Dairy Delights

Session 2023 - 2027

Submitted by:

Abu Tayyab 2023-CS-54

Supervised by:

Dr. Awais Hassan

Course:

CSC-102 Programming Fundamentals

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

Table of Wire frames

1. Introduction:

Dairy Delights Business Application is a console-based application designed to manage operations for a dairy products business. This application provides functionalities for three types of users: Admin, Employees, and Customers. The application offers a user-friendly interface allowing efficient management of tasks related to inventory, sales, and customer interactions.

2. User Roles and Functionalities:

Dairy Delights supports three user roles: Admin, Employees, and Customers. Each role has distinct functionalities tailored to their specific needs.

i) Admin:

1) Add new employee to the system.
2) Remove employees from the system.
3) Set discounts.
4) Remove discounts.
5) Update prices of products.
6) Add more products in inventory.
7) Buy more stock for the inventory.
8) Remove products from the inventory.
9) Review complains form employees.
10) Check review about products form user.
11) Change password of any user.

ii) Employees:

1) View products which are currently for sale.
2) Add new products to the inventory.
3) Remove products from the inventory.
4) Update the prices of products.
5) View available discounts on products.
6) Remove discount.
7) Write details about product so that user can easily understand what product is.
8) Write complains to the Admin.
9) Check review about products form user.
10) Change password but not of Admin.

iii) Customers:

(1) View Products
(2) Add to Cart.

(3) Remove products form the cart or complete delete it.
(4) View what is in the cart.
(5) See Information about a Product.
(6) Give Review about a product.
(7) View Discount Available Right now.
(8) Check out and print the invoice
(9) Change password.

As a	I want to perform	So that I can
Admin	Add new employee.	Add a new employee to the system.
	Remove employee.	Remove employee from the system.
	Set discount.	Offer discount on my products.
	Remove discount.	Remove discount form products.
	Update prices.	Change increase or maybe decrease the price of items which I like.
	Add products.	Add new products for sale in my inventory.
	Buy more stock.	Increase the quantity of products that are available for sale.
	Remove products.	Remove products form inventory which are for sale.
	Review complains.	See that which employee have issues and I can solve it.
	Check review of customer.	See that review form user and is customer liked the product or not.
	Change password.	Can change password of any username without even knowing old passwords.
Employee	View products.	See products available for sale.
	Add products	Add new products for sale in inventory.
	Remove products	Remove products form the inventory
	Update prices.	Change increase or maybe decrease the price of items.
	View available discount.	See discount currently available.
	Remove discount.	Remove discount form the products.
	Writes detail about products.	So that user can easily understand about product.
	Writes complain to Admin.	Writes the issues he is facing to the admin so that issues can be resolved.
	Check review of customer.	See that review form user and is customer liked the product or not.

	Change password.	He can change password but cannot change password of employee.
customer	View products.	See products available for sale.
	Add to cart.	Things he wants to buy.
	Edit the cart.	Delete the whole cart or remove some products from the cart which he does not want to buy now.
	View cart	See what is in the cart right now.
	View discounts	See discount available right now.
	Give feedback	Tell how was the experience
	See information about products	See info about products
	Change password	Change login password
	Check out	Check out and print invoice.

3. Wire Frames:

Figure 1Header

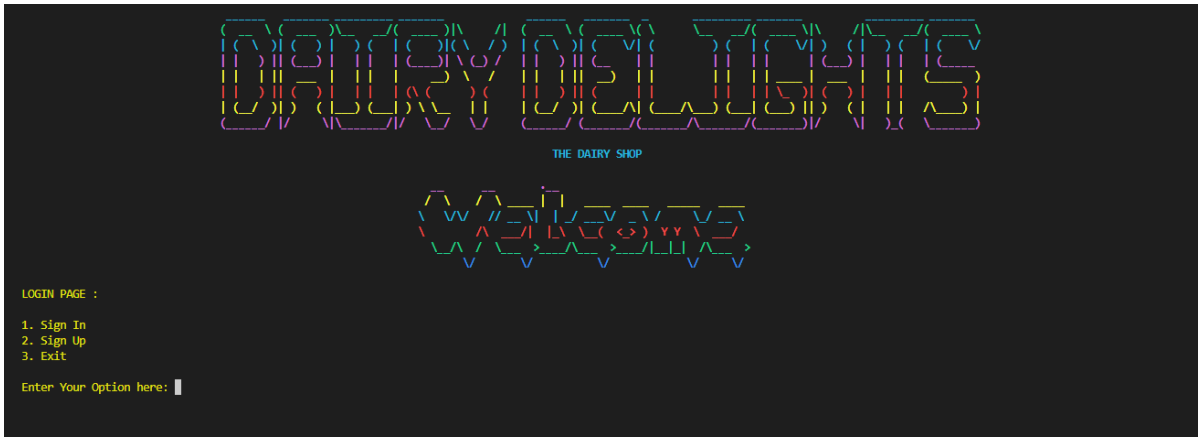

Figure 2 Welcome Page

Figure 3Sign In page

Figure 4 Sign Up

Figure 5 Admin Menu

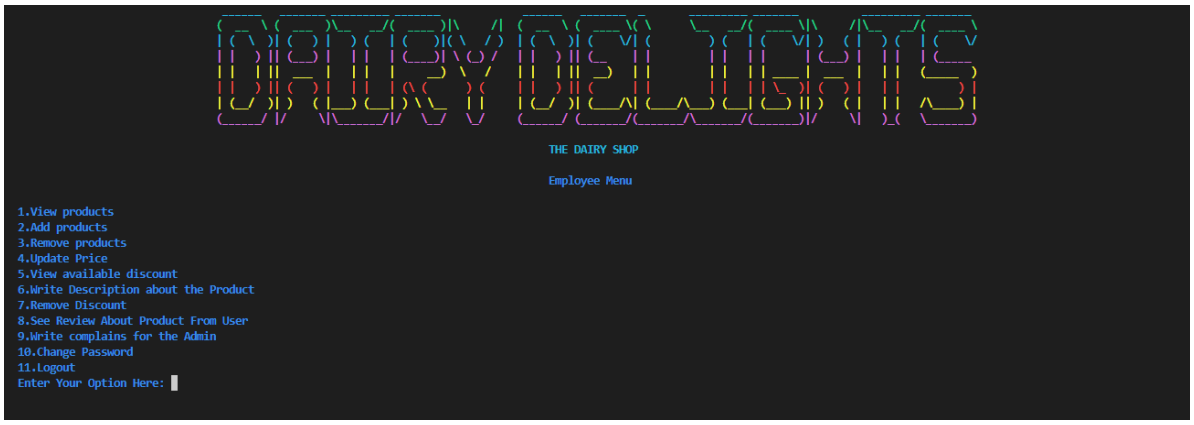

Figure 6 Employee Menu

Figure 7 Customer Menu

Figure 8 Invoice

4.Data Structures

```
string username[100];
string username[100] ;
string role[100];
string complains_FromEmployees[100];
string cart[100];
string products[100];
string products_Description[100];
string Customer_Review_AboutProducts[100];
int products_Price[100];
int cart_Quantity[100];
int products_Quantity[100];
int productcount;
int usercount;
int discount;
int cart_Index;
int review_Index;
int complain_Index;
float cart_Price[100];
```

5.Functions prototypes

```
void Print_header();        // for printing the name of app.
void print_Welcome();       // for printing Welcome.
void printTankYouCapital(); // printing Thank You.
void successful();          // to tell that your instruction is done successfully.
void Unsuccessful();        // to tell that there was some problem which maybe because
invalid input.
void pressAnyKey();         // for holding screen.
void invalid_Input();       // for printing that your Input is wrong.
void print_Thanks();        // for printing thanks.

string login_page();              // it will return the option form login page .
string admin_menu();              // it will display functions of admin and return which
user wants to do.
string input_Name();              // it will take the name of product and return it.
string input_Username();          // it will take username of person and return it.
string input_Password();          // it will take password form user and return ir.
```

string input_Role(); // for taking the role which may be customer or employee.
string input_NewPassword(); // to take new password so that it can be replaced with old one.
string input_NameToViewComplain(); // for asking user to input name so that complain can be traced.
string employee_menu(); // it will display employee options and return what user wants to do.
string input_Description(); // for writing description for the products.
string input_OldPassword(); // for inputing old password to aithorize the password change.
string input_NameForReview(); // for user to give review about product.
string input_UsernameFor_Complain(); // for inputing username so that complain have name form which it has come.
string input_Complain(); // forwriting the complain.
string customer_menu(); // fordisplaying the customer options and returning what he wants to do.
string get_ProductName(); // for taking product name from user.
string get_Opinion_ForDeletion(); // for displaying thatcustomer wants to delete complete cart or just some items.
string input_NameForDescription(); // forinput product name for description.
string input_NameFor_Review(); // for input product name for writing review
string input_Review(); // for writing the review.
string payment_option(); // for asking user what will be the payment method.

// it will take things name and pasword and return what is this its role means customer or employee.
string sign_In(string user, string pass, string username[], string password[], string roleArray[]);
// it will take a username and return the role of that username.
string user_Behind_Username(string user, string username[], string role[], int count);

// it will take username , password, role and then verifies all the things and then enter user in data base.
bool sign_Up(string user, string pass, string role, string username[], string password[], string roleArray[], int count);

// it will add new employee to the data base.
bool add_Employee(string user, string pass, string username[], string password[], string roleArray[], int count);

```
// it will remove employee form data base.
bool remove_Employee(string user, string username[], string password[], string roleArray[], int count);

// it will allow admin to change anyone password withour khnowing their password.
bool change_Password_ForAdmin(string user, string new_password, string username[], string password[], int count);

// it will print all the products that are available right now.
bool print_Products(string products[], int products_Quantity[], int products_Price[], int &productcount);

// it is for that new productscan be added to the store.
bool add_NewProducts(string name, int quantity, int price, string products[], int products_Quantity[], int products_Price[], int &productcount);

// this function will allow user to buy more stock for his shop.
bool buy_MoreStock(string name, int quantity, string products[], int products_Quantity[], int products_Price[], int &productcount);

// so that user entered is valid for exanple it cannot be admin.
bool validUser(string user);

// so that password must be higher or equal to 3 letters.
bool validPassword(string pass);

// to validate the role of user.
bool validRole(string role);

// for removings products from the data base.
bool remove_Product(string name, string products[], int products_Quantity[], int products_Price[], int &productcount);

// for changing the pass word it will need old password in order to change it.
bool change_Password(string user, string old_Password, string new_password, string username[], string password[], int count);

// to see the complain form employee.
```

bool see_ComplainFromEmployee(string name, string username[], string complains_FromEmployees[], int usercount);

// so that description can be given about the products.
bool give_DiscriptionAboutProducts(string name, string discription, string products[], string product_Description[], int &productcount);

// to validate the username so that is cannot be admin or have spelling mistakes.
bool valid_Username(string user, string username[], string role[], int count);

// for user to give review about product whichh he liked or may be ot liked.
bool review_OfProduct(string name, string products[], string customer_Review_AboutProduct[], int productcount);

// for employee to give what problems he is facing,
bool give_Complains(string user, string complain, string username[], string complains_FromEmployees[], int usercount);

// for the customer that it can only change the password of user not any employee to validate this thing this function is used.
bool validRolecustomer(string role);

// for the user to add product to cart and then pay only when heis done shopping.
bool added_InCart(string name, int quantity, string products[], int products_Quantity[], string cart[], int cart_Quantity[], int &cart_index, int product_index);

// for empying the whole cart.
bool Delete_CompleteCart(string cart[], int cart_Quantity[], int &cart_Index);

// for printig what is in the cart so far
bool print_Items_InCart(string cart[], int cart_Quantity[], int cart_Index);

// for deleting some productsfrom the cart.
bool Delete_ProductFromCart(string name, string cart[], int cart_Quantity[], int &cart_Index);

// so that username must be present in data base or it will be invalid.
bool valid_Username_Customer(string user, string username[], string role[], int count);

```
// view details about a specific product.
bool View_DescriptionOfProduct(string name, string products[], string products_Description[], int productcount);

// for user to give review about a product.
bool give_ReviewAboutProducts(string name, string review, string products[], string Customer_Review_AboutProducts[], int productcount);

// to check only card or cash is input byuser nothing else.
bool Validate_Payment_Option(string payment_method);

// at the end to creating the invoie/bill of shopping.
bool print_Invoice(float result, int discount, string payment_Method, float final_Price);
// will update the  existing price of product
bool update_PriceOfProduct(string name, int price, string products[], int products_Price[], int productcount);


int get_percentage_of_Discount();     // for taking percentage of discount.
int remove_Discount(int &discount);   // for removing the discount.
int input_Productquantity();          // for taking the quantity of product.
int input_ProductPrice();             // for entering the price of product.
int available_Discount(int discount); // to see how much discount is currently availabe.
// it will check if product is already preset and user bought more then increase the quantity which was in cart.
int already_PresentThenUpdate_Quantity(string name, int quality, string products[], int products_Quantity[], string cart[], int cart_Quantity[], int productcount, int cart_Index);
int Get_QuantityForCart(); // to ask user how much he wants to buy.
// for checkingout means the products in cart are ready to bought.
float checkoutCart(string cart[], int cart_Quantity[], string products[], int products_Quantity[], int product_price[], float cart_Price[], int &cart_Index, int productcount);
float Total_Amount(float result, int discount, string payment_Method); // for calculating the whole price after discount and tax.
```

6.Flow Chart

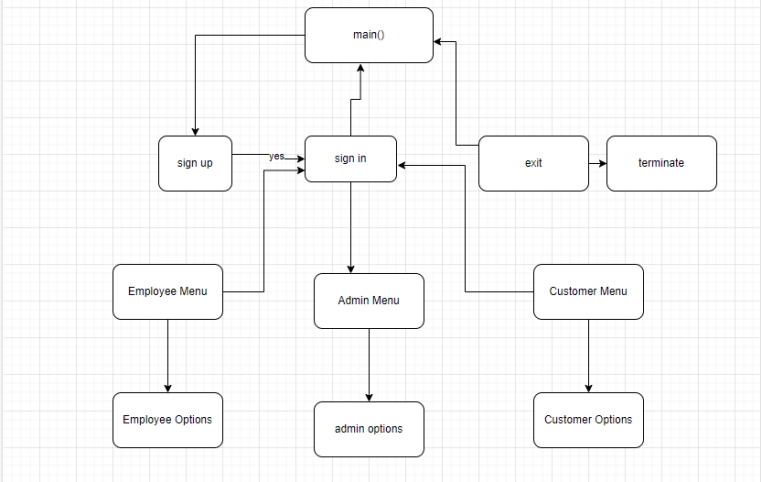

Figure 9 Flow Chart

7.Complete Code of Business Application

```
#include <iostream>
#include <conio.h>
#include <windows.h>
#include <string>
using namespace std;

void Print_header();       // for printing the name of app.
void print_Welcome();      // for printing Welcome.
void printTankYouCapital(); // printing Thank You.
void successful();         // to tell that your instruction is done successfully.
void Unsuccessful();       // to tell that there was some problem which maybe because
invalid input.
void pressAnyKey();        // for holding screen.
void invalid_Input();      // for printing that your Input is wrong.
void print_Thanks();       // for printing thanks.

string login_page();             // it will return the option form login page .
string admin_menu();             // it will display functions of admin and return which
user wants to do.
```

string input_Name(); // it will take the name of product and return it.
string input_Username(); // it will take username of person and return it.
string input_Password(); // it will take password form user and return ir.
string input_Role(); // for taking the role which may be customer or employee.
string input_NewPassword(); // to take new password so that it can be replaced with old one.
string input_NameToViewComplain(); // for asking user to input name so that complain can be traced.
string employee_menu(); // it will display employee options and return what user wants to do.
string input_Description(); // for writing description for the products.
string input_OldPassword(); // for inputing old password to aithorize the password change.
string input_NameForReview(); // for user to give review about product.
string input_UsernameFor_Complain(); // for inputing username so that complain have name form which it has come.
string input_Complain(); // forwriting the complain.
string customer_menu(); // fordisplaying the customer options and returning what he wants to do.
string get_ProductName(); // for taking product name from user.
string get_Opinion_ForDeletion(); // for displaying thatcustomer wants to delete complete cart or just some items.
string input_NameForDescription(); // forinput product name for description.
string input_NameFor_Review(); // for input product name for writing review
string input_Review(); // for writing the review.
string payment_option(); // for asking user what will be the payment method.
// it will take things name and pasword and return what is this its role means customer or employee.
string sign_In(string user, string pass, string username[], string password[], string roleArray[]);
// it will take a username and return the role of that username.
string user_Behind_Username(string user, string username[], string role[], int count);

// it will take username , password, role and then verifies all the things and then enter user in data base.
bool sign_Up(string user, string pass, string role, string username[], string password[], string roleArray[], int count);

// it will add new employee to the data base.

```
bool add_Employee(string user, string pass, string username[], string password[], string roleArray[], int count);

// it will remove employee form data base.
bool remove_Employee(string user, string username[], string password[], string roleArray[], int count);

// it will allow admin to change anyone password withour khnowing their password.
bool change_Password_ForAdmin(string user, string new_password, string username[], string password[], int count);

// it will print all the products that are available right now.
bool print_Products(string products[], int products_Quantity[], int products_Price[], int &productcount);

// it is for that new productscan be added to the store.
bool add_NewProducts(string name, int quantity, int price, string products[], int products_Quantity[], int products_Price[], int &productcount);

// this function will allow user to buy more stock for his shop.
bool buy_MoreStock(string name, int quantity, string products[], int products_Quantity[], int products_Price[], int &productcount);

// so that user entered is valid for exanple it cannot be admin.
bool validUser(string user);

// so that password must be higher or equal to 3 letters.
bool validPassword(string pass);

// to validate the role of user.
bool validRole(string role);

// for removings products from the data base.
bool remove_Product(string name, string products[], int products_Quantity[], int products_Price[], int &productcount);

// for changing the pass word it will need old password in order to change it.
bool change_Password(string user, string old_Password, string new_password, string username[], string password[], int count);
```

// to see the complain form employee.
bool see_ComplainFromEmployee(string name, string username[], string complains_FromEmployees[], int usercount);

// so that description can be given about the products.
bool give_DiscriptionAboutProducts(string name, string discription, string products[], string product_Description[], int &productcount);

// to validate the username so that is cannot be admin or have spelling mistakes.
bool valid_Username(string user, string username[], string role[], int count);

// for user to give review about product whichh he liked or may be ot liked.
bool review_OfProduct(string name, string products[], string customer_Review_AboutProduct[], int productcount);

// for employee to give what problems he is facing,
bool give_Complains(string user, string complain, string username[], string complains_FromEmployees[], int usercount);

// for the customer that it can only change the password of user not any employee to validate this thing this function is used.
bool validRolecustomer(string role);

// for the user to add product to cart and then pay only when heis done shopping.
bool added_InCart(string name, int quantity, string products[], int products_Quantity[], string cart[], int cart_Quantity[], int &cart_index, int product_index);

// for empying the whole cart.
bool Delete_CompleteCart(string cart[], int cart_Quantity[], int &cart_Index);

// for printig what is in the cart so far
bool print_Items_InCart(string cart[], int cart_Quantity[], int cart_Index);

// for deleting some productsfrom the cart.
bool Delete_ProductFromCart(string name, string cart[], int cart_Quantity[], int &cart_Index);

// so that username must be present in data base or it will be invalid.

```
bool valid_Username_Customer(string user, string username[], string role[], int count);

// view details about a specific product.
bool View_DescriptionOfProduct(string name, string products[], string products_Description[], int productcount);

// for user to give review about a product.
bool give_ReviewAboutProducts(string name, string review, string products[], string Customer_Review_AboutProducts[], int productcount);

// to check only card or cash is input byuser nothing else.
bool Validate_Payment_Option(string payment_method);

// at the end to creating the invoie/bill of shopping.
bool print_Invoice(float result, int discount, string payment_Method, float final_Price);
// will update the  existing price of product
bool update_PriceOfProduct(string name, int price, string products[], int products_Price[], int productcount);


int get_percentage_of_Discount();     // for taking percentage of discount.
int remove_Discount(int &discount);   // for removing the discount.
int input_Productquantity();          // for taking the quantity of product.
int input_ProductPrice();             // for entering the price of product.
int available_Discount(int discount); // to see how much discount is currently availabe.
// it will check if product is already preset and user bought more then increase the quantity which was in cart.
int already_PresentThenUpdate_Quantity(string name, int quality, string products[], int products_Quantity[], string cart[], int cart_Quantity[], int productcount, int cart_Index);
int Get_QuantityForCart(); // to ask user how much he wants to buy.
// for checkingout means the products in cart are ready to bought.
float checkoutCart(string cart[], int cart_Quantity[], string products[], int products_Quantity[], int product_price[], float cart_Price[], int &cart_Index, int productcount);
float Total_Amount(float result, int discount, string payment_Method); // for calculating the whole price after discount and tax.

main()
```

```
{

   string username[100] = {"tayyab", "ali", "ahmad"};
// it is an array of usernames.
   string password[100] = {"1234", "1234", "1234"};
// it is parrallel to username and it will store password.
   string role[100] = {"admin", "employee", "customer"};
// it is parrallel to username and it will store roles of users.
   string complains_FromEmployees[100];
// array for storing complain form employees.
   string cart[100];
// cart which will have all the products user wants to buy.
   string products[100] = {"milk", "yougurt", "chedder", "mozerella"};
// for storing the names of products.
   string products_Description[100] = {"Milk is very good for bones", "Sweet Yougurt
which is gluten free", "Type of Cheese which is best for Sandwiches", "Type of Cheese
which is best for Pizza"}; // array of description.
   string Customer_Review_AboutProducts[100];
// array for storing reviews form user.

   int products_Price[100] = {200, 300, 1500, 2000}; // this array will store price of
products.
   int cart_Quantity[100];          // for quantity of products which are in cart
right now.
   int products_Quantity[100] = {80, 20, 8, 10};     // how much stock is awailabe of
products.
   int productcount = 4;            // how much products are available.
   int usercount = 3;               // how much users have ben entere so far.
   int discount = 0;                // right now discount available
   int cart_Index = 0;              // how many items in cart.
   int review_Index = 0;            // how many reviews available.
   int complain_Index = 0;          // how many complains available.
   float cart_Price[100];           // total price of products in cart.

   while (true) // it is infinite loop it will keep repeating untill user wants to exit.
   {
      system("cls");   // for clearing the screen.
      Print_header();  // for printing Header.
      print_Welcome(); // for printing welcome.
      string login_value;
```

```
        login_value = login_page(); // it will tell which option is selected by user.

        if (login_value == "1") // if will execute only when 1 is selected by user.
        {
            string user, pass, person_role; // variable daclaration

            system("cls");
            Print_header(); // agian header will print

            user = input_Username(); // gather info
            pass = input_Password();

            person_role = sign_In(user, pass, username, password, role); // pass to function
which will return a value which will be admin or employee or customer.

            if (person_role == "admin") // it will true only when role id admin.
            {
                while (true) // again infinite loop.
                {
                    system("cls");

                    string admin_option; // variable daclaration

                    admin_option = admin_menu();

                    if (admin_option == "1") // it will execute when user wants to do work on
option 1
                    {
                        bool check; // variable daclaration
                        bool check_employee;
                        bool validate;

                        string user_admin, pass_admin, role_admin;
                        system("cls");
                        Print_header(); // agian header will print
```

```
                user_admin = input_Username();
                validate = validUser(user_admin); // to check the user is valid.
                if (validate)
                {
                   pass_admin = input_Password();
                   validate = validPassword(pass_admin); // then check password validity.
                   if (validate)
                   {
                      role_admin = input_Role(); // then check role validity.
                      validate = validRole(role_admin);
                      if (validate)
                      {
                         check = sign_Up(user_admin, pass_admin, role_admin, username, password, role, usercount); // if itis true then user have been entered in data base.

                         if (check) // if true the add one to numbers of users in data base.
                         {
                            usercount++;
                            pressAnyKey();
                         }
                         else
                         {
                            pressAnyKey();
                         }
                      }
                      else
                      {
                         Unsuccessful(); // it means that input was not right and action cannoy be done.
                         continue;
                      }
                   }
                   else
                   {
                      pressAnyKey(); // agian input mistake.
                      continue;
                   }
```

```
            }
            else // agian action cannot be done because of invalid input.
            {
                pressAnyKey();
                continue;
            }
        }
        else if (admin_option == "2") // in this section a user can be removed.
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string remove_employee_user;

            remove_employee_user = input_Username();

            check = remove_Employee(remove_employee_user, username,
password, role, usercount);

            if (check)
            {
                successful();
                usercount--;
                getch();
            }
            else
            {
                invalid_Input();
                getch();
            }
        }
        else if (admin_option == "3") // this set the discount.
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            discount = get_percentage_of_Discount();
            if (discount >= 100)
            {
```

```
            discount = 0;
            getch();
        }
        else
        {
            successful();
        }
        continue;
    }
    else if (admin_option == "4") // this will remove the discount.
    {
        system("cls");
        Print_header(); // agian header will print
        int remove;
        remove = remove_Discount(discount);
        if (remove > discount)
        {
            getch();
        }
        else
        {
            discount = discount - remove;
            successful();
        }
        continue;
    }
    else if (admin_option == "5") // this will print the available products.
    {
        system("cls");
        Print_header(); // agian header will print
        bool check;
        string name;
        int price;

        name = input_Name();
        price = input_ProductPrice();
        if (price == 65535)
        {
            pressAnyKey();
```

```
                continue;
            }

            if (price != 65535)
            {
                check = update_PriceOfProduct(name,
price,products,products_Price,productcount);
                if (check)
                {
                    successful();
                }
                else
                {
                    pressAnyKey();
                }
                continue;
            }
        }
        else if (admin_option == "6")          // for adding new products.
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;
            int quantity;
            int price;

            name = input_Name();
            quantity = input_Productquantity();
            if (quantity == 65535)
            {
                pressAnyKey();
                continue;
            }
            price = input_ProductPrice();
            if (price == 65535)
            {
                pressAnyKey();
                continue;
```

```
                }

                if (price != 65535)
                {

                    check = add_NewProducts(name, quantity, price, products, products_Quantity, products_Price, productcount);
                    if (check)
                    {
                        successful();
                    }
                    else
                    {
                        pressAnyKey();
                    }
                    continue;
                }
            }
            else if (admin_option == "7") // for buying new stock.
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                string name;
                int quantity;

                name = input_Name();
                quantity = input_Productquantity();
                if(quantity != 65535)
                {
                    check = buy_MoreStock(name, quantity, products, products_Quantity, products_Price, productcount); // for buying more stock;
                    if (check)
                    {
                        successful();
                    }
                    else
                    {
                        pressAnyKey();
```

```
                }

                continue;
            }
            else
            {
                pressAnyKey();
                continue;
            }

        }
        else if (admin_option == "8") // this section will remove products.
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;

            name = input_Name();

            check = remove_Product(name, products, products_Quantity,
products_Price, productcount);
            if (check)
            {
                successful();
            }
            else
            {
                pressAnyKey();
            }
            continue;
        }
        else if (admin_option == "9") // to view complain from employee.
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;
            name = input_NameToViewComplain();
```

```
                check = see_ComplainFromEmployee(name, username, complains_FromEmployees, usercount);

                if (check)
                {
                    pressAnyKey();
                }
                else
                {
                    pressAnyKey();
                }
                continue;
            }
            else if (admin_option == "10")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                string name;
                name = input_NameForReview();

                check = review_OfProduct(name, products, Customer_Review_AboutProducts, productcount);
                if (check)
                {
                    pressAnyKey();
                }
                else
                {
                    pressAnyKey();
                }
                continue;

            }
            else if (admin_option == "11") // to change the password
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
```

```
                bool valid;
                string user;
                string new_password;

                user = input_Username();
                new_password = input_NewPassword();
                valid = validPassword(new_password);
                if (valid)
                {

                    check = change_Password_ForAdmin(user, new_password,
username, password, usercount);
                    if (check)
                    {
                        successful();
                    }
                    else
                    {
                        pressAnyKey();
                    }
                    continue;
                }
                else
                {
                    pressAnyKey();
                    continue;
                }

            }
            else if (admin_option == "12") // to exit the loop
            {
                break;
            }
            else
            {
                invalid_Input();
                continue;
            }
        }
```

```
        }
        else if (person_role == "employee") // it will execute only when role employee is return
        {
            while (true) // infinite loop for asking the employee what he wants to do untill he wants to exit the loop
            {
                string employee_Option;

                employee_Option = employee_menu(); // it will print the employee menu and return which option is returned
                if (employee_Option == "1")       // execute only when option 1 is selected
                {
                    system("cls");
                    Print_header(); // agian header will print
                    bool check;
                    check = print_Products(products, products_Quantity, products_Price, productcount);
                    pressAnyKey();
                    continue;
                }
                else if (employee_Option == "2")
                {
                    system("cls");
                    Print_header(); // agian header will print
                    bool check;
                    string name;
                    int quantity;
                    int price;

                    name = input_Name();
                    quantity = input_Productquantity();
                    if (quantity == 65535)
                    {
                        pressAnyKey();
                        continue;
                    }
                    price = input_ProductPrice();
                    if (price == 65535)
```

```
                {
                    pressAnyKey();
                    continue;
                }

                if (price != 65535)
                {

                    check = add_NewProducts(name, quantity, price, products, products_Quantity, products_Price, productcount);
                    if (check)
                    {
                        successful();
                    }
                    else
                    {
                        pressAnyKey();
                    }
                    continue;
                }
            }
            else if (employee_Option == "3")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                string name;

                name = input_Name();

                check = remove_Product(name, products, products_Quantity, products_Price, productcount);
                if (check)
                {
                    successful();
                }
                else
                {
                    pressAnyKey();
```

```
            }
            continue;
        }
        else if (employee_Option == "4")
        {

            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;
            int price;

            name = input_Name();
            price = input_ProductPrice();
            if (price == 65535)
            {
                pressAnyKey();
                continue;
            }

            if (price != 65535)
            {
                check = update_PriceOfProduct(name,
price,products,products_Price,productcount);
                if (check)
                {
                    successful();
                }
                else
                {
                    pressAnyKey();
                }
                continue;
            }

        }
        else if (employee_Option == "5")
        {
            system("cls");
```

```
            Print_header(); // agian header will print
            int result;

            result = available_Discount(discount);
            getch();
            continue;
        }
        else if (employee_Option == "6")
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string description;
            string name;

            name = input_Name();
            description = input_Description();

            check = give_DiscriptionAboutProducts(name, description, products,
products_Description, productcount);
            if (check)
            {
                successful();
            }
            else
            {
                pressAnyKey();
            }
            continue;
        }
        else if (employee_Option == "7")
        {
            system("cls");
            Print_header(); // agian header will print
            int remove;
            remove = remove_Discount(discount);
            if (remove > discount)
            {
                pressAnyKey();
```

```
            }
            else
            {
                discount = discount - remove;
                successful();
            }
            continue;
        }
        else if (employee_Option == "8")
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;
            name = input_NameForReview();

            check = review_OfProduct(name, products,
Customer_Review_AboutProducts, productcount);
            if (check)
            {
                pressAnyKey();
            }
            else
            {
                pressAnyKey();
            }
            continue;
        }
        else if (employee_Option == "9")
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            string name;
            string complain;

            name = input_UsernameFor_Complain();
            complain = input_Complain();
```

```
                check = give_Complains(name, complain, username,
complains_FromEmployees, usercount);

                if (check)
                {
                    successful();
                }
                else
                {
                    pressAnyKey();
                }
                continue;
            }
            else if (employee_Option == "10")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                bool valid_password;
                bool valid_username;
                bool valid;
                string user;
                string person;
                string old_Password;
                string new_password;

                user = input_Username();
                valid_username = valid_Username(user, username, role, usercount);

                if (valid_username)
                {
                    person = user_Behind_Username(user, username, role, usercount);

                    if (person == "employee")
                    {
                        old_Password = input_OldPassword();

                        new_password = input_NewPassword();
                        valid_password = validPassword(new_password);
```

```
                    if (valid_password)
                    {

                        check = change_Password(user, old_Password, new_password,
username, password, usercount);
                        if (check)
                        {
                            successful();
                        }
                        else
                        {
                            pressAnyKey();
                        }
                        continue;
                    }
                    else
                    {
                        Unsuccessful();
                        continue;
                    }
                }
                else if (person == "customer")
                {
                    new_password = input_NewPassword();
                    valid = validPassword(new_password);
                    if (valid)
                    {

                        check = change_Password_ForAdmin(user, new_password,
username, password, usercount);
                        if (check)
                        {
                            successful();
                        }
                        else
                        {
                            Unsuccessful();
                        }
                        continue;
```

```
                    }
                    else
                    {
                        Unsuccessful();
                    }

                    continue;
                }
            }
            else
            {
                pressAnyKey();
                continue;
            }

        }
        else if (employee_Option == "11")
        {
            break;
        }
        else
        {
            invalid_Input();
            continue;
        }
    }
}
else if (person_role == "customer")
{
    while (true)
    {
        system("cls");
        Print_header(); // agian header will print
        string customer_Option;

        customer_Option = customer_menu();

        if (customer_Option == "1")
        {
```

```
                system("cls");
                Print_header(); // agian header will print

                bool check;
                check = print_Products(products, products_Quantity, products_Price, productcount);
                getch();
                continue;
            }
            else if (customer_Option == "2")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                bool added_toCart;
                int already_Present;
                string name;
                int quantity;

                check = print_Products(products, products_Quantity, products_Price, productcount);
                name = get_ProductName();
                quantity = Get_QuantityForCart();
                if (quantity != 65535)
                {
                    already_Present = already_PresentThenUpdate_Quantity(name, quantity, products, products_Quantity, cart, cart_Quantity, productcount, cart_Index);
                    if (already_Present == 1)
                    {
                        added_toCart = added_InCart(name, quantity, products, products_Quantity, cart, cart_Quantity, cart_Index, productcount);
                        if (added_toCart)
                        {
                            successful();
                        }
                        else
                        {
                            pressAnyKey();
                        }
```

```
            continue;
        }
    }
    else
    {
        pressAnyKey();
    }
    continue;
}
else if (customer_Option == "3")
{
    system("cls");
    Print_header(); // agian header will print
    string option;

    option = get_Opinion_ForDeletion();

    if (option == "1")
    {
        bool check;

        check = Delete_CompleteCart(cart, cart_Quantity, cart_Index);
        if (check)
        {
            successful();
        }
        else
        {
            Unsuccessful();
        }
        continue;
    }
    else if (option == "2")
    {
        bool check;
        bool remove;
        string name;
        check = print_Items_InCart(cart, cart_Quantity, cart_Index);
        if(check)
```

```
                {
                    name = input_Name();

                    remove = Delete_ProductFromCart(name, cart, cart_Quantity,
cart_Index);
                    if (remove)
                    {
                        successful();
                    }
                    else
                    {
                        pressAnyKey();
                    }
                    continue;

                }
                else
                {
                    pressAnyKey();
                }
            }
            continue;
        }
        else if (customer_Option == "4")
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;

            check = print_Items_InCart(cart, cart_Quantity, cart_Index);
            if (check)
            {
                pressAnyKey();
            }
            else
            {
                pressAnyKey();
            }
            continue;
```

```
            }
            else if (customer_Option == "5")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                string name;
                name = input_NameForDescription();
                check = View_DescriptionOfProduct(name, products,
products_Description, productcount);
                if (check)
                {
                    successful();
                }
                else
                {
                    pressAnyKey();
                }
                continue;
            }
            else if (customer_Option == "6")
            {
                system("cls");
                Print_header(); // agian header will print
                bool check;
                string name;
                string review;

                name = input_NameFor_Review();
                review = input_Review();

                check = give_ReviewAboutProducts(name, review, products,
Customer_Review_AboutProducts, productcount);
                if (check)
                {
                    successful();
                }
                else
                {
```

```
                pressAnyKey();
            }
            continue;
        }
        else if (customer_Option == "7")
        {
            system("cls");
            Print_header(); // agian header will print
            int result;

            result = available_Discount(discount);
            pressAnyKey();
            continue;
        }
        else if (customer_Option == "8")
        {
            system("cls");
            Print_header(); // agian header will print
            bool check;
            bool valid_password;
            bool valid_username;
            string user;
            string old_Password;
            string new_password;

            user = input_Username();
            valid_username = valid_Username(user, username, role, usercount);

            if (valid_username)
            {

                old_Password = input_OldPassword();
                valid_username = valid_Username_Customer(user, username, role,
usercount);

                if (valid_username)
                {
                    new_password = input_NewPassword();
                    valid_password = validPassword(new_password);
```

```
                    if (valid_password)
                    {

                        check = change_Password(user, old_Password, new_password,
username, password, usercount);
                        if (check)
                        {
                            successful();
                        }
                        else
                        {
                            pressAnyKey();
                        }
                        continue;
                    }
                    else
                    {
                        pressAnyKey();
                        continue;
                    }
                }
                else
                {
                    Unsuccessful();
                    continue;
                }
            }
            else
            {
                pressAnyKey();
                continue;
            }

        }
        else if (customer_Option == "9")
        {
            system("cls");
            Print_header(); // agian header will print
            float result;
```

```
                bool check;
                string payment_Method;

                payment_Method = payment_option();

                result = checkoutCart(cart, cart_Quantity, products, products_Quantity,
products_Price, cart_Price, cart_Index, productcount);
                check = Validate_Payment_Option(payment_Method);
                if (check)
                {
                   float final_Price;
                   bool invoice;
                   final_Price = Total_Amount(result, discount, payment_Method);

                   invoice = print_Invoice(result, discount, payment_Method,
final_Price);
                   if (invoice)
                   {
                      bool check;

                      check = Delete_CompleteCart(cart, cart_Quantity, cart_Index);
                      printTankYouCapital();

                      getch();
                      continue;
                   }
                   else
                   {
                      Unsuccessful();
                      continue;
                   }
                }
                else
                {
                   invalid_Input();
                   continue;
                }
             }
             else if (customer_Option == "10")
```

```
                {
                    break;
                }
                else
                {
                    invalid_Input();
                    continue;
                }
            }
        }
        else
        {
            getch();
            continue;
        }
    }
    else if (login_value == "2")
    {
        while (true)
        {
            system("cls");
            Print_header();
            bool check;
            bool check_employee;
            bool validate;
            string user_admin, pass_admin, role_admin;

            user_admin = input_Username();
            validate = validUser(user_admin);
            if (validate)
            {
                pass_admin = input_Password();
                validate = validPassword(pass_admin);
                if (validate)
                {
                    role_admin = input_Role();
                    validate = validRolecustomer(role_admin);
                    if (validate)
                    {
```

```
                    check = sign_Up(user_admin, pass_admin, role_admin, username, password, role, usercount);

                    if (check)
                    {
                        usercount++;
                        pressAnyKey();
                        break;
                    }
                }
                else
                {
                    pressAnyKey();
                    break;
                }

            }
            else
            {
                pressAnyKey();
                 break;
            }


        }
        else
        {
            pressAnyKey();
            break;
        }


    }
}
else if (login_value == "3")
{
    print_Thanks();
    break;
}
```

```
        else
        {
            invalid_Input();
            continue;
        }
    }
}

void Print_header()
{
    cout << "\033[1;36m"; // Set text color to cyan
    cout << "                                  ______   ________ __________ ________           ______
________  _      __________ ________         __________ ________ " << endl;
    cout << "\033[1;32m"; // Set text color to green
    cout << "                              (  __  \\ (  ____ )\\__   __/(  ____ )|\\     /|  (  __  \\ (  ____
\\( \\      \\__   __/(  ____ \\|\\     /|\\__   __/(  ____ \\" << endl;
    cout << "\033[1;36m"; // Set text color to cyan
    cout << "                              | (  \\  )| (    ) |   ) (   | (    )|( \\   / )  | (  \\  )| (    \\/| (          ) (
| (    \\/| )   ( |   ) (   | (    \\/" << endl;
    cout << "\033[1;35m"; // Set text color to magenta
    cout << "                              | |   ) || (___) |   | |   | (____)| \\ (_) /   | |   ) || (__    | |          |
|   | |      | (___) |   | |   | (_____" << endl;
    cout << "\033[1;33m"; // Set text color to yellow
    cout << "                              | |   | ||  ___  |   | |   |     __)  \\   /    | |   | ||  __)   | |          | |   |
| ____ |  ___  |   | |   (_____  )" << endl;
    cout << "\033[1;31m"; // Set text color to red
    cout << "                              | |   ) || (   ) |   | |   | (\\ (      ) (     | |   ) || (      | |          | |   | |
\\_  )| (   ) |   | |         ) |" << endl;
    cout << "\033[1;33m"; // Set text color to yellow
    cout << "                              | (__/  )| )   ( |___) (___| ) \\ \\__   | |     | (__/  )| (____/\\|
(____/\\___) (___| (___) || )   ( |   | |   /\\____) |" << endl;
    cout << "\033[1;35m"; // Set text color to magenta
    cout << "                                (_______/ |/      \\|\\_________/|/    \\__/    \\_/      (_______/
(________/(________/\\_________/(________)|/      \\|   )_(    \\_________)" << endl;
    cout << endl;
    cout << "\033[1;36m"; // Set text color to cyan
    cout << "                                                                                    THE DAIRY SHOP"
<< endl;
    cout << "\033[0m"; // Reset text color to default
```

```
    cout << endl;
}
void print_Welcome()
{
    cout << "                    \033[1;35m"; // Set text color to magenta
    cout << "                      __     __      .__                    " << endl;
    cout << "                    \033[1;33m"; // Set text color to yellow
    cout << "                      /  \\    /  \\ ____ | |   ____  ____   _____   ____  " <<
endl;
    cout << "                    \033[1;36m"; // Set text color to cyan
    cout << "                    \\   \\/\\/   // __ \\|  | _/ ___\\/  _ \\ /     \\_/ __ \\ " << endl;
    cout << "                    \033[1;31m"; // Set text color to red
    cout << "                    \\        /\\  ___/|  |_\\  \\___(  <_> )  Y Y  \\  ___/ " << endl;
    cout << "                    \033[1;32m"; // Set text color to green
    cout << "                      \\__/\\  /  \\___  >____/\\___  >____/|__|_|  /\\___  >"
<< endl;
    cout << "                    \033[1;34m"; // Set text color to blue
    cout << "                         \\/       \\/          \\/           \\/     \\/ " << endl;
    cout << "                    \033[0m"; // Reset text color to default
    cout << endl;
}
void printTankYouCapital()
{
    cout << endl;
    cout << "                    \033[1;31m"; // Set text color to red for the title
    cout << "          _________         _______  _       _               _______          "
<< endl;
    Sleep(50);
    cout << "                                  \\__   __/\\     /|(  ___  )( (    /|| \\    /\\  |\\     /|(
___  )|\\     /|";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                                     ) (   | )   ( || (   ) ||  \\  ( ||  \\  / /  ( \\   / )| (   )
|| )   ( |";
    Sleep(50);
    cout << endl;
    Sleep(50);
```

```
    cout << "                                                | |   | (___) || (___) ||   \\ | ||  (_/ /    \\ (_) / | |
| || |   | |";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                                                | |   | ___  ||  ___  || (\\ \\) ||   _ (      \\   /  | |
| || |   | |";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                                                | |   | (   ) || (   ) || | \\   ||  ( \\ \\       ) (   | |   | ||
|   | |";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                                                | |   | )   ( || )   ( || )  \\  ||  /  \\ \\     | |   | (___)
|| (___) |";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                                                   )_(   |/     \\||/     \\||/     )_)|_/    \\/     \\_/
(________)(________)";
    Sleep(50);
    cout << endl;
    Sleep(50);
    cout << "                          \033[0m"; // Reset text color to default
}

void successful()
{
    cout << "Task is completed Successfully" << endl;
    getch();
}
void Unsuccessful()
{
    cout << "Task is not completed Successfully" << endl;
    getch();
}
void print_Thanks()
```

```
{
   cout << "Thank You :)" << endl;
}
string login_page()
{

   string login_value;
   cout<<"\033[;33m";
   cout << "LOGIN PAGE :" << endl;
   cout << endl;
   cout << "1. Sign In " << endl;
   cout << "2. Sign Up " << endl;
   cout << "3. Exit " << endl;
   cout << endl;
   cout << "Enter Your Option here: ";

   getline(cin>>ws, login_value);
   // cin >> login_value;

   return login_value;
   cout<<"\033[0m";

}
string input_Username()
{
   string user;

   cout << "Enter username: ";
   // cin >> user;

   getline(cin>>ws, user);


   return user;
}
string input_Password()
{
   string pass;
   cout << "Enter password: ";
```

```
    // cin >> pass;
    getline(cin>>ws,pass);


    return pass;
}
string input_Role()
{
    string role_admin;

    cout << "Enter Role: ";
    // cin >> role_admin;
    getline(cin>>ws,role_admin);


    return role_admin;
}
string input_NewPassword()
{
    string pass;
    cout << "Enter New Password: ";
    // cin >> pass;
    getline(cin>>ws,pass);

    return pass;
}
bool validUser(string user)
{
    bool containsSpace;

    containsSpace = true;

    for (int index = 0; index < user.length(); index++)
    {
        if (user[index] == ' ')
        {
            containsSpace = false;
            cout<<"User name cannot contain spaces"<<endl;
            break; // exit the loop if a space is found
```

```
        }
    }

    return containsSpace;
}
bool validPassword(string pass)
{
    bool pass_admin;
    if (pass.length() == 4)
    {
        pass_admin = true;
    }
    else
    {
        cout << "Password Must contain  4 letters" << endl;
        pass_admin = false;
    }

    return pass_admin;
}
bool validRolecustomer(string role)
{
    bool Role;

    Role = false;
    if (role == "customer")
    {
        Role = true;
    }
    if (!(Role))
    {
        cout << "You Can only Signup as \"customer\" if you want to sign up as employee
contact your admin" << endl;
    }

    return Role;
}
bool validRole(string role)
{
```

```
    bool Role;

    Role = true;

    return Role;
}
bool sign_Up(string user, string pass, string role, string username[], string password[],
string roleArray[], int count)
{
    bool check = false;

    for (int index = 0; index < count; index++)
    {
        if (user == username[index])
        {
            check = true;
        }
    }

    if (check)
    {
        cout << "Username already exists";
        getch();
    }
    if(!(check))
    {
        if (role == "employee" || role == "customer")
        {
            check = false;
        }
        else
        {
            check = true;
            cout << "Enter a valid role (employee/customer)" << endl;
        }

        if (!check)
        {
            username[count] = user;
```

```
            password[count] = pass;
            roleArray[count] = role;
            count++;
            cout<<"Successfully Signed Up"<<endl;
        }



    }


    return !check;
}
string sign_In(string user, string pass, string username[], string password[], string roleArray[])
{
    int index;
    string result;
    bool check = true;

    for (index = 0; index < 100; index++)
    {
        if (user == username[index] && pass == password[index])
        {
            cout << "Valid" << endl;
            cout << "Press any key to continue";
            getch();
            result = roleArray[index];
            check = false;
        }
    }
    if (check)
    {
        cout << "Invalid username or password" << endl;
        result = "invalid";
    }

    return result;
}
```

```
string admin_menu()
{
   system("cls");
   system("color 09");

   Print_header();

   string option;
   cout << "\033[1;33m";
   cout << "                                                    Admin Menu" <<
endl;
   cout << endl;

   cout << "1. Add New Employee" << endl;
   cout << "2.Remove Employee" << endl;
   cout << "3.Set discount" << endl;
   cout << "4.Remove discount" << endl;
   cout << "5.Update Price" << endl;
   cout << "6.Add More Products" << endl;
   cout << "7.Buy More Stock" << endl;
   cout << "8.remove product" << endl;
   cout << "9.See Complains from Employees" << endl;
   cout << "10.Check review About Product From User" << endl;
   cout << "11.change password" << endl;
   cout << "12. Logout" << endl;
   cout << "Enter Your Option Here:";
   getline(cin>>ws,option);


   return option;
   cout << "\033[0m"; // Reset text color to default
}
bool remove_Employee(string user, string username[], string password[], string
roleArray[], int count)
{
   int user_index, pass_index;
   bool check = false;
```

```
    for (int index = 0; index <= count; index++)
    {
        if (username[index] == user)
        {
            user_index = index;
            check = true;
            break;
        }
    }
    if (check)
    {
        for (int index = user_index; index < count - 2; index++)
        {
            username[index] = username[index + 1];
            password[index] = password[index + 1];
            roleArray[index] = roleArray[index + 1];
        }
    }

    return check;
}
int remove_Discount(int &discount)
{

    int discount_percentage;
    string ans;
    cout << "Right Now Discount offered is :" << discount << " %" << endl;
    cout << "Enter how much discount You Want to Remove : " << endl;
    getline(cin>>ws,ans);

    if (isdigit(ans[0]) && ans[0] != '0') // Check if the input is a valid positive integer
    {
        discount_percentage = stoi(ans);
    }
    else
    {
        cerr << "Error: Invalid input. Please enter a valid positive integer." << endl;
        discount_percentage = 65535;
    }
```

```
    return discount_percentage;
}
int get_percentage_of_Discount()
{
    int discount_percentage;
    string ans;

    cout << "Enter how much flat discount You Want to Offer: " << endl;
    getline(cin>>ws,ans);

    if (isdigit(ans[0]) && ans[0] != '0') // Check if the input is a valid positive integer
    {
        discount_percentage = stoi(ans);
    }
    else
    {
        cerr << "Error: Invalid input. Please enter a valid positive integer." << endl;
        discount_percentage = 65535;
    }

    return discount_percentage;
}
bool make_Employee_Array(string username[], string role[], string employee[])
{
    bool check;

    check = false;
    int employee_index = 0;

    for (int index = 0; index < 100; index++)
    {
        if (role[index] == "employee")
        {
            employee[employee_index] = username[index];
            check = true;
            employee_index++;
        }
    }
```

```
    return check;
}
bool print_Products(string products[], int products_Quantity[], int products_Price[], int
&productcount)
{
    bool check;

    cout << endl;
    cout << "Name:" << endl;
    for (int index = 0; index < productcount; index++)
    {
        cout << index + 1 << ". " << products[index];
        cout << endl;
    }
    cout << endl;

    cout << "Quantity: " << endl;
    for (int index = 0; index < productcount; index++)
    {
        cout << index + 1 << ". " << products_Quantity[index] << " kg.";
        cout << endl;
    }
    cout << endl;

    cout << "Price: " << endl;
    for (int index = 0; index < productcount; index++)
    {
        cout << index + 1 << ". " << products_Price[index] << " Rs.";
        cout << endl;
    }

    check = true;
    return check;
}
string input_Name()
{
    string name;
```

```
    cout << "Enter the Product Name: ";
    getline(cin>>ws,name);


    return name;
}
int input_Productquantity()
{
    int quantity;
    string ans;

    cout << "Enter the Quantity of the Product: ";
    getline(cin>>ws,ans);

    if (isdigit(ans[0]) && ans[0] != '0') // Check if the input is a valid positive integer
    {
        quantity = stoi(ans);
    }
    else
    {
        cerr << "Error: Invalid input. Please enter a valid positive integer." << endl;
        quantity = 65535;
    }

    return quantity;
}
int input_ProductPrice()
{
    int price;
    string ans;

    cout << "Enter the Price of the Product: ";
    getline(cin>>ws,ans);


    if (isdigit(ans[0]) && ans[0] != '0') // Check if the input is a valid positive integer
    {
        price = stoi(ans);
    }
```

```
    else
    {
        cerr << "Error: Invalid input. Please enter a valid positive integer." << endl;
        price = 65535;
    }

    return price;
}
bool add_NewProducts(string name, int quantity, int price, string products[], int products_Quantity[], int products_Price[], int &productcount)
{
    bool check;

    check = true;
    for(int index = 0; index < productcount; index++)
    {
        if(name == products[index])
        {
            check = false;
            cout<<"Product already exist"<<endl;
            break;
        }
    }
    if(check)
    {
        products[productcount] = name;
        products_Quantity[productcount] = quantity;
        products_Price[productcount] = price;
        productcount++;

    }


    return check;
}
bool buy_MoreStock(string name, int quantity, string products[], int products_Quantity[], int products_Price[], int &productcount)
{
    bool check;
```

```
    check = false;

    for (int index = 0; index < productcount; index++)
    {
        if (name == products[index])
        {
            products_Quantity[index] += quantity;
            check = true;
            break;
        }
    }
    if(!(check))
    {
        cout<<"Product Not Fount"<<endl;
    }

    return check;
}
bool remove_Product(string name, string products[], int products_Quantity[], int products_Price[], int &productcount)
{
    bool check;
    int name_index;
    check = false;
    for (int index = 0; index < productcount; index++)
    {
        if (name == products[index])
        {
            name_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        for (int index = name_index; index < productcount - 2; index++)
        {
            products[index] = products[index + 1];
```

```
            products_Quantity[index] = products_Quantity[index + 1];
            products_Price[index] = products_Price[index + 1];
        }
        productcount--;

    }
    if(!(check))
    {
        cout<<"Product Not Found"<<endl;
    }
    return check;
}
bool change_Password_ForAdmin(string user, string new_password, string username[], string password[], int count)
{
    bool check;
    int user_index;
    check = false;
    for (int index = 0; index < count; index++)
    {
        if (user == username[index])
        {
            user_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        password[user_index] = new_password;
    }
    if(!(check))
    {
        cout<<"Username Not Found"<<endl;
    }
    return check;
}

string employee_menu()
```

```
{
   system("cls");
   Print_header();
   string option;
   cout << "\033[1;34m";
   cout << "                                                    Employee Menu" <<
endl;
   cout << endl;
   cout << "1.View products" << endl;
   cout << "2.Add products" << endl;
   cout << "3.Remove products" << endl;
   cout << "4.Update Price" << endl;
   cout << "5.View available discount" << endl;
   cout << "6.Write Description about the Product" << endl;
   cout << "7.Remove Discount" << endl;
   cout << "8.See Review About Product From User" << endl;
   cout << "9.Write complains for the Admin" << endl;
   cout << "10.Change Password" << endl;
   cout << "11.Logout" << endl;
   cout << "Enter Your Option Here: ";

   // cin >> option;
   getline(cin>>ws,option);

   return option;
   cout << "\033[0m"; // Reset text color to default
}
int available_Discount(int discount)
{
   cout << "right Now Available Discount is :" << discount << " %" << endl;
   return 0;
}
bool give_DiscriptionAboutProducts(string name, string discription, string products[],
string product_Description[], int &productcount)
{
   bool check;
   int name_index;
   check = false;
   for (int index = 0; index < productcount; index++)
```

```
    {
        if (name == products[index])
        {
            name_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        product_Description[name_index] = discription;
    }
    if(!(check))
    {
        cout<<"Product Not Found"<<endl;
    }
    return check;
}
string input_Description()
{
    string result;
    cout << "Enter The Description of the product: ";
    getline(cin>>ws,result);


    return result;
}
bool valid_Username(string user, string username[], string role[], int count)
{
    bool check;
    int user_index;
    string result;
    check = false;
    for (int index = 0; index < count; index++)
    {
        if (user == username[index])
        {
            user_index = index;
            check = true;
```

```
            break;
        }
    }
    if(check)
    {
        result = role[user_index];
        if (result == "admin")
        {
            cout<<"Cannot Change password of admin";
            check = false;
            return check;
        }
    }
    if(!(check))
    {
        cout<<"Invalid username"<<endl;

    }

    return check;
}

string customer_menu()
{
    system("cls");
    Print_header();
    string option;
    cout << "\033[1;32m";
    cout << "                                                    Customer Menu" <<
endl;
    cout << endl;
    cout << "1.View products" << endl;
    cout << "2.Add to cart" << endl;
    cout << "3.Edit Cart" << endl;
    cout << "4.View Cart" << endl;
    cout << "5.See information about a product" << endl;
    cout << "6.Give Review about a Product" << endl;
    cout << "7.View discounts" << endl;
    cout << "8.Change password" << endl;
```

```
    cout << "9.Checkout" << endl;
    cout << "10. Logout" << endl;
    cout << "Enter Your Option Here:";

    getline(cin>>ws,option);

    return option;
    cout << "\033[0m"; // Reset text color to default
}
string get_ProductName()
{
    string result;
    cout << "Enter the Product Name you Want to buy: ";
    getline(cin>>ws,result);

    return result;
}
bool added_InCart(string name, int quantity, string products[], int products_Quantity[],
string cart[], int cart_Quantity[], int &cart_index, int product_index)
{
    bool check;
    int name_index;
    check = false;
    for (int index = 0; index < product_index; index++)
    {
        if (name == products[index])
        {
            name_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        if (products_Quantity[name_index] >= quantity)
        {
            if (check)
            {
                cart[cart_index] = name;
```

```
                cart_Quantity[cart_index] = quantity;
                cart_index++;
            }
        }
        else
        {
            check = false;
        }
    }
    else
    {
        cout<<"Product Not Found"<<endl;
    }
    return check;
}
string get_Opinion_ForDeletion()
{
    string option;
    cout << "1.Delete The Complete Cart:" << endl;
    cout << "2.Delete Some Items" << endl;
    getline(cin>>ws,option);

    return option;
}
bool Delete_CompleteCart(string cart[], int cart_Quantity[], int &cart_Index)
{
    bool check;
    check = false;
    for (int index = 0; index < cart_Index; index++)
    {
        cart[index] = "nothing";
        cart_Quantity[index] = 0;
        check = true;
    }
    cart_Index = 0;
    return check;
}
int Get_QuantityForCart()
{
```

```
    string input;
    int result;
    cout << "Enter How much You want to Buy:";
    getline(cin>>ws,input);

    if (isdigit(input[0]) && input[0] != '0') // Check if the input is a valid positive integer
    {
        result = stoi(input);
    }
    else
    {
        cout << "Error: Invalid input. Please enter a valid positive integer." << endl;
        result = 65535;
    }
    return result;
}
bool print_Items_InCart(string cart[], int cart_Quantity[], int cart_Index)
{
    bool check;
    check = false;
    for (int index = 0; index < cart_Index; index++)
    {
        cout << "Product Name: " << cart[index] << endl;
        cout << "Product Price: " << cart_Quantity[index] << endl;
        cout << endl;
        check = true;
        if (cart[index] == "noting")
        {
            check = true;
            break;
        }
    }
    if (!(check))
    {
        cout << "Your Cart is Empty:" << endl;
        check = false;
    }

    return check;
```

```
}
bool Delete_ProductFromCart(string name, string cart[], int cart_Quantity[], int &cart_Index)
{
   bool check;
   int name_index;
   check = false;
   for (int index = 0; index < cart_Index; index++)
   {
      if (name == cart[index])
      {
         name_index = index;
         check = true;
         break;
      }
   }
   if(check)
   {
      for (int index = name_index; index < cart_Index - 1; index++)
      {
         cart[index] = cart[index + 1];
         cart_Quantity[index] = cart_Quantity[index + 1];
      }
      if (check)
      {
         cart_Index--;
      }
   }
   else
   {
      cout<<"product Not Fount"<<endl;
   }
   return check;
}
int already_PresentThenUpdate_Quantity(string name, int quality, string products[], int products_Quantity[], string cart[], int cart_Quantity[], int productcount, int cart_Index)
{
   bool check;
   int result;
```

```
int index_Already_PresentItem;
int index_products;
int total_Quantity;

check = false;
result = 1;

for (int index = 0; index < cart_Index; index++)
{
   if (name == cart[index])
   {
      index_Already_PresentItem = index;
      check = true;
      break;
   }
}
if (check)
{
   for (int index = 0; index < productcount; index++)
   {
      if (name == products[index])
      {
         index_products = index;
         break;
      }
   }

   total_Quantity = cart_Quantity[index_Already_PresentItem] + quality;

   if (total_Quantity <= products_Quantity[index_products])
   {
      result = 0;
      cart_Quantity[index_Already_PresentItem] += quality;
   }
   else
   {
      result = 65535;
   }
}
```

```
    return result;
}
bool valid_Username_Customer(string user, string username[], string role[], int count)
{
    bool check;
    int user_index;
    string result;
    check = true;
    for (int index = 0; index < count; index++)
    {
        if (user == username[index])
        {
            user_index = index;
            break;
        }
    }
    result = role[user_index];
    if (result == "admin")
    {
        cout<<"You Cannot change password of an Admin"<<endl;

        check = false;
    }
    if (result == "employee")
    {
        cout<<"You Cannot change password of an employee"<<endl;
        check = false;
    }
    return check;
}
bool View_DescriptionOfProduct(string name, string products[], string
products_Description[], int productcount)
{
    bool check;
    check = false;
    int description_index;
    for (int index = 0; index < productcount; index++)
    {
```

```
        if (name == products[index])
        {
            description_index = index;
            check = true;
            break;
        }
    }
    if (check)
    {
        cout << "The Description of " << name << " is follows: " << endl;
        cout << products_Description[description_index] << endl;
    }
    else
    {
        cout<<"Product Not Fount"<<endl;
    }

    return check;
}
string input_NameForDescription()
{
    string name;

    cout << "Enter the Product Name about which you want to see description: ";
    getline(cin>>ws,name);
    return name;
}
string input_NameFor_Review()
{
    string name;

    cout << "Enter the Product Name about which you want to give Review: ";
    getline(cin>>ws,name);
    return name;
}
string input_Review()
{
    string result;
    cout << "Enter The Review of the product: ";
```

```
    getline(cin>>ws,result);

    return result;
}
bool give_ReviewAboutProducts(string name, string review, string products[], string
Customer_Review_AboutProducts[], int productcount)
{
    bool check;
    int name_index;
    check = false;
    for (int index = 0; index < productcount; index++)
    {
        if (name == products[index])
        {
            name_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        Customer_Review_AboutProducts[name_index] = review;
    }
    else
    {
        cout<<"Product Not Found"<<endl;
    }
    return check;
}
bool change_Password(string user, string old_Password, string new_password, string
username[], string password[], int count)
{
    bool check;
    int user_index;
    check = false;
    for (int index = 0; index < count; index++)
    {
        if (user == username[index] && password[index] == old_Password)
        {
```

```
            user_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        password[user_index] = new_password;
    }
    if(!(check))
    {
        cout<<"Invalid Old Pasword"<<endl;
    }
    return check;
}
float checkoutCart(string cart[], int cart_Quantity[], string products[], int products_Quantity[], int product_price[], float cart_Price[], int &cart_Index, int productcount)
{

    float result = 0;

    for (int i = 0; i < cart_Index; i++)
    {
        for (int loop = 0; loop < productcount; loop++)
        {
            if (products[loop] == cart[i])
            {
                products_Quantity[loop] -= cart_Quantity[i];
                cart_Price[i] = cart_Quantity[i] * product_price[loop];
                result += cart_Price[i];
            }
        }
    }

    return result;
}
string payment_option()
{
```

```
    string option;

    cout << "How Would You Like to Pay (\"Card\" or \"Cash\"): ";
    getline(cin>>ws,option);
    return option;
}
bool Validate_Payment_Option(string payment_method)
{
    bool check;
    check = false;
    if (payment_method == "card" || payment_method == "Card")
    {
        check = true;
    }
    else if (payment_method == "cash" || payment_method == "Cash")
    {
        check = true;
    }

    return check;
}
float Total_Amount(float result, int discount, string payment_Method)
{
    float final_Price;

    if (payment_Method == "card" || payment_Method == "Card")
    {
        final_Price = result - (result * discount / 100);
        final_Price = final_Price + final_Price * 0.05;
    }
    else if (payment_Method == "cash" || payment_Method == "Cash")
    {
        final_Price = result - (result * discount / 100);
        final_Price = final_Price + final_Price * 0.18;
    }

    return final_Price;
}
bool print_Invoice(float result, int discount, string payment_Method, float final_Price)
```

```
{
   bool check;
   system("cls");

   check = true;
   cout << endl;
   cout << endl;
   cout << endl;
   cout << endl;
   cout << endl;
   cout << "                                                    DAIRY - DELIGHTS"
<< endl;
   cout << "                                        Items Total Price:                  "
<< result << " $" << endl;
   cout << "                                                Currently Discount Available
" << discount << " %" << endl;
   cout << "                                      Payment Method                        "
<< payment_Method << endl;
   cout << "                                        Final Price                         "
<< final_Price << " $" << endl;
   cout << endl;
}
string user_Behind_Username(string user, string username[], string role[], int count)
{
   int user_index;
   bool check;
   string result;

   check = false;
   result = "invalid";

   for (int index = 0; index < count; index++)
   {
      if (user == username[index])
      {
         user_index = index;
         check = true;
         break;
```

```
        }
    }
    if(check)
    {
        result = role[user_index];
    }
    else
    {
        cout<<"Invalid Username"<<endl;
    }

    return result;
}
string input_OldPassword()
{
    string pass;
    cout << "Enter Old password: ";
    getline(cin>>ws,pass);
    return pass;
}
string input_NameForReview()
{
    string name;

    cout << "Enter the Product Name about which you want to see Review: ";
    getline(cin>>ws,name);
    return name;
}
bool review_OfProduct(string name, string products[], string
customer_Review_AboutProduct[], int productcount)
{
    bool check;
    check = false;
    int review_index;
    for (int index = 0; index < productcount; index++)
    {
        if (name == products[index])
        {
            review_index = index;
```

```
            check = true;
            break;
        }
    }
    if (check)
    {
        cout << "The Review of " << name << " is follows: " << endl;
        cout << customer_Review_AboutProduct[review_index] << endl;
    }
    if(!(check))
    {
        cout<<"Product Not Found"<<endl;
    }

    return check;
}
string input_UsernameFor_Complain()
{
    string name;

    cout << "Enter the Valid Username for submitting Complain: ";
    getline(cin>>ws,name);
    return name;
}
string input_Complain()
{
    string result;
    cout << "Enter The issues You are facing: ";
    getline(cin>>ws,result);
    return result;
}
bool give_Complains(string user, string complain, string username[], string
complains_FromEmployees[], int usercount)
{
    bool check;
    int name_index;
    check = false;
    for (int index = 0; index < usercount; index++)
    {
```

```
        if (user == username[index])
        {
            name_index = index;
            check = true;
            break;
        }
    }
    if(check)
    {
        complains_FromEmployees[name_index] = complain;
    }
    if(!(check))
    {
        cout<<"Username not Found"<<endl;
    }
    return check;
}
string input_NameToViewComplain()
{
    string name;

    cout << "Enter the username whose complain you want to review ";
    getline(cin>>ws,name);
    return name;
}
bool see_ComplainFromEmployee(string name, string username[], string
complains_FromEmployees[], int usercount)
{
    bool check;
    check = false;
    int complain_index;
    for (int index = 0; index < usercount; index++)
    {
        if (name == username[index])
        {
            complain_index = index;
            check = true;
            break;
        }
```

```
    }
    if(!(check))
    {
        cout<<"No complain Found Againts: "<<name<<endl;
        cout<<"Either Username is wrong or "<<name<<" is not an Employee"<<endl;
    }
    if (check)
    {
        cout << "The Complain of " << name << " is follows: " << endl;
        cout << complains_FromEmployees[complain_index] << endl;
    }

    return check;
}
void pressAnyKey()
{
    cout << endl;
    cout << "Press Any Key To Continue :)" << endl;
    getch();
}
void invalid_Input()
{
    cout << endl;
    cout << "Invalid Input!" << endl;
    cout << "Press Any Key To Continue :)" << endl;
    getch();
}
bool update_PriceOfProduct(string name, int price, string products[], int
products_Price[], int productcount)
{
    bool check;

    check = false;
    int product_index;
    for(int index = 0; index < productcount; index++)
    {
        if(name == products[index])
        {
            check = true;
```

```
            product_index = index;
            break;
        }
    }
    if(check)
    {
        products_Price[product_index] = price;

    }
    if(!(check))
    {
        cout<<"Product not fount"<<endl;
    }


    return check;
}
```

8.Weakness in the Business Application

1.Using parallel arrays.
2. Complex Nesting of loops and ifs statements.
3.Readability is not very good.
4.Commenting is not very good.

9.Future Directions

1. I will make it much more beautiful
2. Code will be more easy to understand.
3. Use different data structures.
4. I will do a lot better commenting.