Computer Science Unit 2

S.B.A.

Student: Brandon Hinds

Class: 6A2

School: St. Jago High School

Teacher: Ms. Morant

**Table of Contents**

Acknowledgements …………………………………………………………………………………………………………………… page 1

Description of Organization ………………………………………………………………………………………………………. page 2

Definition of Problem ……………………………………………………………………………………………………………….. page 3

Techniques of Analysis ……………………………………………………………………………………………………………… page 4

Data Flow Diagram: Context …………………………………………………………………………………………………….. page 5

Data Flow Diagram: Level 1 ………………………………………………………………………………………………………. page 6

Entity Relationship Diagram ……………………………………………………………………………………………………… page 7

Functional and Non-Functional Requirements ………………………………………………………………………….. page 8

System Architecture …………………………………………………………………………………………………………………. page 9

Structure Chart …………………..…………………………………………………………………………………………………… page 10

User Interface Design ……………………………………………………………………………………………………………… page 11

Algorithm Design …………………………………………………………………………………………………………… pages 12 - 16

Data Structure ………………………………………………………………………………………………………………….…..… page 17

Data Dictionary …………………………………………………………………………………………………………………..…… page 18

Test Plan …………………………………………………………………………………………………………………………. pages 19 - 24

C Code ……………………………………………………………………………………………………………………………. pages 25 - 34

Questionnaire ……………………………………………………………………………………………………………..…. pages 35 - 38

Interview Questions ………………………………………………………………………………………………………………… page 39

Data Results ……………………………………………………………………………………………………………………. pages 40 - 45

**Acknowledgements**

I would like to thank my parents for their support throughout this time while completing this project, I would like to thank my teacher for providing me with the information for this project and I would also like to thank my classmates and all who inspired me to complete this project.

**Description of Organization**

St Jago High School (formerly Beckford & Smith) in Spanish Town, St Catherine, Jamaica, founded in 1744, is one of the oldest, continuously operated schools in the Western Hemisphere. It is renowned for graduating some of Jamaica's senior military officers, world class cricketers, academic scholars, performing artists, and Olympic athletes.

St Jago is an offshoot of the Free School of Saint Jago de la Vega which started when Sir Peter Beckford in his will left 1,000 pound sterling to the construction of a free school, and Francis Smith left 300 pound sterling to the same cause. In 1958, after a merger of Beckford and Smith’s High and the Cathedral High, St. Jago High School was declared opened by then governor general of Jamaica, Sir Kenneth Blackburne.

St. Jago High school is located at Monk Street, Spanish Town, St. Catherine, Jamaica.

The St. Jago High School Library lends books to its users every day. They currently have a system in place but due to bugs and errors in the program it needs to be replaced.

**Definition of Problem**

The St. Jago High School library's book keeping system needs to be able to record all the necessary data for each book. The current system also is unprotected meaning that anyone can use the system without administrative rights. The user has to manually find the file containing the information and read through it to find a particular entry. What can be done is to allow the system to accept the information of each book and when it is to be stored, allow it to be written to a file, allow the user to search the file and at the beginning of the program prompt the user to enter a password that should only be known by whoever is authorized to use the system.

**Techniques of Analysis**

To collect data for the study the qualitative approach of observation was used. This made it easy to see how the user interacts with the current system and identify any problems associated with it. An interview was also conducted with the head librarian about the system and questionnaires were given to other library monitors to gain their insight on the software and tell of any problems associated with it and to find out how the system can be improved. This made it easier to determine the requirement for the system since the end users were involved in the process.

**Data Flow Diagram: Level 1**

Password

Login

1

Librarian

Library Password

Validation

S1 Library Password

2

Book Details

Book Entry

Add Book to Loan Database

S2 All Books’ Details

Specific Book Name

Specific Book Details

3

Return Book to Library

All Books’ Details

All Books’ Details

Request

5

View All Books in Database

Specific Book Details

Specific Book Details

Specific Book Name

Specific Book Details

4

Search Loan Database

**Data Flow Diagram: Context Level**

Book Loan

0

Book Keeping System

Book

Book

Library

Member

Return

Book

Book Order

Book Supplies

Book Supplier

**Entity Relationship Diagram**

Library

Work At Has

Are Monitored By

Head Librarian

Librarians

Monitor

Are Processed By Lend

Books

Name, ISBN, Author,

Students

Publisher

Return Are Loaned To

Name, Class

**Functional Requirements**

1. A password must be entered in order for the user to access the system.

2. The program must accept the name, ISBN, author and publisher of each book being loaned as well as the name and class of the student who borrowed it.

3. The information of each book that has been entered must be stored in a file.

4. The program should allow the user to view the records of all books currently on loan.

5. The program should allow the user to search for a particular book in the records.

**Non-functional Requirements**

1. It is portable meaning that it is able to be used on Windows Vista/7/8/8.1

2. It has a menu driven user interface so that it is user friendly.

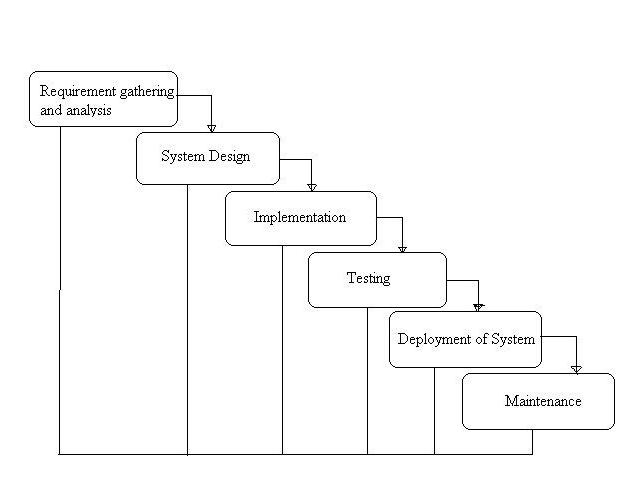
3. The program hides the password being entered by the user.

4. A keyboard will be used as the main input device for the program.

5. The program reacts quickly to input.

**System Architecture**

For the development of this system the waterfall model of software development was chosen. It was chosen because the requirements needed for the system were already identified before developing it. It also makes it easy to track the progress of its development and it is simple and easy to understand and to use. It does not allow the developer to move on to a different stage without completing the current stage which can avoid confusion and errors.



**Structure Chart**

Library Management System

Requested Book Info

Book Name

Delete Book Record

Book Name

Search Book

All Book Data

View All

New Book Data

All Book Data

Return Book

Loan Book

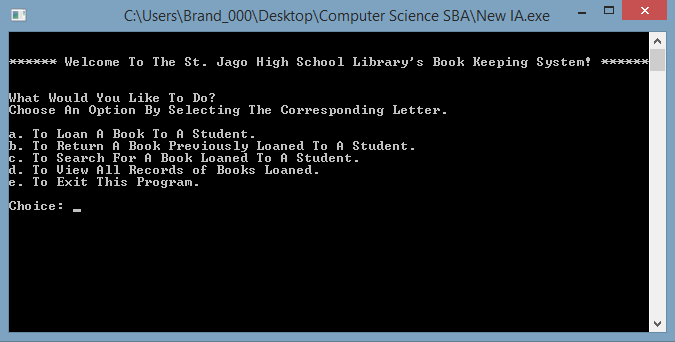
Password

Verification

Login

**User Interface Design**

The user interface for the system was given careful consideration. Since we cannot assume that the user will be a total computer expert or skilled with command line, a menu driven user interface was selected. This makes it more user friendly since all the user would have to do is select the option they want also it limits user error since little typing is required. It also gives clear and unambiguous instructions so that the user can understand what they are doing.



**HIPO Chart**

Return Book to Library

Lend Book to Student

View All Loaned Books

Search for Books

Loan Management

Authentication

Library System

**Algorithm Design (Pseudocode)**

**Start**

Function Main:

Goto Login();

End Function;

Function Login:

Passc = ‘pass’

Write “Enter the Password”;

Read Passe;

If passe := passc:

Write “Success”;

Goto menu();

Else:

Write “Incorrect”;

End Function;

Function Menu:

Write ‘What would you like to do?’

Write ‘a. Lend Book to Student’

Write ‘b. Return Book to Library’

Write ‘c. Search for Borrowed Book’

Write ‘d. View All Borrowed Books’

Write ‘e. Exit Program’

Read Choice;

If choice := ‘a’:

Goto Loan();

Else if choice := ‘b’ :

Goto Return();

Else if choice := ‘c’ :

Goto Search();

Else if choice := ‘d’ :

Goto View();

Else if choice := ‘e’ :

exit();

End Function;

Function Loan:

File = “Loaned\_Books.txt”;

Write “How many books are being loaned?”

Read num\_books;

counter = 0;

While counter < num\_books:

Write “What is the name of the book?”;

Read book\_name;

Write “What is the ISBN of the book?”;

Read book\_ISBN;

Write “Who is the author of the book?”;

Read book\_author;

Write “Who is the publisher of the book?”;

Read book\_publisher;

Write “What is the name of the borrower?”;

Read student\_name;

Write “What is the class of the borrower?”;

Read student\_class;

PrintToFile book\_name;

PrintToFile book\_ISBN;

PrintToFile book\_author;

PrintToFile book\_publisher;

PrintToFile student\_name;

PrintToFile student\_class;

End while;

Goto Menu();

End Function;

Function Return:

File = “Loaned\_Books.txt”;

Write “What is the name of the book being returned?”;

Read return\_book;

Read book\_name from File;

Read book\_ISBN from File;

Read book\_author from File;

Read book\_publisher from File;

Read student\_name from File;

Read student\_class from File;

If book\_name := return\_book:

Delete book\_name from File;

Delete book\_ISBN from File;

Delete book\_author from File;

Delete book\_publisher from File;

Delete student\_name from File;

Delete student\_class from File;

Write “Book has been returned”;

Else:

Write “Book Not Found”;

Goto Menu();

End Function;

Function Search:

File = “Loaned\_Books.txt”;

Write “What is the name of the book you’re looking for?”

Read book\_search;

Read book\_name from File;

Read book\_ISBN from File;

Read book\_author from File;

Read book\_publisher from File;

Read student\_name from File;

Read student\_class from File;

If book\_name := book\_search:

Write book\_name;

Write book\_ISBN;

Write book\_author;

Write book\_publisher;

Write student\_name;

Write student\_class;

Else:

Write “Book Not Found”;

Goto Menu();

End Function;

Function View:

File = “Loaned\_Books.txt”

Read book\_name from File;

Read book\_ISBN from File;

Read book\_author from File;

Read book\_publisher from File;

Read student\_name from File;

Read student\_class from File;

Write book\_name;

Write book\_ISBN;

Write book\_author;

Write book\_publisher;

Write student\_name;

Write student\_class;

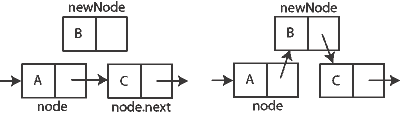
Goto Menu();

End Function;

**End.**

**Data Structure**

For the development of this system a singly linked list seemed to be the most appropriate. It seemed so since inserting and deleting data can be done easily without the need to move elements, space is not wasted (efficient memory usage) and size is not limited.



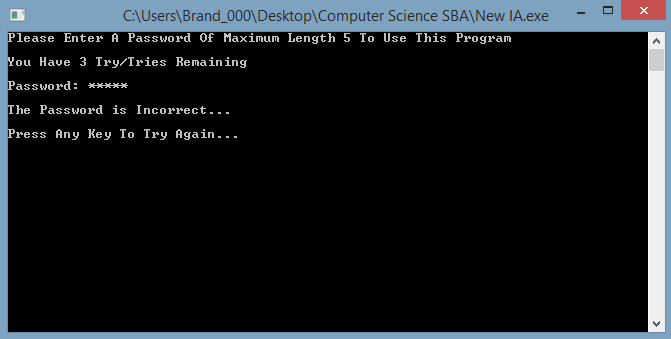
|  |  |  |  |
| --- | --- | --- | --- |
| **Data Description** | **Data Type/Structure** | **Purpose** | **Justification** |
| Name of Book | Text/String | To hold the name of book | Names are a usually a collection of letters |
| ISBN | Integer | To hold the ISBN of book | Books usually have unique numbers known as ISBNs |
| Author | Text/String | To hold the name of the author of the book | The names of authors are usually made up of letters |
| Publisher | Text/String | To hold the name of the publisher of the book | Publishers are usually stored as strings since their names consists of mostly letters |
| Name of Student | Text/String | To hold the name of the student who borrowed the book | Names are stored as a group of characters/letters. |
| Class of Student | Text/String | To hold the class of the student who borrowed the book | The class of a student may be any combination of characters |
| List of Books Loaned | Array of Text/String | To the list of books loaned to students | It is necessary to store them in indices so that they can later be found, retrieved and manipulated |
| Login Attempts | Integer | To hold the amount of times a user has tried to login | The program counts the amount of login attempts. If the user gets the password wrong three times in a row the program will automatically exit |
| Amount of Books Being loaned | Integer | To hold the amount of books being loaned at once | The amount of books being entered at once needs to be known so that the part of the program that records book data iterates |

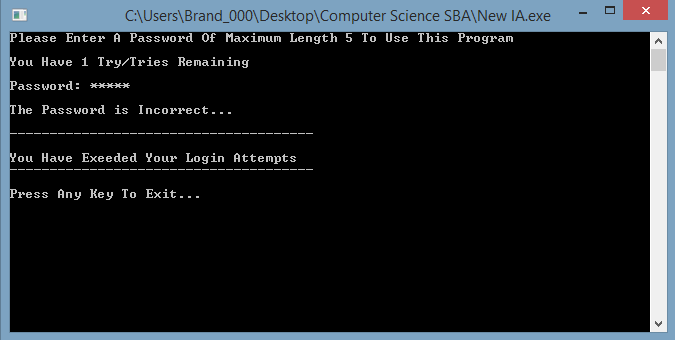
**Data Dictionary**

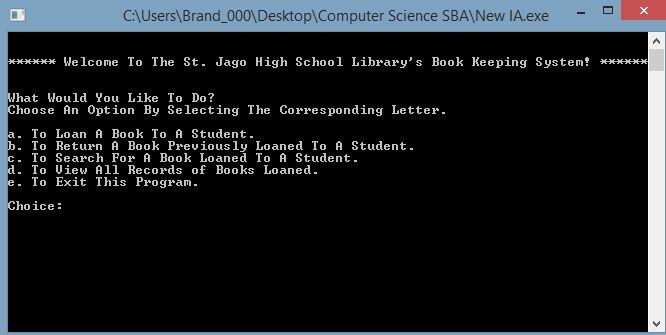
**Test Plan**

Test for login function:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Results** | **Actual Results** |
| The password ‘pass’ | The password should be masked then after entering it the main menu screen should come up afterwards. | The password was masked then after the password was entered the main menu screen came up. |
| Anything else | The program should then say that the password entered is incorrect and prompt the user to enter it again if the amount of login attempts is less than three, otherwise it should exit the program. | The program displayed the error message and prompted the user to enter a new one while the amount of attempts was less than three and when it reached the limit the program ended. |







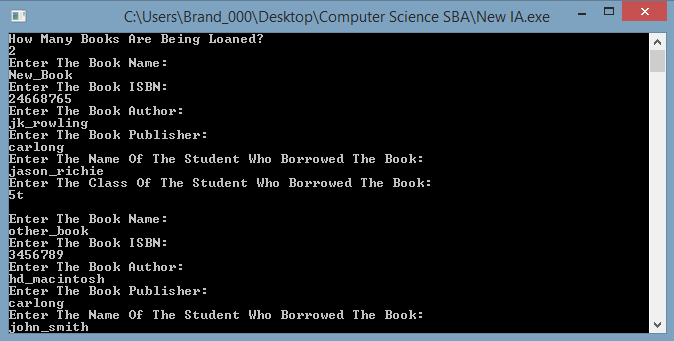
Test plan for main menu:

The user should choose from one of five options

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Results** | **Actual Results** |
| a | The program will execute the loan book function. | The loan book function was executed. |
| b | The program will execute the return book function. | The return book function was executed. |
| c | The program will execute the search function. | The search function was executed. |
| d | The program will execute the view book function. | The view book function was executed. |
| e | The program should end. | The program ended. |
| Any Other Input | An error message should be displayed saying ‘That is not an option’ then the user will be able to select another option. | The error message was displayed and the user was able to select another option. |

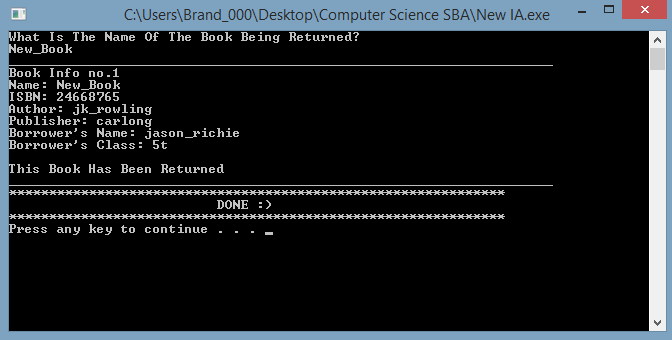
Test Plan for Loan Book function:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Results** | **Actual Results** |
| Number of book being loaned | The program should ask for the next six items by the number entered. | The program repeated asking for the next six items by the number entered. |
| The name of the book | It should store the name of the book in the file containing the books borrowed. | The name of the book was stored in the file containing the books borrowed. |
| The ISBN of the book | It should store the ISBN of the book in the file containing the books borrowed. | The ISBN of the book was stored in the file containing the books borrowed. |
| The author of the book | It should store the name of the author of the book in the file containing the books borrowed. | The name of the author of the book was stored in the file containing the books borrowed. |
| The publisher of the book | It should store the name of the publisher of the book in the file containing the books borrowed. | The name of the publisher of the book was stored in the file containing the books borrowed. |
| The name of the student borrowing the book | The name of the student who borrowed the book should be stored in the file containing the books borrowed. | The name of the student who borrowed the book was stored in the file containing the books borrowed. |
| The class of the student borrowing the book | The class of the student who borrowed the book should be stored in the file containing the books borrowed. | The class of the student who borrowed the book was stored in the file containing the books borrowed. |



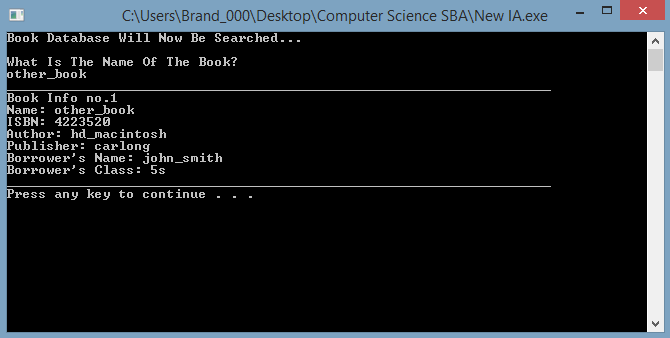
Test Plan for Return Book Function:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Results** | **Actual Results** |
| The name of the book being returned. | The program should search the file for the book and if it is found delete it from the list of books loaned. | The program searched the file and when the book was found it was deleted from the list of books loaned. |



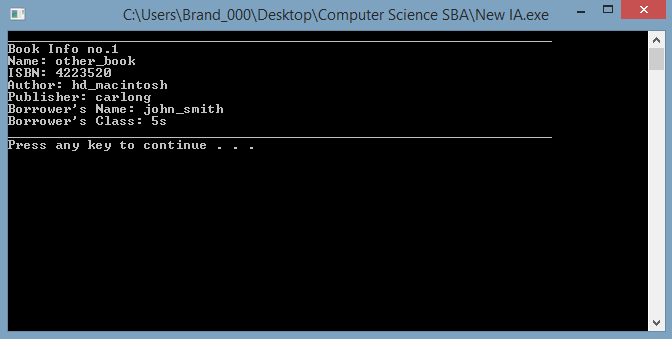
Test Plan for Search Book Function:

|  |  |  |
| --- | --- | --- |
| **Input** | **Expected Results** | **Actual Results** |
| The name of the book the user wishes to find. | The program will search the file containing the books borrowed and when the book is found the information will be displayed. | The program searched the file containing the books borrowed and when the book was found the information was displayed. |



Test Plan for View All Function:

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| The program should read the file containing the information of books borrowed then display the information of each book. | The program read the file containing the information of books borrowed then displayed the information of each book. |



Test Plan for the Exit Function:

|  |  |
| --- | --- |
| **Expected Results** | **Actual Results** |
| The program should end. | The program ended. |

**C Code**

#include <stdio.h>

#include <stdlib.h> //This library is used for the exit function

#include <string.h> //This is used for operations with strings

#include <windows.h> //This is for added effects to the program

struct books{ //This struct is a model of the main information we want from each book

int ISBN; //The ISBN Of The Book

char name[100]; //The Name Of The Book

char author[100]; //The Author Of The Book

char publisher[100]; //The Publisher Of The Book

char SName[100]; //The Name Of The Student who borrowed the book

char Class[10]; //The class of the student who borrowed the book

};

//Function Declaration

int login(); //This function is where the user will login

int menu(); //This function is where the user will get to do whatever options are available

int LoanBook(); //This function will allow the user to add books to the database of books loaned

int ReturnBook(); //This function will manage the books being returned to the library

int SearchBook(); //This function will allow the user to search for a specific book

int ViewRecord(); //This function will allow the user to view all books that have been loaned

//Global Variables

int count; //This is a counter and will also serve as the index for the array which holds the struct

int bookn; //This will be used to accept the number of books the user wants to enter

int tries = 3; //The maximum amount of tries to enter the password in the login function

int attempts = 0; //This is used in the login function

struct books library[4000]; //This array will hold the information of all books borrowed

int main(){

system("color 17"); //This will add color to the program

login();

return 0;

}

int login()

{

system("cls");

char passn[5]; //Password The User Will Enter

int count = 0; //Counter Variable

char p;

attempts++;

printf("Please Enter The Password To Continue\n");

printf("\nYou Have %d Try/Tries Remaining\n", (tries - attempts+1));

printf("\n");

printf("Password: ");

//This while loop is for entering and concealing the password

while(passn[count] != 13){

passn[count] = getch();

p = passn[count];

if (p == 13) break;

else printf("\*");

count++;

}

passn[count]='\0';

count = 0;

if (strcmp(passn, "pass")== 0){

//These next few lines of code is just for fun

system("cls");

for (count = 0; count< 10; count++)

{

printf("\t\t\t Loading Library System |");

Sleep(100);

system("cls");

printf("\t\t\t Loading Library System /");

Sleep(100);

system("cls");

printf("\t\t\t Loading Library System -");

Sleep(100);

system("cls");

}

system("cls");

//Fun Over

menu();

}

else{

printf("\n");

if (attempts < tries){

printf("\nThe Password is Incorrect...\n");

printf("\nPress Any Key To Try Again...\n");

getch();

printf("\n");

}

if (attempts >= tries){

printf("\nThe Password is Incorrect...\n");

printf("\n");

printf("--------------------------------------\n");

printf("\nYou Have Exeeded Your Login Attempts\n");

printf("--------------------------------------\n");

printf("\n");

printf("Press Any Key To Exit...\n");

getch();

exit(0);

}

else{

return main();

}

}

}

int menu(){

system("cls");

fflush(stdin);

char choice; //This is the users choice of what to do in this program

printf("\n");

printf("\n\*\*\*\*\*\* Welcome To The St. Jago High School Library's Book Keeping System! \*\*\*\*\*\*\n");

printf("\n");

printf("What Would You Like To Do?\n");

printf("Choose An Option By Selecting The Corresponding Letter.\n");

printf("\n");

printf("a. To Loan A Book To A Student.\n");

printf("b. To Return A Book Previously Loaned To A Student.\n");

printf("c. To Search For A Book Loaned To A Student.\n");

printf("d. To View All Records of Books Loaned.\n");

printf("e. To Exit This Program.\n");

printf("\n");

printf("Choice: ");

choice = getch();

if (choice == 'a'){

LoanBook(); //This will direct the user to function LoanBook

}

else if (choice == 'b'){

ReturnBook(); //This will direct the user to function ReturnBook

}

else if (choice == 'c'){

SearchBook(); //This will direct the user to function SearchBook

}

else if (choice == 'd'){

ViewRecord(); //This will direct the user to function ViewRecord

}

else if (choice == 'e'){

exit(0); //The user can leave the program through this option

}

else{ //If the user enters anything than what is supposed to be entered:

printf("That Is Not An Option!\n");

system("pause");

return menu(); //An error message will be printed and the program will return to menu

}

}

int LoanBook(){ //This is the function to add books to the file

fflush(stdin);

system("cls");

FILE\*rental;

rental = fopen("Loaned\_Books.txt","a");

printf("How Many Books Are Being Loaned?\n");

scanf("%d", &bookn);

for (count = 0; count < bookn; count++){

printf("Enter The Book Name:\n");

scanf("%s",&library[count].name);

printf("Enter The Book ISBN:\n");

scanf("%d",&library[count].ISBN);

printf("Enter The Book Author:\n");

scanf("%s",&library[count].author);

printf("Enter The Book Publisher:\n");

scanf("%s",&library[count].publisher);

printf("Enter The Name Of The Student Who Borrowed The Book:\n");

scanf("%s",&library[count].SName);

printf("Enter The Class Of The Student Who Borrowed The Book:\n");

scanf("%s",&library[count].Class);

printf("\n");

fprintf(rental, "%s\n" ,library[count].name);

fprintf(rental, "%d\n" ,library[count].ISBN);

fprintf(rental, "%s\n" ,library[count].author);

fprintf(rental, "%s\n" ,library[count].publisher);

fprintf(rental, "%s\n" ,library[count].SName);

fprintf(rental, "%s\n" ,library[count].Class);

fprintf(rental, "\n");

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" DONE :) \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

system("pause");

fclose(rental);

return menu(); //The program returns to menu at the end of the function

}

int ReturnBook()

{

system("cls");

fflush(stdin);

FILE\*borrow;

FILE\*borrow2;

borrow = fopen("Loaned\_Books.txt", "r");

borrow2 = fopen("Loaned\_copy.txt", "a");

char nsearch[100];

if (borrow == NULL){ //If the file is empty or does not exist:

printf("I'm Sorry But Information From The Rental Database Cannot Be Retrieved Right Now.\n");

system("pause");

return menu(); //The error message will be printed and the program returns to menu

}

printf("What Is The Name Of The Book Being Returned?\n");

scanf("%s", &nsearch);

for (count = 0; count < 4000; count++){

fscanf(borrow, "%s\n" ,&library[count].name);

fscanf(borrow, "%d\n" ,&library[count].ISBN);

fscanf(borrow, "%s\n" ,&library[count].author);

fscanf(borrow, "%s\n" ,&library[count].publisher);

fscanf(borrow, "%s\n" ,&library[count].SName);

fscanf(borrow, "%s\n" ,&library[count].Class);

}

for (count = 0; count < 4000; count++){

if (strcmp(nsearch, library[count].name) == 0){

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Book Info no.%d\n", count + 1);

printf("Name: %s\n", library[count].name);

printf("ISBN: %d\n", library[count].ISBN);

printf("Author: %s\n", library[count].author);

printf("Publisher: %s\n" ,library[count].publisher);

printf("Borrower's Name: %s\n" ,library[count].SName);

printf("Borrower's Class: %s\n" ,library[count].Class);

printf("\n");

printf("This Book Has Been Returned\n");

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

else {

if (strcmp("", library[count].name) != 0){

fprintf(borrow2, "%s\n" ,&library[count].name);

fprintf(borrow2, "%d\n" ,&library[count].ISBN);

fprintf(borrow2, "%s\n" ,&library[count].author);

fprintf(borrow2, "%s\n" ,&library[count].publisher);

fprintf(borrow2, "%s\n" ,&library[count].SName);

fprintf(borrow2, "%s\n" ,&library[count].Class);

fprintf(borrow2, "\n");

}

}

}

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" DONE :) \n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

system("pause");

fclose(borrow);

fclose(borrow2);

remove("Loaned\_Books.txt");

rename("Loaned\_copy.txt", "Loaned\_Books.txt" );

return menu();

}

int SearchBook(){ //This is the function to search the file for a book previously entered

system("cls");

fflush(stdin);

FILE\*rental;

rental = fopen("Loaned\_Books.txt","r");

char nsearch[100]; //This will be used to find the name of the book

if (rental == NULL){ //If the file is empty or does not exist:

printf("I'm Sorry But Information From The Rental Database Cannot Be Retrieved Right Now.\n");

system("pause");

return menu(); //The error message will be printed and the program returns to menu

}

else{

printf("Book Database Will Now Be Searched...\n");

printf("\n");

printf("What Is The Name Of The Book?\n");

scanf("%s", &nsearch);

for (count = 0; count < 4000; count++){

fscanf(rental, "%s\n" ,&library[count].name);

fscanf(rental, "%d\n" ,&library[count].ISBN);

fscanf(rental, "%s\n" ,&library[count].author);

fscanf(rental, "%s\n" ,&library[count].publisher);

fscanf(rental, "%s\n" ,&library[count].SName);

fscanf(rental, "%s\n" ,&library[count].Class);

}

for (count = 0; count < 4000; count++){

if (strcmp(nsearch, library[count].name) == 0){

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Book Info no.%d\n", count + 1);

printf("Name: %s\n", library[count].name);

printf("ISBN: %d\n", library[count].ISBN);

printf("Author: %s\n", library[count].author);

printf("Publisher: %s\n" ,library[count].publisher);

printf("Borrower's Name: %s\n" ,library[count].SName);

printf("Borrower's Class: %s\n" ,library[count].Class);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

}

fclose(rental);

system("pause");

return menu(); //The program returns to menu at the end of the function

}

int ViewRecord() //Function to view all records of books loaned

{

fflush(stdin);

system("cls");

FILE\*rental;

rental = fopen("Loaned\_Books.txt","r");

if (rental == NULL){ //If the file is empty or does not exist:

printf("I'm Sorry But Information From The Rental Database Cannot Be Retrieved Right Now.\n");

system("pause");

return menu(); //The error message will be printed and the program returns to menu

}

else{

for (count = 0; count < 4000; count++){

fscanf(rental, "%s\n" ,&library[count].name);

fscanf(rental, "%d\n" ,&library[count].ISBN);

fscanf(rental, "%s\n" ,&library[count].author);

fscanf(rental, "%s\n" ,&library[count].publisher);

fscanf(rental, "%s\n" ,&library[count].SName);

fscanf(rental, "%s\n" ,&library[count].Class);

}

for (count = 0; count < 4000; count++){

if (strcmp("", library[count].name) != 0){

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("Book Info no.%d\n", count + 1);

printf("Name: %s\n", library[count].name);

printf("ISBN: %d\n", library[count].ISBN);

printf("Author: %s\n", library[count].author);

printf("Publisher: %s\n" ,library[count].publisher);

printf("Borrower's Name: %s\n" ,library[count].SName);

printf("Borrower's Class: %s\n" ,library[count].Class);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

}

fclose(rental);

system("pause");

return menu(); //The program returns to menu at the end of the function

}

**Questionnaire**

1. What are the problems that you face with the current system? (You may select more than one)

Bugs

Freezing

Crashes

It has no problems

1. How often do you find yourself frustrated when using the system?

I’ve never used it

Sometimes

Every time

Most Times

1. How long has the system been in place?

I didn’t know we had a system

I don’t know

4+ years

1-3 years

1. How often do you have to lend books?

A few times a week

Not Often

Everyday

Never

1. Do you find yourself in need of help whenever you use the system?

Never use it

Sometimes

Most of the time

Constantly

1. Does the system have proper security measures such as the ability to prevent unwanted access or editing of information?

No

Yes

1. How ‘easy to use’ is the system?

Very Hard

Hard most of the time

Easy with some problems

Very Easy

1. Are the instructions given by the system in order to use it clear, unambiguous and easy to understand?

No

Yes

1. How well can the system perform all of its required functions?

It does none of its required tasks

It can only do some of its required tasks

It does most of its required tasks

It performs all of its required tasks

1. Do you think that the library needs a new system?

No

Yes

**Interview Questions**

1. What are the problems that are currently being experienced by the users of the current library system?
2. Do the users of the current library systems find the system tedious?
3. Does the current system provide enough security measures to protect data?
4. Can the current system perform all of the necessary tasks needed in order to maintain the efficiency of the book loan system?
5. What specific functions does the system need to perform?

**Data Results**

Interview with the head librarian:

Response to question 1:

“They are having problems with the directions given to them by the system. The directions seem to be unclear. They also report that the system crashes and freezes.”

Response to question 2:

“The process of loaning a book to a student is supposed to be simple but the system makes it very tedious. Since it crashes and freezes often and its instructions are hard to follow, we had to resort to recording information in a book.”

Response to question 3:

“Unfortunately the system is unprotected. Since there is no password or any other security measure in place anyone can use the system and edit information.”

Response to question 4:

“The system doesn’t perform all the necessary tasks we require from it. This is also what makes using it tedious since we have to find other ways of fulfilling the necessary tasks.”

Response to question 5:

“The system needs to be able to record all our information, store it in a database and allow us to retrieve the information and delete it. It also needs to be protected so that only authorized users may use it.”

Questionnaire Results:

Question 1:

Question 2:

Question 3:

Question 4:

Question 5:

Question 6:

Question 7:

Question 8:

Question 9:

Question 10: