**Hostel’s Mess Management System**

****

Session 2022\_2026

**Submitted by:**

Iqra Tariq 2022-CS-29

**Supervised by:**

**Maida shahid**

Department of computer science

**University of engineering and technology**

**Lahore Pakistan**

**My purpose**

* The purpose of my application is to automatize the hostel mess functionality and provide both the user and the admin a smart platform to interact with each other.
* The existing system to facilitate mess functionalities for the students is the normal pen and paper method. Any query is addressed by physically visiting the mess and record are taken on a register, and the account information is only available with hostel staff and it requires more effort and lot of time is wasted in this way.
* To use this facility, the user has to register and with the account credentials he can login the connected to the network.
* Once connected the user can use the functionalities of the mess such as accessing his account information, applying for leave and checking the menu etc., without visiting mess physically.
* It enables the admin to view the inventory and access guest details. This application is free of cost for the users.
* Individuals who want to see functionality of the mess can simply log in to the app and have everything on their fingertips.
* Admin can take decision and the inventory for the mess can be change.
* Users can check the mess bill daily. In this way time is saved for users.

In this management system there are two users first is mess in charge (admin) and second

is student. further details are given bellow in the form of users, function requirements and wireframes etc.

**Users:**

There are two users in this system:

1. Admin (mess in charge)
2. Student.

**Functional Requirements:**

Following are functional requirements that re fulfilled by this system. Both admin and student can login into the management system by using unique password and username.

#include <iostream>

#include <fstream>

#include <string>

#include <windows.h>

#include <conio.h>

using namespace std;

// function for sign UP and in

void topHeader();

void submenu(string message);

string signIn(string name, string password);

void loadSignData();

void adminMenu(int found);

void studentMenu(int option);

string getfield(string record, int field);

void clearScreen();

void adminInterface();

void studentInterface();

int login();

// arrays for sign IN and add user

string username[100];

string userpassword[100];

string userrole[100];

int usercount = 0;

// admin functionalities

bool isValid(string namE);

void storeData(string names, string password, string role);

void adduser(string name, string password, string role);

void displaymenu();

void storemenu(string day, string nemeOfItem);

void storemenuInfile(string day, string nemeOfItem);

string getfieldforMenu(string record, int field);

void loadmenu();

void displayprice();

void storePrice(string names, int price);

void storePriceInFile(string itemName, int price);

string getfieldForprice(string record, int field);

void loadPrice();

void viewUser();

void noteForStudent(string date, string note);

void storeNoteinFile(string date, string note);

void loadNote();

string getfieldFornote(string record, int field);

void viewOrder();

void viewfeedback();

void removeuser(string name, string password); // Remove student

string checkUser(string name, string password);

void storeRemoveFile();

void changeMenu(string dayName, string itemName); // change menu

void storechangeMenu();

void checkAttendance();

// student functonalities

void checkmenu();

void checkPrice();

void viewNote();

void markAttendance(string studentName, string date);

void storeAttendanceinFile(string studentNames, string date);

void loadAttendance();

string getfieldForAttendace(string record, int field);

void billCalculate(string day);

void placeOrder(string student, int room, string order);

void viewMyOrder(string name, int room);

void changeOrder(string name, int room, string orderReplace);

void storeChangeOrder();

void removeOrder(string name, int room);

void storeRemoveOrder();

void storeOrderinFile(string student, int room, string order);

void loadOrder();

string getfieldForOrder(string record, int field);

void feedBack(string date, string feedback);

void storeFeedBackInFile(string date, string feedback);

void loadFeedback();

string getfieldForFeedback(string record, int field);

// admin menu arrays

const int TOTAL\_DAYS = 7; // arrays for menu display

string namEofDays[TOTAL\_DAYS];

string itemNames[TOTAL\_DAYS];

int countDays = 0;

const int TOTAL\_Attend = 100; // arrays for attendace mark;

string studentName[TOTAL\_Attend];

string dateArray[TOTAL\_Attend];

;

// int storebill[TOTAL\_ITEM];

int studentCount = 0;

int sumOfattendace = 0;

int countAttentance = 0;

const int TOTAL\_ITEM = 7; // array for price display

string itemName[TOTAL\_ITEM];

int itemPrice[TOTAL\_ITEM];

int countPrice = 0;

const int TOTAL\_DATE = 30;

string dateOfNote[TOTAL\_DATE];

string notice[TOTAL\_DATE];

int countDate = 0;

const int TOTAL\_ORDER = 100;

string nameOfStudent[TOTAL\_ORDER];

int roomNo[TOTAL\_ORDER];

string orderOfStudent[TOTAL\_ORDER];

int countOrder = 0;

const int TOTAL\_FEEDBACK = 0; // array for feedback

string feedbackDate[TOTAL\_FEEDBACK];

string writtenFeedBack[TOTAL\_FEEDBACK];

int feedBackCount = 0;

int time=1;

main()

{

    if(time==1)

    {

    loadSignData();

    loadmenu();

    loadPrice();

    loadNote();

    loadOrder();

    loadAttendance();

    loadFeedback();

    time++;

    }

    system("cls");

     int loginoption=0;

        string result;

    while (loginoption!=2)

    {

        int option;

         topHeader();

            submenu("login");

loginoption=login();

        if (loginoption == 1)

        {

            string name;

            string password;

            cout << "enter your name:";

            cin >> name;

            cout << "enter your password:";

            cin >> password;

            result = signIn(name, password);

        }

        if (loginoption == 2)

        {

            break;

        }

        if (result == "admin")

        {

            system("cls");

             topHeader();

            submenu("admininterface");

            adminInterface();

        }

        if (result == "student")

        {

            topHeader();

            submenu("studeninterface");

 studentInterface();

        }

        else

        {

            cout << result;

        }

        clearScreen();

    }

}

void adminInterface()

{

            int found;

            cout << "1.display menu:" << endl;

            cout << "2.displaly price of item:" << endl;

            cout << "3.Add student:" << endl;

            cout << "4.view user:" << endl;

            cout << "5.write a note for student:" << endl;

            cout << "6.view order:" << endl;

            cout << "7.view feedback:" << endl;

            cout << "8.remove student:" << endl;

            cout << "9.change menu:" << endl;

            cout << "10.check attendancet:" << endl;

            cout << "11.exit:" << endl;

            cout << "select one option:" << endl;

            cin >> found;

            adminMenu(found);

}

void studentInterface()

{

           int view;

            cout << "1.check menu:" << endl;

            cout << "2.check price of an item:" << endl;

            cout << "3.view note:" << endl;

            cout << "4.place order:" << endl;

            cout << "5.feedback:" << endl;

            cout << "6.mark attendace:" << endl;

            cout << "7.change Order:" << endl;

            cout << "8.view Order:" << endl;

            cout << "9.remove Order:" << endl;

            cout << "10.exit:" << endl;

            cout << "select one option:" << endl;

            cin >> view;

            studentMenu(view);

}

int  login()

{

        int option;

         topHeader();

            submenu("signIn");

        string result;

        cout << "1.signIn:" << endl;

        cout << "2.exit:" << endl;

        cout << "select one option:";

        cin >> option;

        return option;

}

void clearScreen()

{

    cout<<"press any key to continue:"<<endl;

    getch();

    system("cls");

}

void adduser(string name, string password, string role)

{

    username[usercount] = name;

    userpassword[usercount] = password;

    userrole[usercount] = role;

    usercount++;

}

bool isValid(string namE)

{

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

    {

        if (username[index] == namE)

        {

            return false;

        }

    }

    return true;

}

void storeData(string names, string password, string role)

{

    fstream file;

    file.open("signUp.txt", ios::app);

    file

        << names << "," << password << "," << role << endl;

    file.close();

}

string signIn(string name, string password)

{

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

    {

        if (username[index] == name && userpassword[index] == password)

        {

            return userrole[index];

        }

    }

    return "something went wrong:";

}

void loadSignData()

{

    string record;

    fstream file;

    file.open("signUp.txt", ios::in);

    while (getline(file, record))

    {

        username[usercount] = getfield(record, 1);

        userpassword[usercount] = getfield(record, 2);

        userrole[usercount] = getfield(record, 3);

        usercount++;

    }

    file.close();

}

string getfield(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void adminMenu(int found)

{

    int adminOption=0;

    if (found == 1)

    {

        system("cls");

        topHeader();

        submenu("dispalymenu");

        displaymenu();

    }

    else if (found == 2)

    {

        system("cls");

        topHeader();

        submenu("displaybill");

        displayprice();

    }

    else if (found == 3)

    {

        system("cls");

        topHeader();

        submenu("addstudent");

        string name;

        string password, role;

        cout << "enter user name:";

        cin >> name;

        cout << "enter user password:";

        cin >> password;

        cout << "enter user role:";

        cin >> role;

        bool check;

        check = isValid(name);

        if (check == true)

        {

            adduser(name, password, role);

            storeData(name, password, role);

            cout << "Add user successfuly:" << endl;

        }

        if (check == false)

        {

            cout << "user Already Exist:" << endl;

        }

    }

    if (found == 4)

    {

        system("cls");

        topHeader();

        submenu("viewtotaluser");

        viewUser();

        getch();

    }

    if (found == 5)

    {

        system("cls");

        topHeader();

        submenu("writeAnote");

        string will;

        cout << "do you want to write a note for students:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string date;

            string noteWrite;

            cout << "enter Date:";

            cin >> date;

            cout << "write a note for students:";

            cin.ignore();

            getline(cin, noteWrite);

            noteForStudent(date, noteWrite);

            storeNoteinFile(date, noteWrite);

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (found == 6)

    {

        system("cls");

        topHeader();

        submenu("viewOrder");

        viewOrder();

    }

    else if (found == 7)

    {

        system("cls");

        topHeader();

        submenu("viewfeedback");

        viewfeedback();

    }

    else if (found == 8)

    {

        system("cls");

        topHeader();

        submenu("removeUser");

        string check;

        string name;

        string passwords;

        cout << "enter the name of user you want to remove:";

        cin >> name;

        cout << "enter the password of user";

        cin >> passwords;

        check = checkUser(name, passwords);

        if (check == "student" || check == "Student")

        {

            removeuser(name, passwords);

        }

        else if (check == "admin" || check == "Admin")

        {

            cout << "You can only remove students:";

        }

        else

        {

            cout << check;

        }

        storeRemoveFile();

    }

    else if (found == 9)

    {

        system("cls");

        topHeader();

        submenu("changeMenu");

        string dayName;

        string nameOfItem;

        cout << "enter the name of day that's item you want to change:";

        cin >> dayName;

        cout << "enter the name of itme which you want to add:";

        cin >> nameOfItem;

        changeMenu(dayName, nameOfItem);

        storechangeMenu();

    }

    else if (found == 10)

    {

        system("cls");

        topHeader();

        submenu("checkAttendace");

        checkAttendance();

        getch();

    }

    else if(found==11)

    {

        main();

    }

    clearScreen();

adminInterface();

}

void studentMenu(int option)

{

    if (option == 1)

    {

        system("cls");

        topHeader();

        submenu("checkmenu");

        checkmenu();

    }

    else if (option == 2)

    {

        system("cls");

        topHeader();

        submenu("checkPrice");

        checkPrice();

    }

    else if (option == 3)

    {

        system("cls");

        topHeader();

        submenu("viewnoteofAdmin");

        viewNote();

    }

    else if (option == 4)

    {

        system("cls");

        topHeader();

        submenu("placeorder");

        string will;

        cout << "Do you want to place the order:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string name, order;

            int room;

            cout << "enter your name:";

            cin >> name;

            cout << "enter your room number:";

            cin >> room;

            cout << "enter your order:";

            cin.ignore();

            getline(cin, order);

            storeOrderinFile(name, room, order);

            placeOrder(name, room, order);

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (option == 5)

    {

        topHeader();

        submenu("givingFeedBack");

        string will;

        cout << "Do you want to give feedback about mess:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string date;

            string feeback;

            cout << "enter date of feedback:";

            cin >> date;

            cout << "write a feedback:";

            cin.ignore();

            getline(cin, feeback);

            feedBack(date, feeback);

            storeFeedBackInFile(date, feeback);

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (option == 6)

    {

        topHeader();

        submenu("markatt");

        string will;

        cout << "Do you want to mark attendance:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string yourName;

            string date;

            cout << "enter your full name:";

            cin.ignore();

            getline(cin, yourName);

            cout << "enter date:";

    getline(cin,date);

            storeAttendanceinFile(yourName, date);

            markAttendance(yourName, date);

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (option == 7)

    {

        topHeader();

        submenu("changeorder");

        string will;

        cout << "Do you want to change your order:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string name, newOrder;

            int room;

            cout << "enter your name:";

            cin >> name;

            cout << "enter your room number:";

            cin >> room;

            cout << "enter the new order:";

            cin >> newOrder;

            changeOrder(name, room, newOrder);

            storeChangeOrder();

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (option == 8)

    {

        topHeader();

        submenu("vieworder");

        string will;

        cout << "Do you want to view your order:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string name;

            int room;

            cout << "enter your name:";

            cin >> name;

            cout << "enter your room number:";

            cin >> room;

            viewMyOrder(name, room);

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if (option == 9)

    {

        topHeader();

        submenu("removeOrder");

        string will;

        cout << "Do you want to remove your order:";

        cin >> will;

        if (will == "yes" || will == "Yes")

        {

            string name;

            int room;

            cout << "enter your name:";

            cin >> name;

            cout << "enter your room number:";

            cin >> room;

            removeOrder(name, room);

            storeRemoveOrder();

        }

        else if (will != "yes" || will != "Yes")

        {

            exit;

        }

    }

    else if(option==10)

    {

         main();

    }

    clearScreen();

    studentInterface();

}

void topHeader()

{

    cout << "# #  #   ## ### ### #    # # ###  ##  ##   # #  #  ###  #   ## ### # # ### ### ###   ## # #  ## ### ### # #  " << endl;

    cout << "# # # # #    #  #   #    ### #   #   #     ### # # # # # # #   #   ### #   # #  #   #   # # #    #  #   ###  " << endl;

    cout << "### # #  #   #  ##  #    ### ##   #   #    ### ### # # ### # # ##  ### ##  # #  #    #   #   #   #  ##  ###  " << endl;

    cout << "# # # #   #  #  #   #    # # #     #   #   # # # # # # # # # # #   # # #   # #  #     #  #    #  #  #   # #  " << endl;

    cout << "# #  #  ##   #  ### ###  # # ### ##  ##    # # # # # # # #  ## ### # # ### # #  #   ##   #  ##   #  ### # #  " << endl;

}

void submenu(string message)

{

    string messages = "Main menu >" + message;

    cout << messages << endl;

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

}

void viewUser()

{

    cout << "name"

         << "\t"

         << "password"

         << "\t"

         << "role" << endl;

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

    {

        cout << username[index] << "\t" << userpassword[index] << "\t" << userrole[index] << endl;

    }

        clearScreen();

adminInterface();

}

void displaymenu()

{

    string name;

    string day;

    for (int index = 0; index < 7; index++)

    {

        cout << "enter the name of day:";

        cin >> day;

        cout << "enter the name of item:";

        cin >> name;

        storemenu(day, name);

        storemenuInfile(day, name);

    }

    clearScreen();

adminInterface();

}

void storemenu(string days, string names)

{

    namEofDays[countDays] = days;

    itemNames[countDays] = names;

    countDays++;

}

void storemenuInfile(string nameOfDay, string nameOfItem)

{

    fstream file;

    file.open("menufile.txt", ios::app);

    file

        << nameOfDay << "," << nameOfItem << endl;

    file.close();

}

void loadmenu()

{

    string record;

    fstream file;

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

    while (getline(file, record))

    {

        namEofDays[countDays] = getfieldforMenu(record, 1);

        itemNames[countDays] = getfieldforMenu(record, 2);

        countDays++;

    }

}

string getfieldforMenu(string record, int field)

{

    int commacount = 1;

    string item;

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

    {

        if (record[index] == ',')

        {

            commacount = commacount + 1;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void checkmenu()

{

    string select;

    cout << "1.sunday:" << endl;

    cout << "2.monday:" << endl;

    cout << "3.tuesaday:" << endl;

    cout << "4.wednesday:" << endl;

    cout << "5.thursday:" << endl;

    cout << "6.friday:" << endl;

    cout << "7.saturday:" << endl;

    cout << "write the name of day:" << endl;

    cin >> select;

    for (int j = 0; j < countDays; j++)

    {

        if (namEofDays[j] == select)

        {

            cout << itemNames[j] << endl;

        }

    }

    clearScreen();

    studentInterface();

}

void displayprice()

{

    string item;

    int price;

    for (int index = 0; index < TOTAL\_ITEM; index++)

    {

        cout << "enter the name of item:";

        cin >> item;

        cout << "enter the price:";

        cin >> price;

        storePrice(item, price);

        storePriceInFile(item, price);

    }

       clearScreen();

    adminInterface();

}

void storePrice(string names, int price)

{

    itemName[countPrice] = names;

    itemPrice[countPrice] = price;

    countPrice++;

}

void storePriceInFile(string itemName, int price)

{

    fstream file;

    file.open("priceFile.txt", ios::app);

    file << itemName << "," << price << endl;

    file.close();

}

void loadPrice()

{

    string record;

    fstream file;

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

    while (getline(file, record))

    {

        itemName[countPrice] = getfieldForprice(record, 1);

        itemPrice[countPrice] = stoi(getfieldForprice(record, 2));

        countPrice++;

    }

    file.close();

}

string getfieldForprice(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void checkPrice()

{

    string nameOfItem;

    cout << "enter the name of item:";

    cin >> nameOfItem;

    for (int j = 0; j < countPrice; j++)

    {

        if (itemName[j] == nameOfItem)

        {

            cout << itemPrice[j] << endl;

        }

    }

      clearScreen();

    studentInterface();

}

void noteForStudent(string date, string note)

{

    string dateOfNote[countDate] = date;

    string notice[countDate] = note;

    countDate++;

    clearScreen();

adminInterface();

}

void storeNoteinFile(string date, string note)

{

    fstream file;

    file.open("noteFile.txt", ios::app);

    file << endl

         << date << "," << note << endl;

    file.close();

}

void loadNote()

{

    string record;

    fstream file;

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

    while (getline(file, record))

    {

        dateOfNote[countDate] = getfieldFornote(record, 1);

        notice[countDate] = getfieldFornote(record, 2);

        countDate++;

    }

}

string getfieldFornote(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void viewNote()

{

    if (countDate > TOTAL\_DATE)

    {

        countDate = 0;

    }

    else if (countDate == 0)

    {

        cout << "no note yet:";

    }

    else if (countDate > 0)

    {

        cout << "Date:"

             << "\t"

             << "Note" << endl;

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

        {

            cout << dateOfNote[index] << "\t" << notice[index] << endl;

        }

    }

}

void placeOrder(string student, int room, string order)

{

    nameOfStudent[countOrder] = student;

    roomNo[countOrder] = room;

    orderOfStudent[countOrder] = order;

    countOrder++;

   clearScreen();

    studentInterface();

}

void storeOrderinFile(string student, int room, string order)

{

    fstream file;

    file.open("orderFile.txt", ios::app);

    file << student << "," << room << "," << order << endl;

    file.close();

}

void loadOrder()

{

    string record;

    fstream file;

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

    while (getline(file, record))

    {

        nameOfStudent[countOrder] = getfieldForOrder(record, 1);

        roomNo[countOrder] = stoi(getfieldForOrder(record, 2));

        orderOfStudent[countOrder] = getfieldForOrder(record, 3);

        countOrder++;

    }

}

string getfieldForOrder(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void viewOrder()

{

    if (countOrder == 0)

    {

        cout << "no Order yet:";

    }

    else if (countOrder > 0)

    {

        cout << "student:  "

             << "room number"

             << "\t"

             << "order" << endl;

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

        {

            cout << nameOfStudent[index] << "\t" << roomNo[index] << "\t"

                 << "\t" << orderOfStudent[index] << endl;

        }

    }

        clearScreen();

adminInterface();

}

void feedBack(string date, string feedback)

{

    feedbackDate[feedBackCount] = date;

    writtenFeedBack[feedBackCount] = feedback;

    feedBackCount++;

}

void storeFeedBackInFile(string date, string feedback)

{

    fstream file;

    file.open("feedbackfile.txt", ios::app);

    file << date << "," << feedback << endl;

    file.close();

}

void loadFeedback()

{

    string record;

    fstream file;

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

    while (getline(file, record))

    {

        feedbackDate[feedBackCount] = getfieldFornote(record, 1);

        writtenFeedBack[feedBackCount] = getfieldFornote(record, 2);

        feedBackCount++;

    }

    file.close();

}

string getfieldForFeedback(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void viewfeedback()

{

    cout << "date"

         << "\t"

         << "feedback" << endl;

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

    {

        cout << feedbackDate[index] << "\t" << writtenFeedBack[index] << endl;

    }

        clearScreen();

adminInterface();

}

void removeuser(string name, string password)

{

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

    {

        if (username[index] == name && userpassword[index] == password)

        {

            username[index] = " ";

            username[index] = username[index + 1];

            userpassword[index] = " ";

            userpassword[index] = userpassword[index + 1];

            userrole[index] = " ";

            userrole[index] = userrole[index + 1];

            usercount--;

        }

    }

        clearScreen();

adminInterface();

}

string checkUser(string name, string password)

{

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

    {

        if (username[index] == name && userpassword[index] == password)

        {

            return userrole[index];

        }

    }

    return "username or password wrong:";

}

void storeRemoveFile()

{

    fstream file;

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

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

    {

        file << username[index] << "," << userpassword[index] << "," << userrole[index] << endl;

    }

    file.close();

}

void changeMenu(string dayName, string itemName)

{

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

    {

        if (namEofDays[index] == dayName)

        {

            itemNames[index] = itemName;

        }

    }

        clearScreen();

adminInterface();

}

void storechangeMenu()

{

    fstream file;

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

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

    {

        file << namEofDays[index] << "," << itemNames[index] << endl;

    }

    file.close();

}

void markAttendance(string studentNames, string date)

{

    studentName[studentCount] = studentNames;

    dateArray[studentCount] = date;

    studentCount++;

}

void storeAttendanceinFile(string studentName, string date)

{

    fstream file;

    file.open("attand.txt", ios::app);

    file << studentName << "," << date << endl;

}

void loadAttendance()

{

    string record;

    fstream file;

    file.open("attand.txt", ios::in);

    while (getline(file, record))

    {

        studentName[studentCount] = getfieldForAttendace(record, 1);

        dateArray[studentCount] = getfieldForAttendace(record, 2);

        studentCount++;

    }

}

string getfieldForAttendace(string record, int field)

{

    string item;

    int commacount = 1;

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

    {

        if (record[index] == ',')

        {

            commacount++;

        }

        else if (commacount == field)

        {

            item = item + record[index];

        }

    }

    return item;

}

void checkAttendance()

{

    if (studentCount == 0)

    {

        cout << "total attendance is" << studentCount << endl;

    }

    else if (studentCount > 0)

    {

          cout << "total attendance is" << studentCount << endl;

        cout << "student"

             << "\t"

             << "date" << endl;

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

        {

            cout << studentName[index] << "\t" << dateArray[index] << endl;

        }

    }

        clearScreen();

adminInterface();

}

void viewMyOrder(string name, int room)

{

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

    {

        if (nameOfStudent[index] == name && roomNo[index] == room)

            cout << "your order is " << orderOfStudent[index] << endl;

    }

       clearScreen();

    studentInterface();

}

void changeOrder(string name, int room, string orderReplace)

{

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

    {

        if (nameOfStudent[index] == name && roomNo[index] == room)

        {

            orderOfStudent[index] = orderReplace;

        }

    }

}

void storeChangeOrder()

{

    fstream file;

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

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

    {

        file << nameOfStudent[index] << "," << roomNo[index] << "," << orderOfStudent[index] << endl;

    }

    file.close();

}

void removeOrder(string name, int room)

{

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

    {

        if (nameOfStudent[index] == name && roomNo[index] == room)

        {

            nameOfStudent[index] = " ";

            nameOfStudent[index] = nameOfStudent[index + 1];

            roomNo[index] = ' ';

            roomNo[index] = roomNo[index + 1];

            orderOfStudent[index] = " ";

            orderOfStudent[index] = orderOfStudent[index + 1];

            countOrder--;

        }

    }

}

void storeRemoveOrder()

{

    fstream file;

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

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

    {

        file << nameOfStudent[index] << "," << roomNo[index] << "," << orderOfStudent[index] << endl;

    }

    file.close();

}