

PES UNIVERSITY EC CAMPUS

SECURE PROGRAMMING

WITH C

# Project Report

LIBRARY MANGEMENT SYSYTEM

# 

# MEMBER DETAILS:

Shreyas G PES2UG19CS387 F

S Preetha PES2UG19CS343 F

Rohan George PES2UG19CS910 G

**Table of Content**

**1.0 INTRODUCTION 2.0 Requirement Specification**

**2.1 Safety Requirement Specification 2.1.1 Influence of the following 2.1.1.1 Programming Language**

**2.1.1.2 Compiler**

**2.1.2 Software features 2.1.2.1 Data Types used**

**2.1.2.2 Functions used and its descriptions**

**2.2 List of recommendations used in the project**

**2.3 Justification for the recommendations used in the software**

**1.1 Problem definition**

**1.2 Need of safe programming**

**3.0 Design and Workflow**

**3.1 Modularity and Reusability**

**3.2 Encapsulation**

**4.0 Implementation (Entire Code)**

**5.0 Conclusion**

**6.0 Reference**



**PES UNIVERSITY EC Campus**

**1 KM before Electronic City, Hosur Road, Bangalore-100**

**Department of Computer Science Engineering**

CERTIFICATE

Certified that the project work entitled **Library Management System by CERT Recommendations – Security Solutions th(at) work** carried out by **Shreyas G** bearing SRN **PES2UG19CS387** and **S Preetha** bearing SRN **PES2UG19CS343** and **Rohan George** bearing SRN **PES2UG19CS910**,Bonafide students of **IV semester** in partial fulfillment for the award of **Bachelor of Engineering** in PES University during the year **2021** .

The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the course **Secured Programming with C**.

**……………………………………….. ………………………………..**

**GUIDE Dr. Sandesh**

**Professor Vishwachetan D HOD, CSE**

**Declaration**

I hereby declare that the project entitled **“Library Management System- Safeguarded by CERT Recommendations – Security Solutions th(at) work”** submitted for Bachelor of Computer Science Engineering degree is my original work and the project has not formed the basis of the awards of any degree, associate ship, fellowship or any other similar titles.

**Signature of the Student:**  Shreyas G

**Place:** Bengaluru

**Date:** 09/05/2021

**Signature of the Student:** S Preetha

**Place:** Bengaluru

**Date:** 09/05/2021

**Signature of the Student:** Rohan George

**Place:** Bengaluru

**Date:** 09/05/2021

**Problem definition:**

To develop a C program to implement a library management system which will include all the functionalities required for daily operations.

The need of this particular application is to solve the needs of library users and provide a platform through which a user can access a library with ease and borrow books and return books with the least intervention.

The program is designed with a login logic and flow which ensures correct matching of user to the book that they have borrowed.

This will ensure a seamless experience for a user who is looking to borrow a book and use a library which is enabled by the application

The application implements all these flows successfully and ensures necessary security vulnerabilities are taken care of and ensure the best possible security standard.

The application uses files to store the status of books and users and attempts to handle all the security vulnerabilities that arise from the use of file handling

**Requirement Specifications:**

Software required for the successful execution of the application

* Windows/Linux/any UNIX based systems (commands are tested and optimized for the same)
* GCC compiler (preferably version 10.3)
* Active terminal interface for interacting with the application

**Testing software used:**

* Splint used for static analysis
* Flawfinder
* Cross matching security specifications from the CERT guidelines

**Problem Definition and Need:**

**Problem Definition:**

To develop a C code providing hotel room booking service facilities, and dealing with the several possible vulnerabilities that could exploit the application.

**Need of safe programming:**

As a programmer, it is not only your job but also moral responsibility to ensure that your codes don’t have any margin which can be later on exploited by any other Black Hat Hacker. This is what *secure coding* is all about.

Secure coding is the practice of developing computer software in a way that guards against the accidental introduction of security vulnerabilities. Defects, bugs and logic flaws are consistently the primary cause of commonly exploited software vulnerabilities*.*

Software security is a top concern today. You can’t risk any security vulnerabilities and that means your code needs to be secure and free of coding errors.

When you think about software security, you probably think about passwords and access control or viruses, spoofing, and phishing attacks. These are common security concerns. And security features, such data encryption and authentication protocols, mitigate these vulnerabilities. But even if you’ve implemented these security features, software can remain vulnerable.

Now, who can decide what safe way of coding is? It is not something that a single programmer can do. Thankfully, we don’t need to bother about it, the SEI CERT Coding Standards has a very nice collection of recommended steps to take to ensure that your program is secure and that also sorted according to the programming languages – C, C++, Java, Perl, and Android.

This project can do both, we could use secure programming to develop a secure software and we also had the scope to wisely use the recommendations such that we could rise the security standards implemented in our project. We needed safe programming for consistent results for mathematical calculations, encryption systems, safe string processing and many other core applications.

The preferences and recommendations prescribed by the CERT has helped us in writing a safe and secure code. Starting from variable description to the complicated memory allocation, the CERT rules have played a major role in minimizing the syntax and run-time errors and also provided a solution to encounter them. Our application needs a safe program and CERT recommendations is of high technical importance to us.

**Safety Requirement Specification:**

**Influence of the following:**

**Programming Language**

C is a programming language that is very close to the memory (hardware) and does not have a heavy software built around it. C language assumes that the user has taken care of his program and doesn’t check for anything, as a developer it is so much more challenging to develop safe programs. For instance, if the input exceeds the memory size allocated C language does not throw an error and there is high risk of the data being lost, therefore we have had to take care of minute details of all possible edge cases in developing our application.

**Compiler**

We use gcc compiler. This is possibly the best compiler that we found that would suit our application, it gives the maximum number of warnings when compared to its counterparts. This way we’ve found an optimal way to make the safest possible code powered by CERT recommendations and rules.

**Data Structures and data types used:**

●Array of structures- to store the personal information and details of each

member of the library

●Character array- to store the names of members/to story the book and

member details

●Long int- to store the customer’s phone number/member unique ID

●Integer array- To store the due date and borrowed date for each

book/member

●Double- To store the penalty amount if any or the borrowing charges

Structures used:

typedef struct libstatus

{

    char title[40];

    char author[40];

    char status[40];

    char user[40];

    char duedate[20];

} libschema;

//we used the above structure to store lib status

typedef struct userd

{

    char name[40];

    char mno[40];

    char email[40];

    char passwd[40];

} uschema;

//used the uschema structure to store the user details

**Functions used and its descriptions:**

**Functions.h**

1. **void signup\_ui();**

Consists of all the signup mechanisms used, will take in all the details required to onboard a user into the system

1. **void login\_ui();**

this is the login ui to enter the details and login to the system

1. **void main\_splash\_screen();**

This function consists of all the main screen elements which is the opening screen to which the user is shown

1. **void clrscr();**

Function used to clear the screen for UI effects

1. **void return\_flow(char \*path, char \*name);**

Function leads the user to the return book flow

1. **void borrow\_flow(char \*path, char \*name);**

Function takes the user to the borrow book flow, allowing him to borrow a book from the

1. **void delay(int number\_of\_seconds);**

Function to add delay for ui based transitions on an input of seconds

1. **int c\_newuser(char \*, char \*, char \*, char \*);**

Function to update the new user to the

1. **void main\_screen\_ui(char \*name);**

Main screen to enter option for borrow return and check availability

1. **int login\_check(char \*name, char \*passwd)**

Login check takes in the name and password and checks it with the file database if the value matches and returns a check value

1. **void show\_avail(char \*, char \*);**

This shows the availability of books in the library db and the particular availability of the book with a basic description of the book

1. **libschema \*libstruct(char \*fp);**

this correlates to the struct libschema which is the format with which the file is to be written, fp which is taken as the parameter is the file pointer to where the library database is to be written

1. **void update\_file(char \*path, libschema \*books);**

This is used to update the file back with the new library schema with updated availability and other details

1. **uschema \*userstruct(char \*fp);**

This is the struct which is used for user schema to be written into the user db with all the required details.

**List of recommendations used in the project:**

We have 49 recommendations used in total.

Below is the list of recommendations used in the application:

RECOMMENDATIONS USED:

1) RECOMMENDATION: STR03-A. Do not inadvertently truncate a null terminated byte string.

2) RECOMMENDATION: Pass the size of the array whenever the array is passed to a function.

3) RECOMMENDATION: PRE04-A. Do not reuse a standard header file name.

4) (BEYOND\_SYLLABUS) RECCOMENDATION: PRE08-C. Guarantee that header file names are unique.

5) RECOMMENDATION: PRE30-C. Do not create a universal character name through concatenation.

6) RECOMMENDATION: DCL02-A. Use visually distinct identifiers.

7) RECOMMENDATION: DCL23-C. Guarantee that mutually visible identifiers are unique.

8) RECOMMENDATION: DCL03-A. Place const as the rightmost declaration specifier.

9) RECOMMENDATION: DCL04-C. Do not declare more than one variable per declaration.

10) RECOMMENDATION: DCL07-C. Include the appropriate type information in function declarators.

11) RECOMMENDATION: DCL00-C. Const-qualify immutable objects.

12) RECOMMENDATION: DCL01-C. Do not reuse variable names in subscopes.

13) RECOMMENDATION: DCL05-C. Use typedefs of non-pointer types only.

14) RECOMMENDATION: DCL06-C. Use meaningful symbolic constants to represent literal values.

15) RECOMMENDATION: DCL08-A. Declare function pointers using compatible types.

16) RECOMMENDATION: EXP03-C. Do not assume the size of a structure is the sum of the sizes of its members.

17) RECOMMENDATION: STR05-C. Use pointers to const when referring to string literals.

18) RULE: STR30-C. Do not attempt to modify string literals.

19) (BEYOND\_SYLLABUS) RECOMMENDATION: STR11-C. Do not specify the bound of a character array initialized with a string literal.

20) RECOMMENDATION: STR31-C. Guarantee that storage for strings has sufficient space for character data and the null terminator.

21) RECOMMENDATION: STR10-C. Do not concatenate different type of string literals.

22) RECOMMENDATION: MEM03-C. Clear sensitive information stored in reusable resources.

37) RECOMMENDATION: MEM06-A. Do not use user-defined functions as parameters to allocation routines.

38) RECOMMENDATION: MEM32-C Detect and handle critical memory allocation errors.

39) RECOMMENDATION: MEM35-C. Allocate sufficient memory for an object.

40) (BEYOND\_SYLLABUS) RECOMMENDATION: INT30-C. Ensure that unsigned integer operations do not wrap.

41) RECOMMENDATION: ENV03-C. Sanitize the environment when invoking external programs.

42) RECOMMENDATION: STR00-A. Use TR 24731 for remediation of existing string manipulation code.

43) RECOMMENDATION: STR01-A. Use managed strings for development of new string manipulation code.

43)RECOMMENDATION: INT02-C. Understand integer conversion rules

44)RECOMMENDATION: INT05-C. Do not use input functions to convert character data if they cannot handle all possible inputs

45) RECOMMENDATION: INT17-C. Define integer constants in an implementation-independent manner

46) RECOMMENDATION: STR37-C. Arguments to character-handling functions must be representable as an unsigned char

47) RECOMMENDATION: CWE-121: Buffer Overflow

48) RECOMMENDATION: CWE-338: Use of Cryptographically Weak Pseudo-Random Number Generator (PRNG)

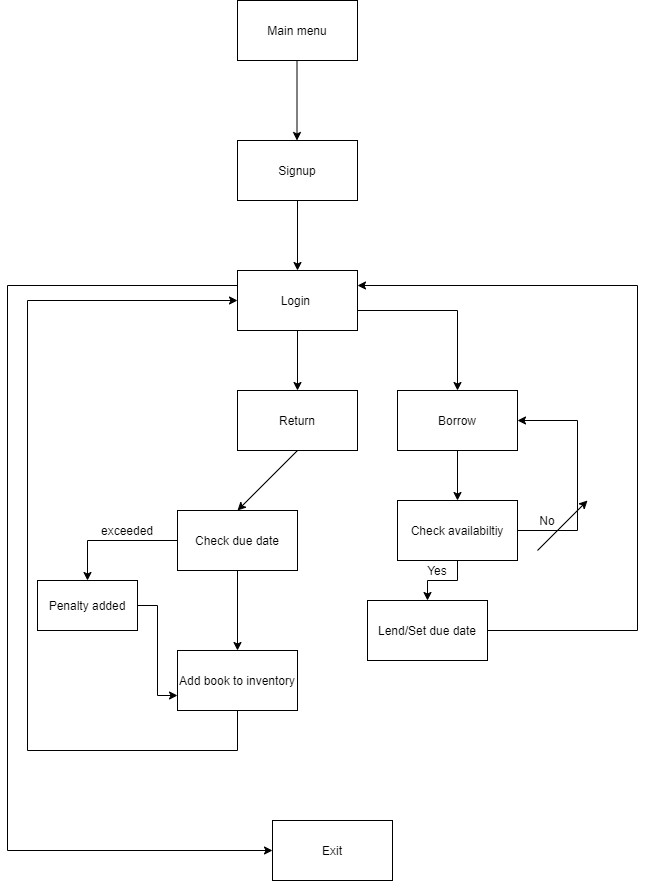
49) INT31-C. Ensure that integer conversions do not result in lost or misinterpreted data

**Justification for the recommendations used in the software:**

We listed down the programming scenarios for every recommendation first and we then developed functions that suited both the recommendation and our project concept.

We have commented out the use of the recommendation wherever needed and the recommendation is specified at the point of usage and anyone can easily make out how the recommendation is used by looking at the scenario it is used at.

**Design and Workflow of the project:**

The pictorial diagram of the layout of our project is: 

**Modularity and Reusability:**

Modularity refers to the extent to which a software can be divided into smaller modules. In our project every function performs a particular operation, this also gives the programmers an additional feature of **re-usability**

**Encapsulation:**

Encapsulation is a mechanism of wrapping the data (variables) together as a single unit and kept safe from outside interference and misuse.

typedef struct libstatus

{

    char title[40];

    char author[40];

    char status[40];

    char user[40];

    char duedate[20];

} libschema;

typedef struct userd

{

    char name[40];

    char mno[40];

    char email[40];

    char passwd[40];

} uschema;

CERT recommendations have helped us in encapsulating several data types into a single structure.

**Implementation**

**Code for the Compliant Version :**

Main.c

#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

void main()

{

    //userstruct("test");

    main\_splash\_screen();

    //show\_avail("userdb.txt");

}

**Helper\_functions.c:**

#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <conio.h>

void signup\_ui()

{

    clrscr();

    char \*name;

    name = (char \*)malloc(20);

    char \*mno;

    mno = (char \*)malloc(20);

    char \*email;

    email = (char \*)malloc(40);

    char \*passwd;

    passwd = (char \*)malloc(40);

    printf("\t\t\t\t\t\tWelcome to signup,enter your details below\n");

    printf("\t\t\t\t\t\tEnter your name :");

    scanf("%s", name);

    printf("\n\t\t\t\t\t\tEnter your mobile number:");

    scanf("%s", mno);

    printf("\n\t\t\t\t\t\tEnter your email id:");

    scanf("%s", email);

    printf("\n\t\t\t\t\t\tEnter your password:");

    scanf("%s", passwd);

    int status = c\_newuser(name, mno, email, passwd);

    free(name);

    free(mno);

    free(email);

    free(passwd);

    if (status == 1)

    {

        printf("\n\n\t\t\t\t\t\tUser succesfully created");

        delay(2);

        clrscr();

        main\_splash\_screen();

    }

    else

        printf("\n\n\t\t\t\t\t\tUser creation unsuccesful");

}

void login\_ui()

{

    clrscr();

    char \*name;

    name = (char \*)malloc(20);

    char lpswd[20];

    char c;

    int z = 0;

    //c = (char \*)malloc(4);

    printf("\t\t\t\t\t\tWelcome to login,enter your details below\n");

    printf("\t\t\t\t\t\tEnter your name :");

    scanf("%s", name);

    printf("\n\t\t\t\t\t\tEnter your password:");

    while ((c = getch()) != 13)

    {

        lpswd[z++] = c;

        printf("%c", '\*');

    }

    lpswd[z] = '\0';

    //printf("%s", lpswd);

    int rvalue = login\_check(name, lpswd);

    //printf("%d", rvalue);

    if (rvalue == 1)

    {

        main\_screen\_ui(name);

    }

    else if (rvalue == 2)

    {

        printf("\n\t\t\t\t\t\tSorry,your entered password is wrong,please try again\n");

        delay(2);

        login\_ui();

    }

    else if (rvalue == 3)

    {

        printf("\n\t\t\t\t\t\tSorry,this user does not exist\n");

        delay(2);

        login\_ui();

    }

    free(name);

}

uschema \*userstruct(char \*fp)

{

    FILE \*lfp = fopen(fp, "r");

    uschema \*uarr = malloc(sizeof(uschema) \* 20);

    char name[40];

    char mno[40];

    char mail[40];

    char passwd[40];

    int count = 0;

    while (fscanf(lfp, "%s %s %s %s\n", name, mno, mail, passwd) != EOF)

    {

        strcpy(uarr[count].name, name);

        strcpy(uarr[count].mno, mno);

        strcpy(uarr[count].email, mail);

        strcpy(uarr[count].passwd, passwd);

        count += 1;

    }

    fclose(lfp);

    return uarr;

}

libschema \*libstruct(char \*fp)

{

    FILE \*lfp = fopen(fp, "r");

    libschema \*uarr = malloc(sizeof(uschema) \* 20);

    char title[40];

    char author[40];

    char status[40];

    char user[40];

    char duedate[40];

    int count = 0;

    int tcount = 0;

    while (tcount != 20)

    {

        strcpy(uarr[tcount].title, "\0");

        tcount += 1;

    }

    while (fscanf(lfp, "%s %s %s %s %s\n", title, author, status, user, duedate) != EOF)

    {

        strcpy(uarr[count].title, title);

        strcpy(uarr[count].author, author);

        strcpy(uarr[count].status, status);

        strcpy(uarr[count].user, user);

        strcpy(uarr[count].duedate, duedate);

        count += 1;

    }

    fclose(lfp);

    return uarr;

}

void update\_file(char \*path, libschema \*books)

{

    FILE \*fp = fopen(path, "w");

    int flag = 0;

    while (flag < 5)

    {

        fprintf(fp, "%s %s %s %s %s\n", books[flag].title, books[flag].author, books[flag].status, books[flag].user, books[flag].duedate);

        flag += 1;

    }

    fclose(fp);

}

void show\_avail(char \*path, char \*name)

{

    clrscr();

    libschema \*listbooks = libstruct(path);

    int cr = 0;

    printf("\t\t\t\tList of available books are\n");

    printf("\t\t\t\tTitle Author Status\n");

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

    while (cr != 5)

    {

        printf("\t\t\t\t%s %s %s\n\n", listbooks[cr].title, listbooks[cr].author, listbooks[cr].status);

        cr += 1;

    }

    main\_screen\_ui(name);

}

int login\_check(char \*name, char \*lpswd)

{

    uschema \*cuser = userstruct("userdb.txt");

    int c = 0;

    while (cuser[c].email != NULL)

    {

        if (strcmp(cuser[c].name, name) == 0)

        {

            //printf("%s\n", lpswd);

            //printf("%s", cuser[c].passwd);

            if (strcmp(cuser[c].passwd, lpswd) == 0)

            {

                return 1;

            }

            else

            {

                return 2;

            }

        }

        c += 1;

    }

    return 3;

}

void main\_splash\_screen()

{

    printf("\t\t\t\t\t\t\*\*\*\*Welcome to our library management system!\*\*\*\*\n");

    printf("\n\t\t\t\t\t\tChoose your option below to access the library\n");

    printf("\n\t\t\t\t\t\t\t1.Signup to the library");

    printf("\n\t\t\t\t\t\t\t2.Login to the library");

    printf("\n\n\t\t\t\t\t\tEnter your option below:\n");

    int chval;

    scanf("\t\t\t\t\t\t\t%d", &chval);

    if (chval == 1)

    {

        signup\_ui();

    }

    else if (chval == 2)

    {

        login\_ui();

    }

    else

    {

        printf("\t\t\t\t\tInvalid input received \n");

        delay(5);

        clrscr();

        main\_splash\_screen();

    }

}

void main\_screen\_ui(char \*name)

{

    printf("\n\t\t\t\t\tYou have succesfully logged into the library\n\n");

    printf("\t\t\t\t\tChose your options below\n");

    printf("\t\t\t\t\t1.Check book availability\n");

    printf("\t\t\t\t\t2.Borrow book\n");

    printf("\t\t\t\t\t3.Return book\n");

    int choice;

    scanf("\t\t\t\t%d", &choice);

    if (choice == 1)

    {

        show\_avail("libdb.txt", name);

    }

    if (choice == 2)

    {

        //show\_avail("libdb.txt", name);

        borrow\_flow("libdb.txt", name);

    }

    if (choice == 3)

    {

        return\_flow("libdb.txt", name);

    }

}

void borrow\_flow(char \*path, char \*name)

{

    libschema \*out = libstruct(path);

    char title[50];

    printf("\t\t\t\tPlease enter the name of the book would like to borrow below:");

    scanf("%s", title);

    int count = 0;

    while (count < 5)

    {

        //printf("%s", out[count].title);

        if (strcmp(out[count].title, title) == 0)

        {

            if (strcmp(out[count].status, "Available") == 0)

            {

                time\_t seconds;

                seconds = (time(NULL) / 3600) + 72;

                char time[50];

                itoa(seconds, time, 10);

                printf("\t\t\t\tyou have successfully borrowed the book\n");

                strcpy(out[count].status, "Borrowed");

                strcpy(out[count].duedate, time);

                strcpy(out[count].user, name);

                update\_file("libdb.txt", out);

                main\_screen\_ui(name);

            }

            else

            {

                printf("\t\t\t\tEntered book is not available\n");

            }

        }

        else

        {

            count += 1;

            continue;

        }

    }

    printf("\t\t\t\tEntered book is not available in our inventory\n");

}

int c\_newuser(char \*name, char \*mno, char \*email, char \*passwd)

{

    FILE \*fp;

    fp = fopen("userdb.txt", "a");

    fprintf(fp, "%s %s %s %s\n", name, mno, email, passwd);

    fclose(fp);

    return 1;

}

void return\_flow(char \*path, char \*name)

{

    libschema \*rflow = libstruct(path);

    int c = 0;

    char opt[10];

    while (c <= 5)

    {

        if (strcmp(rflow[c].user, name) == 0)

        {

            printf("\t\t\t\tYou have one pending book to return\n\n");

            printf("\t\t\t\t%s %s %s\n\n", rflow[c].title, rflow[c].author, rflow[c].status);

            printf("\t\t\t\tPlease enter yes to return this book\n");

            scanf("%s", opt);

            if (strcmp(opt, "yes") == 0)

            {

                int days = atoi(rflow[c].duedate);

                time\_t seconds;

                seconds = (time(NULL) / 3600);

                if ((seconds - days) < 72)

                {

                    printf("\t\t\t\tNo fine will be applied\n\n");

                    strcpy(rflow[c].status, "Available");

                    strcpy(rflow[c].user, "null");

                    strcpy(rflow[c].duedate, "null");

                    update\_file("libdb.txt", rflow);

                    free(rflow);

                    main\_screen\_ui(name);

                }

                else

                {

                    printf("\t\t\t\tPlease pay a fine of Rs.50\n\n");

                    free(rflow);

                    main\_screen\_ui(name);

                }

            }

            else

            {

                printf("\t\t\t\tInvalid input received,please reenter");

                free(rflow);

                return\_flow(path, name);

            }

        }

        else

        {

            c += 1;

            continue;

        }

    }

    if (c > 5)

    {

        printf("\t\t\t\tYou have no pending book to return\n");

        printf("\t\t\t\tReturning to main screen");

        delay(3);

        clrscr();

        free(rflow);

        main\_screen\_ui(name);

    }

}

void clrscr()

{

    system("clear");

}

void delay(int number\_of\_seconds)

{

    // Converting time into milli\_seconds

    int milli\_seconds = 1000 \* number\_of\_seconds;

    // Storing start time

    clock\_t start\_time = clock();

    // looping till required time is not achieved

    while (clock() < start\_time + milli\_seconds)

        ;

}

**Function prototypes for compliant code:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

//#include <conio.h>

typedef struct libstatus

{

    char title[40];

    char author[40];

    char status[40];

    char user[40];

    char duedate[20];

} libschema;

typedef struct userd

{

    char name[40];

    char mno[40];

    char email[40];

    char passwd[40];

} uschema;

void signup\_ui();

void login\_ui();

void main\_splash\_screen();

void clrscr();

void return\_flow(char \*path, char \*name);

void borrow\_flow(char \*path, char \*name);

void delay(int number\_of\_seconds);

int c\_newuser(char \*, char \*, char \*, char \*);

void main\_screen\_ui(char \*name);

int login\_check(char \*name, char \*passwd);

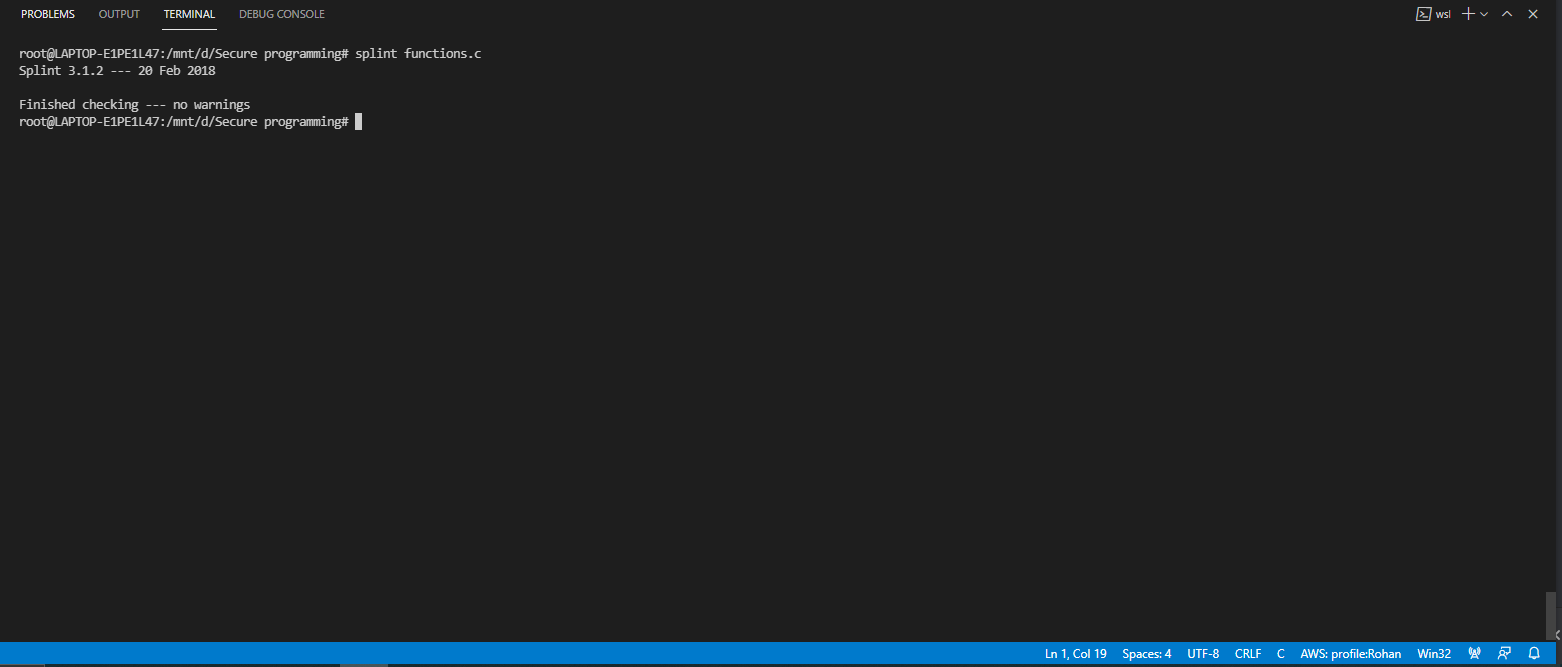
void show\_avail(char \*, char \*);

libschema \*libstruct(char \*fp);

void update\_file(char \*path, libschema \*books);

uschema \*userstruct(char \*fp);

Output :



**Non-compliant code**

#include "functions.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <conio.h>

void signup\_ui()

{

    clrscr();

    char \*name;

    name = (char \*)malloc(20);

    char \*mno;

    mno = (char \*)malloc(20);

    char \*email;

    email = (char \*)malloc(40);

    char \*passwd;

    email = (char \*)malloc(40);

    printf("\t\t\t\t\t\tWelcome to signup,enter your details below\n");

    printf("\t\t\t\t\t\tEnter your name :");

    scanf("%s", name);

    printf("\n\t\t\t\t\t\tEnter your mobile number:");

    scanf("%s", mno);

    printf("\n\t\t\t\t\t\tEnter your email id:");

    scanf("%s", email);

    printf("\n\t\t\t\t\t\tEnter your password:");

    scanf("%s", passwd);

    int status = c\_newuser(name, mno, email, passwd);

    free(name);

    free(mno);

    free(email);

    free(passwd);

    if (status == 1)

    {

        printf("\n\n\t\t\t\t\t\tUser succesfully created");

        delay(2);

        clrscr();

        main\_splash\_screen();

    }

    else

        printf("\n\n\t\t\t\t\t\tUser creation unsuccesful");

}

void login\_ui()

{

    clrscr();

    char \*name;

    name = (char \*)malloc(20);

    char lpswd[20];

    char c;

    int z = 0;

    //c = (char \*)malloc(4);

    printf("\t\t\t\t\t\tWelcome to login,enter your details below\n");

    printf("\t\t\t\t\t\tEnter your name :");

    scanf("%s", name);

    printf("\n\t\t\t\t\t\tEnter your password:");

    while ((c = getch()) != 13)

    {

        lpswd[z++] = c;

        printf("%c", '\*');

    }

    lpswd[z] = '\0';

    //printf("%s", lpswd);

    int rvalue = login\_check(name, lpswd);

    printf("%d", rvalue);

    if (rvalue == 1)

    {

        main\_screen\_ui(name);

    }

    else if (rvalue == 2)

    {

        printf("\n\t\t\t\t\t\tSorry,your entered password is wrong,please try again\n");

        delay(2);

        login\_ui();

    }

    else if (rvalue == 3)

    {

        printf("\n\t\t\t\t\t\tSorry,this user does not exist\n");

        delay(2);

        login\_ui();

    }

}

uschema \*userstruct(char \*fp)

{

    FILE \*lfp = fopen(fp, "r");

    uschema \*uarr = malloc(sizeof(uschema) \* 20);

    char name[40];

    char mno[40];

    char mail[40];

    char passwd[40];

    int count = 0;

    while (fscanf(lfp, "%s %s %s %s\n", name, mno, mail, passwd) != EOF)

    {

        strcpy(uarr[count].name, name);

        strcpy(uarr[count].mno, mno);

        strcpy(uarr[count].email, mail);

        strcpy(uarr[count].passwd, passwd);

        count += 1;

    }

    fclose(lfp);

    return uarr;

}

libschema \*libstruct(char \*fp)

{

    FILE \*lfp = fopen(fp, "r");

    libschema \*uarr = malloc(sizeof(uschema) \* 20);

    char title[40];

    char author[40];

    char status[40];

    char user[40];

    char duedate[40];

    int count = 0;

    int tcount = 0;

    while (tcount != 20)

    {

        strcpy(uarr[tcount].title, "\0");

        tcount += 1;

    }

    while (fscanf(lfp, "%s %s %s %s %s\n", title, author, status, user, duedate) != EOF)

    {

        strcpy(uarr[count].title, title);

        strcpy(uarr[count].author, author);

        strcpy(uarr[count].status, status);

        strcpy(uarr[count].user, user);

        strcpy(uarr[count].duedate, duedate);

        count += 1;

    }

    fclose(lfp);

    return uarr;

}

void update\_file(char \*path, libschema \*books)

{

    FILE \*fp = fopen(path, "w");

    int flag = 0;

    while (flag < 5)

    {

        fprintf(fp, "%s %s %s %s %s", books[flag].title, books[flag].author, books[flag].status, books[flag].user, books[flag].duedate);

        flag += 1;

    }

    fclose(fp);

}

void show\_avail(char \*path, char \*name)

{

    clrscr();

    libschema \*listbooks = libstruct(path);

    int cr = 0;

    printf("\t\t\t\tList of available books are\n");

    printf("\t\t\t\tTitle Author Status\n");

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

    while (cr != 5)

    {

        printf("\t\t\t\t%s %s %s\n\n", listbooks[cr].title, listbooks[cr].author, listbooks[cr].status);

        cr += 1;

    }

    main\_screen\_ui("user");

}

int login\_check(char \*name, char \*lpswd)

{

    uschema \*cuser = userstruct("userdb.txt");

    int c = 0;

    while (cuser[c].email != NULL)

    {

        if (strcmp(cuser[c].name, name) == 0)

        {

            //printf("%s\n", lpswd);

            //printf("%s", cuser[c].passwd);

            if (strcmp(cuser[c].passwd, lpswd) == 0)

            {

                return 1;

            }

            else

            {

                return 2;

            }

        }

        else

        {

            return 3;

        }

        c += 1;

    }

}

void main\_splash\_screen()

{

    printf("\t\t\t\t\t\t\*\*\*\*Welcome to our library management system!\*\*\*\*\n");

    printf("\n\t\t\t\t\t\tChoose your option below to access the library\n");

    printf("\n\t\t\t\t\t\t\t1.Signup to the library");

    printf("\n\t\t\t\t\t\t\t2.Login to the library");

    printf("\n\n\t\t\t\t\t\tEnter your option below:\n");

    int chval;

    scanf("\t\t\t\t\t\t\t%d", &chval);

    if (chval == 1)

    {

        signup\_ui();

    }

    else if (chval == 2)

    {

        login\_ui();

    }

    else

    {

        printf("\t\t\t\t\tInvalid input received \n");

        delay(5);

        clrscr();

        main\_splash\_screen();

    }

}

void main\_screen\_ui(char \*name)

{

    printf("\t\t\t\t\tYou have succesfully logged into the library\n");

    printf("\t\t\t\t\tChose your options below\n");

    printf("\t\t\t\t\t1.Check book availability\n");

    printf("\t\t\t\t\t2.Borrow book\n");

    printf("\t\t\t\t\t3.Return book\n");

    int choice;

    scanf("\t\t\t\t%d", &choice);

    if (choice == 1)

    {

        show\_avail("libdb.txt", name);

    }

    if (choice == 2)

    {

        borrow\_flow("libdb.txt", name);

    }

    if (choice == 3)

    {

        return\_flow("libdb.txt", name);

    }

}

void borrow\_flow(char \*path, char \*name)

{

    libschema \*out = libstruct(path);

    char title[50];

    printf("\t\t\t\tPlease enter the name of the book would like to borrow below");

    scanf("%s", title);

    int count = 0;

    while (count < 5)

    {

        if (strcmp(out[count].title, title) == 0)

        {

            if (strcmp(out[count].status, "Available") == 0)

            {

                time\_t seconds;

                seconds = (time(NULL) / 3600) + 72;

                char time[50];

                itoa(seconds, time, 10);

                printf("\t\t\t\tyou have successfully borrowed the book\n");

                strcpy(out[count].status, "Borrowed");

                strcpy(out[count].duedate, time);

                strcpy(out[count].user, name);

                update\_file("libdb.txt", out);

                main\_screen\_ui(name);

            }

            else

            {

                printf("\t\t\t\tEntered book is not available\n");

            }

        }

        else

        {

            printf("\t\t\t\tEntered book does not belong to our inventory\n");

        }

        count += 1;

    }

}

int c\_newuser(char \*name, char \*mno, char \*email, char \*passwd)

{

    FILE \*fp;

    fp = fopen("userdb.txt", "a");

    fprintf(fp, "%s %s %s %s\n", name, mno, email, passwd);

    fclose(fp);

    return 1;

}

void return\_flow(char \*path, char \*name)

{

    libschema \*rflow = libstruct(path);

    int c = 0;

    char opt[10];

    while (c <= 5)

    {

        if (strcmp(rflow[c].user, name) == 0)

        {

            printf("\t\t\t\tYou have one pending book to return\n\n");

            printf("\t\t\t\t%s %s %s\n\n", rflow[c].title, rflow[c].author, rflow[c].status);

            printf("\t\t\t\tPlease enter yes to return this book\n");

            scanf("%s", opt);

            if (strcmp(opt, "Yes") == 0)

            {

                int days = atoi(rflow[c].duedate);

                time\_t seconds;

                seconds = (time(NULL) / 3600);

                if ((seconds - days) < 72)

                {

                    printf("\t\t\t\tNo fine will be applied\n\n");

                    strcpy(rflow[c].status, "Available");

                    strcpy(rflow[c].user, "null");

                    strcpy(rflow[c].duedate, "null");

                    update\_file("libdb.txt", rflow);

                    main\_screen\_ui(name);

                }

                else

                {

                    printf("\t\t\t\tPlease pay a fine of Rs.50\n\n");

                    main\_screen\_ui(name);

                }

            }

            else

            {

                printf("\t\t\t\tInvalid input received,please reenter");

                return\_flow(path, name);

            }

        }

        else

        {

            printf("\t\t\t\tYou have no pending book to return\n");

            printf("\t\t\t\tReturning to main screen");

            delay(3);

            clrscr();

            main\_screen\_ui(name);

        }

        c += 1;

    }

}

void clrscr()

{

    system("clear");

}

void delay(int number\_of\_seconds)

{

    // Converting time into milli\_seconds

    int milli\_seconds = 1000 \* number\_of\_seconds;

    // Storing start time

    clock\_t start\_time = clock();

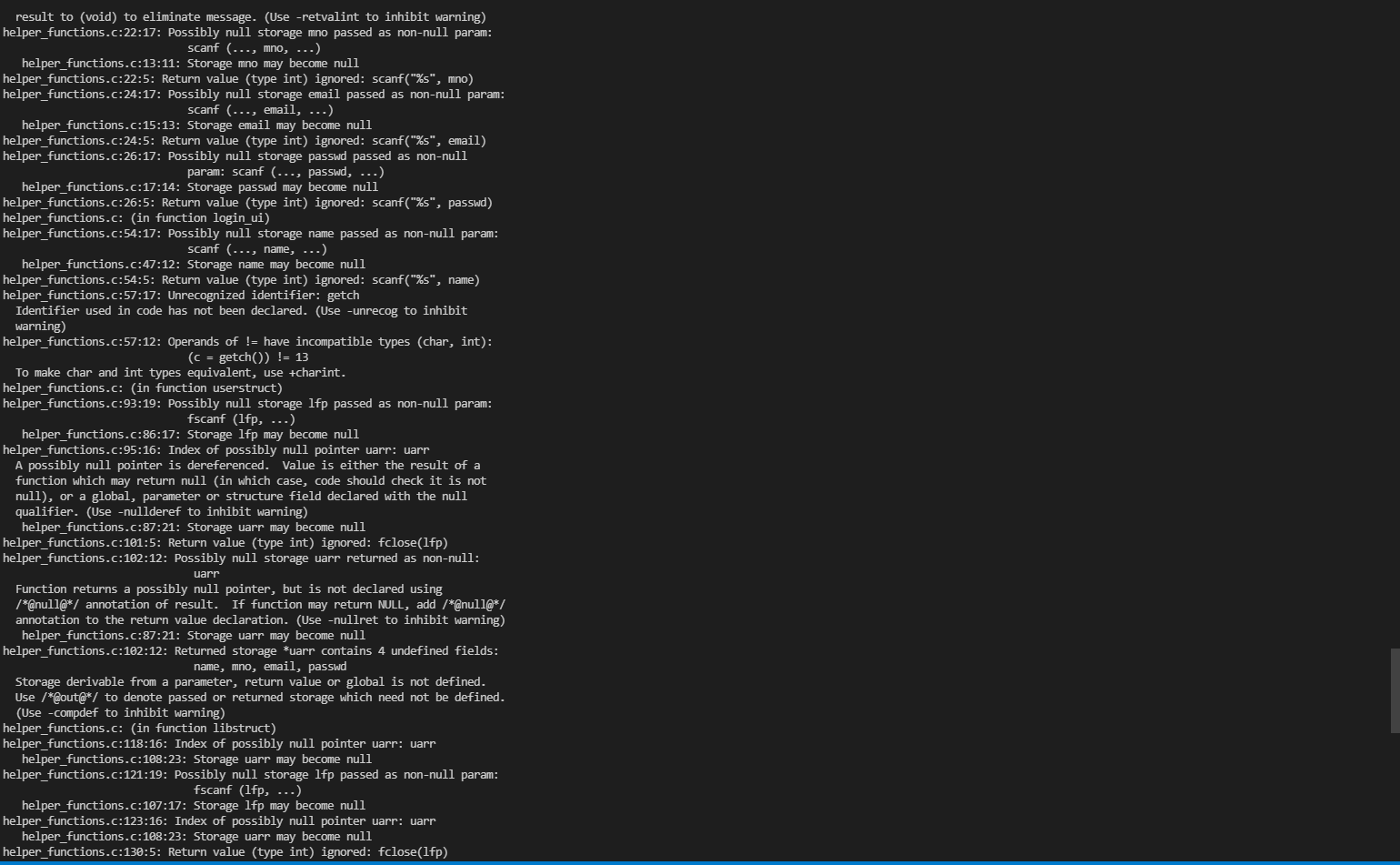
    // looping till required time is not achieved

    while (clock() < start\_time + milli\_seconds)

        ;

}

Splint output:



**Conclusion**

Why did we use CERT recommendations in our project? To prevent hackers from hacking our software. How will they hack it? They would give vulnerable inputs. How did we solve the problem?

There are errors other than syntactical, memory allocation, run-time and logical errors. These errors are not because of a fault from our end, but we were still open for vulnerabilities.

Effective use of static analysis tools, better results can be achieved.

Despite making a correct code logically and syntactically we have to protect ourselves from vulnerabilities, this is the purpose of implementing CERT recommendations in our project.

**References:**

<https://www.perforce.com/sites/default/files/pdfs/how-to-write-secure-code-c.pdf>

<http://web.mit.edu/6.s096/www/lecture/lecture03/secure-C.pdf>

<https://www.geeksforgeeks.org/secure-coding-what-is-it-all-about/>

https://splint.org/documentation/