The Library Management System

The MEDKS Team

Team Lead
Section 1
Section 4
Written Report
Presentation

Team Member
Section 2
Library Inventory

Team Member
Section 3

Team Member
Section 2
Library Inventory

Team Member
Section 1

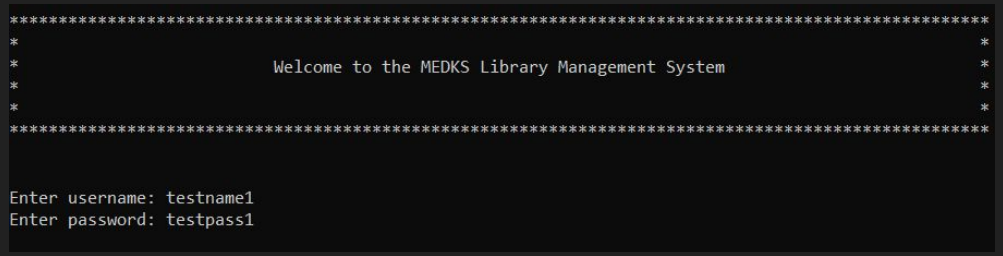
Team Member
Section 3
Library Inventory
Written Report
Presentation

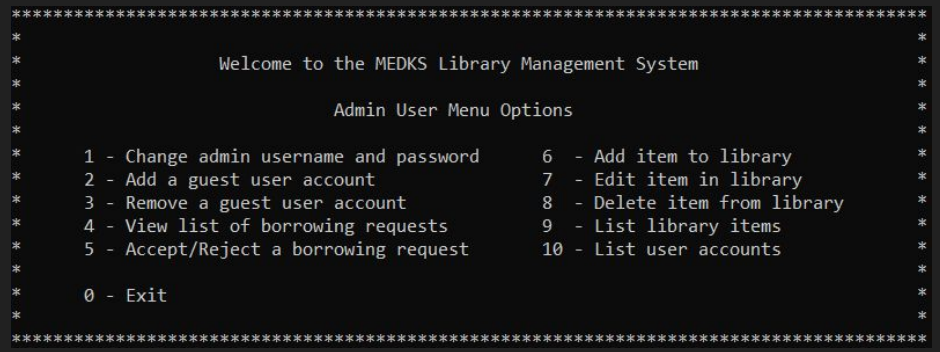

System Descriptions

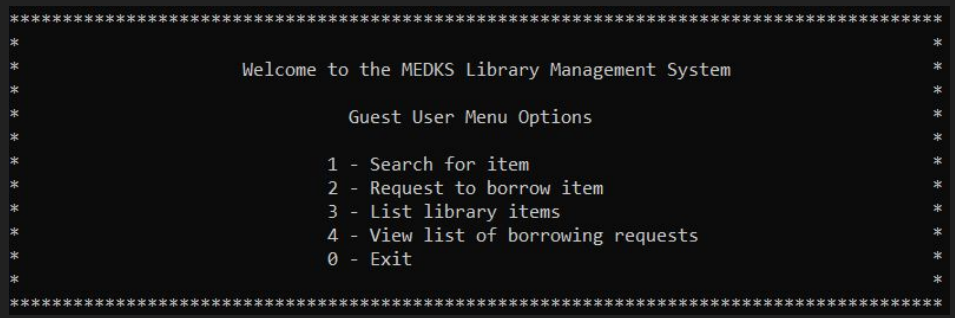
Section 1

Administrator user capabilities:

- Required LMS admin username and password log-in
- Change administrator username and password when desired
- Add guest users to the LMS by creating usernames and passwords
    - Note: guest users do not have the same permissions as the admin user
- Remove guest users by removing all guest associated usernames, passwords, and corresponding recorded files
- Add item(s) to the LMS with varied details
- Delete / edit item(s) in the LMS
- View recorded list of borrowing requests
- Accept / reject borrowing request(s)







Section 2

LMS user capabilities:

- Search through LMS based on certain item details
- Request for borrowing of item(s) in the library for a specific time
- View the history of borrowed items / their requests

```
**************************************************************************
*                                                                        *
*                        Search for Library Items                        *
*                                                                        *
*                                Options                                 *
*                                                                        *
*          1 - Search by ITEM ID       4 - Search by SUBJECT             *
*          2 - Search by TITLE         5 - Search by FORMAT              *
*          3 - Search by AUTHOR        6 - Search by ISBN                *
*                                                                        *
*          0 - Exit                                                      *
*                                                                        *
**************************************************************************
```

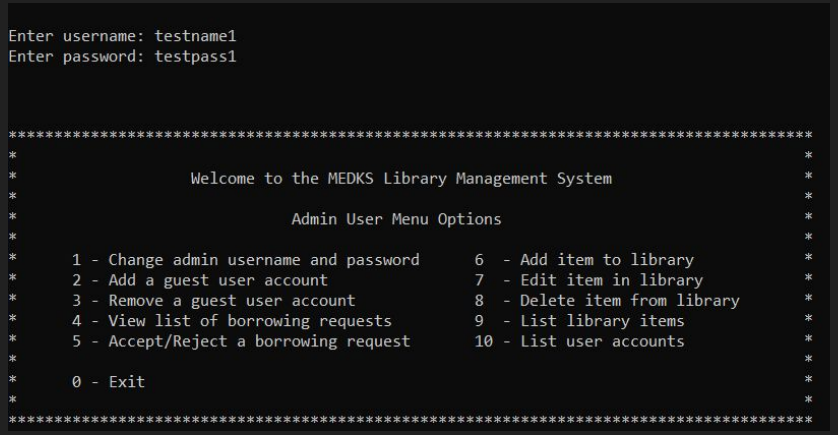
```
*****************************************
***  Request to Borrow a Library Item   ***
*****************************************


Enter ITEM ID Number:6

For how many DAYS?: 12
Borrow request for 12 days successfully submitted.

Press any key to continue...
```

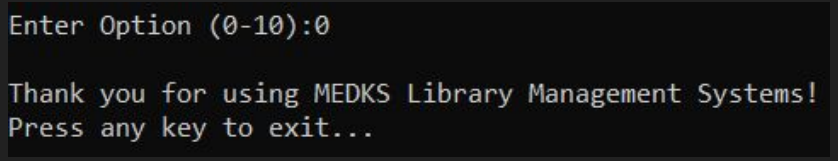
REQ ID	REQUESTER	TITLE	AUTHOR	ISBN	FORMAT	LOC	QTY	PRICE	DAYS
1	testname2	Atlas	The Score	6739820817	CD	888	12	$ 9.99	14
2	testname3	Animal Farm	George Orwell	9780241196	paperback	321	7	$ 5.99	28
3	testname4	Fahrenheit 451	Ray Bradbury	8821407053	hardcover	101	1	$ 14.99	21

Section 3

LMS user-friendly features:

- A welcome page
- A menu of all functions available to the user / admin
- Illustrated reports of requested / returned item(s) in tabular form
- LMS exit function with a ‘Thank You’ message
- Warning labels if the administrator tries to add a new, pre-existing item to the inventory list, a guest user tries to borrow more than three items at a time, or a user search request return null items





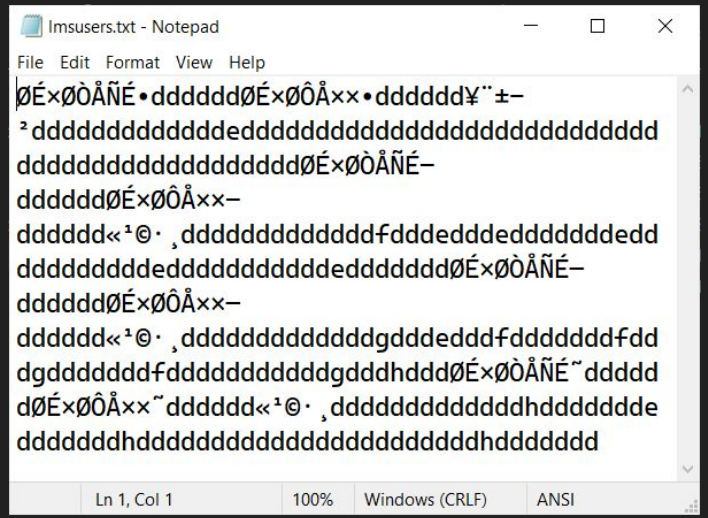

```
#define MAXBOOKSREQUESTED 3
#define MAXBOOKSLOANED 3
```

REQ ID	REQUESTER	TITLE	AUTHOR	ISBN	FORMAT	LOC	QTY	PRICE	DAYS
1	testname2	Atlas	The Score	6739820817	CD	888	12	$ 9.99	14
2	testname3	Animal Farm	George Orwell	9780241196	paperback	321	7	$ 5.99	28
3	testname4	Fahrenheit 451	Ray Bradbury	8821407053	hardcover	101	1	$ 14.99	21

Section 4

Protection features:

- Encryption of all user information in the USERFILE
- Decryption of all user information in the USERFILE




```c
#include <stdio.h>
#include <stdlib.h>

int encrypt(char fname[100]);
int decrypt(char fname[100]);
int createStatus();
```

```c
//Open user account file

if (createStatus() == 1) {
    //printf("Status = 1, won't decrypt pre-encrypted file");
} else if (createStatus() == 0) {
    //printf("Status = 0, encrypted file would be decrypted here");
    decrypt(USERFILE);
}
```

```c
    userfile = fopen(USERFILE, "w");
    if (userfile == NULL) {
      printf("\nError - Unable to write to USERFILE              ");
      printf("\n        User account information will not be stored");
    }
    // write 'user' array records to external file
    else {
      for (i = 1; (i ≤ NumUsers); i++) {
        fwrite(&user[i], sizeof(struct user_record), 1, userfile);
      }
    fclose(userfile);
    encrypt(USERFILE);
    }
}
```

Library Inventory/ Test Data

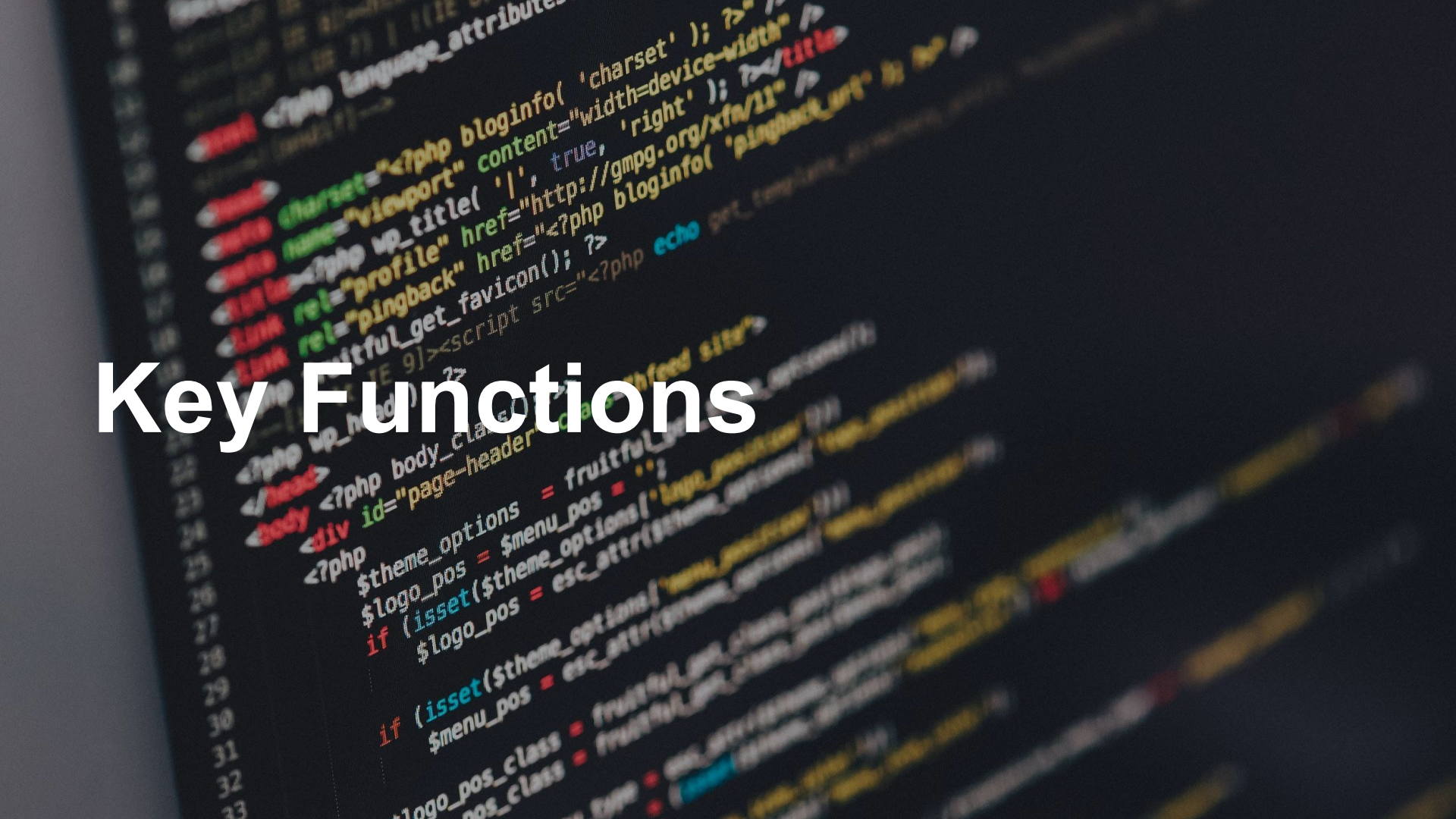
Inventory Format: ID, Title, Author, Subject, ISBN, Format, Location, Quantity, Price

01, Atlas, The Score, Music, 6739820817, 12, Shelf 888, $9.99
02, Harry Potter, J.K Rowling, Literature, 2745394572, 8, Shelf 777, $11.00
03, Animal Farm, George Orwell, Literature, 9780241196, 7, Shelf 321, $5.99
04, Fahrenheit 451, Ray Bradbury, Literature, 8821407053, 1, Shelf 101, $14.99
05, Calculus for Dummies, Mark Ryan, Literature, 9780764524, 17, Shelf 54, $17.99
06, Art of War, Sun Tzu, Literature, 9781982530, 9, Shelf 985, $3.99
07, Cleopatra, The Lumineers, Music, 225830971, 15, Shelf 762, $7.99
08, Weapons of Math Destruction, Cathy O’Neil, Literature, 4782938871, 3, Shelf 66, $4.99
09, Die Hard Series, 20th Century Studios, Film, 8938094712, 5, Shelf 151, $24.99
10, A Night at the Opera, Queen, Music, 3303981343, 12, Shelf 413, $7.99

User Format: Name, Password, Type, ID [Books Loans and Requests exist as well, but are not in table]

testname1, testpass1, ADMIN, 1
testname2, testpass2, GUEST, 2
testname3, testpass3, GUEST, 3
testname4, testpass4, GUEST, 4

Key Functions



Administrator Functions

changeAdminUser()

addGuestUser()

deleteGuestUser()

listRequests()

processRequests()

addItemLibrary()

editItemLibrary()

deleteItemfromLibrary()

listLibrary()

listUsers()

Guest Functions

searchForLibraryItem()

requestToBorrowLibraryItem()

listLibrary()

listRequests()

Linked Functions

encrypt(char fname[100])

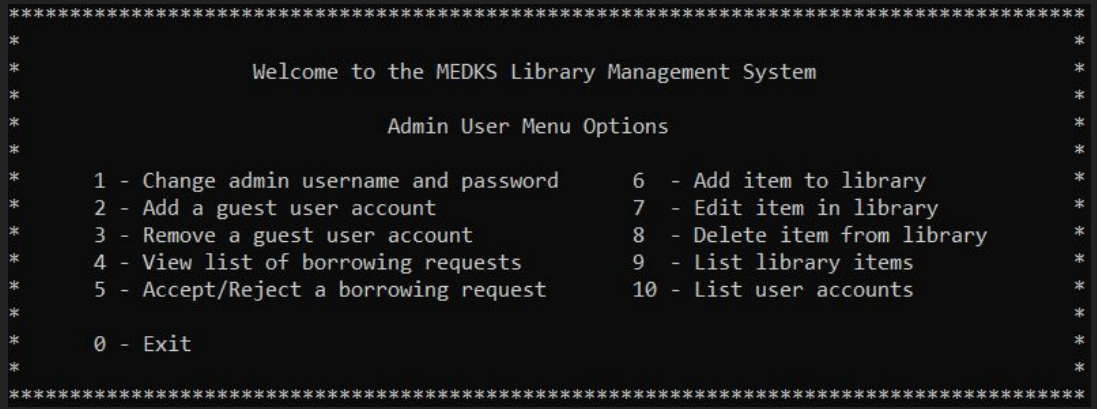
(“C Exercises: Encrypt”)

decrypt(char fname[100])

(“C Exercises: Decrypt”)

createStatus()

(“Access(2)”)



Key Global / Local Variables

LIBRARYFILE

USERFILE

MAXITEMS

MAXSHLEFLOCATION

MAXREQUESTS

MAXUSERS

MAXITEMLENGTH

MAXITEMPRICE

MAXITEMGTY

MAXIDLENGTH

MINIDLENGTH

MAXBOOKSREQUESTED

MAXBOOKSLOANED

ADMIN

GUEST

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "encrypt.h"
#define LIBRARYFILE "lmslibrary.txt"
#define USERFILE "lmsusers.txt"
#define MAXITEMS 1000
#define MAXSHELFLOCATION 10000
#define MAXREQUESTS 1000
#define MAXUSERS 100
#define MAXITEMLENGTH 41
#define MAXITEMQTY 1000
#define MAXITEMPRICE 100000.00
#define MAXIDLENGTH 15
#define MINIDLENGTH 8
#define MAXBOOKSREQUESTED 3
#define MAXBOOKSLOANED 3
#define ADMIN "ADMIN"
#define GUEST "GUEST"
```

```c
struct request_record
{
 int id, userid, itemid, time;
};

struct request_record requestlist[MAXREQUESTS];  //this array contains a list of all borrowing requests

struct item_record
{
 char title[MAXITEMLENGTH], author[MAXITEMLENGTH], subject[MAXITEMLENGTH], isbn[MAXITEMLENGTH], format[MAXITEMLENGTH];
 int itemid, location, qty;
 float price;
}; // creates item_record custom data type

struct item_record library[MAXITEMS]; //this array contains all library content

struct user_record
{
 char name[MAXIDLENGTH], password[MAXIDLENGTH], type[MAXIDLENGTH];
 int id, nbloaned, nbrequested, reqid[MAXBOOKSREQUESTED], booksloaned[MAXBOOKSLOANED], booksrequested[MAXBOOKSREQUESTED];
}; // creates user_record custom data type

struct user_record user[MAXUSERS];  //this array contains all user account information
```

Key Local Variables

MaxAttempts

Placed in functions relating to user/admin login, describes the maximum amount of login attempts that could be made before the LMS software auto-closes (Default = 5)

FormatList[9][MAXITEMLENGTH]

A multidimensional array that stores the 9 possible item formats, “hardcover", "paperback", "journal", "dissertation", "magazine", "newspaper", "audiobook", "ebook", and "CD"


.TXT Files Used

List of Users

A separate text file, “lmsusers.txt” that lists user information: administrator and all possible guests, as well as borrow requests associated with specific users. This file is decrypted upon it being read by the LMS software and is encrypted upon closing the LMS software.

If this file does not exist, it will be created and filled with test data from the above Test User Data Table.

List of Library Items

A separate text file, “lmslibrary.txt” that lists the library inventory: all items and associated details.

If this file does not exist, it will be created and filled with test data that comes from the above Test Library Inventory Table.

Encryption Checker File

This file “checker.txt” is created after first using the LMS software, and re-created if it does not exist.

When the LMS software is started, this file’s existence is checked, and, if not present, decryption of a non-existent or non-encrypted file will not carry out. If present, the encrypted USERFILE will be decrypted upon reading as normal.

The file helps determine whether the USERFILE is to be decrypted or not.

README File

Contains pertinent information about using the MEDKS LMS software

General Findings

- While previous lectures introduced a brief concept of encryption and decryption via Caesar’s cipher, a new concept the group learned was how to encrypt and decrypt a whole text file by shifting ASCII values, rather than individual characters in a single line of code. Researching and learning about this new method of encryption and decryption allowed us to properly protect the information stored in text files in an orderly fashion.

- The research we carried out to form this project not only improved the security for the User’s data, but to make our coding challenge slightly easier. In programming, researching new documentation and examples on how to complete certain tasks is a must to learn how to properly implement them yourself, and our group certainly gained knowledge from the research we had to do to translate the ideas in our mind to C code.

References

“Access(2) - Linux Man Page.” *Die.net*, linux.die.net/man/2/access.

“C Exercises: Encrypt a Text File.” *Atlantic.net*, w3resource, 26 Feb. 2020, www.w3resource.com/c-programming-exercises/file-handling/c-file-handling-exercise-13.php.

“C Exercises: Decrypt a Previously Encrypted File.” *Atlantic.net*, w3resource, 26 Feb. 2020, www.w3resource.com/c-programming-exercises/file-handling/c-file-handling-exercise-14.php.