**Restaurant Management System**

****

Session: 2021 – 2025

**Submitted by:**

**Hamza Tariq 2021-CS-74**

**Supervised by:**

**Ma’am Maida Shahid**

Department of Computer Science

**University of Engineering and Technology**

**Lahore, Pakistan**

**Table of Content**

**Topics Page No**

1. Abstract ………………………………………………………................. 03

2. Users ……………………………………………………………….... 03

3. Functional Requirements ………………………………………………... 03

3.1 Manager ……………………………………………………............... 03

3.2 Customer ……………………………………………………………. 04

3.3 Delivery Boy ………………………………………………………... 04

4. Wireframes ……………………………………………………………… 04

4.1 Manager Menu ………………………………………………………. 05

4.2 Customer Menu …………………………………………………........ 10

4.3 Delivery Boy Menu .………………………………………………..... 14

5. Data Structure …………………………………………...…..................... 16

6. Functions Prototypes ………………………………………………........... 17

7. Flow Chart ……………………………………………………………...... 21

8. Complete Code ………………………………………………................... 22

**Restaurant Management System**

**1.Abstract:**

Name of the project is Restaurant Management System.

In this project Manager can add the number of items he wants in his store fixing their price and quantity.

He can remove or add an item. He can view the items he has added in his store in a sorted manner. He is also capable of viewing items less than a threshold quantity. He as well can view feedbacks of customers. And can view bill of different customers. He can add new users as well.

Customers can place orders which are included in the restaurant (added by manager). They can avail hot deals They can get their delivered. Customers can see their invoice and add their feedbacks in the feedback section.

Delivery boy can deliver his orders and get his payment.

**2.Users:**

There are three users of this system:

* Manager
* Customer
* Delivery Boy

**3.Functional Requirements:**

**3.1 Manager:**

As manager, I can:

1. Add Item
2. View Available Items
3. View Items less than threshold quantity
4. Update an item
5. Remove an item
6. View Feedbacks
7. View Invoice
8. Add new Users

**3.2 Customer:**

As customer, I can:

* 1. View available items
  2. Promotional deals and discounts
  3. Place order
  4. View Receipt
  5. Take Away or delivery
  6. Give reviews &feedbacks

**3.3 Delivery Boy:**

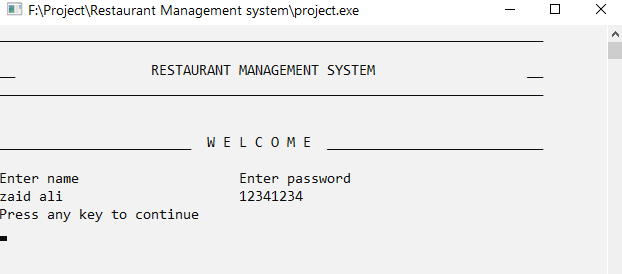
As Delivery Boy, I can:

* 1. View Orders
  2. View selected order
  3. Update Delivery Status
  4. View Earnings

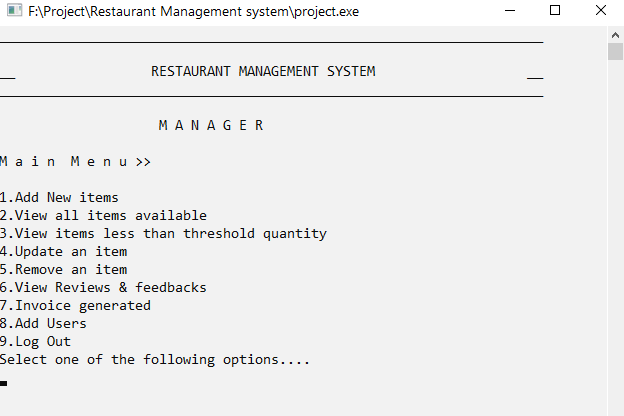
**4.Wireframes:**

**4.1 Main Menu:**

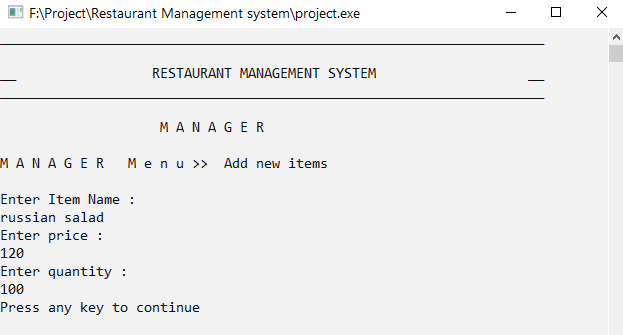
**Manager Login:**

****

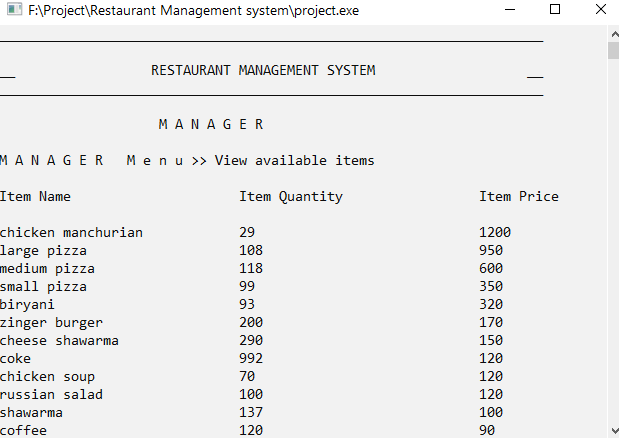
* **Manager Menu:**

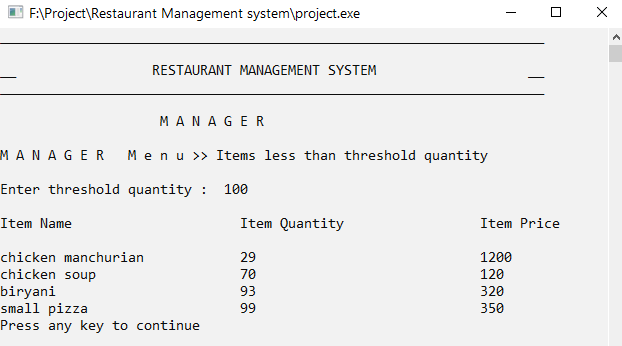
****

**1.Add Items:**

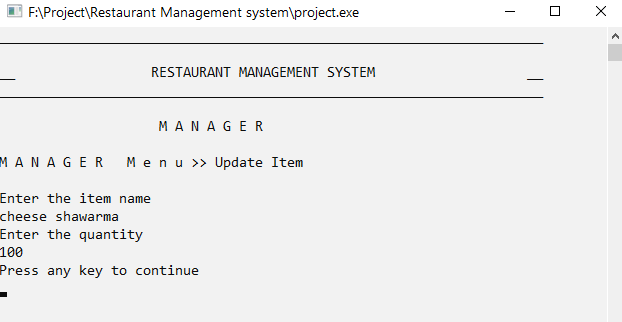
****

**2.View Available items:**

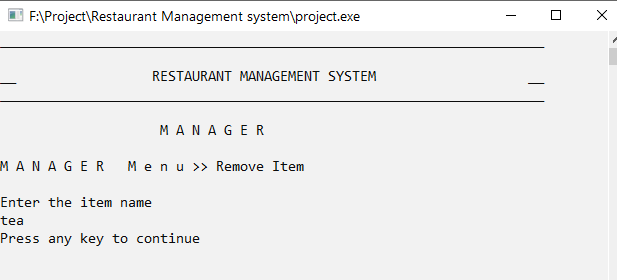
****

**3.View items less than threshold frequency:**

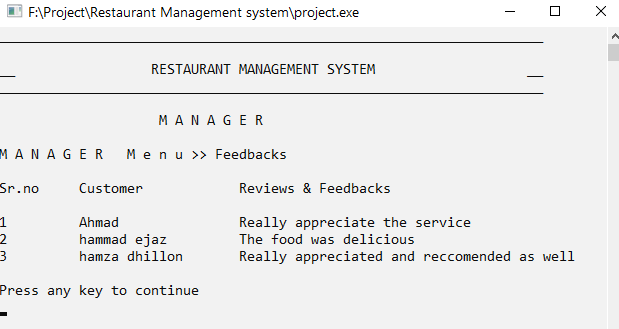
**4.Update Stock:**

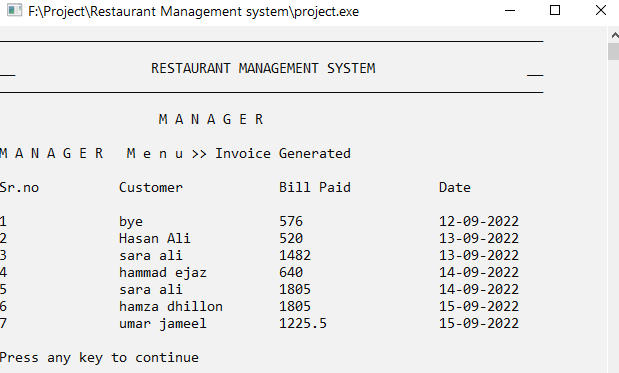
****

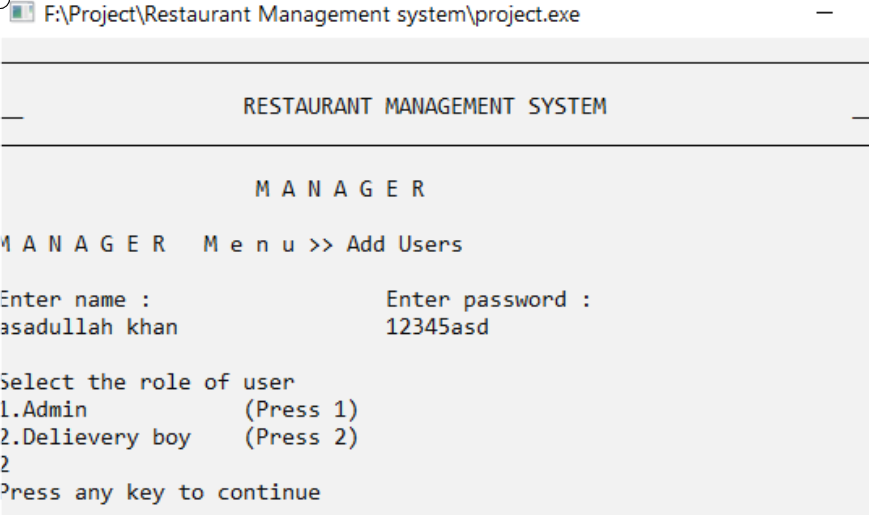
**5.Remove Stock:**

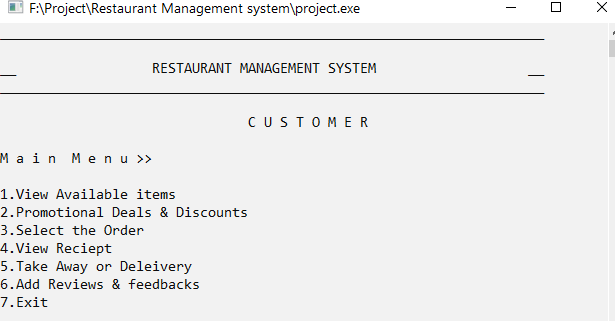
****

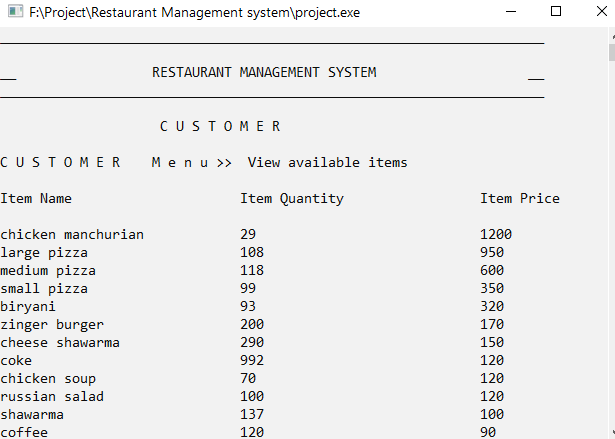
**6.View feedbacks and Reviews**

****

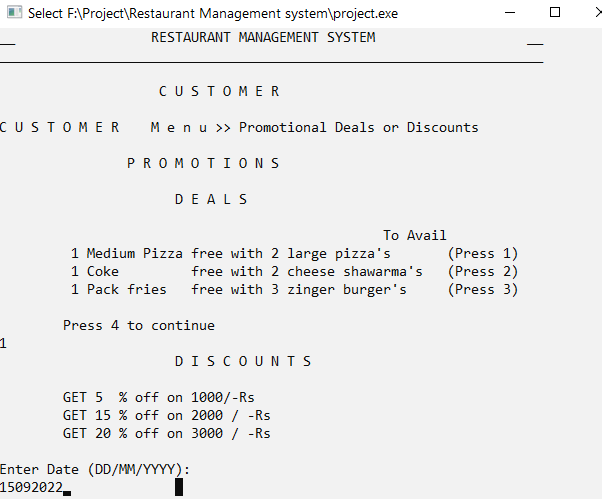
**7.View feedbacks and Reviews**

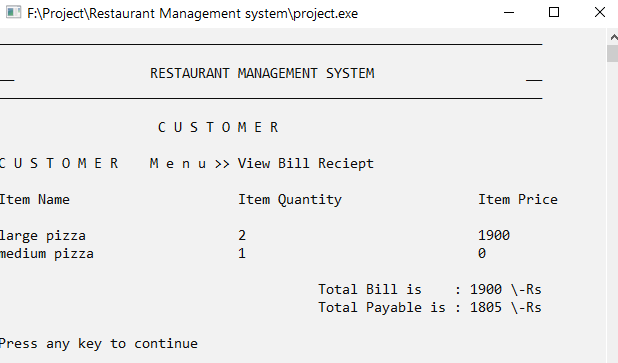
**9.Add Users**

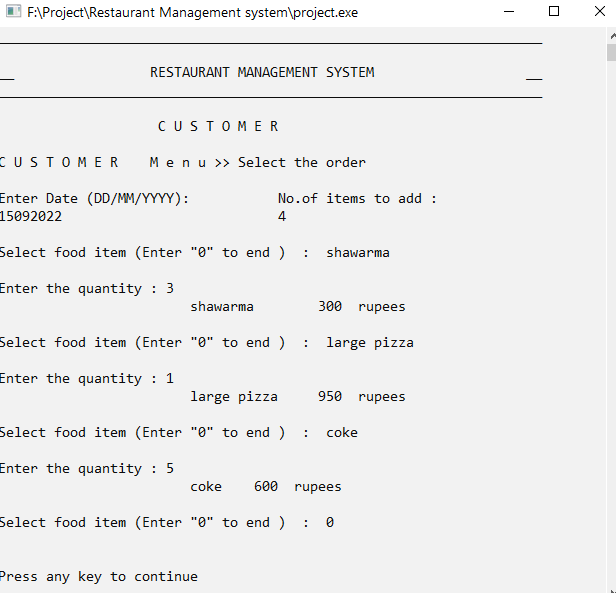
**4.2 Customer Menu:**

**1.View Available Items**

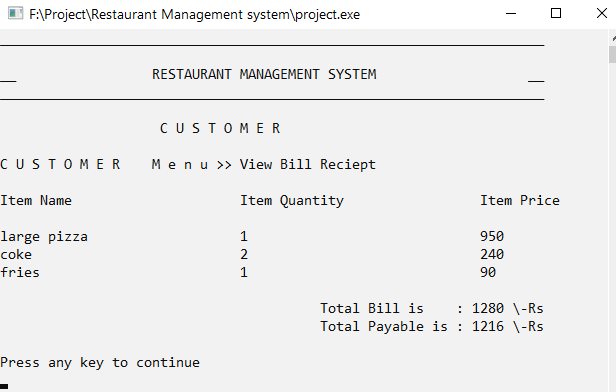
**2.Promotional Deals and Discounts**

****

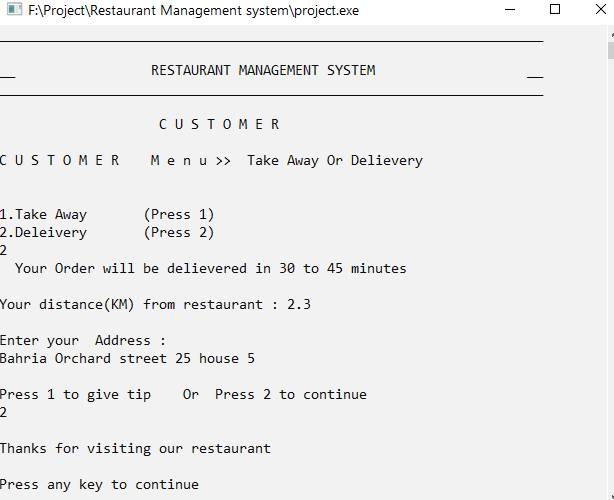
****

**3.Select order**

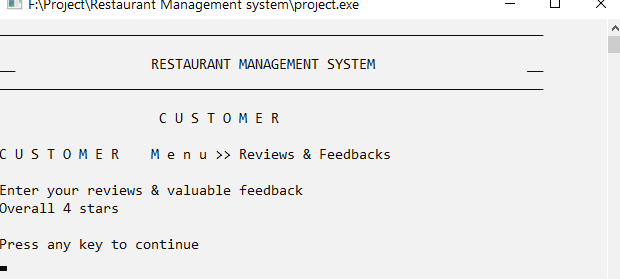
**4.View Receipt**

****

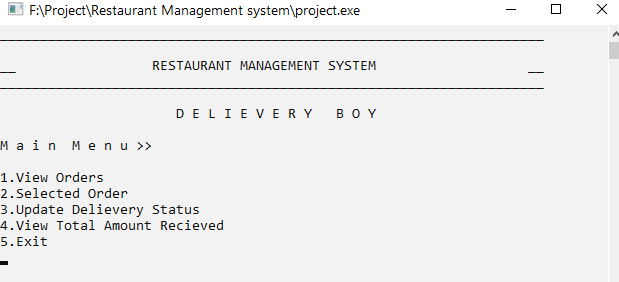
**5.Take Away or Delivery**

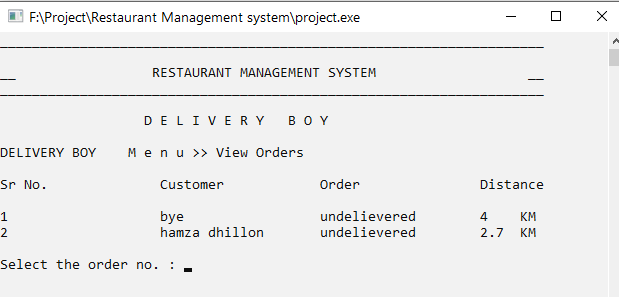
****

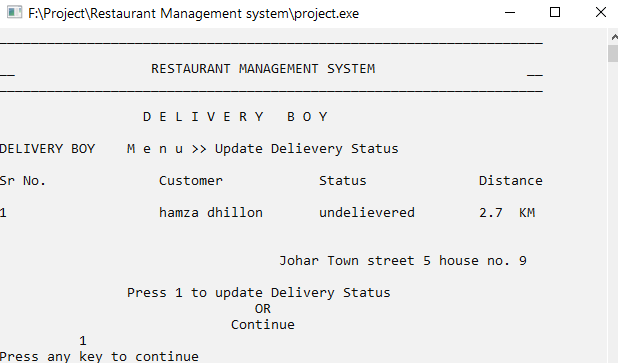
* 1. **Feedbacks &Reviews**

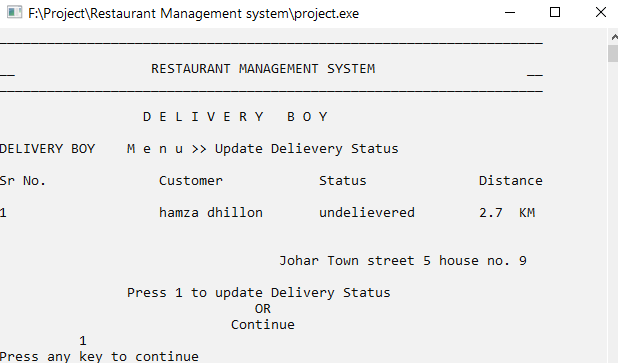
****

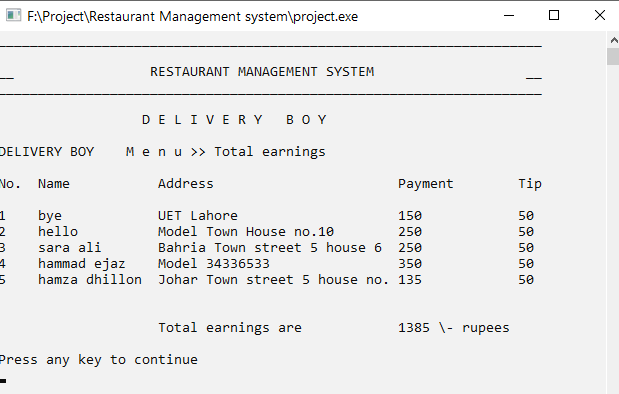
* 1. **Delivery boy Menu**

****

**1.View Orders**

**2.Selected order**

**3.Update Order Status**

**4.View Earning**

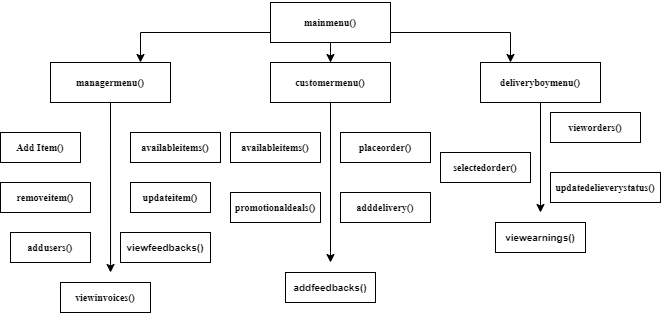
**5.Data Structures:**

* int LIMIT = 100;
* int users = 1;
* string idA[LIMIT] = {"hamza"};
* string passA[LIMIT] = {"11112222"};
* string roleA[LIMIT] = {"ADMIN"};
* int count\_item = 0;
* string Item\_Name[LIMIT];
* int Item\_Price[LIMIT];
* int Item\_Quantity[LIMIT];
* int customer\_item = 0;
* int customer\_price[LIMIT];
* string customer\_Item[LIMIT];
* int customer\_quantity[LIMIT];
* string namemanager[LIMIT];
* string purchase\_date[LIMIT];
* float paid[LIMIT];
* int customer\_count = 0;
* string dateA[LIMIT];
* string delieveryquene[LIMIT];
* string delieverystatus[LIMIT];
* float delieverydistance[LIMIT];
* string delieveryaddress[LIMIT];
* float delieverytip[LIMIT];
* int delieverycount = 0;
* string delname[LIMIT];
* string deladdress[LIMIT];
* float delbill[LIMIT];
* float deltip[LIMIT];
* int newcount = 0;
* string feedback[LIMIT];
* string namefeedback[LIMIT];
* int count\_feedback = 0;
* string op = " ";
* int cicount;
* int customers\_count = 0;

**6. Functions:**

* // Basic/Main Functions Prototypes
* void header();
* void miniheader();
* void clearscreen();
* void gotoxy(int x, int y);
* string mainmenu();
* string managermenu();
* string customermenu();
* string delboymenu();
* void customerheader();
* void deliveryheader();
* void managerheader();
* bool validate(int users, string pass, string passA[]);
* string sign\_in(string name, string pass, int users, string idA[], string passA[], string roleA[]);
* void sign\_up(int users, string idA[], string passA[], string roleA[]);
* void adminsign\_up(int users, string idA[], string passA[], string roleA[]);
* int sign\_in\_index(string name, string pass, int users, string idA[], string passA[], string roleA[]);
* void additem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* void addItemIntoArr(string item, int quantity, int price, int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* int large(int position, int count\_item, int Item\_Price[]);
* void sortingwrtprice(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* void threshold(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* int largequantity(int position, int count\_item, int Item\_Quantity[]);
* void sortingwrtquantity(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* void update\_stock(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* bool removeitem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* string deals(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);
* int availdeals(string op, int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);
* int order(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);
* float calculatebill(int customeritemcount, int customer\_price[]);
* void receipt(int cicount, int customer\_price[], string customer\_Item[], int customer\_quantity[], string purchase\_date[]);
* float discount(int bill);
* bool delievery(int index, int delieverycount, string idA[], string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);
* void addfeedback(int count\_feedback, string feedback[], int index, string idA[], string namefeedback[]);
* void viewfeedback(int count\_feedback, string feedback[], string namefeedback[]);
* int orders(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);
* void selectedorders(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);
* bool delievery\_status(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);
* int removedelievered(int newcount, int count\_item, int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[], string delname[], string deladdress[], float delbill[], float deltip[]);
* void earnings(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);
* float calcearnings(int newcount, float delbill[], float deltip[]);
* bool date(string c);
* string datefunc(string c);
* int managerinvoice(int customer\_price[], int cicount, float paid[], string namemanager[], string purchase\_date[], string dateA[], int customer\_count, int index, string idA[]);
* void printinvoice(int customer\_count, string namemanager[], string purchase\_date[], float paid[]);
* void savedata(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* int datareader(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);
* string parsedata(string line, int count);
* void logindata(int users, string idA[], string passA[], string roleA[]);
* int readlogindata(int users, string idA[], string passA[], string roleA[]);
* void delboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);
* int readdelboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);
* void delquene(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);
* int readquenedata(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);
* void history(int customer\_count, float paid[], string namemanager[], string purchase\_date[]);
* int readhistory(int customer\_count, float paid[], string namemanager[], string purchase\_date[]);
* void storefeedback(int count\_feedback, string namefeedback[], string feedback[]);
* int readhistory(int count\_feedback, string namefeedback[], string feedback[]);

**7.Flowchart:**

****

**8.Complete Code**

#include <iostream>

#include <fstream>

#include <conio.h>

#include <windows.h>

#include <cstring>

#include <string>

using namespace std;

void header();

void miniheader();

void clearscreen();

void gotoxy(int x, int y);

string mainmenu();

string managermenu();

string customermenu();

string delboymenu();

void customerheader();

void deliveryheader();

void managerheader();

bool validate(int users, string pass, string passA[]);

string sign\_in(string name, string pass, int users, string idA[], string passA[], string roleA[]);

void sign\_up(int users, string idA[], string passA[], string roleA[]);

void adminsign\_up(int users, string idA[], string passA[], string roleA[]);

int sign\_in\_index(string name, string pass, int users, string idA[], string passA[], string roleA[]);

void additem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

void addItemIntoArr(string item, int quantity, int price, int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

int large(int position, int count\_item, int Item\_Price[]);

void sortingwrtprice(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

void threshold(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

int largequantity(int position, int count\_item, int Item\_Quantity[]);

void sortingwrtquantity(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

void update\_stock(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

bool removeitem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

string deals(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);

int availdeals(string op, int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);

int order(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[]);

float calculatebill(int customeritemcount, int customer\_price[]);

void receipt(int cicount, int customer\_price[], string customer\_Item[], int customer\_quantity[], string purchase\_date[]);

float discount(int bill);

bool delievery(int index, int delieverycount, string idA[], string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);

void addfeedback(int count\_feedback, string feedback[], int index, string idA[], string namefeedback[]);

void viewfeedback(int count\_feedback, string feedback[], string namefeedback[]);

int orders(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);

void selectedorders(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);

bool delievery\_status(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[]);

int removedelievered(int newcount, int count\_item, int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[], string delname[], string deladdress[], float delbill[], float deltip[]);

void earnings(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);

float calcearnings(int newcount, float delbill[], float deltip[]);

bool date(string c);

string datefunc(string c);

int managerinvoice(int customer\_price[], int cicount, float paid[], string namemanager[], string purchase\_date[], string dateA[], int customer\_count, int index, string idA[]);

void printinvoice(int customer\_count, string namemanager[], string purchase\_date[], float paid[]);

void savedata(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

int datareader(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[]);

string parsedata(string line, int count);

void logindata(int users, string idA[], string passA[], string roleA[]);

int readlogindata(int users, string idA[], string passA[], string roleA[]);

void delboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);

int readdelboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[]);

void delquene(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);

int readquenedata(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[]);

void history(int customer\_count, float paid[], string namemanager[], string purchase\_date[]);

int readhistory(int customer\_count, float paid[], string namemanager[], string purchase\_date[]);

void storefeedback(int count\_feedback, string namefeedback[], string feedback[]);

int readhistory(int count\_feedback, string namefeedback[], string feedback[]);

main()

{

    int LIMIT = 100;

    int users = 1;

    string idA[LIMIT] = {"hamza"};

    string passA[LIMIT] = {"11112222"};

    string roleA[LIMIT] = {"ADMIN"};

    users = readlogindata(users, idA, passA, roleA);

    int count\_item = 0;

    string Item\_Name[LIMIT];

    int Item\_Price[LIMIT];

    int Item\_Quantity[LIMIT];

    // Loading Items from file

    count\_item = datareader(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

    int customer\_item = 0;

    int customer\_price[LIMIT];

    string customer\_Item[LIMIT];

    int customer\_quantity[LIMIT];

    string namemanager[LIMIT];

    string purchase\_date[LIMIT];

    float paid[LIMIT];

    int customer\_count = 0;

    customer\_count = readhistory(customer\_count, paid, namemanager, purchase\_date);

    string dateA[LIMIT];

    string delieveryquene[LIMIT];

    string delieverystatus[LIMIT];

    float delieverydistance[LIMIT];

    string delieveryaddress[LIMIT];

    float delieverytip[LIMIT];

    int delieverycount = 0;

    delieverycount = readquenedata(delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip);

    string delname[LIMIT];

    string deladdress[LIMIT];

    float delbill[LIMIT];

    float deltip[LIMIT];

    int newcount = 0;

    newcount = readdelboydata(newcount, delname, deladdress, delbill, deltip);

    // initializing delievery array by zero

    for (int i = 0; i < LIMIT; i++)

    {

        delieverytip[i] = 0;

    }

    string feedback[LIMIT];

    string namefeedback[LIMIT];

    int count\_feedback = 0;

    count\_feedback = readhistory(count\_feedback, namefeedback, feedback);

    string op = " ";

    int cicount;

    int customers\_count = 0;

    string who;

    do

    {

        system("cls");

        op = mainmenu();

        if (op == "1")

        {

            int count = 0;

            system("cls");

            // header();

            string name, pass;

            do

            {

                header();

                miniheader();

                gotoxy(0, 8);

                cout << "Enter name" << endl;

                gotoxy(30, 8);

                cout << "Enter password" << endl;

                cin.ignore();

                getline(cin, name);

                gotoxy(30, 9);

                cin >> pass;

                who = sign\_in(name, pass, users, idA, passA, roleA);

                if (who == "null")

                {

                    cout << "Wrong Credientials " << endl;

                    clearscreen();

                }

            } while (who == "null");

            // gettting index of the user so that we can find the name of customer

            int index = sign\_in\_index(name, pass, users, idA, passA, roleA);

            clearscreen();

            if (who == "ADMIN")

            {

                system("cls");

                string num = " ";

                while (num != "9")

                {

                    num = managermenu();

                    if (num == "1")

                    {

                        managerheader();

                        cout << " Add new items " << endl;

                        cout << " " << endl;

                        additem(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        count\_item++;

                        clearscreen();

                        savedata(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                    }

                    else if (num == "2")

                    { // sorting w.r.t price

                        managerheader();

                        cout << "View available items " << endl;

                        cout << " " << endl;

                        sortingwrtprice(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        clearscreen();

                    }

                    else if (num == "3")

                    { // items less than threshold quantity  with sorting wrt quantity

                        managerheader();

                        cout << "Items less than threshold quantity" << endl;

                        cout << " " << endl;

                        threshold(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        clearscreen();

                    }

                    else if (num == "4")

                    {

                        managerheader();

                        cout << "Update Item  " << endl;

                        cout << " " << endl;

                        update\_stock(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        clearscreen();

                    }

                    else if (num == "5")

                    {

                        managerheader();

                        cout << "Remove Item  " << endl;

                        cout << " " << endl;

                        if (removeitem(count\_item, Item\_Name, Item\_Price, Item\_Quantity) == true)

                        {

                            count\_item--;

                        }

                        clearscreen();

                    }

                    else if (num == "6")

                    {

                        managerheader();

                        cout << "Feedbacks" << endl;

                        viewfeedback(count\_feedback, feedback, namefeedback);

                        clearscreen();

                    }

                    else if (num == "7")

                    {

                        managerheader();

                        cout << "Invoice Generated  " << endl;

                        printinvoice(customer\_count, namemanager, purchase\_date, paid);

                        cout << " " << endl;

                        clearscreen();

                    }

                    else if (num == "8")

                    {

                        managerheader();

                        cout << "Add Users" << endl;

                        cout << " " << endl;

                        adminsign\_up(users, idA, passA, roleA);

                        users++;

                        clearscreen();

                        logindata(users, idA, passA, roleA);

                    }

                    else if (num == "9")

                    {

                        cout << " Going back to main screen " << endl;

                        clearscreen();

                    }

                    else

                    {

                        cout << "Invalid input" << endl;

                        clearscreen();

                    }

                }

            }

            else if (who == "CUSTOMER")

            {

                system("cls");

                string num2 = " ";

                bool select = false;

                while (num2 != "7")

                {

                    num2 = customermenu();

                    if (num2 == "1")

                    {

                        customerheader();

                        cout << " View available items " << endl;

                        cout << " " << endl;

                        sortingwrtprice(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        clearscreen();

                    }

                    else if (num2 == "2")

                    {

                        customerheader();

                        cout << "Promotional Deals or Discounts" << endl;

                        cout << " " << endl;

                        string op = deals(count\_item, LIMIT, Item\_Name, Item\_Price, Item\_Quantity, customer\_price, customer\_Item, customer\_quantity, dateA);

                        if (op == "1" || op == "2" || op == "3")

                        {

                            cicount = availdeals(op, count\_item, LIMIT, Item\_Name, Item\_Price, Item\_Quantity, customer\_price, customer\_Item, customer\_quantity, dateA);

                            customer\_count = managerinvoice(customer\_price, cicount, paid, namemanager, purchase\_date, dateA, customer\_count, index, idA);

                            select = true;

                        }

                        savedata(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                        clearscreen();

                    }

                    else if (num2 == "3")

                    {

                        customerheader();

                        cout << "Select the order" << endl;

                        cout << " " << endl;

                        select = true;

                        cicount = order(count\_item, LIMIT, Item\_Name, Item\_Price, Item\_Quantity, customer\_price, customer\_Item, customer\_quantity, dateA);

                        customer\_count = managerinvoice(customer\_price, cicount, paid, namemanager, purchase\_date, dateA, customer\_count, index, idA);

                        history(customer\_count, paid, namemanager, purchase\_date);

                        cout << " " << endl;

                        clearscreen();

                        savedata(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

                    }

                    else if (num2 == "4")

                    {

                        customerheader();

                        cout << "View Bill Reciept" << endl;

                        receipt(cicount, customer\_price, customer\_Item, customer\_quantity, purchase\_date);

                        cout << " " << endl;

                        clearscreen();

                    }

                    else if (num2 == "5")

                    {

                        customerheader();

                        cout << " Take Away Or Delievery" << endl;

                        cout << endl;

                        if (select == true)

                        {

                            if (delievery(index, delieverycount, idA, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip) == true)

                            {

                                delieverycount++;

                            }

                        }

                        else

                        {

                            cout << "Please Order some items first" << endl;

                        }

                        cout << " " << endl;

                        clearscreen();

                        delquene(delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip);

                    }

                    else if (num2 == "6")

                    {

                        customerheader();

                        cout << "Reviews & Feedbacks" << endl;

                        addfeedback(count\_feedback, feedback, index, idA, namefeedback);

                        count\_feedback++;

                        storefeedback(count\_feedback, namefeedback, feedback);

                        clearscreen();

                    }

                    else if (num2 == "7")

                    {

                        cout << " Going back to main screen " << endl;

                        clearscreen();

                    }

                    else

                    {

                        cout << "Invalid input" << endl;

                        clearscreen();

                    }

                }

            }

            else if (who == "delboy")

            {

                int delievery\_index;

                system("cls");

                string num3 = " ";

                while (num3 != "5")

                {

                    num3 = delboymenu();

                    if (num3 == "1")

                    {

                        deliveryheader();

                        cout << "View Orders " << endl;

                        delievery\_index = orders(delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress);

                        clearscreen();

                    }

                    else if (num3 == "2")

                    {

                        deliveryheader();

                        cout << "Selected Orders " << endl;

                        selectedorders(delievery\_index, delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress);

                        clearscreen();

                    }

                    else if (num3 == "3")

                    {

                        deliveryheader();

                        cout << "Update Delievery Status " << endl;

                        if (delievery\_status(delievery\_index, delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress))

                        {

                            newcount = removedelievered(newcount, count\_item, delievery\_index, delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip, delname, deladdress, delbill, deltip);

                            delieverycount--;

                        }

                        clearscreen();

                        delquene(delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip);

                    }

                    else if (num3 == "4")

                    {

                        deliveryheader();

                        cout << "Total earnings " << endl;

                        earnings(newcount, delname, deladdress, delbill, deltip);

                        clearscreen();

                    }

                    else if (num3 == "5")

                    {

                        cout << " Going back to main screen " << endl;

                        clearscreen();

                    }

                    else

                    {

                        cout << "Invalid input" << endl;

                        clearscreen();

                    }

                    delboydata(newcount, delname, deladdress, delbill, deltip);

                }

            }

        }

        else if (op == "2")

        {

            system("cls");

            header();

            sign\_up(users, idA, passA, roleA);

            users++;

            clearscreen();

            logindata(users, idA, passA, roleA);

        }

        else if (op == "3")

        {

            cout << " Thanks for using Restaurant Management System " << endl;

            clearscreen();

        }

        else

        {

            cout << "Invalid input" << endl;

            clearscreen();

        }

    } while (op != "3");

    savedata(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

    logindata(users, idA, passA, roleA);

    delboydata(newcount, delname, deladdress, delbill, deltip);

    delquene(delieverycount, delieveryquene, delieverystatus, delieverydistance, delieveryaddress, delieverytip);

    history(customer\_count, paid, namemanager, purchase\_date);

    storefeedback(count\_feedback, namefeedback, feedback);

}

void managerheader()

{

    header();

    cout << "                    M A N A G E R " << endl;

    cout << " " << endl;

    cout << "M A N A G E R   M e n u >> ";

}

void customerheader()

{

    header();

    cout << "                    C U S T O M E R  " << endl;

    cout << " " << endl;

    cout << "C U S T O M E R    M e n u >> ";

}

void deliveryheader()

{

    header();

    cout << "                  D E L I V E R Y   B O Y " << endl;

    cout << " " << endl;

    cout << "DELIVERY BOY    M e n u >> ";

}

void header()

{

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

    cout << endl;

    cout << "\_\_                 RESTAURANT MANAGEMENT SYSTEM                   \_\_" << endl;

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

    cout << endl;

}

void clearscreen()

{

    cout << "Press any key to continue " << endl;

    getch();

    system("CLS");

}

void gotoxy(int x, int y)

{

    COORD coordinates; // coordinates is declared as COORD

    coordinates.X = x; // defining x-axis

    coordinates.Y = y; // defining  y-axis

    SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coordinates);

}

string mainmenu()

{

    string op;

    header();

    cout << "                             W e l c o m e                                " << endl;

    cout << endl;

    cout << "SIGN IN        (Press 1 to Sign in ) " << endl;

    cout << "           OR  " << endl;

    cout << "SIGN UP        (Press 2 to Sign up )" << endl

         << endl;

    cout << "Exit           (Press 3 to Exit )" << endl;

    cin >> op;

    return op;

}

void miniheader()

{

    cout << endl;

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  W E L C O M E  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

string sign\_in(string name, string pass, int users, string idA[], string passA[], string roleA[])

{

    for (int i = 0; i < users; i++)

    {

        if (name == idA[i] && pass == passA[i])

        {

            return roleA[i];

        }

    }

    return "null";

}

int sign\_in\_index(string name, string pass, int users, string idA[], string passA[], string roleA[])

{

    for (int i = 0; i < users; i++)

    {

        if (name == idA[i] && pass == passA[i])

        {

            return i;

        }

    }

}

bool validate(int users, string pass, string passA[])

{

    // For Same Passwords

    for (int i = 0; i < users; i++)

    {

        if (pass == passA[i])

        {

            return false;

        }

    }

    return true;

}

void sign\_up(int users, string idA[], string passA[], string roleA[])

{

    cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  W E L C O M E  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

    cout << endl;

    string pass;

    cout << "Enter your name : " << endl;

    cin.ignore();

    getline(cin, idA[users]);

    cout << "Enter your password : " << endl;

    cin >> pass;

    // Validating Passwords(8 characters)

    int count = 0;

    while (pass[count] != '\0')

    {

        count++;

    }

    while (count != 8)

    {

        count = 0;

        cout << "Enter your password (Only 8 Characters): " << endl;

        cin >> pass;

        while (pass[count] != '\0')

        {

            count++;

        }

    }

    count = 0;

    // validation in case of two same passwords

    while (validate(users, pass, passA) == false)

    {

        cout << "Please select another password ! It's already taken" << endl;

        cout << "Enter your password : " << endl;

        cin >> pass;

        while (pass[count] != '\0')

        {

            count++;

        }

        while (count != 8)

        {

            count = 0;

            cout << "Enter your password (Only 8 Characters): " << endl;

            cin >> pass;

            while (pass[count] != '\0')

            {

                count++;

            }

        }

    }

    passA[users] = pass;

    roleA[users] = "CUSTOMER";

    users++;

}

void adminsign\_up(int users, string idA[], string passA[], string roleA[])

{

    cout << endl;

 string pass;

    gotoxy(0, 9);

    cout << "Enter name : " << endl;

    gotoxy(30, 9);

    cout << "Enter password : " << endl;

    gotoxy(0, 10);

    cin.ignore();

    getline(cin, idA[users]);

    gotoxy(30, 10);

    cin >> pass;

    // Validating Passwords(8 characters)

    int count = 0;

    int y = 9;

    while (pass[count] != '\0')

    {

        count++;

    }

    while (count != 8)

    {

        y = 9;

        count = 0;

        gotoxy(30, y);

        cout << "Enter password (Only 8 Characters): " << endl;

        y = y + 1;

        gotoxy(30, y);

        cin >> pass;

        while (pass[count] != '\0')

        {

            count++;

        }

    }

    // validation in case of two same passwords

    count = 0;

    while (validate(users, pass, passA) == false)

    {

        y = y + 1;

        gotoxy(0, y);

        cout << "Please select another password ! It's already taken" << endl;

        y = y + 1;

        gotoxy(0, y);

        cout << "Enter your password : " << endl;

        y = y + 1;

        gotoxy(0, y);

        cin >> pass;

        while (pass[count] != '\0')

        {

            count++;

        }

        while (count != 8)

        {

            count = 0;

            cout << "Enter your password (Only 8 Characters): " << endl;

            cin >> pass;

            while (pass[count] != '\0')

            {

                count++;

            }

        }

    }

    int option;

    y = y + 3;

    gotoxy(0, y);

    cout << "Select the role of user" << endl

         << endl;

    y = y + 1;

    gotoxy(0, y);

    cout << "1.Admin            (Press 1)     " << endl;

    y = y + 1;

    gotoxy(0, y);

    cout << "2.Delievery boy    (Press 2)     " << endl;

    y = y + 1;

    gotoxy(0, y);

    cin >> option;

    passA[users] = pass;

    if (option == 1)

    {

        roleA[users] = "ADMIN";

    }

    else

    {

        roleA[users] = "delboy";

    }

}

string managermenu()

{

    string num;

    header();

    cout << "                    M A N A G E R " << endl;

    cout << " " << endl;

    cout << "M a i n  M e n u >>" << endl;

    cout << " " << endl;

    cout << "1.Add New items " << endl;

    cout << "2.View all items available" << endl;

    cout << "3.View items less than threshold quantity" << endl;

    cout << "4.Update an item" << endl;

    cout << "5.Remove an item " << endl;

    cout << "6.View Reviews & feedbacks" << endl;

    cout << "7.Invoice generated" << endl;

    cout << "8.Add Users " << endl;

    cout << "9.Log Out " << endl;

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

    cin >> num;

    clearscreen();

    return num;

}

string customermenu()

{

    string num2;

    header();

    cout << "                               C U S T O M E R " << endl;

    cout << " " << endl;

    cout << "M a i n  M e n u >>" << endl;

    cout << " " << endl;

    cout << "1.View Available items " << endl;

    cout << "2.Promotional Deals & Discounts " << endl;

    cout << "3.Select the Order " << endl;

    cout << "4.View Reciept" << endl;

    cout << "5.Take Away or Deleivery " << endl;

    cout << "6.Add Reviews & feedbacks" << endl;

    cout << "7.Exit" << endl;

    cin >> num2;

    clearscreen();

    return num2;

}

string delboymenu()

{

    string num3;

    header();

    cout << "                      D E L I E V E R Y   B O Y " << endl;

    cout << " " << endl;

    cout << "M a i n  M e n u >>" << endl;

    cout << " " << endl;

    cout << "1.View Orders " << endl;

    cout << "2.Selected Order " << endl;

    cout << "3.Update Delievery Status " << endl;

    cout << "4.View Total Amount Recieved" << endl;

    cout << "5.Exit" << endl;

    cin >> num3;

    clearscreen();

    return num3;

}

void additem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    string item;

    int quantity;

    int price;

    cout << "Enter Item Name : " << endl;

    cin.ignore();

    getline(cin, item);

    // Validation in case of price in negative integer

    int count = 0;

    do

    {

        if (count >= 1)

        {

            cout << "Wrong Input " << endl;

        }

        cout << "Enter price : " << endl;

        cin >> price;

        count++;

    } while (price < 0);

    // Validation in case of quantity in negative integer

    count = 0;

    do

    {

        if (count >= 1)

        {

            cout << "Wrong Input " << endl;

        }

        cout << "Enter quantity : " << endl;

        cin >> quantity;

        count++;

    } while (quantity < 0);

    addItemIntoArr(item, quantity, price, count\_item, Item\_Name, Item\_Price, Item\_Quantity);

}

void addItemIntoArr(string item, int quantity, int price, int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    Item\_Name[count\_item] = item;

    Item\_Price[count\_item] = price;

    Item\_Quantity[count\_item] = quantity;

}

void sortingwrtprice(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    for (int i = 0; i < count\_item; i++)

    {

        int largest = large(i, count\_item, Item\_Price);

        int large\_price = Item\_Price[i];

        int large\_quantity = Item\_Quantity[i];

        string large\_name = Item\_Name[i];

        Item\_Price[i] = Item\_Price[largest];

        Item\_Quantity[i] = Item\_Quantity[largest];

        Item\_Name[i] = Item\_Name[largest];

        Item\_Price[largest] = large\_price;

        Item\_Quantity[largest] = large\_quantity;

        Item\_Name[largest] = large\_name;

    }

    gotoxy(0, 9);

    cout << "Item Name" << endl;

    gotoxy(30, 9);

    cout << "Item Quantity" << endl;

    gotoxy(60, 9);

    cout << "Item Price" << endl;

    int y = 11;

    for (int i = 0; i < count\_item; i++)

    {

        int x = 0;

        gotoxy(x, y);

        cout << Item\_Name[i] << endl;

        x = x + 30;

        gotoxy(x, y);

        cout << Item\_Quantity[i] << endl;

        x = x + 30;

        gotoxy(x, y);

        cout << Item\_Price[i] << endl;

        y++;

    }

}

int large(int position, int count\_item, int Item\_Price[])

{

    int large\_price = Item\_Price[position];

    int index = position;

    for (int i = position; i < count\_item; i++)

    {

        if (large\_price < Item\_Price[i])

        {

            large\_price = Item\_Price[i];

            index = i;

        }

    }

    return index;

}

void threshold(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    // first of all sorting

    sortingwrtquantity(count\_item, Item\_Name, Item\_Price, Item\_Quantity);

    int threshold;

    cout << "Enter threshold quantity :  ";

    cin >> threshold;

    int count = 0;

    gotoxy(0, 11);

    cout << "Item Name" << endl;

    gotoxy(30, 11);

    cout << "Item Quantity" << endl;

    gotoxy(60, 11);

    cout << "Item Price" << endl;

    for (int i = 0; i < count\_item; i++)

    {

        if (Item\_Quantity[i] < threshold)

        {

            count++;

        }

    }

    if (count == 0)

    {

        gotoxy(0, 13);

        cout << "No item less than threshold quantity" << endl;

    }

    else

    {

        bool flag = false;

        int y = 13;

        for (int i = 0; i < count\_item; i++)

        {

            if (Item\_Quantity[i] < threshold)

            {

                int x = 0;

                gotoxy(x, y);

                cout << Item\_Name[i] << endl;

                x = x + 30;

                gotoxy(x, y);

                cout << Item\_Quantity[i] << endl;

                x = x + 30;

                gotoxy(x, y);

                cout << Item\_Price[i] << endl;

                y++;

                flag = true;

            }

            if (flag == false)

            {

                y--;

            }

            else

            {

                flag = false;

            }

        }

    }

}

void sortingwrtquantity(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    for (int i = 0; i < count\_item; i++)

    {

        int largest = largequantity(i, count\_item, Item\_Quantity);

        int large\_price = Item\_Price[i];

        int large\_quantity = Item\_Quantity[i];

        string large\_name = Item\_Name[i];

        Item\_Price[i] = Item\_Price[largest];

        Item\_Quantity[i] = Item\_Quantity[largest];

        Item\_Name[i] = Item\_Name[largest];

        Item\_Price[largest] = large\_price;

        Item\_Quantity[largest] = large\_quantity;

        Item\_Name[largest] = large\_name;

    }

}

int largequantity(int position, int count\_item, int Item\_Quantity[])

{

    int large\_quantity = Item\_Quantity[position];

    int index = position;

    for (int i = position; i < count\_item; i++)

    {

        if (large\_quantity > Item\_Quantity[i])

        {

            large\_quantity = Item\_Quantity[i];

            index = i;

        }

    }

    return index;

}

void update\_stock(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    string item;

    bool flag = false;

    int quantity;

    cout << "Enter the item name " << endl;

    cin.ignore();

    getline(cin, item);

    for (int z = 0; z < count\_item; z++)

    {

        if (item == Item\_Name[z])

        {

            cout << "Enter the quantity " << endl;

            cin >> quantity;

            Item\_Quantity[z] = Item\_Quantity[z] + quantity;

            flag = true;

        }

    }

    if (flag == true)

    {

        flag = false;

    }

    else

    {

        cout << " Sorry ! Item is not in our restaurant." << endl;

        flag = false;

    }

}

bool removeitem(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    string item;

    bool flag = false;

    cout << "Enter the item name " << endl;

    cin >> item;

    for (int z = 0; z < count\_item; z++)

    {

        if (item == Item\_Name[z])

        {

            for (int i = z; i < count\_item; i++)

            {

                Item\_Name[i] = Item\_Name[i + 1];

                Item\_Price[i] = Item\_Price[i + 1];

                Item\_Quantity[i] = Item\_Quantity[i + 1];

            }

            flag = true;

        }

    }

    if (flag == false)

    {

        cout << " Sorry ! Item is not in our restaurant." << endl;

    }

    return flag;

}

string deals(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[])

{

    bool flag = false;

    string op = "";

    cout << "\t\t"

         << "P R O M O T I O N S" << endl;

    cout << endl;

    cout << "\t\t"

         << "      D E A L S" << endl;

    cout << endl;

    cout << "                                                To Avail" << endl;

    cout << "\t"

         << " 1 Medium Pizza free with 2 large pizza's       (Press 1)" << endl;

    cout << "\t"

         << " 1 Coke         free with 2 cheese shawarma's   (Press 2) " << endl;

    cout << "\t"

         << " 1 Pack fries   free with 3 zinger burger's     (Press 3)" << endl;

    cout << endl;

    cout << "\t"

         << "Press 4 to continue "

         << endl;

    while (op != "4")

    {

        cin >> op;

        if (op == "1")

        {

            flag = true;

            break;

        }

        else if (op == "2")

        {

            flag = true;

            break;

        }

        else if (op == "3")

        {

            flag = true;

            break;

        }

    }

    cout << "\t\t"

         << "      D I S C O U N T S" << endl;

    cout << endl;

    cout << "\t"

         << "GET 5  % off on 1000/-Rs" << endl;

    cout << "\t"

         << "GET 15 % off on 2000 / -Rs " << endl;

    cout << "\t"

         << "GET 20 % off on 3000 / -Rs " << endl;

    cout << " " << endl;

    return op;

}

int availdeals(string op, int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[])

{ // emptying arrays for new customer

    for (int i = 0; i < LIMIT; i++)

    {

        customer\_price[i] = 0;

        customer\_Item[i] = "";

        customer\_quantity[i] = 0;

    }

    int customeritemcount = 0;

    string datemain;

    cout << "Enter Date (DD/MM/YYYY): " << endl;

    cin >> datemain;

    dateA[0] = datefunc(datemain);

    for (int z = 0; z < count\_item; z++)

    {

        if (Item\_Name[z] == "large pizza")

        {

            // making a new array to calculate invoice

            customer\_Item[customeritemcount] = "large pizza";

            customer\_quantity[customeritemcount] = 2;

            // Minusing the quantity from manager item array which cusomer purchased

            Item\_Quantity[z] = Item\_Quantity[z] - customer\_quantity[customeritemcount];

            customer\_price[customeritemcount] = customer\_quantity[customeritemcount] \* Item\_Price[z];

            customeritemcount++;

        }

        if (Item\_Name[z] == "medium pizza")

        {

            // making a new array to calculate invoice

            customer\_Item[customeritemcount] = "medium pizza";

            customer\_quantity[customeritemcount] = 1;

            Item\_Quantity[z] = Item\_Quantity[z] - customer\_quantity[customeritemcount];

            customer\_price[customeritemcount] = 0;

            customeritemcount++;

        }

    }

    return customeritemcount;

}

int order(int count\_item, int LIMIT, string Item\_Name[], int Item\_Price[], int Item\_Quantity[], int customer\_price[], string customer\_Item[], int customer\_quantity[], string dateA[])

{

    // emptying arrays for new customer

    for (int i = 0; i < LIMIT; i++)

    {

        customer\_price[i] = 0;

        customer\_Item[i] = "";

        customer\_quantity[i] = 0;

    }

    int customeritemcount = 0;

    string item, datemain;

    int n;

    gotoxy(0, 9);

    cout << "Enter Date (DD/MM/YYYY): " << endl;

    gotoxy(0, 10);

    cin >> datemain;

    dateA[0] = datefunc(datemain);

    gotoxy(35, 9);

    cout << "No.of items to add : " << endl;

    gotoxy(35, 10);

    cin >> n;

    for (int i = 0; i < n; i++)

    {

        cout << endl;

        cout << "Select food item (Enter \"0\" to end )  :  ";

        cin.ignore();

        getline(cin, item);

        cout << endl;

        if (item == "0")

        {

            break;

        }

        bool flag = false;

        for (int z = 0; z < count\_item; z++)

        {

            if (item == Item\_Name[z])

            {

                // making a new array to calculate invoice

                customer\_Item[customeritemcount] = item;

                cout << "Enter the quantity : ";

                cin >> customer\_quantity[customeritemcount];

                // if the item is in less quantity than that user want

                if (customer\_quantity[customeritemcount] > Item\_Quantity[z])

                {

                    cout << "Item not available in this quantity  " << endl;

                    cout << endl;

                    cout << "Available quantity  " << Item\_Quantity[z] << endl;

                    i--;

                    flag = true;

                    break;

                }

                // Minusing the quantity from manager item array which cusomer purchased

                Item\_Quantity[z] = Item\_Quantity[z] - customer\_quantity[customeritemcount];

                customer\_price[customeritemcount] = customer\_quantity[customeritemcount] \* Item\_Price[z];

                cout << "\t\t\t" << Item\_Name[z] << "\t" << customer\_price[customeritemcount] << "  rupees" << endl;

                customeritemcount++;

                flag = true;

            }

        }

        if (flag == true)

        {

            flag = false;

        }

        else

        {

            cout << " Sorry! Item is not in our menu." << endl;

            i--;

        }

    }

    return customeritemcount;

}

float calculatebill(int customeritemcount, int customer\_price[])

{

    float total;

    for (int i = 0; i < customeritemcount; i++)

    {

        total = total + customer\_price[i];

    }

    return total;

}

void receipt(int cicount, int customer\_price[], string customer\_Item[], int customer\_quantity[], string purchase\_date[])

{

    gotoxy(0, 9);

    cout << "Item Name" << endl;

    gotoxy(30, 9);

    cout << "Item Quantity" << endl;

    gotoxy(60, 9);

    cout << "Item Price" << endl;

    int y = 11;

    for (int i = 0; i < cicount; i++)

    {

        int x = 0;

        gotoxy(x, y);

        cout << customer\_Item[i] << endl;

        x = x + 30;

        gotoxy(x, y);

        cout << customer\_quantity[i] << endl;

        x = x + 30;

        gotoxy(x, y);

        cout << customer\_price[i] << endl;

        y++;

    }

    cout << endl;

    cout << "\t\t\t\t\t"

         << "Total Bill is    : " << calculatebill(cicount, customer\_price) << " \\-Rs";

    cout << endl;

    cout << "\t\t\t\t\t"

         << "Total Payable is : " << discount(calculatebill(cicount, customer\_price)) << " \\-Rs" << endl;

}

float discount(int bill)

{

    float discount, total;

    if (bill >= 1000 && bill < 2000)

    {

        discount = bill \* 5 / 100.0;

        return bill - discount;

    }

    else if (bill >= 2000 && bill < 3000)

    {

        discount = bill \* 15 / 100.0;

        return bill - discount;

    }

    else if (bill >= 3000)

    {

        discount = bill \* 20 / 100.0;

        return bill - discount;

    }

}

bool delievery(int index, int delieverycount, string idA[], string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[])

{

    string option;

    cout << endl;

    cout << "1.Take Away       (Press 1)" << endl;

    cout << "2.Deleivery       (Press 2)" << endl;

    cin >> option;

    if (option == "1")

    {

        cout << "Wait for 10 to 20 minutes" << endl;

        cout << endl;

        cout << "Thanks for visiting our restaurant" << endl;

        return false;

    }

    else if (option == "2")

    {

        cout << "  Your Order will be delievered in 30 to 45 minutes" << endl;

        string address;

        delieveryquene[delieverycount] = idA[index];

        delieverystatus[delieverycount] = "undelievered";

        cout << endl;

        cout << "Your distance(KM) from restaurant : ";

        cin >> delieverydistance[delieverycount];

        cout << endl;

        cout << "Enter your  Address : " << endl;

        cin.ignore();

        getline(cin, delieveryaddress[delieverycount]);

        cout << endl;

        int op;

        cout << "Press 1 to give tip    Or  ";

        cout << "Press 2 to continue  " << endl;

        cin >> op;

        if (op == 1)

        {

            cout << "enter tip : ";

            cin >> delieverytip[delieverycount];

        }

        cout << endl;

        cout << "Thanks for visiting our restaurant" << endl;

        return true;

    }

}

int managerinvoice(int customer\_price[], int cicount, float paid[], string namemanager[], string purchase\_date[], string dateA[], int customer\_count, int index, string idA[])

{

    if (customer\_price[0] > 0)

    {

        float y = discount(calculatebill(cicount, customer\_price));

        paid[customer\_count] = y;

        purchase\_date[customer\_count] = dateA[0];

        namemanager[customer\_count] = idA[index];

        customer\_count++;

    }

    dateA[0] = "";

    return customer\_count;

}

void printinvoice(int customer\_count, string namemanager[], string purchase\_date[], float paid[])

{

    gotoxy(0, 9);

    cout << "Sr.no" << endl;

    gotoxy(15, 9);

    cout << "Customer" << endl;

    gotoxy(35, 9);

    cout << "Bill Paid" << endl;

    gotoxy(55, 9);

    cout << "Date" << endl;

    int y = 11;

    for (int i = 0; i < customer\_count; i++)

    {

        gotoxy(0, y);

        cout << i + 1 << endl;

        gotoxy(15, y);

        cout << namemanager[i] << endl;

        gotoxy(35, y);

        cout << paid[i] << endl;

        gotoxy(55, y);

        cout << purchase\_date[i] << endl;

        y++;

    }

}

void addfeedback(int count\_feedback, string feedback[], int index, string idA[], string namefeedback[])

{

    cout << endl;

    cout << "Enter your reviews & valuable feedback " << endl;

    cin.ignore();

    getline(cin, feedback[count\_feedback]);

    namefeedback[count\_feedback] = idA[index];

    cout << " " << endl;

}

void viewfeedback(int count\_feedback, string feedback[], string namefeedback[])

{

    gotoxy(0, 9);

    cout << "Sr.no" << endl;

    gotoxy(10, 9);

    cout << "Customer" << endl;

    gotoxy(30, 9);

    cout << "Reviews & Feedbacks" << endl;

    int y = 11;

    for (int i = 0; i < count\_feedback; i++)

    {

        gotoxy(0, y);

        cout << i + 1 << endl;

        gotoxy(10, y);

        cout << namefeedback[i] << endl;

        gotoxy(30, y);

        cout << feedback[i] << endl;

        y++;

    }

    cout << endl;

}

int orders(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[])

{

    gotoxy(0, 9);

    cout << "Sr No." << endl;

    gotoxy(20, 9);

    cout << "Customer" << endl;

    gotoxy(40, 9);

    cout << "Order" << endl;

    gotoxy(60, 9);

    cout << "Distance" << endl;

    int y = 11;

    for (int i = 0; i < delieverycount; i++)

    {

        gotoxy(0, y);

        cout << i + 1 << endl;

        gotoxy(20, y);

        cout << delieveryquene[i] << endl;

        gotoxy(40, y);

        cout << delieverystatus[i] << endl;

        gotoxy(60, y);

        cout << delieverydistance[i] << endl;

        gotoxy(65, y);

        cout << "KM" << endl;

        y++;

    }

    int order;

    y = y + 1;

    gotoxy(0, y);

    cout << "Select the order no. : ";

    cin >> order;

    gotoxy(35, y);

    y++;

    while (delieverydistance[order - 1] == 0)

    {

        gotoxy(0, y);

        cout << "Select valid order no. : ";

        cin >> order;

        gotoxy(35, y);

        y++;

    }

    cout << delieveryaddress[order - 1] << endl;

    cout << endl;

    return order - 1;

}

void selectedorders(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[])

{

    gotoxy(0, 9);

    cout << "Sr No." << endl;

    gotoxy(20, 9);

    cout << "Customer" << endl;

    gotoxy(40, 9);

    cout << "status" << endl;

    gotoxy(60, 9);

    cout << "Distance" << endl;

    int y = 11;

    gotoxy(0, y);

    cout << "1" << endl;

    gotoxy(20, y);

    cout << delieveryquene[delievery\_index] << endl;

    gotoxy(40, y);

    cout << delieverystatus[delievery\_index] << endl;

    gotoxy(60, y);

    cout << delieverydistance[delievery\_index] << endl;

    gotoxy(65, y);

    cout << "KM" << endl;

    y = y + 2;

    gotoxy(30, y);

    cout << delieveryaddress[delievery\_index] << endl;

    cout << endl;

}

bool delievery\_status(int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[])

{

    gotoxy(0, 9);

    cout << "Sr No." << endl;

    gotoxy(20, 9);

    cout << "Customer" << endl;

    gotoxy(40, 9);

    cout << "Status" << endl;

    gotoxy(60, 9);

    cout << "Distance" << endl;

    int y = 11;

    gotoxy(0, y);

    cout << "1" << endl;

    gotoxy(20, y);

    cout << delieveryquene[delievery\_index] << endl;

    gotoxy(40, y);

    cout << delieverystatus[delievery\_index] << endl;

    gotoxy(60, y);

    cout << delieverydistance[delievery\_index] << endl;

    gotoxy(65, y);

    cout << "KM" << endl;

    y = y + 3;

    gotoxy(35, y);

    cout << delieveryaddress[delievery\_index] << endl;

    cout << endl;

    y = y + 2;

    gotoxy(10, y);

    int op;

    cout << "      Press 1 to update Delivery Status    " << endl;

    y = y + 1;

    gotoxy(10, y);

    cout << "                      OR " << endl;

    y = y + 1;

    gotoxy(10, y);

    cout << "                   Continue" << endl;

    y = y + 1;

    gotoxy(10, y);

    cin >> op;

    bool flag = false;

    if (op == 1)

    {

        delieverystatus[delievery\_index] = "delievered";

        flag = true;

    }

    return flag;

}

int removedelievered(int newcount, int count\_item, int delievery\_index, int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[], string delname[], string deladdress[], float delbill[], float deltip[])

{

    delname[newcount] = delieveryquene[delievery\_index];

    deladdress[newcount] = delieveryaddress[delievery\_index];

    deltip[newcount] = delieverytip[delievery\_index];

    delbill[newcount] = delieverydistance[delievery\_index] \* 50.0;

    for (int i = delievery\_index; i < delieverycount; i++)

    {

        delieveryquene[i] = delieveryquene[i + 1];

        delieverystatus[i] = delieverystatus[i + 1];

        delieverydistance[i] = delieverydistance[i + 1];

        delieveryaddress[i] = delieveryaddress[i + 1];

        delieverytip[i] = delieverytip[i + 1];

    }

    newcount++;

    return newcount;

}

void earnings(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    gotoxy(0, 9);

    cout << "No." << endl;

    gotoxy(5, 9);

    cout << "Name" << endl;

    gotoxy(20, 9);

    cout << "Address" << endl;

    gotoxy(50, 9);

    cout << "Payment" << endl;

    gotoxy(65, 9);

    cout << "Tip" << endl;

    int y = 11;

    for (int i = 0; i < newcount; i++)

    {

        gotoxy(0, y);

        cout << i + 1 << endl;

        gotoxy(5, y);

        cout << delname[i] << endl;

        gotoxy(20, y);

        cout << deladdress[i] << endl;

        gotoxy(50, y);

        cout << delbill[i] << endl;

        gotoxy(65, y);

        cout << deltip[i] << endl;

        y++;

    }

    float total = calcearnings(newcount, delbill, deltip);

    y = y + 2;

    gotoxy(20, y);

    cout << "Total earnings are " << endl;

    gotoxy(50, y);

    cout << total << " \\- rupees" << endl;

    cout << endl;

}

float calcearnings(int newcount, float delbill[], float deltip[])

{

    float total;

    for (int i = 0; i < newcount; i++)

    {

        total = total + delbill[i] + deltip[i];

    }

    return total;

}

string datefunc(string c)

{

    int count = 0;

    while (c[count] != '\0')

    {

        count++;

    }

    while (count != 8)

    {

        system("cls");

        header();

        cout << "                    C U T O M E R  " << endl;

        cout << " " << endl;

        cout << "C U S T O M E R    M e n u >> Select the order" << endl;

        cout << " " << endl;

        gotoxy(0, 9);

        cout << "Enter valid Date (DD/MM/YYYY): " << endl;

        gotoxy(0, 10);

        cin >> c;

        gotoxy(35, 9);

        count = 0;

        while (c[count] != '\0')

        {

            count++;

        }

    }

    while (date(c) != true)

    {

        system("cls");

        header();

        cout << "                    C U T O M E R  " << endl;

        cout << " " << endl;

        cout << "C U S T O M E R    M e n u >> Select the order" << endl;

        cout << " " << endl;

        gotoxy(0, 9);

        cout << "Enter valid Date (DD/MM/YYYY): " << endl;

        gotoxy(0, 10);

        cin >> c;

        gotoxy(35, 9);

        int count = 0;

        while (c[count] != '\0')

        {

            count++;

        }

        while (count != 8)

        {

            system("cls");

            header();

            cout << "                    C U T O M E R  " << endl;

            cout << " " << endl;

            cout << "C U S T O M E R    M e n u >> Select the order" << endl;

            cout << " " << endl;

            gotoxy(0, 9);

            cout << "Enter valid Date (DD/MM/YYYY): " << endl;

            gotoxy(0, 10);

            cin >> c;

            gotoxy(35, 9);

            count = 0;

            while (c[count] != '\0')

            {

                count++;

            }

        }

        date(c);

    }

    char word[10];

    int z = 0;

    for (int i = 0; i < 11; i++)

    {

        if (i == 2 || i == 5)

        {

            word[i] = '-';

            z--;

        }

        else

        {

            word[i] = c[z];

        }

        z++;

    }

    return word;

}

bool date(string c)

{

    string a, b, e;

    int x = 0, y = 0, z = 0;

    int i = 0;

    while (c[i] != '\0')

    {

        i++;

        if (i == 2)

        {

            for (int i = 0; i < 2; i++)

            {

                a[i] = c[i];

            }

            x = stoi(a);

        }

        if (i == 4)

        {

            int loop = 0;

            for (int i = 2; i < 4; i++)

            {

                b[loop] = c[i];

                loop++;

            }

            y = stoi(b);

        }

        if (i == 8)

        {

            int loop = 0;

            for (int i = 4; i < 8; i++)

            {

                e[loop] = c[i];

                loop++;

            }

            z = stoi(e);

        }

    }

    bool flag = false;

    if (x > 0 && x < 31 && y > 0 && y < 13 && z == 2022)

    {

        flag = true;

    }

    return flag;

}

void savedata(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    fstream file;

    file.open("items.txt", ios::out);

    for (int i = 0; i < count\_item; i++)

    {

        file << Item\_Name[i] << "," << Item\_Price[i] << "," << Item\_Quantity[i] << endl;

    }

    file.close();

}

int datareader(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    count\_item = 0;

    fstream file;

    file.open("items.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                Item\_Name[count] = parsedata(line, 1);

                Item\_Price[count] = stoi(parsedata(line, 2));

                Item\_Quantity[count] = stoi(parsedata(line, 3));

                count\_item++;

                count++;

            }

        }

        file.close();

    }

    return count\_item;

}

string parsedata(string line, int count)

{

    string line1 = "";

    int number = 0;

    int commas1 = 1;

    for (int z = 0; line[z] != '\0'; z++)

    {

        if (line[z] == ',')

        {

            commas1++;

            continue;

        }

        if (commas1 == count)

        {

            line1 += line[z];

        }

    }

    return line1;

}

void logindata(int users, string idA[], string passA[], string roleA[])

{

    fstream file;

    file.open("users.txt", ios::out);

    for (int i = 0; i < users; i++)

    {

        file << idA[i] << "," << passA[i] << "," << roleA[i] << endl;

    }

    file.close();

}

int readlogindata(int users, string idA[], string passA[], string roleA[])

{

    users = 0;

    fstream file;

    file.open("users.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                idA[count] = parsedata(line, 1);

                passA[count] = parsedata(line, 2);

                roleA[count] = parsedata(line, 3);

                users++;

                count++;

            }

        }

        file.close();

    }

    return users;

}

void delboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    fstream file;

 gotoxy(20, y);

    cout << "Total earnings are " << endl;

    gotoxy(50, y);

    cout << total << " \\- rupees" << endl;

    cout << endl;

}

float calcearnings(int newcount, float delbill[], float deltip[])

{

    float total;

    for (int i = 0; i < newcount; i++)

    {

        total = total + delbill[i] + deltip[i];

    }

    return total;

}

string datefunc(string c)

{

    int count = 0;

    while (c[count] != '\0')

    {

        count++;

    }

    while (count != 8)

    {

        system("cls");

        header();

        cout << "                    C U T O M E R  " << endl;

        cout << " " << endl;

        cout << "C U S T O M E R    M e n u >> Select the order" << endl;

        cout << " " << endl;

        gotoxy(0, 9);

        cout << "Enter valid Date (DD/MM/YYYY): " << endl;

        gotoxy(0, 10);

        cin >> c;

        gotoxy(35, 9);

        count = 0;

        while (c[count] != '\0')

        {

            count++;

        }

    }

    while (date(c) != true)

    {

        system("cls");

        header();

        cout << "                    C U T O M E R  " << endl;

        cout << " " << endl;

        cout << "C U S T O M E R    M e n u >> Select the order" << endl;

        cout << " " << endl;

        gotoxy(0, 9);

        cout << "Enter valid Date (DD/MM/YYYY): " << endl;

        gotoxy(0, 10);

        cin >> c;

        gotoxy(35, 9);

        int count = 0;

        while (c[count] != '\0')

        {

            count++;

        }

        while (count != 8)

        {

            system("cls");

            header();

            cout << "                    C U T O M E R  " << endl;

            cout << " " << endl;

            cout << "C U S T O M E R    M e n u >> Select the order" << endl;

            cout << " " << endl;

            gotoxy(0, 9);

            cout << "Enter valid Date (DD/MM/YYYY): " << endl;

            gotoxy(0, 10);

            cin >> c;

            gotoxy(35, 9);

            count = 0;

            while (c[count] != '\0')

            {

                count++;

            }

        }

        date(c);

    }

    char word[10];

    int z = 0;

    for (int i = 0; i < 11; i++)

    {

        if (i == 2 || i == 5)

        {

            word[i] = '-';

            z--;

        }

        else

        {

            word[i] = c[z];

        }

        z++;

    }

    return word;

}

bool date(string c)

{

    string a, b, e;

    int x = 0, y = 0, z = 0;

    int i = 0;

    while (c[i] != '\0')

    {

        i++;

        if (i == 2)

        {

            for (int i = 0; i < 2; i++)

            {

                a[i] = c[i];

            }

            x = stoi(a);

        }

        if (i == 4)

        {

            int loop = 0;

            for (int i = 2; i < 4; i++)

            {

                b[loop] = c[i];

                loop++;

            }

            y = stoi(b);

        }

        if (i == 8)

        {

            int loop = 0;

            for (int i = 4; i < 8; i++)

            {

                e[loop] = c[i];

                loop++;

            }

            z = stoi(e);

        }

    }

    bool flag = false;

    if (x > 0 && x < 31 && y > 0 && y < 13 && z == 2022)

    {

        flag = true;

    }

    return flag;

}

void savedata(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    fstream file;

    file.open("items.txt", ios::out);

    for (int i = 0; i < count\_item; i++)

    {

        file << Item\_Name[i] << "," << Item\_Price[i] << "," << Item\_Quantity[i] << endl;

    }

    file.close();

}

int datareader(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    count\_item = 0;

    fstream file;

    file.open("items.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                Item\_Name[count] = parsedata(line, 1);

                Item\_Price[count] = stoi(parsedata(line, 2));

                Item\_Quantity[count] = stoi(parsedata(line, 3));

                count\_item++;

                count++;

            }

        }

        file.close();

    }

    return count\_item;

}

string parsedata(string line, int count)

{

    string line1 = "";

    int number = 0;

    int commas1 = 1;

    for (int z = 0; line[z] != '\0'; z++)

    {

        if (line[z] == ',')

        {

            commas1++;

            continue;

        }

        if (commas1 == count)

        {

            line1 += line[z];

        }

    }

    return line1;

}

void logindata(int users, string idA[], string passA[], string roleA[])

{

    fstream file;

    file.open("users.txt", ios::out);

    for (int i = 0; i < users; i++)

    {

        file << idA[i] << "," << passA[i] << "," << roleA[i] << endl;

    }

    file.close();

}

int readlogindata(int users, string idA[], string passA[], string roleA[])

{

    users = 0;

    fstream file;

    file.open("users.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                idA[count] = parsedata(line, 1);

                passA[count] = parsedata(line, 2);

                roleA[count] = parsedata(line, 3);

                users++;

                count++;

            }

        }

        file.close();

    }

    return users;

}

void delboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    fstream file;

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

    for (int i = 0; i < newcount; i++)

    {

        file << delname[i] << "," << deladdress[i] << "," << delbill[i] << "," << deltip[i] << endl;

    }

    file.close();

}

int readdelboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    newcount = 0;

    fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                delname[count] = parsedata(line, 1);

                deladdress[count] = parsedata(line, 2);

                delbill[count] = stof(parsedata(line, 3));

                deltip[count] = stof(parsedata(line, 4));

                newcount++;

                count++;

            }

        }

        file.close();

    }

    return newcount;

}

void delquene(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[])

{

    fstream file;

    file.open("quene.txt", ios::out);

    for (int i = 0; i < delieverycount; i++)

    {

        file << delieveryquene[i] << "," << delieverystatus[i] << "," << delieverydistance[i] << "," << delieveryaddress[i] << "," << delieverytip[i] << endl;

    }

    file.close();

}

int readquenedata(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[])

{

    delieverycount = 0;

    fstream file;

    file.open("quene.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                delieveryquene[count] = parsedata(line, 1);

                delieverystatus[count] = parsedata(line, 2);

                delieverydistance[count] = stof(parsedata(line, 3));

                delieveryaddress[count] = parsedata(line, 4);

                delieverytip[count] = stof(parsedata(line, 5));

                delieverycount++;

                count++;

            }

        }

        file.close();

    }

    return delieverycount;

}

void history(int customer\_count, float paid[], string namemanager[], string purchase\_date[])

{

    fstream file;

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

    for (int i = 0; i < customer\_count; i++)

    {

        file << namemanager[i] << "," << paid[i] << "," << purchase\_date[i] << endl;

    }

    file.close();

}

int readhistory(int customer\_count, float paid[], string namemanager[], string purchase\_date[])

{

    customer\_count = 0;

    fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

        system("cls");

        header();

        cout << "                    C U T O M E R  " << endl;

        cout << " " << endl;

        cout << "C U S T O M E R    M e n u >> Select the order" << endl;

        cout << " " << endl;

        gotoxy(0, 9);

        cout << "Enter valid Date (DD/MM/YYYY): " << endl;

        gotoxy(0, 10);

        cin >> c;

        gotoxy(35, 9);

        int count = 0;

        while (c[count] != '\0')

        {

            count++;

        }

        while (count != 8)

        {

            system("cls");

            header();

            cout << "                    C U T O M E R  " << endl;

            cout << " " << endl;

            cout << "C U S T O M E R    M e n u >> Select the order" << endl;

            cout << " " << endl;

            gotoxy(0, 9);

            cout << "Enter valid Date (DD/MM/YYYY): " << endl;

            gotoxy(0, 10);

            cin >> c;

            gotoxy(35, 9);

            count = 0;

            while (c[count] != '\0')

            {

                count++;

            }

        }

        date(c);

    }

    char word[10];

    int z = 0;

    for (int i = 0; i < 11; i++)

    {

        if (i == 2 || i == 5)

        {

            word[i] = '-';

            z--;

        }

        else

        {

            word[i] = c[z];

        }

        z++;

    }

    return word;

}

 for (int i = 0; i < 11; i++)

    {

        if (i == 2 || i == 5)

        {

            word[i] = '-';

            z--;

        }

        else

        {

            word[i] = c[z];

        }

        z++;

    }

    return word;

}

bool date(string c)

{

    string a, b, e;

    int x = 0, y = 0, z = 0;

    int i = 0;

    while (c[i] != '\0')

    {

        i++;

        if (i == 2)

        {

            for (int i = 0; i < 2; i++)

            {

                a[i] = c[i];

            }

            x = stoi(a);

        }

        if (i == 4)

        {

            int loop = 0;

            for (int i = 2; i < 4; i++)

            {

                b[loop] = c[i];

                loop++;

            }

            y = stoi(b);

        }

        if (i == 8)

        {

            int loop = 0;

            for (int i = 4; i < 8; i++)

            {

                e[loop] = c[i];

                loop++;

            }

            z = stoi(e);

        }

    }

    bool flag = false;

 int loop = 0;

            for (int i = 4; i < 8; i++)

            {

                e[loop] = c[i];

                loop++;

            }

            z = stoi(e);

        }

    }

    bool flag = false;

    if (x > 0 && x < 31 && y > 0 && y < 13 && z == 2022)

    {

        flag = true;

    }

    return flag;

}

void savedata(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    fstream file;

    file.open("items.txt", ios::out);

    for (int i = 0; i < count\_item; i++)

    {

        file << Item\_Name[i] << "," << Item\_Price[i] << "," << Item\_Quantity[i] << endl;

    }

    file.close();

}

int datareader(int count\_item, string Item\_Name[], int Item\_Price[], int Item\_Quantity[])

{

    count\_item = 0;

    fstream file;

    file.open("items.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

 {

            getline(file, line);

            if (line != "")

            {

      Item\_Name[count] = parsedata(line, 1);

                Item\_Price[count] = stoi(parsedata(line, 2));

                Item\_Quantity[count] = stoi(parsedata(line, 3));

                count\_item++;

                count++;

            }

        }

        file.close();

    }

    return count\_item;

}

string parsedata(string line, int count)

{

    string line1 = "";

    int number = 0;

    int commas1 = 1;

    for (int z = 0; line[z] != '\0'; z++)

    {

        if (line[z] == ',')

        {

            commas1++;

            continue;

        }

        if (commas1 == count)

        {

            line1 += line[z];

        }

    }

    return line1;

}

void logindata(int users, string idA[], string passA[], string roleA[])

{

    fstream file;

    file.open("users.txt", ios::out);

    for (int i = 0; i < users; i++)

    {

     file << idA[i] << "," << passA[i] << "," << roleA[i] << endl;

    }

    file.close();

}

int readlogindata(int users, string idA[], string passA[], string roleA[])

{

    users = 0;

    fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                idA[count] = parsedata(line, 1);

                passA[count] = parsedata(line, 2);

                roleA[count] = parsedata(line, 3);

                users++;

                count++;

            }

        }

        file.close();

    }

    return users;

}

void delboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    fstream file;

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

    for (int i = 0; i < newcount; i++)

    {

        file << delname[i] << "," << deladdress[i] << "," << delbill[i] << "," << deltip[i] << endl;

    }

    file.close();

}

int readdelboydata(int newcount, string delname[], string deladdress[], float delbill[], float deltip[])

{

    newcount = 0;

    fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                delname[count] = parsedata(line, 1);

                deladdress[count] = parsedata(line, 2);

                delbill[count] = stof(parsedata(line, 3));

                deltip[count] = stof(parsedata(line, 4));

                newcount++;

                count++;

            }

        }

        file.close();

    }

    return newcount;

}

void delquene(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[])

{

    fstream file;

    file.open("quene.txt", ios::out);

    for (int i = 0; i < delieverycount; i++)

    {

        file << delieveryquene[i] << "," << delieverystatus[i] << "," << delieverydistance[i] << "," << delieveryaddress[i] << "," << delieverytip[i] << endl;

    }

    file.close();

}

int readquenedata(int delieverycount, string delieveryquene[], string delieverystatus[], float delieverydistance[], string delieveryaddress[], float delieverytip[])

{

    delieverycount = 0;

    fstream file;

    file.open("quene.txt", ios::in);

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                delieveryquene[count] = parsedata(line, 1);

                delieverystatus[count] = parsedata(line, 2);

                delieverydistance[count] = stof(parsedata(line, 3));

                delieveryaddress[count] = parsedata(line, 4);

                delieverytip[count] = stof(parsedata(line, 5));

                delieverycount++;

                count++;

            }

        }

        file.close();

    }

    return delieverycount;

}

void history(int customer\_count, float paid[], string namemanager[], string purchase\_date[])

{

    fstream file;

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

    for (int i = 0; i < customer\_count; i++)

    {

        file << namemanager[i] << "," << paid[i] << "," << purchase\_date[i] << endl;

    }

    file.close();

}

int readhistory(int customer\_count, float paid[], string namemanager[], string purchase\_date[])

{

    customer\_count = 0;

    fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                namemanager[count] = parsedata(line, 1);

                paid[count] = stof(parsedata(line, 2));

                purchase\_date[count] = parsedata(line, 3);

                customer\_count++;

                count++;

            }

        }

        file.close();

    }

    return customer\_count;

}

    {

        cout << "file not found";

    }

    else

    {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                namemanager[count] = parsedata(line, 1);

                paid[count] = stof(parsedata(line, 2));

                purchase\_date[count] = parsedata(line, 3);

                customer\_count++;

                count++;

            }

        }

        file.close();

    }

    return customer\_count;

}

void storefeedback(int count\_feedback, string namefeedback[], string feedback[])

{

    fstream file;

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

    for (int i = 0; i < count\_feedback; i++)

    {

        file << namefeedback[i] << "," << feedback[i] << endl;

    }

    file.close();

}

int readhistory(int count\_feedback, string namefeedback[], string feedback[])

{

    count\_feedback = 0;

 fstream file;

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

    if (!file)

    {

        cout << "file not found";

    }

    else

 {

        string line;

        int count = 0;

        while (!file.eof())

        {

            getline(file, line);

            if (line != "")

            {

                namefeedback[count] = parsedata(line, 1);

                feedback[count] = parsedata(line, 2);

                count\_feedback++;

                count++;

            }

        }

        file.close();

    }

    return count\_feedback;

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting **Grade:** | A | 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. | B | 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 - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -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:** | A | 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:** | A | 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:** | A | 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:** | A | 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. |
| Sorting Features  **Grade:** | A | Sorting Feature is working but sorted data is not useful for project. | Sorting feature is partial implemented | Project do not contain sorting |
| Modularity  **Grade:** | A | 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)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| Validations  **Grade:** | A | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| Recommendation Feature | Proper meaning full recommendation is present into system | Partial Recommendation is implemented | C | Not implemented |
| Presentation and Demo  **Grade:** | A | 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:** | A | 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. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 - **Algorithms** and Flow Charts of all functions- **Test Cases** are defined -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. |
| Sorting Features  **Grade:** | Sort working 100% and generating useful report | Sorting Feature is working but sorted data is not useful for project. | Sorting feature is partial implemented | Project do not contain sorting |
| 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)- Demo Data Functionality Added-At least Two Unit Tests are defined. | | | | |
| 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 |
| Recommendation Feature | Proper meaning full recommendation is present into system | Partial Recommendation is implemented | Implemented but not meaning full. | 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. |