CHAPTER 1

INTRODUCTION

The Library Management System is library management program for the purpose of monitoring and controlling the transaction in a library. This data structure project on the library management system gives us the complete information about the daily transactions done in a library.

1.1 IMPORTANCE OF THE TOPIC

We need to maintain the record of new and retrieve the details of books available in the library which mainly focuses on basic operations like adding new entry, new information, searching books and calculating the price if there are delayed returns. It features a familiar and well thought-out, an attractive user interface, combined with strong searching, insertion and reporting capabilities.

The report generation facility of the library helps to get a good idea of which is the book borrowed by the members. Here the end users are only the librarian. To maintain and update the records and also to cater the needs of the users.

1.2 WHERE IS IT USED? IT'S APPLICATIONS

The library management system application is majorly used and designed based on the college library. But this is just the library where there are gate books for just the Computer Science, Electronics & Communications and the Mechanical Engineering but anyone can borrow the books based on the particular book in the list when the librarian selects a branch.

1.3 WHAT WE DID?

We used the concept of linked list to develop this project. We have used array of strings to store the book names and the author names. We have made the job of the librarian easier we he has to just use single characters to record the data. It is really easy and simple to be used and can help in easy calculations for the due fees as well.

The librarian would just have to enter the borrow date and the return date and the late fees will automatically be generated by this application. The librarian can also search on the basis of late book returns.

The late return fess is calculated based on the number of days that the book was borrowed, that is if the book was borrowed for more than fifteen whole days and then display it an organised manner and call the person if necessary.

1.4 OVERVIEW

A college library management is a project that manages and stores books information electronically according to student's needs. The system helps the librarian to keep a constant track of all the books available in the library. It allows the admin to choose from the book list. It becomes necessary for colleges to keep a continuous check on the books issued and returned and even calculate fine.

This task if carried out manually will be tedious and includes chances of mistakes. These errors are avoided by allowing the system to keep track of information such as issue date, last date to return the book and even fine information and thus there is no need to keep manual track of this information which thereby avoids chances of mistakes.

Thus this system reduces manual work to a great extent allows smooth flow of library activities by removing chances of errors in the details.

In the next part of this report we will see what we have actually done in this project, how we have done it and most importantly why we have done it, along with the proof of outcomes and the basic flow chart of how this code runs along with some pseudo codes and snippets of the data structures used along in this project and how it can be improvised later.

CHAPTER 2

PROBLEM STATEMENT

The main aim of this library is to provide gate books for the students who are willing to study and start preparing for the gate examinations in the CSE,ECE and the ME department of the college. So to make the data entry for the librarian more easy and accessible, we have made this console based data structures program using linked list and array of strings, So here are the problems that were faced ,

2.1 EXISTING SYSTEM

The existing system is a manual one. Different records are maintained for different transactions of the Library. When a new transaction takes place, the Librarian enters the details of the transactions in a new file depending upon the type of the transaction. The staffs have to maintain different type of operation like keeping details of the members i.e. General member as well as Student, detail records of books, keeping track of members newly registered moreover financial transactions like late fees for the period.

The reports are generated from one point time to other time for various operations; these should also be produced to the higher authority in timely manner. The information regarding the system needs interaction and presentation at regular intervals of time.

2.2 PROBLEMS WITH THE EXISTING SYSTEM

Since the system is a manual one, the new transactions become time consuming. The procedure of creation and deletion with the new transaction requires much interaction with the system. Those records may already be in use at some level of processing. This causes delay in the process.

The lack of availability of information in the same manner between different modules of the system slows the whole process. The lack of coordination of the staffs and procedures in a horizontal manner can also speed down the whole process, which requires more or less uninterrupted flow of information.

2.3 NEED FOR COMPUTERIZED SYSTEM

A computerized system is needed mainly because of the lack of speed in the manual system. Computerized system provides speed with accuracy. The new transaction entries can be made instantly. The modification and deletion of records regarding the transaction can be done in no time.

The new system will not only make modification and new transaction faster but also speedup the report generation. The generation of reports will become time effective and the scope of generation will also increase. The information flow will become faster. The transparency level will also increase. The coordination between different modules will also increase. The manpower and the paper work needed for maintaining the operation of the Library will be reduced by the introduction of the computerized system.

This application would also save a lot of time and paper for the library since everything is going to be digitalized and the librarian has to enter only the issue date and the return date and if there is delayed return of the book, then the delay fees is automatically calculated and displayed along with his phone number so that he can be contacted later on.

CHAPTER 3

OBJECTIVES

The objective of the Library Management System is to handle the entire activity of a library. The application keeps track of all the information of the books along with its authors in the gate library along with the required details of who borrowed it.

By using library management system, the operation of borrowing and managing inventories is paperless. This system provides a user-friendly data entry. This system will store all the books and members information that borrowed the book and also display the late fees of that particular consumer.

Analysis, working on the preliminary should accomplish the following objectives:

1. Benefits to be provided by the system

Here the system replaces a manual system of maintenance of member details, book details and monetary transactions.

2. Gathering Information

This deals with the gathering of information of the way the organizers are maintaining the above said records, their manuals if present or if they likely to public them for the development of the new system.

3. Getting Knowledge of the project request

This is concerned with user's expectation from the system to be developed. This is however already covered in the "Identification of need".

4. Rough estimation of the costing for the system

This is concerned with an outline costing needed to develop the system as a system must not only be work effective but also must be cost effective to be developed and to be brought for usage.

5. Feasibility of development

Based on all the information gathered the next step is to find out whether the project is suitable for development, whether the project could be developed or not under the circumstances planed for its development.

CHAPTER 4

SYSTEM REQUIREMENTS

A System Requirements Specification (SRS) (also known as a Software Requirements Specification) is a document or set of documentation that describes the features and behaviour of a system or software application. It includes a variety of elements that attempts to define the intended functionality required by the customer to satisfy their different users.

In addition to specifying how the system should behave, the specification also defines at a high-level the main business processes that will be supported, what simplifying assumptions have been made and what key performance parameters will need to be met by the system.

4.1 SOFTWARE REQUIREMENTS

The software requirements are description of features and functionalities of the target system. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view.

It defines how the intended software will interact with hardware, external interfaces, speed of operation, response time of system, portability of software across various platforms, maintainability, speed of recovery after crashing, Security, Quality, Limitations etc.

The software requirements in our program are

1. Programming language: C.
2. Compiler: gcc compiler with Kali as the client.
3. Operating System: Kali Linux, Ubuntu, Windows.
4. Some externally installed library functions for array of strings or else, this code externally used can throw some warnings. But with this installed, the code is error and warning free.

4.2 HARDWARE REQUIREMENTS

There are hardware requirements, also known as system requirements, for every OS we are going to use. These requirements include the minimum processor speed, memory, and disk space required to install Windows. In almost all cases, you will want to make sure that your hardware exceeds these requirements to provide adequate performance for the services and applications running on the server. The table below outlines the minimum hardware requirements to execute this project.

1. Processor: Any 3rd Gen+ Intel Processor or even a 1st gen AMD processor.
2. RAM: Since here it was done on VMware Workstation 15, it might require minimum of 4GB of RAM, if it was just done on Ubuntu then 2GB of RAM.
3. Graphics: Minimal.

4.3 WHY KALI LINUX?

Kali Linux (successor of Backtrack) was originally made as a tool set for advance penetration testing and finding vulnerabilities in network systems.

Kali is based on Debian, However, unlike Debian it is focused on forensics and programming. For this reason Kali preinstalls packages relevant for programming. Kali also actively seeks bugs in string-related packages. Thus Kali saves you from finding and installing string packages. It also keeps you informed about bugs in these packages. Furthermore it provides a community platform for those interested in forensics.

The main point is still that Kali preinstalls useful things for a specific domain, and it provides a community platform. General purpose distributions like Debian and Ubuntu do not have a similar focus.

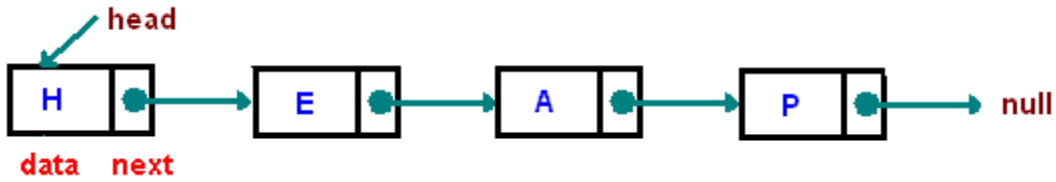
Apart from useful tools, Kali Linux is a wonderful Linux distribution with all the inbuilt functions for character array, hence it is more preferable.

CHAPTER 5

METHODOLOGY

In our project, the data structure used is the linked list. A linked list is a linear data structure where each element is a separate object.

5.1 LINKED LIST

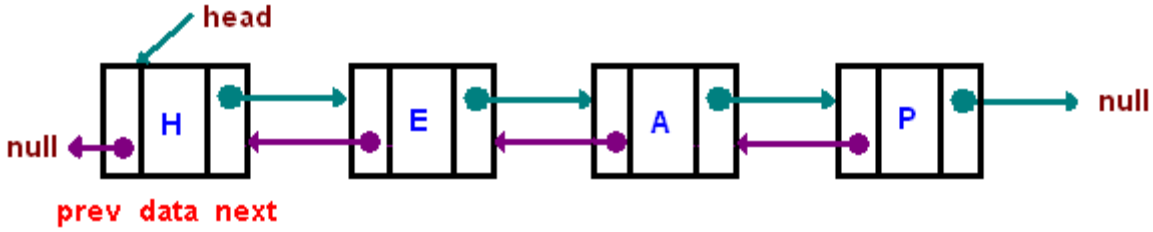


Each element (we will call it a node) of a list is comprising of two items - the data and a reference to the next node. The last node has a reference to null. The entry point into a linked list is called the head of the list. It should be noted that head is not a separate node, but the reference to the first node. If the list is empty then the head is a null reference. A linked list is a dynamic data structure. The number of nodes in a list is not fixed and can grow and shrink on demand. Any application which has to deal with an unknown number of objects will need to use a linked list.

5.1.1 TYPES OF LINKED LIST

A singly linked list is described above

A doubly linked list is a list that has two references, one to the next node and another to previous node.



Another important type of a linked list is called a circular linked list where last node of the list points back to the first node (or the head) of the list. But in our case we have used on the singly linked list.

In this project, we have focused on some of the basic operations on linked lists such as adding, deleting and updating node elements. In an array, adding or deleting an entry requires shifting of the array to “fill” or “create” holes.

Updating an entry in an array only requires accessing entry by its index and changing its value. One of the best features of the array data structure is its ability to access entries in a random access manner. That is, given the index of the entry we can easily find it.

Unlike arrays, the entry point into any linked list is the head of the list.

Head of the list is not a node but a reference to the first node in the list. In other words, head can be considered a lvalue. In an empty list the value of head is equal to null. We also know that any linked list ends with null pointer (unless it is a circular linked list where last node is connected to the head node).

We must take great care in manipulating linked lists since any misguided link in the middle will make the entire list inaccessible. This is because the only way to navigate a list is by using a reference to the next node from the current node. Often referred as "memory leaks", if a reference to a node is lost, then the entire list from that point on may not be accessible.

5.1.2 BASIC OPERATIONS

Some of the basic operations that can be performed on a linked list are

1. Traversing a linked list.
2. Appending a new node (to the end) of the list
3. Prepending a new node (to the beginning) of the list
4. Inserting a new node to a specific position on the list
5. Deleting a node from the list
6. Updating the data of a linked list node

5.1.3 TRAVERSING A LINKED LIST

A linked list is a linear data structure that needs to be traversed starting from the head node until the end of the list. Unlike arrays, where random access is possible, linked list requires access to its nodes through sequential traversal. Traversing a linked list is important in many applications.

For example, we may want to print a list or search for a specific node in the list. Or we may want to perform an advanced operation on the list as we traverse the list. The algorithm for traversing a list is fairly trivial.

1.Start with the head of the list. Access the content of the head node if it is not null.

2. Then go to the next node(if exists) and access the node information

3. Continue until no more nodes (that is, you have reached the last node)

5.2 FLOW CHART

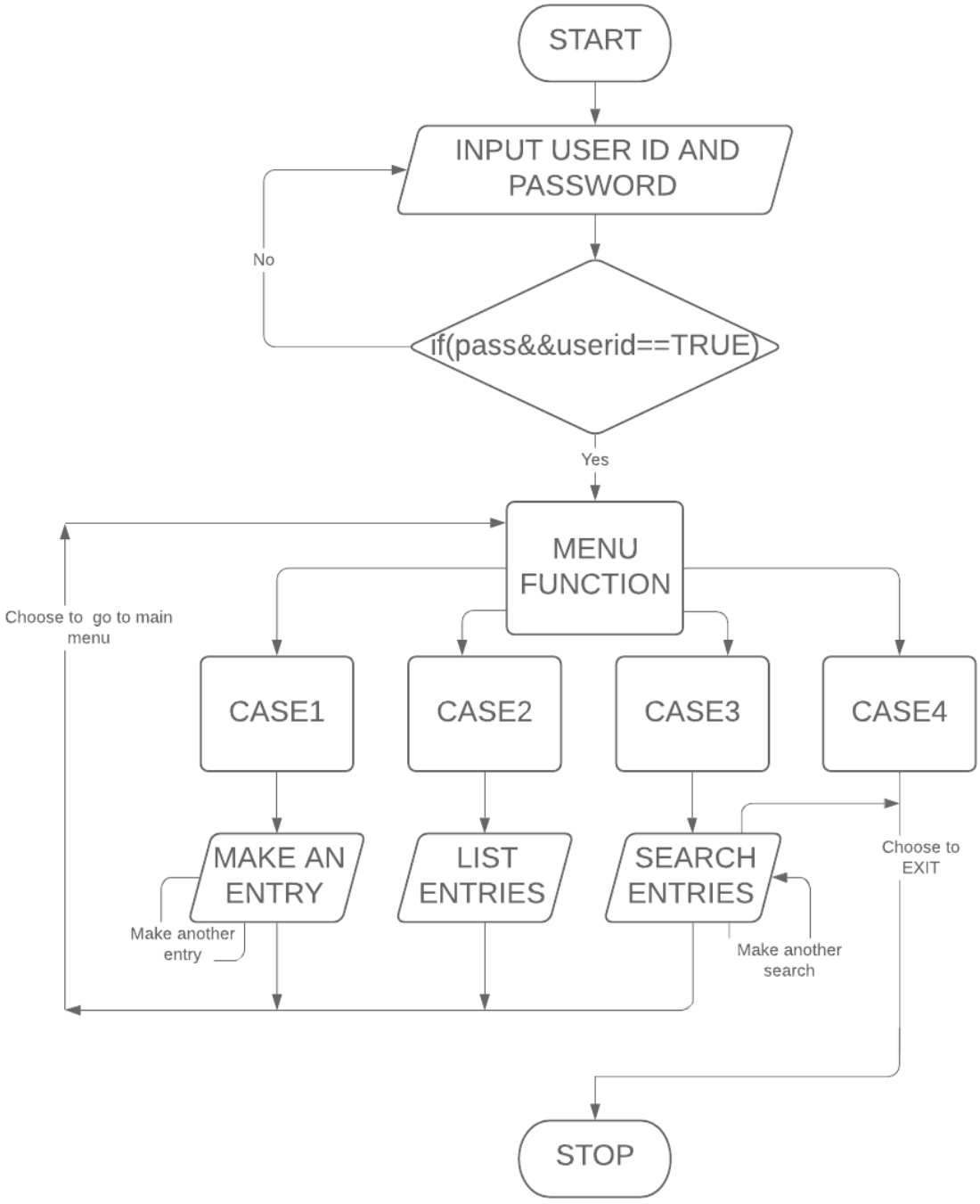


Fig 5.1 Flow Chart of Library Management System.

CHAPTER 6

IMPLEMENTATION

At first, the librarian enters the username and password that limits access into the library's database. The rest of the program executes only after the right username and password has been entered.

Code:

```
char username[15];
char password[12];
printf("\t\t\t\t\t\t\t\tEnter your username:");
scanf("%s",&username);
printf("\t\t\t\t\t\t\t\tEnter your password:");
scanf("%s",&password);
if(strcmp(username,"library1")==0)
{
   if(strcmp(password,"****")==0)
      {
            printf("\n\t\t\t\t\t\tWelcome.Login Success!");
            menu();
   }
      else
      {
            printf("\nWrong password");
      }
}
else
{
printf("\nUser doesn't exist");
}
```

After this if the login is successful, the main menu is displayed which has the option of creating, searching and listing the records.

```
int ch;
char tidi;
do
{
printf("CHOOSE THE OPERATION YOU WANT TO PERFORM\n\n");
printf("1.MAKE AN ENTRY\n");
printf("2.LIST ALL ENTRIES\n");
printf("3.SEARCH FOR AN ENTRY\n");
printf("4.EXIT\n")
printf("ENTER YOUR CHOICE:");
scanf("%d",&ch);
switch(ch)
{
        case 1:entry();break;
        case 2:display(); break;
        case 3:search(); break;
        case 4:exit(0);
}
printf("\n\nDO YOU WANT GO TO HOME PAGE AGAIN / EXIT?[Y/N]\n");
scanf(" %c",&tidi);
}while(tidi=='Y' || tidi=='y');
```

Then the entry function is called if option 1 is chosen , display function is called if option 2 is chosen and search function is called if option 3 is chosen. The pseudo code for these functions are shown below.

```
void entry()
{
        char tidi;
        do
        {
        char usn[11];int n;
        struct node *temp;
```

```
        temp=read();
        printf("\nEnter the branch from which it is picked\n");
        printf("1.COMPUTER SCIENCE\n");
        printf("2.ELECTRONICS AND COMMUNICATION\n");
        printf("3.MECHANICAL\n");
        printf("MAKE YOUR CHOICE:");
        scanf("%d",&n);
        switch(n)
        {
                case 1:cse();break;
                case 2:ece();break;
                case 3:me();break;
        }
        printf("\nDO YOU WANT TO MAKE ANOTHER ENTRY[Y/N]\n");
        scanf(" %c",&tidi);
        }while(tidi=='Y' || tidi=='y');
}
```

Again CSE, ME or ECE can be chosen here that will display the books of those particular branches.

```
void display()
{
        struct node *disp;
        disp=start;
        int i=0;
        printf("\n\n-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------\n");
        printf("%5s%13s%12s%75s%10s                                    Borrow_Date
Return_Date%14s%17s\n","Sl.No","USN","Phone","Name of Book","Branch","Total
Days","Late fee(INR)");
        printf("-------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------\n");
        if(disp==NULL)
```

```
            printf("\n\t\t\t\tThere are no entries!");
        else
        {
            while(disp!=NULL)
            {
                printf("%5d%13s%12s%75s%10s %02d/%02d/%d
%02d/%02d/%d%14d%17d",++i,disp->usn,disp->ph,disp->b,disp->br,disp->d1,disp-
>m1,disp->y1,disp->d2,disp->m2,disp->y2,disp->days1,disp->la);
                printf("\n--------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
                disp=disp->link;
            }
        }
}
```

The display function here is implemented here so that it can display all the entries that are present in the linked list along with the data. And here is one of the search functions.

```
char tidi;
        do
        {
        int ch;
        printf("\nWhat do you like to search by ?");
        printf("\n1.Customer USN");
        printf("\n2.Customer phone number");
        printf("\n3.No. of days for which the book was Borrowed");
        printf("\n4.Coustomers with delayed returns");
        printf("\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:srusn();break;
            case 2:srphn();break;
            case 3:srdays();break;
            case 4:srdel();break;
```

```
	}
	printf("\nDO YOU WANT TO MAKE ANOTHER SEARCH[Y/N]:");
	scanf(" %c",&tidi);
	}while(tidi=='Y' || tidi=='y');
```

So again based on the option that the user selects, different search operations can be performed.

Here is just the pseudo code for searching based on the USN.

```
void srusn()
{
	printf("\n\n\nEnter the USN you want to search by:");
	char sus[12];
	int lol=1,i=0;
	scanf("%s",&sus);
	struct node *srch;
	srch=start;
		printf("\n\n-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------\n");
		printf("%5s%13s%12s%75s%10s                                    Borrow_Date
Return_Date%14s%17s\n","Sl.No","USN","Phone","Name of Book","Branch","Total
Days","Late fee(INR)");
		printf("-------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------\n");
	if(start==NULL)
		printf("There are no Entries");
	else
		srch=start;
	while(srch!=NULL)
	{
		if(strcmp(srch->usn,sus)==0)
		{
```

```
                printf("%5d%13s%12s%75s%10s %02d/%02d/%d
%02d/%02d/%d%14d%17d",++i,srch->usn,srch->ph,srch->b,srch->br,srch->d1,srch-
>m1,srch->y1,srch->d2,srch->m2,srch->y2,srch->days1,srch->la);
                printf("\n--------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
        lol++;
        }
        srch=srch->link;
    }
    if(lol==1)
        printf("\n There are no entries matching entered data\n\n");
}
```

Here is just the pseudo code for searching based on the phone number.

```
void srphn()
{
    printf("\n\n\nEnter the phone number you want to search by:");
    char sph[11];
    int lol=1,i=0;
    scanf("%s",&sph);
    struct node *srch;
    srch=start;
        printf("\n\n-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
        printf("%5s%13s%12s%75s%10s Borrow_Date
Return_Date%14s%17s\n","Sl.No","USN","Phone","Name of Book","Branch","Total
Days","Late fee(INR)");
        printf("-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
    if(start==NULL)
        printf("There no Entries");
    else
        srch=start;
    while(srch!=NULL)
```

```
	{
		if(strcmp(srch->ph,sph)==0)
		{
			printf("%5d%13s%12s%75s%10s %02d/%02d/%d
%02d/%02d/%d%14d%17d",++i,srch->usn,srch->ph,srch->b,srch->br,srch->d1,srch-
>m1,srch->y1,srch->d2,srch->m2,srch->y2,srch->days1,srch->la);
			printf("\n--------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
		lol++;
		}
		srch=srch->link;
	}
	if(lol=1)
		printf("\n There are no entries matching entered data\n\n");
}
```

Here is just the pseudo code for searching based on the number of days.

```
void srdays()
{
	printf("\n\n\nEnter the number of days for which the book was borrowed:");
	int dis;
	int lol=1,i=0;
	scanf("%d",&dis);
	struct node *srch;
	srch=start;
		printf("\n\n-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
		printf("%5s%13s%12s%75s%10s Borrow_Date
Return_Date%14s%17s\n","Sl.No","USN","Phone","Name of Book","Branch","Total
Days","Late fee(INR)");
		printf("-----------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
	if(start==NULL)
		printf("There are no Entries!");
```

```
	else
		srch=start;
	while(srch!=NULL)
	{
		if(srch->days1==dis)
		{
			printf("%5d%13s%12s%75s%10s %02d/%02d/%d
%02d/%02d/%d%14d%17d",++i,srch->usn,srch->ph,srch->b,srch->br,srch->d1,srch-
>m1,srch->y1,srch->d2,srch->m2,srch->y2,srch->days1,srch->la);
			printf("\n--------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------\n");
		lol++;
		}
		srch=srch->link;
	}
	if(lol==1)
		printf("\n There are no entries matching entered data\n\n");
}
```

Here is the pseudo code for searching based on delayed returns.

```
void srdel()
{
	int lol=1,i=0;
	struct node *srch;
	srch=start;
		printf("\n\n-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------\n");
		printf("%5s%13s%12s%75s%10s Borrow_Date
Return_Date%14s%17s\n","Sl.No","USN","Phone","Name of Book","Branch","Total
Days","Late fee(INR)");
		printf("-----------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------\n");
	if(start==NULL)
		printf("There are no Entries");
```

```
        else
                srch=start;
        while(srch!=NULL)
        {
                if(srch->days1>15)
                {
                        printf("%5d%13s%12s%75s%10s                %02d/%02d/%d
%02d/%02d/%d%14d%17d",++i,srch->usn,srch->ph,srch->b,srch->br,srch->d1,srch-
>m1,srch->y1,srch->d2,srch->m2,srch->y2,srch->days1,srch->la);
                        printf("\n--------------------------------------------------------------------------
-------------------------------------------------------------------------------------------\n");
                lol++;
                }
                srch=srch->link;
        }
        if(lol==1)
                printf("\n There are no entries matching entered data\n\n");
}
```

And also one of the most difficult part of this code was to able to calculate the number of days between 2 given dates especially with the complication of the month of February.

```
int isLeapYear(int year)
{
    int flag = 0;
    if (year % 100 == 0) {
        if (year % 400 == 0) {
            flag = 1;
        } else {
            flag = 0;
        }
    } else if (year % 4 == 0) {
        flag = 1;
```

```
    }
    return flag;
}
int calcdaysandprice()
{
    int year, month, date, i, days;
    int cyear, cmonth, cdate, tmpMon, tmpYear;
    char str[100];
       temp->days1=0;
    printf("\nENTER THE DATE THE BOOK WAS BORROWED(DD/MM/YYYY):");
    scanf("%d/%d/%d", &date, &month, &year);
    printf("\nENTER THE DATE THE BOOK WAS RETURNED(DD/MM/YYYY):");
    scanf("%d/%d/%d", &cdate, &cmonth, &cyear);
    tmpMon = month;
    tmpYear = year;
       temp->d1=date;
       temp->m1=month;
       temp->y1=year;
       temp->d2=cdate;
       temp->m2=cmonth;
       temp->y2=cyear;
    if (cyear == year && cmonth == month) {
          temp->days1 = cdate - date + 1;
          return 0;
    }
    if (month == 2) {
          temp->days1 = daysInMonth[month - 1] - date + 1 + isLeapYear(year);
    } else {
          temp->days1 = daysInMonth[month - 1] - date + 1;
    }
    if (month == 12) {
          month = 0;
          year++;
```

```
    }
    while (year <= cyear) {
        for (i = month; i < 12; i++) {
            if ((year == cyear) && (i == cmonth - 1)) {
                temp->days1 = temp->days1 + cdate;
                break;
            }
            if (i == 1) {
                temp->days1 = temp->days1 + daysInMonth[i] + isLeapYear(year);
            } else {
                temp->days1 = temp->days1 + daysInMonth[i];
            }
        }
        month = 0;
        year++;
    }
    int late=0;
      days=temp->days1;
      if(days>15)
      {
            days=days-15;
            late=days*2;
      }
      return late;
}
```

This function accepts the date in a particular format and then stores the number of days for which the book was borrowed and also calculates its late fees if it exceeds 15 days at the rate 2 rupees per day.

So since we have used linked list in our project, we had to create a user defined function or a structure along with a pointer variable *start that holds the address of all these user defined structure members. The user defined function used in this project are struct node along with pointer variable *start and struct node *read() which reads the USN and phone number of the customer.

Here is the code snippet for the linked list that we have used in the code.

```
struct node
{
        char usn[12];
        char b[50],a[50],ph[11],br[5];
        int d1,d2,m1,m2,y1,y2;
        int days1,la;
        struct node *link;
}*start;
struct node *temp,*p;
struct node *read()
{
        char us[12],phn[11];
        printf("Enter USN\n");
        scanf("%s",&us);
        printf("Enter Phone number\n");
        scanf("%s",&phn);
        temp=(struct node*)malloc(sizeof(struct node));
        strcpy(temp->usn,us);
        strcpy(temp->ph,phn);
        temp->link=NULL;
        p=temp;
        return p;
}
```

This read() function is called while reading the borrower's data in the entry() function as shown in the pseudo code.

CHAPTER 7

RESULTS

The results here show what has been done and the outcome of this project along with the screen shots of all the possible outcomes of our code.

Fig 7.1 First we start of with the main function where the librarian has to make the login by entering the correct login details.

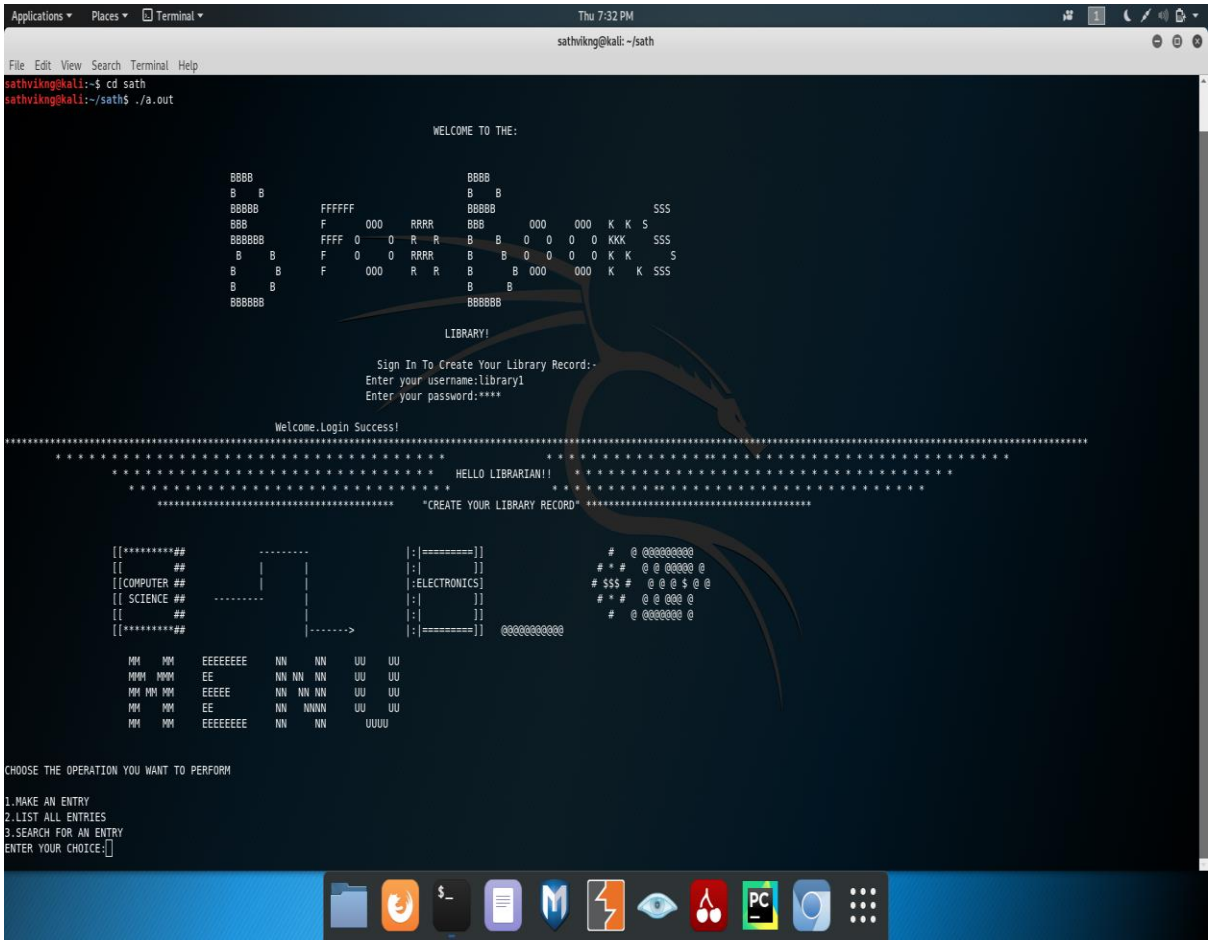


Fig 7.1 Login page with successful Login.

So once the login is successful, the login is completed, and the menu will be displayed (as shown in Fig 7.2). If the login is unsuccessful then the following message will be displayed.

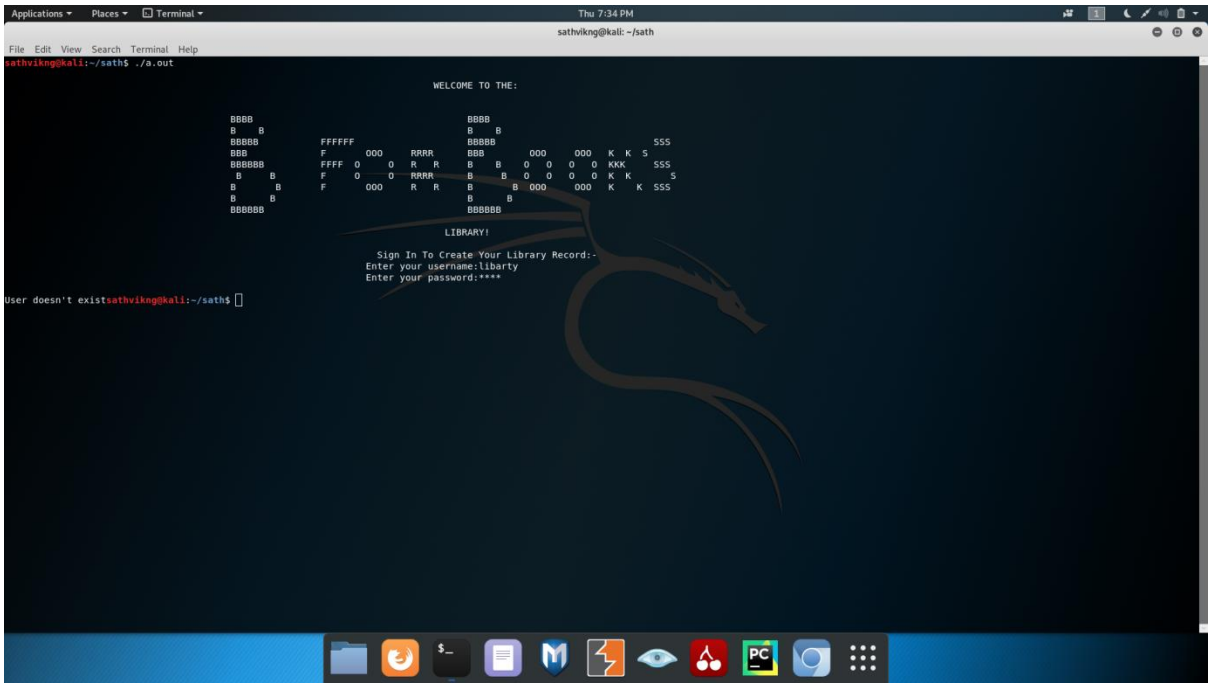


Fig 7.2 Login page with Unsuccessful Login.

Here it shows that the user does not exist if the entered User ID is wrong and if the User ID is correct, then the message will be displayed as invalid password entered.

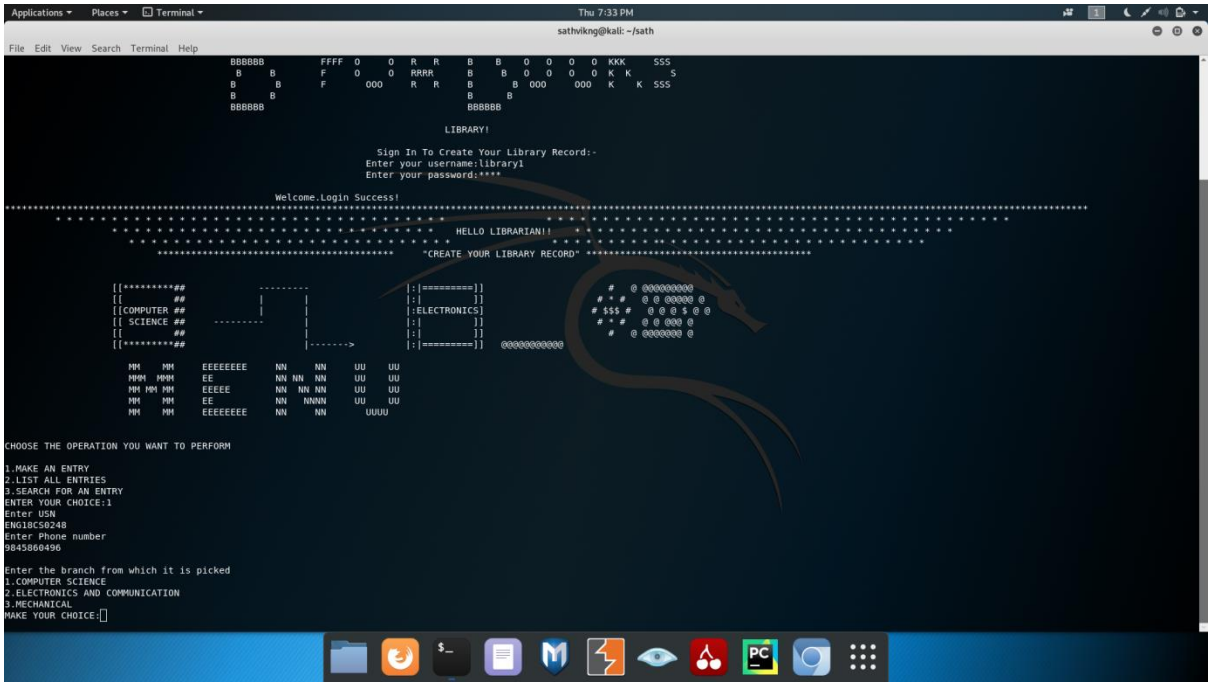


Fig 7.3 Choosing from the options in the menu.

Fig 7.3 shows that once you choose option 1 i.e to make an entry, then the librarian chooses the branch from which the book is taken.

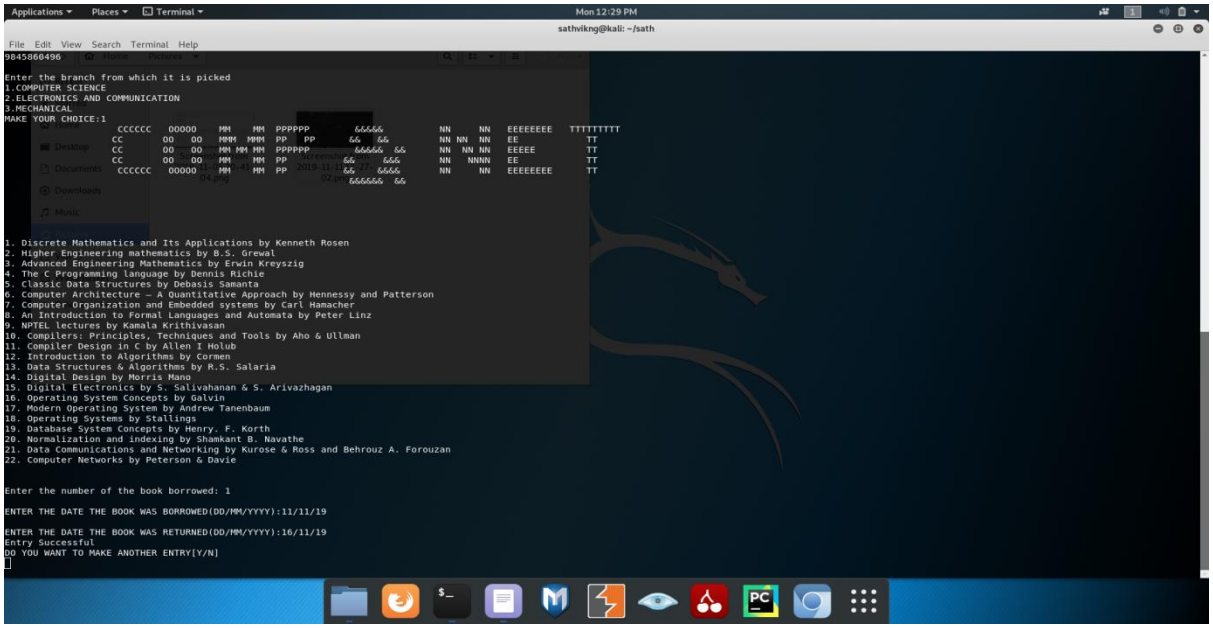


Fig 7.4 Choosing a book with CSE as the chosen option.

Fig 7.4 shows, once you have chosen computer science as the option then it shows the list of gate books that is available in the library under the department, so would be the case for the other branches as well. Then you need to enter the date of borrowing and return. When you choose to make another entry then what happens is shown in Fig 7.5.

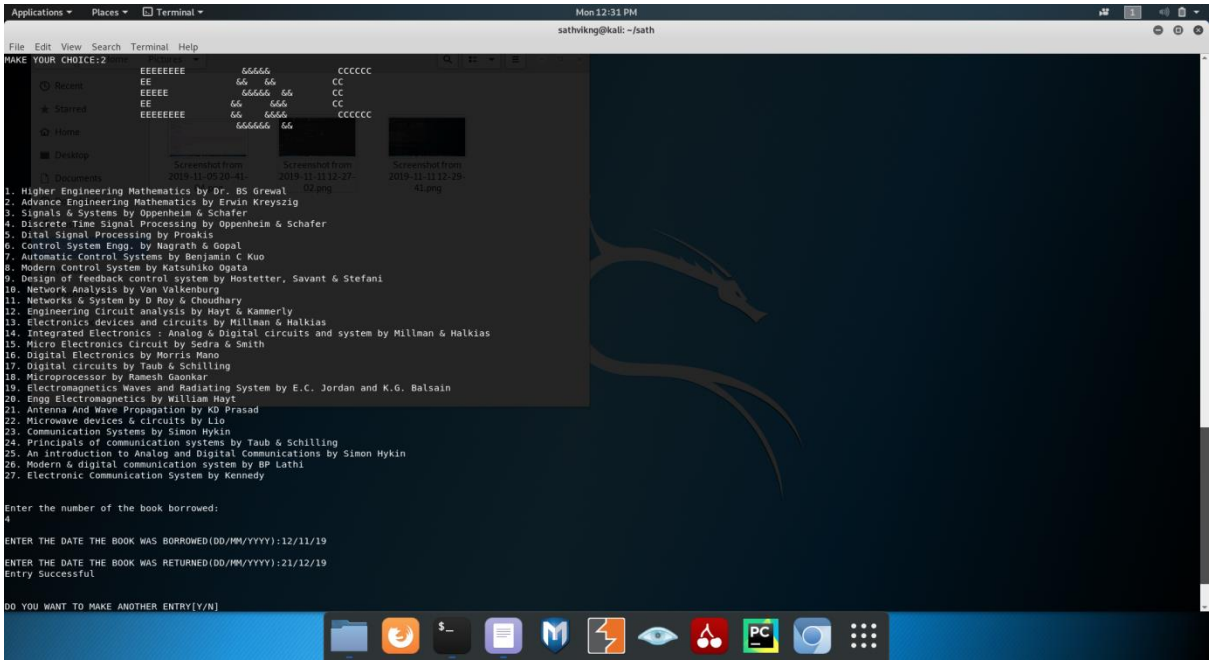


Fig 7.5 Choosing a book with ECE as the chosen option.

Fig 7.5 shows what is going to happen when you choose to make another entry and you choose EC as the option you want to pick the book from.

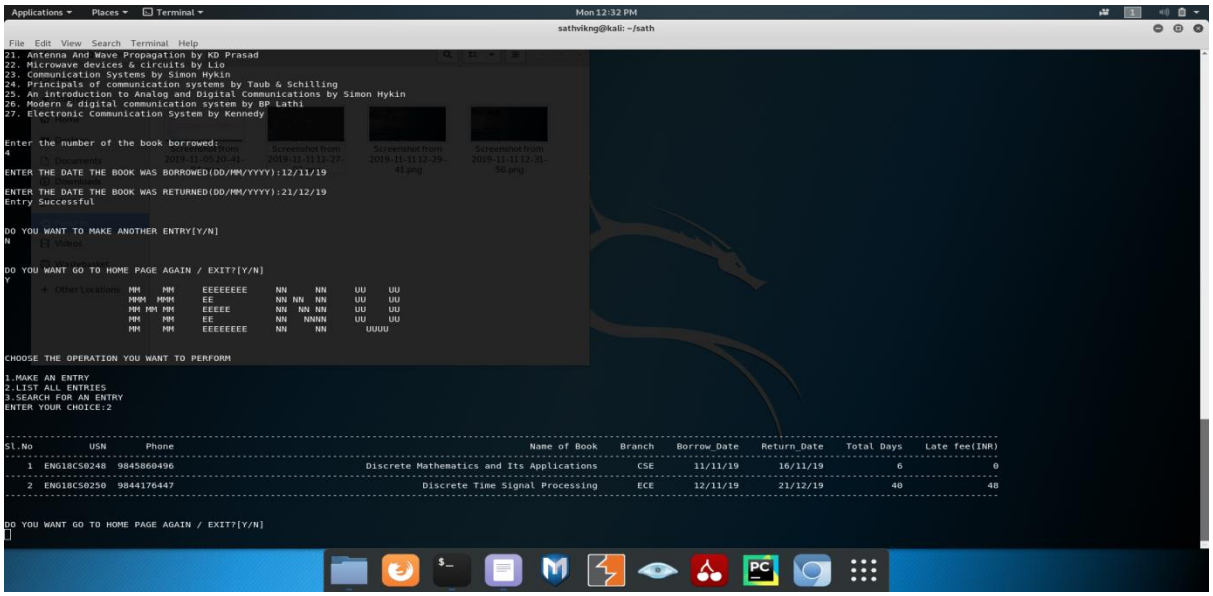


Fig 7.6 Listing all the entries.

Fig 7.6 shows that when you choose to go back to the main menu again it again goes back to the main menu and then once you choose to List all the entries that you have made, it displays all the entries that have been made with the calculation of the late fees and also the number of days for which the book was borrowed and then you can go back to the main menu after that as shown in Fig 7.7.

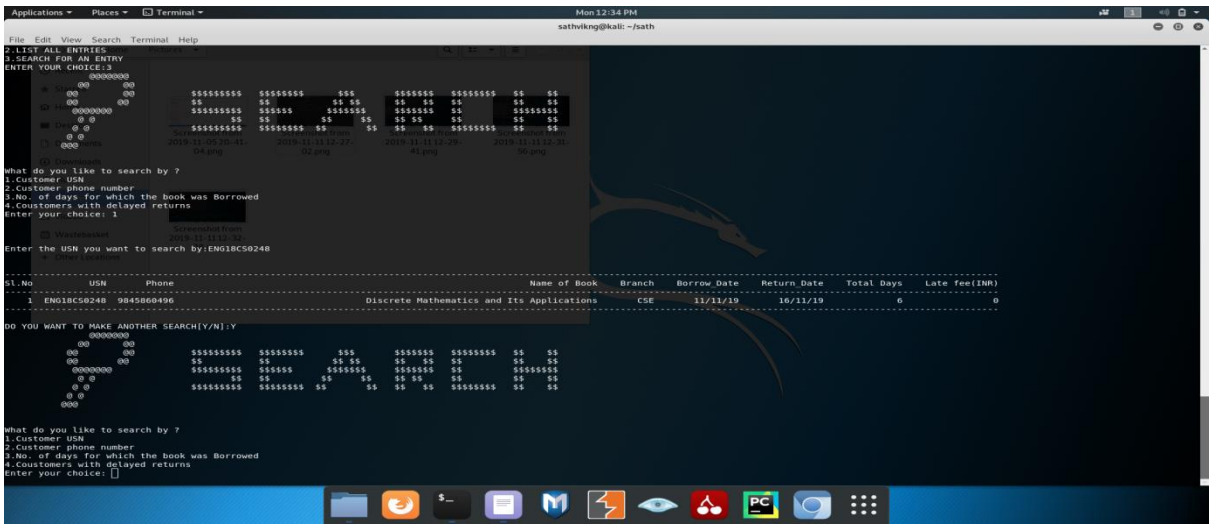


Fig 7.7 Search Result for searching based on USN.

Fig 7.7 shows that once you select to go to the search option, you will be displayed with four ways in which the librarian can search for an entry and once you have chosen to search by USN you can enter the USN you would want to search by. Then you can choose to Search again or return to the main menu.

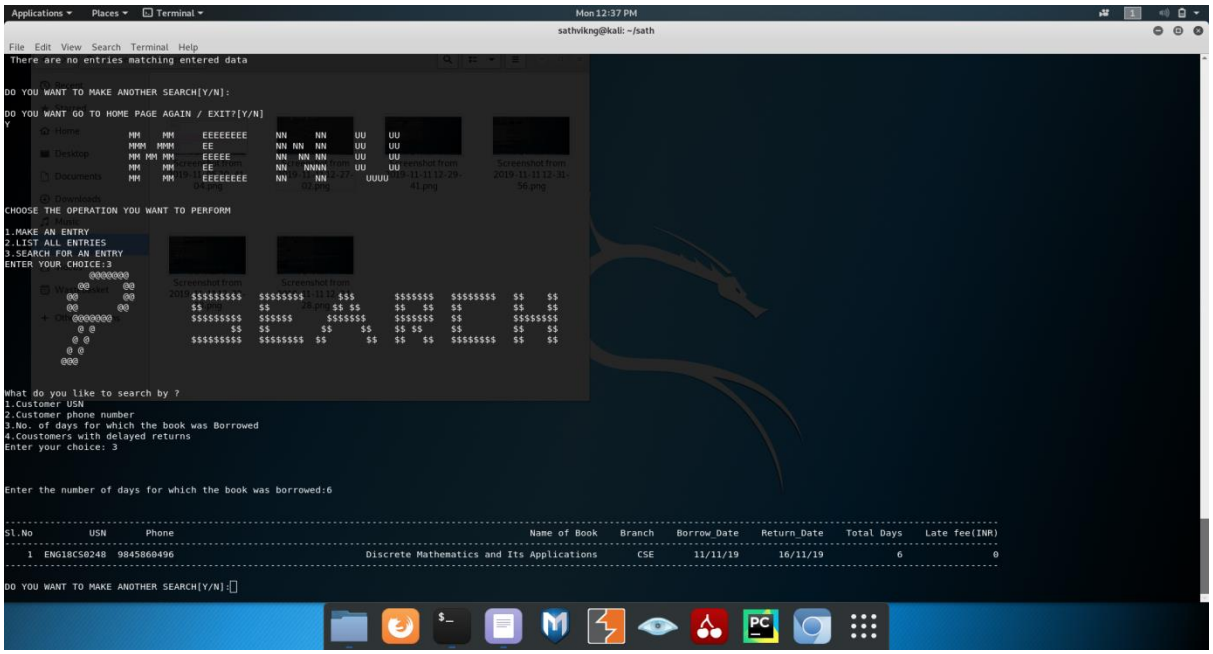


Fig 7.8 Search Result for searching based on the days.

Fig 7.8 shows that once you have chosen to search based on the number of days for which the book was borrowed after choosing to search again, the librarian just needs to enter the number of days for which the book was borrowed and you can directly get the result for that.

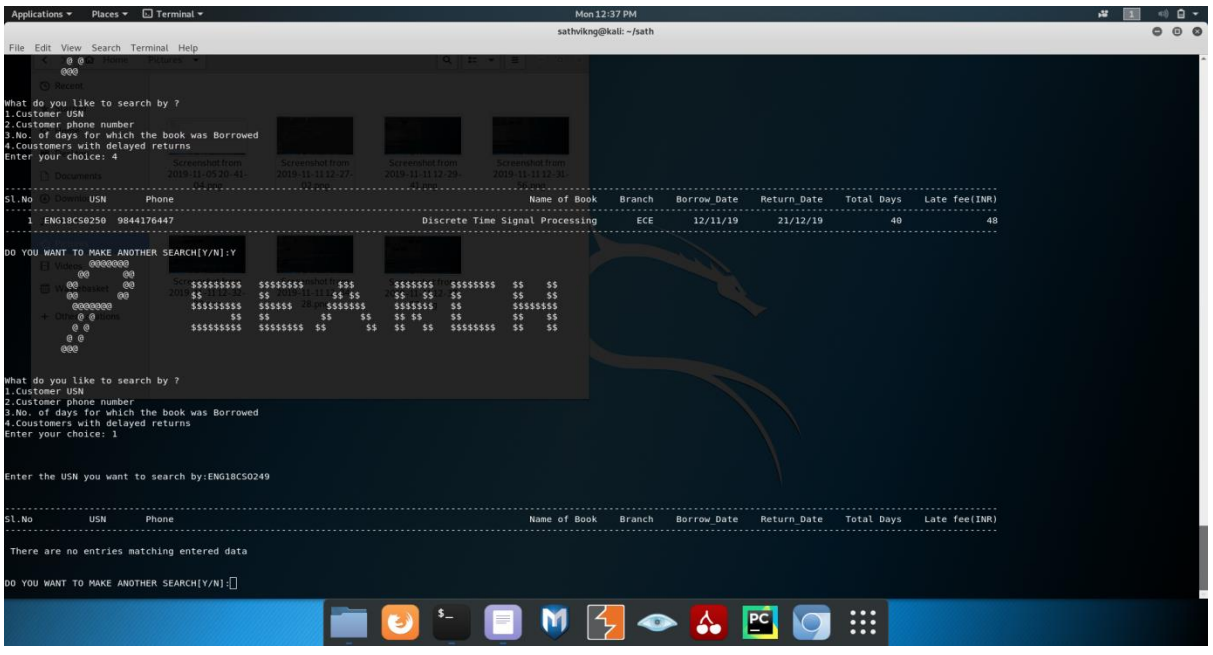


Fig 7.9 Choosing to enter or quit again.

After entering the data, if the program does not find any search results for the entered data, then the code would return that there are no entries matching the entered data.

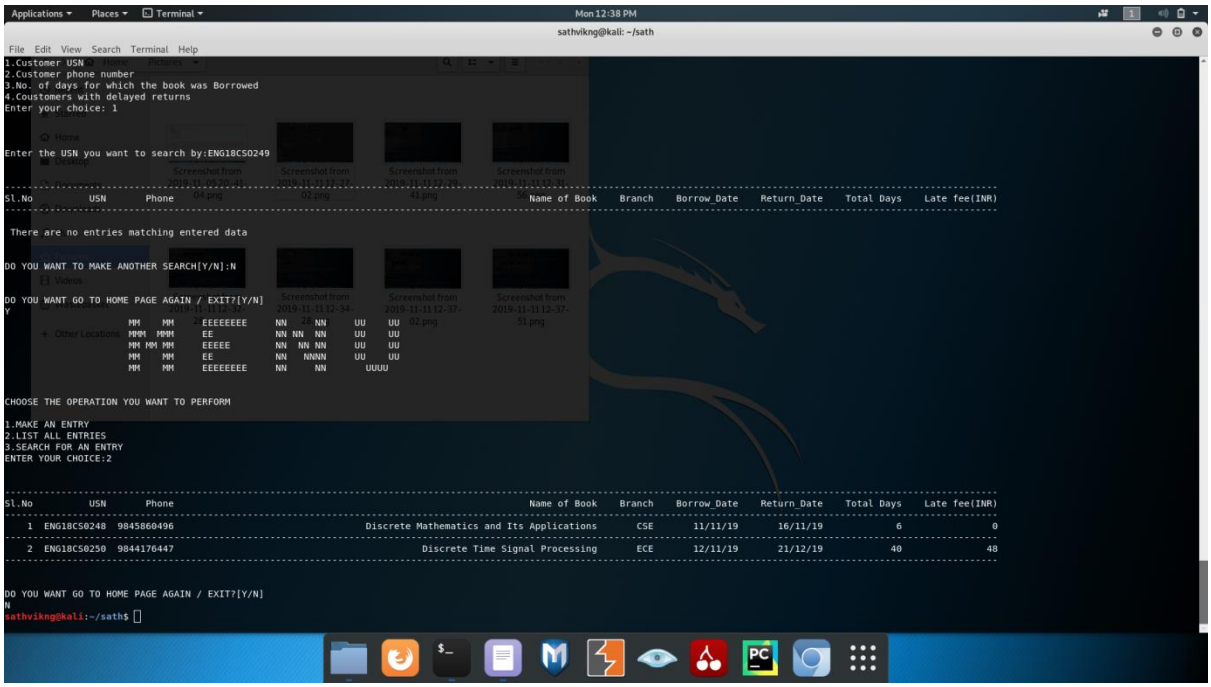


Fig 7.10 Showing unsuccessful search results.

Fig 7.10 is the final figure which shows that once you are done with all the operations you want to perform, you can just press N to quit the code execution and all the entries that were supposed to be made have been made successfully and the code has been executed and met the librarian requirements.

CHAPTER 8

CONCLUSION

8.1 ACCOMPLISHMENTS

We have successfully completed the project error free based on the requirements of the daily usage of a librarian. So with regard to the linked list, this is what we have accomplished.

In this project we learned how to traverse, append, prepend, insert and delete nodes from a linked list. These operations are quite useful in many practical applications. Suppose you are the programmer responsible for an application like Microsoft power point. We can think of a power point presentation as a list, whose nodes are the individual slides.

In the process of creating and managing the entries we used the concepts such as inserting and deleting nodes learned in this project. Many other applications may require you to think of an abstraction like a list, where list can be easily maintained. In this project we have learnt more advanced list operations such as reversing a list, swapping nodes and sorting a list etc.

8.2 WHAT WE WANT TO DO DIFFERENTLY AND FUTURE WORK

We here could use file functions so that we could use the output for once the console has been restarted again, or we would want to link the entered data to a cloud, where we can access data from anywhere and however we want.

We would also like to make this software to be able to create as many accounts possible and then link those accounts to the cloud, we just like to make it for multiple users rather than just the librarian.

We also like the teachers or the librarians to put up some fresh news or even put up some notices on the public page (such as workshops or seminars being held in the institution) of the library login so that it is visible to the non-users as well.

There is a future scope of this facility, that many more features such as online lectures video tutorials can be added by teachers as well as online assignments submission facility, a feature of group chat where students can discuss various issues of engineering and gate problems can be added to this project thus making it more interactive more user friendly and project which fulfils every users requirements in the best way possible.

REFERENCES

1. https://www.cs.cmu.edu/~adamchik/15-121/lectures/Linked%20Lists/linked%20lists.html
2. https://stackoverflow.com/questions/982388/how-to-implement-a-linked-list-in-c
3. https://www.geeksforgeeks.org/data-structures/linked-list/
4. https://www.hackerrank.com/challenges/insert-a-node-at-the-tail-of-a-linked-list/problem
5. https://guide.freecodecamp.org/computer-science/data-structures/linked-lists/
6. https://www.lucidchart.com/documents/edit/3ca1f8fb-2d4d-43e4-8abf-4b101a75b50a/m-5o7ONTd-nK
7. https://www.geeksforgeeks.org/time-function-in-c/
8. http://see-programming.blogspot.com/2013/06/c-program-to-find-number-of-days.html