**Online Shopping management System**



Session: 2022 – 2026

**Submitted by:**

Muhammad Adnan 2022-CS-59

**Supervised by:**

Mam Maida Shahid

Mr. Irzam Liaqat

Department of Computer Science

**University of Engineering and Technology**

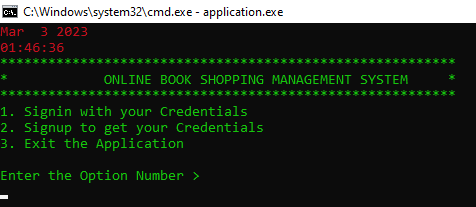
**Lahore Pakistan**

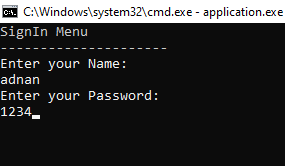
**Here you can find the major parts of your Business Application documentation**

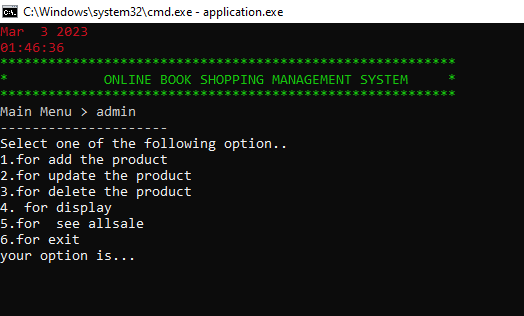
* **Short Description of your project**
  + I want to make a program to help online shopping to maintain their expanse records, and calculate their sales .it also helps the customer to buy
  + Needed things on their own. It saves the record of available stocks and sales.
  + It saves the waste of time.
* **Users of Application (minimum 2 users for your project)**
  + This application is particularly for online business(shopping) provide (like books, fast foods, and electric device).
  + My main target is E-Comerce
  + Admin: Admin will access the additional feature like prices, sales and total income
* **Functional Requirements**
  + Functional requirements are a detailed description of what a system, product, or service must do to meet the needs and expectations
  + Use the following format to write the Functional Requirements

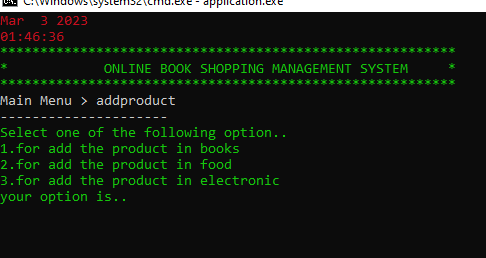
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User Story ID | *As a* | *I want to perform* | | *So that I can* |
| **1** | Admin | Load the data from files | | See the details of products, prices, quantity and sale |
| **2** | Admin | Add the products | | Add the new products |
| **3** | Admin | update the products | | update the prices of products |
| **4** | Admin | delete the products | | delete the unavailable products |
| **5** | Admin | See the available stocks | | view the available stocks |
| **6** | Admin | See the sale of each products. | | view the sale of each products |
| **7** | Admin | See the total sale of all products | | View the total sale of products |
| **8** | User | Search the product | | Search the available stock |
| **9** | User | See the product details | | View the products details |
| **10** | User | order the products | | order the products |
|  | | |

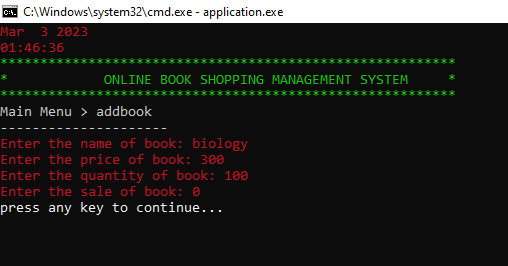
* **Wireframes**

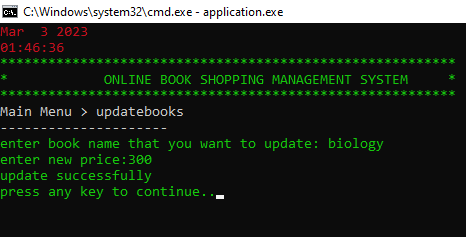
****

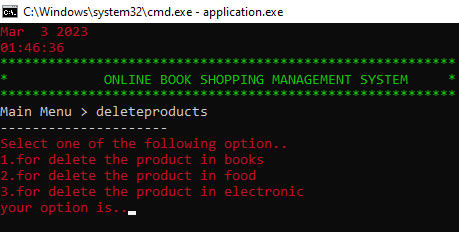
****

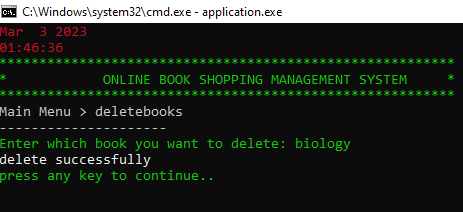
****

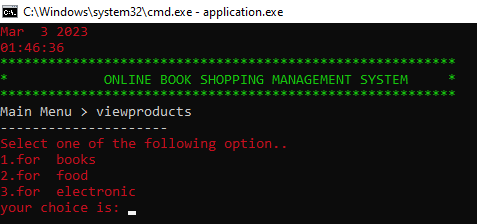
****

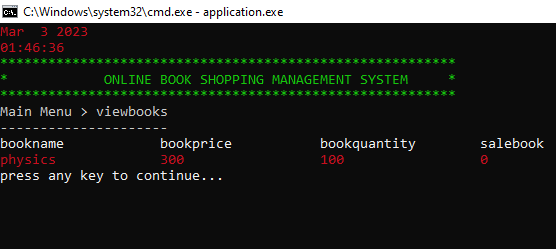
****

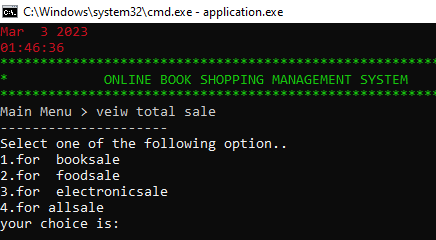
****

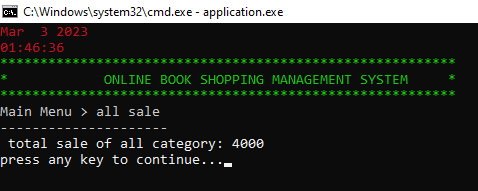
****

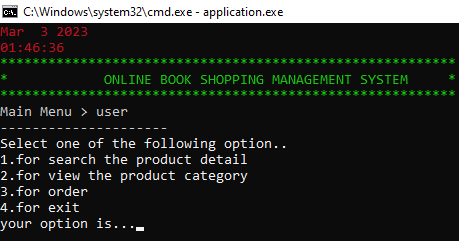
****

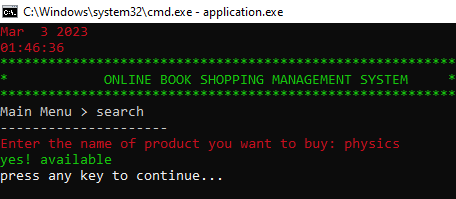
****

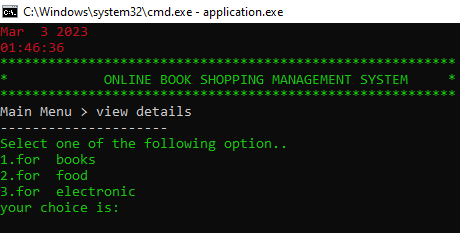
****

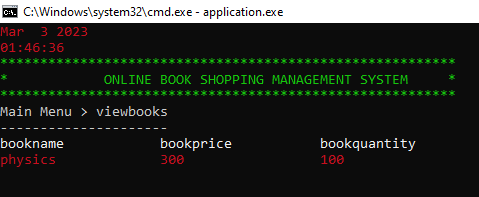
****

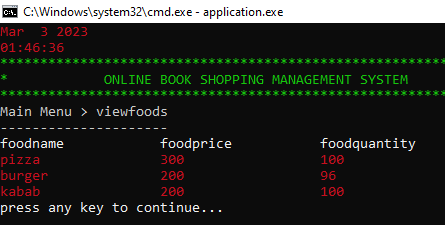
****

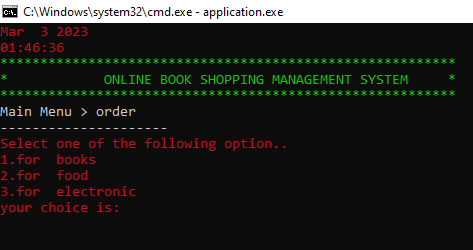
****

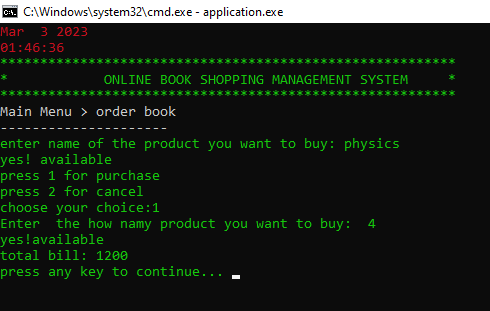
****

****

****

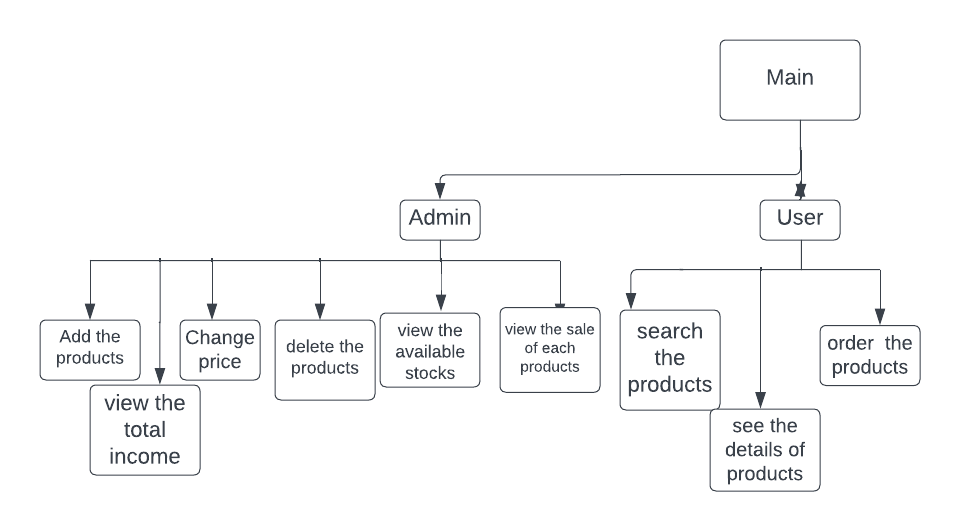
****

****

****

**Important Instructions**

* **Data Structures (Parallel Arrays)**
* // arrays
* string bookname[50];
* string foodname[50];
* string electronicname[50];
* int bookprice[50], foodprice[50], electronicprice[50];
* int bookquantity[50];
* int foodquantity[50];
* int electronicquantity[50];
* int salebook[50];
* int salefood[50];
* int saleelectronic[50];
* **Function Prototypes**
* void header();
* const int userArrSize = 50;
* string users[userArrSize];
* string passwords[userArrSize];
* string roles[userArrSize];
* int usersCount = 0;
* void loginmenu();
* string signIn(string name, string password);
* bool signUp(string name, string password, string role);
* void loadsignup();
* string inputsignup(string name, int num);
* void subMenuBeforeMainMenu(string submenu);
* void subMenu(string submenu);
* int loginMenu();
* int loginOption = 0;
* void admin();
* // admin functions
* void updateproducts();
* void updatebook();
* void storeupdatebook();
* void updatefood();
* void storeupdatefood();
* void updateelectronic();
* void storeupdateelectronic();
* void addproducts();
* void addbook();
* void storebook(string bookname, int bookprice, int bookquantity, int salebook);
* void addelectronic();
* void storeelectronic(string electronicname, int electronicprice, int electronicquantity, int saleelectronic);
* void addfood();
* void storefood(string foodname, int foodprice, int foodquantity, int salefood);
* void deleteproducts();
* void deletebook();
* void storedeletebook();
* void deletefood();
* void storedeletefood();
* void deleteelectronic();
* void storedeleteelectronic();
* void display();
* void displaybook();
* void displayfood();
* void displayelectronic();
* //  loadfunctions
* void loadbook();
* void loadfood();
* void loadelectronic();
* string inputbook(string, int);
* string inputfood(string, int);
* string inputelectronic(string, int);
* void storesalebook();
* void storesalefood();
* void storesaleelectronic();
* void allsalebook();
* void storetotalbooksale();
* void storetotalfoodsale();
* void storetotalelectronicsale();
* void loadtotalbooksale();
* void loadtotalfoodsale();
* void loadtotalelectronicsale();
* void allsalefood();
* void allsaleelectronic();
* void allsale();
* void storeincome();
* void loadincome();
* void totalsale();
* // user funtions
* void user();
* void search();
* void viewdetails();
* void viewbook();
* void viewfood();
* void viewelectronic();
* void order();
* void orderbook();
* void orderfood();
* void orderelectronic();
* **Functions Working Flow**
  + Here you have to draw a diagram that will show how you are calling your functions. This will show how you have designed the flow of your code.
  + Here is an example of your Functions Working Flow diagram.



* **Complete Code of the Business Application**
* #include <iostream>
* #include <fstream>
* #include <conio.h>
* #include <iomanip>
* #include <windows.h>
* #include <time.h>
* using namespace std;
* void isnotvalid();
* HANDLE h = GetStdHandle(STD\_OUTPUT\_HANDLE);
* // arrays
* string bookname[50];
* string foodname[50];
* string electronicname[50];
* int bookprice[50], foodprice[50], electronicprice[50];
* int bookquantity[50];
* int foodquantity[50];
* int electronicquantity[50];
* int salebook[50];
* int salefood[50];
* int saleelectronic[50];
* // Arrays index
* int idxbook = 0;
* int idxfood = 0;
* int idxelectronic = 0;
* int booksale = 0;
* int foodsale = 0;
* int electronicsale = 0;
* int income = 0;
* // functions//
* void header();
* const int userArrSize = 50;
* string users[userArrSize];
* string passwords[userArrSize];
* string roles[userArrSize];
* int usersCount = 0;
* void loginmenu();
* string signIn(string name, string password);
* bool signUp(string name, string password, string role);
* void loadsignup();
* string inputsignup(string name, int num);
* void subMenuBeforeMainMenu(string submenu);
* void subMenu(string submenu);
* int loginMenu();
* int loginOption = 0;
* void admin();
* //  loadfunctions
* void loadbook();
* void loadfood();
* void loadelectronic();
* string inputbook(string, int);
* string inputfood(string, int);
* string inputelectronic(string, int);
* // admin functions
* void updateproducts();
* void updatebook();
* void storeupdatebook();
* void updatefood();
* void storeupdatefood();
* void updateelectronic();
* void storeupdateelectronic();
* void addproducts();
* void addbook();
* void storebook(string bookname, int bookprice, int bookquantity, int salebook);
* void addelectronic();
* void storeelectronic(string electronicname, int electronicprice, int electronicquantity, int saleelectronic);
* void addfood();
* void storefood(string foodname, int foodprice, int foodquantity, int salefood);
* void deleteproducts();
* void deletebook();
* void storedeletebook();
* void deletefood();
* void storedeletefood();
* void deleteelectronic();
* void storedeleteelectronic();
* void display();
* void displaybook();
* void displayfood();
* void displayelectronic();
* // sales functions
* void storesalebook();
* void storesalefood();
* void storesaleelectronic();
* void allsalebook();
* void storetotalbooksale();
* void storetotalfoodsale();
* void storetotalelectronicsale();
* void loadtotalbooksale();
* void loadtotalfoodsale();
* void loadtotalelectronicsale();
* void allsalefood();
* void allsaleelectronic();
* void allsale();
* void storeincome();
* void loadincome();
* void totalsale();
* // user funtions
* void user();
* void search();
* void viewdetails();
* void viewbook();
* void viewfood();
* void viewelectronic();
* void order();
* void orderbook();
* void orderfood();
* void orderelectronic();
* int number;
* main()
* {
* system("cls");
* header();
* loadbook();
* loadfood();
* loadelectronic();
* loadtotalbooksale();
* loadtotalfoodsale();
* loadtotalelectronicsale();
* loadsignup();
* while (loginOption != 3)
* {
* system("cls");
* header();
* loginOption = loginMenu();
* SetConsoleTextAttribute(h, 4);
* subMenuBeforeMainMenu("Login");
* storeincome();
* if (loginOption == 1)
* {
* system("cls");
* string name;
* string password;
* string role;
* subMenuBeforeMainMenu("SignIn");
* cout << "Enter your Name: " << endl;
* cin>> name;
* cout << "Enter your Password: " << endl;
* cin >> password;
* role = signIn(name, password);
* SetConsoleTextAttribute(h, 15);
* if (role == "admin")
* {
* admin();
* }
* else if (role == "user")
* {
* user();
* }
* else if (role == "undefined")
* {
* cout << "You Entered wrong Credentials" << endl;
* }
* }
* else if (loginOption == 2)
* {
* system("cls");
* string name;
* string password;
* string role;
* SetConsoleTextAttribute(h, 7);
* subMenuBeforeMainMenu("SignUp");
* cout << "Enter your Name: " << endl;
* cin>> name;
* cout << "Enter your Password: " << endl;
* cin >> password;
* cout << "Enter your Role (admin or user and manager): " << endl;
* cin >> role;
* SetConsoleTextAttribute(h, 15);
* bool isValid = signUp(name, password, role);
* if (isValid)
* {
* cout << "SignedUp Successfully" << endl;
* }
* if (!isValid)
* {
* cout << "Users in the System have exceeded the Capacity" << endl;
* }
* cout << "press any key to continue..";
* getch();
* }
* }
* }
* int loginMenu()
* {
* int opti;
* SetConsoleTextAttribute(h, 10);
* cout << "1. Signin with your Credentials" << endl;
* cout << "2. Signup to get your Credentials" << endl;
* cout << "3. Exit the Application" << endl;
* cout << endl;
* cout << "Enter the Option Number > ";
* while (!(cin >> opti))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* return opti;
* }
* string signIn(string name, string password)
* {
* for (int index = 0; index < usersCount; index++)
* {
* if (users[index] == name && passwords[index] == password)
* {
* return roles[index];
* }
* }
* return "undefined";
* }
* void appsignin(string users, string passwords, string roles)
* {
* fstream file;
* file.open("signup.txt", ios::app);
* file << endl
* << users << ',' << passwords << ',' << roles;
* file.close();
* }
* bool signUp(string name, string password, string role)
* {
* if (usersCount < userArrSize)
* {
* users[usersCount] = name;
* passwords[usersCount] = password;
* roles[usersCount] = role;
* usersCount++;
* appsignin(users[usersCount], passwords[usersCount], roles[usersCount]);
* return true;
* }
* else
* {
* return false;
* }
* }
* void loadsignup()
* {
* string line;
* fstream file;
* file.open("signup.txt", ios::in);
* while (getline(file, line))
* {
* users[usersCount] = inputsignup(line, 1);
* passwords[usersCount] = inputsignup(line, 2);
* roles[usersCount] = inputsignup(line, 3);
* usersCount++;
* }
* }
* string inputsignup(string line, int num)
* {
* int commacount = 1;
* string item;
* for (int idx = 0; idx < line.length(); idx++)
* {
* if (line[idx] == ',')
* {
* commacount++;
* }
* else if (commacount == num)
* {
* item = item + line[idx];
* }
* }
* return item;
* }
* void subMenuBeforeMainMenu(string submenu)
* {
* SetConsoleTextAttribute(h, 7);
* string message = submenu + " Menu";
* cout << message << endl;
* cout << "---------------------" << endl;
* SetConsoleTextAttribute(h, 15);
* }
* void subMenu(string submenu)
* {
* SetConsoleTextAttribute(h, 7);
* string message = "Main Menu > " + submenu;
* cout << message << endl;
* cout << "---------------------" << endl;
* SetConsoleTextAttribute(h, 15);
* }
* void loginmenu()
* {
* SetConsoleTextAttribute(h, 7);
* cout << "1. for admin" << endl;
* cout << "2. for user" << endl;
* cout << " 3.for exit" << endl;
* SetConsoleTextAttribute(h, 15);
* }
* void admin()
* {
* system("cls");
* header();
* int option;
* while (option != 6)
* {
* system("cls");
* header();
* SetConsoleTextAttribute(h, 7);
* subMenu("admin");
* cout << "Select one of the following option.." << endl;
* cout << "1.for add the product" << endl;
* cout << "2.for update the product" << endl;
* cout << "3.for delete the product" << endl;
* cout << "4. for display " << endl;
* cout << "5.for  see allsale" << endl;
* cout << "6.for exit" << endl;
* cout << "your option is...";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* addproducts();
* }
* else if (option == 2)
* {
* updateproducts();
* }
* else if (option == 3)
* {
* deleteproducts();
* }
* else if (option == 4)
* {
* display();
* }
* else if (option == 5)
* {
* totalsale();
* }
* }
* }
* void header()
* {
* SetConsoleTextAttribute(h, 4);
* cout << \_\_DATE\_\_ << endl;
* cout << \_\_TIME\_\_ << endl;
* SetConsoleTextAttribute(h, 10);
* cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
* cout << "\*            ONLINE BOOK SHOPPING MANAGEMENT SYSTEM     \*" << endl;
* cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;
* SetConsoleTextAttribute(h, 15);
* }
* void loadbook()
* {
* fstream file;
* string word;
* file.open("addbookfile.txt", ios::in);
* while (getline(file, word))
* {
* bookname[idxbook] = inputbook(word, 1);
* bookprice[idxbook] = stoi(inputbook(word, 2));
* bookquantity[idxbook] = stoi(inputbook(word, 3));
* salebook[idxbook] = stoi(inputbook(word, 4));
* idxbook++;
* }
* file.close();
* }
* string inputbook(string word, int field)
* {
* int commacount = 1;
* string item;
* for (int idx = 0; idx < word.length(); idx++)
* {
* if (word[idx] == ',')
* {
* commacount++;
* }
* else if (commacount == field)
* {
* item = item + word[idx];
* }
* }
* return item;
* }
* void loadfood()
* {
* fstream file;
* string word;
* file.open("addfoodfile.txt", ios::in);
* while (getline(file, word))
* {
* foodname[idxfood] = inputfood(word, 1);
* foodprice[idxfood] = stoi(inputfood(word, 2));
* foodquantity[idxfood] = stoi(inputfood(word, 3));
* salefood[idxfood] = stoi(inputfood(word, 4));
* idxfood++;
* }
* file.close();
* }
* string inputfood(string word, int field)
* {
* int commacount = 1;
* string item;
* for (int idx = 0; idx < word.length(); idx++)
* {
* if (word[idx] == ',')
* {
* commacount++;
* }
* else if (commacount == field)
* {
* item = item + word[idx];
* }
* }
* return item;
* }
* void loadelectronic()
* {
* fstream file;
* string word;
* file.open("addelectronicfile.txt", ios::in);
* while (getline(file, word))
* {
* electronicname[idxelectronic] = inputelectronic(word, 1);
* electronicprice[idxelectronic] = stoi(inputelectronic(word, 2));
* electronicquantity[idxelectronic] = stoi(inputelectronic(word, 3));
* saleelectronic[idxelectronic] = stoi(inputelectronic(word, 4));
* idxelectronic++;
* }
* file.close();
* }
* string inputelectronic(string word, int field)
* {
* int commacount = 1;
* string item;
* for (int idx = 0; idx < word.length(); idx++)
* {
* if (word[idx] == ',')
* {
* commacount++;
* }
* else if (commacount == field)
* {
* item = item + word[idx];
* }
* }
* return item;
* }
* void addproducts()
* {
* system("cls");
* header();
* subMenu("addproduct");
* int option;
* SetConsoleTextAttribute(h, 10);
* cout << "Select one of the following option.." << endl;
* cout << "1.for add the product in books" << endl;
* cout << "2.for add the product in food" << endl;
* cout << "3.for add the product in electronic" << endl;
* cout << "your option is..";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* addbook();
* }
* if (option == 2)
* {
* addfood();
* }
* if (option == 3)
* {
* addelectronic();
* }
* cout << "press any key to continue... ";
* getch();
* }
* void storebook(string bookname, int bookprice, int bookquantity, int salebook)
* {
* fstream file;
* file.open("addbookfile.txt", ios::app);
* file << endl
* << bookname << ',' << bookprice << ',' << bookquantity << ',' << salebook;
* file.close();
* }
* void addbook()
* {
* system("cls");
* header();
* subMenu("addbook");
* SetConsoleTextAttribute(h, 4);
* cout << "Enter the name of book: ";
* cin >> bookname[idxbook];
* cout << "Enter the price of book: ";
* cin >> bookprice[idxbook];
* cout << "Enter the quantity of book: ";
* cin >> bookquantity[idxbook];
* cout << "Enter the sale of book: ";
* cin >> salebook[idxbook];
* SetConsoleTextAttribute(h, 15);
* storebook(bookname[idxbook], bookprice[idxbook], bookquantity[idxbook], salebook[idxbook]);
* idxbook++;
* }
* void addfood()
* {
* system("cls");
* header();
* subMenu("addfood");
* SetConsoleTextAttribute(h, 10);
* cout << "Enter the name of food: ";
* cin >> foodname[idxfood];
* cout << "Enter the price of food: ";
* cin >> foodprice[idxfood];
* cout << "Enter the quantity of food: ";
* cin >> foodquantity[idxfood];
* cout << "Enter the sale of food: ";
* cin >> salefood[idxfood];
* SetConsoleTextAttribute(h, 15);
* storefood(foodname[idxfood], foodprice[idxfood], foodquantity[idxfood], salefood[idxfood]);
* idxfood++;
* }
* void storefood(string foodname, int foodprice, int foodquantity, int salefood)
* {
* fstream file;
* file.open("addfoodfile.txt", ios::app);
* file << endl
* << foodname << ',' << foodprice << ',' << foodquantity << ',' << salefood;
* file.close();
* }
* void addelectronic()
* {
* system("cls");
* header();
* subMenu("addelectronic");
* SetConsoleTextAttribute(h, 10);
* cout << "Enter the name of electric: ";
* cin >> electronicname[idxelectronic];
* cout << "Enter the price of electric: ";
* while (!(cin >> electronicprice[idxelectronic]))
* {
* isnotvalid();
* }
* cout << "Enter the quantity of electric: ";
* while (!(cin >> electronicquantity[idxelectronic]))
* {
* isnotvalid();
* }
* cout << "Enter the sale of electric: ";
* while (!(cin >> saleelectronic[idxelectronic]))
* {
* isnotvalid();
* }
* storeelectronic(electronicname[idxelectronic], electronicprice[idxelectronic], electronicquantity[idxelectronic], saleelectronic[idxelectronic]);
* idxelectronic++;
* }
* void storeelectronic(string electronicname, int electronicprice, int electronicquantity, int saleelectronic)
* {
* fstream file;
* file.open("addelectronicfile.txt", ios::app);
* file << endl
* << electronicname << ',' << electronicprice << ',' << electronicquantity << ',' << saleelectronic;
* file.close();
* }
* void updateproducts()
* {
* system("cls");
* header();
* subMenu("updateproducts");
* int option;
* SetConsoleTextAttribute(h, 4);
* cout << "Select one of the following option.." << endl;
* cout << "1.for update the product in books" << endl;
* cout << "2.for update the product in food" << endl;
* cout << "3.for update the product in electronic" << endl;
* cout << "your option is..";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* updatebook();
* }
* if (option == 2)
* {
* updatefood();
* }
* if (option == 3)
* {
* updateelectronic();
* }
* SetConsoleTextAttribute(h, 10);
* cout << "press any key to continue..";
* SetConsoleTextAttribute(h, 15);
* getch();
* }
* void updatebook()
* {
* system("cls");
* header();
* subMenu("updatebooks");
* bool exist = false;
* SetConsoleTextAttribute(h, 10);
* string bookupdate;
* cout << "enter book name that you want to update: ";
* cin >> bookupdate;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxbook; i++)
* {
* if (bookupdate == bookname[i])
* {
* SetConsoleTextAttribute(h, 10);
* cout << "enter new price:";
* cin >> bookprice[i];
* exist = true;
* break;
* }
* }
* storeupdatebook();
* if (exist == true)
* {
* cout << "update successfully" << endl;
* }
* else
* {
* cout << "update unsuccessfully" << endl;
* cout << "this book has  not exists already" << endl;
* }
* }
* void storeupdatebook()
* {
* fstream file;
* file.open("addbookfile.txt", ios::out);
* for (int idx = 0; idx < idxbook; idx++)
* {
* file << bookname[idx] << ',' << bookprice[idx] << ',' << bookquantity[idx] << ',' << salebook[idx] << endl;
* }
* file.close();
* }
* void updatefood()
* {
* system("cls");
* header();
* subMenu("updatefood");
* string which;
* bool exist;
* SetConsoleTextAttribute(h, 10);
* string foodupdate;
* cout << "enter food name that you want to update: ";
* cin >> foodupdate;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxfood; i++)
* {
* if (foodupdate == foodname[i])
* {
* SetConsoleTextAttribute(h, 10);
* cout << "enter new price:";
* cin >> foodprice[i];
* exist = true;
* break;
* }
* }
* storeupdatefood();
* if (exist == true)
* {
* cout << "update successfully" << endl;
* }
* else
* {
* cout << "update unsuccessfully" << endl;
* cout << "this food has  not exists already" << endl;
* }
* }
* void storeupdatefood()
* {
* fstream file;
* file.open("addfoodfile.txt", ios::out);
* for (int idx = 0; idx < idxfood; idx++)
* {
* file << foodname[idx] << ',' << foodprice[idx] << ',' << foodquantity[idx] << ',' << salefood[idx] << endl;
* }
* file.close();
* }
* void updateelectronic()
* {
* system("cls");
* header();
* subMenu("electronicbooks");
* bool exist;
* SetConsoleTextAttribute(h, 10);
* string electronicupdate;
* cout << "enter device name: ";
* cin >> electronicupdate;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxelectronic; i++)
* {
* if (electronicupdate == electronicname[i])
* {
* SetConsoleTextAttribute(h, 10);
* cout << "enter new price:";
* cin >> electronicprice[i];
* exist = true;
* break;
* }
* }
* storeupdateelectronic();
* if (exist == true)
* {
* cout << "update successfully" << endl;
* }
* else
* {
* cout << "update unsuccessfully" << endl;
* cout << "this device has  not exists already" << endl;
* }
* }
* void storeupdateelectronic()
* {
* fstream file;
* file.open("addelectronicfile.txt", ios::out);
* for (int idx = 0; idx < idxelectronic; idx++)
* {
* file << electronicname[idx] << ',' << electronicprice[idx] << ',' << electronicquantity[idx] << ',' << saleelectronic[idx] << endl;
* }
* file.close();
* }
* void deleteproducts()
* {
* system("cls");
* header();
* subMenu("deleteproducts");
* int option;
* SetConsoleTextAttribute(h, 4);
* cout << "Select one of the following option.." << endl;
* cout << "1.for delete the product in books" << endl;
* cout << "2.for delete the product in food" << endl;
* cout << "3.for delete the product in electronic" << endl;
* cout << "your option is..";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* deletebook();
* }
* if (option == 2)
* {
* deletefood();
* }
* if (option == 3)
* {
* deleteelectronic();
* }
* SetConsoleTextAttribute(h, 10);
* cout << "press any key to continue..";
* SetConsoleTextAttribute(h, 15);
* getch();
* }
* void deletebook()
* {
* system("cls");
* header();
* subMenu("deletebooks");
* string which;
* bool exist;
* SetConsoleTextAttribute(h, 10);
* cout << "Enter which book you want to delete: ";
* cin >> which;
* int find;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxbook; i++)
* {
* if (which == bookname[i])
* {
* find = i;
* exist = true;
* break;
* }
* }
* if (exist == true)
* {
* for (int i = find; i < idxbook; i++)
* {
* bookname[i] = bookname[i + 1];
* bookprice[i] = bookprice[i + 1];
* bookquantity[i] = bookquantity[i + 1];
* salebook[i] = salebook[i + 1];
* }
* idxbook--;
* storedeletebook();
* cout << "delete successfully" << endl;
* }
* else
* {
* cout << "delete unsuccessfully" << endl;
* cout << "this book has  not exists already" << endl;
* }
* }
* void storedeletebook()
* {
* fstream file;
* file.open("addbookfile.txt", ios::out);
* for (int idx = 0; idx < idxbook; idx++)
* {
* file << bookname[idx] << ',' << bookprice[idx] << ',' << bookquantity[idx] << ',' << salebook[idx] << endl;
* }
* file.close();
* }
* void deletefood()
* {
* system("cls");
* header();
* subMenu("deletefoods");
* string which;
* bool exist;
* SetConsoleTextAttribute(h, 10);
* cout << "Enter which food you want to delete: ";
* cin >> which;
* SetConsoleTextAttribute(h, 15);
* int find;
* for (int i = 0; i < idxfood; i++)
* {
* if (which == foodname[i])
* {
* find = i;
* exist = true;
* break;
* }
* }
* if (exist == true)
* {
* for (int i = find; i < idxfood - 1; i++)
* {
* foodname[i] = foodname[i + 1];
* foodprice[i] = foodprice[i + 1];
* foodquantity[i] = foodquantity[i + 1];
* salefood[i] = salefood[i + 1];
* }
* idxfood--;
* storedeletefood();
* cout << "delete successfully" << endl;
* }
* else
* {
* cout << "delete unsuccessfully" << endl;
* cout << "this food has  not exists already" << endl;
* }
* }
* void storedeletefood()
* {
* fstream file;
* file.open("addfoodfile.txt", ios::out);
* for (int idx = 0; idx < idxfood; idx++)
* {
* file << foodname[idx] << ',' << foodprice[idx] << ',' << foodquantity[idx] << ',' << salefood[idx] << endl;
* }
* file.close();
* }
* void deleteelectronic()
* {
* system("cls");
* header();
* subMenu("deleteelectronic");
* string which;
* bool exist;
* SetConsoleTextAttribute(h, 10);
* cout << "Enter which device you want to delete: ";
* cin >> which;
* int find;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxelectronic; i++)
* {
* if (which == electronicname[i])
* {
* find = i;
* exist = true;
* break;
* }
* }
* if (exist == true)
* {
* for (int i = find; i < idxelectronic; i++)
* {
* electronicname[i] = electronicname[i + 1];
* electronicprice[i] = electronicprice[i + 1];
* electronicquantity[i] = electronicquantity[i + 1];
* saleelectronic[i] = saleelectronic[i + 1];
* }
* idxelectronic--;
* storedeleteelectronic();
* cout << "delete successfully" << endl;
* }
* else
* {
* cout << "delete unsuccessfully" << endl;
* cout << "this device has  not exists already" << endl;
* }
* }
* void storedeleteelectronic()
* {
* fstream file;
* file.open("addelectronicfile.txt", ios::out);
* for (int idx = 0; idx < idxelectronic; idx++)
* {
* file << electronicname[idx] << ',' << electronicprice[idx] << ',' << electronicquantity[idx] << ',' << saleelectronic[idx] << endl;
* }
* file.close();
* }
* void display()
* {
* system("cls");
* header();
* subMenu("viewproducts");
* int option;
* SetConsoleTextAttribute(h, 4);
* cout << "Select one of the following option.." << endl;
* cout << "1.for  books" << endl;
* cout << "2.for  food" << endl;
* cout << "3.for  electronic" << endl;
* cout << "your choice is: ";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* displaybook();
* }
* if (option == 2)
* {
* displayfood();
* }
* if (option == 3)
* {
* displayelectronic();
* }
* cout << "press any key to continue... ";
* getch();
* }
* void displaybook()
* {
* system("cls");
* header();
* SetConsoleTextAttribute(h, 10);
* subMenu("viewbooks");
* cout << left << setw(20) << "bookname"
* << left << setw(20)
* << "bookprice"
* << left << setw(20)
* << "bookquantity"
* << left << setw(20)
* << "salebook" << endl;
* for (int i = 0; i < idxbook; i++)
* {
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << bookname[i] << left << setw(20)
* << bookprice[i] << left << setw(20)
* << bookquantity[i] << left << setw(20)
* << salebook[i] << endl;
* }
* SetConsoleTextAttribute(h, 15);
* }
* void displayfood()
* {
* system("cls");
* header();
* SetConsoleTextAttribute(h, 10);
* subMenu("viewfoods");
* cout << left << setw(20) << "foodname"
* << left << setw(20)
* << "foodprice"
* << left << setw(20)
* << "foodquantity" << left << setw(20) << "salefood" << endl;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxfood; i++)
* {
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << foodname[i] << left << setw(20)
* << foodprice[i] << left << setw(20)
* << foodquantity[i] << left << setw(20) << salefood[i] << endl;
* }
* SetConsoleTextAttribute(h, 15);
* }
* void displayelectronic()
* {
* system("cls");
* header();
* subMenu("viewelectricdevice");
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << "electronicname"
* << left << setw(20)
* << "electronicprice"
* << left << setw(20)
* << "electronicquantity"
* << left << setw(20)
* << "saledevice" << endl;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxelectronic; i++)
* {
* SetConsoleTextAttribute(h, 10);
* cout << left << setw(20) << electronicname[i] << left << setw(20)
* << electronicprice[i]
* << left << setw(20) << electronicquantity[i] << left << setw(20) << saleelectronic[i] << endl;
* }
* SetConsoleTextAttribute(h, 15);
* }
* void user()
* {
* system("cls");
* bool run = true;
* while (run)
* {
* system("cls");
* header();
* subMenu("user");
* int option;
* cout << "Select one of the following option.." << endl;
* cout << "1.for search the product detail" << endl;
* cout << "2.for view the product category" << endl;
* cout << "3.for order " << endl;
* cout << "4.for exit" << endl;
* cout << "your option is...";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* if (option == 1)
* {
* search();
* }
* if (option == 2)
* {
* viewdetails();
* }
* if (option == 3)
* {
* order();
* }
* if (option == 4)
* {
* run = false;
* }
* }
* }
* void search()
* {
* system("cls");
* header();
* subMenu("search");
* string option;
* SetConsoleTextAttribute(h, 4);
* cout << "Enter the name of product you want to buy: ";
* cin >> option;
* SetConsoleTextAttribute(h, 15);
* bool count = false;
* for (int idx = 0; idx < idxbook; idx++)
* {
* if (option == bookname[idx])
* {
* count = true;
* break;
* }
* }
* for (int idx = 0; idx < idxfood; idx++)
* {
* if (option == foodname[idx])
* {
* count = true;
* break;
* }
* }
* for (int idx = 0; idx < idxelectronic; idx++)
* {
* if (option == electronicname[idx])
* {
* count = true;
* break;
* }
* }
* if (count == true)
* {
* SetConsoleTextAttribute(h, 10);
* cout << "yes! available" << endl;
* }
* else
* {
* cout << "not available" << endl;
* }
* SetConsoleTextAttribute(h, 15);
* cout << "press any key to continue...";
* getch();
* }
* void viewdetails()
* {
* system("cls");
* header();
* subMenu("view details");
* int option;
* SetConsoleTextAttribute(h, 10);
* cout << "Select one of the following option.." << endl;
* cout << "1.for  books" << endl;
* cout << "2.for  food" << endl;
* cout << "3.for  electronic" << endl;
* cout << "your choice is: ";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* viewbook();
* }
* else if (option == 2)
* {
* viewfood();
* }
* else if (option == 3)
* {
* viewelectronic();
* }
* cout << "press any key to continue... ";
* getch();
* }
* void viewbook()
* {
* system("cls");
* header();
* SetConsoleTextAttribute(h, 10);
* subMenu("viewbooks");
* cout << left << setw(20) << "bookname"
* << left << setw(20)
* << "bookprice"
* << left << setw(20)
* << "bookquantity" << endl;
* for (int i = 0; i < idxbook; i++)
* {
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << bookname[i] << left << setw(20)
* << bookprice[i] << left << setw(20)
* << bookquantity[i] << endl;
* }
* getch();
* SetConsoleTextAttribute(h, 15);
* }
* void viewfood()
* {
* system("cls");
* header();
* SetConsoleTextAttribute(h, 10);
* subMenu("viewfoods");
* cout << left << setw(20) << "foodname"
* << left << setw(20)
* << "foodprice"
* << left << setw(20)
* << "foodquantity" << endl;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxfood; i++)
* {
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << foodname[i]
* << left << setw(20) << foodprice[i]
* << left << setw(20) << foodquantity[i] << endl;
* }
* SetConsoleTextAttribute(h, 15);
* }
* void viewelectronic()
* {
* system("cls");
* header();
* subMenu("viewelectricdevice");
* SetConsoleTextAttribute(h, 4);
* cout << left << setw(20) << "electronicname"
* << left << setw(20)
* << "electronicprice"
* << left << setw(20)
* << "electronicquantity" << endl;
* SetConsoleTextAttribute(h, 15);
* for (int i = 0; i < idxelectronic; i++)
* {
* SetConsoleTextAttribute(h, 10);
* cout << left << setw(20) << electronicname[i]
* << left << setw(20) << electronicprice[i]
* << left << setw(20) << electronicquantity[i] << endl;
* }
* SetConsoleTextAttribute(h, 15);
* }
* void order()
* {
* system("cls");
* header();
* subMenu("order");
* int option;
* SetConsoleTextAttribute(h, 4);
* cout << "Select one of the following option.." << endl;
* cout << "1.for  books" << endl;
* cout << "2.for  food" << endl;
* cout << "3.for  electronic" << endl;
* cout << "your choice is: ";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* SetConsoleTextAttribute(h, 15);
* if (option == 1)
* {
* orderbook();
* }
* else if (option == 2)
* {
* orderfood();
* }
* else if (option == 3)
* {
* orderelectronic();
* }
* cout << "press any key to continue... ";
* getch();
* }
* void orderbook()
* {
* system("cls");
* header();
* subMenu("order book");
* string option;
* int confirm;
* SetConsoleTextAttribute(h, 10);
* cout << "enter name of the product you want to buy: ";
* cin >> option;
* int b;
* bool count = false;
* for (int idx = 0; idx < idxbook; idx++)
* {
* if (option == bookname[idx])
* {
* count = true;
* b = idx;
* break;
* }
* }
* if (count == true)
* {
* cout << "yes! available" << endl;
* SetConsoleTextAttribute(h, 10);
* cout << "press 1 for purchase " << endl;
* cout << "press 2 for cancel " << endl;
* cout << "choose your choice:";
* cin >> confirm;
* if (confirm == 1)
* {
* int quant;
* cout << "Enter  the how namy product you want to buy:  ";
* cin >> quant;
* if (quant < bookquantity[b])
* {
* cout << "yes!available" << endl;
* salebook[b] = salebook[b] + quant;
* bookquantity[b] = bookquantity[b] - quant;
* booksale = booksale + (quant \* bookprice[b]);
* cout << "total bill: " << quant \* bookprice[b] << endl;
* storetotalbooksale();
* storesalebook();
* }
* else
* {
* cout << "not available";
* }
* }
* }
* else
* {
* cout << "not available" << endl;
* }
* }
* void storesalebook()
* {
* fstream file;
* file.open("addbookfile.txt", ios::out);
* for (int idx = 0; idx < idxbook; idx++)
* {
* file << bookname[idx] << ',' << bookprice[idx] << ',' << bookquantity[idx] << ',' << salebook[idx] << endl;
* }
* file.close();
* }
* void orderfood()
* {
* system("cls");
* header();
* subMenu("order food");
* string option;
* int confirm;
* cout << "enter name of the product you want to buy: ";
* cin >> option;
* int b;
* bool count = false;
* for (int idx = 0; idx < idxfood; idx++)
* {
* if (option == foodname[idx])
* {
* count = true;
* b = idx;
* break;
* }
* }
* if (count = true)
* {
* cout << "yes! available" << endl;
* cout << "press 1 for purchase " << endl;
* cout << "press 2 for cancel " << endl;
* cout << "choose your choice:";
* while (!(cin >> confirm))
* {
* isnotvalid();
* }
* if (confirm == 1)
* {
* int quant;
* cout << "Enter  the how namy product you want to buy:  ";
* while (!(cin >> quant))
* {
* isnotvalid();
* }
* if (quant < foodquantity[b])
* {
* cout << "yes! AVAILABLE" << endl;
* salefood[b] = salefood[b] + quant;
* foodquantity[b] = foodquantity[b] - quant;
* foodsale = foodsale + (quant \* foodprice[b]);
* cout << "total bill: " << quant \* foodprice[b] << endl;
* storesalefood();
* storetotalfoodsale();
* }
* else
* {
* cout << "not available";
* }
* }
* }
* else
* {
* cout << "not available" << endl;
* }
* }
* void storesalefood()
* {
* fstream file;
* file.open("addfoodfile.txt", ios::out);
* for (int idx = 0; idx < idxfood; idx++)
* {
* file << foodname[idx] << ',' << foodprice[idx] << ',' << foodquantity[idx] << ',' << salefood[idx] << endl;
* }
* file.close();
* }
* void orderelectronic()
* {
* system("cls");
* header();
* subMenu("order electricdevice");
* string option;
* int confirm;
* cout << "enter name of the product you want to buy: ";
* cin >> option;
* int b;
* bool count = false;
* for (int idx = 0; idx < idxelectronic; idx++)
* {
* if (option == electronicname[idx])
* {
* count = true;
* b = idx;
* break;
* }
* }
* if (count = true)
* {
* cout << "yes! available" << endl;
* cout << "press 1 for purchase " << endl;
* cout << "press 2 for cancel " << endl;
* cout << "choose your choice:";
* while (!(cin >> confirm))
* {
* isnotvalid();
* }
* if (confirm == 1)
* {
* int quant;
* cout << "Enter  the how namy product you want to buy:  ";
* while (!(cin >> quant))
* {
* isnotvalid();
* }
* if (quant < electronicquantity[b])
* {
* cout << "yes! available" << endl;
* saleelectronic[b] = saleelectronic[b] + quant;
* electronicquantity[b] = electronicquantity[b] - quant;
* electronicsale = electronicsale + (quant \* electronicprice[b]);
* cout << "total bill: " << quant \* electronicprice[b] << endl;
* storesaleelectronic();
* storetotalelectronicsale();
* }
* }
* }
* else
* {
* cout << "not available" << endl;
* }
* }
* void storesaleelectronic()
* {
* fstream file;
* file.open("addelectronicfile.txt", ios::out);
* for (int idx = 0; idx < idxelectronic; idx++)
* {
* file << electronicname[idx] << ',' << electronicprice[idx] << ',' << electronicquantity[idx] << ',' << saleelectronic[idx] << endl;
* }
* file.close();
* }
* void allsalebook()
* {
* system("cls");
* header();
* subMenu("veiw booksale");
* cout << left << setw(20) << "booksale"
* << left << setw(20)
* << "bookquantity" << endl;
* for (int i = 0; i < idxbook; i++)
* {
* cout << left << setw(20) << salebook[i]
* << left << setw(20) << bookquantity[i] << endl;
* }
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* cout << "total book sale: " << booksale << endl;
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* getch();
* }
* void storetotalbooksale()
* {
* fstream file;
* file.open("salebook.txt", ios::out);
* file << booksale;
* file.close();
* }
* void loadtotalbooksale()
* {
* fstream file;
* file.open("salebook.txt", ios::in);
* file >> booksale;
* file.close();
* }
* void allsalefood()
* {
* system("cls");
* header();
* subMenu("view foodsale");
* cout << left << setw(20) << "foodsale"
* << left << setw(20)
* << "foodquantity" << endl;
* for (int i = 0; i < idxfood; i++)
* {
* cout << left << setw(20) << salefood[i]
* << left << setw(20) << foodquantity[i] << endl;
* }
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* cout << "total food sale: " << foodsale << endl;
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* getch();
* }
* void storetotalfoodsale()
* {
* fstream file;
* file.open("salefood.txt", ios::out);
* file << foodsale;
* file.close();
* }
* void loadtotalfoodsale()
* {
* fstream file;
* file.open("salefood.txt", ios::in);
* file >> foodsale;
* file.close();
* }
* void allsaleelectronic()
* {
* system("cls");
* header();
* subMenu("veiw device sale");
* cout << left << setw(20) << "electronicsale"
* << left << setw(20)
* << "electronicquantity" << endl;
* for (int i = 0; i < idxelectronic; i++)
* {
* cout << left << setw(20) << saleelectronic[i]
* << left << setw(20) << electronicquantity[i] << endl;
* }
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* cout << "total electronic sale: " << electronicsale << endl;
* cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;
* getch();
* }
* void storetotalelectronicsale()
* {
* fstream file;
* file.open("saleelectronic.txt", ios::out);
* file << electronicsale;
* file.close();
* }
* void loadtotalelectronicsale()
* {
* fstream file;
* file.open("saleelectronic.txt", ios::in);
* file >> electronicsale;
* file.close();
* }
* void totalsale()
* {
* system("cls");
* header();
* subMenu("veiw total sale");
* int option;
* cout << "Select one of the following option.." << endl;
* cout << "1.for  booksale" << endl;
* cout << "2.for  foodsale" << endl;
* cout << "3.for  electronicsale" << endl;
* cout << "4.for allsale" << endl;
* cout << "your choice is: ";
* while (!(cin >> option))
* {
* isnotvalid();
* }
* if (option == 1)
* {
* allsalebook();
* }
* else if (option == 2)
* {
* allsalefood();
* }
* else if (option == 3)
* {
* allsaleelectronic();
* }
* else if (option == 4)
* {
* allsale();
* }
* cout << "press any key to continue...";
* getch();
* }
* void allsale()
* {
* system("cls");
* header();
* subMenu("all sale");
* income = booksale + foodsale + electronicsale;
* cout << " total sale of all category: " << income << endl;
* storeincome();
* loadincome();
* }
* void loadincome()
* {
* fstream file;
* file.open("income.txt", ios::in);
* file >> income;
* file.close();
* }
* void storeincome()
* {
* fstream file;
* file.open("income.txt", ios::out);
* file << income;
* file.close();
* }
* void isnotvalid()
* {
* cout << "error: entera number";
* cin.clear();
* cin.ignore(123, '\n');
* }
* **Weakness in the Business Application**
  + Write the weaknesses that you think your business application has.
* **Future Directions**
  + Give the future directions that you want to do to improve your business application in the next semester.

**Formatting Instructions**

1. Heading Size is 16
2. Sub heading size is 14
3. Further heading size is 13
4. Make your heading font bold
5. Text Font size in the paragraph is 12
6. Use Times New Roman Font Style
7. Text paragraphs should be justified. (Justify is feature of MS World)
8. Your code line spacing should be 1. Also remove the spacing before and after the paragraph in your code to make it compact in the word file.

**Other General Instructions**

1. Attach 2 Rubrics after the documentation. Your evaluation will be done according to the rubrics.
2. Rubric format is provided below.
3. Make sure all components in the rubric are present in your code and documentation.
4. Rubric should be on single page (Do not spread it to two pages)
5. Final Documentation should be submitted in Tape Binding.

**Student Reg. No. :**   **Student Name.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A-Extensive Evidence** | **B-Convincing Evidence** | **C-Limited Evidence** | **D-No Evidence** |
| Documentation Formatting  **Grade:** | All the documentation meets all the criteria. | Documentation is well formatted but some of the criteria is not fulfilled. | Documentation is required a lot of improvement. | Documentation is not Available |
| **Documentation Formatting Criteria:** In **Binder**, **Title** Page, **Header**-Footers, Font **Style**, Font **Size** all are all consistence and according to given **guidelines**. Project **Poster** is professionally design and well presented | | | | |
| Documentation Contents  **Grade:** | Documentation includes all of the criteria. | Documentation meet more than 80% of the criteria given. | Documentation meet more than 50% of the criteria. | When the documentation meet less than 50% of the criteria. |
| **Documentation Contents Criteria:** **Title** Page - **Table** of Contents - Project **Abstract** - **Functional** Requirements - **Wire** Frames –**Data Flow** Diagram-**Data** Structure (Arrays)-**Function** Headers and Description -Project **Code.** - **Weakness** in the Project and **Future** Directions. - **Conclusion** and What your **Learn** from the Project and Course and What is your **Future** Planning. | | | | |
| Project Complexity  **Grade:** | Project has at least 2 user’s types and each user has at least 5 functionalities. | Project complexity meet 80% criteria given in extensive evidence | Project complexity meet 50% criteria given in extensive evidence | Project complexity meet less than 50% criteria given in extensive evidence |
| Code Style  **Grade:** | All Code style criteria is followed | All code style criteria followed but some improvements required | lot of improvements required in coding style. | **Did not follow** code style, |
| **Code Style Criteria:**  Consistent code style. Code is well indented. Variable and Function names are well defined.  White Spaces are well used. Comments are added. | | | | |
| Code Documentation Mapping  **Grade:** | Code and documentation is synchronized. | Code and documentation does not synchronized at **some** places | Code and documentation does not synchronized at **many** places | Code and documentation **does not** synchronized. |
| Data Structure (Arrays)  **Grade:** | Data structure is sufficient for the project requirements | Data Structure is sufficient but require improvement to meet project requirements. | Data structure is not sufficient and need a lot of improvement | Data Structure is not properly identified and declared. |
| Modularity  **Grade:** | Meet all Modularity criteria | Meet all Modularity criteria but at some places it is missing | Do not sufficiently meet the modularity criteria. | No modularity or very minimum modularity. |
| **Modularity criteria:** Functions are defined for each major feature. Functions are independent (identify from parameter list and return types). | | | | |
| Validations  **Grade:** | Validations on all number type inputs are applied | Validations are applied but at some places it is missing. | Validations are missing at lot of places | No Validations are used |
| File Handling  **Grade:** | Separate files for separate data. Data in csv format | File handing require some improvements | File handing require a lot of improvements | Not implemented |
| Aesthetics of the User Interface  **Grade:** | UI is presentable. Proper coloring, Headers and clear screen is done | UI require some improvements | UI require a lot of improvements | Not implemented |
| Presentation and Demo  **Grade:** | Presentation and Demo was 100% working | Presentation and Demo require some improvements | Presentation and Demo require a lot of improvements | Presentation was not ok and Demo was not working |
| Student Understanding with the Code.  **Grade:** | Student has complete understanding how the code is working and knows the concept. | Student has good understand but some place he does not know the concepts | Student has a very little understand and lack the major concepts. | Student does not have any level of understanding of the code. |

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