LIBRARY MANAGEMENT SYSTEM

PROJECT REPORT

18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY

(2018 Regulation)

II Year/ III Semester

Academic Year: 2022 -2023

By

SHREYA RANJAN (RA2111003010301)

ARZOB SEN (RA2111003010303)

SAPTAPARNO PATRA (RA2111003010311)

Under the guidance of

Ms. HEMA M

Assistant Professor

Department of Computing Technologies

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Kattankulathur, Kancheepuram

NOVEMBER 2022

BONAFIDE

This is to certify that **18CSC202J - OBJECT ORIENTED DESIGN AND PROGRAMMING LABORATORY project report** titled **“LIBRARY MANAGEMENT SYSTEM”** is the bonafide work of **SHREYA RANJAN (RA2111003010301) , ARZOB SEN (RA211003010311) , SAPTAPARNO PATRA (RA2111003010311)** who undertook the task of completing the project within the allotted time.

Signature of the Guide	**Signature of the II Year Academic Advisor**
Dr. Hema M.	------------------------
Assistant Professor	**Professor and Head**
Department of CINTEL,	Department of CINTEL
SRM Institute of Science and Technology	SRM Institute of Science and Technology

About the course -

18CSC202J- Object Oriented Design and Programming is a 4 credit courses with **L T P C as 3-0-2-4** (Tutorial modified as Practical from 2018 Curriculum onwards)

Objectives -

The student should be made to:

- Learn the basics of OOP concepts in C++.
- Learn the basics of OOP analysis and design skills.
- Be exposed to the UML design diagrams.
- Be familiar with the various testing techniques.

Course Learning Rationale (CLR): The purpose of learning this course is to -

1.Utilize class and build domain models for real-time programs.

2.Utilize method overloading and operator overloading for real-time application development program

3.Utilize inline, friend and virtual functions and create application development programs.

4.Utilize exceptional handling and collections for real-time object-oriented programming applications.

5.Construct UML component diagram and deployment diagram for design of applications

6.Create programs using object-oriented approach and design methodologies for real-time application development.

Course Learning Outcomes (CLO): At the end of this course, learners will be able to -

1.Identify the class and build a domain model.

2.Construct programs using method overloading and operator overloading.

3. Create programs using inline, friend and virtual functions, construct programs using standard templates.

4.Construct programs using exceptional handling and collections.

5.Create UML component diagram and deployment diagram.

6.Create programs using object oriented approach and design methodologies.

COURSE ASSESSMENT PLAN FOR OODP LAB

S.No	List of Experiments	Course Learning Outcomes (CLO)	Blooms Level	PI	No of Programs in each session
1.	Implementation of I/O Operations in C++	CLO-1	Understand	2.8.1	10
2.	Implementation of Classes and Objects in C++	CLO-1	Apply	2.6.1	10
3,	To develop a problem statement. 1. From the problem statement, Identify Use Cases and develop the Use Case model. 2. From the problem statement, Identify the conceptual classes and develop a domain model with a UML Class diagram.	CLO-1	Analysis	4.6.1	Mini Project Given
4.	Implementation of Constructor Overloading and Method Overloading in C++	CLO-2	Apply	2.6.1	10
5.	Implementation of Operator Overloading in C++	CLO-2	Apply	2.6.1	10
6.	Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams and Collaboration diagrams	CLO-2	Analysis	4.6.1	Mini Project Given
7.	Implementation of Inheritance concepts in C++	CLO-3	Apply	2.6.1	10
8.	Implementation of Virtual function & interface concepts in C++	CLO-3	Apply	2.6.1	10
9.	Using the identified scenarios in your project, draw relevant state charts and activity diagrams.	CLO-3	Analysis	4.6.1	Mini Project Given
10.	Implementation of Templates in C++	CLO-3	Apply	2.6.1	10
11.	Implementation of Exception of Handling in C++	CLO-4	Apply	2.6.1	10
12.	Identify the User Interface, Domain objects, and Technical Services. Draw the partial layered, logical architecture diagram with UML package diagram notation such as Component Diagram, Deployment Diagram.	CLO-5	Analysis	4.6.1	Mini Project Given
13.	Implementation of STL Containers in C++	CLO-6	Apply	2.6.1	10
14.	Implementation of STL associate containers and algorithms in C++	CLO-6	Apply	2.6.1	10
15.	Implementation of Streams and File Handling in C++	CLO-6	Apply	2.6.1	10

LIST OF EXPERIMENTS FOR UML DESIGN AND MODELING -

To develop a mini-project by following the exercises listed below -

1. To develop a problem statement.
2. Identify Use Cases and develop the Use Case model.
3. Identify the conceptual classes and develop a domain model with UML Class diagrams.
4. Using the identified scenarios, find the interaction between objects and represent them using UML Sequence diagrams.
5. Draw relevant statecharts and activity diagrams.
6. Identify the User Interface, Domain objects, and Technical services. Draw the partial layered, logical architecture diagram with UML package diagram notation.

Suggested Software Tools -

StarUML, visual studio code (c++), eclipse.

ABSTRACT

With the advancement of technology, it is imperative to exalt all the systems in a user-friendly manner. The Library Management system (LMS) acts as a tool to transform traditional libraries into digital libraries. In traditional libraries, the students/user has to search for books which is a hassle process and there is no proper maintenance of databases about issues/fines. The overall progress of work is slow and it is impossible to generate a fast report. The librarians have the allotted work for arranging, sorting books in the book sales. At the same time, they have to check and monitor all the lent/borrowed book details with its fine. It is a tedious process to work simultaneously in different sectors. LMS will assist the librarians to work easily. The LMS supports the librarians to encounter all the issues concurrently. The users need not stand in a queue for a long period to return/borrow a book from the library. The single PC contains all the data in it. The librarians have to access the system and provide an entry in it. Through LMS, the librarian can find the book in the bookshelves. The LMS is designed with the basic features such as a librarian can add/view/update/delete books and students' details in it. Once he/she ingress into the system, they can modify any data in the database. The authorized person can only access the LMS system, they have to log in with their user id and password. As mentioned, the LMS is designed in a user-friendly manner, so the admin can smoothly activate the system without expert advice. Every data is stored and retrieved from the SQL database so it is highly secure. Thus our system contributes its new approach towards the digital library setup.

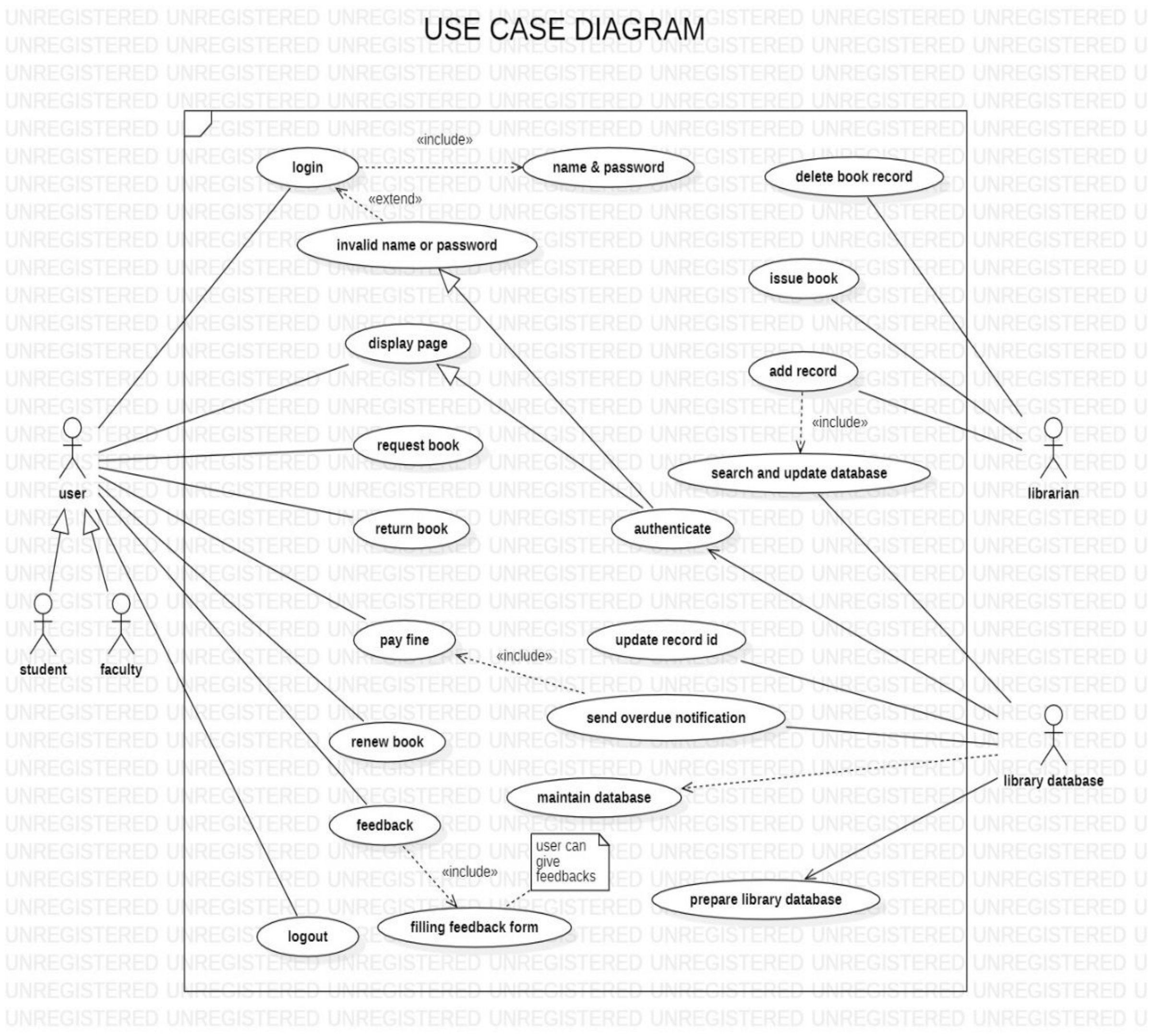
MODULE DESCRIPTION

Following are the modular descriptions involved in the library management system -

- **Student registration module.**
- **Search book, courses module.**
- **Fully automated features using masters.**
- **Complete book details category - wise.**
- **Simple automated book issue and receive system with least manual entries.**
- **Dynamic reports of available and issued books.**
- **Cursor implementation and File modules are used.**
- **Fine calculation system for delay of book.**
- **Dynamic subscription management.**
- **Easily accessible inventory management.**
- **Librarian’s easy transaction management.**
- **Book availability checking details.**
- **Log In/Out - password protected module.**

UML DIAGRAMS

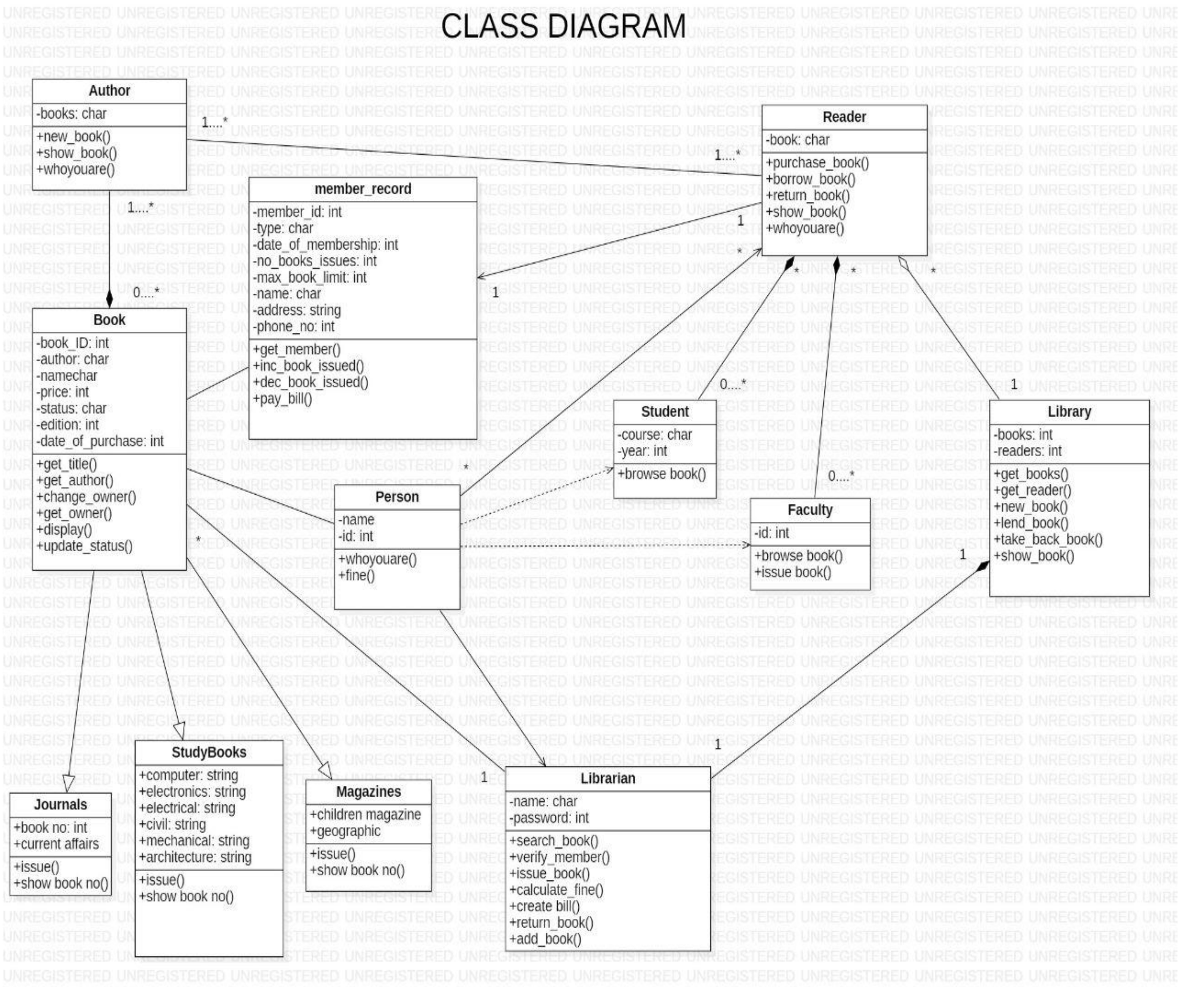
Use case diagram



Here, the above diagram shows the designing of a use case diagram for the library management system. Some scenarios of the system are as follows -

1. **User who registers himself as a new user initially is regarded as staff or student for the library system.**
   - **For the user to get registered as a new user, registration forms are available that are needed to be fulfilled by the user.**
   - **After registration, a library card is issued to the user by the librarian. On the library card, an ID is assigned to the cardholder or user.**
2. **After getting the library card, a new book is requested by the user as per their requirement.**
3. **After requesting, the desired book or the requested book is reserved by the user that means no other user can request for that book.**
4. **Now, the user can renew a book that means the user can get a new due date for the desired book if the user has renewed them.**
5. **If the user somehow forgets to return the book before the due date, then the user pays fine. Or if the user forgets to renew the book till the due date, then the book will be overdue and the user pays fine.**
6. **Users can fill the feedback form available if they want to.**
7. **Librarians have a key role in this system. Librarian adds the records in the library database about each student or user every time issuing the book or returning the book, or paying a fine.**
8. **Librarian also deletes the record of a particular student if the student leaves the college or passed out from the college. If the book no longer exists in the library, then the record of the particular book is also deleted.**
9. **Updating databases is the important role of Librarian.**

Class diagram



A Library Management System Class Diagram is a form of structural (UML) diagram that depicts the structure of a system. This is designed by displaying the system's classes, attributes, methods, and relationships between classes.

Class diagrams reveal the class structure blueprint of the Library Management System. It is used to model the items that make up the system and depict their relationships. This is to define the function of an object and the operation it provides. A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

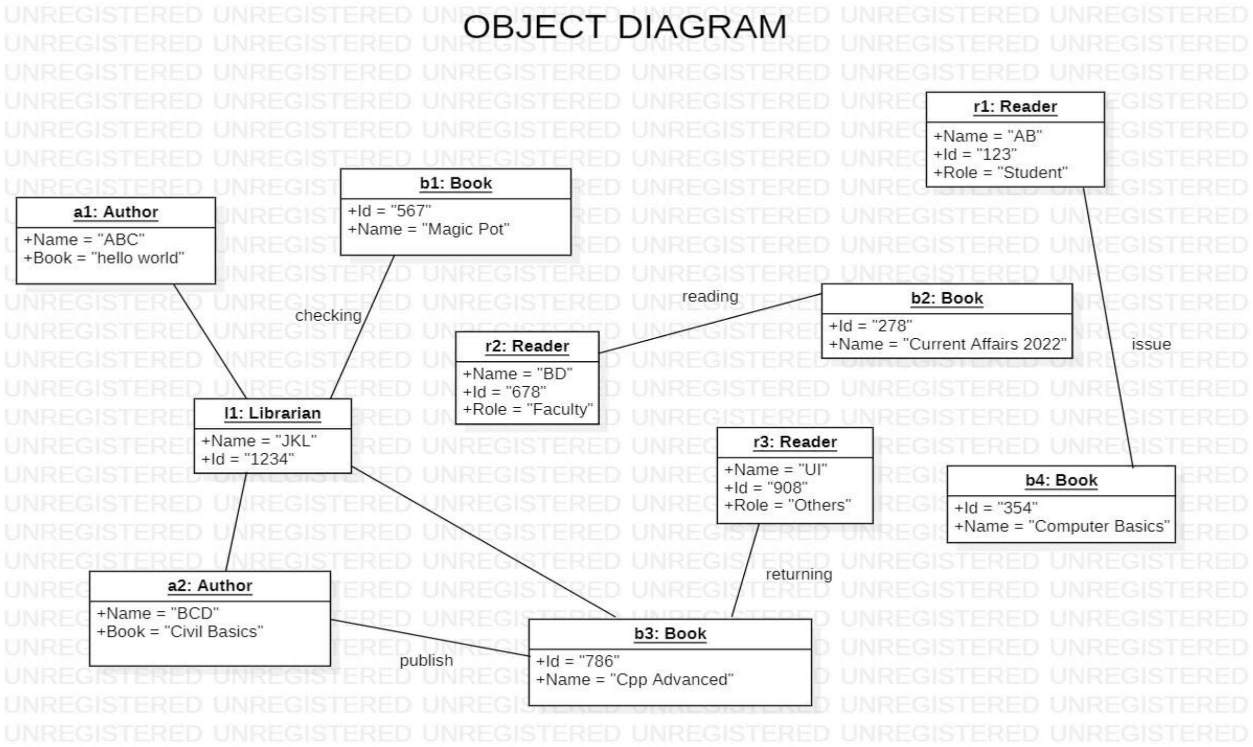
The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling, translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

- **The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.**
- **The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.**
- **The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.**

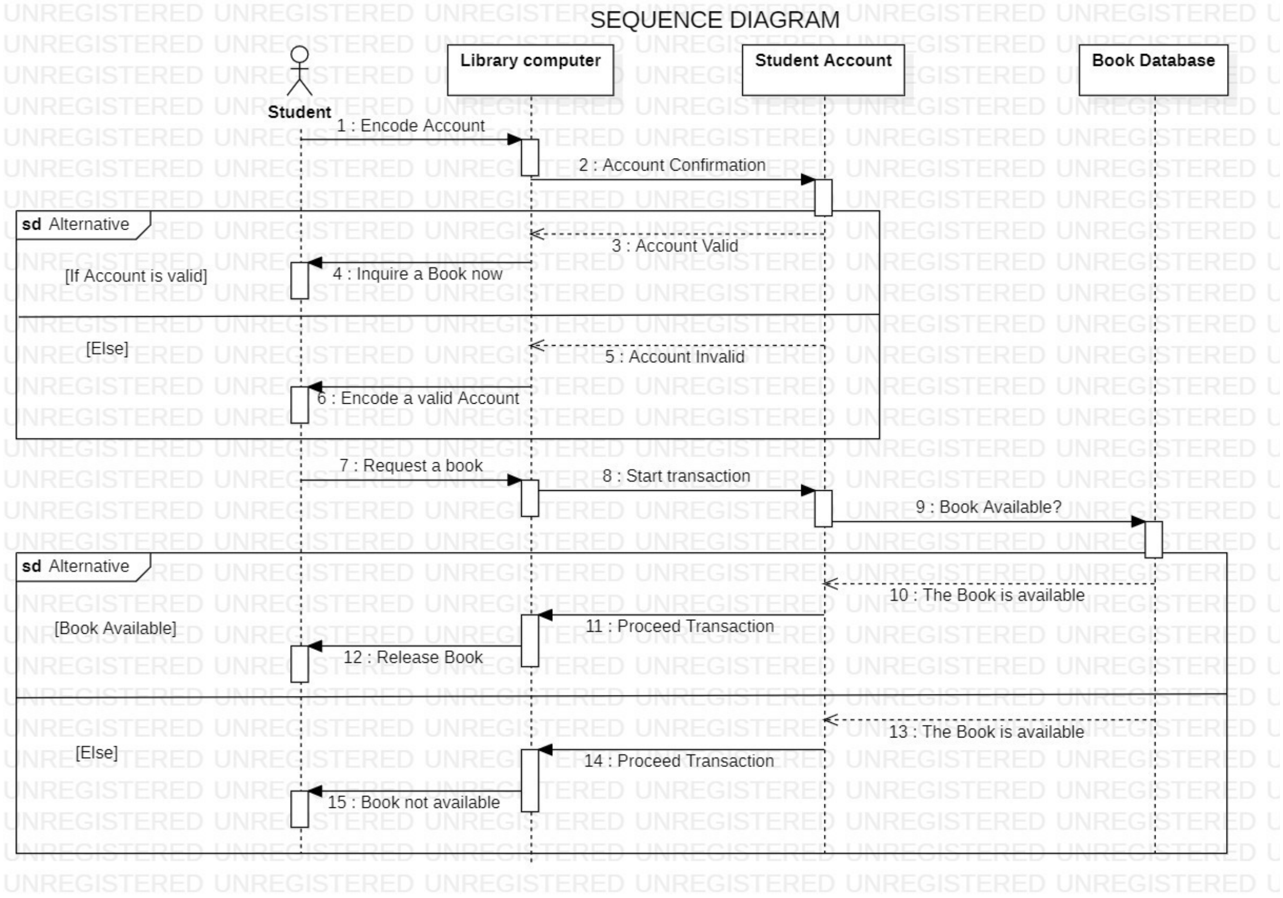
In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between them. In detailed modeling, the classes of the conceptual design are often split into subclasses.

Object diagram



In object-oriented programming, an object diagram in the Unified Modeling Language (UML) is a diagram that shows a complete or partial view of the structure of a modeled system at a specific time.In the Unified Modeling Language (UML), an object diagram focuses on some particular set of objects and attributes, and the links between these instances. A correlated set of object diagrams provides insight into how an arbitrary view of a system is expected to evolve over time. Early UML specifications described object diagrams as such:"An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, namely to show examples of data structure.The latest UML 2.5 specification does not explicitly define object diagrams, but provides a notation for instances of classifiers.Object diagrams and class diagrams are closely related and use almost identical notation.Both diagrams are meant to visualize static structure of a system. While class diagrams show classes, object diagrams display instances of classes (objects). Object diagrams are more concrete than class diagrams. They are often used to provide examples or act as test cases for class diagrams. Only aspects of current interest in a model are typically shown on an object diagram.

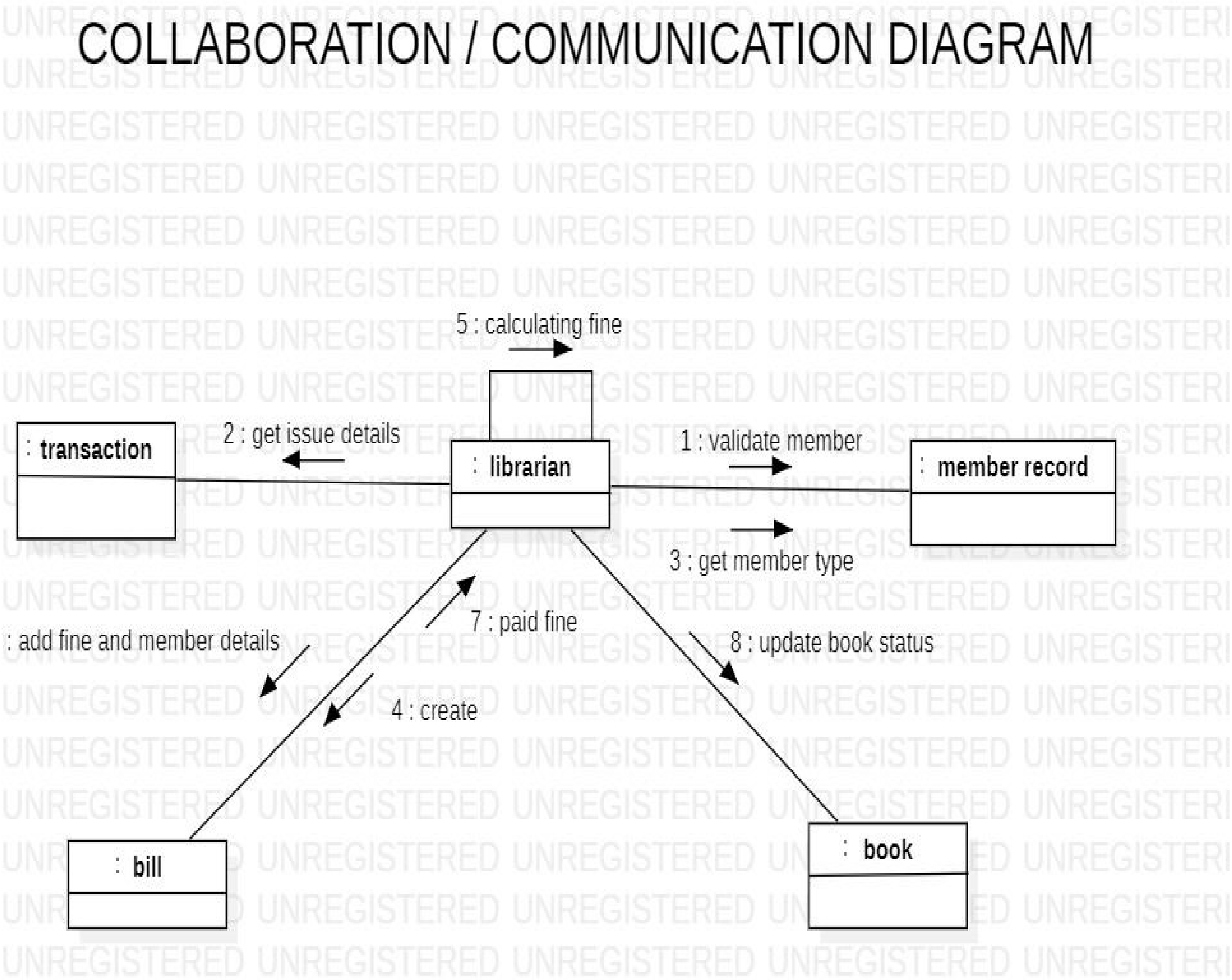
Sequence diagram



The Sequence Diagram for Library Management System represents the scenario and the messages that must be passed between objects. This is done in order for the scenario's functionality to be realized. It's an interaction diagram that shows how activities are carried out, including when and how messages are sent.

The sequence diagram given shows 4 objects which are: the student, student account, Book and book database. These objects were based on practical activities that happen in library management. The sequence of messages then was plotted below the objects to determine how the process is being performed.

Collaboration diagram

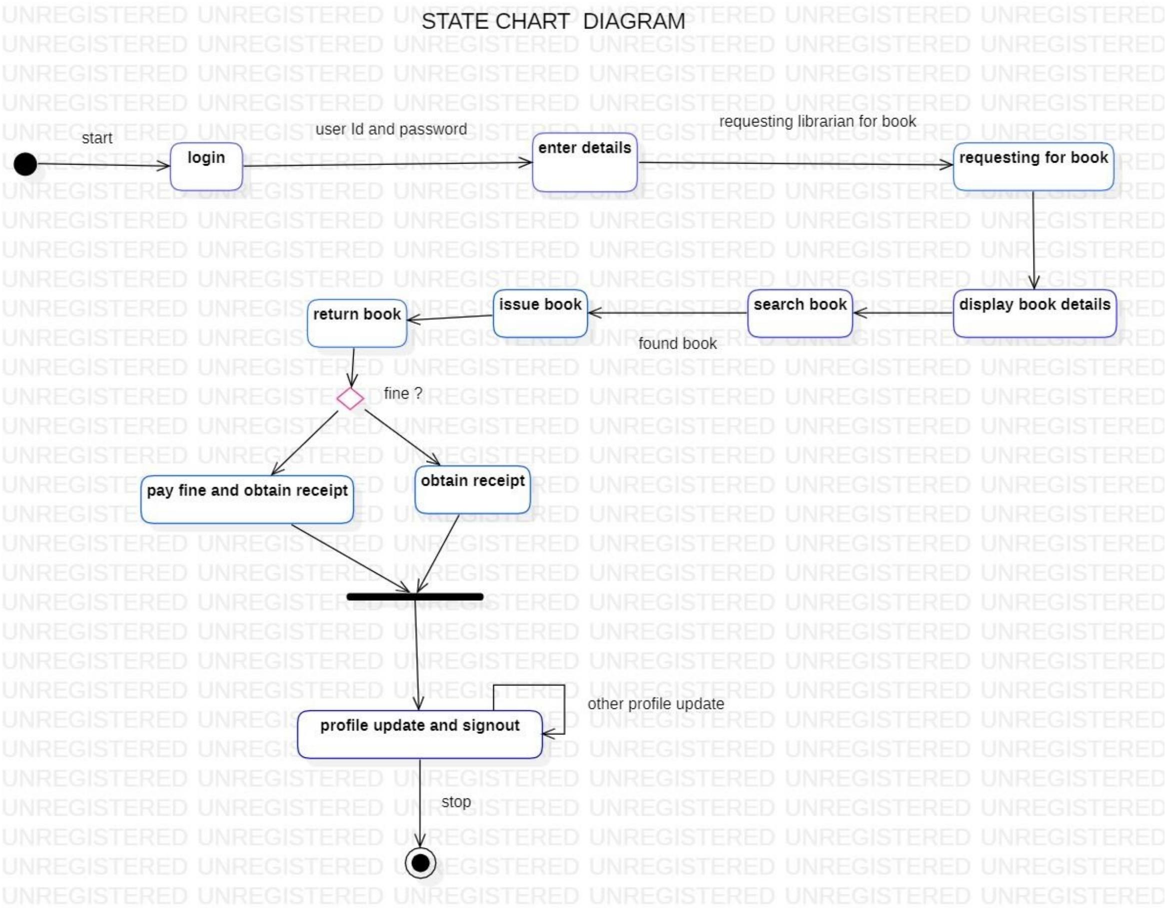


The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behavior of a system.

However, communication diagrams use the free-form arrangement of objects and links as used in Object diagrams. In order to maintain the ordering of messages in such a free-form diagram, messages are labeled with a chronological number and placed near the link the message is sent over. Reading a communication diagram involves starting at message 1.0, and following the messages from object to object.

Communication diagrams show much of the same information as sequence diagrams, but because of how the information is presented, some of it is easier to find in one diagram than the other. Communication diagrams show which elements each one interacts with better, but sequence diagrams show the order in which the interactions take place more clearly.

State chart diagram



The name of the diagram itself clarifies the purpose of the diagram and other details. It describes different states of a component in a system. The states are specific to a component/object of a system.A Statechart diagram describes a state machine. State machine can be defined as a machine which defines different states of an object and these states are controlled by external or internal events. Activity diagram explained in the next chapter, is a special kind of a Statechart diagram. As the Statechart diagram defines the states, it is used to model the lifetime of an object.Statechart diagram is one of the five UML diagrams used to model the dynamic nature of a system.

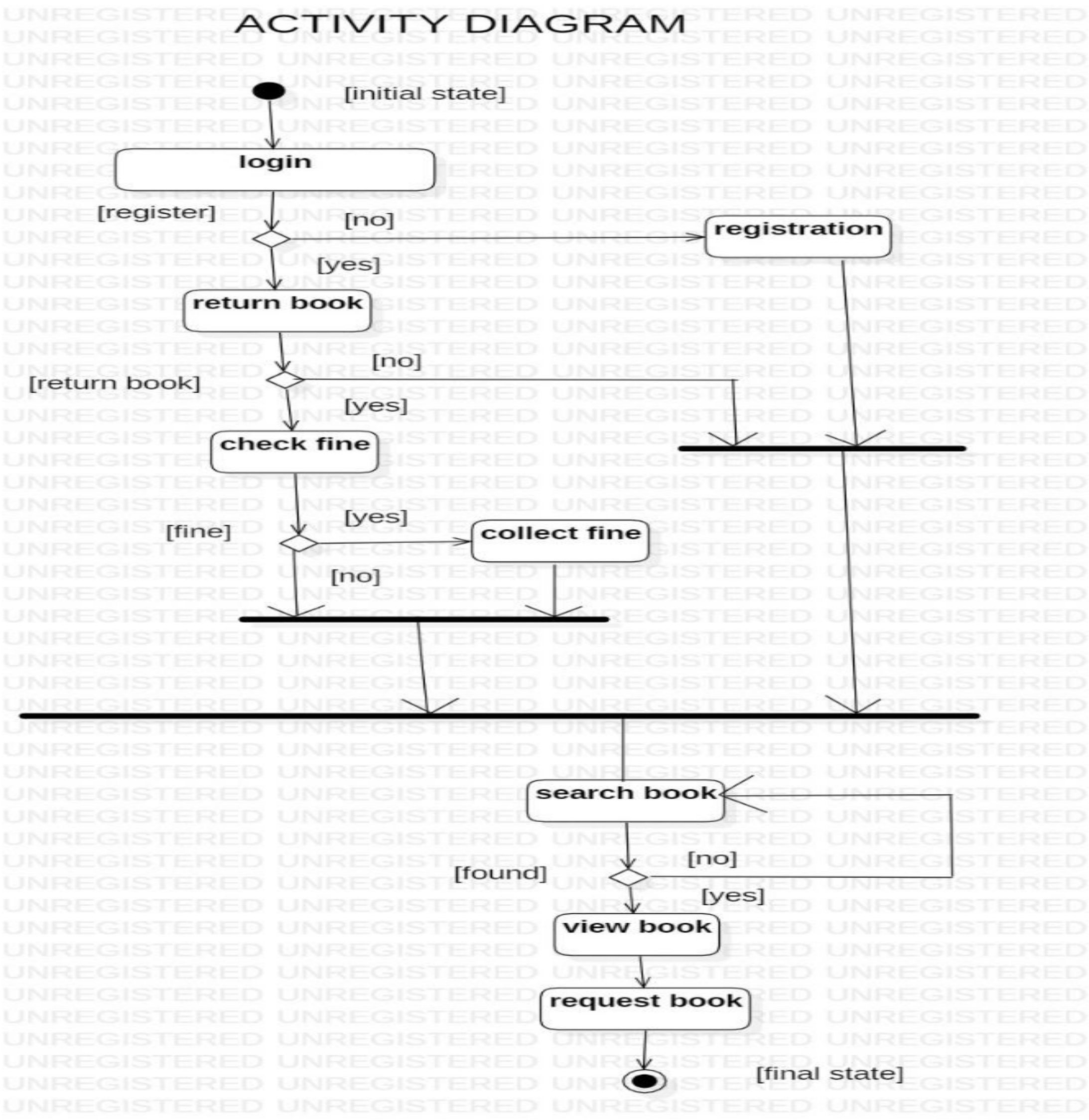
They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events.Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object exists and it changes when some event is triggered. The most important purpose of a Statechart diagram is to model the lifetime of an object from creation to termination.

Statechart diagrams are also used for forward and reverse engineering of a system. However, the main purpose is to model the reactive system.

Following are the main purposes of using Statechart diagrams −

- **To model the dynamic aspect of a system.**
- **To model the lifetime of a reactive system.**
- **To describe different states of an object during its lifetime.**
- **Define a state machine to model the states of an object.**

Activity diagram

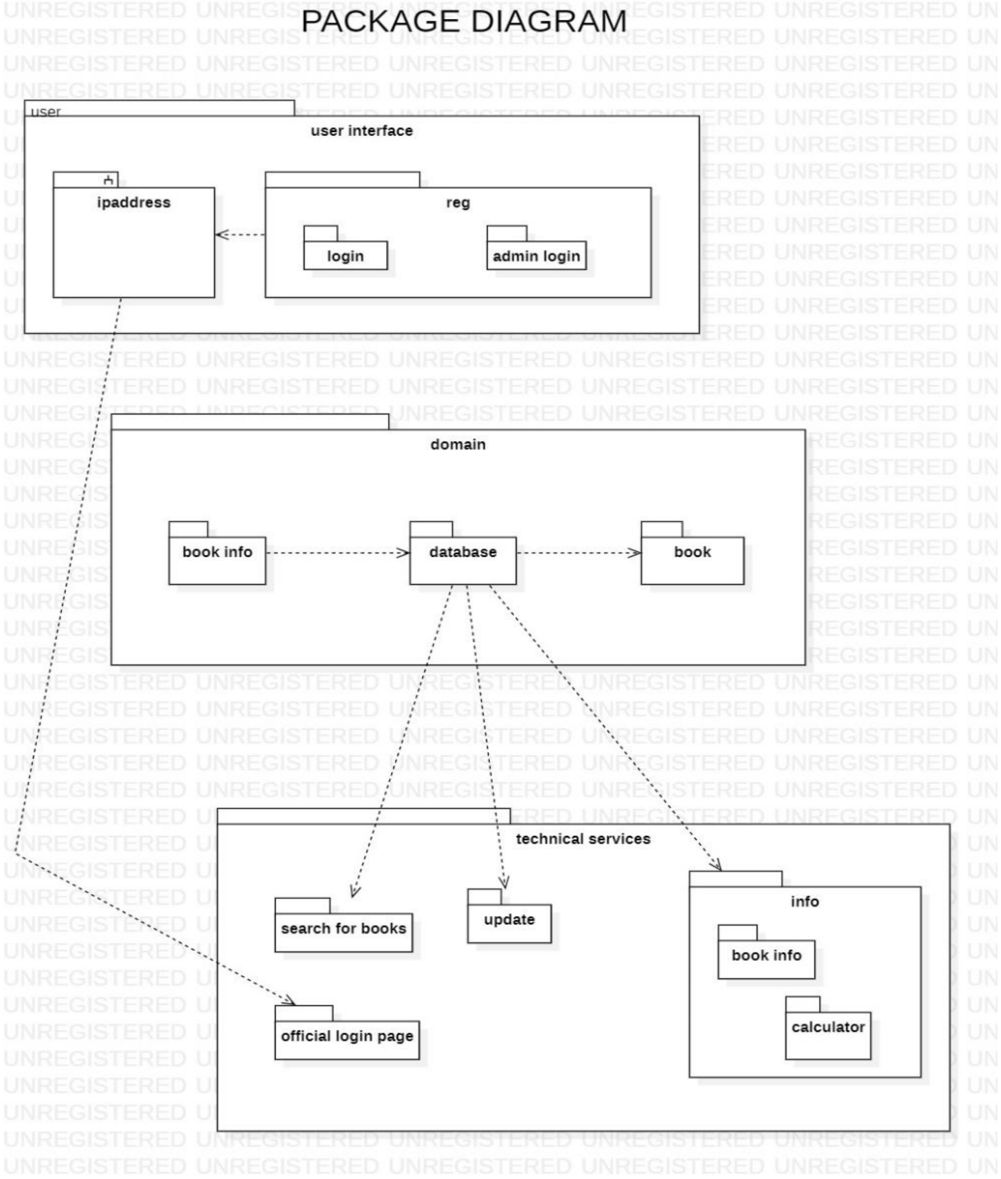


Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The basic purpose of activity diagrams is similar to the other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but the activity diagram is used to show the message flow from one activity to another.Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

Package diagram



A package diagram in the Unified Modeling Language depicts the dependencies between the packages that make up a model.

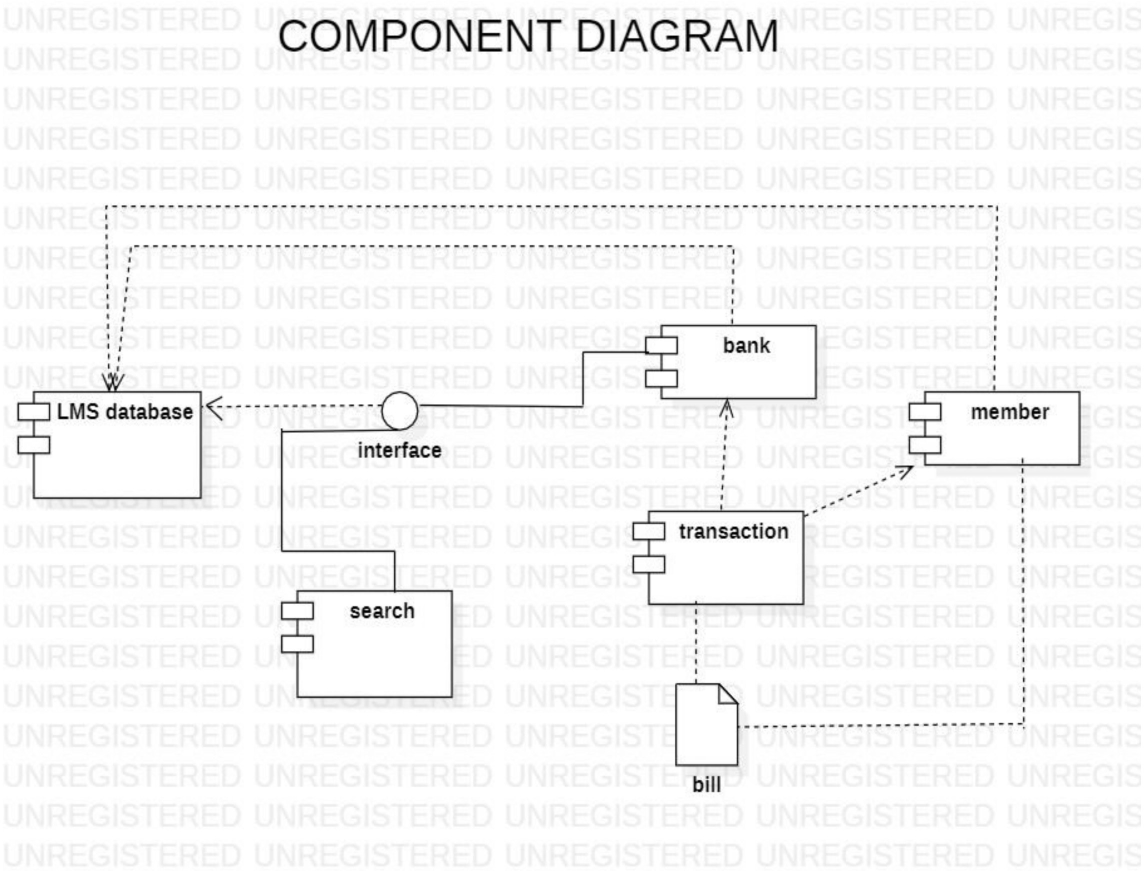
In addition to the standard UML Dependency relationship, there are two special types of dependencies defined between packages:

- **package import**
- **package merge**

A package import is a relationship between an importing namespace and a package, indicating that the importing namespace adds the names of the members of the package to its own namespace.] By default, an unlabeled dependency between two packages is interpreted as a package import relationship. In this relationship, elements within the target package will be imported into the source package.

A package merge is "a directed relationship between two packages, that indicates that the contents of the two packages are to be combined. It is very similar to Generalisation in the sense that the source element conceptually adds the characteristics of the target element to its own characteristics resulting in an element that combines the characteristics of both. In this relationship, if an element exists within both the source package and the target package, then the source element's definition will be expanded to include the target element's definition.

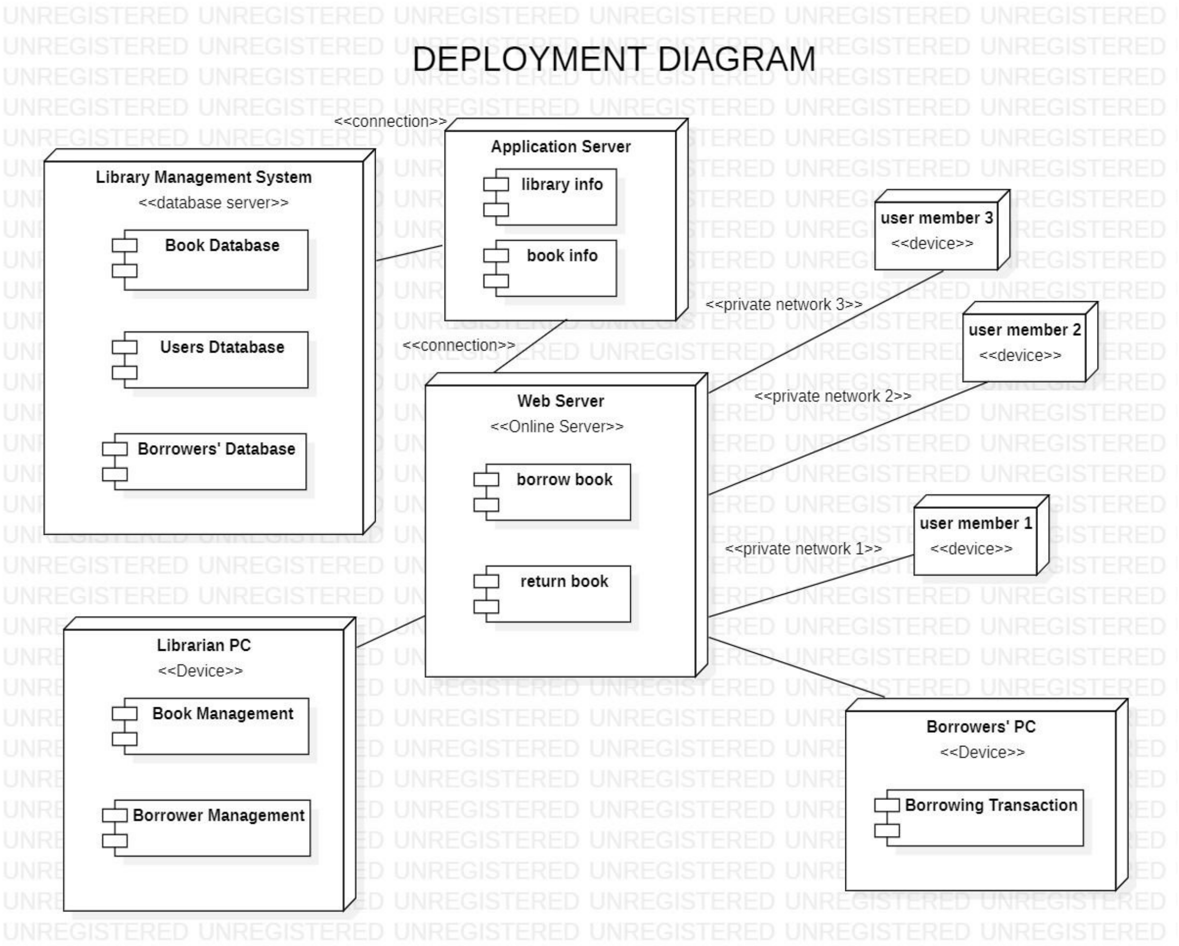
Component diagram



A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationships on the system.

The component diagram extends the information given in a component notation element. One way of illustrating the provided and required interfaces by the specified component is in the form of a rectangular compartment attached to the component element. Another accepted way of presenting the interfaces is to use the ball-and-socket graphic convention. A *provided* dependency from a component to an interface is illustrated with a solid line to the component using the interface from a "lollipop", or ball, labeled with the name of the interface. A required usage dependency from a component to an interface is illustrated by a half-circle, or socket, labeled with the name of the interface, attached by a solid line to the component that requires this interface. Inherited interfaces may be shown with a lollipop, preceding the name label with a caret symbol. To illustrate dependencies between the two, use a solid line with a plain arrowhead joining the socket to the lollipop.

Deployment diagram



A Deployment Diagram shows the configuration of run-time processing nodes and the components that live on them. Deployment diagrams address the static deployment view of architecture. They are related to component diagrams in that a node typically encloses one or more components.The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

1. **Device Node**
2. **Execution Environment Node**

Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.

CODE

```
#include<iostream>
#include<windows.h>
#include<stdio.h>
#include<conio.h>
#include <stdlib.h>
#include<string.h>                  //contains strcmp(),strcpy(),strlen(),etc
#include<ctype.h>                   //contains toupper(), tolower(),etc
#include<dos.h>                    //contains _dos_getdate
#include<time.h>

using namespace std;
//#include<bios.h>

#define RETURNTIME 15

char catagories[][15]={"Computer","Electronics","Electrical","Civil","Mechnnical","Architecture"};
void returnfunc(void);
void mainmenu(void);
void addbooks(void);
void deletebooks(void);
void editbooks(void);
void searchbooks(void);
void issuebooks(void);
void viewbooks(void);
void closeapplication(void);
int  getdata();
int  checkid(int);
int t(void);
//void show_mouse(void);
void Password();
void issuerecord();
void loaderanim();

//list of global files that can be acceed form anywhere in program
FILE *fp,*ft,*fs;
```

```
COORD coord = {0, 0};
//list of global variable
int s;
char findbook;
char password[10]={"1234"};

void gotoxy (int x, int y)
{
coord.X = x; coord.Y = y; // X and Y coordinates
SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
}

struct meroDate
{
int mm,dd,yy;
};
struct books
{
int id;
char stname[20];
char name[20];
char Author[20];
int quantity;
float Price;
int count;
int rackno;
char *cat;
struct meroDate issued;
struct meroDate duedate;
};
struct books a;
int main()
{
Password();
getch();
return 0;

}
void mainmenu()
{
```

```
//loaderanim();
system("cls");
//    textbackground(13);
int i;
gotoxy(20,3);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2 MAIN MENU
\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
//    show_mouse();
gotoxy(20,5);
printf("\xDB\xDB\xDB\xDB\xB2 1. Add Books   ");
gotoxy(20,7);
printf("\xDB\xDB\xDB\xDB\xB2 2. Delete books");
gotoxy(20,9);
printf("\xDB\xDB\xDB\xDB\xB2 3. Search Books");
gotoxy(20,11);
printf("\xDB\xDB\xDB\xDB\xB2 4. Issue Books");
gotoxy(20,13);
printf("\xDB\xDB\xDB\xDB\xB2 5. View Book list");
gotoxy(20,15);
printf("\xDB\xDB\xDB\xDB\xB2 6. Edit Book's Record");
gotoxy(20,17);
printf("\xDB\xDB\xDB\xDB\xB2 7. Close Application");
gotoxy(20,19);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,20);
t();
gotoxy(20,21);
printf("Enter your choice:");
switch(getch())
{
case '1':
addbooks();
break;
case '2':
deletebooks();
break;
case '3':
searchbooks();
break;
case '4':
```

```
issuebooks();
break;
case '5':
viewbooks();
break;
case '6':
editbooks();
break;
case '7':
{
system("cls");
gotoxy(16,3);
printf("\tLibrary Management System");
gotoxy(16,4);
printf("\tMini Project");
gotoxy(16,5);
printf("\tis brought to you by");
gotoxy(16,7);
printf("\tShreya, Arzob, Saptaparno");
gotoxy(16,8);
printf("**********************************************");
gotoxy(16,10);
printf("***********************************************");
gotoxy(16,11);
printf("***********************************************");
gotoxy(16,13);
printf("************************************************");
gotoxy(10,17);
printf("Exiting in 3 second..........>");
//flushall();
Sleep(3000);
exit(0);
}
default:
{
gotoxy(10,23);
printf("\aWrong Entry!!Please re-entered correct option");
if(getch())
mainmenu();
}
```

```
}
}
void addbooks(void)    //funtion that add books
{
system("cls");
int i;
gotoxy(20,5);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2SELECT
CATEGOIES\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,7);
printf("\xDB\xDB\xDB\xDB\xB2 1. Computer");
gotoxy(20,9);
printf("\xDB\xDB\xDB\xDB\xB2 2. Electronics");
gotoxy(20,11);
printf("\xDB\xDB\xDB\xDB\xB2 3. Electrical");
gotoxy(20,13);
printf("\xDB\xDB\xDB\xDB\xB2 4. Civil");
gotoxy(20,15);
printf("\xDB\xDB\xDB\xDB\xB2 5. Mechanical");
gotoxy(20,17);
printf("\xDB\xDB\xDB\xDB\xB2 6. Architecture");
gotoxy(20,19);
printf("\xDB\xDB\xDB\xDB\xB2 7. Back to main menu");
gotoxy(20,21);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,22);
printf("Enter your choice:");
scanf("%d",&s);
if(s==7)

mainmenu() ;
system("cls");
fp=fopen("Bibek.dat","ab+");
if(getdata()==1)
{
a.cat=catagories[s-1];
fseek(fp,0,SEEK_END);
fwrite(&a,sizeof(a),1,fp);
fclose(fp);
```

```
gotoxy(21,14);
printf("The record is sucessfully saved");
gotoxy(21,15);
printf("Save any more?(Y / N):");
if(getch()=='n')
mainmenu();
else
system("cls");
addbooks();
}
}
void deletebooks()    //function that delete items from file fp
{
system("cls");
int d;
char another='y';
while(another=='y')  //infinite loop
{
system("cls");
gotoxy(10,5);
printf("Enter the Book ID to  delete:");
scanf("%d",&d);
fp=fopen("Bibek.dat","rb+");
rewind(fp);
while(fread(&a,sizeof(a),1,fp)==1)
{
if(a.id==d)
{

gotoxy(10,7);
printf("The book record is available");
gotoxy(10,8);
printf("Book name is %s",a.name);
gotoxy(10,9);
printf("Rack No. is %d",a.rackno);
findbook='t';
}
}
if(findbook!='t')
{
```

```
gotoxy(10,10);
printf("No record is found modify the search");
if(getch())
mainmenu();
}
if(findbook=='t' )
{
gotoxy(10,9);
printf("Do you want to delete it?(Y/N):");
if(getch()=='y')
{
ft=fopen("test.dat","wb+");  //temporary file for delete
rewind(fp);
while(fread(&a,sizeof(a),1,fp)==1)
{
if(a.id!=d)
{
fseek(ft,0,SEEK_CUR);
fwrite(&a,sizeof(a),1,ft); //write all in tempory file except that
}                          //we want to delete
}
fclose(ft);
fclose(fp);
remove("Bibek.dat");
rename("test.dat","Bibek.dat"); //copy all item from temporary file to fp except that
fp=fopen("Bibek.dat","rb+");    //we want to delete
if(findbook=='t')
{
gotoxy(10,10);
printf("The record is sucessfully deleted");
gotoxy(10,11);
printf("Delete another record?(Y/N)");
}
}
else
mainmenu();
fflush(stdin);
another=getch();
}
}
```

```
gotoxy(10,15);
mainmenu();
}
void searchbooks()
{
system("cls");
int d;
printf("*****************************Search Books*********************************");
gotoxy(20,10);
printf("\xDB\xDB\xDB\xB2 1. Search By ID");
gotoxy(20,14);
printf("\xDB\xDB\xDB\xB2 2. Search By Name");
gotoxy( 15,20);
printf("Enter Your Choice");
fp=fopen("Bibek.dat","rb+"); //open file for reading propose
rewind(fp);   //move pointer at the begining of file
switch(getch())
{
case '1':
{
system("cls");
gotoxy(25,4);
printf("****Search Books By Id****");
gotoxy(20,5);
printf("Enter the book id:");
scanf("%d",&d);
gotoxy(20,7);
printf("Searching........");
while(fread(&a,sizeof(a),1,fp)==1)
{
if(a.id==d)
{
Sleep(2);
gotoxy(20,7);
printf("The Book is available");
gotoxy(20,8);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,9);
printf("\xB2 ID:%d",a.id);gotoxy(47,9);printf("\xB2");
gotoxy(20,10);
printf("\xB2 Name:%s",a.name);gotoxy(47,10);printf("\xB2");
gotoxy(20,11);
printf("\xB2 Author:%s ",a.Author);gotoxy(47,11);printf("\xB2");
gotoxy(20,12);
printf("\xB2 Qantity:%d ",a.quantity);gotoxy(47,12);printf("\xB2"); gotoxy(47,11);printf("\xB2");
gotoxy(20,13);
printf("\xB2 Price:Rs.%.2f",a.Price);gotoxy(47,13);printf("\xB2");
gotoxy(20,14);
printf("\xB2 Rack No:%d ",a.rackno);gotoxy(47,14);printf("\xB2");
gotoxy(20,15);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
findbook='t';
}


}
if(findbook!='t')  //checks whether conditiion enters inside loop or not
{
gotoxy(20,8);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,9);printf("\xB2");  gotoxy(38,9);printf("\xB2");
gotoxy(20,10);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(22,9);printf("\aNo Record Found");
}
gotoxy(20,17);
printf("Try another search?(Y/N)");
if(getch()=='y')
searchbooks();
else
mainmenu();
break;
}
case '2':
{
char s[15];
system("cls");
gotoxy(25,4);
printf("****Search Books By Name****");
gotoxy(20,5);
printf("Enter Book Name:");
```

```
scanf("%s",s);
int d=0;
while(fread(&a,sizeof(a),1,fp)==1)
{
if(strcmp(a.name,(s))==0) //checks whether a.name is equal to s or not
{
gotoxy(20,7);
printf("The Book is available");
gotoxy(20,8);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,9);
printf("\xB2 ID:%d",a.id);gotoxy(47,9);printf("\xB2");
gotoxy(20,10);
printf("\xB2 Name:%s",a.name);gotoxy(47,10);printf("\xB2");
gotoxy(20,11);
printf("\xB2 Author:%s",a.Author);gotoxy(47,11);printf("\xB2");
gotoxy(20,12);
printf("\xB2 Qantity:%d",a.quantity);gotoxy(47,12);printf("\xB2");
gotoxy(20,13);
printf("\xB2 Price:Rs.%.2f",a.Price);gotoxy(47,13);printf("\xB2");
gotoxy(20,14);
printf("\xB2 Rack No:%d ",a.rackno);gotoxy(47,14);printf("\xB2");
gotoxy(20,15);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\x
B2\xB2\xB2\xB2\xB2\xB2");
d++;
}


}
if(d==0)
{
gotoxy(20,8);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,9);printf("\xB2");  gotoxy(38,9);printf("\xB2");
gotoxy(20,10);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(22,9);printf("\aNo Record Found");
}
gotoxy(20,17);
printf("Try another search?(Y/N)");
```

```
if(getch()=='y')
searchbooks();
else
mainmenu();
break;
}
default :
getch();
searchbooks();
}
fclose(fp);
}
void issuebooks(void)  //function that issue books from library
{
int t;

system("cls");
printf("************************************ISSUE SECTION**************************");
gotoxy(10,5);
printf("\xDB\xDB\xDB\xDb\xB2 1. Issue a Book");
gotoxy(10,7);
printf("\xDB\xDB\xDB\xDb\xB2 2. View Issued Book");
gotoxy(10,9);
printf("\xDB\xDB\xDB\xDb\xB2 3. Search Issued Book");
gotoxy(10,11);
printf("\xDB\xDB\xDB\xDb\xB2 4. Remove Issued Book");
gotoxy(10,13);
printf("\xDB\xDB\xDB\xDb\xB2 5. Calculate Fine");
gotoxy(10,16);
printf("Enter a Choice:");
switch(getch())
{
case '1':  //issue book
{
system("cls");
int c=0;
char another='y';
while(another=='y')
{
system("cls");
```

```
gotoxy(15,4);
printf("***Issue Book section***");
gotoxy(10,6);
printf("Enter the Book Id:");
scanf("%d",&t);
fp=fopen("Bibek.dat","rb");
fs=fopen("Issue.dat","ab+");
if(checkid(t)==0) //issues those which are present in library
{
gotoxy(10,8);
printf("The book record is available");
gotoxy(10,9);
printf("There are %d unissued books in library ",a.quantity);
gotoxy(10,10);
printf("The name of book is %s",a.name);
gotoxy(10,11);
printf("Enter student name:");
scanf("%s",a.stname);
//struct dosdate_t d; //for current date
//_dos_getdate(&d);
//a.issued.dd=d.day;
//a.issued.mm=d.month;
//a.issued.yy=d.year;
gotoxy(10,12);
printf("Issued date=%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy);
gotoxy(10,13);
printf("The BOOK of ID %d is issued",a.id);
a.duedate.dd=a.issued.dd+RETURNTIME;   //for return date
a.duedate.mm=a.issued.mm;
a.duedate.yy=a.issued.yy;
if(a.duedate.dd>30)
{
a.duedate.mm+=a.duedate.dd/30;
a.duedate.dd-=30;

}
if(a.duedate.mm>12)
{
a.duedate.yy+=a.duedate.mm/12;
a.duedate.mm-=12;
}
gotoxy(10,14);
printf("To be return:%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);
fseek(fs,sizeof(a),SEEK_END);
fwrite(&a,sizeof(a),1,fs);
fclose(fs);
c=1;
}
if(c==0)
{
gotoxy(10,11);
printf("No record found");
}
gotoxy(10,15);
printf("Issue any more(Y/N):");
fflush(stdin);
another=getche();
fclose(fp);
}

break;
}
case '2':  //show issued book list
{
system("cls");
int j=4;
printf("***********************************Issued book list**************************************\n");
gotoxy(2,2);
printf("STUDENT NAME    CATEGORY    ID    BOOK NAME    ISSUED DATE    RETURN DATE");
fs=fopen("Issue.dat","rb");
while(fread(&a,sizeof(a),1,fs)==1)
{

gotoxy(2,j);
printf("%s",a.stname);
gotoxy(18,j);
printf("%s",a.cat);
gotoxy(30,j);
printf("%d",a.id);
```

```
gotoxy(36,j);
printf("%s",a.name);
gotoxy(51,j);
printf("%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy );
gotoxy(65,j);
printf("%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);

gotoxy(50,25);
//          printf("Current date=%d-%d-%d",d.day,d.month,d.year);
j++;

}
fclose(fs);
gotoxy(1,25);
returnfunc();
}
break;
case '3':   //search issued books by id
{
system("cls");
gotoxy(10,6);
printf("Enter Book ID:");
int p,c=0;
char another='y';
while(another=='y')
{

scanf("%d",&p);
fs=fopen("Issue.dat","rb");
while(fread(&a,sizeof(a),1,fs)==1)
{
if(a.id==p)
{
issuerecord();
gotoxy(10,12);
printf("Press any key.......");
getch();
issuerecord();
c=1;
}
```

```
}
fflush(stdin);
fclose(fs);
if(c==0)
{
gotoxy(10,8);
printf("No Record Found");
}
gotoxy(10,13);
printf("Try Another Search?(Y/N)");
another=getch();
}
}
break;
case '4':  //remove issued books from list
{
system("cls");
int b;
FILE *fg;  //declaration of temporary file for delete
char another='y';
while(another=='y')
{
gotoxy(10,5);
printf("Enter book id to remove:");
scanf("%d",&b);
fs=fopen("Issue.dat","rb+");
while(fread(&a,sizeof(a),1,fs)==1)
{
if(a.id==b)
{
issuerecord();
findbook='t';
}
if(findbook=='t')
{
gotoxy(10,12);
printf("Do You Want to Remove it?(Y/N)");
if(getch()=='y')
{
```

```
fg=fopen("record.dat","wb+");
rewind(fs);
while(fread(&a,sizeof(a),1,fs)==1)
{
if(a.id!=b)
{
fseek(fs,0,SEEK_CUR);
fwrite(&a,sizeof(a),1,fg);
}
}
fclose(fs);
fclose(fg);
remove("Issue.dat");
rename("record.dat","Issue.dat");
gotoxy(10,14);
printf("The issued book is removed from list");

}


}
if(findbook!='t')
{
gotoxy(10,15);
printf("No Record Found");
}
}
gotoxy(10,16);
printf("Delete any more?(Y/N)");
another=getch();
}
}
case '5':
{
   int days;
   cout<<"Enter number of days overdue "<<endl;
   cin>>days;
   cout<<"Fine = "<<(days * 5)<<endl;
}
break;
default:
```

```
gotoxy(10,18);
printf("\aWrong Entry!!");
getch();
issuebooks();
break;
}
gotoxy(1,30);
returnfunc();
}
void viewbooks(void)  //show the list of book persists in library
{
int i=0,j;
system("cls");
gotoxy(1,1);
printf("********************************Book List****************************");
gotoxy(2,2);
printf(" CATEGORY     ID    BOOK NAME     AUTHOR       QTY     PRICE     RackNo ");
j=4;
fp=fopen("Bibek.dat","rb");
while(fread(&a,sizeof(a),1,fp)==1)
{
gotoxy(3,j);
printf("%s",a.cat);
gotoxy(16,j);
printf("%d",a.id);
gotoxy(22,j);
printf("%s",a.name);
gotoxy(36,j);
printf("%s",a.Author);
gotoxy(50,j);
printf("%d",a.quantity);
gotoxy(57,j);
printf("%.2f",a.Price);
gotoxy(69,j);
printf("%d",a.rackno);
printf("\n\n");
j++;
i=i+a.quantity;
}
gotoxy(3,25);
```

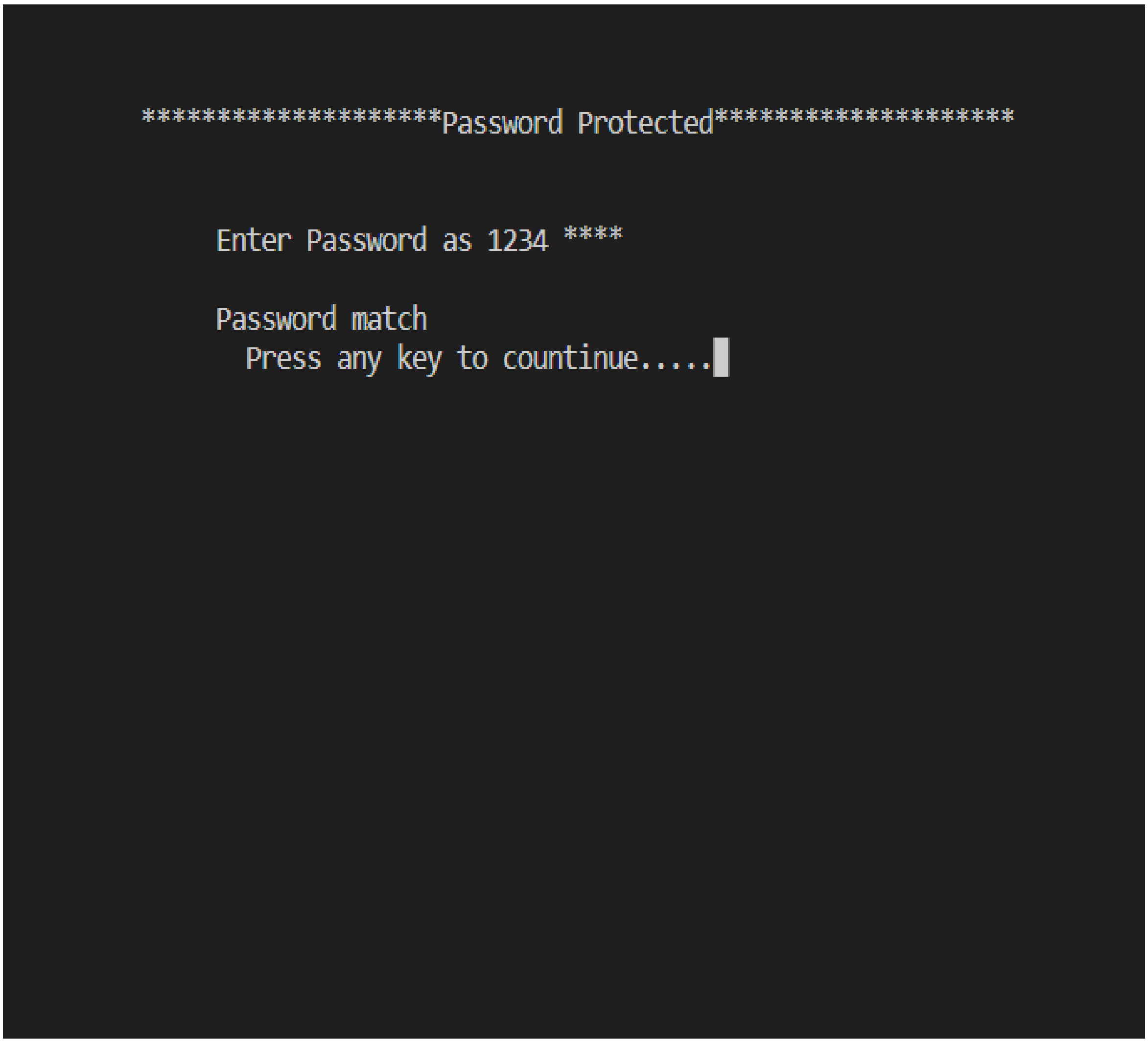
```
printf("Total Books =%d",i);
fclose(fp);
gotoxy(35,25);
returnfunc();
}
void editbooks(void)  //edit information about book
{
system("cls");
int c=0;
int d,e;
gotoxy(20,4);
printf("****Edit Books Section****");
char another='y';
while(another=='y')
{
system("cls");
gotoxy(15,6);
printf("Enter Book Id to be edited:");
scanf("%d",&d);
fp=fopen("Bibek.dat","rb+");
while(fread(&a,sizeof(a),1,fp)==1)
{
if(checkid(d)==0)
{
gotoxy(15,7);
printf("The book is availble");
gotoxy(15,8);
printf("The Book ID:%d",a.id);
gotoxy(15,9);
printf("Enter new name:");scanf("%s",a.name);
gotoxy(15,10);
printf("Enter new Author:");scanf("%s",a.Author);
gotoxy(15,11);
printf("Enter new quantity:");scanf("%d",&a.quantity);
gotoxy(15,12);
printf("Enter new price:");scanf("%f",&a.Price);
gotoxy(15,13);
printf("Enter new rackno:");scanf("%d",&a.rackno);
gotoxy(15,14);
printf("The record is modified");
fseek(fp,ftell(fp)-sizeof(a),0);
fwrite(&a,sizeof(a),1,fp);
fclose(fp);
c=1;
}
if(c==0)
{
gotoxy(15,9);
printf("No record found");
}
}
gotoxy(15,16);
printf("Modify another Record?(Y/N)");
fflush(stdin);
another=getch() ;
}
returnfunc();
}
void returnfunc(void)
{
{
printf(" Press ENTER to return to main menu");
}
a:
if(getch()==13) //allow only use of enter
mainmenu();
else
goto a;
}
int getdata()
{
int t;
gotoxy(20,3);printf("Enter the Information Below");
gotoxy(20,4);printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(20,5);
printf("\xB2");gotoxy(46,5);printf("\xB2");
gotoxy(20,6);
printf("\xB2");gotoxy(46,6);printf("\xB2");
gotoxy(20,7);
printf("\xB2");gotoxy(46,7);printf("\xB2");
```

```
gotoxy(20,8);
printf("\xB2");gotoxy(46,8);printf("\xB2");
gotoxy(20,9);
printf("\xB2");gotoxy(46,9);printf("\xB2");
gotoxy(20,10);
printf("\xB2");gotoxy(46,10);printf("\xB2");
gotoxy(20,11);
printf("\xB2");gotoxy(46,11);printf("\xB2");
gotoxy(20,12);
printf("\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2\xB2");
gotoxy(21,5);
printf("Category:");
gotoxy(31,5);
printf("%s",catagories[s-1]);
gotoxy(21,6);
printf("Book ID:\t");
gotoxy(30,6);
scanf("%d",&t);
if(checkid(t) == 0)
{
gotoxy(21,13);
printf("\aThe book id already exists\a");
getch();
mainmenu();
return 0;
}
a.id=t;
gotoxy(21,7);
printf("Book Name:");gotoxy(33,7);
scanf("%s",a.name);
gotoxy(21,8);
printf("Author:");gotoxy(30,8);
scanf("%s",a.Author);
gotoxy(21,9);
printf("Quantity:");gotoxy(31,9);
scanf("%d",&a.quantity);
gotoxy(21,10);
printf("Price:");gotoxy(28,10);
scanf("%f",&a.Price);
gotoxy(21,11);
```

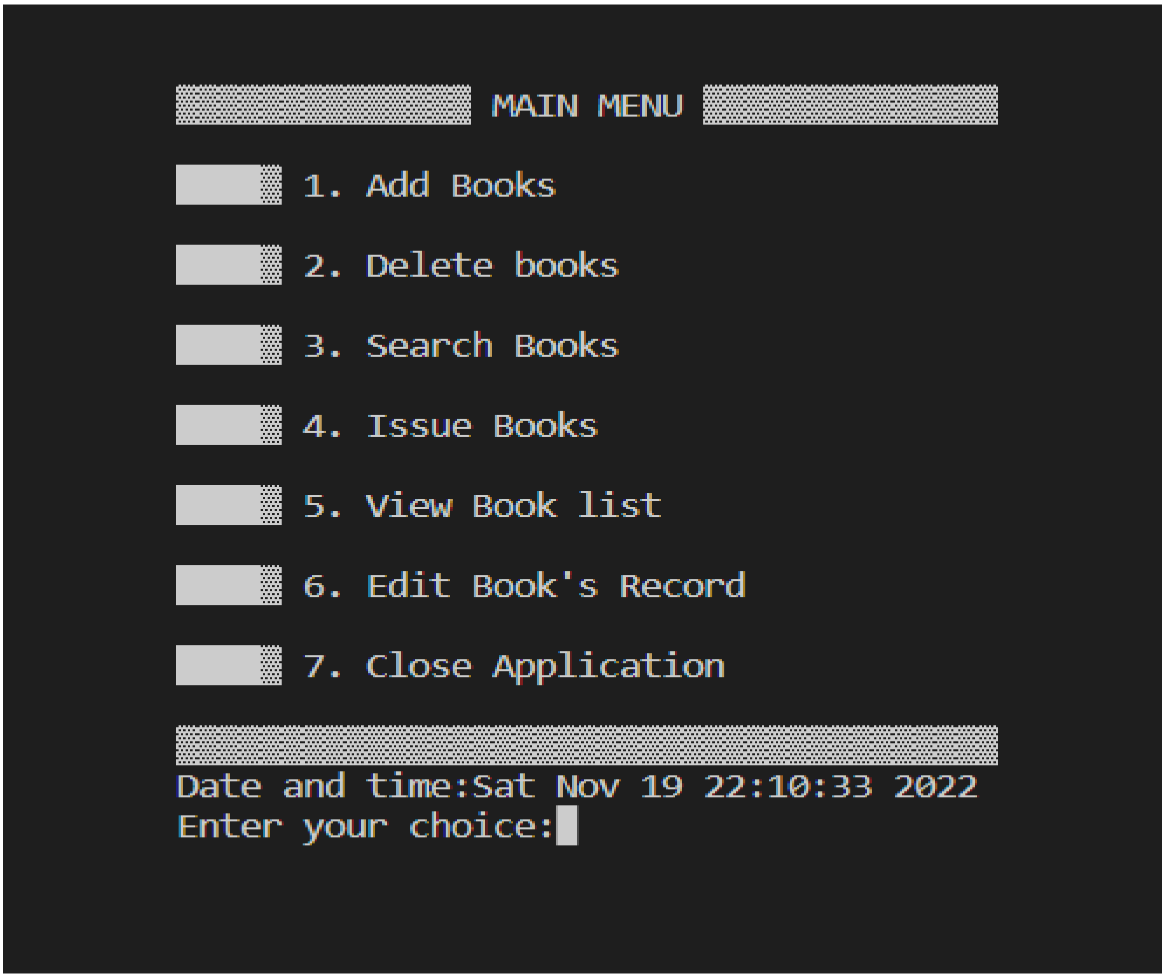
```
printf("Rack No:");gotoxy(30,11);
scanf("%d",&a.rackno);
return 1;
}
int checkid(int t)  //check whether the book is exist in library or not
{
rewind(fp);
while(fread(&a,sizeof(a),1,fp)==1)
if(a.id==t)
return 0;  //returns 0 if book exits
return 1; //return 1 if it not
}
int t(void) //for time
{
time_t t;
time(&t);
printf("Date and time:%s\n",ctime(&t));

return 0 ;
}

void Password(void) //for password option
{

system("cls");
char d[25]="Password Protected";
char ch,pass[10];
int i=0,j;
//textbackground(WHITE);
//textcolor(RED);
gotoxy(10,4);
for(j=0;j<20;j++)
{
Sleep(50);
printf("*");
}
for(j=0;j<20;j++)
{
Sleep(50);
printf("%c",d[j]);
}
for(j=0;j<20;j++)
{
Sleep(50);
printf("*");
}
gotoxy(10,10);
gotoxy(15,7);
printf("Enter Password as 1234 ");

while(ch!=13)
{
ch=getch();

if(ch!=13 && ch!=8){
putch('*');
pass[i] = ch;
i++;
}
}
pass[i] = '\0';
if(strcmp(pass,password)==0)
{

gotoxy(15,9);
//textcolor(BLINK);
printf("Password match");
gotoxy(17,10);
printf("Press any key to countinue.....");
getch();
mainmenu();
}
else
{
gotoxy(15,16);
printf("\aWarning!! Incorrect Password");
getch();
Password();
}
}
```

```
void issuerecord()  //display issued book's information
{
system("cls");
gotoxy(10,8);
printf("The Book has taken by Mr. %s",a.stname);
gotoxy(10,9);
printf("Issued Date:%d-%d-%d",a.issued.dd,a.issued.mm,a.issued.yy);
gotoxy(10,10);
printf("Returning Date:%d-%d-%d",a.duedate.dd,a.duedate.mm,a.duedate.yy);
}
```

OUTPUT

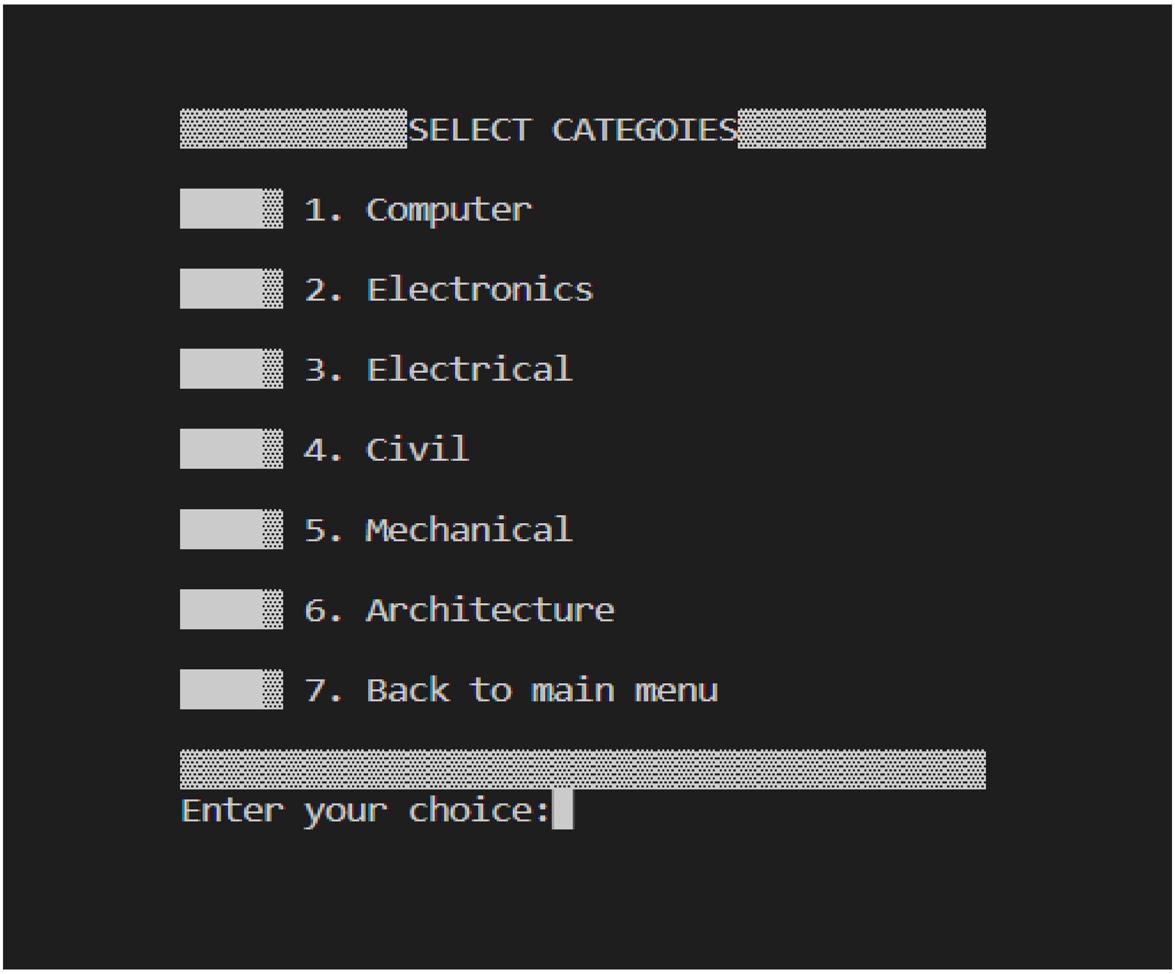


```
********************Password Protected********************


     Enter Password as 1234 ****

     Password match
       Press any key to countinue.....
```



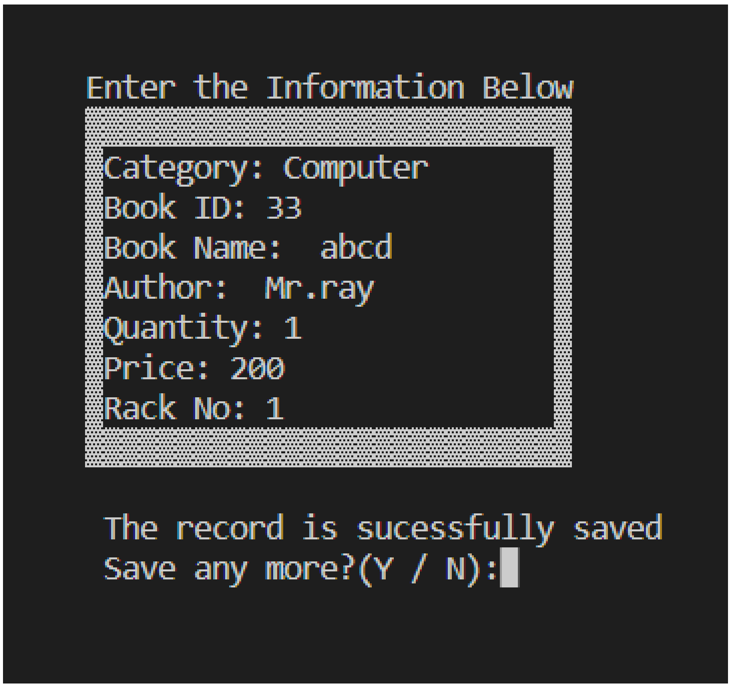
```
▒▒▒▒▒▒▒▒▒▒▒▒▒▒ MAIN MENU ▒▒▒▒▒▒▒▒▒▒▒▒▒▒

█████▒ 1. Add Books

█████▒ 2. Delete books

█████▒ 3. Search Books

█████▒ 4. Issue Books

█████▒ 5. View Book list

█████▒ 6. Edit Book's Record

█████▒ 7. Close Application

▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
Date and time:Sat Nov 19 22:10:33 2022
Enter your choice:
```

After entering ‘1’ -



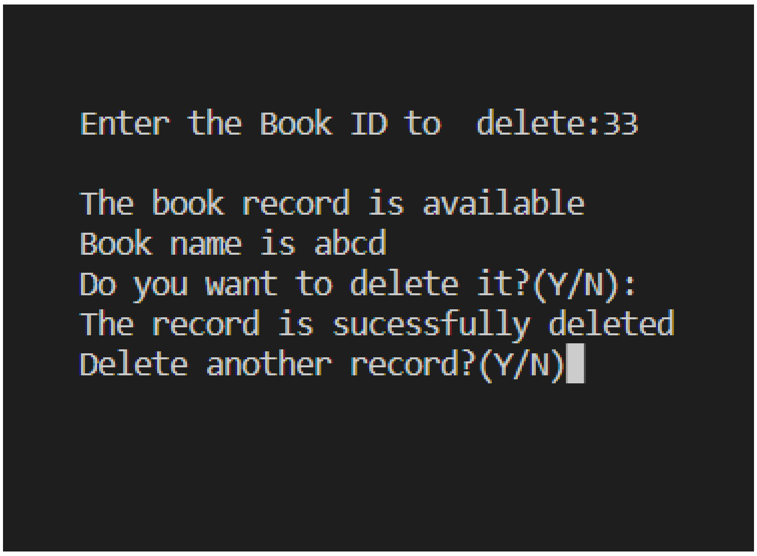
```
▒▒▒▒▒▒▒▒▒▒▒SELECT CATEGOIES▒▒▒▒▒▒▒▒▒▒▒▒

▒▒▒▒▒ 1. Computer

▒▒▒▒▒ 2. Electronics

▒▒▒▒▒ 3. Electrical

▒▒▒▒▒ 4. Civil

▒▒▒▒▒ 5. Mechanical

▒▒▒▒▒ 6. Architecture

▒▒▒▒▒ 7. Back to main menu

▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒
Enter your choice:█
```

Entering ‘1’ again -



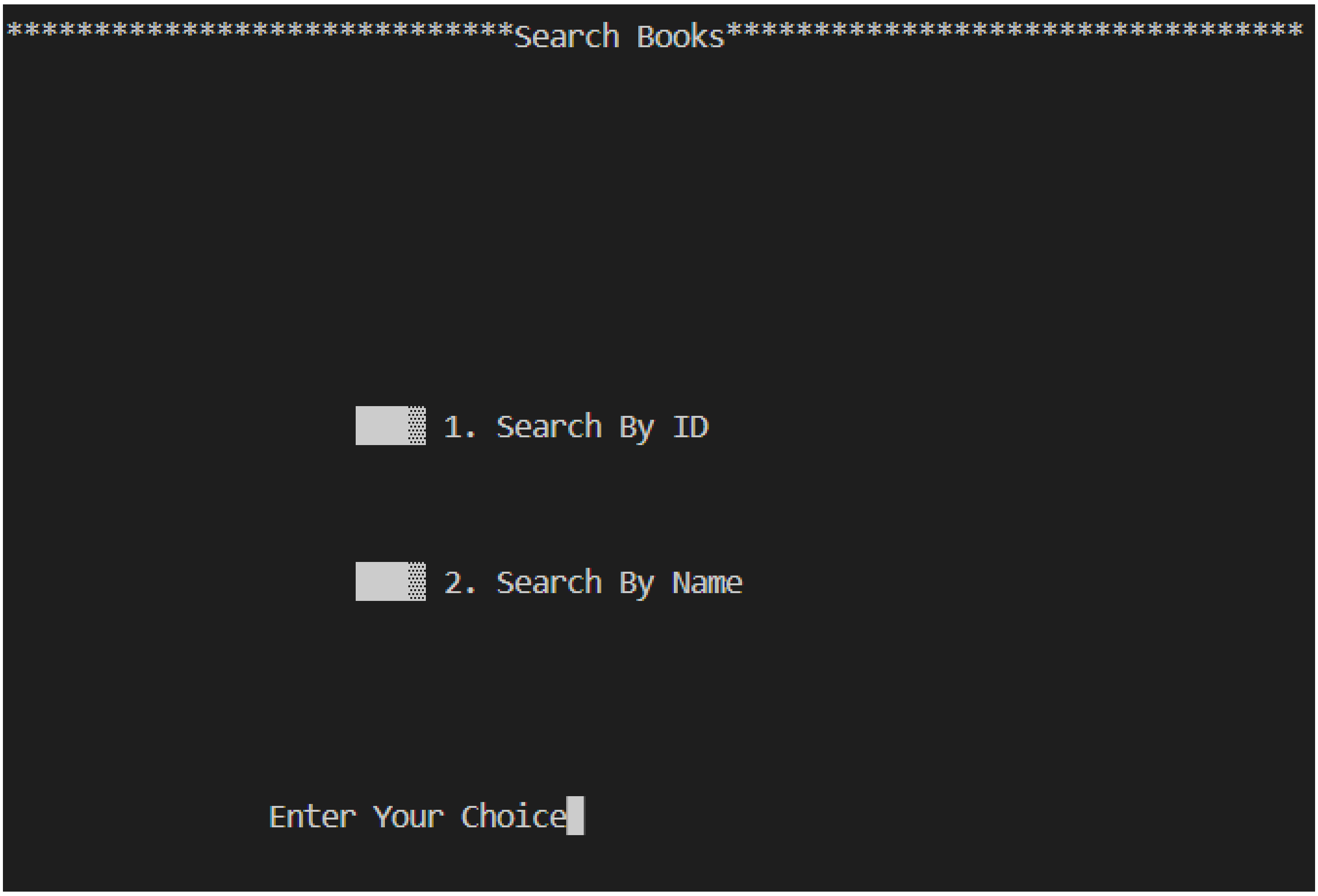
```
Enter the Information Below
Category: Computer
Book ID: 33
Book Name:  abcd
Author:  Mr.ray
Quantity: 1
Price: 200
Rack No: 1

 The record is sucessfully saved
 Save any more?(Y / N):█
```

Deletion -



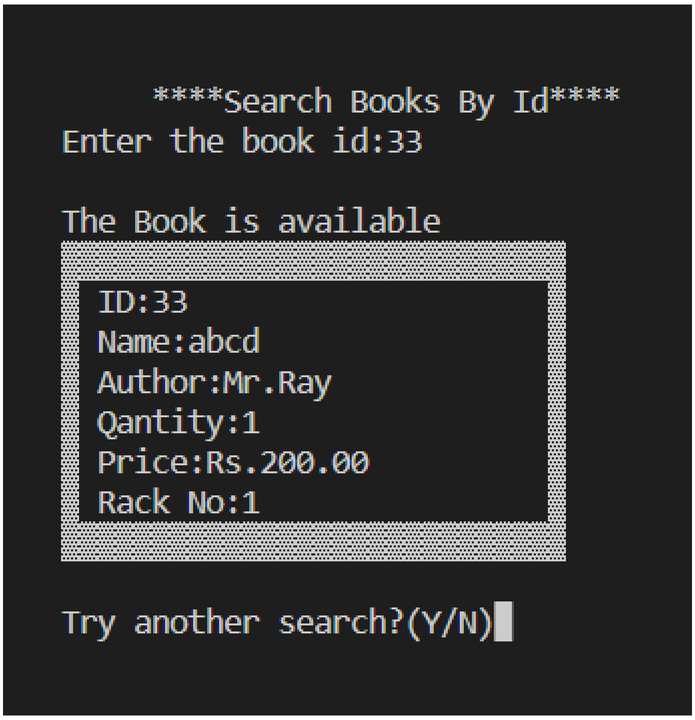
```
    Enter the Book ID to  delete:33

    The book record is available
    Book name is abcd
    Do you want to delete it?(Y/N):
    The record is sucessfully deleted
    Delete another record?(Y/N)█
```

Search Book:



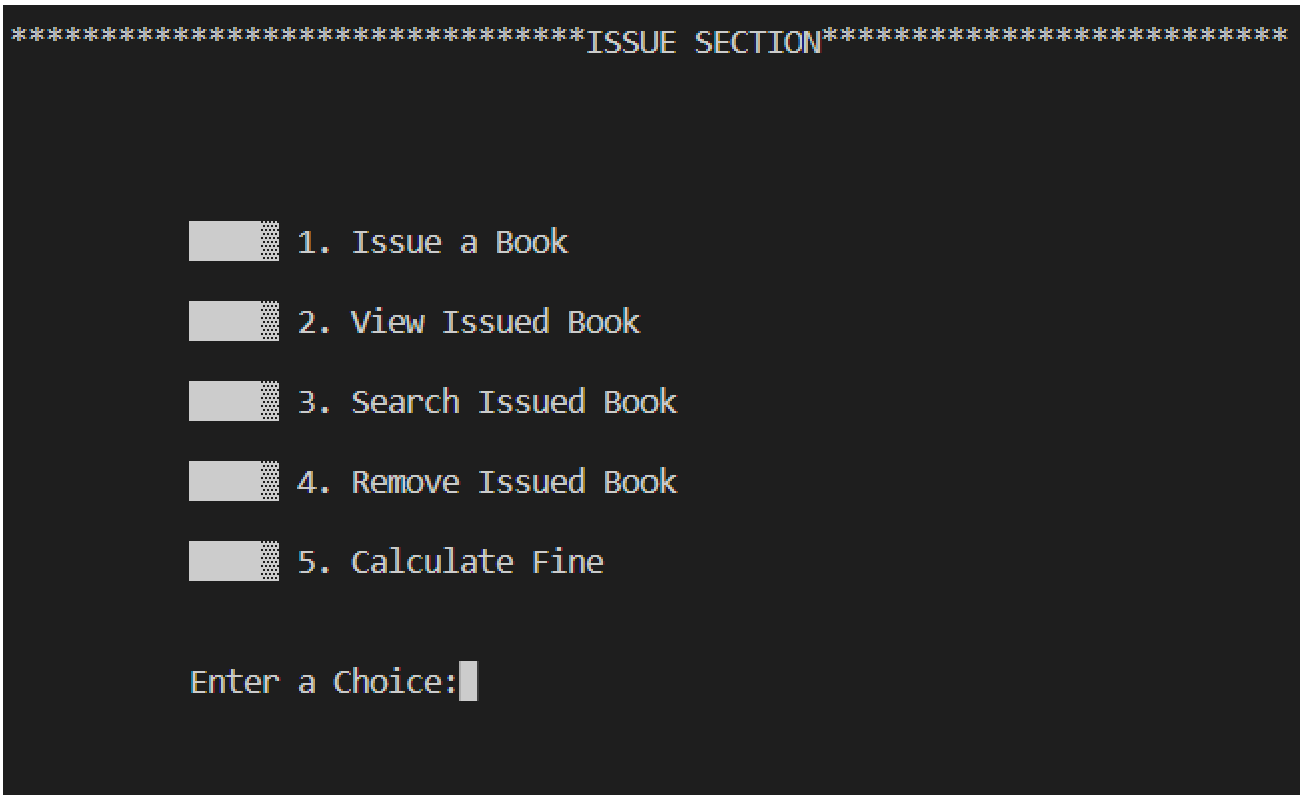
```
*****************************Search Books*********************************









                    ███▒ 1. Search By ID


                    ███▒ 2. Search By Name




               Enter Your Choice█
```

Entering ‘1’:



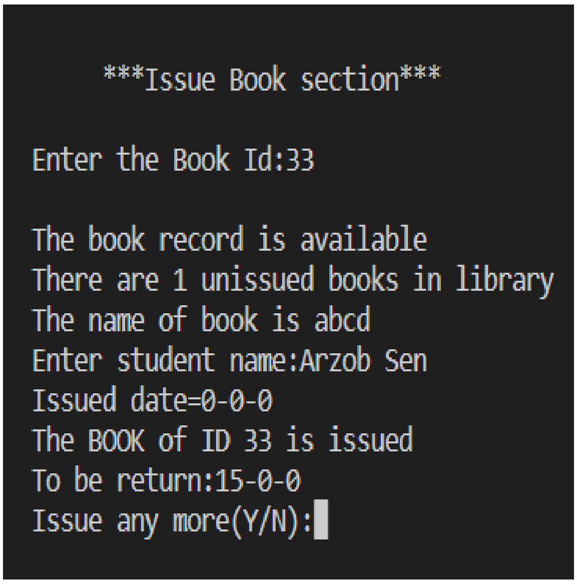
```
    ****Search Books By Id****
Enter the book id:33

The Book is available

   ID:33
   Name:abcd
   Author:Mr.Ray
   Qantity:1
   Price:Rs.200.00
   Rack No:1


Try another search?(Y/N)
```

Issuing Book:



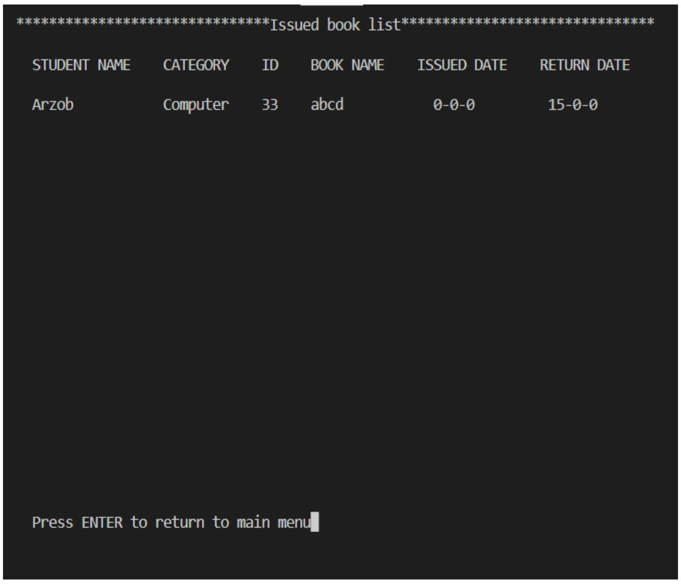
```
********************************ISSUE SECTION**************************




             1. Issue a Book

             2. View Issued Book

             3. Search Issued Book

             4. Remove Issued Book

             5. Calculate Fine


          Enter a Choice:
```

Entering ‘1’:

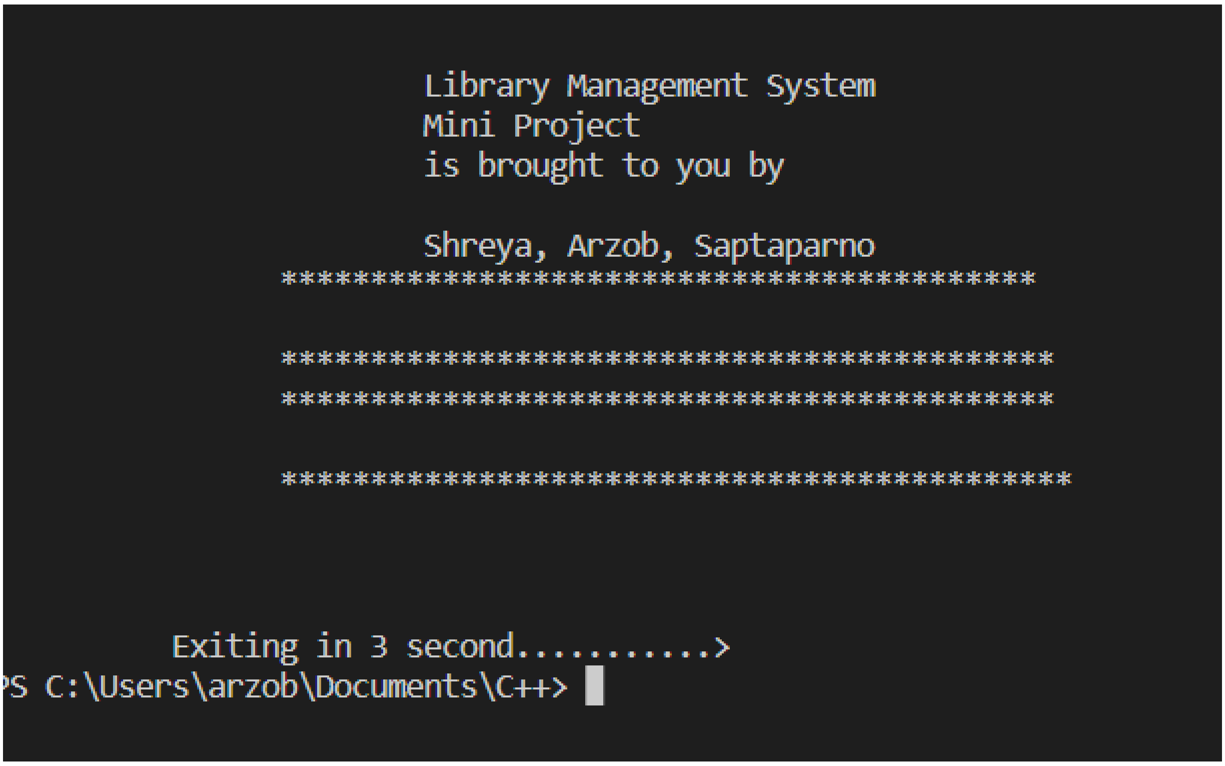


```
     ***Issue Book section***

 Enter the Book Id:33

 The book record is available
 There are 1 unissued books in library
 The name of book is abcd
 Enter student name:Arzob Sen
 Issued date=0-0-0
 The BOOK of ID 33 is issued
 To be return:15-0-0
 Issue any more(Y/N):
```

Entering ‘2’:



```
*******************************Issued book list*******************************

  STUDENT NAME    CATEGORY    ID    BOOK NAME    ISSUED DATE    RETURN DATE

  Arzob           Computer    33    abcd           0-0-0          15-0-0










  Press ENTER to return to main menu
```



```
                        Library Management System
                        Mini Project
                        is brought to you by

                        Shreya, Arzob, Saptaparno
                ******************************************

                *******************************************
                *******************************************

                ********************************************




          Exiting in 3 second...........>
PS C:\Users\arzob\Documents\C++>
```

Conclusion

All in all, the Library Management System UML Diagrams work together to achieve their most desired functions. All of these are designed to guide what should be the behavior and structure of the Library Management System.This project of "LIBRARY MANAGEMENT" gives us the complete information about the library.Throughout the project, the focus has been on presenting information in an easy and intelligible manner. The project is useful for those who want to know about the Library Management System. The software takes care of all the requirements of a library and is capable of providing easy and effective storage of information related to books and users.

Thus, the result shows that the system help for a quick management of data, reduces workload and expedite work procedure for a quick service of the library.

References

- **Michael R Blaha, James R Rumbaugh. The Object-Oriented Modeling and Design with UML, Second Edition. Dorling Kindersley, India: Pearson Prentice Hall, 2010.**
- **https://www.javatpoint.com/**
- **https://www.geeksforgeeks.org/**
- C++ Classes and Objects - W3Schools Online Web
- Tutorials UML Tutorial - UML Tutorial - Javatpoint