl;

for (int counting = 0; counting < noOfMedicinestoBuy; counting++)

{

cout << " " << counting + 1 << ")"

<< "\t\t" << medicineName[counting] << "\t\t\t" << prices[counting] << "\t\t\t" << quantityOfSpecifiedMedicine[counting] << endl;

totalBill = totalBill + (prices[counting] \* quantityOfSpecifiedMedicine[counting]);

}

cout << " Total Bill is : " << totalBill << endl;

earning = totalBill;

}

void editCart()

{

int index;

cout << " Enter the Medicine (number/index) you want to buy after editing/change : ";

index = validateInteger();

cout << " Enter quantity again: ";

quantityOfSpecifiedMedicine[index] = validateInteger();

}

void deleteCartMediicine()

{

int index;

cout << " Enter the Medicine (number/index) you want to Delete : ";

index = validateInteger();

quantityOfSpecifiedMedicine[index] = 0;

prices[index] = 0;

medicineName[index];

}

int deleteCart()

{

// cout << " Enter indx of the CartMedicine you want to Delete : ";

// // cin >> indx;

// indx = validateInteger();

int deleteOption;

cout << " Enter zero to Delete : ";

cin >> deleteOption;

if (deleteOption == 0)

{

for (int indx = 0; indx < noOfMedicinestoBuy; indx++)

{

cart[indx] = 0;

}

noOfMedicinestoBuy = 0;

totalBill = 0;

cout << endl

<< " Deleted Successfuly ! " << endl;

}

return totalBill;

}

void differentDrugCategories()

{

// int size;

cout << " The available Medicine Catogaries are : " << endl;

cout << " Fever"

<< "\t"

<< "Cough"

<< "\t"

<< "Pain" << endl;

for (int count = 0; count < medicineCount; count++)

{

cout << " " << medicineName[count] << "\t\t" << medicineName[count + 1] << "\t\t" << medicineName[count + 2] << endl;

}

}

void printBillSlip()

{

bool printOption;

viewCartandTotal();

cout << " Press 1 to print the Bill Slip or 0 to not : ";

cin >> printOption;

if (printOption == 1)

{

cout << endl

<< " Slip printed Successfuly ! " << endl;

}

else if (printOption == 0)

{

cout << endl

<< " Could't print Slip ! " << endl;

}

}

void checkUpPlan()

{

int optionType;

optionLimit(1, 4);

cout << " 1)Coud't find an Medicine . " << endl;

cout << " 2)Coud't get cured by Medicine . " << endl;

cout << " 3)Pain is to much . " << endl;

cout << " 4)Want to meet a doctor . " << endl;

cout << " Chose one of the Above Optins : ";

// cin >> optionType;

optionType = validateInteger();

if (optionType == 1 || optionType == 2 || optionType == 3 || optionType == 4)

{

cout << " Meet Dr.Taha Saleem at DHQ Hospital, Kasur from 8'O (am) Clock to 4'O Clock (pm) .";

}

}

void UsearchMedicine()

{

char optionType;

cout << " For quick Search enter S or s : ";

cin >> optionType;

if (optionType == 's' || optionType == 'S')

{

searchMedicine(medicineCount);

}

}

void jobNotifications()

{

bool optiontoAdd;

if (employeeCount == 0)

{

cout << endl

<< " There is a Seat of an Employee ! " << endl

<< " Do you want to be an Employee (Enter 1 for yes and 0 for no): ";

cin >> optiontoAdd;

if (optiontoAdd == 1)

{

addUserASEmployee();

}

}

else

{

cout << endl

<< " There is no Seat for an Employee !" << endl;

}

}

void addUserASEmployee()

{

int employeeCount = 1;

addEmployee(employeeCount);

// for (int count = 0; count < employeeCount; count++)

// {

// cout << " Enter Your Name : ";

// cin >> employeeName[count];

// cout << " Enter your Entry Date : ";

// cin >> entryDate[count];

// cout << " Enter estimed Salary you require: ";

// cin >> employeeSalary[count];

// cout << endl

// << " Added you as an employee Successfuly ! " << endl;

// }

}

void userFeedback()

{

cout << " Do you want to add Comment//Feedback (Enter 1 for yes and 0 for no): ";

// cin >> optiontoAddcomment;

optiontoAddcomment = validateInteger();

if (optiontoAddcomment == 1)

{

cout << " Enter remarks (and press Enter 2 or 3 times to continue ! ) : ";

getline(cin, Comment);

load\_commebts\_inArr(Comment);

}

else

{

cout << endl

<< " No comment was added ! " << endl;

}

}

void onlineSurvices()

{

bool optiontoSubscribe;

string Comment;

cout << " Do you want to Subscribe our survices (Enter 1 for yes and 0 for no): ";

cin >> optiontoSubscribe;

// optiontoSubscribe = validateInteger();

if (optiontoSubscribe == 1)

{

cout << " Search JAZZAD on YouTube and write yes if Subscribed my Channel : " << endl;

getline(cin, Comment);

if (Comment == "yes")

{

cout << endl

<< " Subscribed survices Succesfully ! " << endl;

}

else

{

cout << endl

<< " No survices Subscribed ! " << endl;

}

}

else if (optiontoSubscribe == 0)

{

cout << endl

<< " No survices Subscribed yet. " << endl;

// clearScreen();

}

}

void showSortedEmployeeSalaries(int employeeCount)

{

int largest\_idx;

int temp;

for (int idx = 0; idx < employeeCount; idx++)

{

largest\_idx = employeelargest(idx);

temp = employeeSalary[largest\_idx];

employeeSalary[largest\_idx] = employeeSalary[idx];

employeeSalary[idx] = temp;

}

cout << " Sorted Salaries are : " << endl

<< endl;

cout << " ";

for (int idx = 0; idx < employeeCount; idx++)

{

cout << employeeSalary[idx] << ", ";

}

}

int employeelargest(int s)

{

int large = -1;

int large\_index;

for (int idx = s; idx < employeeCount; idx++)

{

if (large < employeeSalary[idx])

{

large = employeeSalary[idx];

large\_index = idx;

}

}

return large\_index;

}

void showSortedmedicines(int medicineCount)

{

int largest\_idx;

int temp;

for (int idx = 0; idx < medicineCount; idx++)

{

largest\_idx = medicinelargest(idx);

temp = prices[largest\_idx];

prices[largest\_idx] = prices[idx];

prices[idx] = temp;

}

cout << " Sorted medicines are : " << endl

<< endl;

cout << " ";

for (int idx = 0; idx < medicineCount; idx++)

{

cout << prices[idx] << ", ";

}

}

int medicinelargest(int s)

{

int large = -100;

int large\_index;

for (int idx = s; idx < medicineCount; idx++)

{

if (large < prices[idx])

{

large = prices[idx];

large\_index = idx;

}

}

return large\_index;

}

* **Weakness in the Business Application**
  + My File Handling is not perfect. There are some Points where it is not working Perfect. But, I think I have done an Excellent job.
* **Future Directions**
  + There is always room for improvement. I would like to improve my application in file handling and make it Perfect by adding some more functionalities.
  + I would also look to improve user interface.

**Student Reg. No. :**  2022-CS-139  **Student Name. Muhammad Taha Saleem**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |

**Student Reg. No. :**  2022-CS-139  **Student Name. Muhammad Taha Saleem**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

|  |  |
| --- | --- |
| **Checked by:** |  |
| **Comments:** |  |