**Daraz Online Management System**



Session: 2021 – 2025

**Submitted by:**

Abdul Mateen 2021-CS-190

**Supervised to:**

Dr. Maida Shahid

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**DARAZ**

**Online Management System**

**DESCRIPTION:**

This Software is based on an E-commerce. . It is basically for management of Products for Admin, Seller and buyer .It helps Admin and Seller to keep record of their shop. By using this program it has become easy to buy and sell products online. You can buy any products online using this program. It has basically three users Admin, Seller and Buyer. People can create their account and as Seller and can expand their business online. People instead of going out and buying Products get their order delivered at home .It is basically has a large amount of uses.

**Users of the Program:**

There are three users of this Program:

* ADMIN (To manage the seller account )
* SELLER (To sell products online )
* CUSTOMER ( To buy products online )

**Functional Requirements:**

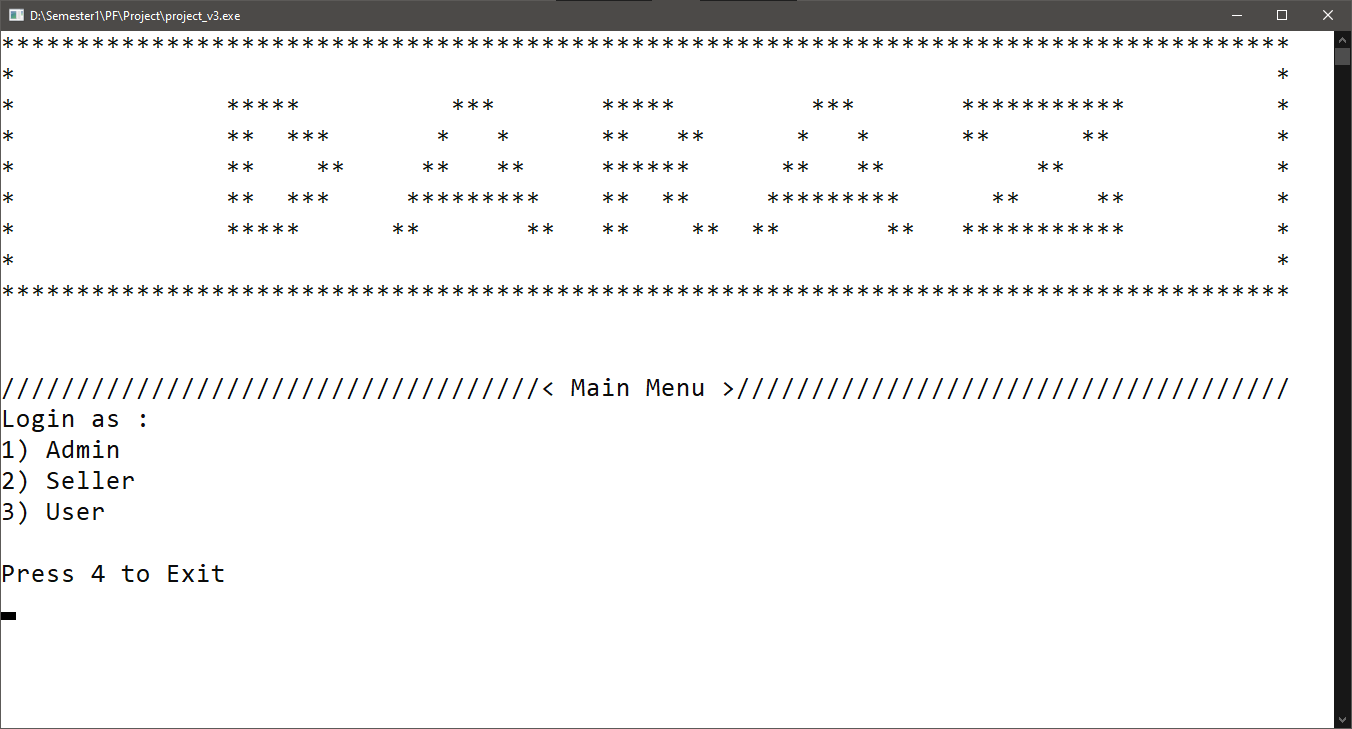
There are some basic functional requirements of this program:

1. The admin Password is 1234 (Which you can change later).
2. It is basically a totally working program to maintain business.
3. Admin can delete seller account.
4. Only admin can create seller’s account.
5. Only seller can sell products.
6. This program can handle large number of products.
7. Seller can delete products.
8. Seller can update price of added products.
9. Seller can deliver products to customer.
10. Customer needs to login to view products.
11. Customer can view products and order it

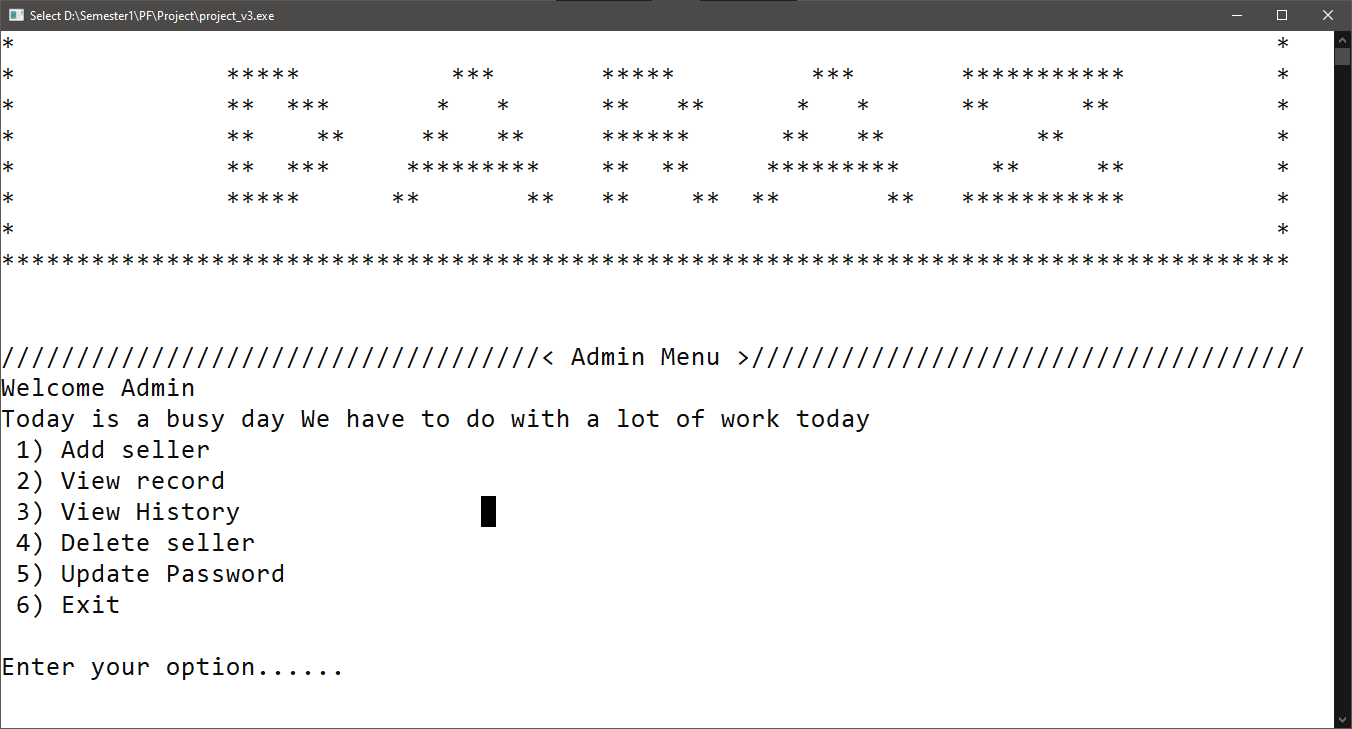
**Wireframes:**

**(Admin Menu)**

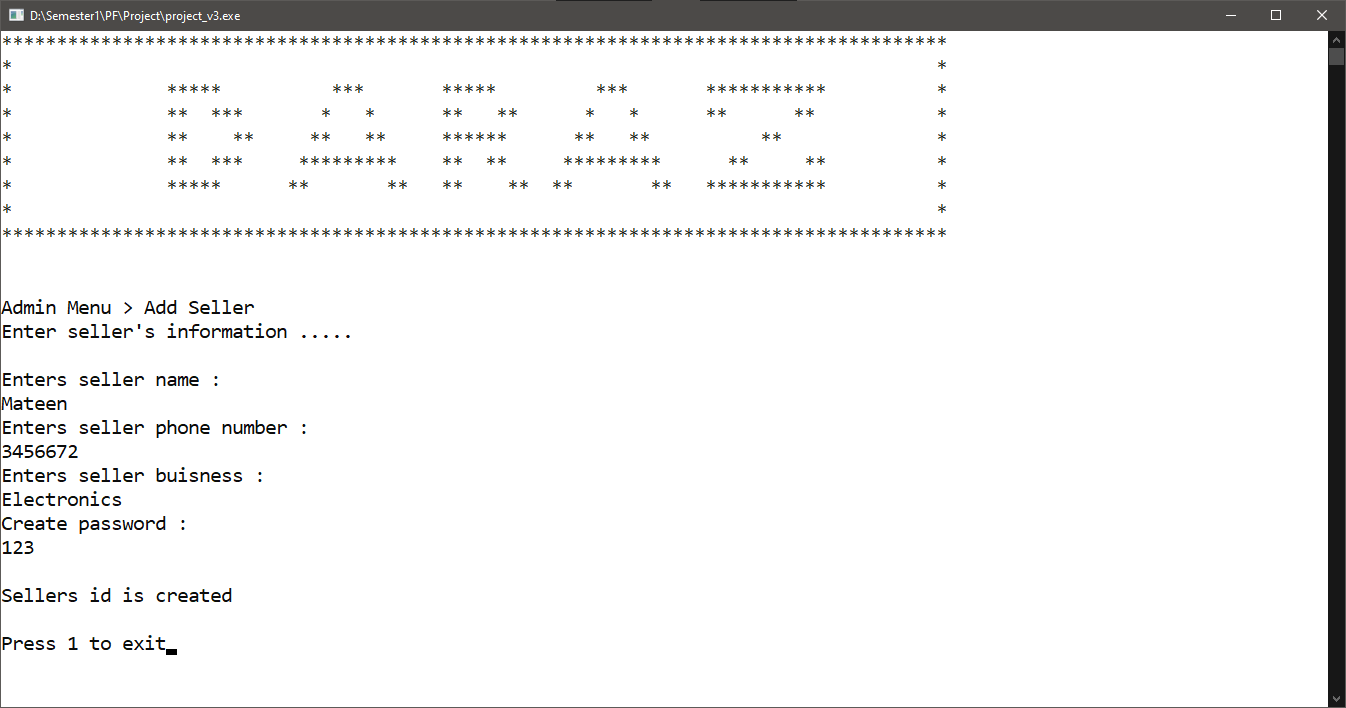
Login Menu:



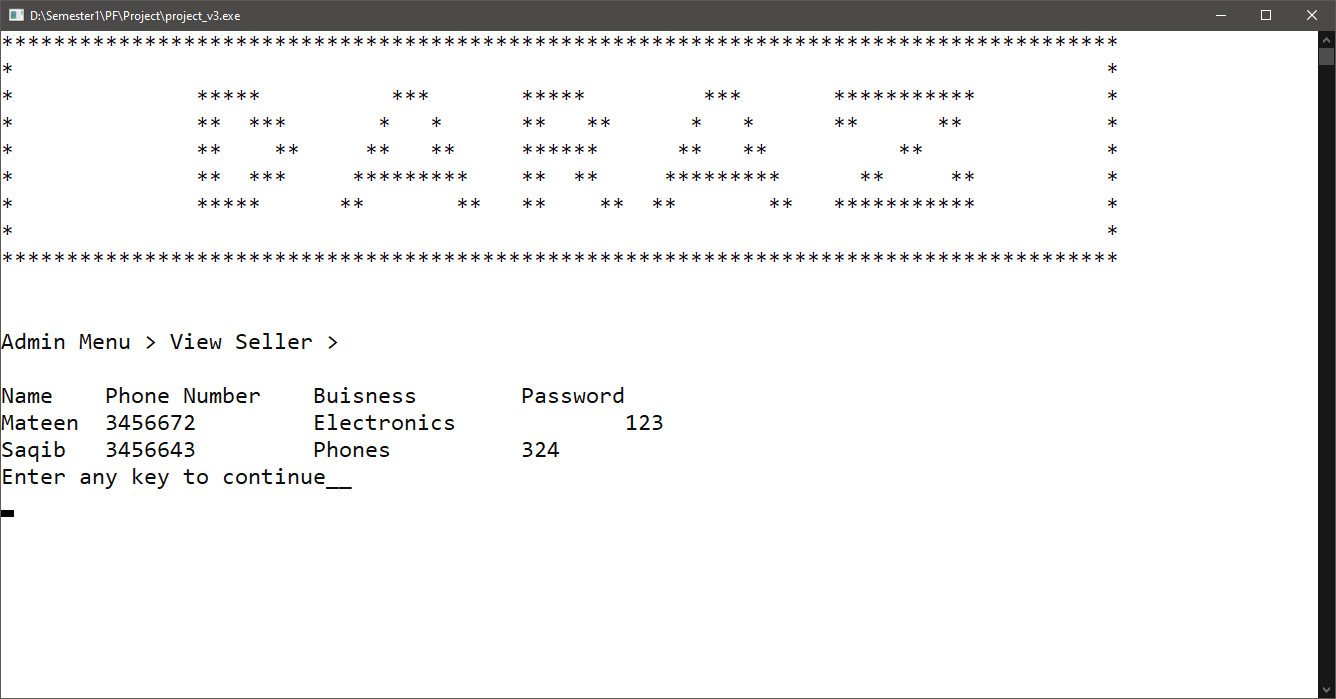
Admin menu:



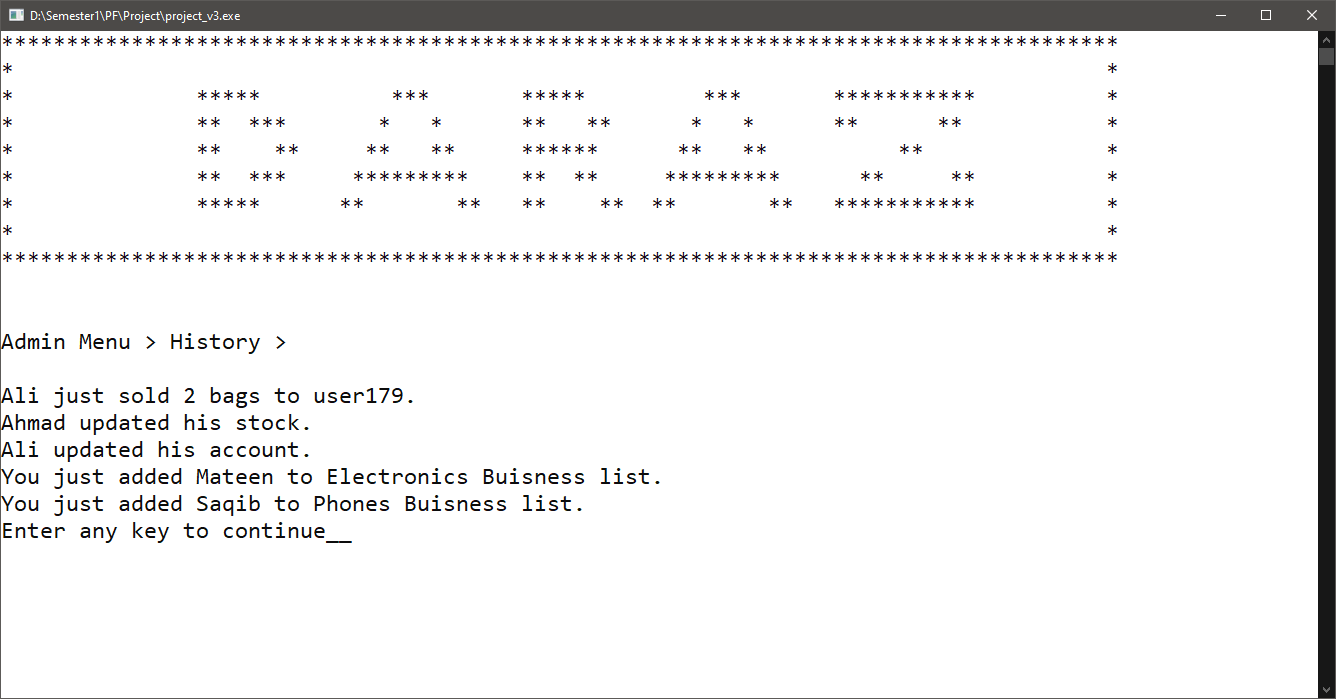
Add Product:



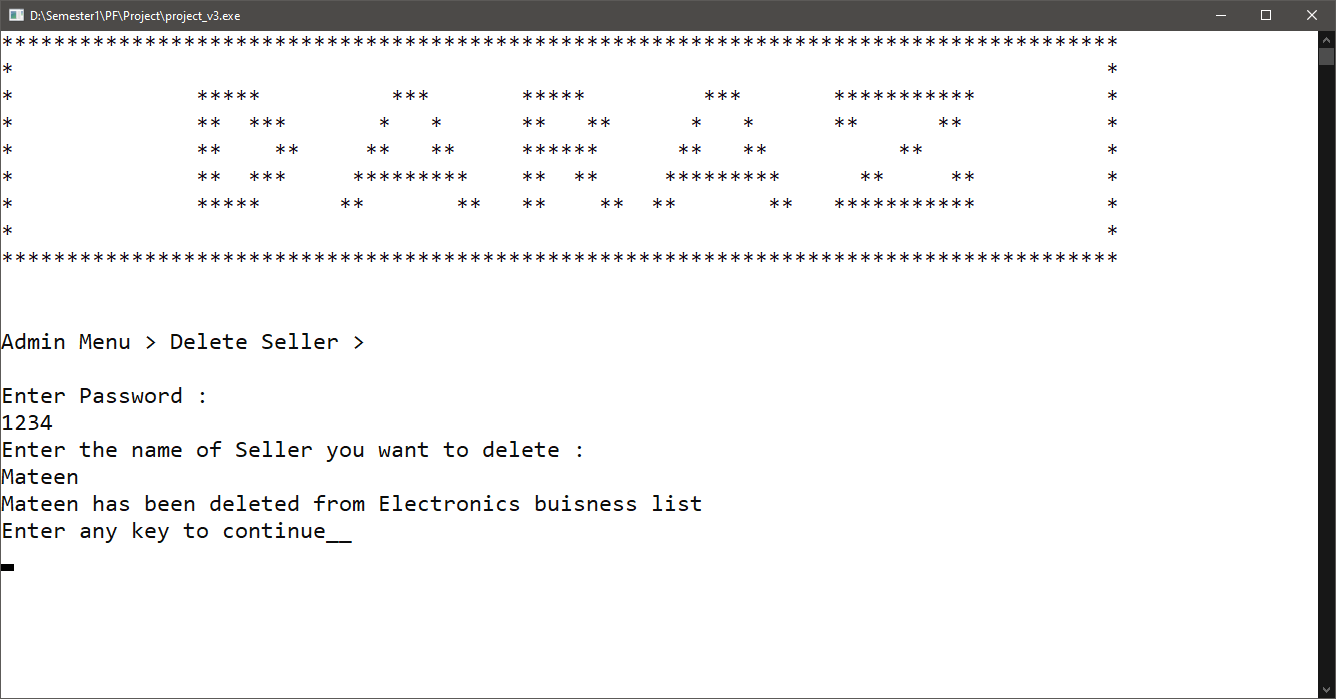
View Sellers:



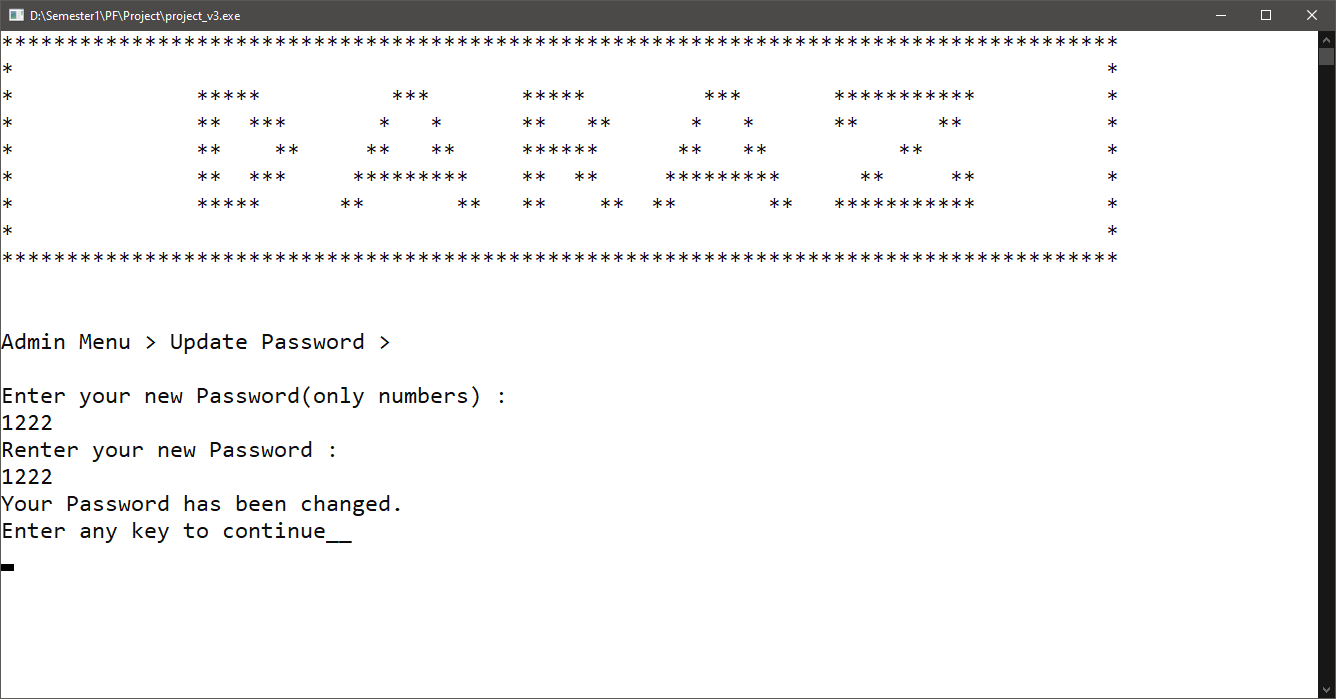
View History:



Delete Seller:

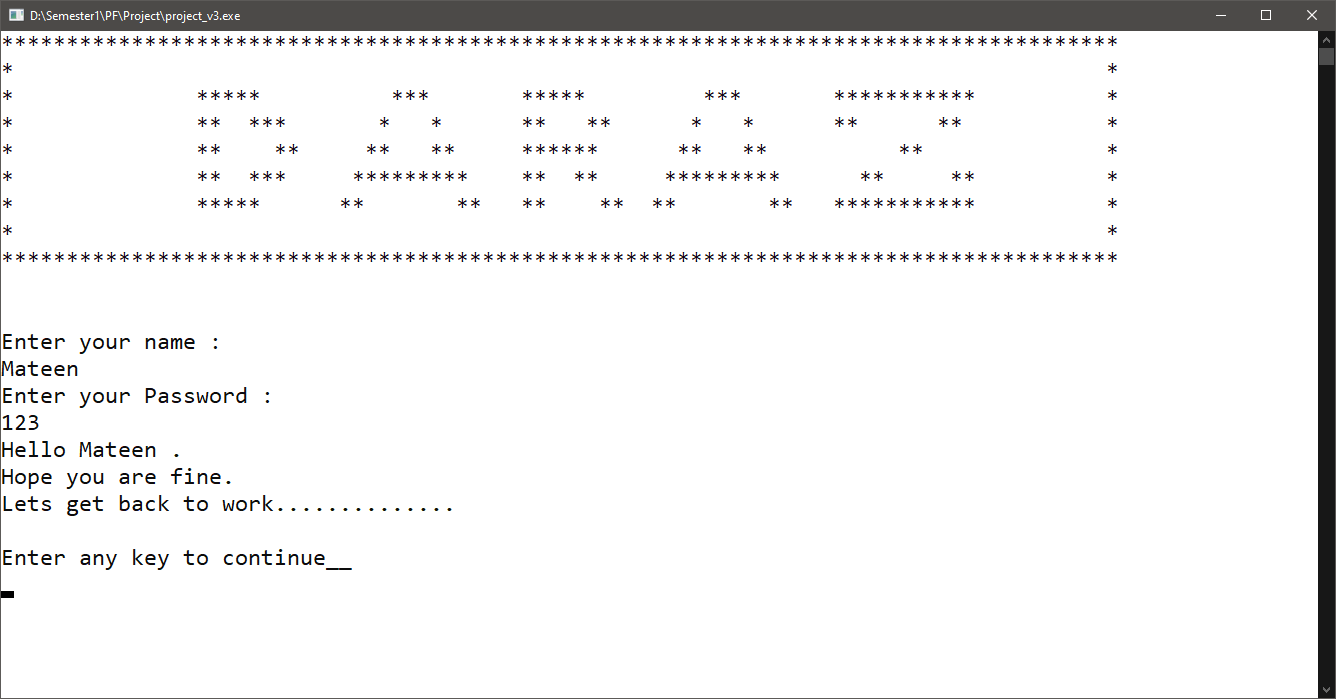


Update Password:



**(Seller Menu)**

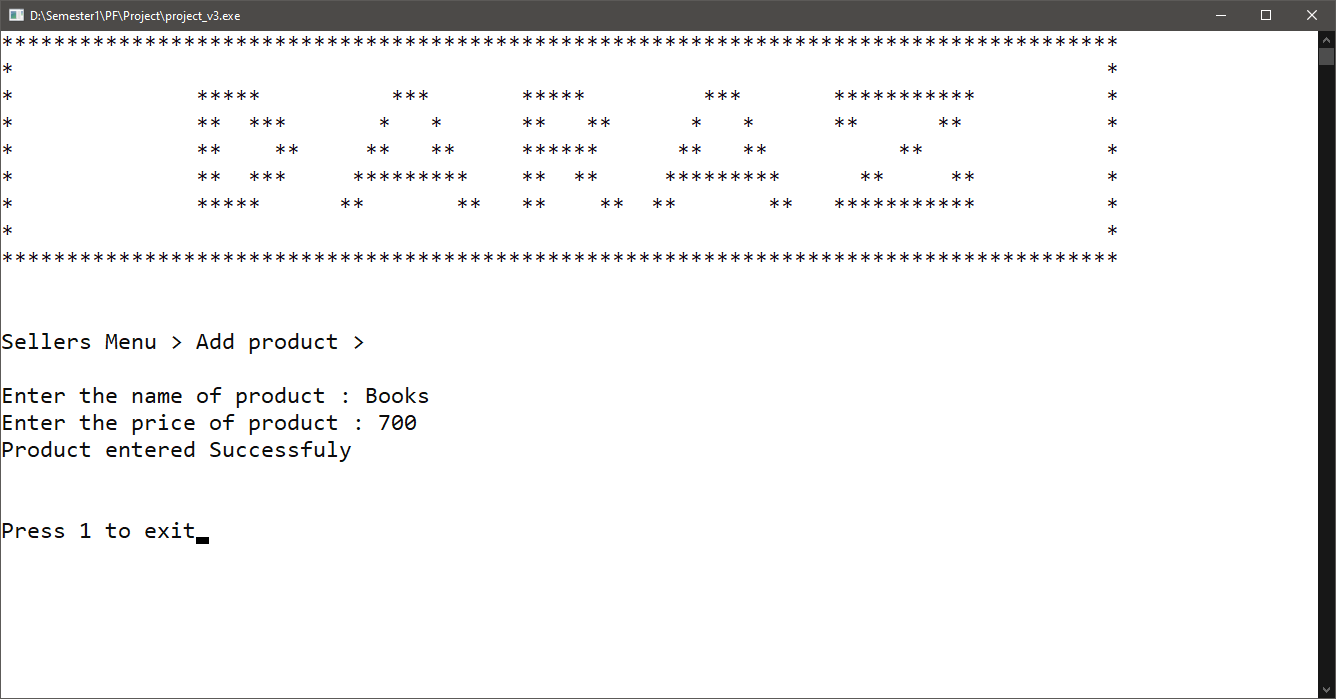
Login Menu:



Seller Menu:



Add Product:



View Products:



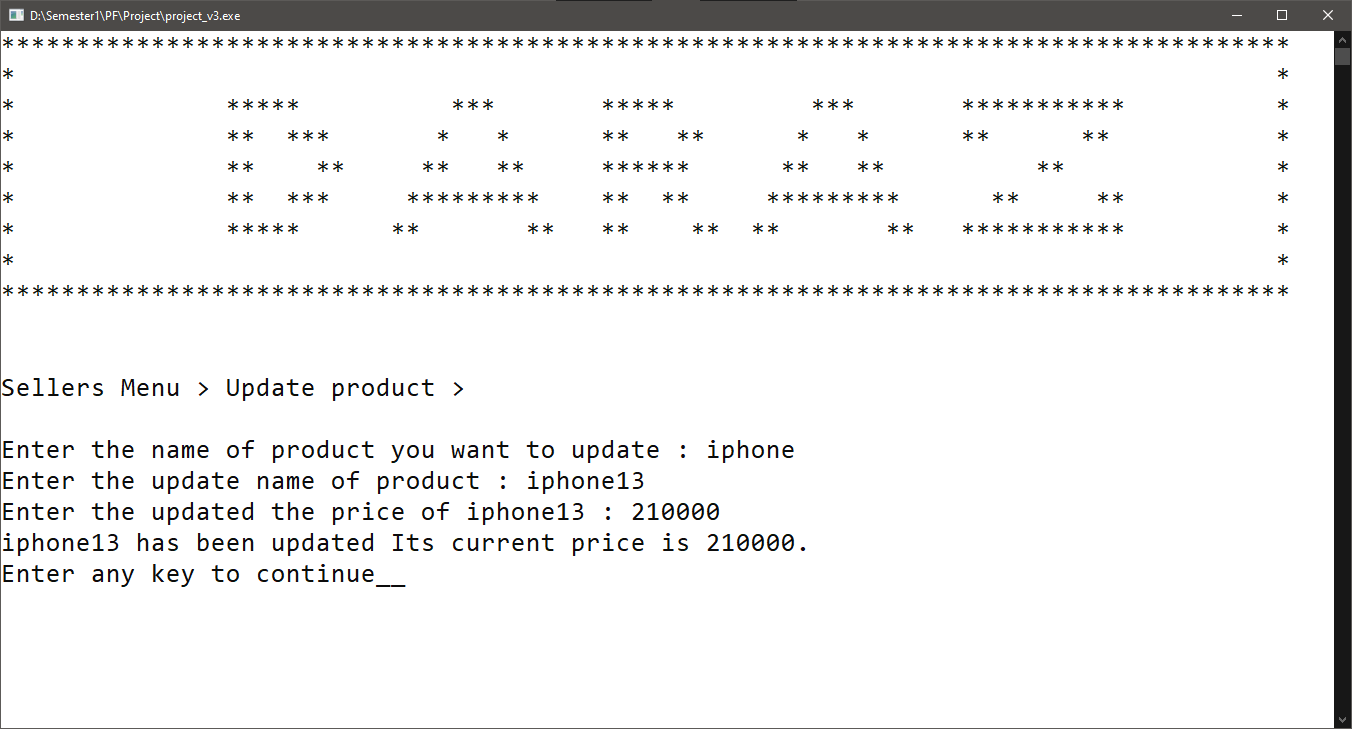
Sort Products (By price):



Delete Product:



Update Product:



**(Customer Menu)**

Login Menu:



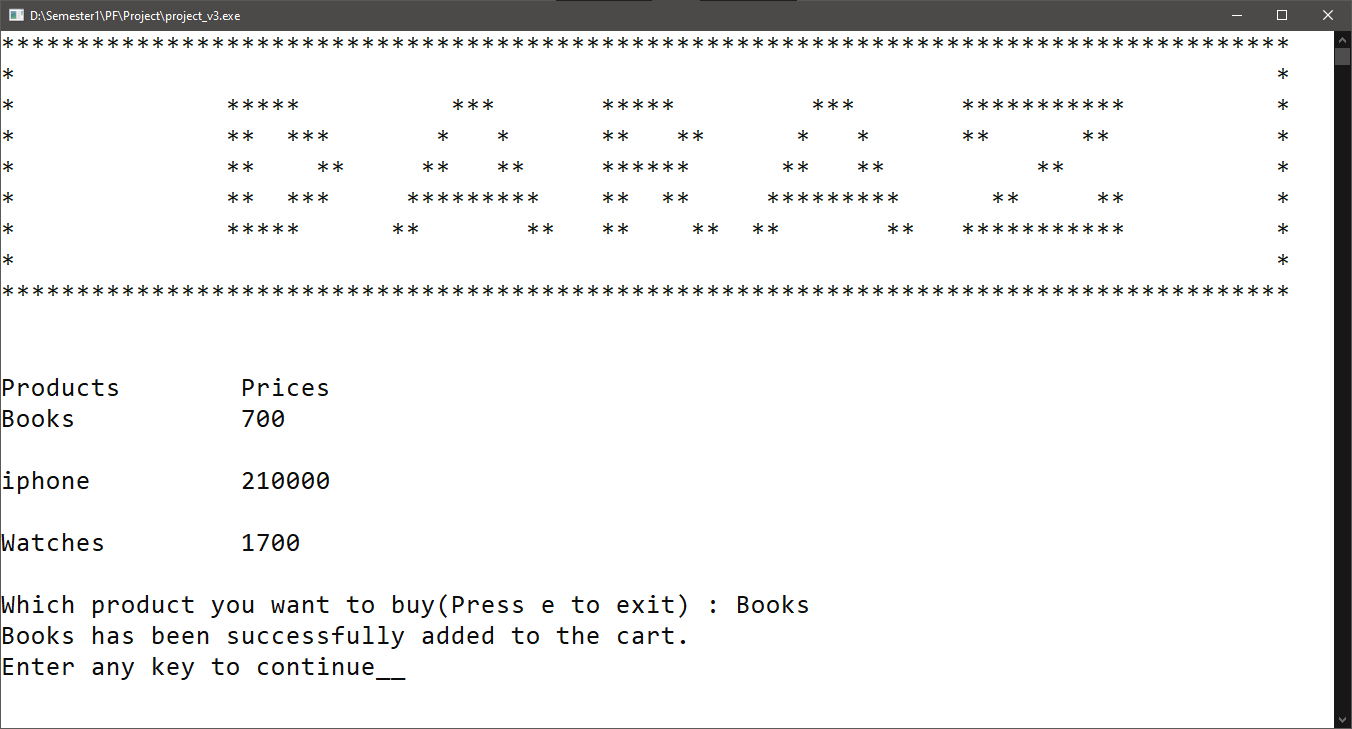
Create Account:



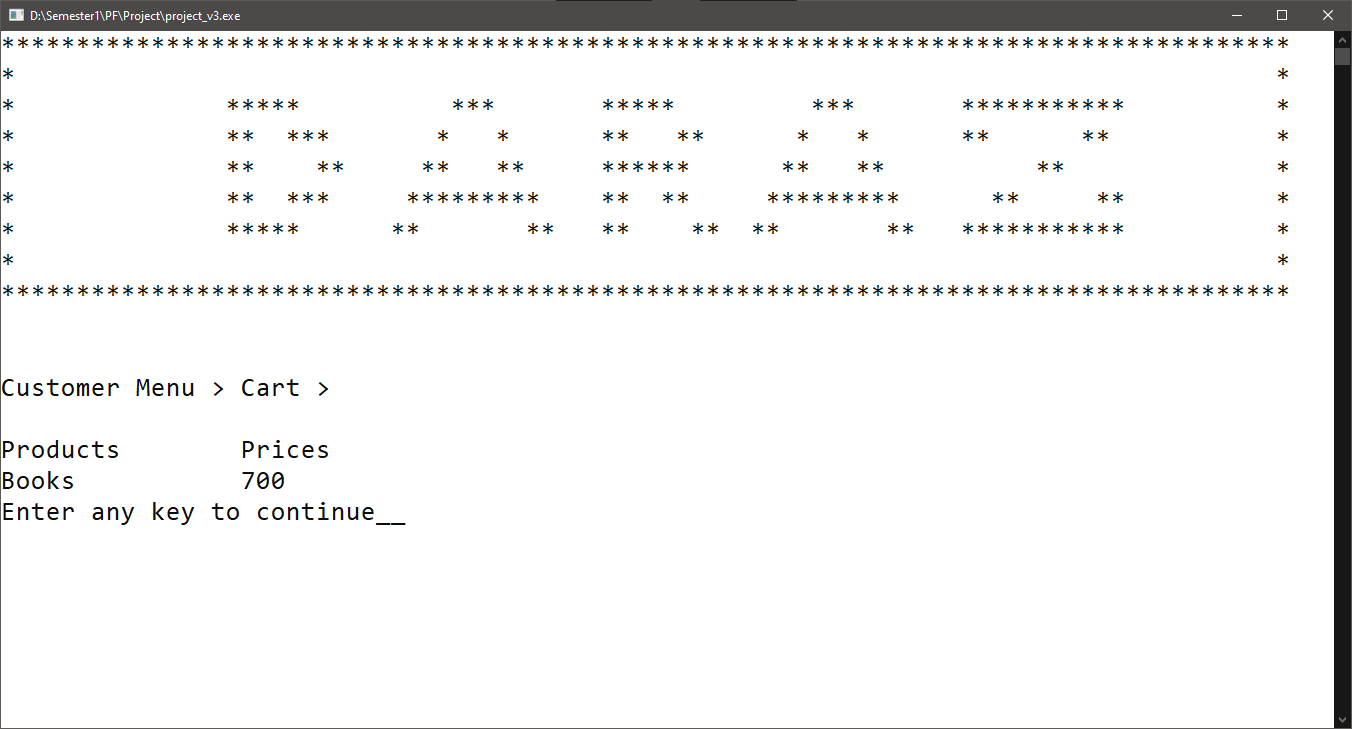
Customer Menu:



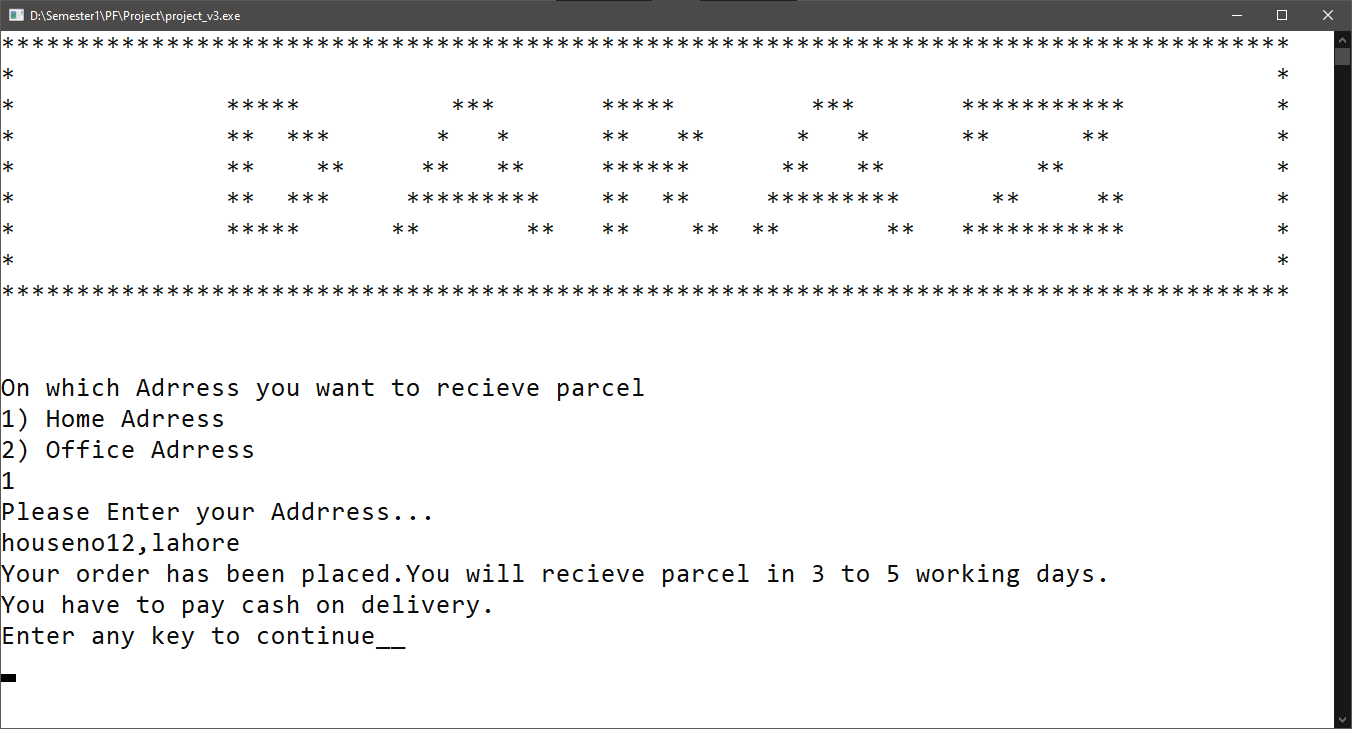
View Items:



View Cart:



Payment:



**Data Structures:**

Int admin\_pass=1234;

**const int SELLER\_COUNT=10;**

**const int PRODUCT\_COUNT=20;**

**const int CUST\_COUNT = 10;**

**const int USER\_COUNT = 5 ;**

**string seller\_name[SELLER\_COUNT];**

**int seller\_phone\_num[SELLER\_COUNT]={0};**

**string seller\_buisness[SELLER\_COUNT];**

**string seller\_password[SELLER\_COUNT];**

**string products[PRODUCT\_COUNT];**

**int prices[PRODUCT\_COUNT]={0};**

**string user\_nameA[CUST\_COUNT];**

**string user\_passwordA[CUST\_COUNT];**

**string buy\_productA[USER\_COUNT];**

**int buy\_product\_price[USER\_COUNT]={0};**

**Function Prototypes:**

* **Void header()**
* **Char main\_menu()**
* **String parse(string row,int coloumn)**
* **Void clear\_screen()**

**Admin**

* **Char admin\_menu() //Default password is 1234**
* **Void add\_seller\_to\_array(string name,int phone\_num,string business,string pass)**
* **Void add\_seller()**
* **Void load seller\_to\_array()**
* **Void show\_seller\_data()**
* **Void delete\_seller()**
* **Void update\_seller()**
* **Void load\_seller\_data\_to\_array()**

**Seller**

* **Bool check\_product()**
* **Bool check\_seller()**
* **Int check\_name()**
* **Int seller\_menu()**
* **Void add\_product\_to\_array(string product,int price)**
* **Void add\_product()**
* **Void Load\_products\_to\_array()**
* **Void show\_products()**
* **Void sorting()**
* **Void store\_seller\_data\_to\_file()**
* **Void store\_products\_to\_file()**

**Customer**

* **Char customer\_menu()**
* **Void create\_account\_customer()**
* **Void add\_cust\_to\_array()**
* **Void show\_products\_cust()**
* **Void store\_cust\_data()**
* **Void load\_cust\_data()**

**Function Work Flow:**

**Code:**

#include <iostream>

#include <stdlib.h>

#include <fstream>

#include <conio.h>

#include <string>

using namespace std;

////////////////////////////ARRAYS Declaration//////////////////////

char op;

int num\_seller = 0;

const int SELLER\_COUNT=10;

string seller\_name[SELLER\_COUNT];

string seller\_phone\_num[SELLER\_COUNT]={""};

string seller\_buisness[SELLER\_COUNT];

string seller\_password[SELLER\_COUNT];

int num\_products=0;

const int PRODUCT\_COUNT=20;

string products[PRODUCT\_COUNT];

int prices[PRODUCT\_COUNT]={0};

int num\_cust=0,num\_product1=0;

const int CUST\_COUNT = 10;

const int USER\_COUNT = 5 ;

string user\_nameA[CUST\_COUNT];

string user\_passwordA[CUST\_COUNT];

string buy\_productA[USER\_COUNT];

int buy\_product\_price[USER\_COUNT]={0};

/////////////////FUNCTIONS////////////////////////

void header()

{

system("cls");

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

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

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

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

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

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

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

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

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

cout << endl;

cout << endl;

}

char main\_menu()

{

cout << "////////////////////////////////////< Main Menu >/////////////////////////////////////" << endl;

cout << "Login as : " << endl;

cout << "1) Admin" << endl;

cout << "2) Seller" << endl;

cout << "3) User" << endl;

cout << endl;

cout << "Press 4 to Exit" << endl;

cin >> op;

return op;

}

string parse(string row, int column)

{

int dollarCount = 1;

string item;

int index = 0;

while (true)

{

char ch = row[index];

if (ch == '\0')

return item;

if (ch == '$')

{

dollarCount++;

}

else if (dollarCount == column)

{

item = item + ch;

}

index++;

}

return item;

}

char admin\_menu()

{

cout << "////////////////////////////////////< Admin Menu >/////////////////////////////////////" << endl<<endl;

cout << "Welcome Admin" << endl;

cout << "Today is a busy day We have to do with a lot of work today " << endl;

cout << " 1) Add seller" << endl;

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

cout << " 3) View History" << endl;

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

cout << " 5) Update Password" << endl;

cout << " 6) Exit" << endl;

cout << " " << endl;

cout << "Enter your option......" << endl;

cin >> op;

return op;

}

void add\_seller\_to\_array(string name,string ph,string buisness,string pass)

{

seller\_name[num\_seller]=name;

seller\_phone\_num[num\_seller]=ph;

seller\_buisness[num\_seller]=buisness;

seller\_password[num\_seller]=pass;

num\_seller = num\_seller+1;

}

void add\_seller()

{

for(int x = num\_seller; x < SELLER\_COUNT ; x++)

{

int exit=0;

header();

string name,buisness,pass,phone;

cout << "Admin Menu > Add Seller" << endl;

cout << "Enter seller's information ....." << endl<< endl;

cin.ignore();

cout << "Enters seller name : ";

getline(cin,name);

cout << "Enters seller phone number : " ;

getline(cin,phone);

cout << "Enters seller buisness : ";

getline(cin,buisness);

cout << "Create password : ";

getline(cin,pass);

cout<< endl;

add\_seller\_to\_array(name,phone,buisness,pass);

cout << "Sellers id is created " << endl;

cout<<endl;

cout<<"Press 1 to exit";

cin >> exit;

if(exit == 1 )

{

break;

}

}

}

bool check\_product()

{

bool yt=0;

for (int x = 0; x < num\_products; x++)

{

if (prices[x] > 0)

{

yt = 1;

}

}

return yt;

}

bool check\_seller()

{

bool yt=0;

for (int x = 0; x < num\_seller; x++)

{

if (seller\_phone\_num[x] != " ")

{

yt = 1;

}

}

return yt;

}

int check\_name(string buyp)

{

int yt = -1;

for (int x = 0; x < num\_products; x++)

{

if (products[x] == buyp)

{

yt = x;

}

}

return yt;

}

void load\_seller\_data\_to\_array()

{

fstream show;

show.open("seller.txt" , ios::in);

while(!show.eof())

{

string name,phone,buisness,pass,str;

getline(show,str);

name = parse(str,1);

phone =parse(str,2);

buisness = parse(str,3);

pass =parse(str,4);

if(pass != "") add\_seller\_to\_array(name,phone,buisness,pass);

}

show.close();

}

void show\_seller\_data()

{

for (int x = 0; x < num\_seller; x++)

{

if(seller\_phone\_num[x] != "")

{

cout.setf(ios::left);

cout.width(30);

cout << seller\_name[x];

cout.width(30);

cout<< seller\_phone\_num[x];

cout.width(30);

cout<< seller\_buisness[x];

cout.width(30);

cout<< seller\_password[x] << endl;

}

}

}

void clear\_screen()

{

cout << "Enter any key to continue....";

getch();

system("cls");

}

int seller\_menu()

{

cout << "////////////////////////////////////< Seller Menu >/////////////////////////////////////" << endl<<endl;

cout << " Welcome Seller" << endl;

cout << " Today we have to do some tasks" << endl;

cout << " 1) Add a product to your list " << endl;

cout << " 2) View items you sale " << endl;

cout << " 3) Sort by price " << endl;

cout << " 4) Delete an item " << endl;

cout << " 5) Update price of an item " << endl;

cout << " 6) Exit" << endl;

cout<< endl;

cout << "Enter your option......";

cin >> op;

return op;

}

void add\_product\_to\_array(string product,int price)

{

products[num\_products] = product;

prices[num\_products] = price;

num\_products++;

}

void add\_product()

{

int exit =0;

for(int x = num\_products ; x < PRODUCT\_COUNT ; x++)

{

system("cls");

header();

string product;

int price;

cout << "Sellers Menu > Add product >" << endl << endl;

cin.ignore();

cout << "Enter the name of product : ";

getline(cin,product);

cout << "Enter the price of product : ";

cin >> price;

add\_product\_to\_array(product,price);

cout << "Product entered Successfuly" << endl <<endl;

cout<<"Press 1 to exit";

cin >> exit;

if(exit == 1 )

{

system("cls");

break;

}

}

}

void load\_products\_to\_array()

{

fstream productf;

string str;

int countlines=0;

productf.open("product.txt" , ios::in);

while(!productf.eof())

{

string product, prc;

int price=0;

getline(productf,str);

if(str=="")

{

break;

}

product=parse(str,1);

prc= parse(str,2);

price=stoi(prc);

add\_product\_to\_array(product,price);

}

productf.close();

}

void show\_products()

{

bool yt = check\_product();

if (yt == 1)

{

cout.setf(ios::left);

cout.width(30);

cout << "Products";

cout.width(30);

cout<<"Prices" << endl;

for (int x = 0; x < num\_products; x++)

{

if (prices[x] > 0)

{

cout.setf(ios::left);

cout.width(30);

cout << products[x];

cout.width(30);

cout << prices[x] << endl;

}

}

}

else

{

cout << "No Products to Display." << endl;

}

clear\_screen();

}

void sorting()

{

bool yt = check\_product();

if (yt == 1)

{

for (int x = 0 ; x < num\_products; x++)

{

for (int m = x+1 ; m < num\_products; m++)

{

if(prices[m] > prices[x])

{

swap(products[x], products[m]);

swap(prices[x], prices[m]);

}

}

}

show\_products();

}

else

{

cout << "You have entered no products" << endl;

cout << "Go to option 1 to add some products" << endl;

clear\_screen();

}

}

void delete\_product()

{

string delete\_product;

int index=-1;

bool yt = check\_product();

if (yt == 1 )

{

cout << "Enter the name of product you want to delete : ";

cin >> delete\_product;

for (int x = 0; x < num\_products; x++)

{

if (delete\_product == products[x])

{

index = x;

}

}

if(index == -1)

{

cout<<"There is no such product."<<endl;

}

else

{

cout << products[index] << " has been deleted." << endl;

products[index] = " ";

prices[index] = 0;

}

}

else

{

cout << "You have not entered any record." << endl;

}

clear\_screen();

}

void update\_seller()

{

cout << "Sellers Menu > Update product >" << endl << endl;

bool yt = check\_product();

if (yt == 1)

{

string update\_product;

int index = -1;

bool yt = check\_product();

if (yt == 1)

{

cout << "Enter the name of product you want to update : ";

cin >> update\_product;

for (int x = 0; x < num\_products; x++)

{

if (update\_product == products[x])

{

index = x;

}

}

if (index == -1)

{

cout << "There is no such Product." << endl;

}

else

{

cout << "Enter the update name of product : ";

cin >> products[index];

cout << "Enter the updated the price of " << products[index] << " : ";

cin >> prices[index];

cout << products[index] << " has been updated Its current price is " << prices[index] << "." << endl;

}

}

else

{

cout << "You have not entered any record." << endl;

}

}

clear\_screen();

}

char customer\_menu()

{

cout << "////////////////////////////////////< Customer Menu >/////////////////////////////////////" << endl

<< endl;

cout << "1) View Items" << endl;

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

cout << "3) Payment" << endl;

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

cout << endl;

cout << "Enter your option...." << endl;

cin >> op;

return op;

}

void add\_cust\_to\_array(string name, string pass)

{

user\_nameA[num\_cust] = name;

user\_passwordA[num\_cust] = pass;

num\_cust++;

}

void create\_account\_customer()

{

string name, pass;

cin.ignore();

cout << "Enter your name : ";

getline(cin,name);

cout << "Create your Password : ";

getline(cin,pass);

cout << "Welcome " << user\_nameA[num\_cust] << endl;

add\_cust\_to\_array(name, pass);

cout << "Your account has been created Successfully." << endl;

}

void show\_products\_cust()

{

bool yt = check\_product();

if (yt == 1)

{

string buy\_products;

int count=1;

if (num\_product1 < 5)

{

while (num\_product1 < 5)

{

header();

cout.setf(ios::left);

cout.width(30);

cout << "Products";

cout.width(30);

cout<<"Prices" << endl;

for (int x = 0; x < num\_products; x++)

{

if (prices[x] != 0)

{

cout.setf(ios::left);

cout.width(30);

cout << products[x];

cout.width(30);

cout << prices[x] << endl;

}

}

cin.ignore();

cout << "Which product you want to buy(Press e to exit) : ";

getline(cin,buy\_products);

if(buy\_products == "e")

{

break;

}

int yt = check\_name(buy\_products);

if (yt != -1)

{

buy\_productA[num\_product1] = buy\_products;

buy\_product\_price[num\_product1] = prices[yt];

num\_product1++;

cout <<buy\_products<<" has been successfully added to the cart."<<endl;

clear\_screen();

break;

}

else

{

cout << "There is no such product." << endl;

cout << "Please enter again." << endl;

}

clear\_screen();

}

}

else

{

cout <<"You have reached your limit"<<endl;

clear\_screen();

}

}

else

{

cout << "No Products to Display." << endl;

clear\_screen();

}

}

void store\_seller\_data\_to\_file()

{

fstream str;

int x=0;

str.open("seller.txt", ios::out);

while(x < num\_seller)

{

if(seller\_phone\_num[x] != "")

{

str << endl;

str << seller\_name[x] << "$";

str << seller\_phone\_num[x] << "$";

str << seller\_buisness[x] << "$";

str << seller\_password[x];

}

x++;

}

str.close();

}

void store\_products\_to\_file()

{

fstream store;

int x=0;

store.open("product.txt",ios::out);

while(x < num\_products)

{

if(prices[x] > 0)

{

store << products[x] << "$";

store << prices[x];

store << endl;

}

x++;

}

store.close();

}

void load\_customer\_data()

{

fstream file;

string name,pass,str;

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

while(!file.eof())

{

getline(file,str);

name=parse(str,1);

pass=parse(str,2);

add\_cust\_to\_array(name , pass);

}

file.close();

}

void store\_cust\_data()

{

fstream file;

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

int x=0;

while(x < num\_cust)

{

if(user\_nameA[x] != "")

{

file << user\_nameA[x] << "$" << user\_passwordA[x] << endl;

}

x++;

}

file.close();

}

///////////////////////////////////////// End of Function /////////////////////////////////

///////////////////////////////////////// MAIN ///////////////////////////////////////

main()

{

load\_seller\_data\_to\_array();

load\_products\_to\_array();

load\_customer\_data();

string delete\_seller;

char main\_menu\_option;

string admin\_pass = "1234";

string pass;

while (main\_menu\_option != '4')

{

header();

main\_menu\_option = main\_menu();

if (main\_menu\_option == '1') ////////////////////// Admin Loop /////////////////////

{

int index = -1;

header();

cout << "Enter Pasword to continue to the Admin Menu........."<<endl<<endl;

cin.ignore();

cout << "Enter password : ";

getline(cin,pass);

system("cls");

if (pass == admin\_pass)

{

char admin\_option = '9';

string delete\_seller;

while (admin\_option != '6')

{

header();

admin\_option = admin\_menu();

header();

if (admin\_option == '1')

{

add\_seller();

}

else if (admin\_option == '2')

{

cout << "Admin Menu > View Seller > " << endl<< endl;

if (check\_seller())

{

cout.setf(ios::left);

cout.width(30);

cout << "Name";

cout.width(30);

cout<<"Phone Number";

cout.width(30);

cout<<"Buisness";

cout.width(30);

cout<<"Password"<< endl;

show\_seller\_data();

}

else

{

cout << "You have not entered any data yet...." << endl;

}

clear\_screen();

}

else if (admin\_option == '3')

{

cout << "Admin Menu > History > " << endl<< endl;

cout << "Ali just sold 2 bags to user179. " << endl;

cout << "Ahmad updated his stock. " << endl;

cout << "Ali updated his account." << endl;

for (int x = 0; x < num\_seller; x++)

{

if (seller\_phone\_num[x] != "")

{

cout << "You just added " << seller\_name[x] << " to " << seller\_buisness[x] << " Buisness list." << endl;

}

}

clear\_screen();

}

else if (admin\_option == '4')

{

cout << "Admin Menu > Delete Seller >" << endl<< endl;

cout << "Enter Password : " << endl;

cin >> pass;

if (pass == admin\_pass)

{

cin.ignore();

cout << "Enter the name of Seller you want to delete : ";

getline(cin,delete\_seller);

for(int x = 0 ; x <= num\_seller ; x++)

{

if(delete\_seller == seller\_name[x])

{

index = x ;

break;

}

}

if(index == -1 )

{

cout << "There is no Seller named as " << delete\_seller << endl;

}

else

{

cout << seller\_name[index] << " has been deleted from " << seller\_buisness[index] << " buisness list." << endl;

seller\_name[index] = "";

seller\_buisness[index] = "";

seller\_password[index] = "";

seller\_phone\_num[index] = "";

}

}

else

{

cout << "You have Entered the wrong Password" << endl;

}

clear\_screen();

}

else if (admin\_option == '5')

{

cout << "Admin Menu > Update Password >" << endl << endl;

cout << "Enter your Old Password : "<<endl;

cin >> pass;

if(pass == admin\_pass)

{

string confirm\_pass,pass1;

string cancel\_pass;

while (cancel\_pass != "c" )

{

header();

cout << "Admin Menu > Update Password >" << endl << endl;

cin.ignore();

cout << "Enter your new Password : ";

getline(cin,pass1);

cout << "Renter your new Password : " ;

getline(cin,confirm\_pass);

if (pass1 == confirm\_pass)

{

if (admin\_pass == pass1 || admin\_pass == confirm\_pass)

{

cout << "New Password can not be the old password." << endl;

cout << "Please enter again..........(Press C to cancel)" << endl;

cin >> cancel\_pass;

}

else

{

admin\_pass = pass1;

cout << "Your Password has been changed." << endl;

break;

}

}

else

{

cout << "Password does not match." << endl;

cout << "Please enter again..........(Press C to cancel)" << endl;

cin >> cancel\_pass;

}

}

}

else

{

cout << "You have Entered the wrong Password." << endl;

}

clear\_screen();

}

else if (admin\_option == '6')

{

cout << "Thanks for using our Program.." << endl;

cout << "We are happy to serve you" << endl;

clear\_screen();

}

else

{

cout<<"You have entered the wrong input"<<endl;

cout<<endl;

}

}

}

else

{

cout << "You have Entered the wrong Password" << endl;

clear\_screen();

}

}

else if(main\_menu\_option == '2') ////////////////// Seller Loop ///////////////////////

{

string pass="0",name;

int index = -1;

char seller\_option = '9';

header();

cin.ignore();

cout <<"Enter your name : ";

getline(cin,name);

cout<<"Enter your Password : ";

getline(cin,pass);

for(int x = 0 ; x < num\_seller; x++)

{

if(pass == seller\_password[x] && name == seller\_name[x])

{

index = x ;

}

}

if(index != -1)

{

cout << "Hello " << seller\_name[index]<<" ." << endl;

cout << "Hope you are fine." << endl;

cout << "Lets get back to work.............." << endl<<endl;

clear\_screen();

while (seller\_option != '6')

{

header();

seller\_option = seller\_menu();

header();

if (seller\_option == '1')

{

add\_product();

}

else if(seller\_option == '2')

{

cout << "Sellers Menu > View Products >" << endl << endl;

show\_products();

}

else if(seller\_option == '3')

{

cout << "Sellers Menu > Sorted product >" << endl << endl;

sorting();

}

else if(seller\_option == '4')

{

delete\_product();

}

else if(seller\_option == '5')

{

update\_seller();

}

else if(seller\_option == '6')

{

cout << "Thanks for using our Program.." << endl;

cout << "We are happy to serve you" << endl;

clear\_screen();

}

else

{

cout <<"You have entered the wrong input"<<endl;

}

}

}

else

{

cout << "You have entered the wrong Password" << endl;

clear\_screen();

}

}

else if(main\_menu\_option == '3')

{

char customer\_option ='9';

char exit=' ';

char option;

while (exit != 'e')

{

header();

string user\_name,user\_password;

cout << endl;

cout << "1) Create new account" << endl;

cout << "2) Login (If you already have an account)" << endl;

cout << " Press e to exit"<<endl <<endl;

cin >> exit;

if (exit == '1')

{

create\_account\_customer();

clear\_screen();

}

else if (exit == '2')

{

string user\_name, user\_password;

int index = -1;

cin.ignore();

cout << "Enter your name : ";

getline(cin,user\_name);

cout << "Enter your Password : ";

getline(cin,user\_password);

for ( int x = 0; x < num\_cust; x++ )

{

if (user\_name == user\_nameA[x] && user\_password == user\_passwordA[x])

{

index = x;

}

}

if (index != -1)

{

while(customer\_option != '4')

{

header();

customer\_option = customer\_menu();

header();

if(customer\_option == '1')

{

cout << "Customer Menu > View Products >" << endl<<endl;

show\_products\_cust();

}

else if (customer\_option == '2')

{

cout << "Customer Menu > Cart >" << endl<<endl;

bool yt = check\_product;

if (yt == 1)

{

cout.setf(ios::left);

cout.width(30);

cout << "Products";

cout.width(30);

cout<<"Prices" << endl;

for (int x = 0; x < num\_product1; x++)

{

cout.setf(ios::left);

cout.width(30);

cout << buy\_productA[x];

cout.width(30);

cout << buy\_product\_price[x] << endl;

}

}

else

{

cout << "No Products to Display." << endl;

}

clear\_screen();

}

else if (customer\_option == '3')

{

cout << "Customer Menu > Payment > " << endl<<endl;

int op1;

cout << "Select the payment method : " << endl;

cout << "1) Debit Card " << endl;

cout << "2) Cash On Dilivered " << endl;

cin >> op1;

if (op1 == 1)

{

int tr;

string credit\_card\_num;

while (true)

{

header();

cin.ignore();

cout << "Please enter your credit card number : ";

getline(cin,credit\_card\_num);

cout << "Please enter correct Card number ";

cout << "Press (1) to exit" << endl;

cin >> tr;

if (tr == 1)

{

break;

}

}

if (tr == 1)

{

break;

}

}

else if (op1 == 2)

{

header();

string location;

string addrress;

cout << "On which Adrress you want to recieve parcel" << endl;

cout << "1) Home Adrress" << endl;

cout << "2) Office Adrress" << endl;

cin.ignore();

getline(cin,location);

cout << "Please Enter your Addrress..." << endl;

cin.ignore();

getline(cin,addrress);

cout << "Your order has been placed.You will recieve parcel in 3 to 5 working days." << endl;

cout << "You have to pay cash on delivery." << endl;

}

else

{

cout <<"You have entered the wrong input"<<endl;

}

clear\_screen();

}

else if (customer\_option == '4')

{

cout << "Thanks for using our Program.." << endl;

cout << "We are happy to serve you" << endl;

exit='e';

clear\_screen();

}

else

{

cout << "You have entered the wrong input" << endl;

}

}

}

else

{

cout << "Sorry it seems like you do not have any account..." << endl;

cout << "Try to connect later...." << endl;

}

clear\_screen();

}

else if(exit == 'e'){}

else

{

header();

cout <<"You have entered the Wrong input."<<endl;

clear\_screen();

}

system("cls");

}

}

else if(main\_menu\_option == '4')

{

store\_seller\_data\_to\_file();

store\_products\_to\_file();

store\_cust\_data();

cout << "Thanks for using our Program.." << endl;

cout << "We are happy to serve you" << endl;

clear\_screen();

}

else

{

header();

cout <<"You have entered the Wrong input."<<endl;

}

}

}

**Test Cases:**

Same as above wireframes.

**Student Reg. No. :**  2021-CS-190  **Student Name.**  Abdul Mateen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 are 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 consistent and according to given **guidelines**. The project **Poster** is professionally designed and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets 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 you **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | The project has at least 2 user 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 are followed | All code style criteria followed but some improvements required | A lot of improvements are 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 are synchronized. | Code and documentation do not synchronize at **some** places | Code and documentation do not synchronize in **many** places | Code and documentation **do not** synchronize. |
| Data Structure (Arrays)  **Grade:** | The data structure is sufficient for the project requirements | Data Structure is sufficient but requires improvement to meet project requirements. | The data structure is not sufficient and needs a lot of improvement | Data Structure is not properly identified and declared. |
| Sorting Features  **Grade:** | Sort working 100% and generating a useful report | The sorting feature is working but sorted data is not useful for the project. | The sorting feature is partial implemented | The project does 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 a lot of places | No Validations are used |
| Recommendation Feature | The proper meaning full recommendation is present in the 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 | The presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | The student has a complete understanding of how the code is working and knows the concept. | The student has good understanding but in some places, he does not know the concepts | The student has very little understanding and lacks the major concepts. | The student does not have any level of understanding of the code. |

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

**Student Reg. No. :**  2021-CS-190  **Student Name.**  Abdul Mateen

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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 are 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 consistent and according to given **guidelines**. The project **Poster** is professionally designed and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meets more than 80% of the criteria given. | Documentation meets more than 50% of the criteria. | When the documentation meets 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 you **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | The project has at least 2 user 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 are followed | All code style criteria followed but some improvements required | A lot of improvements are 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 are synchronized. | Code and documentation do not synchronize at **some** places | Code and documentation do not synchronize in **many** places | Code and documentation **do not** synchronize. |
| Data Structure (Arrays)  **Grade:** | The data structure is sufficient for the project requirements | Data Structure is sufficient but requires improvement to meet project requirements. | The data structure is not sufficient and needs a lot of improvement | Data Structure is not properly identified and declared. |
| Sorting Features  **Grade:** | Sort working 100% and generating a useful report | The sorting feature is working but sorted data is not useful for the project. | The sorting feature is partial implemented | The project does 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 a lot of places | No Validations are used |
| Recommendation Feature | The proper meaning full recommendation is present in the 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 | The presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | The student has a complete understanding of how the code is working and knows the concept. | The student has good understanding but in some places, he does not know the concepts | The student has very little understanding and lacks the major concepts. | The student does not have any level of understanding of the code. |

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