

**University of Central Punjab**

**BS Computer Sciences**

**Programming Fundamentals**

**Submitted by:**

**Group Members:**

***Afifa Muhammad* L1F22BSCS0478 .**

***Muhammad Abdullah***  **L1F22BSCS0032**

***Saad ur rehman (head)* L1F22BSCS1074**

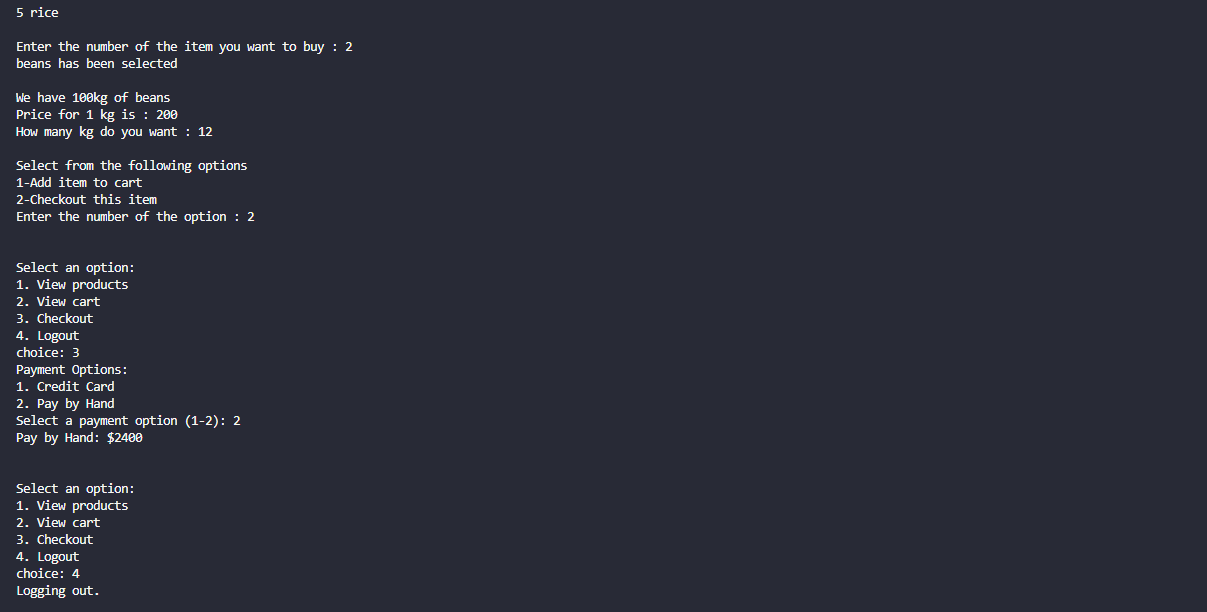
**Submitted to:**

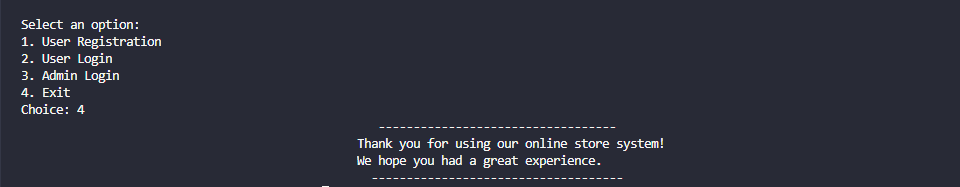
***Sir Hafiz Bilal***

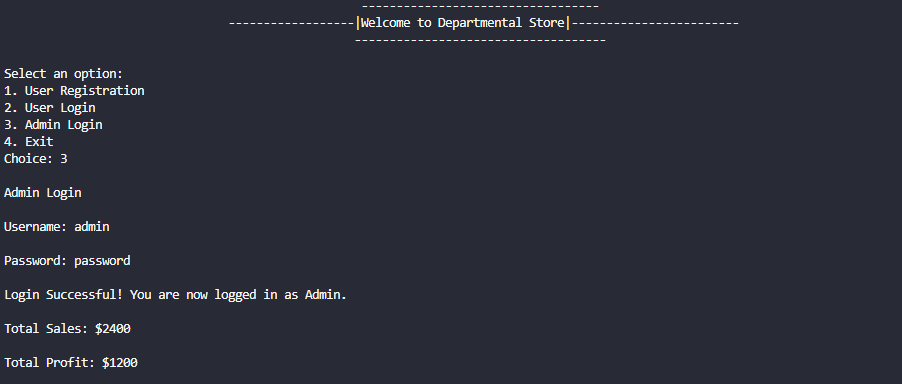
DEPARTMENTAL STORE

Sample run:



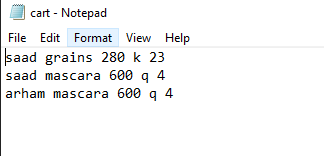


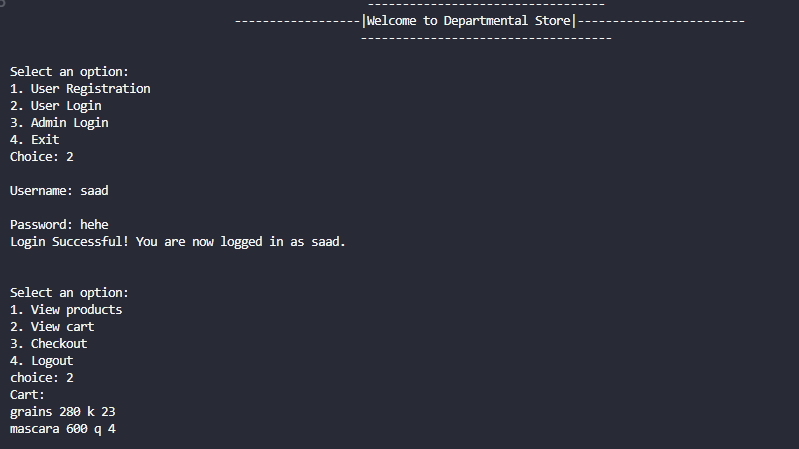


ADMIN:  


CART.txt:

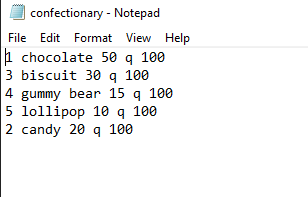
User’s username will be used to filter only those items that he/she added into the cart



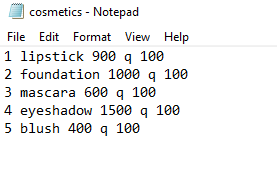


PRODUCT’s FILE

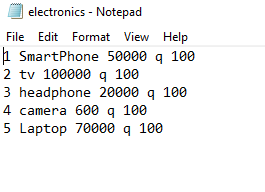
Fetching data and then manipulating it to get the desired results.

confectionart.txt  


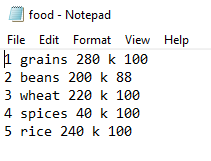
cosmetics.txt



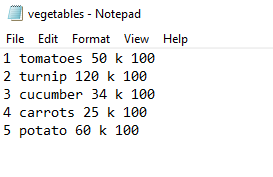
electronics.txt



food.txt

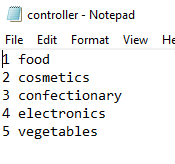


vegetables.txt



controller.txt

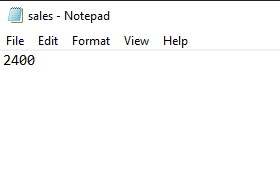
Purpose of this file is to impose the number system instead of having the user to write the whole category.



After user will enter the number of the desired category program will fetch the name after that number and then add “.txt” after it to make it a file name.

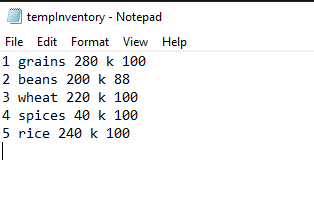
**sales.txt**

sales.txt will get updated when user checkout from the store. This contain reading the previous data from the file and then adding the current user’s total to find the total sales in the departmental store



tempInventory.txt

This file is used to decrease the quantity and to sort the element in each file after a user will buy anything. This mostly contain function like reading the whole data from the file except for the data that a user bought. After copying all the element from the user selected category file it will then paste it in the original user’s selected file. After that sort() function will read the whole data again from user’s selected file and then copy it into tempInventory.txt but in order e.g. 1,2,3,4,5. And then the copying() function will run again. It hold significance in sorting and decrementing the quantity from the original file.



CODE:

#include <iostream>

#include <fstream>

#include <cstring>

using namespace std;

char enteredUsername[50];

char loggedInUser[50];

const int Users = 100;    // Maximum number of user profiles

const int Products = 100; // Maximum number of products

char userProfiles[Users][50]; // Array to store user profiles

char passwords[Users][50];    // Array to store passwords

char products[Products][50];  // Array to store products in the store

double prices[Products];      // Array to store prices

int product\_Count = 0;        // Current number of products in the store

double totalSales = 0.0;

double totalProfit = 0.0;

int total = 0;

int \*price\_int = **new** int; // price of the item

void Message()

{

    cout << "\t\t\t\t          ---------------------------------- " << endl;

    cout << "\t\t\t\t\t\t"

         << "Thank you for using our online store system!" << endl;

    cout << "\t\t\t\t\t\t"

         << "We hope you had a great experience." << endl;

    cout << "\t\t\t\t             ------------------------------------  " << endl;

}

bool processCreditCardPayment(double *totalAmount*)

{

    cout << "Credit Card Payment: $" << *totalAmount* << std::endl;

    int pin;

    cout << "Enter your PIN number: ";

    cin >> pin;

    // Process credit card payment using the entered PIN

    // Add your credit card payment processing logic here

    // Assume the payment is successful for demonstration purposes

    return true;

}

void updating\_sales()

{

*fstream* reading;

    reading.open("sales.txt", *ios*::in);

    reading >> totalSales;

    reading.close();

    totalSales += total;

*fstream* writing;

    writing.open("sales.txt", *ios*::out);

    writing << totalSales;

    writing.close();

}

void checkout()

{

    cout << "Payment Options:" << endl;

    cout << "1. Credit Card" << endl;

    cout << "2. Pay by Hand" << endl;

    int paymentOption;

    cout << "Select a payment option (1-2): ";

    cin >> paymentOption;

    switch (paymentOption)

    {

    case 1:

        // Process credit card payment

        if (processCreditCardPayment(total))

        {

            cout << "Credit card payment successful." << endl;

            updating\_sales();

            // Additional processing after successful payment

        }

        else

        {

            cout << "Credit card payment failed." << endl;

            // Additional processing after failed payment

        }

        break;

    case 2:

        // Pay by hand

        cout << "Pay by Hand: $" << total << endl;

        updating\_sales();

        // Add manual payment processing code here

        break;

    default:

        cout << "Invalid payment option selected." << endl;

        break;

    }

}

void billing(int *qty*, int *kg*, char *name*[], char *unit*)

{

    total = total + \*(price\_int) \* (*qty* + *kg*);

}

int user\_Count = 0; // Current number of user profiles

int size\_of\_array(char *array*[])

{

    int inc = 0;

    while (*array*[inc] != '\0')

    {

        inc++; // FUNCTION FOR ARRAY

    }

    return inc;

}

void ShowCart()

{

*fstream* cart;

    cart.open("cart.txt", *ios*::in);

    char data\_from\_file[100];

    char name\_from\_user[100];

    int j = 0, i = 0;

    int show = 0;

    int size\_of\_name = size\_of\_array(loggedInUser);

    while (cart.getline(data\_from\_file, 100))

    {

        show = 0;

        j = 0;

        i = 0;

        while (data\_from\_file[show] != ' ')

        {

            name\_from\_user[j] = data\_from\_file[show];

            show++;

            j++;

        }

        name\_from\_user[j] = '\0';

        j = 0;

        while (loggedInUser[i] == name\_from\_user[j])

        {

            i++;

            j++;

            if (i == size\_of\_name)

            {

                while (data\_from\_file[show] == ' ')

                {

                    show++;

                }

                while (data\_from\_file[show] != '\0')

                {

                    cout << data\_from\_file[show]; // required data\_from\_file can use to get price all together show the total amount and bill it ty.

                    show++;

                }

                cout << endl;

            }

        }

    }

}

void cart(char *name*[], int *qty*, int *kg*, char *unit*, char *price*[], char *number*)

{

*fstream* cart;

    cart.open("cart.txt", *ios*::app);

    if (*unit* == 'k')

    {

        cart << loggedInUser << " " << *name* << " " << *price* << " " << *unit* << " " << *kg* << endl;

    }

    if (*unit* == 'q')

    {

        cart << loggedInUser << " " << *name* << " " << *price* << " " << *unit* << " " << *qty* << endl;

    }

    cart.close();

}

int record\_counter(char *category*[])

{

*fstream* file;

    int records = 0;

    char data\_from\_file[100];

    file.open(*category*, *ios*::in);

    if (!file.is\_open())

    {

        cout << "Error occured in opening " << *category* << ".txt \n";

    }

    else

    {

        while (file.getline(data\_from\_file, 100))

        {

            if (!(data\_from\_file[0] >= '1' && data\_from\_file[0] <= '9'))

            { // finding records in a file

                break;

            }

            else

            {

                records++;

            }

        }

        return records;

    }

}

void Copying(char *category*[])

{

    char Data\_from\_file[100];

*fstream* Transfer\_Data;

    Transfer\_Data.open("tempInventory.txt", *ios*::in);

    if (!Transfer\_Data.is\_open())

    {

        cout << "Some Error occured in opening tempInventory.txt\n";

    }

    else

    {

*fstream* Writing\_data;

        Writing\_data.open(*category*, *ios*::out);

        while (Transfer\_Data.getline(Data\_from\_file, 100)) // COPY FROM tempInventory to the used category file

        {

            Writing\_data << Data\_from\_file << endl;

        }

        Writing\_data.close();

    }

    Transfer\_Data.close();

}

void sort(char *category*[])

{

*fstream* reading\_records;

    char data\_from\_file[100];

    reading\_records.open(*category*, std::*ios*::in);

    int records = 0;

    records = record\_counter(*category*);

    reading\_records.close();

*fstream* reading;

    reading.open(*category*, *ios*::in);

    if (!reading.is\_open())

    {

        cout << "Error occurred while opening the file for sorting.\n";

    }

    else

    {

        int i = 1;

        bool end = true;

*fstream* clear\_file;

        clear\_file.open("tempInventory.txt", *ios*::out); // clearing tempInventory.txt

        clear\_file.close();

*fstream* writing;

        writing.open("tempInventory.txt", *ios*::app); // appending the sorted data

        for (int it = 1; it <= records + 1; it++)

        {

            reading.clear();

            reading.seekg(0, *ios*::beg);

            bool found = false;

            while (reading.getline(data\_from\_file, 100))

            {

                if (data\_from\_file[0] - '0' == it)

                {

                    found = true;

                    int j = 0;

                    while (data\_from\_file[j] != '\0')

                    { // algorithm to find the data from low to high

                        writing << data\_from\_file[j];

                        j++;

                    }

                    writing << '\n';

                }

            }

        }

        writing.close();

        reading.close();

    }

    Copying(*category*);

}

void sorting\_file(char *array*[], int *minus*, int *size*, char *category*[], char *number*)

{

    int temp = *minus*;

*fstream* reading;

    reading.open(*category*, *ios*::in);

    if (!reading.is\_open())

    {

        cout << "Error opening the " << *category* << ".txt file\n";

    }

    else

    {

*fstream* writing;

        writing.open("tempInventory.txt", *ios*::out);

        if (!writing.is\_open())

        {

            cout << "Some error occurred in opening tempInventory.txt\n";

        }

        else

        {

            char Data\_from\_file[100];

            bool found = false;

            while (reading.getline(Data\_from\_file, 100))

            {

*minus* = temp;

                found = false;

                while (*array*[*minus*] == Data\_from\_file[*minus*] && Data\_from\_file[0] == *number*)

                {

                    if (*minus* == *size* - 1) // writing in temp inventory those items that wasnt selected by user and those items with they new quantity.

                    {

                        found = true;

                        break;

                    }

*minus*++;

                }

                if (!found)

                {

                    writing << Data\_from\_file << endl;

                }

            }

            writing.close();

            reading.close();

        }

    }

    Copying(*category*); // to update our original category files with items with new quantity

    sort(*category*);

}

int Decrementing\_quantity(int *quantity*, char *name*[], char *unit*, int *size*, int *Minus*, char *array*[], char *category*[], int *qty*, int *kg*, char *number*)

{

*fstream* appending\_file;

    appending\_file.open(*category*, *ios*::app);

    if (!appending\_file.is\_open())

    {

        cout << "Some error occured opening" << *category* << ".txt file\n";

    }

    else

    {

        if (*unit* == 'k')

        {

            appending\_file << "\n"

                           << *number* << " " << *name* << " " << \*(price\_int) << " " << *unit* << " " << *quantity* - *kg*;

        }

        if (*unit* == 'q')

        {

            appending\_file << "\n"

                           << *number* << " " << *name* << " " << \*(price\_int) << " " << *unit* << " " << *quantity* - *qty*;

        }

    }

    appending\_file.close();

    sorting\_file(*array*, *Minus*, *size*, *category*, *number*);

    billing(*qty*, *kg*, *name*, *unit*);

}

int Converting\_char\_to\_int(char *price*[], char *quantity*[])

{

    int quantity\_int = 0;

    \*(price\_int) = 0;

    int i = 0;

    while (*quantity*[i] != '\0')

    {

        quantity\_int = quantity\_int \* 10 + (*quantity*[i] - '0'); // formula for converting it

        i++;

    }

    i = 0;

    while (*price*[i] != '\0')

    {

        \*(price\_int) = \*(price\_int)\*10 + (*price*[i] - '0');

        i++;

    }

    return quantity\_int;

}

char Finding\_parameters(char *array*[], char *category*[])

{

    int size, Minus\_from\_size;

    size = size\_of\_array(*array*); // size of array from file

    char price[100] = {};

    char unit;

    char name[100] = {};

    char quantity[100] = {};

    int i = 0, j = 0;

    char number;

    while (*array*[i] != '\0')

    {

        while (*array*[i] != ' ')

        {

            number = *array*[i];

            // finding NUMBERS (1,2,3,4)

            i++;

        }

        while (*array*[i] == ' ')

        {

            i++;

        }

        while (*array*[i] != ' ') // when there are no spaces (name of the item)

        {

            name[j] = *array*[i];

            i++;

            j++;

        }

        j = 0;

        while (*array*[i] == ' ') // ignoring spaces

        {

            i++;

        }

        while (*array*[i] != ' ') // for prices

        {

            price[j] = *array*[i];

            j++;

            i++;

        }

        price[j] = '\0';

        j = 0;

        while (*array*[i] == ' ') // spaces

        {

            i++;

        }

        while (*array*[i] != ' ') // unit

        {

            unit = *array*[i];

            i++;

        }

        while (*array*[i] == ' ') // spaces

        {

            i++;

        }

        Minus\_from\_size = i;

        while (*array*[i] != ' ') // quantity

        {

            if (i == size)

            {

                break;

            }

            else

            {

                quantity[j] = *array*[i];

                j++;

                i++;

            }

        }

        quantity[j] = '\0';

    }

    int kg = 0, qty = 0;

    int option;

    int quantity\_int;

    quantity\_int = Converting\_char\_to\_int(price, quantity); // converting quantity from char to int to decrement it

    if (quantity\_int == 0)

    {

        cout<<endl;

        cout << "Item you are looking for currently is not availible\n"; // if quanitity is zero

        // would add a welcome msg here

    }

    cout << name << " has been selected\n";

    if (unit == 'k')

    {

        cout<<endl;

        cout << "We have " << quantity << "kg of " << name << endl;

        cout << "Price for 1 kg is : " << price << endl;

        cout << "How many kg do you want : ";

        cin >> kg;

        if (kg > quantity\_int)

        {

            cout<<endl;

            cout << "Amount you entered exceeded the current quanitity\nPlease enter an amount within the quantity.\n"; // if entered more than availible quantity

            cout << "How many kg do you want : ";

            cin >> kg;

        }

        else

        {

            cout<<endl;

            cout << "Select from the following options\n";

            cout << "1-Add item to cart\n";

            cout << "2-Checkout this item\n";

            cout << "Enter the number of the option : ";

            cin >> option;

        }

    }

    if (unit == 'q')

    {

        cout<<endl;

        cout << "We have " << quantity << " of " << name << endl;

        cout << "Price for 1 unit is : " << price << endl;

        cout << "How many " << name << " you want : ";

        cin >> qty;

        if (qty > quantity\_int)

        {

            cout<<endl;

            cout << "Amount you entered exceeded the current quanitity\nPlease enter an amount within the quantity.\n"; // if entered more than availible quantity

            cout << "How many kg do you want : ";

            cin >> qty;

        }

        else

        {

            cout<<endl;

            cout << "Select from the following options\n";

            cout << "1-Add item to cart\n";

            cout << "2-Checkout this item\n";

            cout << "Enter the number of the option : ";

            cin >> option;

        }

    }

    if (option == 1)

    {

        cart(name, qty, kg, unit, price, number);

    }

    if (option == 2)

    {

        Decrementing\_quantity(quantity\_int, name, unit, size, Minus\_from\_size, *array*, *category*, qty, kg, number); // appending and decreasing quantity

    }

}

bool finding\_specific\_data(char *array*[], char *name*)

{

    bool found = false;

    int i = 0;

    if (*array*[0] == *name*)

    {

        found = true;

    }

    return found;

}

int Asking\_input()

{

    cout<<endl;

    cout << "Choose one category from following.\n1- food\n2- cosmetics\n3- confectionary\n4- electronics\n5- vegetables\n";

    int show = 0;

    cout<<endl;

    char ask\_category; // incremental variable to show data\_from\_file\_category

    char category[100];

    cout << "Enter the number of the category you want to buy from : ";

    cin >> ask\_category; // for category (food,electronics)

    int in = 0;

*fstream* controller;

    controller.open("controller.txt", *ios*::in); // will find the data according to the number entered by user

    if (!controller.is\_open())

    {

        cout << "Some error occured in opening controller.txt" << endl;

    }

    else

    {

        int c = 0, d = 0;

        bool found\_controller = false;

        char reading\_from\_controller[100];

        while (controller.getline(reading\_from\_controller, 100))

        {

            found\_controller = false;

            if (reading\_from\_controller[0] == ask\_category)

            {

                int size\_of\_controller = size\_of\_array(reading\_from\_controller);

                while (reading\_from\_controller[c] != ' ') // for the number

                {

                    c++;

                }

                while (reading\_from\_controller[c] == ' ') // spaces

                {

                    c++;

                }

                while (reading\_from\_controller[c] != ' ')

                {

                    if (c == size\_of\_controller)

                    {

                        found\_controller = true;

                        break;

                    }

                    else

                    {

                        category[d] = reading\_from\_controller[c]; // adding the name of the file in to the array category array for more info see the controller.txt

                        c++;

                        d++;

                    }

                    category[d] = '\0';

                }

                if (found\_controller)

                {

                    break;

                }

            }

        }

    }

    controller.close();

*fstream* reading\_items;

    char filename[100] = {".txt"};      // for extension

    int size = size\_of\_array(category); // size function which will give category array size which will then use for merge

    while (filename[in] != '\0')

    {

        category[size] = filename[in]; // merging .txt with category array to make it a valid filename

        size++;

        in++;

    }

    category[size] = '\0';

    char data\_from\_file\_category[100];

    reading\_items.open(category, *ios*::in);

    if (!reading\_items.is\_open())

    {

        cout << "Some error occured in opening " << category << ".txt file\n";

    }

    else

    {

        // SHOW THE ITEM IN CATEGORY

        while (reading\_items.getline(data\_from\_file\_category, 100))

        {

            int size\_of\_items = size\_of\_array(data\_from\_file\_category);

            show = 0;

            while (data\_from\_file\_category[show] != ' ')

            {

                cout << data\_from\_file\_category[show]; // show numbers

                show++;

            }

            while (data\_from\_file\_category[show] == ' ')

            {

                cout << data\_from\_file\_category[show]; // show spaces

                show++;

            }

            while (data\_from\_file\_category[show] != ' ' && show < size\_of\_items) // condition to only read names

            {

                cout << data\_from\_file\_category[show];

                show++;

            }

            cout << endl;

        }

    }

    reading\_items.close();

    char array[100] = {};

    char input[100] = {};

    bool found;

    cout<<endl;

    char item\_number;

    cout << "Enter the number of the item you want to buy : ";

    cin >> item\_number; // asking for input within the file

*fstream* file;

    int i = 0, j = 0;

    char data\_from\_file[100] = {};

    file.open(category, *ios*::in);

    if (!file.is\_open())

    {

        cout << "Some error occured opening " << category << ".txt file\n";

    }

    else

    {

        while (file.getline(data\_from\_file, 100))

        {

            found = finding\_specific\_data(data\_from\_file, item\_number); // finding that specific item in file

            if (found)

            {

                Finding\_parameters(data\_from\_file, category); // if found it will find its price,quanity,unit

                break;

            }

            else

            {

                continue;

            }

        }

    }

    file.close();

}

// Function to compare two strings

bool stringCompare(const char \**str1*, const char \**str2*)

{

    int i = 0;

    while (*str1*[i] != '\0' || *str2*[i] != '\0')

    {

        if (*str1*[i] != *str2*[i])

            return false;

        i++;

    }

    return true;

}

// Function to copy one string to another

void stringCopy(char \**destination*, const char \**source*)

{

    int i = 0;

    while (*source*[i] != '\0')

    {

*destination*[i] = *source*[i];

        i++;

    }

*destination*[i] = '\0';

}

// Function to handle user registration

void userRegistration()

{

    char newUser[50];

    char newPassword[50];

cout<<endl;

    cout << "Enter a username: ";

    cin >> newUser;

cout<<endl;

    cout << "Enter a password: ";

    cin >> newPassword;

    // Store the new user profile and password

    stringCopy(userProfiles[user\_Count], newUser);

    stringCopy(passwords[user\_Count], newPassword);

    user\_Count++;

    // Save user data to a file

*ofstream* userFile("users.txt", std::*ios*::app);

    if (userFile.is\_open())

    {

        userFile << newUser << " " << newPassword << std::endl;

        userFile.close();

    }

    cout << "Registration successful!\n";

}

// Function to handle user login

bool userLogin(char *loggedInUser*[50])

{

    char enteredUsername[50];

    char enteredPassword[50];

    cout<<endl;

    cout << "Username: ";

    cin >> enteredUsername;

cout<<endl;

    cout << "Password: ";

    cin >> enteredPassword;

    // Check if the entered credentials match any user profile

    for (int i = 0; i < user\_Count; i++)

    {

        if (stringCompare(userProfiles[i], enteredUsername) && stringCompare(passwords[i], enteredPassword))

        {

            stringCopy(*loggedInUser*, userProfiles[i]);

            return true;

        }

    }

    return false;

}

// Function to handle shopping

void shopping(char *loggedInUser*[50])

{

    char cart[Products][50]; // Shopping cart for the logged-in user

    int cart\_Count = 0;      // Current number of products in the cart

    int choice;

    while (true)

    {

        cout<<endl;

        cout << "\nSelect an option:\n";

        cout << "1. View products\n";

        cout << "2. View cart\n";

        cout << "3. Checkout\n";

        cout << "4. Logout\n";

        cout << "choice: ";

        cin >> choice;

        switch (choice)

        {

        case 1:

            // Display available products (from database or array)

            Asking\_input();

            break;

        case 2:

            // View the cart

            cout << "Cart:\n";

            ShowCart();

            break;

        case 3:

            // Checkout (process the order)

            checkout();

            // Calculate total price, update inventory, generate order confirmation, etc.

            break;

        case 4:

            // Logout

            cout << "Logging out.\n";

            return;

        default:

            cout << "Invalid choice.\n";

            break;

        }

    }

}

// Function to handle admin login

void adminLogin()

{

    char adminUsername[] = "admin";

    char adminPassword[] = "password";

    char enteredPassword[50];

    cout<<endl;

    cout << "Admin Login\n";

    cout<<endl;

    cout << "Username: ";

    cin >> enteredUsername;

    cout<<endl;

    cout << "Password: ";

    cin >> enteredPassword;

    if (stringCompare(adminUsername, enteredUsername) && stringCompare(adminPassword, enteredPassword))

    {

        cout<<endl;

        cout << "Login Successful! You are now logged in as Admin.\n";

*fstream* reading;

        reading.open("sales.txt", *ios*::in);

        reading >> totalSales;

        reading.close();

        totalProfit = totalSales \* 0.5;

        cout<<endl;

        cout << "Total Sales: $" << totalSales << endl;

        cout<<endl;

        cout << "Total Profit: $" << totalProfit << endl;

        // Perform admin tasks (manage sales and profit)

    }

    else

    {

        cout << "Login Failed! Invalid username or password.\n";

    }

}

int main()

{

    cout << "\t\t\t\t          ---------------------------------- " << endl;

    cout << "\t\t\t\t------------------|Welcome to Departmental Store|------------------------ " << endl;

    cout << "\t\t\t\t             ------------------------------------  " << endl;

    // Load user data from a file (if available)

*ifstream* userFile("users.txt");

    if (userFile.is\_open())

    {

        char username[50];

        char password[50];

        while (userFile >> username >> password)

        {

            stringCopy(userProfiles[user\_Count], username);

            stringCopy(passwords[user\_Count], password);

            user\_Count++;

        }

        userFile.close();

    }

    // Load product data from a file (if available)

*ifstream* productFile("products.txt");

    if (productFile.is\_open())

    {

        char product[50];

        double price;

        while (productFile >> product >> price)

        {

            stringCopy(products[product\_Count], product);

            prices[product\_Count] = price;

            product\_Count++;

        }

        productFile.close();

    }

    int option;

    while (true)

    {

        cout << "\nSelect an option:\n";

        cout << "1. User Registration\n";

        cout << "2. User Login\n";

        cout << "3. Admin Login\n";

        cout << "4. Exit\n";

        cout << "Choice: ";

        cin >> option;

        switch (option)

        {

        case 1:

            userRegistration();

            break;

        case 2:

        {

            if (userLogin(loggedInUser))

            {

                cout << "Login Successful! You are now logged in as " << loggedInUser << ".\n";

                shopping(loggedInUser);

            }

            else

            {

                cout << "Login Failed! Invalid username or password.\n";

            }

            break;

        }

        case 3:

            adminLogin();

            break;

        case 4:

            Message();

            return 0;

        default:

            cout << "Invalid choice.\n";

            break;

        }

    }

    return 0;

}