**Book Store Management System**

**(City Book Center)**



Session: 2022 – 2026

**Submitted by:**

Saqlain Mansab 2022-CS-80

**Supervised by:**

Mam Maida Shahid Sir Irzam Liaquat

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[Short Description 3](#_Toc128733447)

[Users of Application 3](#_Toc128733448)

[Functional Requirements 3](#_Toc128733449)

[Wireframes 4](#_Toc128733450)

[Data Structures (Parallel Arrays) 9](#_Toc128733451)

[Function Prototypes 10](#_Toc128733452)

[Functions Working Flow 13](#_Toc128733453)

[Complete Code of Business Application 14](#_Toc128733455)

[Weakness in the Business Application 49](#_Toc128733456)

[Future Directions 49](#_Toc128733457)

Short Description

This project is done to manage a Book Store. Sorting the books and then managing them is a difficult process and it requires a lot of hard work. This project will aid the shop owners to manage their book stores. In addition to that, it will keep track of their daily and monthly income. It will provide the users and customers a generic outlook of the books present on the shop. Moreover, it will make them available with the current price of books with the discount offered by the shopkeepers. The admin will be able to get a computerized receipt of the purchased books. At the end, the customer will also be able to give suggestions and reviews related to his/her experience. The user will also be able to lend a book for specified period of time. The cost will vary for lending time. The program will issue a username and password for first time and then recognize the credentials for next time.

Users of Application

1. **Admin/Owner**

The admin will be the main user of the program. He will provide the main content of books. He can update the price of books and offer different discount according to his will. He will also be able to calculate the payable amount with the help of program.

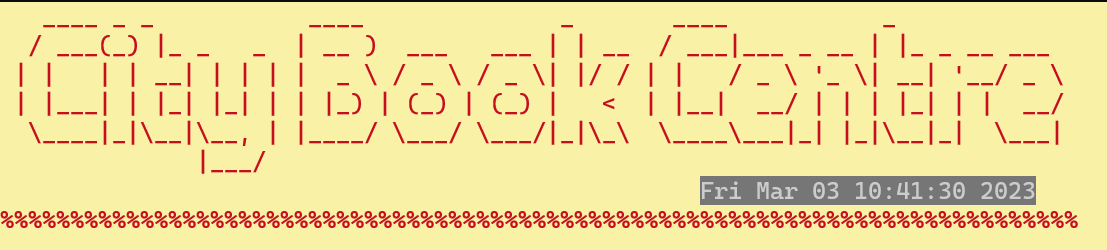
1. **User/Customer**

The user can also temporarily use the program. He will be able to view the list of books presented by the admin. He can mark his favorite books and purchase them. He will be able to track his orders and give suggestions about the Book Shop.

Functional Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **User**  **ID** | **As a** | **I want to** | **So that I can** |
| **1.** | **Admin** | 1. View the Available books on the Shop 2. Update the books available 3. Update the price and quantity of books 4. Take Reviews from user 5. View the Purchased books 6. Check the daily earning | Manage my Book Store and track my income |
| **2.** | **User** | 1. View the list of provided books 2. Purchase a book 3. Lend a book 4. Give Suggestion and reviews | Approach my favorite books and utilize them easily |

Wireframes

****

**Figure 1: Header**

**A picture containing table

Description automatically generated**

**Figure 2: LogIn Screen**

**Table

Description automatically generated**

**Figure 3: SignUp Menu**

**Table

Description automatically generated**

**Figure 4: SignIn Menu**

**Text

Description automatically generated**

**Figure 5: Admin Main Menu**

**Text

Description automatically generated**

**Figure 6: User Main Menu**

**Timeline

Description automatically generated with medium confidence**

**Figure 7: List of BooksTimeline

Description automatically generated with low confidence**

**Figure 8: Instructions**

**A picture containing text

Description automatically generated**

**Figure 9: Add a Book**

**Timeline

Description automatically generated**

**Figure 10: Income Statistics**

**Text

Description automatically generated**

**Figure 11: History of User**

Data Structures (Parallel Arrays)

string users[100]; // name of users registered in the program

string passwords[100]; // password of users registered in the program

string roles[100]; // roles of users registered in the program

string refUserName ; // this is the username of user which is currently working in the program

string refUserPassword ; // this is the password of user which is currently working in the program

int usercount = 0;

string wholePurchasedBooksNames[100]; // name of whole books present in carts of all users

int wholePurchasedBooksPrice[100]; // price of whole books present in carts of all users

string wholePurchasedBooksUser[100]; // name of whole user having books in cart

string wholePurchasedBooksPassword[100]; // passwords of above users

int wholePurchasedBooksArrayCount = 0;

string wholeBoughtBooksNames[100]; // name of whole books present in carts of all users

int wholeBoughtBooksPrice[100]; // price of whole books present in carts of all users

string wholeBoughtBooksUser[100]; // name of whole user having books in cart

string wholeBoughtBooksPassword[100]; // passwords of above users

int wholeBoughtBooksArrayCount = 0;

string bookNames[100]; // these are the active books in the program

int bookPrice[100]; // these are the active books prices in the program

int bookStock[100]; // these are the active books quantity in the program

int bookLendingCost[100]; // these are the active books lending cost in the program

int booksArrayCount = 0;

string purchasedBooks[100]; // list of purchased books by a specific user

int priceOfPurchasedBooks[100]; // price of purchased books by a specific user

int purchasedBooksCount = 0;

string boughtBooks[100]; // list of bought books by a specific user

int priceOfBoughtBooks[100]; // price of bought books by a specific user

int boughtBooksCount = 0;

string suggestionArray[100]; // to store text of suggestions

string suggestionBookArray[100]; // to store respective book with the text

int suggestionCount = 0;

Function Prototypes

**// Main Functions**

void header();

int menu();

int adminMenu();

int userMenu();

string signIn(string userName, string password2);

bool signUp(string user, string password, string role);

**//check Functions**

bool checkaddBook(string name, int price, int stock, int lendingPrice);

bool checkStringCondition(string word);

bool checkIntegerCondition(string word);

**// File Handling functions**

void addUserDatafromFile();

void addWholePurchasedBooksDatafromFile();

void addWholeBoughtBooksDatafromFile();

void addBoughtBookDatafromFile(string username, string password);

void addPurchasedBookDatafromFile(string username, string password);

void transferDatatoNextfile(string username, string password);

void addBookDatafromFile();

void updateBooksDatainFile();

void loadIncomeData();

void loadSuggestions();

**// Books related Functions**

void listOfBooks();

void addBook(string name, int price, int stock, int lendingCost);

void updateBook(string name, int price, int stock, int lendingCost);int removeBook(int idx);

bool findBook(string checkBook);

void purchaseBook();

**// Process Functions**

void giveSuggestion();

int incomeCalculation();

**// Animation functions for Instruction Menu**

void printInstructions();

void moveInstructions();

void eraseInstructions();

**// Animation functions for Suggestion Menu**

void printSuggestionBox();

void eraseSuggestionBox();

void moveSuggestionBox();

**// Helping Functions**

void clearScreen();

void clearScreen2();

void subMenubeforeMainMenu(string subMenu);

void subMenu(string subMenu);

void printTime();

void doColor(int index);

void establishColor();

void printProgressBar(int x, int y);

string parseData(string data, int field);

void gotoxy(int x, int y);

void printVerticalLine(int x, int y, int limit);

void printHorizontalLine(int x, int y, int limit);

Functions Working Flow

**Main Menu Flow**

**Admin Menu Flow**

**User Menu Flow**

Complete Code of Business Application

**Note**: The code provided is just content of the program. Proper Indentation is removed to manage it on pages.

// Function Definitions

int main()

{

addUserDatafromFile();

addBookDatafromFile();

addWholePurchasedBooksDatafromFile();

addWholeBoughtBooksDatafromFile();

loadIncomeData();

loadSuggestions();

int loginOption = 0; // deal with signUp and signIn Function

int admin, user; // Main Menu Function for both Admin and user

establishColor();

clearScreen2();

while (loginOption != 3)

{

subMenubeforeMainMenu("Login");

loginOption = menu();

clearScreen2();

// signUp

if (loginOption == 1)

{

string userSignUp, passwordSignUp, role;

bool isPresent;

subMenubeforeMainMenu("SignUp");

cout << " U S E R N A M E (WOS): " << endl;

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

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

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

cout << endl;

cout << " P A S S W O R D (must be 8 characters): " << endl;

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

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

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

cout << endl;

cout << " R O L E (Admin/User) : " << endl;

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

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

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

cout << endl;

// take name at provided reference

cin.ignore();

gotoxy(18, 14);

getline(cin, userSignUp);

// take password at provided reference

gotoxy(18, 19);

cin >> passwordSignUp;

cin.ignore();

// take role at provided reference

gotoxy(18, 24);

cin >> role;

cout << endl;

clearScreen();

subMenubeforeMainMenu("Processing");

isPresent = signUp(userSignUp, passwordSignUp, role);

int lengthOfPassword = passwordSignUp.length();

if (isPresent == false)

{

if (role == "Admin" || role == "User")

{

if (lengthOfPassword <= 8)

{

printProgressBar(20, 15);

users[usercount] = userSignUp;

passwords[usercount] = passwordSignUp;

roles[usercount] = role;

usercount++;

fstream file;

file.open("UsersData.txt", ios::app);

file << userSignUp << ",";

file << passwordSignUp << ",";

file << role << endl;

file.close();

cout << endl;

cout << endl;

cout << "User is Saved successfully" << endl;

}

else

{

cout << endl;

cout << "Please give password of maximum 8 characters" << endl;

cout << endl;

cout << "Please Try Again" << endl;

}

}

else

{

cout << endl;

cout << "Please give a Valid Role" << endl;

cout << endl;

cout << "Please Try Again" << endl;

}

}

else

{

cout << endl;

cout << "The User is already Present." << endl;

cout << endl;

cout << "Please Try Again" << endl;

}

cout << endl;

clearScreen();

}

// signIn

else if (loginOption == 2)

{

clearScreen2();

string userSignIn, passwordSignIn, roleFound;

subMenubeforeMainMenu("SignIn");

cout << " U S E R N A M E : " << endl;

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

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

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

cout << endl;

cout << endl;

cout << " P A S S W O R D : " << endl;

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

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

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

cout << endl;

cout << endl;

// take name at provided reference

gotoxy(18, 14);

cin >> userSignIn;

// take password at provided reference

gotoxy(18, 20);

cin >> passwordSignIn;

// store in separate variable to be used further

refUserName = userSignIn;

refUserPassword = passwordSignIn;

roleFound = signIn(userSignIn, passwordSignIn);

// Admin Menu

if (roleFound == "Admin")

{

admin = 0;

while (admin != 9)

{

clearScreen2();

subMenubeforeMainMenu("Admin Main");

doColor(135);

gotoxy(65, 9);

cout << "[ " << refUserName << " ]";

gotoxy(0, 12);

doColor(228);

// create a animated menu for Admin

printVerticalLine(0, 11, 16); // left most line

printHorizontalLine(1, 11, 60); // upper most line

printVerticalLine(61, 11, 16); // right most line

printHorizontalLine(1, 26, 60); // lower most line

printHorizontalLine(1, 13, 60); // line below header

establishColor();

admin = adminMenu();

clearScreen();

// List of Books

if (admin == 1)

{

subMenu("List Of Books");

listOfBooks();

clearScreen();

}

// Add a Book

else if (admin == 2)

{

bool bookadding = true;

bool bookAddingSuggestion;

while (bookadding)

{

string name;

string price, stock, lendingCost;

subMenu("Add a Book");

doColor(10);

cout << " A D D A B O O K " << endl;

cout << endl;

establishColor();

cout << "Enter the Name of Book : ";

getline(cin.ignore(), name);

cout << endl;

cout << "Enter the Price of Book : ";

cin >> price;

cout << endl;

cout << "Enter the Quantity of Book in Stock : ";

cin >> stock;

cout << endl;

cout << "Enter the Cost for Lending of Book(/d) : ";

cin >> lendingCost;

cout << endl;

if (checkIntegerCondition(price) && checkIntegerCondition(stock) && checkIntegerCondition(lendingCost))

{

bookAddingSuggestion = checkaddBook(name, stoi(price), stoi(stock), stoi(lendingCost));

if (bookAddingSuggestion == false)

{

addBook(name, stoi(price), stoi(stock), stoi(lendingCost));

}

else

{

cout << "The book is already Present" << endl;

cout << endl;

}

string sugg;

cout << "Do You want to Enter Again(Yes/No) : ";

cin >> sugg;

if (sugg == "No")

{

bookadding = false;

}

else if (sugg == "Yes")

{

}

else

{

cout << endl;

cout << "Invalid Value Entered" << endl;

cout << endl;

cout << "Loop terminated! Try Again" << endl;

bookadding = false;

}

}

else

{

cout << "You have entered Invalid Prices! Try Again" << endl;

bookadding = false;

}

cout << endl;

clearScreen();

}

}

// Update a Book

else if (admin == 3)

{

subMenu("Update a Book");

doColor(10);

cout << " U P D A T E A B O O K " << endl;

cout << endl;

establishColor();

string updateBookGiven;

cout << "Enter the Name of Book you want to Update : ";

cin >> updateBookGiven;

cout << endl;

bool updateBookPresent;

updateBookPresent = findBook(updateBookGiven);

if (updateBookPresent == true)

{

cout << "The book Founded Successfully" << endl;

cout << endl;

clearScreen();

subMenu("Update a Book");

doColor(10);

cout << " U P D A T E A B O O K " << endl;

cout << endl;

establishColor();

string name;

int price, stock, Lcost;

cout << "Enter the new Name of Book : ";

cin >> name;

cout << endl;

cout << "Enter the new Price of Book : ";

cin >> price;

cout << endl;

cout << "Enter the new Quantity of Book : ";

cin >> stock;

cout << endl;

cout << "Enter the new Lending Cost of Book : ";

cin >> Lcost;

cout << endl;

updateBook(name, price, stock, Lcost);

updateBooksDatainFile();

cout << "The book is Updated Successfully" << endl;

cout << endl;

}

else

{

cout << "Book Not Found" << endl;

}

clearScreen();

}

// Remove a Book

else if (admin == 4)

{

subMenu("Remove a Book");

string removalBook;

bool removalBookPresent;

doColor(10);

cout << " R E M O V E A B O O K " << endl;

cout << endl;

establishColor();

cout << "Enter the Name of Book : ";

cin >> removalBook;

cout << endl;

removalBookPresent = findBook(removalBook);

if (removalBookPresent == true)

{

cout << "Book found successfully" << endl;

cout << endl;

string option;

cout << "Do you surely want to remove Book(Yes/No): ";

cin >> option;

cout << endl;

if (option == "Yes")

{

removeBook(idx);

updateBooksDatainFile();

cout << "Book removed Successfully" << endl;

}

else if (option == "No")

{

cout << "Access Denied" << endl;

}

else

{

cout << "Invalid Value Entered" << endl;

}

cout << endl;

}

else

{

cout << "Book Not Found" << endl;

}

clearScreen();

}

// whole Purchased books

else if (admin == 5)

{

doColor(10);

cout << "P U R C H A S E D B O O K S" << endl;

cout << endl;

establishColor();

cout << "Following are the Books purchased till Now" << endl;

cout << endl;

doColor(240);

cout << left << setw(10) << "Sr No." << setw(15) << "Username" << setw(15) << "Name" << setw(15) << "Price" << endl;

cout << endl;

establishColor();

for (int z = 0; z < wholeBoughtBooksArrayCount; z++)

{

cout << left << setw(10) << z + 1 << setw(15) << wholeBoughtBooksUser[z] << setw(15) << wholeBoughtBooksNames[z] << setw(15) << wholeBoughtBooksPrice[z] << endl;

}

cout << endl;

clearScreen();

}

// View Suggestions

else if (admin == 6)

{

subMenu("Reviews Section");

cout << endl;

doColor(240);

cout << left << setw(10) << "Sr No." << setw(15) << "Book Name" << setw(15) << "Review" << endl;

establishColor();

cout << endl;

for (int z = 0; z < suggestionCount; z++)

{

cout << left << setw(10) << z + 1 << setw(15) << suggestionBookArray[z] << setw(15) << suggestionArray[z] << endl;

}

cout << endl;

clearScreen();

}

// Instructions

else if (admin == 7)

{

subMenu("Instructions");

moveInstructions();

cout << endl;

clearScreen();

}

// Income Statistics

else if (admin == 8)

{

subMenu("Income Statistics");

incomeCalculation();

clearScreen();

}

}

}

// User Menu

else if (roleFound == "User")

{

user = 0;

purchasedBooksCount = 0;

addPurchasedBookDatafromFile(refUserName, refUserPassword);

while (user != 7)

{

clearScreen2();

subMenubeforeMainMenu("User Main");

doColor(135);

gotoxy(65, 9);

cout << "[ " << refUserName << " ]";

gotoxy(0, 12);

printVerticalLine(0, 11, 14); // left most line

printHorizontalLine(1, 11, 60); // upper most line

printVerticalLine(61, 11, 14); // right most line

printHorizontalLine(1, 24, 60); // lower most line

printHorizontalLine(1, 13, 60); // line below header

establishColor();

user = userMenu();

clearScreen();

// view books

if (user == 1)

{

subMenu("List of Books");

listOfBooks();

clearScreen();

}

// Purchase a Book

else if (user == 2)

{

subMenu("Purchase a Book");

string book;

bool presence;

string purchasingOption;

cout << endl;

cout << "Enter the name of Book: ";

cin >> book;

cout << endl;

presence = findBook(book);

if (presence == true)

{

cout << "Book Found Successfully" << endl;

cout << endl;

cout << "Following are its details" << endl;

cout << endl;

// cout <<"This is the index of Book in array: "<< idx;

cout << left << setw(10) << "Sr No." << setw(15) << "Name" << setw(15) << "Qty" << setw(15) << "Price" << setw(15) << "Lending Cost" << endl;

cout << endl;

cout << left << setw(10) << idx + 1 << setw(15) << bookNames[idx] << setw(15) << bookStock[idx] << setw(15) << bookPrice[idx] << setw(15) << bookLendingCost[idx] << endl;

cout << endl;

cout << "Do you want to Purchase this Book(Yes/No): ";

cin >> purchasingOption;

cout << endl;

if (purchasingOption == "Yes")

{

if (bookStock[idx] > 0)

{

purchaseBook();

}

else

{

cout << "Book Sold Out. Visit Next Time!!! Thank You" << endl;

}

}

else if (purchasingOption == "No")

{

cout << "Access Denied" << endl;

}

else

{

cout << "Invalid Option" << endl;

}

}

else

{

cout << "Book not Found";

cout << endl;

}

cout << endl;

clearScreen();

}

// Cart

else if (user == 3)

{

subMenu("Cart");

cout << endl;

int total = 0;

cout << "Following are the books added to the Cart" << endl;

cout << endl;

doColor(240);

cout << left << setw(10) << "Sr No." << setw(15) << "Name" << setw(15) << "Price" << endl;

cout << endl;

establishColor();

for (int z = 0; z < purchasedBooksCount; z++)

{

cout << left << setw(10) << z + 1 << setw(15) << purchasedBooks[z] << setw(15) << priceOfPurchasedBooks[z] << endl;

total += priceOfPurchasedBooks[z];

// cout << left << setw(10) << z + 1 << setw(15) << purchasedBooks[z] + 1 << setw(15) << bookNames[purchasedBooks[z]] << setw(15) << bookStock[purchasedBooks[z]] << setw(15) << bookPrice[purchasedBooks[z]] << setw(15) << bookLendingCost[purchasedBooks[z]] << endl;

// total += bookPrice[purchasedBooks[z]];

}

cout << endl;

cout << right << setw(25) << "Total Amount: " << total << endl;

cout << endl;

cout << "SPACEBAR = Purchase || ESCAPE / ELSE = Quit" << endl;

cout << endl;

getch();

// Payment Method

if (GetAsyncKeyState(VK\_SPACE))

{

// cout << " Purchase" << endl;

clearScreen();

subMenu("Cart > Payment");

cout << endl;

int paymentOption;

cout << "Select a Payment Method" << endl;

cout << endl;

cout << "1. Online Payment" << endl;

cout << endl;

cout << "2. Cash on Delivery" << endl;

cout << endl;

cout << "Your Option.....";

cin >> paymentOption;

cout << endl;

// Online Payment

if (paymentOption == 1)

{

// cout << "Online Payment Menu" << endl;

clearScreen();

subMenu("Cart > Payment > Online Payment");

cout << endl;

int onlinePaymentOption;

cout << "Select a Payment Method" << endl;

cout << endl;

cout << "1. Bank Transfer" << endl;

cout << endl;

cout << "2. EasyPaisa / JazzCash" << endl;

cout << endl;

cout << "Your Option.....";

cin >> onlinePaymentOption;

cout << endl;

// Bank Transfer

if (onlinePaymentOption == 1)

{

string bankNumber;

string bankAccountPassword;

cout << "Bank Transfer" << endl;

cout << endl;

cout << "Enter the Account Number(WOD): ";

cin >> bankNumber;

cout << endl;

cout << "Enter the Account Password: ";

cin >> bankAccountPassword;

cout << endl;

if (checkIntegerCondition(bankNumber))

{

transferDatatoNextfile(refUserName, refUserPassword);

cout << "Order Placed Successfully" << endl;

}

else

{

cout << "Give a Valid Account Number" << endl;

}

}

// EasyPaisa

else if (onlinePaymentOption == 2)

{

string cellNumber;

string cellAccountPassword;

cout << "EasyPaisa / JazzCash" << endl;

cout << endl;

cout << "Enter the Account Number(WOD): ";

cin >> cellNumber;

cout << endl;

cout << "Enter the Account Password: ";

cin >> cellAccountPassword;

cout << endl;

if (checkIntegerCondition(cellNumber))

{

transferDatatoNextfile(refUserName, refUserPassword);

cout << "Order Placed Successfully" << endl;

}

else

{

cout << "Give a Valid Account Number" << endl;

}

}

// cout << endl;

}

// Cash on Delivery

else if (paymentOption == 2)

{

cout << "The cash will be charged on Delivery" << endl;

cout << endl;

transferDatatoNextfile(refUserName, refUserPassword);

cout << "Order Placed Successfully" << endl;

}

cout << endl;

}

// Quiting Menu

else if (GetAsyncKeyState(VK\_ESCAPE))

{

cout << " The Order is Cancelled Successfully " << endl;

cout << endl;

}

else

{

cout << " The Order is Cancelled Successfully " << endl;

cout << endl;

}

clearScreen();

}

// Lend a Book

else if (user == 4)

{

subMenu("Lend a Book");

string book;

bool presence;

string lendingOption;

cout << endl;

cout << "Enter the name of Book: ";

cin >> book;

cout << endl;

presence = findBook(book);

if (presence == true)

{

cout << "Book Found Successfully" << endl;

cout << endl;

cout << "Following are its details" << endl;

cout << endl;

cout << left << setw(10) << "Sr No." << setw(15) << "Name" << setw(15) << "Qty" << setw(15) << "Price" << setw(15) << "Lending Cost" << endl;

cout << endl;

cout << left << setw(10) << idx + 1 << setw(15) << bookNames[idx] << setw(15) << bookStock[idx] << setw(15) << bookPrice[idx] << setw(15) << bookLendingCost[idx] << endl;

cout << endl;

cout << "Do you want to Lend this Book(Yes/No): ";

cin >> lendingOption;

cout << endl;

if (lendingOption == "Yes")

{

if (bookStock[idx] > 0)

{

clearScreen();

subMenu("Lend a Book > Payment");

cout << endl;

int paymentOption;

cout << "Select a Payment Method" << endl;

cout << endl;

cout << "1. Online Payment" << endl;

cout << endl;

cout << "2. Cash on Delivery" << endl;

cout << endl;

cout << "Your Option.....";

cin >> paymentOption;

cout << endl;

// Online Payment

if (paymentOption == 1)

{

// cout << "Online Payment Menu" << endl;

clearScreen();

subMenu("Lend a Book > Payment > Online Payment");

cout << endl;

int onlinePaymentOption;

cout << "Select a Payment Method" << endl;

cout << endl;

cout << "1. Bank Transfer" << endl;

cout << endl;

cout << "2. EasyPaisa / JazzCash" << endl;

cout << endl;

cout << "Your Option.....";

cin >> onlinePaymentOption;

cout << endl;

// Bank Transfer

if (onlinePaymentOption == 1)

{

string bankNumber;

string bankAccountPassword;

cout << "Bank Transfer" << endl;

cout << endl;

cout << "Enter the Account Number(WOD): ";

cin >> bankNumber;

cout << endl;

cout << "Enter the Account Password: ";

cin >> bankAccountPassword;

cout << endl;

if (checkIntegerCondition(bankNumber))

{

cout << "Your Order is posted Successfully" << endl;

bookStock[idx]--;

}

else

{

cout << "Give a Valid Account Number" << endl;

}

}

// EasyPaisa

else if (onlinePaymentOption == 2)

{

string cellNumber;

string cellAccountPassword;

cout << "EasyPaisa / JazzCash" << endl;

cout << endl;

cout << "Enter the Account Number(WOD): ";

cin >> cellNumber;

cout << endl;

cout << "Enter the Account Password: ";

cin >> cellAccountPassword;

cout << endl;

if (checkIntegerCondition(cellNumber))

{

cout << "Your Order is posted Successfully" << endl;

bookStock[idx]--;

}

else

{

cout << "Give a Valid Account Number" << endl;

}

}

// cout << endl;

}

// Cash on Delivery

else if (paymentOption == 2)

{

cout << "The cash will be charged on Delivery" << endl;

cout << endl;

cout << "Your Order is posted Successfully" << endl;

bookStock[idx]--;

}

}

else

{

cout << "The Book is not Available in Stock. Sorry for Inconvenience" << endl;

}

}

else if (lendingOption == "No")

{

cout << "Access Denied" << endl;

}

else

{

cout << "Invalid Option" << endl;

}

}

else

{

cout << "Book not Found";

}

cout << endl;

clearScreen();

}

// Previous Orders

else if (user == 5)

{

boughtBooksCount = 0;

addBoughtBookDatafromFile(refUserName, refUserPassword);

subMenu("Previous Orders");

cout << endl;

cout << "Following were your Previous Orders" << endl;

cout << endl;

cout << left << setw(10) << "Sr No." << setw(15) << "Name" << setw(15) << "Price" << endl;

cout << endl;

for (int z = 0; z < boughtBooksCount; z++)

{

cout << left << setw(10) << z + 1 << setw(15) << boughtBooks[z] << setw(15) << priceOfBoughtBooks[z] << endl;

}

cout << endl;

clearScreen();

}

// Give Suggestions

else if (user == 6)

{

subMenu("Suggestion Box");

giveSuggestion();

clearScreen();

}

}

}

else

{

cout << endl;

cout << endl;

cout << "User Not Found. Try Again" << endl;

clearScreen();

}

cout << endl;

}

}

}

void header()

{

doColor(228);

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

cout << " / \_\_\_(\_) |\_ \_ \_ | \_\_ ) \_\_\_ \_\_\_ | | \_\_ / \_\_\_|\_\_\_ \_ \_\_ | |\_ \_ \_\_ \_\_\_ " << endl;

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

cout << " | |\_\_\_| | |\_| |\_| | | |\_) | (\_) | (\_) | < | |\_\_| \_\_/ | | | |\_| | | \_\_/ " << endl;

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

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

cout << " " << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" << endl;

establishColor();

}

void printTime()

{

doColor(135);

time\_t now = time(0);

string output = ctime(&now);

cout << output;

establishColor();

}

int menu()

{

int option;

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

cout << " \* 1. S I G N - U P \* " << endl;

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

cout << endl

<< endl;

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

cout << " \* 2. S I G N - I N \* " << endl;

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

cout << endl

<< endl;

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

cout << " \* 3. E X I T \* " << endl;

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

cout << endl

<< endl;

cout << "Your Option: ";

cin >> option;

return option;

}

int adminMenu()

{

// here line breaks are given by gotoxy

int option;

gotoxy(2, 12);

cout << " A D M I N M A I N M E N U" << endl;

gotoxy(2, 14);

cout << "1. View the list of Available Books" << endl;

gotoxy(2, 15);

cout << "2. Add a Book" << endl;

gotoxy(2, 16);

cout << "3. Update a Book" << endl;

gotoxy(2, 17);

cout << "4. Remove a Book" << endl;

gotoxy(2, 18);

cout << "5. View purchased Books" << endl;

gotoxy(2, 19);

cout << "6. View the Reviews and Suggestions" << endl;

gotoxy(2, 20);

cout << "7. Instructions" << endl;

gotoxy(2, 21);

cout << "8. View the Income Statistics" << endl;

gotoxy(2, 22);

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

gotoxy(2, 23);

cout << "Please Choose your Option..... ";

cin >> option;

gotoxy(2, 25);

return option;

}

int userMenu()

{

// here line breaks are given by gotoxy

gotoxy(2, 12);

cout << "U S E R M A I N M E N U" << endl;

int option;

gotoxy(2, 14);

cout << "1. View the list of Available Books" << endl;

gotoxy(2, 15);

cout << "2. Purchase a Book" << endl;

gotoxy(2, 16);

cout << "3. View your Cart" << endl;

gotoxy(2, 17);

cout << "4. Lend a Book" << endl;

gotoxy(2, 18);

cout << "5. Track my Previous Orders" << endl;

gotoxy(2, 19);

cout << "6. Give Suggestion" << endl;

gotoxy(2, 20);

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

gotoxy(2, 21);

cout << "Please Choose your Option..... ";

cin >> option;

gotoxy(2, 23);

establishColor();

return option;

}

void listOfBooks()

{

doColor(10);

cout << "L I S T O F B O O K S" << endl;

cout << endl;

doColor(240);

cout << left << setw(10) << "Sr No." << setw(15) << "Name" << setw(15) << "Qty" << setw(15) << "Price" << setw(15) << "Lending Cost" << endl;

establishColor();

// printing the list of available books

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

{

cout << left << setw(10) << x + 1 << setw(15) << bookNames[x] << setw(15) << bookStock[x] << setw(15) << bookPrice[x] << setw(15) << bookLendingCost[x] << endl;

}

}

bool checkaddBook(string name, int price, int stock, int lendingPrice)

{

// check that either book is present earlier or not

char bookadding = true;

while (bookadding)

{

bool bookPresent = false;

for (int y = 0; y <= booksArrayCount; y++)

{

if ((bookNames[y] == name) && (bookPrice[y] == price) && (bookStock[y] == stock) && (bookLendingCost[y] == lendingPrice))

{

bookPresent = true;

}

}

return bookPresent;

}

}

void updateBook(string name, int price, int stock, int lendingCost)

{

// this will update the book at given index

bookNames[idx] = name;

bookPrice[idx] = price;

bookStock[idx] = stock;

bookLendingCost[idx] = Lcost;

}

int removeBook(int idx)

{

for (int x = idx; x < booksArrayCount; x++)

{

bookNames[x] = bookNames[x + 1];

bookPrice[x] = bookPrice[x + 1];

bookStock[x] = bookStock[x + 1];

bookLendingCost[x] = bookLendingCost[x + 1];

}

booksArrayCount--;

}

int incomeCalculation()

{

cout << " Following is the Analysis of Your Income" << endl;

cout << endl;

cout << " Total Orders Placed : " << ordersCounter << endl;

cout << " Total Income : " << incomeCounter << endl;

cout << endl;

}

void giveSuggestion()

{

string suggestionBook;

string text = "";

int count = 0;

string input;

bool suggestionBookPresent = false;

cout << " S U G G E S T I O N S B O X " << endl;

cout << endl;

cout << "Enter the Name of Book : ";

cin >> suggestionBook;

cout << endl;

suggestionBookPresent = findBook(suggestionBook);

if (suggestionBookPresent == true)

{

gotoxy(0, 15);

cout << endl;

cout << "Enter Your Reviews and Suggestions " << endl;

cout << "DOUBLE ENTER = SUBMIT" << endl;

cin.ignore();

moveSuggestionBox();

int h = 15;

while (true)

{

if (count == 6)

{

break;

}

gotoxy(1, h);

getline(cin, input);

if (input.empty())

{

break;

}

text += input;

count++;

h++;

}

cout << endl;

suggestionArray[suggestionCount] = text;

suggestionBookArray[suggestionCount] = suggestionBook;

suggestionCount++;

fstream file;

file.open("suggestions.txt", ios::app);

file << suggestionBook << ",";

file << text << endl;

file.close();

gotoxy(0, 22);

}

else

{

cout << "Book Not Found" << endl;

cout << endl;

}

}

void clearScreen()

{

cout << "Press Any key to continue....." << endl;

getch();

system("cls");

header();

gotoxy(50, 6);

printTime();

cout << endl;

cout << endl;

}

void clearScreen2()

{

system("cls");

header();

gotoxy(50, 6);

printTime();

cout << endl;

cout << endl;

}

bool signUp(string user, string password, string role)

{

bool isPresent = false;

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

{

if (users[x] == user && passwords[x] == password)

{

isPresent = true;

break;

}

}

return isPresent;

}

string signIn(string userName, string password2)

{

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

{

if (users[x] == userName && passwords[x] == password2)

{

return roles[x];

}

}

return "Undefined";

}

void gotoxy(int x, int y)

{

COORD coordinates;

coordinates.X = x;

coordinates.Y = y;

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

}

void subMenubeforeMainMenu(string subMenu)

{

doColor(226);

string message = subMenu + " Menu";

cout << message << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" << endl;

cout << endl;

establishColor();

}

void subMenu(string subMenu)

{

doColor(226);

string message = "Main Menu > " + subMenu;

cout << message << endl;

cout << "%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" << endl;

doColor(135);

gotoxy(65, 9);

cout << "[ " << refUserName << " ]";

gotoxy(0, 12);

establishColor();

}

void addUserDatafromFile()

{

string data;

fstream file;

file.open("UsersData.txt", ios::in);

while (getline(file, data))

{

users[usercount] = parseData(data, 1);

passwords[usercount] = parseData(data, 2);

roles[usercount] = parseData(data, 3);

usercount++;

}

file.close();

}

void addBookDatafromFile()

{

int op = 0;

string bookDataLine;

fstream bookData;

bookData.open("BooksData.txt", ios::in);

while (getline(bookData, bookDataLine))

{

bookNames[booksArrayCount] = parseData(bookDataLine, 1);

bookPrice[booksArrayCount] = stoi(parseData(bookDataLine, 2));

bookStock[booksArrayCount] = stoi(parseData(bookDataLine, 3));

bookLendingCost[booksArrayCount] = stoi(parseData(bookDataLine, 4));

booksArrayCount++;

}

bookData.close();

}

void addPurchasedBookDatafromFile(string username, string password)

{

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

{

if (wholePurchasedBooksUser[x] == username && wholePurchasedBooksPassword[x] == password)

{

purchasedBooks[purchasedBooksCount] = wholePurchasedBooksNames[x];

priceOfPurchasedBooks[purchasedBooksCount] = wholePurchasedBooksPrice[x];

purchasedBooksCount++;

}

}

}

void addBoughtBookDatafromFile(string username, string password)

{

string temporaryUser;

string temporaryPassword;

string boughtBookDataLine;

fstream file;

file.open("UserBoughtBooks.txt", ios::in);

while (getline(file, boughtBookDataLine))

{

temporaryUser = parseData(boughtBookDataLine, 1);

temporaryPassword = parseData(boughtBookDataLine, 2);

if (temporaryPassword == password && temporaryUser == username)

{

boughtBooks[boughtBooksCount] = parseData(boughtBookDataLine, 3);

priceOfBoughtBooks[boughtBooksCount] = stoi(parseData(boughtBookDataLine, 4));

boughtBooksCount++;

}

}

}

void transferDatatoNextfile(string username, string password)

{

fstream file;

file.open("userBoughtBooks.txt", ios::app);

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

{

file << username << ",";

file << password << ",";

file << purchasedBooks[x] << ",";

file << priceOfPurchasedBooks[x] << endl;

ordersCounter++;

incomeCounter += priceOfPurchasedBooks[x];

fstream dataFile;

dataFile.open("incomeData.txt", ios::out);

dataFile << ordersCounter << endl;

dataFile << incomeCounter << endl;

dataFile.close();

}

file.close();

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

{

if (wholePurchasedBooksUser[x] == username && wholePurchasedBooksPassword[x] == password)

{

wholePurchasedBooksUser[x] = wholePurchasedBooksUser[x + 1];

wholePurchasedBooksPassword[x] = wholePurchasedBooksPassword[x + 1];

wholePurchasedBooksNames[x] = wholePurchasedBooksNames[x + 1];

wholePurchasedBooksPrice[x] = wholePurchasedBooksPrice[x + 1];

wholePurchasedBooksArrayCount--;

x--;

}

}

purchasedBooksCount = 0;

fstream updateFile;

updateFile.open("UserPurchasedBooks.txt", ios::out);

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

{

updateFile << wholePurchasedBooksUser[x] << ",";

updateFile << wholePurchasedBooksPassword[x] << ",";

updateFile << wholePurchasedBooksNames[x] << ",";

updateFile << wholePurchasedBooksPrice[x] << endl;

}

updateFile.close();

}

void updateBooksDatainFile()

{

fstream file;

file.open("BooksData.txt", ios::out);

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

{

file << bookNames[x] << endl;

file << bookPrice[x] << endl;

file << bookStock[x] << endl;

file << bookLendingCost[x] << endl;

}

file.close();

}

bool findBook(string checkBook)

{

bool checkVariable = false;

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

{

if (bookNames[x] == checkBook)

{

idx = x;

checkVariable = true;

break;

}

}

return checkVariable;

}

string parseData(string data, int field)

{

int commaCount = 1;

string item;

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

{

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

{

commaCount++;

}

else if (commaCount == field)

{

item += data[x];

}

}

return item;

}

void printProgressBar(int x, int y)

{

char character = 219;

int idx = 0;

int k = 224;

gotoxy(x + 11, y + 1);

cout << "Loading....";

while (idx < 30)

{

doColor(k);

gotoxy(x, y);

cout << character;

Sleep(150);

idx++;

x++;

k++;

if (k == 238)

{

k = 224;

}

}

establishColor();

}

void printVerticalLine(int x, int y, int limit)

{

int k = 1;

for (int idx = y; idx < limit + y; idx++)

{

doColor(k);

gotoxy(x, idx);

char character = 186;

cout << character;

Sleep(20);

k++;

if (k == 15)

{

k = 1;

}

}

establishColor();

}

void printHorizontalLine(int x, int y, int limit)

{

int k = 1;

for (int idx = x; idx < limit + x; idx++)

{

doColor(k);

gotoxy(idx, y);

cout << "=";

Sleep(10);

k++;

if (k == 15)

{

k = 1;

}

}

establishColor();

}

void purchaseBook()

{

purchasedBooks[purchasedBooksCount] = bookNames[idx];

priceOfPurchasedBooks[purchasedBooksCount] = bookPrice[idx];

purchasedBooksCount++;

fstream newFile;

newFile.open("UserPurchasedbooks.txt", ios::app);

newFile << refUserName << ",";

newFile << refUserPassword << ",";

newFile << bookNames[idx] << ",";

newFile << bookPrice[idx] << endl;

newFile.close();

cout << "Book Added to your Cart Successfully" << endl;

bookStock[idx] = bookStock[idx] - 1;

}

void addWholePurchasedBooksDatafromFile()

{

string bookDataLine;

fstream file;

file.open("UserPurchasedBooks.txt", ios::in);

while (getline(file, bookDataLine))

{

wholePurchasedBooksUser[wholePurchasedBooksArrayCount] = parseData(bookDataLine, 1);

wholePurchasedBooksPassword[wholePurchasedBooksArrayCount] = parseData(bookDataLine, 2); wholePurchasedBooksNames[wholePurchasedBooksArrayCount] = parseData(bookDataLine, 3);

wholePurchasedBooksPrice[wholePurchasedBooksArrayCount] = stoi(parseData(bookDataLine, 4));

wholePurchasedBooksArrayCount++;

}

file.close();

}

void addWholeBoughtBooksDatafromFile()

{

string bookDataLine;

fstream file;

file.open("UserBoughtBooks.txt", ios::in);

while (getline(file, bookDataLine))

{

wholeBoughtBooksUser[wholeBoughtBooksArrayCount] = parseData(bookDataLine, 1);

wholeBoughtBooksPassword[wholeBoughtBooksArrayCount] = parseData(bookDataLine, 2);

wholeBoughtBooksNames[wholeBoughtBooksArrayCount] = parseData(bookDataLine, 3);

wholeBoughtBooksPrice[wholeBoughtBooksArrayCount] = stoi(parseData(bookDataLine, 4));

wholeBoughtBooksArrayCount++;

}

file.close();

}

void doColor(int index)

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, index);

}

void establishColor()

{

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, 224);

}

bool checkStringCondition(string word)

{

int number = word.length();

bool result = true;

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

{

if (!isalpha(word[x]))

{

result = false;

}

}

return result;

}

bool checkIntegerCondition(string word)

{

int number = word.length();

bool result = true;

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

{

if (!isdigit(word[x]))

{

result = false;

}

}

return result;

}

void printInstructions()

{

cout << "Following are some Instructions for the Program" << endl;

cout << endl;

cout << "1. Only Admin can Add, Update and Remove Books" << endl;

cout << "2. The Prices are only liable to the Admin" << endl;

cout << "3. The Admin holds the Copyright claims of the program" << endl;

}

void eraseInstructions()

{

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

}

void moveInstructions()

{

int k = 224;

for (int y = 25; y > 11; y--)

{

doColor(k);

gotoxy(0, y);

eraseInstructions();

gotoxy(0, y - 1);

printInstructions();

Sleep(100);

k++;

if (k == 237)

{

k = 224;

}

}

establishColor();

}

void loadSuggestions()

{

string suggestionsWholeLine;

fstream file;

file.open("suggestions.txt", ios::in);

while (getline(file, suggestionsWholeLine))

{

suggestionBookArray[suggestionCount] = parseData(suggestionsWholeLine, 1);

suggestionArray[suggestionCount] = parseData(suggestionsWholeLine, 2);

suggestionCount++;

}

file.close();

}

void loadIncomeData()

{

fstream dataFile;

dataFile.open("incomeData.txt", ios::in);

dataFile >> ordersCounter;

dataFile >> incomeCounter;

dataFile.close();

}

void printSuggestionBox()

{

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

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

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

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

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

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

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

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

}

void eraseSuggestionBox()

{

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

cout << " " << endl;

}

void moveSuggestionBox()

{

int k = 224;

for (int y = 25; y > 14; y--)

{

doColor(k);

gotoxy(0, y);

eraseSuggestionBox();

gotoxy(0, y - 1);

printSuggestionBox();

Sleep(100);

k++;

if (k == 238)

{

k = 224;

}

}

establishColor();

}

void addBook(string name, int price, int stock, int lendingCost)

{

// adding book details in array

bookNames[booksArrayCount] = name;

bookPrice[booksArrayCount] = stoi(price);

bookStock[booksArrayCount] = stoi(stock);

bookLendingCost[booksArrayCount] = stoi(lendingCost);

booksArrayCount++;

// adding book details in file

fstream file;

file.open("BooksData.txt", ios::app);

file << name << ",";

file << price << ",";

file << stock << ",";

file << lendingCost << endl;

file.close();

}

Weakness in the Business Application

There are some shortcomings in the program. The application is not connected with Internet and thus can not offer online services. The Database system is not much efficient. As we have to store data in the files. The Users don’t possess a unique identity number on which there data can be managed. The Cash transfer can not be ensured by the program. Also, the program is running on Command Prompt hence, better visualization can not be presented. It’s a little difficult to make it attractive or user friendly.

Future Directions

In future, I will try to provide a graphical view of the program. I shall try to publish this on Internet. So, maximum people can approach the program and avail its benefits. The Admin can offer discount on the day bases or by his own will. Moreover, I will try to add more Users in the program depending upon their roles.