**Super Market Management System**



Session: 2021 – 2025

**Submitted by:**

Muhammad Umair Shahid 2021-CS-144

**Supervised by:**

Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Description:**

My goal is to develop a business application that provides ease to the Shopkeepers and Super Market Employees and Customers. Following are some of the things that can help Admin, Employees & Customers to access easily with each other.

* Admin can easily check how much income is Produced & all other general things.
* Employees can add, update, delete and print bills. Customers can view product and see offered discount,
* Admin adds new employee so that there is no complexity left.
* Admin can easily calculate the gross income and net income & can keep its record for future needs, also admin can display some useful notifications to the Employees like about offer discount and Employees can also get that notice so that they can implement discount.

There are some major problems that the normal Shop keepers and Super market managers faces during managing the market tasks. This program will help them overcoming such problems.

I am developing a program that will help both user, Employee & admin to manage the common problems easily, like admin can give notice of important announcements.

**Users:**

* Admin
* Employee
* Customers

**Functional Requirements:**

Follow are some functional Requirements fulfilled by this system. Admin, Employees and Customers can login into the management system with unique usernames & passwords.

**Admin**

* “1. View Stock”
* “2. Add new Employees”
* “3. Gross Income”
* “4. Transfer Salaries”
* “5. Pay Bills
* “6. Total Profit”
* “7. History of Sales”
* “8. Offer Discount”
* “9. Leave Message
* “0. Return to Starting Menu”

**Employees**

* “1. Add Stock”
* “2. Update Stock”
* “3. Delete Stock”
* “4. View Stock”
* “5. Sale Products”
* “6. Check Admin Message”
* “7. View and Maintain Stock”
* “0. Return to Main Menu”

**Customers**

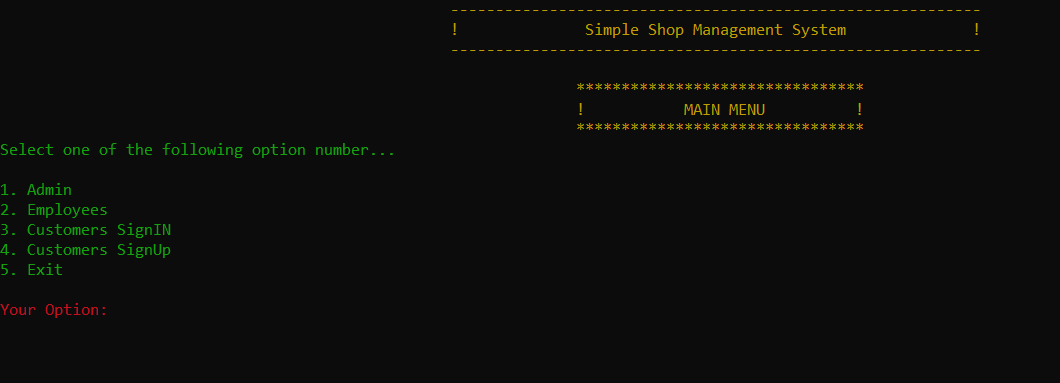
* “1. View Products”
* “2. View Discounted Products”
* “3. Return to Main Menu”

**Wireframes & Explanation**

* **Login**

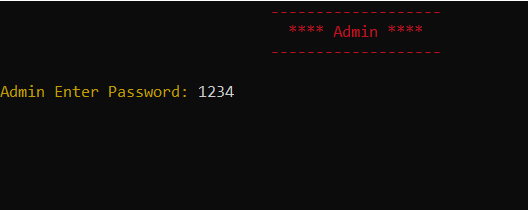
As you start the program the login page will appear, the Admin, Employee, Customers, Customer Sign Up and Exit, Admin can login the program using the unique Password. Employees can login through their unique usernames & passwords assigned to them by Admin. Customers can Sign Up and then can Sigh In.

* **Main Login Screen**



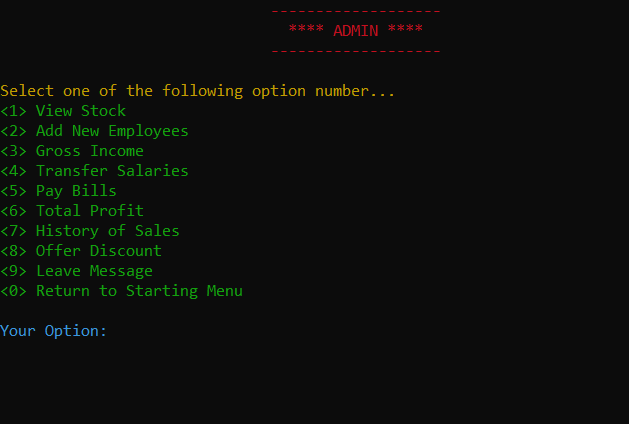
**Admin’s Login**

As the option 1 is selected from the main login the below screen pops up,



**Admin’s Main Menu**

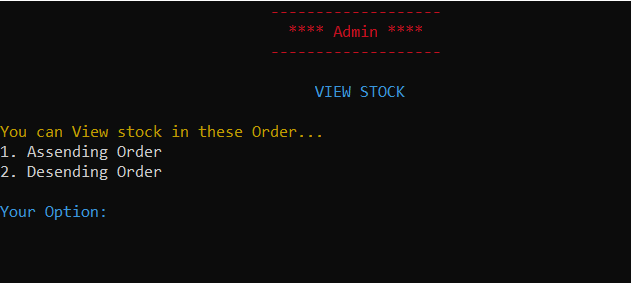
When the correct password is entered, the admins main menu opens,



**Option 1**

Admin can view stock in the following options, added by the Employees.

* Assending Order
* Desending Order

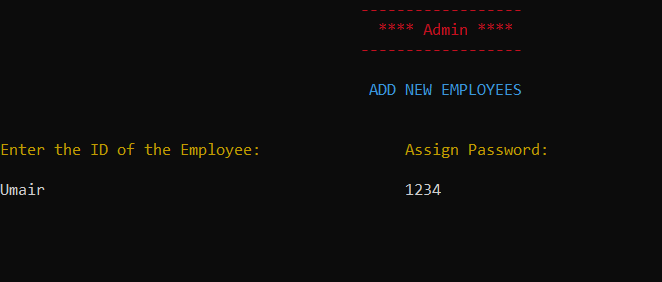






**Option 2**

Admin Can add Employees.



**Option 3**

Admin can view the gross Income generated.

**Option 4**

Admin can transfer the salaries of the Employees.

**Option 5**

Admin can view the total bills like Electricity, Wifi.

**Option 6**

Admin can view the Total Profit income generated after transferring Salaries, buying Stock & paying Bills.

**Option 7**

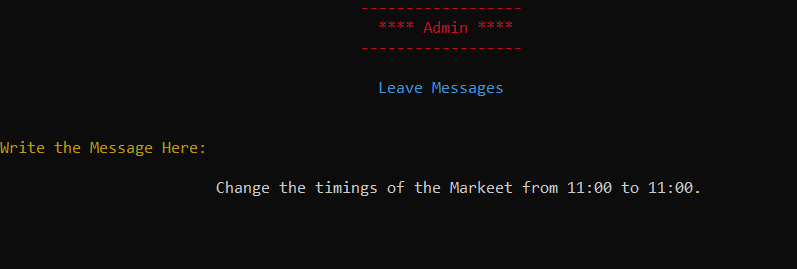
Admin can view the History of Sales.

**Option 8**

Admin can offer discount.

**Option 9**

Admin can leave message for the Employees.

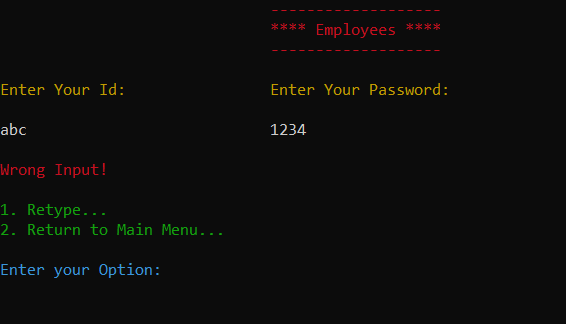


**Option 0**

Admin can return to Main Menu or Log Out.

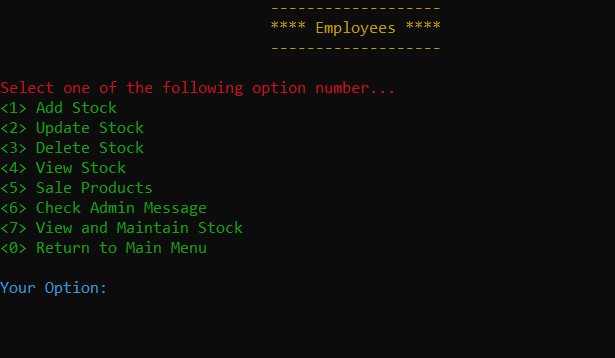
**Employees Login**

As the option No 2 is selected from the main Menu option, the Employees login menu will pop up. In case of wrong Input.



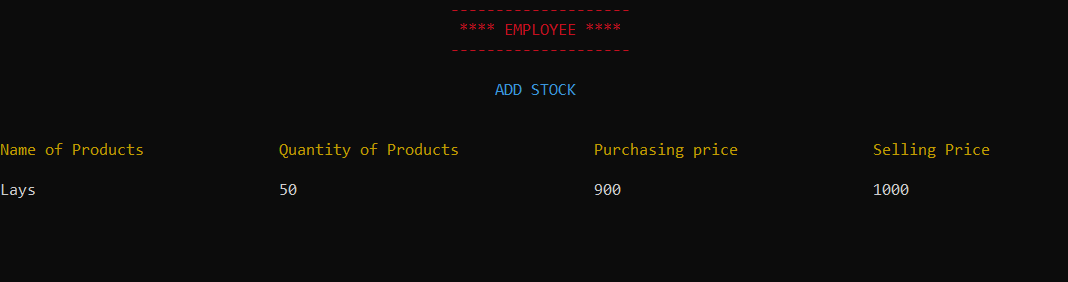
**Employee Main Menu**

After entering the correct id and password credentials, the main menu for Employee comes up.



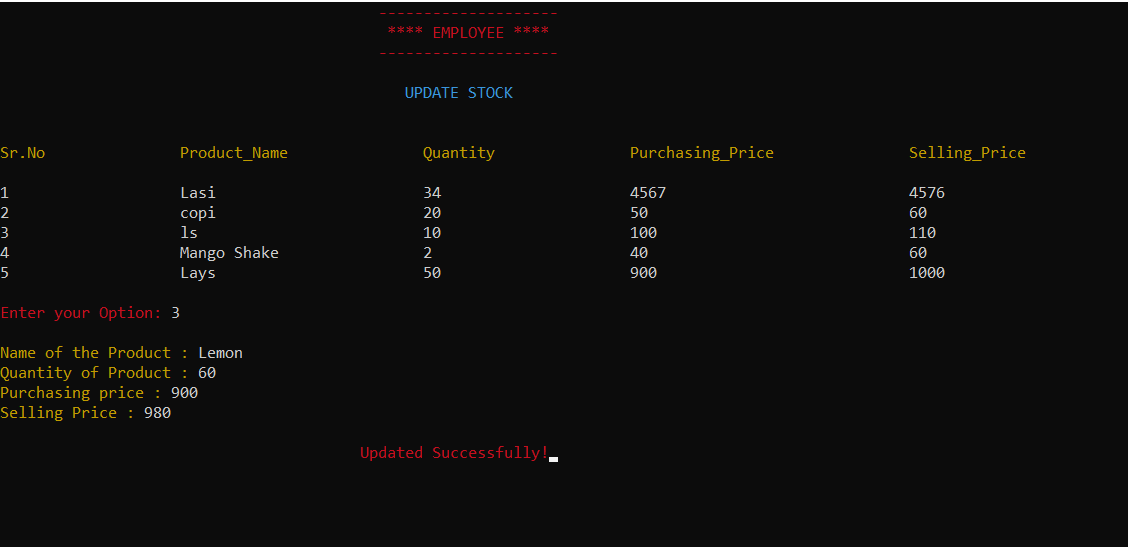
**Option 1**

The Employee can add the Products.

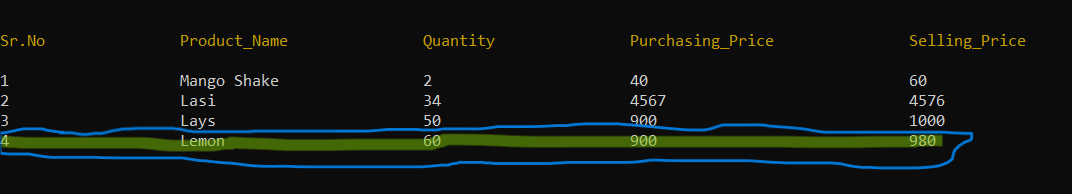


**Option 2**

The Employee can update the Products.



Updated Product is

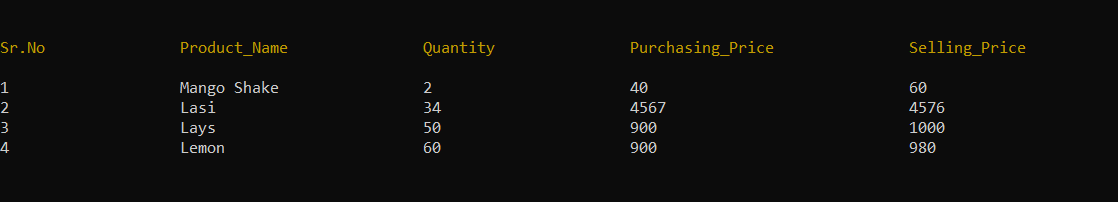


**Option 3**

The student can delete the Products.



After Deleting the Product:



**Option 4**

The Employee can view the Products same as in the Admin.

**Option 5**

The Employee can sale the Products and Print Bill.

**Option 6**

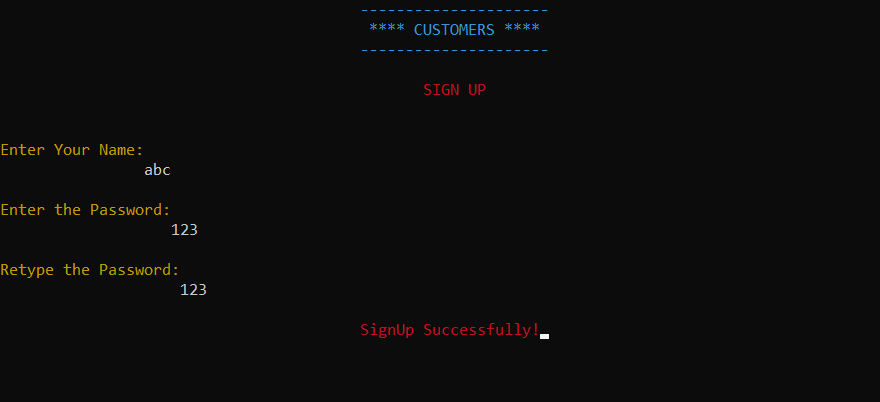
The Employee can check the message forwarded by the Admin.

**Option 7**

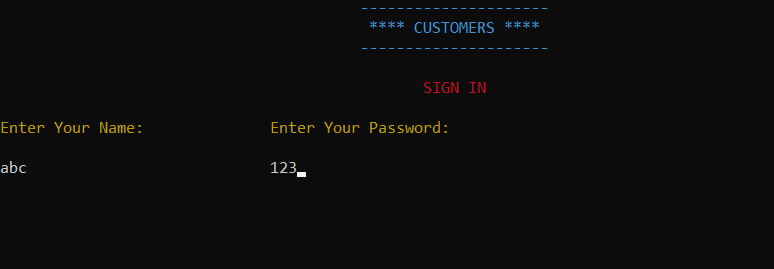
The Employee can maintain the stock like if the quantity of items are less than specific value than add this into the buying cart and order it.

**Customer Sign Up**

As the option No 4 is selected from the main Login Menu option, the Customer Sign Up menu will pop up.

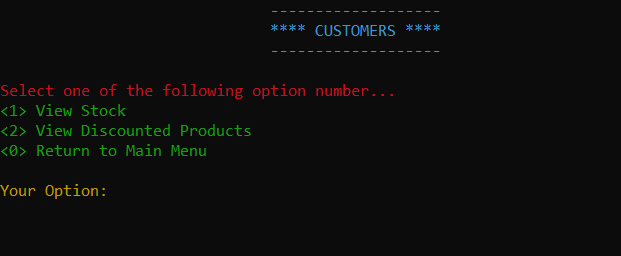


**Customer Sign In**



**Customer Main Menu**

After entering the correct name and password credentials, the main menu for Customers comes up.



**Option 1**

The Customer can view products and add Products to cart.

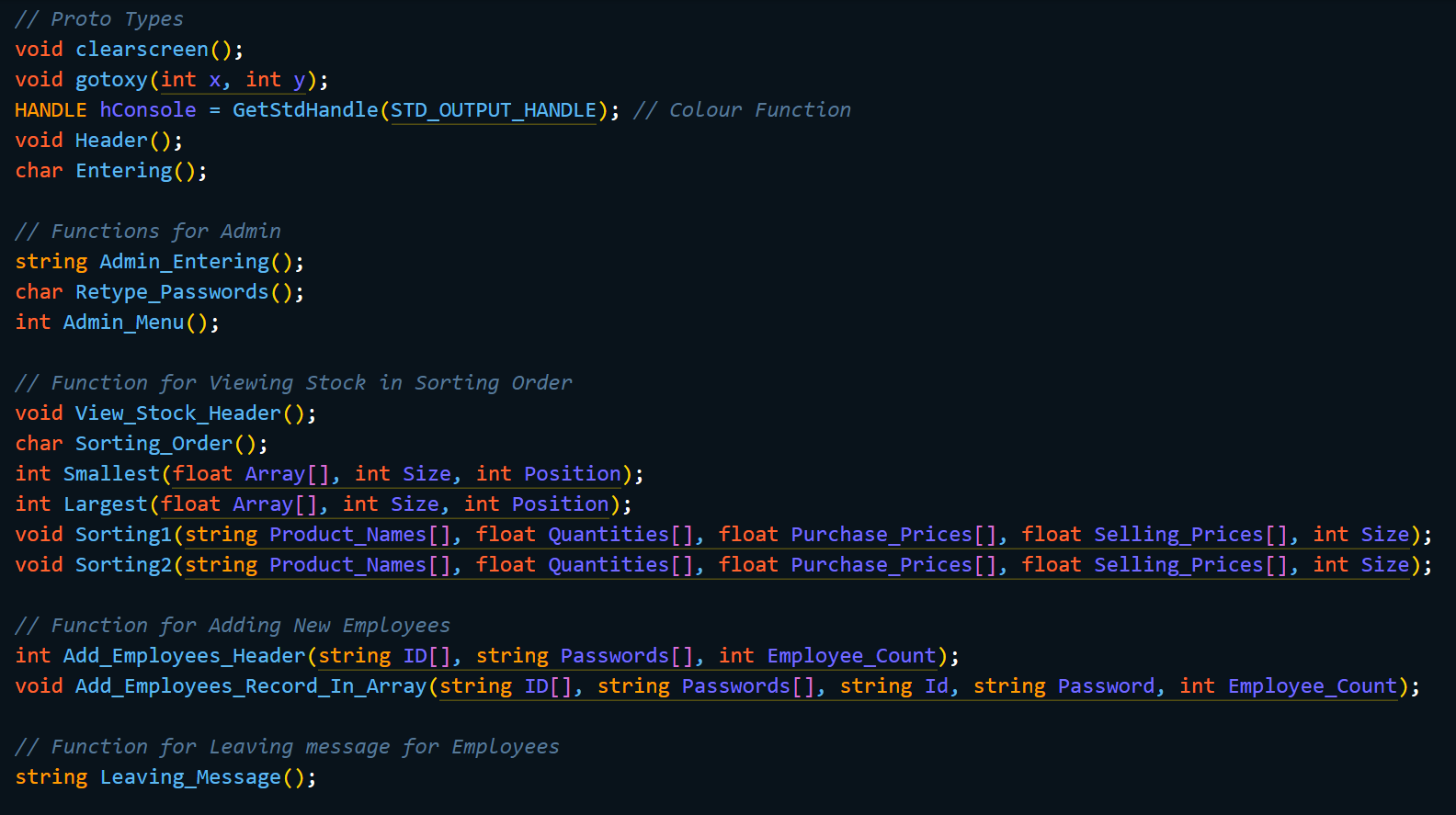
**Option 2**

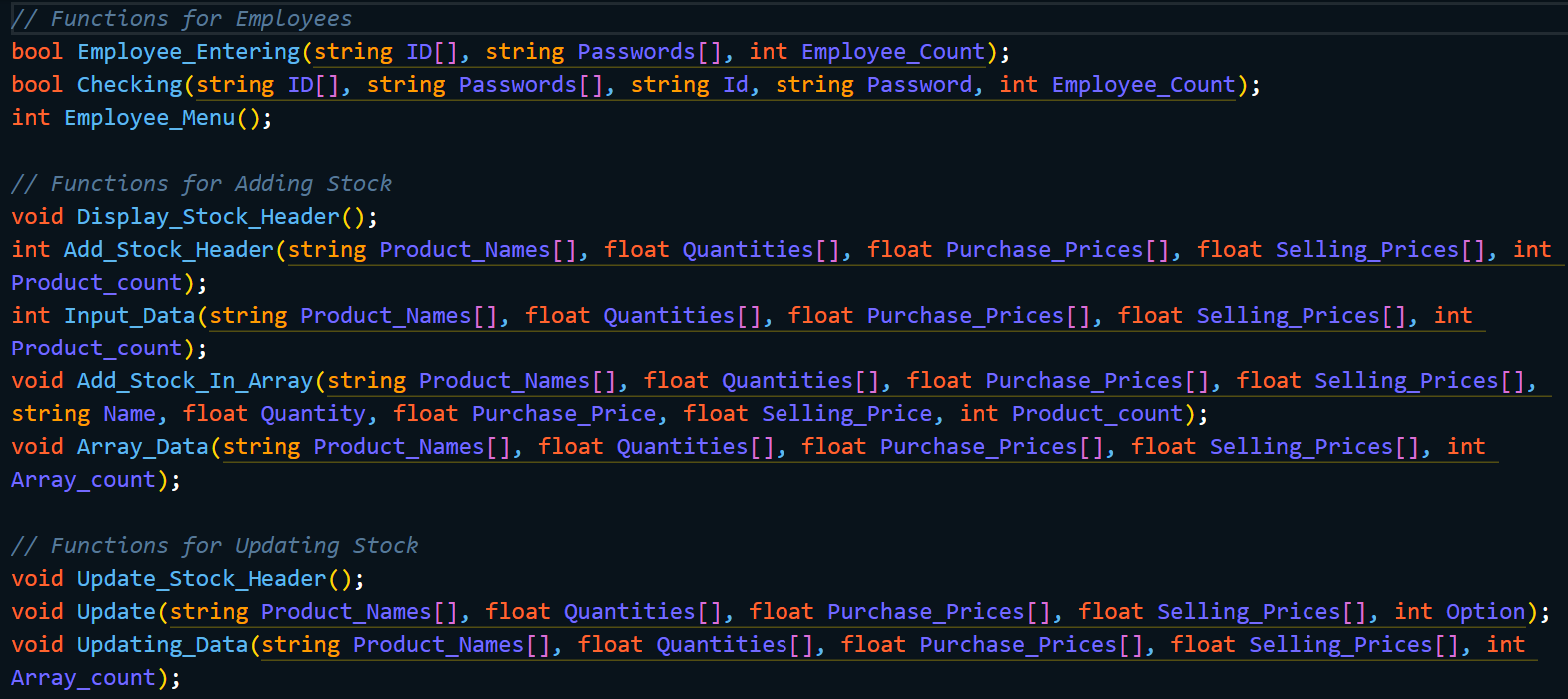
The Customer can view the discounted Products.

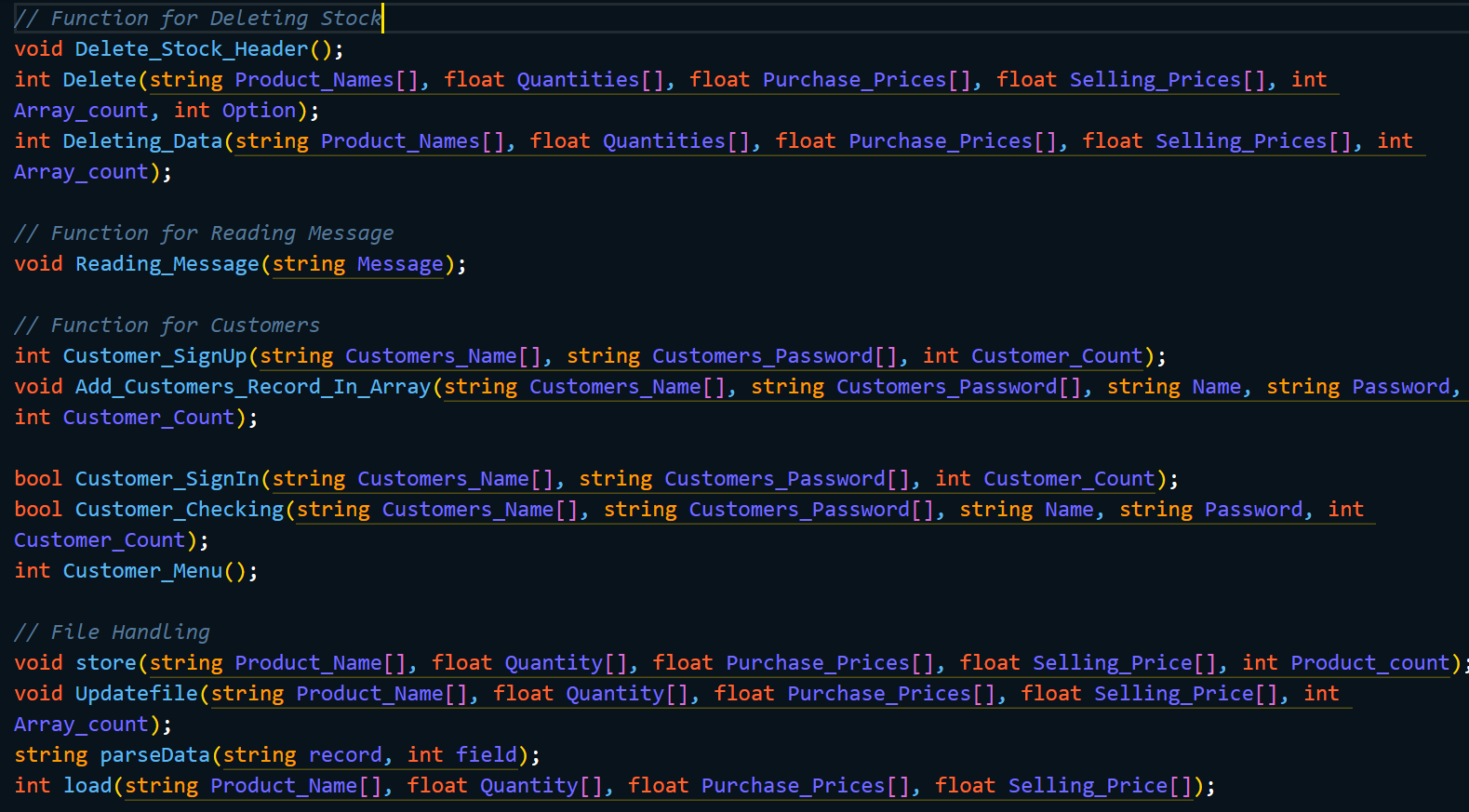
**Option 0**

The Customers can return to Main Menu.

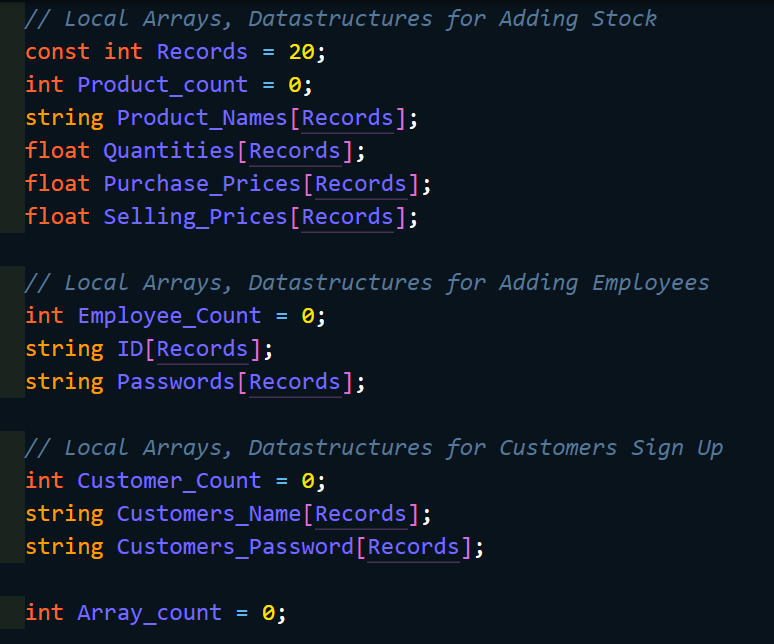
**Functions Prototypes**







**Data Structures**



**Working Flow**

**Main ( )**

**Admin**

**Customers**

**Employees**

**6. Total Profit**

**5. Pay Bills**

**4. Transfer Salaries**

**3. Gross Income**

**1. View Products**

**2. Add New Employees**

1. **Return to Main Menu**

**7**. **History of Sales**

**8. Offer Discount**

**9. Leave Message**

1. **Add Product**

**3. Delete Product**

**2. Update Product**

**5. Sale Products**

**4. View Product**

**0. Return to Main Menu**

**7. Main Stock**

**6. Check Admin Message**

**3. Return to Main Menu**

**2. Discounted Products**

**1.View Products**

**Complete Code:**

#include <iostream>  *//input,output*

#include <conio.h>   *//clearscreen*

#include <stdlib.h>  *//getch*

#include <windows.h> *//gotoxy*

#include <fstream>   *//filehandling*

using namespace std;

*// Proto Types*

void clearscreen();

void gotoxy(int x, int y);

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE); *// Colour Function*

void Header();

char Entering();

*// Functions for Admin*

string Admin\_Entering();

char Retype\_Passwords();

int Admin\_Menu();

*// Function for Viewing Stock in Sorting Order*

void View\_Stock\_Header();

char Sorting\_Order();

int Smallest(float Array[], int Size, int Position);

int Largest(float Array[], int Size, int Position);

void Sorting1(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Size);

void Sorting2(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Size);

*// Function for Adding New Employees*

int Add\_Employees\_Header(string ID[], string Passwords[], int Employee\_Count);

void Add\_Employees\_Record\_In\_Array(string ID[], string Passwords[], string Id, string Password, int Employee\_Count);

*// Function for Leaving message for Employees*

string Leaving\_Message();

*// Functions for Employees*

bool Employee\_Entering(string ID[], string Passwords[], int Employee\_Count);

bool Checking(string ID[], string Passwords[], string Id, string Password, int Employee\_Count);

int Employee\_Menu();

*// Functions for Adding Stock*

void Display\_Stock\_Header();

int Add\_Stock\_Header(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Product\_count);

int Input\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Product\_count);

void Add\_Stock\_In\_Array(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], string Name, float Quantity, float Purchase\_Price, float Selling\_Price, int Product\_count);

void Array\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count);

*// Functions for Updating Stock*

void Update\_Stock\_Header();

void Update(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Option);

void Updating\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count);

*// Function for Deleting Stock*

void Delete\_Stock\_Header();

int Delete(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count, int Option);

int Deleting\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count);

*// Function for Reading Message*

void Reading\_Message(string Message);

*// Function for Customers*

int Customer\_SignUp(string Customers\_Name[], string Customers\_Password[], int Customer\_Count);

void Add\_Customers\_Record\_In\_Array(string Customers\_Name[], string Customers\_Password[], string Name, string Password, int Customer\_Count);

bool Customer\_SignIn(string Customers\_Name[], string Customers\_Password[], int Customer\_Count);

bool Customer\_Checking(string Customers\_Name[], string Customers\_Password[], string Name, string Password, int Customer\_Count);

int Customer\_Menu();

*// File Handling*

void store(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[], int Product\_count);

void Updatefile(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[], int Array\_count);

string parseData(string record, int field);

int load(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[]);

main()

{

*// Local Arrays, Datastructures for Adding Stock*

  const int Records = 20;

  int Product\_count = 0;

  string Product\_Names[Records];

  float Quantities[Records];

  float Purchase\_Prices[Records];

  float Selling\_Prices[Records];

*// Local Arrays, Datastructures for Adding Employees*

  int Employee\_Count = 0;

  string ID[Records];

  string Passwords[Records];

*// Local Arrays, Datastructures for Customers Sign Up*

  int Customer\_Count = 0;

  string Customers\_Name[Records];

  string Customers\_Password[Records];

  int Array\_count = 0;

  system("CLS");

  while (true) *// Main While loop If the Entering Option is wrong than it will continue to run , Like in Admin Enetring option*

  {

    char Main\_Option = Entering(); *// Checking that who is entering*

    if (Main\_Option == '1') *// At 1 there is Admin*

    {

      while (true) *// Incase if user want to return to main menu then this while loop will break*

      {

        system("CLS");

        string Password = Admin\_Entering();

        if (Password == "1234") *// if Password match than do this*

        {

          system("CLS");

          while (true)

          {

            int Option = 1;

            while (Option != 0)

            {

              Option = Admin\_Menu(); *// Here admin select what to do*

              system("CLS");

              if (Option == 1) *// View Stock*

              {

                char Option = '0';

                Option = Sorting\_Order(); *// Ask in which do you want to see the sorting with quantity based*

                system("CLS");

                View\_Stock\_Header();

                Array\_count = load(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices);

                if (Option == '1') *// Assending*

                {

                  Sorting1(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                else if (Option == '2') *// Decesending*

                {

                  Sorting2(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                Updatefile(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                system("CLS");

              }

              else if (Option == 2) *// Add New Employees*

              {

                Employee\_Count = Add\_Employees\_Header(ID, Passwords, Employee\_Count);

              }

              else if (Option == 3) *// Gross Income*

              {

                cout << "Gross Income" << endl;

              }

              else if (Option == 4) *// Transfer Salaries*

              {

                cout << "Transfer Salaries" << endl;

              }

              else if (Option == 5) *// Pay Bills*

              {

                cout << "Pay Bills";

              }

              else if (Option == 6) *// Total Profit*

              {

                cout << "Total Profit";

              }

              else if (Option == 7) *// History of Sales*

              {

                cout << "History of Sales";

              }

              else if (Option == 8) *// Offer Discount*

              {

                cout << "Offer Discount";

              }

              else if (Option == 9) *// Leave Message*

              {

                string Message = Leaving\_Message();

              }

              else if (Option != 0) *// Incase of wrong option in Admin Menu*

              {

                SetConsoleTextAttribute(hConsole, 4); *// Red*

                gotoxy(30, 0);

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

                gotoxy(30, 1);

                cout << "  \*\*\*\* Admin \*\*\*\*  ";

                gotoxy(30, 2);

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

                SetConsoleTextAttribute(hConsole, 7); *// White*

                gotoxy(0, 4);

                cout << "You Enter a Wrong Number! ";

                getch();

                system("CLS");

              }

            }

            break; *// if Option ==  0*

          }

          break; *// if Option ==  0,Return to Main Menu*

        } *// if Password is correct*

        else if (Password != "1234") *// if Admin password is incorrect then*

        {

          char Option = Retype\_Passwords();

          if (Option == '1')

          {

            SetConsoleTextAttribute(hConsole, 8); *// Gray*

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

            getch();

          }

          else if (Option == '2')

          {

            system("CLS");

            break;

          }

        }

      }

    }

    else if (Main\_Option == '2') *// At 2 there are Employees*

    {

      while (true) *// Incase if user want to return to main menu then this while loop will break*

      {

        system("CLS");

        bool check = true;

        check = Employee\_Entering(ID, Passwords, Employee\_Count);

        if (check == true) *// if Id and Passwords are correct then do this*

        {

          system("CLS");

          while (true)

          {

            int Option = 1;

            while (Option != 0)

            {

              Option = Employee\_Menu(); *// Here Employee select what to do*

              system("CLS");

              if (Option == 1) *// Add\_Product*

              {

                Product\_count = Add\_Stock\_Header(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Product\_count);

                store(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Product\_count);

                if (Product\_count > Records) *// if Products which is added is greater than Records than this*

                {

                  cout << "No more Space! " << endl;

                  system("CLS");

                }

              }

              else if (Option == 2) *// Update Stock*

              {

                Update\_Stock\_Header();

                Array\_count = load(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices);

                Updating\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                Updatefile(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

              }

              else if (Option == 3) *// Delete Stock*

              {

                Delete\_Stock\_Header();

                Array\_count = load(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices);

                Array\_count = Deleting\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                Updatefile(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

              }

              else if (Option == 4) *// View Stock*

              {

                char Option = '0';

                Option = Sorting\_Order(); *// Ask in which do you want to see the sorting with quantity based*

                system("CLS");

                View\_Stock\_Header();

                Array\_count = load(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices);

                if (Option == '1') *// Assending*

                {

                  Sorting1(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                else if (Option == '2') *// Decesending*

                {

                  Sorting2(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                Updatefile(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                system("CLS");

              }

              else if (Option == 6) *// Reading Message*

              {

                string Message = Leaving\_Message();

                Reading\_Message(Message);

              }

              else if (Option != 0) *// Incase of wrong option in Employee Menu*

              {

                SetConsoleTextAttribute(hConsole, 4); *// Red*

                gotoxy(30, 0);

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

                gotoxy(30, 1);

                cout << "\*\*\*\* Employees \*\*\*\*";

                gotoxy(30, 2);

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

                SetConsoleTextAttribute(hConsole, 7); *// White*

                gotoxy(0, 4);

                cout << "You Enter a Wrong Number! ";

                getch();

                system("CLS");

              }

            }

            break; *// if Option ==  0*

          }

          break; *// if Option ==  0,Return to Main Menu*

        }

        else if (check != true) *// if Id and Passwords are incorrect then do this*

        {

          char Option = Retype\_Passwords();

          if (Option == '1')

          {

            SetConsoleTextAttribute(hConsole, 8); *// Gray*

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

            getch();

          }

          else if (Option == '2')

          {

            system("CLS");

            break;

          }

        }

      }

    }

    else if (Main\_Option == '3') *// At 3 there is Customer*

    {

      while (true) *// Incase if user want to return to main menu then this while loop will break*

      {

        system("CLS");

        bool check = true;

        check = Customer\_SignIn(Customers\_Name, Customers\_Password, Customer\_Count);

        if (check == true) *// if Id and Passwords are correct then do this*

        {

          system("CLS");

          while (true)

          {

            int Option = 1;

            while (Option != 0)

            {

              Option = Customer\_Menu(); *// Here Customers can select what to do*

              system("CLS");

              if (Option == 1) *// View Stock*

              {

                char Option = '0';

                Option = Sorting\_Order(); *// Ask in which do you want to see the sorting with quantity based*

                system("CLS");

                View\_Stock\_Header();

                Array\_count = load(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices);

                if (Option == '1') *// Assending*

                {

                  Sorting1(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                else if (Option == '2') *// Decesending*

                {

                  Sorting2(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                  Array\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                }

                Updatefile(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count);

                system("CLS");

              }

              else if (Option != 0) *// Incase of wrong option in Customers Menu*

              {

                SetConsoleTextAttribute(hConsole, 4); *// Red*

                gotoxy(30, 0);

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

                gotoxy(30, 1);

                cout << "\*\*\*\* CUSTOMERS \*\*\*\*";

                gotoxy(30, 2);

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

                SetConsoleTextAttribute(hConsole, 7); *// White*

                gotoxy(0, 4);

                cout << "You Enter a Wrong Number! ";

                getch();

                system("CLS");

              }

            }

            break; *// if Option ==  0*

          }

          break; *// if Option ==  0,Return to Main Menu*

        }

        else if (check != true) *// if Id and Passwords are incorrect then do this*

        {

          char Option = Retype\_Passwords();

          if (Option == '1')

          {

            SetConsoleTextAttribute(hConsole, 8); *// Gray*

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

            getch();

          }

          else if (Option == '2')

          {

            system("CLS");

            break;

          }

        }

      }

    }

    else if (Main\_Option == '4') *// At 4 Customers can sign up*

    {

      Customer\_Count = Customer\_SignUp(Customers\_Name, Customers\_Password, Customer\_Count);

    }

    else if (Main\_Option == '5') *// At 5 Exit*

    {

      system("CLS");

      Header();

      cout << "\n";

      SetConsoleTextAttribute(hConsole, 7);

      cout << "Thanks for using this Applicatin! " << endl;

      break;

    } *// Main Exit from Function*

    else

    {

      system("CLS");

      Header();

      cout << "\n";

      SetConsoleTextAttribute(hConsole, 4); *// Red*

      cout << "You Enter a Wrong Number! " << endl;

      SetConsoleTextAttribute(hConsole, 7); *// White*

      clearscreen();

    } *// Incase of wrong option in Main Options*

  } *// End Main While*

} *// End Main*

*// Function Definations*

void clearscreen() *// Function for clear screen*

{

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

  getch();

  system("CLS");

}

void gotoxy(int x, int y) *// function to use for going on specfic Position on console*

{

  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);

}

void Header() *// Header on Start*

{

  SetConsoleTextAttribute(hConsole, 6); *// yellow*

  gotoxy(50, 0);

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

  gotoxy(50, 1);

  cout << "!              Simple Shop Management System              !";

  gotoxy(50, 2);

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

  gotoxy(50, 4);

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

  gotoxy(50, 5);

  cout << "              !           MAIN MENU          !";

  gotoxy(50, 6);

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

  cout << endl;

}

char Entering() *// Who Is Entering*

{

  Header();

  SetConsoleTextAttribute(hConsole, 2); *// Green*

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

  cout << endl;

  cout << "1. Admin" << endl;

  cout << "2. Employees" << endl;

  cout << "3. Customers SignIN" << endl;

  cout << "4. Customers SignUp" << endl;

  cout << "5. Exit" << endl;

  cout << endl;

  char Op;

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  cout << "Your Option: ";

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Op;

  return Op;

}

string Admin\_Entering() *// Admin password enetring headrer and return Password for checking*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(30, 0);

  cout << "-------------------";

  gotoxy(30, 1);

  cout << "  \*\*\*\* Admin \*\*\*\*  ";

  gotoxy(30, 2);

  cout << "-------------------";

  string Password;

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 4);

  cout << "Admin Enter Password: ";

  SetConsoleTextAttribute(hConsole, 7); *// white*

  cin >> Password;

  return Password;

}

char Retype\_Passwords() *// Fuunction if the Password or Ids entered is wrong*

{

  cout << endl;

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  cout << "Wrong Input!" << endl;

  cout << endl;

  SetConsoleTextAttribute(hConsole, 2); *// Green*

  cout << "1. Retype..." << endl;

  cout << "2. Return to Main Menu..." << endl;

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  cout << endl;

  cout << "Enter your Option: ";

  char Opt;

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Opt;

  return Opt;

}

int Admin\_Menu() *// Menu Function for Admin , Select a option what to do and to return to main*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(30, 0);

  cout << "-------------------";

  gotoxy(30, 1);

  cout << "  \*\*\*\* ADMIN \*\*\*\*  ";

  gotoxy(30, 2);

  cout << "-------------------";

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 4);

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

  SetConsoleTextAttribute(hConsole, 2); *// Green*

  cout << "<1> View Stock " << endl;

  cout << "<2> Add New Employees " << endl;

  cout << "<3> Gross Income " << endl;

  cout << "<4> Transfer Salaries " << endl;

  cout << "<5> Pay Bills " << endl;

  cout << "<6> Total Profit " << endl;

  cout << "<7> History of Sales " << endl;

  cout << "<8> Offer Discount " << endl;

  cout << "<9> Leave Message" << endl;

  cout << "<0> Return to Starting Menu" << endl;

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(0, 16);

  cout << "Your Option: ";

  int Opt;

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Opt;

  return Opt;

}

void View\_Stock\_Header() *// Function Declaration For View Stock Header*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 0);

  cout << "-------------------";

  gotoxy(40, 1);

  cout << "  \*\*\*\* Admin \*\*\*\*  ";

  gotoxy(40, 2);

  cout << "-------------------";

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 4);

  cout << "     VIEW STOCK";

  Display\_Stock\_Header();

}

char Sorting\_Order() *// Function to ask for which order user wanted to see the Products*

{

  SetConsoleTextAttribute(hConsole, 4);

  gotoxy(30, 0);

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

  gotoxy(30, 1);

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

  gotoxy(30, 2);

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

  SetConsoleTextAttribute(hConsole, 3);

  gotoxy(30, 4);

  cout << "     VIEW STOCK" << endl;

  char Option = 0;

  gotoxy(0, 6);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "You can View stock in these Order...";

  gotoxy(0, 7);

  SetConsoleTextAttribute(hConsole, 7);

  cout << "1. Assending Order " << endl;

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

  gotoxy(0, 10);

  SetConsoleTextAttribute(hConsole, 3);

  cout << "Your Option: ";

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Option;

  return Option;

}

int Smallest(float Array[], int Size, int Position) *// find the Index where there is smallest Number in the Array for Asending order sorting*

{

  int index = Position;

  int temp = Array[Position];

  for (int x = Position; x < Size; x++)

  {

    if (temp > Array[x])

    {

      temp = Array[x];

      index = x;

    }

  }

  return index;

}

int Largest(float Array[], int Size, int Position) *// find the Index where there is Largest Number in the Array for Descending order sorting*

{

  int index = Position;

  int temp = Array[Position];

  for (int x = Position; x < Size; x++)

  {

    if (temp < Array[x])

    {

      temp = Array[x];

      index = x;

    }

  }

  return index;

}

void Sorting1(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Size)

{

  for (int x = 0; x < Size; x++)

  {

    int Ind = Smallest(Quantities, Size, x);

    int temp = Quantities[x];

    Quantities[x] = Quantities[Ind];

    Quantities[Ind] = temp;

    string temp1 = Product\_Names[x];

    Product\_Names[x] = Product\_Names[Ind];

    Product\_Names[Ind] = temp1;

    int temp2 = Purchase\_Prices[x];

    Purchase\_Prices[x] = Purchase\_Prices[Ind];

    Purchase\_Prices[Ind] = temp2;

    int temp3 = Selling\_Prices[x];

    Selling\_Prices[x] = Selling\_Prices[Ind];

    Selling\_Prices[Ind] = temp3;

  }

}

void Sorting2(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Size)

{

  for (int x = 0; x < Size; x++)

  {

    int Ind = Largest(Quantities, Size, x);

    int temp = Quantities[x];

    Quantities[x] = Quantities[Ind];

    Quantities[Ind] = temp;

    string temp1 = Product\_Names[x];

    Product\_Names[x] = Product\_Names[Ind];

    Product\_Names[Ind] = temp1;

    int temp2 = Purchase\_Prices[x];

    Purchase\_Prices[x] = Purchase\_Prices[Ind];

    Purchase\_Prices[Ind] = temp2;

    int temp3 = Selling\_Prices[x];

    Selling\_Prices[x] = Selling\_Prices[Ind];

    Selling\_Prices[Ind] = temp3;

  }

}

int Add\_Employees\_Header(string ID[], string Passwords[], int Employee\_Count) *// Functon for Adding Employees*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 0);

  cout << "------------------";

  gotoxy(40, 1);

  cout << "  \*\*\*\* Admin \*\*\*\* ";

  gotoxy(40, 2);

  cout << "------------------";

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 4);

  cout << " ADD NEW EMPLOYEES";

  string Id, Password;

  SetConsoleTextAttribute(hConsole, 6);

  gotoxy(0, 7);

  cout << "Enter the ID of the Employee: ";

  gotoxy(0, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin.ignore();

  getline(cin, Id);

  gotoxy(45, 7);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Assign Password: ";

  gotoxy(45, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Password;

  Add\_Employees\_Record\_In\_Array(ID, Passwords, Id, Password, Employee\_Count);

  Employee\_Count++;

  system("CLS");

  return Employee\_Count;

}

void Add\_Employees\_Record\_In\_Array(string ID[], string Passwords[], string Id, string Password, int Employee\_Count)

{

  ID[Employee\_Count] = Id;

  Passwords[Employee\_Count] = Password; *// Function For Adding Employees Data from variable to Array*

}

string Leaving\_Message() *// Function Header for Leaving Message*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 0);

  cout << "------------------";

  gotoxy(40, 1);

  cout << "  \*\*\*\* Admin \*\*\*\* ";

  gotoxy(40, 2);

  cout << "------------------";

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 4);

  cout << "  Leave Messages";

  string Message;

  SetConsoleTextAttribute(hConsole, 6);

  gotoxy(0, 7);

  cout << "Write the Message Here: ";

  gotoxy(24, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin.ignore();

  getline(cin, Message);

  system("CLS");

  return Message;

}

bool Employee\_Entering(string ID[], string Passwords[], int Employee\_Count) *// Employee entering Assigned Id and Password*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(30, 0);

  cout << "-------------------";

  gotoxy(30, 1);

  cout << "\*\*\*\* Employees \*\*\*\*";

  gotoxy(30, 2);

  cout << "-------------------";

  string Id, Password;

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 4);

  cout << "Enter Your Id: ";

  SetConsoleTextAttribute(hConsole, 7); *// white*

  gotoxy(0, 6);

  cin.ignore();

  getline(cin, Id);

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(30, 4);

  cout << "Enter Your Password: ";

  SetConsoleTextAttribute(hConsole, 7); *// white*

  gotoxy(30, 6);

  getline(cin, Password);

  bool flag = true;

  flag = Checking(ID, Passwords, Id, Password, Employee\_Count);

  return flag;

}

bool Checking(string ID[], string Passwords[], string Id, string Password, int Employee\_Count)

{

  for (int x = 0; x < Employee\_Count; x++)

  {

    if (ID[x] == Id && Passwords[x] == Password)

    {

      return true;

    }

  }

  return false;

}

int Employee\_Menu() *// Menu Function for Employee , Select a option what to do and to return to main*

{

  SetConsoleTextAttribute(hConsole, 6); *// Red*

  gotoxy(30, 0);

  cout << "-------------------";

  gotoxy(30, 1);

  cout << "\*\*\*\* Employees \*\*\*\*";

  gotoxy(30, 2);

  cout << "-------------------";

  SetConsoleTextAttribute(hConsole, 4); *// Yellow*

  gotoxy(0, 4);

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

  SetConsoleTextAttribute(hConsole, 2); *// Green*

  cout << "<1> Add Stock " << endl;

  cout << "<2> Update Stock " << endl;

  cout << "<3> Delete Stock " << endl;

  cout << "<4> View Stock " << endl;

  cout << "<5> Sale Products " << endl;

  cout << "<6> Check Admin Message " << endl;

  cout << "<7> View and Maintain Stock " << endl;

  cout << "<0> Return to Main Menu" << endl;

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(0, 14);

  cout << "Your Option: ";

  int Opt;

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Opt;

  return Opt;

}

void Display\_Stock\_Header() *// function for just display*

{

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 7);

  cout << "Sr.No";

  gotoxy(20, 7);

  cout << "Product\_Name";

  gotoxy(47, 7);

  cout << "Quantity";

  gotoxy(70, 7);

  cout << "Purchasing\_Price";

  gotoxy(101, 7);

  cout << "Selling\_Price";

  SetConsoleTextAttribute(hConsole, 7); *// White*

}

int Add\_Stock\_Header(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Product\_count)

{

  SetConsoleTextAttribute(hConsole, 4);

  gotoxy(50, 0);

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

  gotoxy(50, 1);

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

  gotoxy(50, 2);

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

  SetConsoleTextAttribute(hConsole, 3);

  gotoxy(50, 4);

  cout << "     ADD STOCK" << endl;

  int Count = Input\_Data(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Product\_count);

  system("CLS");

  return Count;

}

int Input\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Product\_count) *// Function for taking Input*

{

  string Name;

  float Quantity, Purchase\_Price, Selling\_Price;

  SetConsoleTextAttribute(hConsole, 6);

  gotoxy(0, 7);

  cout << "Name of Products";

  gotoxy(0, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin.ignore();

  getline(cin, Name);

  gotoxy(31, 7);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Quantity of Products";

  gotoxy(31, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Quantity;

  gotoxy(66, 7);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Purchasing price";

  gotoxy(66, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Purchase\_Price;

  gotoxy(97, 7);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Selling Price";

  gotoxy(97, 9);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Selling\_Price;

  Add\_Stock\_In\_Array(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Name, Quantity, Purchase\_Price, Selling\_Price, Product\_count);

  Product\_count++;

  return Product\_count;

}

void Add\_Stock\_In\_Array(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], string Name, float Quantity, float Purchase\_Price, float Selling\_Price, int Product\_count)

{

  Product\_Names[Product\_count] = Name;

  Quantities[Product\_count] = Quantity; *// Function For Adding Stock from variable to Array*

  Purchase\_Prices[Product\_count] = Purchase\_Price;

  Selling\_Prices[Product\_count] = Selling\_Price;

}

void Array\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count) *// Displaying the Data of the Array*

{

  int y = 9;

  for (int x = 0; x < Array\_count; x++)

  {

    gotoxy(0, y);

    cout << x + 1;

    gotoxy(20, y);

    cout << Product\_Names[x];

    gotoxy(47, y);

    cout << Quantities[x];

    gotoxy(70, y);

    cout << Purchase\_Prices[x];

    gotoxy(101, y);

    cout << Selling\_Prices[x];

    y++;

  }

  getch();

  system("CLS");

}

void Update\_Stock\_Header() *// Function Declaration For Updating Stock Header*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(42, 0);

  cout << "--------------------";

  gotoxy(42, 1);

  cout << " \*\*\*\* EMPLOYEE \*\*\*\* ";

  gotoxy(42, 2);

  cout << "--------------------";

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(42, 4);

  cout << "   UPDATE STOCK";

  Display\_Stock\_Header();

}

void Update(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Option) *// Function for Update the Stock*

{

  string Name;

  float Quantity, Purchase\_Price, Selling\_Price;

  cout << endl;

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Name of the Product : ";

  SetConsoleTextAttribute(hConsole, 7);

  cin.ignore();

  getline(cin, Name);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Quantity of Product : ";

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Quantity;

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Purchasing price : ";

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Purchase\_Price;

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Selling Price : ";

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Selling\_Price;

  Product\_Names[Option - 1] = Name;

  Quantities[Option - 1] = Quantity;

  Purchase\_Prices[Option - 1] = Purchase\_Price;

  Selling\_Prices[Option - 1] = Selling\_Price;

}

void Updating\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count) *// Array before updating , Displaying the Array*

{

  int y = 9;

  for (int x = 0; x < Array\_count; x++)

  {

    gotoxy(0, y);

    cout << x + 1;

    gotoxy(20, y);

    cout << Product\_Names[x];

    gotoxy(47, y);

    cout << Quantities[x];

    gotoxy(70, y);

    cout << Purchase\_Prices[x];

    gotoxy(101, y);

    cout << Selling\_Prices[x];

    y++;

  }

  int Option = 1;

  gotoxy(0, y + 1);

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  cout << "Enter your Option: ";

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Option;

  if (Option > 0)

  {

    Update(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Option);

    gotoxy(40, y + 8);

    SetConsoleTextAttribute(hConsole, 4); *// Red*

    cout << "Updated Successfully!";

    getch();

  }

  system("CLS");

}

void Delete\_Stock\_Header() *// Function Declaration For Deleting Stock Header*

{

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(42, 0);

  cout << "--------------------";

  gotoxy(42, 1);

  cout << " \*\*\*\* EMPLOYEE \*\*\*\* ";

  gotoxy(42, 2);

  cout << "--------------------";

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(42, 4);

  cout << "   DELETE STOCK";

  Display\_Stock\_Header();

}

int Delete(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count, int Option) *// Actual Function for Deleting the Data from the Array*

{

  if (Array\_count > 0)

  {

    for (int x = Option - 1; x < Array\_count - 1; x++)

    {

      Product\_Names[x] = Product\_Names[x + 1];

      Quantities[x] = Quantities[x + 1];

      Purchase\_Prices[x] = Purchase\_Prices[x + 1];

      Selling\_Prices[x] = Selling\_Prices[x + 1];

    }

    Array\_count--; *// Because after Deletition , Size of the Array will be decrease*

  }

  return Array\_count;

}

int Deleting\_Data(string Product\_Names[], float Quantities[], float Purchase\_Prices[], float Selling\_Prices[], int Array\_count) *// Array before Deleting , Displaying the Array*

{

  int y = 9;

  for (int x = 0; x < Array\_count; x++)

  {

    gotoxy(0, y);

    cout << x + 1;

    gotoxy(20, y);

    cout << Product\_Names[x];

    gotoxy(47, y);

    cout << Quantities[x];

    gotoxy(70, y);

    cout << Purchase\_Prices[x];

    gotoxy(101, y);

    cout << Selling\_Prices[x];

    y++;

  }

  int Option;

  gotoxy(0, y + 1);

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  cout << "Enter your Option: ";

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Option;

  if (Option > 0)

  {

    Array\_count = Delete(Product\_Names, Quantities, Purchase\_Prices, Selling\_Prices, Array\_count, Option);

    gotoxy(40, y + 3);

    SetConsoleTextAttribute(hConsole, 4); *// Red*

    cout << "Dleted Successfuly!";

    getch();

  }

  system("CLS");

  return Array\_count;

}

void Reading\_Message(string Message) *// Function Header for Reading Message*

{

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 0);

  cout << "--------------------";

  gotoxy(40, 1);

  cout << " \*\*\*\* EMPLOYEE \*\*\*\* ";

  gotoxy(40, 2);

  cout << "--------------------";

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 4);

  cout << "  Read Messages";

  cout << Message;

  getch();

  system("CLS");

}

int Customer\_SignUp(string Customers\_Name[], string Customers\_Password[], int Customer\_Count) *// Function for Customers Signup*

{

  system("CLS");

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 0);

  cout << "---------------------";

  gotoxy(40, 1);

  cout << " \*\*\*\* CUSTOMERS \*\*\*\* ";

  gotoxy(40, 2);

  cout << "---------------------";

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 4);

  cout << "       SIGN UP";

  string Name, Password, Retype\_Password;

  SetConsoleTextAttribute(hConsole, 6);

  gotoxy(0, 7);

  cout << "Enter Your Name:";

  gotoxy(16, 8);

  SetConsoleTextAttribute(hConsole, 7);

  cin.ignore();

  getline(cin, Name);

  gotoxy(0, 10);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Enter the Password:";

  gotoxy(19, 11);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Password;

  gotoxy(0, 13);

  SetConsoleTextAttribute(hConsole, 6);

  cout << "Retype the Password:";

  gotoxy(20, 14);

  SetConsoleTextAttribute(hConsole, 7);

  cin >> Retype\_Password;

  if (Password == Retype\_Password)

  {

    Add\_Customers\_Record\_In\_Array(Customers\_Name, Customers\_Password, Name, Password, Customer\_Count);

    Customer\_Count++;

    gotoxy(40, 16);

    SetConsoleTextAttribute(hConsole, 4); *// Red*

    cout << "SignUp Successfully!";

    getch();

  }

*// else // incase if retype Password is incorrect then do this*

*// {*

*// }*

  system("CLS");

  return Customer\_Count;

}

void Add\_Customers\_Record\_In\_Array(string Customers\_Name[], string Customers\_Password[], string Name, string Password, int Customer\_Count)

{

  Customers\_Name[Customer\_Count] = Name;

  Customers\_Password[Customer\_Count] = Password; *// Function For Adding Employees Data from variable to Array*

}

bool Customer\_SignIn(string Customers\_Name[], string Customers\_Password[], int Customer\_Count) *// for customers SignIn*

{

  system("CLS");

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(40, 0);

  cout << "---------------------";

  gotoxy(40, 1);

  cout << " \*\*\*\* CUSTOMERS \*\*\*\* ";

  gotoxy(40, 2);

  cout << "---------------------";

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(40, 4);

  cout << "       SIGN IN";

  string Name, Password;

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 6);

  cout << "Enter Your Name: ";

  SetConsoleTextAttribute(hConsole, 7); *// white*

  gotoxy(0, 8);

  cin.ignore();

  getline(cin, Name);

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(30, 6);

  cout << "Enter Your Password: ";

  SetConsoleTextAttribute(hConsole, 7); *// white*

  gotoxy(30, 8);

  getline(cin, Password);

  bool flag = true;

  flag = Checking(Customers\_Name, Customers\_Password, Name, Password, Customer\_Count);

  return flag;

}

bool Customer\_Checking(string Customers\_Name[], string Customers\_Password[], string Name, string Password, int Customer\_Count)

{

  for (int x = 0; x < Customer\_Count; x++)

  {

    if (Customers\_Name[x] == Name && Customers\_Password[x] == Password)

    {

      return true;

    }

  }

  return false;

}

int Customer\_Menu() *// Menu Function for Customer , Select a option what to do and to return to main*

{

  SetConsoleTextAttribute(hConsole, 3); *// Aqua*

  gotoxy(30, 0);

  cout << "-------------------";

  gotoxy(30, 1);

  cout << "\*\*\*\* CUSTOMERS \*\*\*\*";

  gotoxy(30, 2);

  cout << "-------------------";

  SetConsoleTextAttribute(hConsole, 4); *// Red*

  gotoxy(0, 4);

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

  SetConsoleTextAttribute(hConsole, 2); *// Green*

  cout << "<1> View Stock " << endl;

  cout << "<2> View Discounted Products " << endl;

  cout << "<0> Return to Main Menu" << endl;

  SetConsoleTextAttribute(hConsole, 6); *// Yellow*

  gotoxy(0, 9);

  cout << "Your Option: ";

  int Opt;

  SetConsoleTextAttribute(hConsole, 7); *// White*

  cin >> Opt;

  return Opt;

}

void store(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[], int Product\_count)

{

  fstream myFile;

  myFile.open("Stock.txt", ios::app);

  for (int x = Product\_count - 1; x < Product\_count; x++)

  {

    myFile << Product\_Name[x] << "," << Quantity[x] << "," << Purchase\_Prices[x] << "," << Selling\_Price[x] << endl;

  }

  myFile.close();

}

void Updatefile(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[], int Product\_count)

{

  fstream myFile;

  myFile.open("Stock.txt", ios::out);

  for (int x = 0; x < Product\_count; x++)

  {

    myFile << Product\_Name[x] << "," << Quantity[x] << "," << Purchase\_Prices[x] << "," << Selling\_Price[x] << endl;

  }

  myFile.close();

}

string parseData(string record, int field)

{

  int comma = 1;

  string item;

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

  {

    if (record[x] == ',')

    {

      comma = comma + 1;

    }

    else if (comma == field)

    {

      item = item + record[x];

    }

  }

  return item;

}

int load(string Product\_Name[], float Quantity[], float Purchase\_Prices[], float Selling\_Price[])

{

  fstream f;

  string record;

  int idx = 0;

  f.open("Stock.txt", ios::in);

  while (!(f.eof()))

  {

    getline(f, record);

    if (record != "")

    {

      Product\_Name[idx] = parseData(record, 1);

      Quantity[idx] = stof(parseData(record, 2));

      Purchase\_Prices[idx] = stof(parseData(record, 3));

      Selling\_Price[idx] = stof(parseData(record, 4));

      idx++;

    }

  }

  f.close();

  return idx;

}

**Student Reg. No:**  2021-CS-144  **Student Name.**  Muhammad Umair Shahid

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **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. |