# Garden Fresh Organics

# 

# Session 2023-2027

# Submitted By:

Muhammad Mahad Saffi 2023-CS-59

# Supervised By:

# Dr. Awais Hassan

# Course:

# CSC-102 Program­ming Fundamentals

Department of Computer Science

# University of Engineering and Technology

# Lahore Pakistan

Table of Contents

[Garden Fresh Organics 1](#_Toc153753948)

[1. Introduction 5](#_Toc153753958)

[i. What is Garden Fresh Organics: 5](#_Toc153753959)

[ii. Contribution in CS: 6](#_Toc153753960)

[iii. Advantages: 6](#_Toc153753961)

[2. User Types: 6](#_Toc153753963)

[i. Admin (owner): 6](#_Toc153753964)

[ii. Investor: 7](#_Toc153753965)

[iii. Supplier: 7](#_Toc153753966)

[iv. Employee: 7](#_Toc153753967)

[v. Customer: 7](#_Toc153753968)

[3. Functional Requirements: 7](#_Toc153753969)

[4. Wireframes: 10](#_Toc153754031)

[5. Data Structures (Parallel Arrays): 13](#_Toc153754032)

[6. Function Prototypes: 13](#_Toc153754033)

[7. Function Working Flow: 17](#_Toc153754034)

[8. Complete Code of the Business Application: 18](#_Toc153754035)

[9. Weakness in the Business Application: 115](#_Toc153754036)

[10. Future Directions: 115](#_Toc153754037)

# Introduction

## What is Garden Fresh Organics:

Garden Fresh Organics is an innovative business management system aimed at revolutionizing the way we manage and distribute fresh produce, including fruits, vegetables, and dairy products. This innovative application is designed to usher in a digital era for the trading of natural organic products, simplifying and expediting the processes involved. By leveraging modern technology, our system empowers farmers, suppliers, and retailers to enhance their operations in the realm of fresh produce sales and procurement, facilitating:

* Swift and efficient Transactions
* Real-time monitoring of inventory
* Improved business management and overall productivity.

## Contribution in CS:

The system involves the collection, storage, and analysis of extensive data related to product inventory, pricing, and market trends. This offers opportunities for advancements in database management, data analytics, and data visualization techniques, contributing to the field of data science.

## Advantages:

There are multiple advantages of this program, which we will discuss below:

* Real-Time Information
* Inventory Management
* Market Expansion
* Customer Satisfaction
* Customization
* Supports Local Agriculture
* Security and Privacy



# User Types:

There are multiple users of this application. Every user has its own domain on which he/she can work and access data. For example, an owner of the shop can access data of employees as well as data of products. However, employees can only access their own data and the data of products. In this way, different domains are created for different type of users in this program. Following are the users of this program.

## Admin (owner):

Admin or owner has the overall control of the system. It can add employee and can remove employee. It can set employees salary and can update them with the passage of time. It can see the whole inventory and can remove or add any item at any time if he wants. He can create, modify or delete the accounts of employee as well as of suppliers. He can also see the profile of any supplier and employee.

## Investor:

He is the one who invests in the business. His domain of check and balance is lesser than admin but greater than any other user. He can see the products. He can access information related to the financial performance and growth of the platform. He can also see the profiles of employee and suppliers. He can add or remove any item from list if it is available in inventory. He can update his password.

## Supplier:

He can login to supplier dashboard and can add new items with their prices. He can access to his sales and financial transactions.

## Employee:

He can view items and can take items from inventory to selling list. He can suspend remove or update items from inventory and from list and can view his profile.

## Customer:

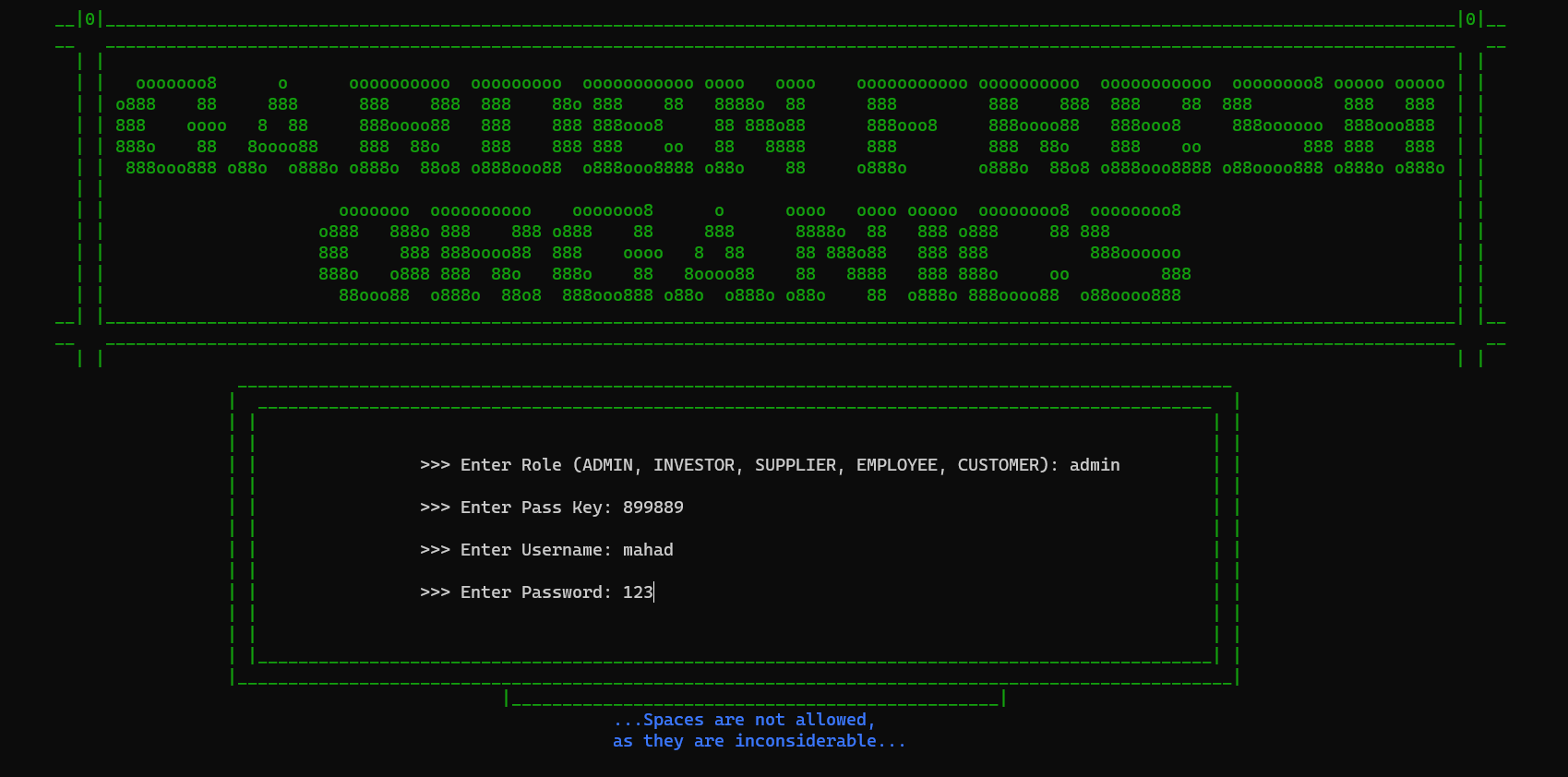
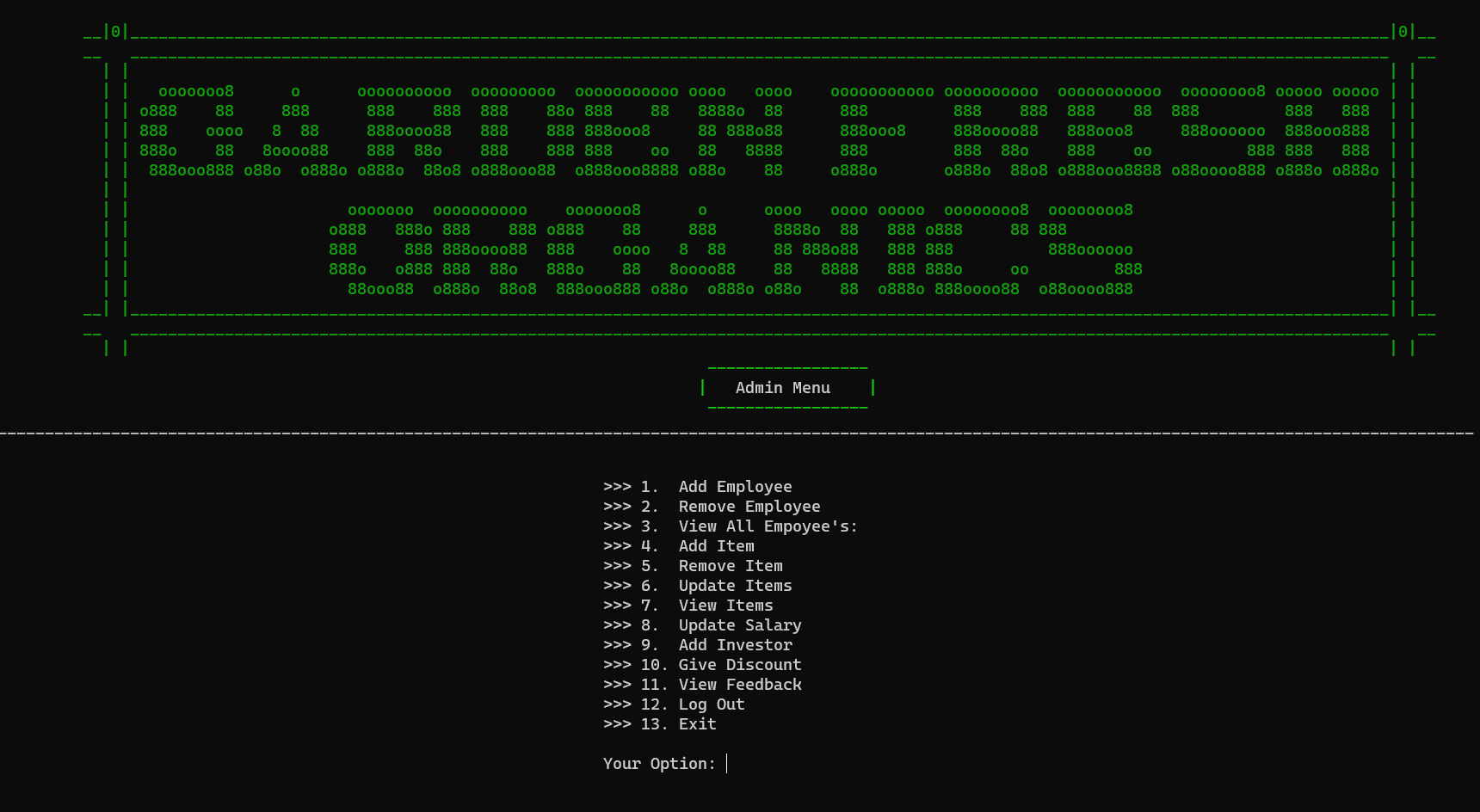
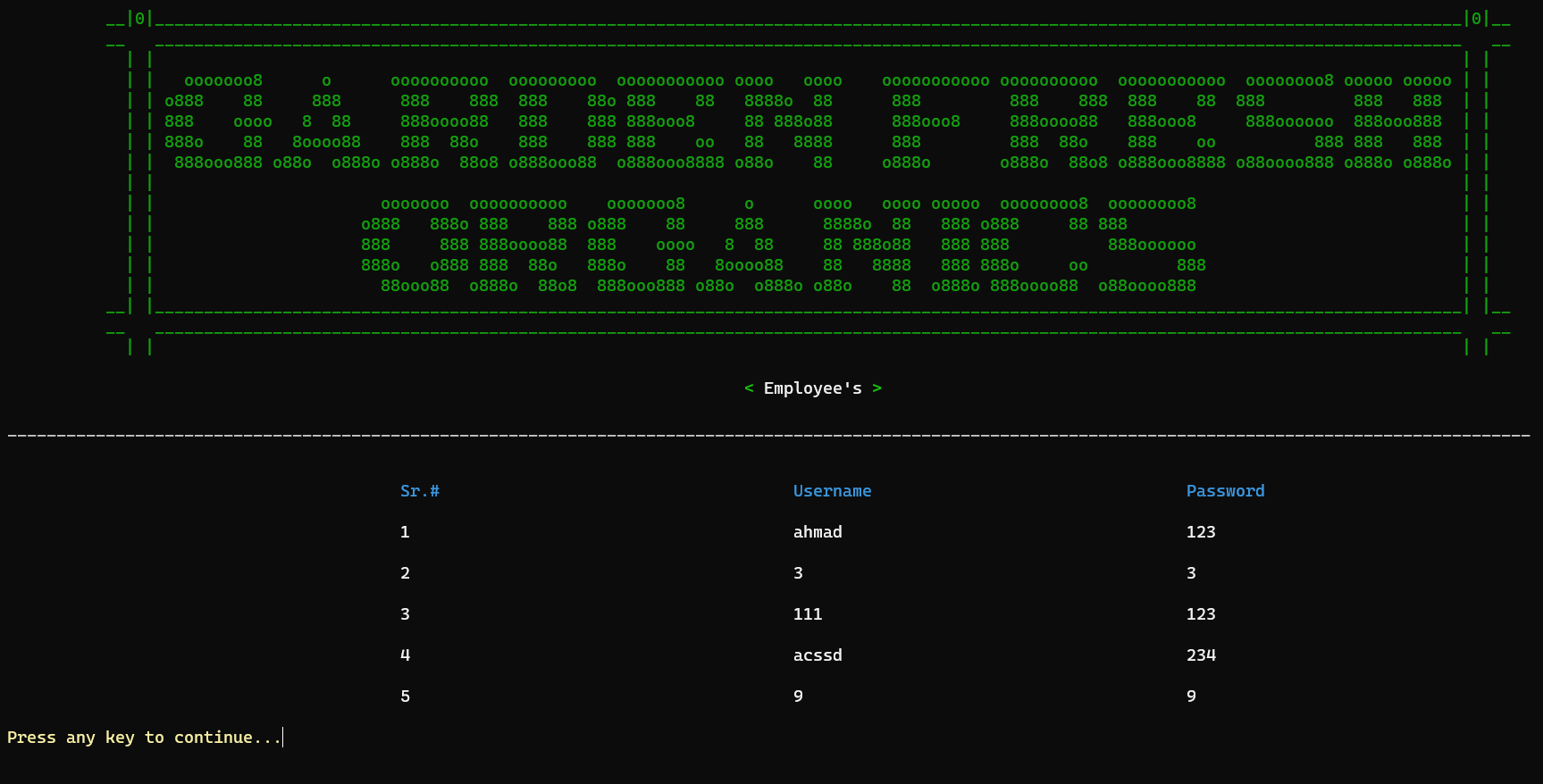
He can register and login and can change his password. He can view items and can add them to cart and in the end, can see discount and purchase them. Also can give feedback.

# Functional Requirements:

|  |  |  |
| --- | --- | --- |
| Users | **Functions** | **Details** |
| Admin | Add employee | He can add employee |
| Remove employee | He can remove employee ID |
| Add and remove items | He can add or remove products |
| Login | He can login to his profile |
| View items for sale | He can view the items which is currently for sale |
| Add or Update salary | He can update or add salary of employee |
| Add investor | He can add an investor and give him specific authority |
| Give discount | He can add discount to items |
| **Investor** | View profile | He can view his profile by logging in |
| View revenue | He can review the growth and profit |
| View employee profile | He can view Employees profile |
| View Profile of suppliers | He can view profile of suppliers |
| View Items | He can view items |
| **Supplier** | View profile | Can view his profile |
| Offer products | Offer his goods |
| Offer discount on products | Give discount on items |
| Check his receiving’s | Can check his balance after selling |
| Withdraw balance | Can withdraw his money |
| View Directories | He can view directories of items |
| **Employee** | View Profile | Can view his profile |
| Login and change password | Login and can change password |
| View inventory | Can view inventory |
| View items for sale | View items in sales list |
| Add or remove items | Can add or remove items |
| Update items | Can update items |
| **Customer** | View profile | He can view his profile |
| Login and change password | He can login his profile |
| View items | He can view items |
| Add items to cart | He can add items to cart |
| View bill | He can view his overall bill |
| Give feedback | He can give feedbacks |
| Log Out | He can log out of his account |

# Wireframes:

Some wireframes of this application are given below.

* First of all we got interface in which we got sign In, sign Up, Existed usernames and exit options.
* We have then Sign Up menu.After sign Up, We have to come to Sign In to go to further options.
* After signing in we come to the user’s profile. It can be admin or investor, supplier or employee or may be customer depending upon the user’s sign in.
* We can view all the employee’s in admin menu
* We can give feedback as a customer which will then admin can see in its menu.



# Data Structures (Parallel Arrays):

string username[100], password[100], role[100], addEmployeeSalary[100], itemName[100], inventoryProduct[100], inventoryPrice[100], inventoryQuantity[100], itemPrice[100], itemQuantity[100], cartName[100], cartPrice[100], cartQuantity[100];

# Function Prototypes:

void addUserToFile(const string &name, const string &password, const string &role, const string &salary);

string getField(string record, int field);

string isPresent(string name, string password, string names[], string passwords[], string role[], int index);

int readIndex(string filename);

void dataTransfer(string username[], string password[], string role[], string addEmployeeSalary[], string itemName[], string itemPrice[], string itemQuantity[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string cartName[], string cartPrice[], string cartQuantity[], int& index, int& itemIndex, int& inventoryIndex, int& cartIndex, int& addEmpIndex);

bool containsOnlyAlphabets(string);

bool containsOnlyIntegers(string);

int convertToInteger(string);

string removeSpaces(const string &input);

string getInput(string variable);

void viewProfile(int &index, string &profile, string username[], string password[], string role[]);

void header(); void fancyHeader(); void signHeader(); void signUpHeader(); void head();

void menuHeader(string menu); void subMenuHeader(string subMenu); void cursorHide(); void cursorShow(); void gotoxy(int x, int y);

void resizeConsole(); void setColor(int color); void resetColor(); string signMenu();

bool signUp(string Name, string Pass, string Ro, string username[], string password[], string role[], int index);

string signIn(int &index, string &profile, string username[], string password[], string role[]);

void existedUsernames(string username[]);

// User Menu's

int adminMenu(); int investorMenu(); int supplierMenu(); int employeeMenu(); int customerMenu();

// Admin's Options

void addEmp(int &index, int &addEmpIndex, string username[], string password[], string role[], string addEmployeeSalary[]);

bool checkRole(string Ro);void removeEmp(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[]);

void viewEmployees(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[]);

void addItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void removeItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void updateItem(int &index, int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void viewItems(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void addOrUpdateSalary(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[]);

void addInvestor(int &index, string username[], string password[], string role[]);

void discount(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void viewFeedback(); void logout();

// Investor's Options

void changePass(int &index, string &profile, string username[], string password[]);

void viewRevenue(int &profit);

void SuppliersProfiles(int &index, int &viewSerial, string username[], string password[], string role[]);

// Supplier's Options

void addProductsToInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

void offerDiscount(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

void checkReceiving(int &inventoryIndex, int &balance, int supplierBalance, string inventoryPrice[], string inventoryQuantity[]);

void withDrawBalance(int &balance, int& supplierBalance);

void viewInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

// Employee Functions

void inventoryToSale(int &index, int &inventoryIndex, int &profit, int &itemIndex, string itemName[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string itemPrice[], string itemQuantity[]);

void viewEmployeeProfile(int &index, string &profile, string username[], string password[], string role[], string addEmployeeSalary[]);

// Customer Functions

void itemsToCart(int &itemIndex, int &cartIndex, string itemName[], string itemPrice[], string itemQuantity[], string cartName[], string cartPrice[], string cartQuantity[]);

void viewBill(int &cartIndex, string cartName[], string cartPrice[], string cartQuantity[]);

void giveFeedback(int &excellent, int &good, int &bad, int &worst);

// Invalid

void invalidUser();void invalidOption();void invalidCredential();void invalidSerialNo();

// Press Key

void pressKey();

## Function Working Flow:

­­

# Complete Code of the Business Application:

#include <iostream>

#include <fstream>

#include <windows.h>

#include <conio.h>

#include <string>

#include <cctype>

using namespace std;

void addUserToFile(const string &name, const string &password, const string &role, const string &salary);

string getField(string record, int field);

string isPresent(string name, string password, string names[], string passwords[], string role[], int index);

int readIndex(string filename);

void dataTransfer(string username[], string password[], string role[], string addEmployeeSalary[], string itemName[], string itemPrice[], string itemQuantity[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string cartName[], string cartPrice[], string cartQuantity[], int& index, int& itemIndex, int& inventoryIndex, int& cartIndex, int& addEmpIndex);

bool containsOnlyAlphabets(string);

bool containsOnlyIntegers(string);

int convertToInteger(string);

string removeSpaces(const string &input);

string getInput(string variable);

void viewProfile(int &index, string &profile, string username[], string password[], string role[]);

void header(); void fancyHeader(); void signHeader(); void signUpHeader(); void head();

void menuHeader(string menu); void subMenuHeader(string subMenu); void cursorHide(); void cursorShow(); void gotoxy(int x, int y);

void resizeConsole(); void setColor(int color); void resetColor(); string signMenu();

bool signUp(string Name, string Pass, string Ro, string username[], string password[], string role[], int index);

string signIn(int &index, string &profile, string username[], string password[], string role[]);

void existedUsernames(string username[]);

// User Menu's

int adminMenu();

int investorMenu();

int supplierMenu();

int employeeMenu();

int customerMenu();

// Admin's Options

void addEmp(int &index, int &addEmpIndex, string username[], string password[], string role[], string addEmployeeSalary[]);

bool checkRole(string Ro);void removeEmp(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[]);

void viewEmployees(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[]);

void addItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void removeItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void updateItem(int &index, int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void viewItems(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void addOrUpdateSalary(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[]);

void addInvestor(int &index, string username[], string password[], string role[]);

void discount(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[]);

void viewFeedback();void logout();

// Investor's Options

void changePass(int &index, string &profile, string username[], string password[]);

void viewRevenue(int &profit);

void SuppliersProfiles(int &index, int &viewSerial, string username[], string password[], string role[]);

// Supplier's Options

void addProductsToInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

void offerDiscount(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

void checkReceiving(int &inventoryIndex, int &balance, int supplierBalance, string inventoryPrice[], string inventoryQuantity[]);

void withDrawBalance(int &balance, int& supplierBalance);

void viewInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[]);

// Employee Functions

void inventoryToSale(int &index, int &inventoryIndex, int &profit, int &itemIndex, string itemName[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string itemPrice[], string itemQuantity[]);

void viewEmployeeProfile(int &index, string &profile, string username[], string password[], string role[], string addEmployeeSalary[]);

// Customer Functions

void itemsToCart(int &itemIndex, int &cartIndex, string itemName[], string itemPrice[], string itemQuantity[], string cartName[], string cartPrice[], string cartQuantity[]);

void viewBill(int &cartIndex, string cartName[], string cartPrice[], string cartQuantity[]);

void giveFeedback(int &excellent, int &good, int &bad, int &worst);

// Invalid

void invalidUser();void invalidOption();void invalidCredential();void invalidSerialNo();

// Press Key

void pressKey();

// Secret Key

string secretKey = "899889";

// Indexes

int x1 = 10, x2 = 20, x3 = 30, x4 = 40, x5 = 50, x6 = 60, x7 = 70, x8 = 80;

int y1 = 10, y2 = 20, y3 = 30, y4 = 40, y5 = 50, y6 = 60, y7 = 70;

int main()

{

resizeConsole();

int index = 0, addEmpIndex = 0, itemIndex = 0, inventoryIndex = 0, cartIndex = 0, viewSerial = 1, employeeSerial = 0;

string profile = "", secretKey = "899889";

int supplierBalance = 0, balance = 0, profit = 0;

string username[100], password[100], role[100], addEmployeeSalary[100], itemName[100], inventoryProduct[100], inventoryPrice[100], inventoryQuantity[100], itemPrice[100], itemQuantity[100], cartName[100], cartPrice[100], cartQuantity[100];

int excellent = 0, good = 0, bad = 0, worst = 0;

cursorHide(); fancyHeader();

while (true)

{

string Name, Pass, Ro, salary; bool exitLoop = false;

dataTransfer(username, password, role, addEmployeeSalary, itemName, itemPrice, itemQuantity, inventoryProduct, inventoryPrice, inventoryQuantity, cartName, cartPrice, cartQuantity, index, itemIndex, inventoryIndex, cartIndex, addEmpIndex);

string sign = signMenu();

if (sign == "1")

{

string passkey; signUpHeader(); gotoxy(46, 22); cout << ">>> Enter Role (ADMIN, INVESTOR, SUPPLIER, EMPLOYEE, CUSTOMER): "; Ro = getInput(Ro);

for (int i = 0; i < Ro.length(); i++) Ro[i] = toupper(Ro[i]);

if (checkRole(Ro))

{

if (Ro == "ADMIN" || Ro == "INVESTOR")

{

gotoxy(46, 24); cout << ">>> Enter Pass Key: "; passkey = getInput(passkey);

if (passkey == secretKey)

{

gotoxy(46, 26); cout << ">>> Enter Username: "; Name = getInput(Name);

gotoxy(46, 28); cout << ">>> Enter Password: "; Pass = getInput(Pass);

bool answer = signUp(Name, Pass, Ro, username, password, role, index);

if (answer == true)

{

username[index] = Name; password[index] = Pass; role[index] = Ro; string salary = "";

addUserToFile(Name, Pass, Ro, salary); index++;

gotoxy(46, 30); cout << "\e[0;92mSuccessfully Registered....\e[0m"; Sleep(1000);

}

else

{

gotoxy(46, 30); invalidUser();

}

}

else

{

gotoxy(46, 26); cout << "\e[0;31mInvalid passKey...\e[0m"; Sleep(1000);

}

}

else if (Ro == "EMPLOYEE")

{

gotoxy(46, 24); cout << ">>> Enter Username: "; Name = getInput(Name);

gotoxy(46, 26); cout << ">>> Enter Password: "; Pass = getInput(Pass);

gotoxy(46, 28); cout << ">>> Enter Salary: "; salary = getInput(salary);

bool answer = signUp(Name, Pass, Ro, username, password, role, index);

if (answer == true && containsOnlyIntegers(salary))

{

username[index] = Name; password[index] = Pass; role[index] = Ro; index++;

addUserToFile(Name, Pass, Ro, salary);

gotoxy(46, 30); cout << "\e[0;92mSuccessfully Registered....\e[0m"; Sleep(1000);

}

else

{

gotoxy(46, 28); invalidUser();

}

}

else

{

gotoxy(46, 24); cout << ">>> Enter Username: "; Name = getInput(Name);

gotoxy(46, 26); cout << ">>> Enter Password: "; Pass = getInput(Pass);

bool answer = signUp(Name, Pass, Ro, username, password, role, index);

if (answer == true)

{

username[index] = Name; password[index] = Pass; role[index] = Ro; string salary = ""; index++;

addUserToFile(Name, Pass, Ro, salary);

gotoxy(46, 28); cout << "\e[0;92mSuccessfully Registered....\e[0m"; Sleep(1000);

}

else

{

gotoxy(46, 28); invalidUser();

}

}

}

else

{

gotoxy(46, 24); cout << "\e[0;31mInvalid Role...\e[0m"; Sleep(1000);

}

}

else if (sign == "2")

{

string result = signIn(index, profile, username, password, role);

while (!exitLoop)

{

if (result == "admin")

{

menuHeader(" Admin Menu");

int result2 = adminMenu();

switch (result2)

{

case 1: addEmp(index, addEmpIndex, username, password, role, addEmployeeSalary); break;

case 2: removeEmp(index, viewSerial, employeeSerial, username, password, role, addEmployeeSalary); break;

case 3: viewEmployees(index, viewSerial, employeeSerial, username, password, role); break;

case 4: addItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 5: removeItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 6: updateItem(index, itemIndex, itemName, itemPrice, itemQuantity); break;

case 7: viewItems(itemIndex, itemName, itemPrice, itemQuantity); break;

case 8: addOrUpdateSalary(index, viewSerial, employeeSerial, username, password, role, addEmployeeSalary); break;

case 9: addInvestor(index, username, password, role); break;

case 10: discount(itemIndex, itemName, itemPrice, itemQuantity); break;

case 11: viewFeedback(); break;

case 12: logout(); exitLoop = true; break;

case 13: return 0;

default: gotoxy(x6 + 5, y4); invalidOption();

}

}

else if (result == "investor")

{

menuHeader("Investor Menu");

int result2 = investorMenu();

switch (result2)

{

case 1: viewProfile(index, profile, username, password, role); break;

case 2: changePass(index, profile, username, password); break;

case 3: viewEmployees(index, viewSerial, employeeSerial, username, password, role); break;

case 4: viewRevenue(profit); break;

case 5: SuppliersProfiles(index, viewSerial, username, password, role); break;

case 6: viewItems(itemIndex, itemName, itemPrice, itemQuantity); break;

case 7: addItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 8: removeItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 9: logout(); exitLoop = true; break;

case 10: return 0;

default: invalidOption(); break;

}

}

else if (result == "supplier")

{

menuHeader("Supplier Menu");

int result2 = supplierMenu();

switch (result2)

{

case 1: viewProfile(index, profile, username, password, role); break;

case 2: addProductsToInventory(inventoryIndex, inventoryProduct, inventoryPrice, inventoryQuantity); break;

case 3: offerDiscount(inventoryIndex, inventoryProduct, inventoryPrice, inventoryQuantity); break;

case 4: checkReceiving(inventoryIndex, balance, supplierBalance, inventoryPrice, inventoryQuantity); break;

case 5: withDrawBalance(balance, supplierBalance); break;

case 6: viewItems(itemIndex, itemName, itemPrice, itemQuantity); break;

case 7: logout(); exitLoop = true; break;

case 8: return 0;

default: invalidOption(); break;

}

}

else if (result == "employee")

{

menuHeader("Employee Menu");

int result2 = employeeMenu();

switch (result2)

{

case 1: viewEmployeeProfile(index, profile, username, password, role, addEmployeeSalary); break;

case 2: changePass(index, profile, username, password); break;

case 3: viewInventory(inventoryIndex, inventoryProduct, inventoryPrice, inventoryQuantity); break;

case 4: inventoryToSale(index, inventoryIndex, profit, itemIndex, itemName, inventoryProduct, inventoryPrice, inventoryQuantity, itemPrice, itemQuantity); break;

case 5: viewItems(itemIndex, itemName, itemPrice, itemQuantity); break;

case 6: addItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 7: removeItem(itemIndex, itemName, itemPrice, itemQuantity); break;

case 8: updateItem(index, itemIndex, itemName, itemPrice, itemQuantity); break;

case 9: logout(); exitLoop = true; break;

case 10: return 0;

default: invalidOption(); break;

}

}

else if (result == "customer")

{

menuHeader("Customer Menu");

int result2 = customerMenu();

switch (result2)

{

case 1: viewProfile(index, profile, username, password, role); break;

case 2: changePass(index, profile, username, password); break;

case 3: viewItems(itemIndex, itemName, itemPrice, itemQuantity); break;

case 4: itemsToCart(itemIndex, cartIndex, itemName, itemPrice, itemQuantity, cartName, cartPrice, cartQuantity); break;

case 5: viewBill(cartIndex, cartName, cartPrice, cartQuantity); break;

case 6: giveFeedback(excellent, good, bad, worst); break;

case 7: logout(); exitLoop = true; break;

case 8: return 0;

default: invalidOption(); break;

}

}

else if (result == "invalid")

{

gotoxy(x6 + 6, y2 + 8); invalidUser(); break;

}

}

}

else if (sign == "3")

{

existedUsernames(username);

}

else if (sign == "4")

{

return 0;

}

else

{

gotoxy(59, 29); invalidOption();

}

}

}

// Header's

void header()

{

system("cls");

cout << "\n\e[0;32m";

gotoxy(10, 1);

cout << "\_\_|0|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|0|\_\_" << endl;

gotoxy(10, 2);

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

gotoxy(10, 3);

cout << " | | | | " << endl;

gotoxy(10, 4);

cout << " | | ooooooo8 o oooooooooo ooooooooo ooooooooooo oooo oooo ooooooooooo oooooooooo ooooooooooo oooooooo8 ooooo ooooo | | " << endl;

gotoxy(10, 5);

cout << " | | o888 88 888 888 888 888 88o 888 88 8888o 88 888 888 888 888 88 888 888 888 | | " << endl;

gotoxy(10, 6);

cout << " | | 888 oooo 8 88 888oooo88 888 888 888ooo8 88 888o88 888ooo8 888oooo88 888ooo8 888oooooo 888ooo888 | | " << endl;

gotoxy(10, 7);

cout << " | | 888o 88 8oooo88 888 88o 888 888 888 oo 88 8888 888 888 88o 888 oo 888 888 888 | | " << endl;

gotoxy(10, 8);

cout << " | | 888ooo888 o88o o888o o888o 88o8 o888ooo88 o888ooo8888 o88o 88 o888o o888o 88o8 o888ooo8888 o88oooo888 o888o o888o | | " << endl;

gotoxy(10, 9);

cout << " | | | | " << endl;

gotoxy(10, 10);

cout << " | | ooooooo oooooooooo ooooooo8 o oooo oooo ooooo oooooooo8 oooooooo8 | | " << endl;

gotoxy(10, 11);

cout << " | | o888 888o 888 888 o888 88 888 8888o 88 888 o888 88 888 | | " << endl;

gotoxy(10, 12);

cout << " | | 888 888 888oooo88 888 oooo 8 88 88 888o88 888 888 888oooooo | | " << endl;

gotoxy(10, 13);

cout << " | | 888o o888 888 88o 888o 88 8oooo88 88 8888 888 888o oo 888 | | " << endl;

gotoxy(10, 14);

cout << " | | 88ooo88 o888o 88o8 888ooo888 o88o o888o o88o 88 o888o 888oooo88 o88oooo888 | | " << endl;

gotoxy(10, 15);

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

gotoxy(10, 16);

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

gotoxy(10, 17);

cout << " | | | | " << endl;

cout << "\e[0m";

}

void fancyHeader()

{

system("cls");

cout << "\n\e[0;32m";

gotoxy(10, 1);

Sleep(100);

cout << "\_\_|0|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|0|\_\_" << endl;

gotoxy(10, 2);

Sleep(100);

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

gotoxy(10, 3);

Sleep(100);

cout << " | | | | " << endl;

gotoxy(10, 4);

Sleep(100);

cout << " | | ooooooo8 o oooooooooo ooooooooo ooooooooooo oooo oooo ooooooooooo oooooooooo ooooooooooo oooooooo8 ooooo ooooo | | " << endl;

gotoxy(10, 5);

Sleep(250);

cout << " | | o888 88 888 888 888 888 88o 888 88 8888o 88 888 888 888 888 88 888 888 888 | | " << endl;

gotoxy(10, 6);

Sleep(500);

cout << " | | 888 oooo 8 88 888oooo88 888 888 888ooo8 88 888o88 888ooo8 888oooo88 888ooo8 888oooooo 888ooo888 | | " << endl;

gotoxy(10, 7);

Sleep(200);

cout << " | | 888o 88 8oooo88 888 88o 888 888 888 oo 88 8888 888 888 88o 888 oo 888 888 888 | | " << endl;

gotoxy(10, 8);

Sleep(25);

cout << " | | 888ooo888 o88o o888o o888o 88o8 o888ooo88 o888ooo8888 o88o 88 o888o o888o 88o8 o888ooo8888 o88oooo888 o888o o888o | | " << endl;

gotoxy(10, 9);

Sleep(20);

cout << " | | | | " << endl;

gotoxy(10, 10);

Sleep(10);

cout << " | | ooooooo oooooooooo ooooooo8 o oooo oooo ooooo oooooooo8 oooooooo8 | | " << endl;

gotoxy(10, 11);

Sleep(10);

cout << " | | o888 888o 888 888 o888 88 888 8888o 88 888 o888 88 888 | | " << endl;

gotoxy(10, 12);

Sleep(10);

cout << " | | 888 888 888oooo88 888 oooo 8 88 88 888o88 888 888 888oooooo | | " << endl;

gotoxy(10, 13);

Sleep(10);

cout << " | | 888o o888 888 88o 888o 88 8oooo88 88 8888 888 888o oo 888 | | " << endl;

gotoxy(10, 14);

Sleep(10);

cout << " | | 88ooo88 o888o 88o8 888ooo888 o88o o888o o88o 88 o888o 888oooo88 o88oooo888 | | " << endl;

gotoxy(10, 15);

Sleep(5);

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

gotoxy(10, 16);

Sleep(5);

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

gotoxy(10, 17);

cout << " | | | | " << endl;

cout << "\e[0m";

}

void signHeader()

{

int x = 52;

int y = 17;

gotoxy(x, y = y + 1);

cout << "\e[0;32m \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

gotoxy(x, y = y + 1);

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

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

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

gotoxy(x, y = y + 1);

cout << " |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\e[0m" << endl;

}

void signUpHeader()

{

int x = 25;

int y = 17;

gotoxy(x, y = y + 1);

cout << "\e[0;32m \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

gotoxy(x, y = y + 1);

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

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

cout << " | | | | " << endl;

gotoxy(x, y = y + 1);

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

gotoxy(x, y = y + 1);

cout << " |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\e[0m" << endl;

}

void head()

{

system("cls");

header();

}

void menuHeader(string menu)

{

int x = 65, y = 18;

cursorHide();

head();

cout << "\u001b[32;1m";

// header

gotoxy(x + 10, y);

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

gotoxy(x + 10, y + 1);

cout << "| |" << endl;

gotoxy(x + 10, y + 2);

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

cout << "\u001b[0m";

gotoxy(x + 13, y + 1);

cout << menu;

gotoxy(0, y + 3);

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

gotoxy(0, y + 4);

cout << " ";

cursorShow();

}

void subMenuHeader(string subMenu)

{

int x = 65, y = 18;

cursorHide();

head();

// Sub Menu

gotoxy(x + 10, y + 1);

cout << subMenu;

gotoxy(0, y + 3);

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

gotoxy(0, y + 4);

cout << " ";

cursorShow();

}

void setColor(int color)

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), color);

}

void resetColor()

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 15);

}

void colorReset()

{

cout << "\u001b[0m";

}

void cursorHide()

{

HANDLE hStdOut = NULL;

CONSOLE\_CURSOR\_INFO curInfo;

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleCursorInfo(hStdOut, &curInfo);

curInfo.bVisible = FALSE;

SetConsoleCursorInfo(hStdOut, &curInfo);

}

void cursorShow()

{

HANDLE hStdOut = NULL;

CONSOLE\_CURSOR\_INFO curInfo;

hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleCursorInfo(hStdOut, &curInfo);

curInfo.bVisible = TRUE;

SetConsoleCursorInfo(hStdOut, &curInfo);

}

void gotoxy(int x, int y)

{

COORD coordinates;

coordinates.X = x;

coordinates.Y = y;

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

}

// Function to check valid role

bool checkRole(string Ro)

{

if (Ro == "ADMIN" || Ro == "INVESTOR" || Ro == "SUPPLIER" || Ro == "EMPLOYEE" || Ro == "CUSTOMER")

{ return true; }

else

{ return false; }

}

// Function to view one's profile

void viewProfile(int &index, string &profile, string username[], string password[], string role[])

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Profile \u001b[32;1m>\u001b[0m");

string record; ifstream data("User.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

if (profile == getField(record, 1))

{

gotoxy(70, 26);

cout << "\u001b[37;1m>>> Username: \u001b[0m" << getField(record, 1);

gotoxy(70, 28);

cout << "\u001b[37;1m>>> Password: \u001b[0m" << getField(record, 2);

gotoxy(70, 30);

cout << "\u001b[37;1m>>> Role: \u001b[0m" << getField(record, 3);

}

}

}

else {

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

{

if (username[i] == profile)

{

gotoxy(70, 26);

cout << "\u001b[37;1m>>> Username: \u001b[0m" << username[i];

gotoxy(70, 28);

cout << "\u001b[37;1m>>> Password: \u001b[0m" << password[i];

gotoxy(70, 30);

cout << "\u001b[37;1m>>> Role: \u001b[0m" << role[i];

}

}

} pressKey();

}

// Function for Sign up & Sign In Options menu

string signMenu()

{

string option = "0"; int x = 59, y = 20;

header(); signHeader(); cout << endl; cursorShow();

gotoxy(x6 + 5, 34);

cout << "\u001b[34;1m...Spaces are not allowed,";

gotoxy(x6 + 5, 35);

cout << "as they are inconsiderable...";

colorReset();

gotoxy(x, y = y + 2);

cout << "1. Sign Up: " << endl;

gotoxy(x, y = y + 2);

cout << "2. Sign In: " << endl;

gotoxy(x, y = y + 2);

cout << "3. Existed Usernames" << endl;

gotoxy(x, y = y + 2);

cout << "4. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

option = getInput(option);

return option;

}

// Function to check validity of signUp

bool signUp(string Name, string Pass, string Ro, string username[], string password[], string role[], int index)

{

bool result = true;fstream data;string record;

data.open("User.txt", ios::in);

while (getline(data, record))

{

string userNameFile = getField(record, 1);

if (Name == userNameFile)

{

result = false;

break;

}

}

data.close(); return result;

}

// Functions to check the user by sign In

string signIn(int &index, string &profile, string username[], string password[], string role[])

{

string userName1 = "";string password1 = "";

cursorHide();head();signHeader();cursorShow();

gotoxy(x6 + 5, 34);

cout << "\u001b[34;1m...Spaces are not allowed,";

gotoxy(x6 + 5, 35);

cout << "as they are inconsiderable...";

colorReset();

gotoxy(x6 + 6, y2 + 2);

cout << ">>> Enter Username: ";

userName1 = getInput(userName1);

gotoxy(x6 + 6, y2 + 6);

cout << ">>> Enter Password: ";

password1 = getInput(password1);

string result = isPresent(userName1, password1, username, password, role, index);

profile = userName1;

if (result == "ADMIN")

{ return "admin"; }

else if (result == "INVESTOR")

{ return "investor"; }

else if (result == "SUPPLIER")

{ return "supplier"; }

else if (result == "EMPLOYEE")

{ return "employee"; }

else if (result == "CUSTOMER")

{ return "customer"; }

else

{ return "invalid"; }

}

// Function to display all existed Usernames

void existedUsernames(string username[])

{

subMenuHeader("< Usernames >");

ifstream data("User.txt");

string record; int i = 0;

if (!data.is\_open())

{

for (int i = 0, y = 23; username[i] != ""; i++)

{

gotoxy(75, y + i + 1);

cout << "... " << username[i];

}

}

while (getline(data, record))

{

gotoxy(75, 23 + i);

cout << "... " << getField(record, 1);

i++;

}

pressKey();

}

// Menu's

int adminMenu()

{

string strOption = "";

int x = 65, y = 18;

gotoxy(x, y = y + 6);

cout << ">>> 1. Add Employee" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 2. Remove Employee" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 3. View All Empoyee's: " << endl;

gotoxy(x, y = y + 1);

cout << ">>> 4. Add Item" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 5. Remove Item" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 6. Update Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 7. View Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 8. Update Salary" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 9. Add Investor" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 10. Give Discount" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 11. View Feedback" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 12. Log Out" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 13. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

strOption = getInput(strOption);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

return option;

}

else

{ return -1; }

}

int investorMenu()

{

string strOption = "";

int x = 65, y = 18;

gotoxy(x, y = y + 6);

cout << ">>> 1. View Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 2. Change password" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 3. View Employee Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 4. View Revenue" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 5. View Supplier Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 6. View Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 7. Add Item" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 8. Remove Item" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 9. Log Out" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 10. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

strOption = getInput(strOption);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

return option;

}

else

{ return -1; }

}

int supplierMenu()

{

string strOption = "";

int x = 65, y = 18;

gotoxy(x, y = y + 6);

cout << ">>> 1. View Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 2. Offer Products" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 3. Offer Discount On Products" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 4. Check Receiving" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 5. Withdraw Balance" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 6. View Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 7. Log Out" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 8. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

strOption = getInput(strOption);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

return option;

}

else

{ return -1; }

}

int employeeMenu()

{

string strOption = "";

int x = 65, y = 18;

gotoxy(x, y = y + 6);

cout << ">>> 1. View Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 2. Change Password" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 3. View Inventory" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 4. Move Items From Inventory To For Sale" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 5. View Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 6. Add Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 7. Remove Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 8. Update Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 9. Log Out" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 10. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

strOption = getInput(strOption);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

return option;

}

else

{ return -1; }

}

int customerMenu()

{

string strOption = "";

int x = 65, y = 18;

gotoxy(x, y = y + 6);

cout << ">>> 1. View Profile" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 2. Change Password" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 3. View Items" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 4. Add Items To Cart" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 5. View Bill" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 6. Give Feedback" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 7. Log Out" << endl;

gotoxy(x, y = y + 1);

cout << ">>> 8. Exit" << endl;

gotoxy(x, y = y + 2);

cout << "Your Option: ";

strOption = getInput(strOption);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

return option;

}

else

{ return -1; }

}

// Functions to view invalid option //

void invalidUser()

{ cout << "\e[0;31mInvalid Username or Password...\e[0m"; Sleep(1000); }

void invalidOption()

{ cout << "\e[0;31mInvalid Entry...\e[0m";Sleep(250); }

void invalidCredential()

{ cout << "\e[0;31mInvalid Credentials...\e[0m"; Sleep(500); }

void invalidSerialNo()

{ cout << "\e[0;31mWrong Serial Number...\e[0m"; Sleep(500); }

void pressKey()

{ cout << "\u001b[33;1m\n\nPress any key to continue...\u001b[0m"; getch(); }

// Admin Functions //

// Function to add an employee to the system

void addEmp(int &index, int &addEmpIndex, string username[], string password[], string role[], string addEmployeeSalary[])

{

int x = 60; string tempSalary; string Name, Pass;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Add Employee \u001b[32;1m>\u001b[0m");

gotoxy(x, 25);

cout << ">>> Enter the Username of the employee you want to add: ";

Name = getInput(Name);

gotoxy(x, 26);

cout << ">>> Enter password of the employee you want to add: ";

Pass = getInput(Pass);

bool answer = signUp(Name, Pass, "EMPLOYEE", username, password, role, index);

if (answer == true)

{

// Add the following lines to input the salary

gotoxy(x, 27);

cout << ">>> Enter the salary of the employee: ";

tempSalary = getInput(tempSalary);

if (containsOnlyIntegers(tempSalary))

{

username[index] = Name;

password[index] = Pass;

role[index] = "EMPLOYEE";

addEmployeeSalary[index] = tempSalary;

index++;

// File Handling

addUserToFile(Name, Pass, "EMPLOYEE", tempSalary);

cout << "\n\t\t\t\t\t\t\t\t\e[0;92mEmployee added successfully...\e[0m";

Sleep(1000);

}

else

{ gotoxy(x, 28); invalidUser(); }

}

}

// Function to remove an employee from the system

void removeEmp(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[])

{

string strSerial = "";

viewEmployees(index, viewSerial, employeeSerial, username, password, role);

cout << "\n\n\n\t\t\t\t\t>>> Enter the serial No. of Employee: ";

strSerial = getInput(strSerial);

if (containsOnlyIntegers(strSerial))

{

int serial = convertToInteger(strSerial);

if (serial > 0 && serial <= employeeSerial)

{

// File Handling

int count = 0;

string record;

ifstream data("User.txt");

ofstream temp("Temp.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

if (getField(record, 3) == "EMPLOYEE")

{

count++;

if (serial == count)

{

continue;

}

}

temp << record << endl;

}

data.close(); temp.close();

remove("User.txt"); rename("Temp.txt", "User.txt");

}

else {

int employeeIndex = -1; // Variable to store the index of the employee to be removed

int count = 0; // Variable to count employees with the correct role

// Find the index of the employee with the specified serial number and role

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

{

if (role[i] == "EMPLOYEE")

{

count++;

if (count == serial)

{

employeeIndex = i; break;

}

}

}

if (employeeIndex != -1)

{

// Shift the elements to remove the employee

for (int i = employeeIndex; i < index - 2; i++)

{

username[i] = username[i + 1];

password[i] = password[i + 1];

role[i] = role[i + 1];

addEmployeeSalary[i] = addEmployeeSalary[i + 1];

}

index--;

cout << "\n\t\t\t\t\t\e[0;92mEmployee removed successfully...\e[0m";

Sleep(1000);

}

else

{ invalidSerialNo(); }

}

}

else

{ invalidSerialNo(); }

}

else

{ invalidCredential(); }

}

// Function to view employees in the system

void viewEmployees(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[])

{

int y = 24;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Employee's \u001b[32;1m>\u001b[0m");

setColor(3);

gotoxy(40, 24); cout << "Sr.#"; gotoxy(80, 24); cout << "Username";

gotoxy(120, 24); cout << "Password"; resetColor();

// File Handling

string record; ifstream data("User.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

if (getField(record, 3) == "EMPLOYEE")

{

gotoxy(40, y + 2);

cout << viewSerial;

gotoxy(80, y + 2);

cout << getField(record, 1);

gotoxy(120, y + 2);

cout << getField(record, 2);

viewSerial++;

y += 2;

}

}

// Close the file

data.close();

}

else { //If file does not opens

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

{

if (role[i] == "EMPLOYEE")

{

gotoxy(40, y + 2);

cout << viewSerial;

gotoxy(80, y + 2);

cout << username[i];

gotoxy(120, y + 2);

cout << password[i];

viewSerial++;

y += 2;

}

}

}

employeeSerial = viewSerial;

viewSerial = 1;

pressKey();

}

// Function to add an item to the system

void addItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[])

{

string Name, Price, Quantity;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Add Items \u001b[32;1m>\u001b[0m");

gotoxy(0, 25);

cout << "\t\t\t\t\t>>> Enter the name of the item: ";

Name = getInput(Name);

cout << "\t\t\t\t\t>>> Enter the price of the item: ";

Price = getInput(Price);

cout << "\t\t\t\t\t>>> Enter the quantity of the item: ";

Quantity = getInput(Quantity);

if (containsOnlyIntegers(Price) && containsOnlyIntegers(Quantity))

{

// File Handling

ofstream data("Items.txt", ios::app);

data << Name << "," << Price << "," << Quantity << "," << "" << endl;

data.close();

itemIndex++;

cout << "\t\t\t\t\t\e[0;92mItem Entered successfully...\e[0m";

Sleep(1000);

}

else

{

gotoxy(40, 29);

invalidCredential();

}

}

// Function to remove an item from the system

void removeItem(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[])

{

string strSr = "";

viewItems(itemIndex, itemName, itemPrice, itemQuantity);

cout << "\n\n\n\t\t\t\t\t>>> Enter the Serial Number of the item you want to delete: ";

strSr = getInput(strSr);

if (containsOnlyIntegers(strSr))

{

int sr = convertToInteger(strSr);

itemIndex = readIndex("Items.txt"); //Recount the item index when program starts

if (sr <= itemIndex && sr > 0)

{

// File Handling

int count = 0;

string record;

ifstream data("Items.txt");

ofstream temp("Temp.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

count++;

if (sr == count)

{ continue; }

temp << record << endl;

}

data.close(); temp.close();

remove("Items.txt"); rename("Temp.txt", "Items.txt");

}

else { //If file does not exist

for (int i = 0; i < itemIndex - 1; i++)

{

if (sr - 1 == i)

{

itemName[i] = itemName[i + 1];

itemPrice[i] = itemPrice[i + 1];

itemQuantity[i] = itemQuantity[i + 1];

cout << "\n\t\t\t\t\t\e[0;92mItem Removed Successfully...\e[0m";

Sleep(500);

break;

}

}

itemIndex--;

}

}

else

{ invalidSerialNo(); }

}

else

{ invalidCredential(); }

}

// Function to update information about an item in the system

void updateItem(int &index, int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[])

{

string sr, price, quantity, name;

viewItems(itemIndex, itemName, itemPrice, itemQuantity);

cout << "\n\t\t\t\t\t>>> Enter the Serial Number of the item you want to update: ";

sr = getInput(sr);

cout << "\n\t\t\t\t\t>>> Enter New Name: ";

name = getInput(sr);

cout << "\n\t\t\t\t\t>>> Enter New Price: ";

price = getInput(price);

cout << "\n\t\t\t\t\t>>> Enter New Quantity: ";

quantity = getInput(quantity);

if (containsOnlyIntegers(sr) && containsOnlyIntegers(price) && containsOnlyIntegers(quantity))

{

int intSr = convertToInteger(sr), intPrice = convertToInteger(price), intQuantity = convertToInteger(quantity);

if (intSr <= itemIndex && intSr > 0)

{

int count = 0;

string record;

ifstream data("Items.txt"); ofstream temp("Temp.txt");

bool fileExists = data.good();

if(fileExists)

{

while (getline(data, record))

{

count++; //Counter to check serial Number

if (intSr == count)

{

temp << name << "," << price << "," << quantity << "," << "" << endl;

cout << "\e[0;92mItem Updated Successfully...\e[0m\n";

Sleep(500);

continue;

}

temp << record << endl;

}

data.close(); temp.close();

remove("Items.txt"); rename("Temp.txt", "Items.txt");

}

else {

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

{

if (intSr - 1 == i)

{

itemName[i] = name;

itemPrice[i] = price;

itemQuantity[i] = quantity;

cout << "\e[0;92mItem Updated Successfully...\e[0m\n"; Sleep(500);

}

}

}

}

else

{ invalidSerialNo(); }

}

else

{ invalidCredential(); }

}

// Function to view items in the system

void viewItems(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[])

{

int y = 23, i = 1;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Items \u001b[32;1m>\u001b[0m");

gotoxy(30, y);

cout << "Sr.#";

gotoxy(60, y);

cout << "Name"; gotoxy(90, y); cout << "Price";

gotoxy(120, y); cout << "Quantity";

// File Handling

string record; ifstream data("Items.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

gotoxy(30, y + 2);

cout << i;

gotoxy(60, y + 2);

cout << getField(record, 1);

gotoxy(90, y + 2);

cout << getField(record, 2);

gotoxy(120, y + 2);

cout << getField(record, 3);

y += 2;

i++;

}

}

else {

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

{

gotoxy(30, y + 2);

cout << i + 1;

gotoxy(60, y + 2);

cout << itemName[i];

gotoxy(90, y + 2);

cout << itemPrice[i];

gotoxy(120, y + 2);

cout << itemQuantity[i];

y += 2;

}

}

pressKey();

}

// Function to add or update the salary of an employee

void addOrUpdateSalary(int &index, int &viewSerial, int &employeeSerial, string username[], string password[], string role[], string addEmployeeSalary[])

{

string name = "", salary = "";

viewEmployees(index, viewSerial, employeeSerial, username, password, role);

cout << "\n\t\t\t\t\t>>> Enter the name of the employee: ";

name = getInput(name);

cout << "\n\t\t\t\t\t>>> Enter the amount of salary: ";

salary = getInput(salary);

if (containsOnlyAlphabets(name))

{

if (containsOnlyIntegers(salary))

{

string record; ifstream data("User.txt"); ofstream temp("Temp.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

if (getField(record, 3) == "EMPLOYEE" && getField(record, 1) == name)

{

temp << getField(record, 1) << "," << getField(record, 2) << "," << "EMPLOYEE" << "," << salary << endl;

cout << "\e[0;92mSalary Updated Successfully...\e[0m\n";Sleep(500); continue;

}

temp << record << endl;

}

data.close(); temp.close(); //Close both files

remove("User.txt"); rename("Temp.txt", "User.txt"); //Remove previous and rename temp file

}

else {

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

{

if (name == username[i])

{

addEmployeeSalary[i] = salary;

cout << "\e[0;92mSalary Updated Successfully...\e[0m\n"; Sleep(500);

}

}

}

}

else

{ invalidCredential(); }

}

else

{ invalidCredential(); }

}

// Function to add an investor to the system

void addInvestor(int &index, string username[], string password[], string role[])

{

string name, pass, passKey;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Add Investor \u001b[32;1m>\u001b[0m");

gotoxy(70, 23);

cout << ">>> Enter pass Key: ";

passKey = getInput(passKey);

if (passKey == secretKey)

{

cout << "\n\t\t\t\t\t>>> Enter the username of the investor: ";

name = getInput(name);

cout << "\t\t\t\t\t>>> Enter the Password of the investor: ";

pass = getInput(pass);

bool answer = signUp(name, pass, "INVESTOR", username, password, role, index);

if (answer == true)

{

username[index] = name;

password[index] = pass;

role[index] = "INVESTOR";

string salary = "";

index++;

// File Handling

addUserToFile(name, pass, "INVESTOR", salary);

cout << "\t\t\t\t\t\e[0;92mInvestor is successfully added...\e[0m"; Sleep(1000);

}

else

{ invalidUser(); }

}

else

{ cout << "\n\t\t\t\t\tInvalid Pass Key..."; Sleep(500); }

}

// Function to apply a discount to an item by admin

void discount(int &itemIndex, string itemName[], string itemPrice[], string itemQuantity[])

{

string strSr = ""; string strDiscount = "";

viewItems(itemIndex, itemName, itemPrice, itemQuantity);

cout << "\n\t\t\t\t\t>>> Enter the serial No. of the product: ";

strSr = getInput(strSr);

cout << "\t\t\t\t\t>>> Enter the percentage(%) of discount: ";

strDiscount = getInput(strDiscount);

if (containsOnlyIntegers(strSr) && containsOnlyIntegers(strDiscount))

{

int sr = convertToInteger(strSr), discount = convertToInteger(strDiscount);

itemIndex = readIndex("Items.txt");

if (sr <= itemIndex && sr > 0)

{

int count = 0; string record; ifstream data("Items.txt"); ofstream temp("Temp.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

count++;

if (sr == count)

{

// Converting price to integer price

int Price = convertToInteger(getField(record, 2));

// Discount formula

Price = Price - (Price \* discount) / 100;

temp << getField(record, 1) << "," << to\_string(Price) << "," << getField(record, 3) << "," << to\_string(discount) << endl;continue;

}

temp << record << endl;

}

data.close(); temp.close(); // Close both files

remove("Items.txt"); rename("Temp.txt", "Items.txt"); //Remove and rename Files

}

else {

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

{

// convert string itemPrice to integer

int itemPrice1 = convertToInteger(itemPrice[i]);

if (sr - 1 == i)

{

itemPrice1 = itemPrice1 - (itemPrice1 \* discount) / 100;

// Convert item price again to string and give value to itemPrice array

itemPrice[i] = to\_string(itemPrice1);

}

}

}

}

else

{ invalidSerialNo(); }

}

else

{ invalidCredential(); }

}

// Function to view feedback statistics

void viewFeedback()

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Feedback \u001b[32;1m>\u001b[0m");

ifstream data("Feedback.txt");

int option, excellentOption = 0, goodOption = 0, badOption = 0, worstOption = 0;

while (data >> option)

switch (option)

{

case 1:

excellentOption++; break;

case 2:

goodOption++; break;

case 3:

badOption++; break;

case 4:

worstOption++; break;

}

cout << "\n\t\t\t\t\t\t>>> Excellent Experience: " << excellentOption;

cout << "\n\t\t\t\t\t\t>>> Good Experience: " << goodOption;

cout << "\n\t\t\t\t\t\t>>> Bad Experience: " << badOption;

cout << "\n\t\t\t\t\t\t>>> Worst Experience: " << worstOption << endl;

pressKey();

}

// Function to logout

void logout()

{

head();

cout << "\n\u001b[1m\u001b[36;1m\u001b[1mLogging Out...\u001b[0m" << endl;

for (int i = 0; i < 100; i++)

{

Sleep(10);

cout << "\033[1000D" << i + 1 << "%" << flush;

}

cout << endl;

}

// Investor Functions //

// Function to change the password

void changePass(int &index, string &profile, string username[], string password[])

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Change Password \u001b[32;1m>\u001b[0m");

string pass;

// To read data from file

string record; ifstream data("User.txt"); ofstream temp("Temp.txt");

while (getline(data, record))

{

if (getField(record, 1) == profile)

{

gotoxy(75, 24);

cout << ">>> Old password is: " << getField(record, 2);

gotoxy(75, 25);

cout << ">>> Enter New Password: ";

pass = getInput(pass);

temp << getField(record, 1) << "," << pass << "," << getField(record, 3) << "," << getField(record, 4) << endl;

gotoxy(75, 27);

cout << "\e[0;92mPassword changed successfully...\e[0m";

Sleep(1000); continue;

}

temp << record << endl;

}

data.close(); temp.close();

// Remove and rename the text files

remove("User.txt"); rename("Temp.txt", "User.txt");

// Store information in arrays also

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

{

if (username[i] == profile)

{ password[i] = pass; }

}

}

// Function to view the income or revenue

void viewRevenue(int &profit)

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m View Revenue \u001b[32;1m>\u001b[0m");

gotoxy(75, 27);

cout << ">>> Revenue is: " << profit;

pressKey();

}

// Function to view all suppliers profiles

void SuppliersProfiles(int &index, int &viewSerial, string username[], string password[], string role[])

{

int y = 24;

subMenuHeader("\u001b[1m\u001b[32;1m<\u001b[0m\u001b[1m Suppliers Profiles \u001b[32;1m>\u001b[0m");

gotoxy(40, 24);

cout << "Sr.#"; gotoxy(80, 24);cout << "Username";

gotoxy(120, 24); cout << "Password";

string record; ifstream data("User.txt");

bool fileExists = data.good();

if (fileExists)

{

while(getline(data, record))

{

if (getField(record, 3) == "SUPPLIER")

{

gotoxy(40, y + 2);

cout << viewSerial;

gotoxy(80, y + 2);

cout << getField(record, 1);

gotoxy(120, y + 2);

cout << getField(record, 2);

viewSerial++;

y += 2;

}

}

}

else { //If file does not opens

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

{

if (role[i] == "SUPPLIER")

{

gotoxy(40, y + 2);

cout << viewSerial;

gotoxy(80, y + 2);

cout << username[i];

gotoxy(120, y + 2);

cout << password[i];

viewSerial++;

y += 2;

}

}

}

viewSerial = 1;

pressKey();

}

// Supplier's Functions //

// Function to add item in the inventory by the supplier

void addProductsToInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[])

{

string name = "", price = "", quantity = "";

subMenuHeader("\u001b[1m\u001b[32;1m<\u001b[0m\u001b[1m Add Products \u001b[32;1m>\u001b[0m");

cout << "\n\n\t\t\t\t\t\t>>> Enter the Product name: ";

name = getInput(name);

cout << "\t\t\t\t\t\t>>> Enter the product price: ";

price = getInput(price);

cout << "\t\t\t\t\t\t>>> Enter the product quantity: ";

quantity = getInput(quantity);

if (containsOnlyIntegers(price) && containsOnlyIntegers(quantity))

{

// File Hnadling

ofstream data("Inventory.txt", ios::app);

data << name << "," << price << "," << quantity << endl;

data.close();

// Also add items in arrays

inventoryProduct[inventoryIndex] = name;

inventoryPrice[inventoryIndex] = price;

inventoryQuantity[inventoryIndex] = quantity;

inventoryIndex++;

cout << "\n\n";

cout << "\e[0;92mProduct Added Successfully...\e[0m";

Sleep(500);

}

else

{ invalidCredential(); }

}

// Function to view the inventory

void viewInventory(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[])

{

int y = 24, i = 1;

subMenuHeader("\u001b[1m\u001b[32;1m<\u001b[0m\u001b[1m Inventory \u001b[32;1m>\u001b[0m");

gotoxy(30, 24);

cout << "Sr No.";

gotoxy(60, 24);

cout << "Name";

gotoxy(90, 24);

cout << "Price";

gotoxy(120, 24);

cout << "Quantity";

string record; ifstream data("Inventory.txt");

bool fileExists = data.good(); // check if file exists

if (fileExists)

{

while (getline(data, record))

{

gotoxy(30, y + 2);

cout << i;

gotoxy(60, y + 2);

cout << getField(record, 1);

gotoxy(90, y + 2);

cout << getField(record, 2);

gotoxy(120, y + 2);

cout << getField(record, 3);

y += 2;

i++;

}

}

else { //Display if file does not exist

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

{

gotoxy(30, y + 2);

cout << i + 1;

gotoxy(60, y + 2);

cout << inventoryProduct[i];

gotoxy(90, y + 2);

cout << inventoryPrice[i];

gotoxy(120, y + 2);

cout << inventoryQuantity[i];

y += 2;

}

}

pressKey();

}

// Function to give discount on inventory products by supplier

void offerDiscount(int &inventoryIndex, string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[])

{

string strSr = "", strDiscount = "";

viewInventory(inventoryIndex, inventoryProduct, inventoryPrice, inventoryQuantity);

cout << "\n";

cout << "\t\t\t\t\t>>> Enter the serial Number of item for discount: ";

strSr = getInput(strSr);

cout << "\t\t\t\t\t>>> Enter the percentage(%) of discount: ";

strDiscount = getInput(strDiscount);

if (containsOnlyIntegers(strSr) && containsOnlyIntegers(strDiscount))

{

int sr = convertToInteger(strSr), discount = convertToInteger(strDiscount);

if (sr > 0 && sr <= inventoryIndex + 1)

{

int count = 0; string record; ifstream data("Inventory.txt"); ofstream temp("Temp.txt");

bool fileExists = data.good();

if(fileExists)

{

while (getline(data, record))

{

count++;

if (sr == count)

{

int Price = convertToInteger(getField(record, 2));

Price = Price - (Price \* discount) / 100;

temp << getField(record, 1) << "," << to\_string(Price) << "," << getField(record, 3) << endl;

continue;

}

temp << record << endl;

}

data.close(); temp.close();

remove("Inventory.txt"); rename("Temp.txt", "Inventory.txt");

cout << "\e[0;92m\n\t\t\t\t\tItem Discounted...\e[0m";

Sleep(500);

}

else {

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

{

int price = convertToInteger(inventoryPrice[i]);

if (sr - 1 == i)

{

inventoryPrice[i] = to\_string(price - (price \* discount) / 100);

cout << "\e[0;92m\n\t\t\t\t\tItem Discounted...\e[0m";

Sleep(500);

}

}

}

}

else

{ invalidSerialNo(); }

}

else

{ invalidCredential(); }

}

// Function for supplier to check its total sale

void checkReceiving(int &inventoryIndex, int &balance, int supplierBalance, string inventoryPrice[], string inventoryQuantity[])

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Your Receiving \u001b[32;1m>\u001b[0m");

inventoryIndex = readIndex("Inventory.txt");

gotoxy(60, 25);

cout << ">>> The amount of products you sold: " << inventoryIndex;

string record; ifstream data("Inventory.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

balance += ( convertToInteger(getField(record, 2)) \* convertToInteger(getField(record, 3)) );

}

data.close();

if (supplierBalance == 0)

{

gotoxy(60, 27);

cout << ">>> Your balance is: \u001b[37;1m" << balance << "\u001b[0m Rupee";

}

else {

gotoxy(60, 27);

cout << ">>> Your balance is: \u001b[37;1m" << supplierBalance << "\u001b[0m Rupee";

}

}

else {

gotoxy(60, 29);

cout << "File not found! >> Make Sure File Exists...";

}

pressKey();

}

// Function for suppier to withdraw its income

void withDrawBalance(int &balance, int& supplierBalance)

{

string strBalanceWithdraw;

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Withdraw Balance \u001b[32;1m>\u001b[0m");

if (supplierBalance == 0)

{

cout << "\n\t\t\t\t\t>>> Your balance is: \u001b[37;1m" << balance << "\u001b[0m Rupee" << endl;

}

else {

balance = supplierBalance;

cout << "\n\t\t\t\t\t>>> Your balance is: \u001b[37;1m" << supplierBalance << "\u001b[0m Rupee" << endl;

}

cout << "\t\t\t\t\t>>> Enter the amount you want to withdraw: ";

strBalanceWithdraw = getInput(strBalanceWithdraw);

if (containsOnlyIntegers(strBalanceWithdraw))

{

int balanceWithdraw = convertToInteger(strBalanceWithdraw);

if (balanceWithdraw <= balance)

{

supplierBalance = balance - balanceWithdraw;

cout << "\t\t\t\t\t\e[0;92mBalance withdraw successfully...\e[0m"; Sleep(500);

cout << "\n\n\t\t\t\t\tNew balance is: \u001b[37;1m" << supplierBalance << "\u001b[0m Rupee"; //Bright White color

pressKey();

}

else

{ invalidOption(); }

}

else

{ invalidCredential(); }

}

// Employee Functions //

// Function for employee to see its profile

void viewEmployeeProfile(int &index, string &profile, string username[], string password[], string role[], string addEmployeeSalary[])

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Your Profile \u001b[32;1m>\u001b[0m");

string record; ifstream data("User.txt");

bool fileExists = data.good(); // check if file exists

if(fileExists)

{

while(getline(data, record))

{

if (profile == getField(record, 1))

{

gotoxy(60, 24);

cout << ">>> Username: " << getField(record, 1);

gotoxy(60, 26);

cout << ">>> Password: " << getField(record, 2);

gotoxy(60, 28);

cout << ">>> Role: " << getField(record, 3);

gotoxy(60, 30);

cout << ">>> Salary: " << getField(record, 4); break;

}

}

data.close();

}

else {

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

{

if (username[i] == profile)

{

gotoxy(60, 24);

cout << ">>> Username: " << username[i];

gotoxy(60, 26);

cout << ">>> Password: " << password[i];

gotoxy(60, 28);

cout << ">>> Role: " << role[i];

gotoxy(60, 30);

cout << ">>> Salary: " << addEmployeeSalary[i];

break;

}

}

}

pressKey();

}

// Function for employee to move the items from inventory to the place where customer can see and buy them

void inventoryToSale(int &index, int &inventoryIndex, int &profit, int &itemIndex, string itemName[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string itemPrice[], string itemQuantity[])

{

string strSr = "", strPrice = "";

viewInventory(inventoryIndex, inventoryProduct, inventoryPrice, inventoryQuantity);

cout << "\n\n\t\t\t\t\t>>> Enter the Serial No. Of Product: ";

strSr = getInput(strSr);

cout << "\n\t\t\t\t\t>>> Enter New Price: ";

strPrice = getInput(strPrice);

if (containsOnlyIntegers(strSr) && containsOnlyIntegers(strPrice))

{

int sr = convertToInteger(strSr);

int newPrice = convertToInteger(strPrice);

int count = 0;

inventoryIndex = readIndex("Inventory.txt");

if (sr < inventoryIndex + 1 && sr > 0)

{

string record; ifstream data("Inventory.txt");

bool fileExists = data.good();

if (fileExists)

{

ofstream temp("Temp.txt"); ofstream data1("Items.txt", ios::app);

while (getline(data, record))

{

count++;

if (count == sr)

{

data1 << getField(record, 1) << "," << newPrice << "," << getField(record, 3) << endl;

int previousPrice = convertToInteger(getField(record, 2));

profit += ( (newPrice - previousPrice) \* convertToInteger(getField(record, 3)) ); continue;

}

temp << record << endl;

}

data.close(); temp.close(); data1.close();

remove("Inventory.txt");

rename("Temp.txt", "Inventory.txt");

}

else {

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

{

if (sr - 1 == i)

{

// To move product to items list

itemName[i] = inventoryProduct[i];

itemPrice[i] = inventoryPrice[i];

itemQuantity[i] = inventoryQuantity[i];

itemIndex++;

// To delete current product from inventory

for (int j = i; j < inventoryIndex; j++)

{

inventoryProduct[j] = inventoryProduct[j + 1];

inventoryPrice[j] = inventoryPrice[j + 1];

inventoryQuantity[j] = inventoryQuantity[j + 1];

}

inventoryIndex--;

int newInventoryPrice = convertToInteger(inventoryPrice[i]);

profit += (newInventoryPrice - newPrice);

break;

}

}

}

}

else

{ invalidCredential(); }

}

else

{ invalidCredential(); }

}

// Customer Functions

// Function for customer to see its cart

void itemsToCart(int &itemIndex, int &cartIndex, string itemName[], string itemPrice[], string itemQuantity[], string cartName[], string cartPrice[], string cartQuantity[])

{

string strSr = "", strQuantity = "";

viewItems(itemIndex, itemName, itemPrice, itemQuantity);

cout << "\n\t\t\t\t\t>>> Enter the serial No. of item you want to add: ";

strSr = getInput(strSr);

cout << "\t\t\t\t\t>>> Enter the quanity: ";

strQuantity = getInput(strQuantity);

if (containsOnlyIntegers(strSr) && containsOnlyIntegers(strQuantity))

{

// Convert credentials to integers for calculations

int sr = convertToInteger(strSr), quantity = convertToInteger(strQuantity);

int count = 0; string record; ifstream data("Items.txt");

bool fileExists = data.good();

if(fileExists)

{

ofstream temp("Temp.txt"); ofstream data1("Cart.txt", ios::app);

while (getline(data, record))

{

count++;

if (sr == count)

{

int itemQ = convertToInteger(getField(record, 3));

if (quantity <= itemQ)

{

data1 << record << endl;

continue;

}

else {

invalidCredential();

}

}

temp << record << endl;

}

data.close(); data1.close(); temp.close();

remove("Items.txt"); rename("Temp.txt", "Items.txt");

}

else {

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

{

int itemQ = convertToInteger(itemQuantity[i]);

if (sr - 1 == i)

{

if (quantity <= itemQ)

{

cartName[cartIndex] = itemName[i];

cartPrice[cartIndex] = itemPrice[i];

cartQuantity[cartIndex] = to\_string(quantity);

cartIndex++;

itemQuantity[i] = to\_string(itemQ - quantity);

if (itemQuantity[i] == "0")

{

itemName[i] = itemName[i + 1];

itemPrice[i] = itemPrice[i + 1];

itemIndex--;

}

cout << "\t\t\t\t\t\e[0;92mItem Added To cart Successfullly...\e[0m";

Sleep(500);

}

else

{

invalidCredential(); break;

}

}

}

}

}

else

{ invalidCredential(); }

}

// Function for customer to see its bill

void viewBill(int &cartIndex, string cartName[], string cartPrice[], string cartQuantity[])

{

int y = 10;

system("cls");

system("color f0");

gotoxy(75, 5);

cout << "< Bill >";

gotoxy(37, 10);

cout << "Product Name";

gotoxy(77, 10);

cout << "Product Price";

gotoxy(117, 10);

cout << "Product Quantity";

string record; ifstream data("Cart.txt");

bool fileExists = data.good();

if (fileExists)

{

while (getline(data, record))

{

gotoxy(40, y + 2);

cout << getField(record, 1);

gotoxy(80, y + 2);

cout << getField(record, 2);

gotoxy(120, y + 2);

cout << getField(record, 3);

y += 2;

}

}

else {

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

{

gotoxy(40, y + 2);

cout << cartName[i];

gotoxy(80, y + 2);

cout << cartPrice[i];

gotoxy(120, y + 2);

cout << cartQuantity[i];

y += 2;

}

}

pressKey();

colorReset();

}

// Function for cusotmer to give feedback

void giveFeedback(int &excellent, int &good, int &bad, int &worst)

{

subMenuHeader("\u001b[32;1m<\u001b[0m\u001b[1m Your Feedback \u001b[32;1m>\u001b[0m");

ofstream data("Feedback.txt", ios::app);

string strOption = "";

cout << "\n\t\t\t\t\t>>> How was your experience:\n";

cout << "\t\t\t\t\t1. Excellent :>\n\t\t\t\t\t2. Good :)\n\t\t\t\t\t3. Bad :/\n\t\t\t\t\t4. Worst :(\n\n";

cout << "\t\t\t\t\tYour Option: ";

strOption = getInput(strOption);

cout << "\n\t\t\t\t\t\u001b[32;1mThanks for your feedback...\u001b[0m";

Sleep(500);

if (containsOnlyIntegers(strOption))

{

int option = convertToInteger(strOption);

if (option > 0 && option < 5)

{

// Store option in file

data << option << endl;

switch (option)

{

case 1: excellent++; break;

case 2: good++; break;

case 3: bad++; break;

case 4: worst++; break;

default: option = 0; break;

}

}

else

{ invalidOption(); }

}

else

{ invalidCredential(); }

}

// Validations

bool containsOnlyAlphabets(string word) // It validates if the required input are alphabets

{

bool result = true;

for (int i = 0; word[i] != '\0'; i++)

{

int asciiCode = word[i];

if (asciiCode < 65 || asciiCode > 122)

{

result = false;

break;

}

}

return result;

}

bool containsOnlyIntegers(string integer) // It validates if the required input are integers

{

bool result = true;

for (int i = 0; integer[i] != '\0'; i++)

{

int asciiCode = integer[i];

if (asciiCode < 48 || asciiCode > 57)

{

result = false;

break;

}

}

return result;

}

int convertToInteger(string integer)

{

int result = 0;

for (int i = 0; i < integer.length(); i++)

{

// Mutiply result by 10 so that integer is move to left position

result = result \* 10 + (integer[i] - '0');

}

return result;

}

// This function removes the spaces in input

string removeSpaces(const string &input)

{

string result;

for (int i = 0; i < input.length(); i++)

{

char character = input[i];

if (character != ' ')

{

result += character;

}

}

return result;

}

// This function gets the input without spaces

string getInput(string variable)

{

string newString;

getline(cin, variable);

newString = removeSpaces(variable);

return newString;

}

// Function to add user information to a file

void addUserToFile(const string &name, const string &password, const string &role, const string &salary)

{

ofstream file("User.txt", ios::app); // Open file in append mode

if (role == "EMPLOYEE")

{

file << name << "," << password << "," << role << "," << salary << endl;

}

else

{

file << name << "," << password << "," << role << endl;

}

file.close(); // Close the file

}

// To read specific field in data

string getField(string record, int field)

{

int commaCount = 1;

string item;

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

{

if (record[x] == ',')

{

commaCount = commaCount + 1;

}

else if (commaCount == field)

{

item = item + record[x];

}

}

return item;

}

// Function to check from file whether the name is already present in file

string isPresent(string name, string password, string names[], string passwords[], string role[], int index)

{

string resultRole = "INVALID";

// To read data from file

string record;

fstream data;

data.open("User.txt", ios::in);

while (getline(data, record))

{

string nameInFile = getField(record, 1);

string passwordInFile = getField(record, 2);

if (nameInFile == name && passwordInFile == password)

{

string role = getField(record, 3);

resultRole = role;

break;

}

}

data.close();

// If data is not present in file read it from arrays

for (int x = 0; x < index; x++)

{

if (name == names[x] && password == passwords[x])

{ return role[x]; }

}

return resultRole;

}

//Function to read the indexes

int readIndex(string filename)

{

int index = 0;

string record; ifstream data(filename);

while (getline(data, record))

{ index++; }

data.close(); return index;

}

// Function for sending data into arrays

void dataTransfer(string username[], string password[], string role[], string addEmployeeSalary[], string itemName[], string itemPrice[], string itemQuantity[], string inventoryProduct[], string inventoryPrice[], string inventoryQuantity[], string cartName[], string cartPrice[], string cartQuantity[], int& index, int& itemIndex, int& inventoryIndex, int& cartIndex, int& addEmpIndex)

{

int i = 0, j = 0;

string record; ifstream data("User.txt");

while (getline(data, record))

{

username[i] = getField(record, 1);

password[i] = getField(record, 2);

role[i] = getField(record, 3);

addEmployeeSalary[i] = getField(record, 4);

i++; //For Index

if (getField(record, 3) == "EMPLOYEE") //For addEmpIndex

{

j++;

}

}

index = i;

i = 0; data.close();

string record1; ifstream data1("Items.txt");

while(getline(data1, record1))

{

itemName[i] = getField(record1, 1);

itemPrice[i] = getField(record1, 2);

itemQuantity[i] = getField(record1, 3);

i++;

}

itemIndex = i;

i = 0; data1.close();

string record2; ifstream data2("Inventory.txt");

while(getline(data2, record2))

{

inventoryProduct[i] = getField(record2, 1);

inventoryPrice[i] = getField(record2, 2);

inventoryQuantity[i] = getField(record2, 3);

i++;

}

inventoryIndex = i;

i = 0; data2.close();

string record3; ifstream data3("Cart.txt");

while(getline(data3, record3))

{

cartName[i] = getField(record3, 1);

cartPrice[i] = getField(record3, 2);

cartQuantity[i] = getField(record3, 3);

i++;

}

cartIndex = i;

}

# Weakness in the Business Application:

* This application has multiple users seeing the same functions. E.g., a supplier with certain username and password can see his balance but also other supplier with different password will see the same balance in his account.
* A same bill is displaying to customers with different usernames and passwords.
* If we change the integer type thing in files it cannot show error but print as it is in the application.

# Future Directions:

* I want to make it user differentiable application. It should have a different whole file for a specific user. It should print revenue and bill for each user independently.
* I would love to make it in graphical interface.
* I want it to handle digital transactions.