***LIBRARY MANAGEMENT SYSTEM***

**INDEX**

S.NO TITLE PAGE.NO

1. INTRODUCTION 1
2. ANALYSIS 2
3. Scope of the project
4. Software requiremen

specification

1. System design 6
2. Data flow diagram 7

5 module specification 8

6 description of function and

Pointers 11

7 code 22

**INTRODUCTION**

**“ LIBRARY MANAGEMENT SYSTEM”**  is one of the most important and essential part of our daily life . It requires the software system to get more information easy and accuracy .

The project **“ LIBRARY MANAGEMENT SYSTEM “** includes Addition of books , view of added books , search of the required books etc .

The software uses Addition of books and deletion of books and view of books items in the library .

***ANALYSIS***

1. ***scope of the project :-***

The aim of the project is to find the required book in the library . In the “ **LIBRARY MANAGEMENT SYSTEM”** we need to do different task like

* REGISTRATION OF THE ACCOUNT
* ADDING OF THE BOOKS
* DELETING OF THE BOOKS
* VIEW OF THE BOOKS
* SEARCH OF THE BOOKS
* MODIFY THE DATA

**REGISTRATION OF THE ACCOUNT :-**

In Registration option add the client information as client name and client address and client phone number and client library card number .

**ADDING OF THE BOOKS :-**

The New Books is added to the list of the books that the client required and in the library

**DELETION OF THE BOOKS:-**

The Book must be deleted in the list if the client does not required the book.

**VIEW OF THE BOOKS:-**

The library management system requires to view the list of the books which was there in the library .

**SEARCH OF THE BOOKS :-**

The software must be design easily to search of the books for the requirement based on the clients.

**MODIFY THE DATA :-**

The data must be updated whenever it required .

**SOFTWARE REQUIREMENT SPECIFICATIONS**

1. **FUNCTIONALITY :-**

The Functionality requirement specify which output should be produced by the given input

1. **PERFORMANCE :-**

This part of SRS specify performance constraints in the software system .

All the requirement relating to the performance characteristics of the system must be clearly specified

1. **DESIGN CONSTRAINTS :-**

There are number of factors in the client environment that may restrict the choices of the design

1. **EXTERNAL INTERFACE :-**

All possible interaction of the software with client hardware & other software should be clearly specified .

**SYSTEM DESIGN**



**DATA FLOW DIAGRAM**

Data base

Registration

Display

Personal data

Client

Customer information

Search

client data

Modify

Delete

Enter the

client no

**MODULE SPECIFICATION**

1. **REGISTRATION :-**

In Registration option add the client information client name ,client address and books details .

1. **DISPLAY :-**

In Display data read all the client information such as client name and client address and book name and author .

**3)SEARCH :-**

In search option ,search the information about particular client such client data and client information and books details .

**4 ) MODIFY :-**

In Modification , modify all the required data .

1. **DELETE :-**

In delete option ,delete the unwanted files in the given data .

**“LIBRARY MANAGEMENT SYSTEM “** consists of

**Isfileexists() :-**

The function verifies that the file must be created or not . If the file exist the function return 1 otherwise return 0 .

int isFileExists(const char\*path)

{

//Try to open file

FILE\*fp = fopen(path,”rb”);

Int status = 0;

//if file does not exists

**if(fp! = NULL)**

**{**

**Status =1;**

**//File exists hence close file**

**fclose(fp);**

**}**

**return status;**

**init() :-**

This function create the file if not exist and copies the default password in file header structure .

Void init()

{

File\*fp = NULL ;

int status = 0 ;

const char defaultUsername[] ="abcdefgh\n";

const char defaultPassword[] ="abcdefgh\n";

sFileHeader fileHeaderInfo = {0};

status = isFileExists(FILE\_NAME);

**if**(!status)

{

//create the binary file

fp = fopen(FILE\_NAME,"wb");

**if**(fp != **NULL**)

{

//Copy default password

strncpy(fileHeaderInfo.password,defaultPassword,sizeof(defaultPassword));

strncpy(fileHeaderInfo.username,defaultUsername,sizeof(defaultUsername));

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

}

}

}

**Print Message center() :-**

This function prints the message in the middle of the head massage. I have passed the message in this function as per the operation.

void printMessageCenter(const char\* message)

{

int len =0;

int pos = 0;

//calculate how many space need to print

len = (78 - strlen(message))/2;

printf("\t\t\t");

**for**(pos =0 ; pos < len ; pos++)

{

//print space

printf(" ");

}

//print message

printf("%s",message);

}

**HEAD MESSAGE () :-**

It prints the message on the top of the console and prints the message as per operation.

void headMessage(const char \*message)

{

system("cls");

printf("\t\t\t###########################################################################");

printf("\n\t\t\t############ ############");

printf("\n\t\t\t############ Library management System Project in C ############");

printf("\n\t\t\t############ ############");

printf("\n\t\t\t###########################################################################");

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

printMessageCenter(message);

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

}

**welcomeMessage():**

This function displays the first welcomes screen of the “Library management system project” and asks the user to press any key to access the library application.

void welcomeMessage()

{

headMessage("www.aticleworld.com");

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

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

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

printf("\n\t\t\t = WELCOME =");

printf("\n\t\t\t = TO =");

printf("\n\t\t\t = LIBRARY =");

printf("\n\t\t\t = MANAGEMENT =");

printf("\n\t\t\t = SYSTEM =");

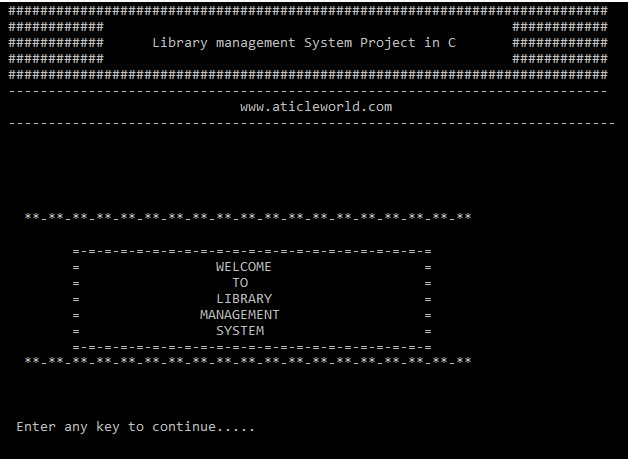
printf("\n\t\t\t =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");

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

printf("\n\n\n\t\t\t Enter any key to continue.....");

getch();

}



**isNameValid():**

It validates the user name, author name ..etc. I have permitted this function to take the space in names.

int isNameValid(const char \*name)

{

int validName = 1;

int len = 0;

int index = 0;

len = strlen(name);

**for**(index =0; index <len ; ++index)

{

**if**(!(isalpha(name[index])) && (name[index] != '\n') && (name[index] != ' '))

{

validName = 0;

break;

}

}

**return** validName;

}

**isValidDate():**

This function read the date in the format of dd/mm/yyyy also it validates the entered date.

// Function to check leap year.

//Function returns 1 if leap year

int IsLeapYear(int year)

{

**return** (((year % 4 == 0) &&

(year % 100 != 0)) ||

(year % 400 == 0));

}

// returns 1 if given date is valid.

int isValidDate(Date \*validDate)

{

//check range of year,month and day

**if** (validDate->yyyy > MAX\_YR ||

validDate->yyyy < MIN\_YR)

**return** 0;

**if** (validDate->mm < 1 || validDate->mm > 12)

**return** 0;

**if** (validDate->dd < 1 || validDate->dd > 31)

**return** 0;

//Handle feb days in leap year

**if** (validDate->mm == 2)

{

**if** (IsLeapYear(validDate->yyyy))

**return** (validDate->dd <= 29);

**else**

**return** (validDate->dd <= 28);

}

//handle months which has only 30 days

**if** (validDate->mm == 4 || validDate->mm == 6 ||

validDate->mm == 9 || validDate->mm == 11)

**return** (validDate->dd <= 30);

**return** 1;

}

**menu():**

This function displays the library menu and asks the user to select the option. If the user selects 0, then the application will close.

void menu()

{

int choice = 0;

**do**

{

headMessage("MAIN MENU");

printf("\n\n\n\t\t\t1.Add Books");

printf("\n\t\t\t2.Search Books");

printf("\n\t\t\t3.View Books");

printf("\n\t\t\t4.Delete Book");

printf("\n\t\t\t5.Update Password");

printf("\n\t\t\t0.Exit");

printf("\n\n\n\t\t\tEnter choice => ");

scanf("%d",&choice);

switch(choice)

{

case 1:

addBookInDataBase();

break;

case 2:

searchBooks();

break;

case 3:

viewBooks();

break;

case 4:

deleteBooks();

break;

case 5:

updateCredential();

break;

case 0:

printf("\n\n\n\t\t\t\tThank you!!!\n\n\n\n\n");

exit(1);

break;

default:

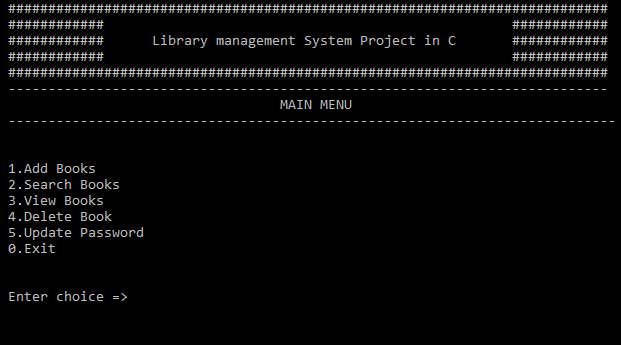
printf("\n\n\n\t\t\tINVALID INPUT!!! Try again...");

} //Switch Ended

}

**while**(choice!=0); //Loop Ended

}



#### addBookInDataBase():

This function opens the binary file in append mode and writes the book and the details.

// Add books in list

**void** addBookInDataBase()

{

int days;

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

int status = 0;

fp = fopen(FILE\_NAME,"ab+");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

headMessage("ADD NEW BOOKS");

printf("\n\n\t\t\tENTER YOUR DETAILS BELOW:");

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

printf("\n\t\t\tBook ID NO = ");

fflush(stdin);

scanf("%u",&addBookInfoInDataBase.books\_id);

**do**

{

printf("\n\t\t\tBook Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.bookName,MAX\_BOOK\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.bookName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

printf("\n\t\t\tAuthor Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.authorName,MAX\_AUTHOR\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.authorName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

printf("\n\t\t\tStudent Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.studentName,MAX\_STUDENT\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.studentName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

//get date year,month and day from user

printf("\n\t\t\tEnter date in format (day/month/year): ");

scanf("%d/%d/%d",&addBookInfoInDataBase.bookIssueDate.dd,&addBookInfoInDataBase.bookIssueDate.mm,&addBookInfoInDataBase.bookIssueDate.yyyy);

//check date validity

status = isValidDate(&addBookInfoInDataBase.bookIssueDate);

**if** (!status)

{

printf("\n\t\t\tPlease enter a valid date.\n");

}

}

**while**(!status);

fwrite(&addBookInfoInDataBase,sizeof(addBookInfoInDataBase), 1, fp);

fclose(fp);

}

#### searchBooks():

This function opens the binary file in reading mode and asks the user to enter the book name which wants to search.  If the book is not available in the list, it shows the message book not find in records.

void searchBooks()

{

int found = 0;

char bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

int status = 0;

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("\n\t\t\tFile is not opened\n");

exit(1);

}

headMessage("SEARCH BOOKS");

//put the control on books detail

**if** (fseek(fp,FILE\_HEADER\_SIZE,SEEK\_SET) != 0)

{

fclose(fp);

printf("\n\t\t\tFacing issue while reading file\n");

exit(1);

}

printf("\n\n\t\t\tEnter Book Name to search:");

fflush(stdin);

fgets(bookName,MAX\_BOOK\_NAME,stdin);

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

**if**(!strcmp(addBookInfoInDataBase.bookName, bookName))

{

found = 1;

break;

}

}

**if**(found)

{

printf("\n\t\t\tBook id = %u\n",addBookInfoInDataBase.books\_id);

printf("\t\t\tBook name = %s",addBookInfoInDataBase.bookName);

printf("\t\t\tBook authorName = %s",addBookInfoInDataBase.authorName);

printf("\t\t\tBook issue date(day/month/year) = (%d/%d/%d)",addBookInfoInDataBase.bookIssueDate.dd,

addBookInfoInDataBase.bookIssueDate.mm, addBookInfoInDataBase.bookIssueDate.yyyy);

}

**else**

{

printf("\n\t\t\tNo Record");

}

fclose(fp);

printf("\n\n\n\t\t\tPress any key to go to main menu.....");

getchar();

}

#### viewBooks():

It opens the file in reading mode and read and display all the stored book details. If there is no book available in the records, then it displays the message record is empty.

// view all books function

**void** viewBooks()

{

int found = 0;

char bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

int status = 0;

unsigned int countBook = 1;

headMessage("VIEW BOOKS DETAILS");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

**if** (fseek(fp,FILE\_HEADER\_SIZE,SEEK\_SET) != 0)

{

fclose(fp);

printf("Facing issue while reading file\n");

exit(1);

}

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

printf("\n\t\t\tBook Count = %d\n\n",countBook);

printf("\t\t\tBook id = %u",addBookInfoInDataBase.books\_id);

printf("\n\t\t\tBook name = %s",addBookInfoInDataBase.bookName);

printf("\t\t\tBook authorName = %s",addBookInfoInDataBase.authorName);

printf("\t\t\tBook issue date(day/month/year) = (%d/%d/%d)",addBookInfoInDataBase.bookIssueDate.dd,

addBookInfoInDataBase.bookIssueDate.mm, addBookInfoInDataBase.bookIssueDate.yyyy);

found = 1;

++countBook;

}

fclose(fp);

**if**(!found)

{

printf("\n\t\t\tNo Record");

}

printf("\n\n\t\t\tPress any key to go to main menu.....");

fflush(stdin);

getchar();

}

#### deleteBooks():

This function asks the book id from the user for the book want to delete. In this function, I am creating a temporary binary file and copy all the data from the existing file except the book whose book id entered by the user. In the last renamed the temporary bin file with an existing binary file.

void deleteBooks()

{

int found = 0;

int bookDelete = 0;

sFileHeader fileHeaderInfo = {0};

char bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

FILE \*tmpFp = **NULL**;

int status = 0;

headMessage("Delete Books Details");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

tmpFp = fopen("tmp.bin","wb");

**if**(tmpFp == **NULL**)

{

fclose(fp);

printf("File is not opened\n");

exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, tmpFp);

printf("\n\t\t\tEnter Book ID NO. for delete:");

scanf("%d",&bookDelete);

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

**if**(addBookInfoInDataBase.books\_id != bookDelete)

{

fwrite(&addBookInfoInDataBase,sizeof(addBookInfoInDataBase), 1, tmpFp);

}

**else**

{

found = 1;

}

}

(found)? printf("\n\t\t\tRecord deleted successfully....."):printf("\n\t\t\tRecord not found");

fclose(fp);

fclose(tmpFp);

remove(FILE\_NAME);

rename("tmp.bin",FILE\_NAME);

}

#### updateCredential():

This function opens the file in rb+ mode (reading and writing). It asks the user for the new username and password. After taking the password and username it closes the application. Now user can use the application with a new password and username.

void updateCredential(**void**)

{

sFileHeader fileHeaderInfo = {0};

FILE \*fp = **NULL**;

unsigned char userName[MAX\_SIZE\_USER\_NAME] = {0};

unsigned char password[MAX\_SIZE\_PASSWORD] = {0};

headMessage("Update Credential");

fp = fopen(FILE\_NAME,"rb+");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

**if** (fseek(fp,0,SEEK\_SET) != 0)

{

fclose(fp);

printf("\n\t\t\tFacing issue while updating password\n");

exit(1);

}

printf("\n\n\t\t\tNew Username:");

fflush(stdin);

fgets(userName,MAX\_SIZE\_USER\_NAME,stdin);

printf("\n\n\t\t\tNew Password:");

fflush(stdin);

fgets(password,MAX\_SIZE\_PASSWORD,stdin);

strncpy(fileHeaderInfo.username,userName,sizeof(userName));

strncpy(fileHeaderInfo.password,password,sizeof(password));

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

printf("\n\t\t\tYour Password has been changed successfully");

printf("\n\t\t\ttLogin Again:");

fflush(stdin);

getchar();

exit(1);

}

**COMPLETE CODE OF LIBRARY MANAGEMENT SYSTEM :-**

**#include <stdio.h>**

**#include <time.h>**

**#include <string.h>**

**#define MAX\_YR 9999**

**#define MIN\_YR 1900**

**#define MAX\_SIZE\_USER\_NAME 30**

**#define MAX\_SIZE\_PASSWORD 20**

**#define FILE\_NAME "CPROGRAMMING LANGAGUE "**

// Macro related to the books info

**#define MAX\_BOOK\_NAME 50**

**#define MAX\_AUTHOR\_NAME 50**

**#define MAX\_STUDENT\_NAME 50**

**#define MAX\_STUDENT\_ADDRESS 300**

**#define FILE\_HEADER\_SIZE sizeof(sFileHeader)**

//structure to store date

**typedef** **struct**

{

**int** yyyy;

**int** mm;

**int** dd;

} Date;

**typedef** **struct**

{

**char** username[MAX\_SIZE\_USER\_NAME];

**char** password[MAX\_SIZE\_PASSWORD];

} sFileHeader;

**typedef** **struct**// to call in program

{

unsigned **int** books\_id; // declare the integer data type

**char** bookName[MAX\_BOOK\_NAME];// declare the character data type

**char** authorName[MAX\_AUTHOR\_NAME];// declare the charecter data type

**char** studentName[MAX\_STUDENT\_NAME];// declare the character data type

**char** studentAddr[MAX\_STUDENT\_ADDRESS];// declare the character data type

Date bookIssueDate;// declare the integer data type

} s\_BooksInfo;

**void** printMessageCenter(const **char**\* message)

{

**int** len =0;

**int** pos = 0;

//calculate how many space need to print

len = (78 - strlen(message))/2;

printf("\t\t\t");

**for**(pos =0 ; pos < len ; pos++)

{

//print space

printf(" ");

}

//print message

printf("%s",message);

}

**void** headMessage(const **char** \*message)

{

system("cls");

printf("\t\t\t###########################################################################");

printf("\n\t\t\t############ ############");

printf("\n\t\t\t############ Library management System Project in C ############");

printf("\n\t\t\t############ ############");

printf("\n\t\t\t###########################################################################");

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

printMessageCenter(message);

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

}

**void** welcomeMessage()

{

headMessage("www.CPROGRAMMINGLANGAGUE .com");

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

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

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

printf("\n\t\t\t = WELCOME =");

printf("\n\t\t\t = TO =");

printf("\n\t\t\t = LIBRARY =");

printf("\n\t\t\t = MANAGEMENT =");

printf("\n\t\t\t = SYSTEM =");

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

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

printf("\n\n\n\t\t\t Enter any key to continue.....");

getch();

}

**int** isNameValid(const **char** \*name)

{

**int** validName = 1;

**int** len = 0;

**int** index = 0;

len = strlen(name);

**for**(index =0; index <len ; ++index)

{

**if**(!(isalpha(name[index])) && (name[index] != '\n') && (name[index] != ' '))

{

validName = 0;

**break**;

}

}

**return** validName;

}

// Function to check leap year.

//Function returns 1 if leap year

**int** IsLeapYear(**int** year)

{

**return** (((year % 4 == 0) &&

(year % 100 != 0)) ||

(year % 400 == 0));

}

// returns 1 if given date is valid.

**int** isValidDate(Date \*validDate)

{

//check range of year,month and day

**if** (validDate->yyyy > MAX\_YR ||

validDate->yyyy < MIN\_YR)

**return** 0;

**if** (validDate->mm < 1 || validDate->mm > 12)

**return** 0;

**if** (validDate->dd < 1 || validDate->dd > 31)

**return** 0;

//Handle feb days in leap year

**if** (validDate->mm == 2)

{

**if** (IsLeapYear(validDate->yyyy))

**return** (validDate->dd <= 29);

**else**

**return** (validDate->dd <= 28);

}

//handle months which has only 30 days

**if** (validDate->mm == 4 || validDate->mm == 6 ||

validDate->mm == 9 || validDate->mm == 11)

**return** (validDate->dd <= 30);

**return** 1;

}

// Add books in list

**void** addBookInDataBase()

{

**int** days;

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

**int** status = 0;

fp = fopen(FILE\_NAME,"ab+");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

headMessage("ADD NEW BOOKS");

printf("\n\n\t\t\tENTER YOUR DETAILS BELOW:");

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

printf("\n\t\t\tBook ID NO = ");

fflush(stdin);

scanf("%u",&addBookInfoInDataBase.books\_id);

**do**

{

printf("\n\t\t\tBook Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.bookName,MAX\_BOOK\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.bookName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

printf("\n\t\t\tAuthor Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.authorName,MAX\_AUTHOR\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.authorName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

printf("\n\t\t\tStudent Name = ");

fflush(stdin);

fgets(addBookInfoInDataBase.studentName,MAX\_STUDENT\_NAME,stdin);

status = isNameValid(addBookInfoInDataBase.studentName);

**if** (!status)

{

printf("\n\t\t\tName contain invalid character. Please enter again.");

}

}

**while**(!status);

**do**

{

//get date year,month and day from user

printf("\n\t\t\tEnter date in format (day/month/year): ");

scanf("%d/%d/%d",&addBookInfoInDataBase.bookIssueDate.dd,&addBookInfoInDataBase.bookIssueDate.mm,&addBookInfoInDataBase.bookIssueDate.yyyy);

//check date validity

status = isValidDate(&addBookInfoInDataBase.bookIssueDate);

**if** (!status)

{

printf("\n\t\t\tPlease enter a valid date.\n");

}

}

**while**(!status);

fwrite(&addBookInfoInDataBase,sizeof(addBookInfoInDataBase), 1, fp);

fclose(fp);

}

// search books

**void** searchBooks()

{

**int** found = 0;

**char** bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

**int** status = 0;

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("\n\t\t\tFile is not opened\n");

exit(1);

}

headMessage("SEARCH BOOKS");

//put the control on books detail

**if** (fseek(fp,FILE\_HEADER\_SIZE,SEEK\_SET) != 0)

{

fclose(fp);

printf("\n\t\t\tFacing issue while reading file\n");

exit(1);

}

printf("\n\n\t\t\tEnter Book Name to search:");

fflush(stdin);

fgets(bookName,MAX\_BOOK\_NAME,stdin);

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

**if**(!strcmp(addBookInfoInDataBase.bookName, bookName))

{

found = 1;

**break**;

}

}

**if**(found)

{

printf("\n\t\t\tBook id = %u\n",addBookInfoInDataBase.books\_id);

printf("\t\t\tBook name = %s",addBookInfoInDataBase.bookName);

printf("\t\t\tBook authorName = %s",addBookInfoInDataBase.authorName);

printf("\t\t\tBook issue date(day/month/year) = (%d/%d/%d)",addBookInfoInDataBase.bookIssueDate.dd,

addBookInfoInDataBase.bookIssueDate.mm, addBookInfoInDataBase.bookIssueDate.yyyy);

}

**else**

{

printf("\n\t\t\tNo Record");

}

fclose(fp);

printf("\n\n\n\t\t\tPress any key to go to main menu.....");

getchar();

}

// v books function

**void** viewBooks()

{

**int** found = 0;

**char** bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

**int** status = 0;

unsigned **int** countBook = 1;

headMessage("VIEW BOOKS DETAILS");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

**if** (fseek(fp,FILE\_HEADER\_SIZE,SEEK\_SET) != 0)

{

fclose(fp);

printf("Facing issue while reading file\n");

exit(1);

}

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

printf("\n\t\t\tBook Count = %d\n\n",countBook);

printf("\t\t\tBook id = %u",addBookInfoInDataBase.books\_id);

printf("\n\t\t\tBook name = %s",addBookInfoInDataBase.bookName);

printf("\t\t\tBook authorName = %s",addBookInfoInDataBase.authorName);

printf("\t\t\tBook issue date(day/month/year) = (%d/%d/%d)\n\n",addBookInfoInDataBase.bookIssueDate.dd,

addBookInfoInDataBase.bookIssueDate.mm, addBookInfoInDataBase.bookIssueDate.yyyy);

found = 1;

++countBook;

}

fclose(fp);

**if**(!found)

{

printf("\n\t\t\tNo Record");

}

printf("\n\n\t\t\tPress any key to go to main menu.....");

fflush(stdin);

getchar();

}

// delete function

**void** deleteBooks()

{

**int** found = 0;

**int** bookDelete = 0;

sFileHeader fileHeaderInfo = {0};

**char** bookName[MAX\_BOOK\_NAME] = {0};

s\_BooksInfo addBookInfoInDataBase = {0};

FILE \*fp = **NULL**;

FILE \*tmpFp = **NULL**;

**int** status = 0;

headMessage("Delete Books Details");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

tmpFp = fopen("tmp.bin","wb");

**if**(tmpFp == **NULL**)

{

fclose(fp);

printf("File is not opened\n");

exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, tmpFp);

printf("\n\t\t\tEnter Book ID NO. for delete:");

scanf("%d",&bookDelete);

**while** (fread (&addBookInfoInDataBase, sizeof(addBookInfoInDataBase), 1, fp))

{

**if**(addBookInfoInDataBase.books\_id != bookDelete)

{

fwrite(&addBookInfoInDataBase,sizeof(addBookInfoInDataBase), 1, tmpFp);

}

**else**

{

found = 1;

}

}

(found)? printf("\n\t\t\tRecord deleted successfully....."):printf("\n\t\t\tRecord not found");

fclose(fp);

fclose(tmpFp);

remove(FILE\_NAME);

rename("tmp.bin",FILE\_NAME);

}

**void** updateCredential(**void**)

{

sFileHeader fileHeaderInfo = {0};

FILE \*fp = **NULL**;

unsigned **char** userName[MAX\_SIZE\_USER\_NAME] = {0};

unsigned **char** password[MAX\_SIZE\_PASSWORD] = {0};

headMessage("Update Credential");

fp = fopen(FILE\_NAME,"rb+");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

**if** (fseek(fp,0,SEEK\_SET) != 0)

{

fclose(fp);

printf("\n\t\t\tFacing issue while updating password\n");

exit(1);

}

printf("\n\n\t\t\tNew Username:");

fflush(stdin);

fgets(userName,MAX\_SIZE\_USER\_NAME,stdin);

printf("\n\n\t\t\tNew Password:");

fflush(stdin);

fgets(password,MAX\_SIZE\_PASSWORD,stdin);

strncpy(fileHeaderInfo.username,userName,sizeof(userName));

strncpy(fileHeaderInfo.password,password,sizeof(password));

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

printf("\n\t\t\tYour Password has been changed successfully");

printf("\n\t\t\ttLogin Again:");

fflush(stdin);

getchar();

exit(1);

}

**void** menu()

{

**int** choice = 0;

**do**

{

headMessage("MAIN MENU");

printf("\n\n\n\t\t\t1.Add Books");

printf("\n\t\t\t2.Search Books");

printf("\n\t\t\t3.View Books");

printf("\n\t\t\t4.Delete Book");

printf("\n\t\t\t5.Update Password");

printf("\n\t\t\t0.Exit");

printf("\n\n\n\t\t\tEnter choice => ");

scanf("%d",&choice);

**switch**(choice)

{

**case** 1:

addBookInDataBase();

**break**;

**case** 2:

searchBooks();

**break**;

**case** 3:

viewBooks();

**break**;

**case** 4:

deleteBooks();

**break**;

**case** 5:

updateCredential();

**break**;

**case** 0:

printf("\n\n\n\t\t\t\tThank you!!!\n\n\n\n\n");

exit(1);

**break**;

**default**:

printf("\n\n\n\t\t\tINVALID INPUT!!! Try again...");

} //Switch Ended

}

**while**(choice!=0); //Loop Ended

}

//login password

**void** login()

{

unsigned **char** userName[MAX\_SIZE\_USER\_NAME] = {0};

unsigned **char** password[MAX\_SIZE\_PASSWORD] = {0};

**int** L=0;

sFileHeader fileHeaderInfo = {0};

FILE \*fp = **NULL**;

headMessage("Login");

fp = fopen(FILE\_NAME,"rb");

**if**(fp == **NULL**)

{

printf("File is not opened\n");

exit(1);

}

fread (&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

**do**

{

printf("\n\n\n\t\t\t\tUsername:");

fgets(userName,MAX\_SIZE\_USER\_NAME,stdin);

printf("\n\t\t\t\tPassword:");

fgets(password,MAX\_SIZE\_PASSWORD,stdin);

**if**((!strcmp(userName,fileHeaderInfo.username)) && (!strcmp(password,fileHeaderInfo.password)))

{

menu();

}

**else**

{

printf("\t\t\t\tLogin Failed Enter Again Username & Password\n\n");

L++;

}

}

**while**(L<=3);

**if**(L>3)

{

headMessage("Login Failed");

printf("\t\t\t\tSorry,Unknown User.");

getch();

system("cls");

}

}

**int** isFileExists(const **char** \*path)

{

// Try to open file

FILE \*fp = fopen(path, "rb");

**int** status = 0;

// If file does not exists

**if** (fp != **NULL**)

{

status = 1;

// File exists hence close file

fclose(fp);

}

**return** status;

}

**void** init()

{

FILE \*fp = **NULL**;

**int** status = 0;

const **char** defaultUsername[] ="sowmya\n";

const **char** defaultPassword[] ="sowmya\n";

sFileHeader fileHeaderInfo = {0};

status = isFileExists(FILE\_NAME);

**if**(!status)

{

//create the binary file

fp = fopen(FILE\_NAME,"wb");

**if**(fp != **NULL**)

{

//Copy default password

strncpy(fileHeaderInfo.password,defaultPassword,sizeof(defaultPassword));

strncpy(fileHeaderInfo.username,defaultUsername,sizeof(defaultUsername));

fwrite(&fileHeaderInfo,FILE\_HEADER\_SIZE, 1, fp);

fclose(fp);

}

}

}

**int** main()

{

init();

welcomeMessage();

login();

**return** 0;

}