Library Management System

Mini Project Report -Database Lab (DSE 2260)

Department of Data Science & Computer Applications



B. Tech Data Science

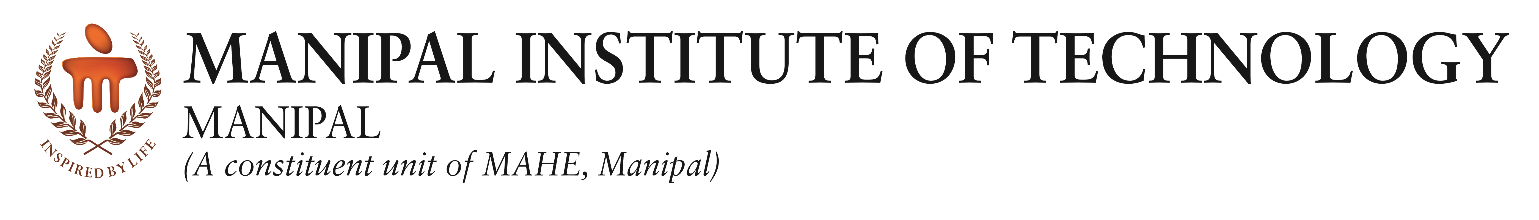
4th Semester – Batch: B1/B2/B3/B4

Submitted By

|  |  |
| --- | --- |
| Jonala Venkat Vardhan | 200968158 |
| Mihir Agarwal | 200968160 |
| Shobhit Jain | 200968162 |
| Soham Ghosh | 200968166 |

**Mentored By**

|  |  |
| --- | --- |
| Vinayak M | Archana H/ Shameem |
| Assistant Professor-Senior | Assistant Professor-Senior |
| DSCA, MIT | DSCA, MIT |



Date:

**CERTIFICATE**

This is to certify that the <Jonala Venkat Vardhan (200968158), Mihir Agarwal (200968160), Shobhit Jain(200968162), Soham Ghosh(200968166)>, have successfully executed a mini project titled “Library Management System” rightly brining fore the competencies and skill sets they have gained during the course- Database Lab (DSE 2262 & DSE ), thereby resulting in the culmination of this project.

|  |  |
| --- | --- |
| **Vinayak M** | **Archana H / Shameem** |
| **Assistant Professor-Senior** | **Assistant Professor-Senior** |
| **DSCA, MIT** | **DSCA, MIT** |

**ABSTRACT**

All libraries house many books, offer membership to numerous people, lend books to an equally large crowd, and even offer employment to quite a few people. Recording, maintaining, and organising all this information manually is a time consuming and complex task. This makes the process of lending and returning books, registering for membership in the library, changing personal details in library records, and keeping track of books, all very difficult. In the modern fast paced world, where time is limited, the work needs to be done instantly and not over the course of days or weeks. Be it from requesting for a book, to updating the personal information, everything needs to be completed quickly and efficiently. Hence, we need a Library Management System which can efficiently and quickly perform all of the above-mentioned tasks of storing information on books, staff and books issued.

The system is designed by first creating the conceptual design which involves the ER diagram, followed by the Logical design wherein we define the schema and data dictionary. Finally, the physical design is approached by creating the relations and constraints.

This system allows for all processes to be completed in a timely and efficient manner. The significance of this system is that it reduces overheads and increases productivity. The librarian can maintain it easily and it keeps on track the records of all transactions of books that have taken place in the library. It enables the librarians and users to save time on daunting tasks and enhances efficiency.

The Library Management System is beneficial to both the users and the librarians. It makes entire process online where student can search books, staff can generate reports and do book transactions. It also has a facility for user login where users can login and can see status of books issued and any or all pending dues that there might be.

Contents

**1. Introduction 4**

**2. Synopsis 5**

**2.1 Proposed System 5**

**2.2 Objectives 5**

**3. Functional Requirements 6**

**4. Detailed Design 8**

**4.1 ER Diagram 8**

**4.2 Schema Diagram 8**

**4.3 Data Dictionary 10**

**4.4 Relational Model Implementation 11**

**4.5 Queries 13**

**4.7 Triggers 13**

**4.8 Stored Procedures 14**

**4.9 Stored Functions 17**

**5. Implementation Functional Requirements 20**

**6. Result 20**

**7. Conclusion and Future Work 22**

Chapter 1

Introduction

Library Management System has a lot of importance in today’s world. Today, people are becoming victims of the progression of technology and are wasting their precious time on electronic gadgets and virtual reality. And libraries in schools and colleges are becoming abandoned places. Everything needs to be revamped and modernized to attract the students and common people back among the shelves, where true knowledge needs to be explored.

Libraries are facing maintenance problems for a very long time. Libraries are suffering from many issues like improper management, lack of space, ineffective staff etc. It is a meticulous and attentive task to maintain and store old records, books and materials and without a proper system, it becomes disorganized for the readers in the library. Libraries are also facing a lack of financial support to meet the demands they required for the change.

Libraries in schools and universities must cater to such demands. So, to keep in touch with the latest technological advancements, it is time for libraries to decrease their dependency on manual work and take the aid of computers and sophisticated technology for its management and services. In this system, the librarians can keep track of the issue date, return date, books of different genres with the help of a unique BookId, number of books issues, number of copies of books, borrower’s details, book loans and branch details.

With the help of barcodes, the whole process becomes rapid and flawless. It categorizes the books as per its genre. With the help of E-Mail and Messages, the librarian can send important notices regarding book returns, lists of books available, a list of new stock, holidays, important events, book fairs, fines/penalties etc. This kind of system helps students to themselves up to date with the books issued to them. It is easily accessible anytime and anywhere for anyone. A well-organized library management system enhances the efficiency of the library and helps supervise the library resources to save time and effort.

Chapter 2

Synopsis

**2.1 Proposed System**

It has created a facility where students after logging in their accounts can view the list of books been borrowed and its issue date and the return date as well. The librarian after logging into his/her account can rise sundry reports such as student report, issue report, pedagogic report etc.

The proposed system contains the following features:

* The students will have to register their account online.
* Individually each member will have his account through which he/she can access the information he/she needs.
* Book details like authors, number of copies totally maintained by library, present available number of books, reference books, non-reference books etc. all this information can be very useful.
* Regarding the members designation, number of books was issued.
* Issue dates and returns of each member is maintained separately and fine charged if there is any delay in returning the book.
* Administrator can add, update the books.

It gives accurate results and improves the reliability

**2.2 Objectives**

The main objective of the work is:

* Search availability of books
* Borrowers can find books issued and the date of return
* Virtual library facility
* Tracking all issues of books and pending fines

**Chapter 3**

**Functional Requirements**

The report module is uniquely included in Library Management System. If the user’s position is admin, the utilizer can engender different kinds of reports lists of students registered, list of books issued and return reports. All these modules can avail librarians to manage the library with more accommodative and in a more efficient way as compared to library systems which are not computerized.

**3.1 Register New User/Librarian**

This feature can be performed by all users to register new user to create account.

Functional requirements:

* System must be able to verify information
* System must be able to reject if information entered is wrong

**3.2 Register New Book**

This feature allows to add new books to the library

Functional requirements:

* System must be able to verify information
* System must be able to enter number of copies.
* System must be able to not allow two books having same book id.

**3.2 Register New Branch**

This feature allows to add new books to the library

Functional requirements:

* System must be able to verify information
* System must be able to not allow two branches having same book id.

**3.3 Issue a Book**

This feature is allows for issuing books.

Functional requirements:

* System must be able to search the database based on select search type
* System must be able to filter book based on keyword entered
* System must be able to check for availability of the book

**3.4 Return a Book**

This feature is used for the return of books.

Functional requirements:

* System must be able to search the database based on select search type
* System must be able to filter book\_loans based on keyword entered
* System must be able to check the due date and actual return date and accordingly display a fine if any.

**3.5 Search a Book**

This feature is used for the return of books.

Functional requirements:

* System must be able to search the database based on select search type
* System must be able to filter book based on keyword entered
* System must display all branches and number of copies available at each branch for each book.

­

**Chapter 4**

**Detailed Design**

**4.1 ER Diagram**

Diagram

Description automatically generated

**4.2 Schema Diagram**

**Book(** BookId, Title)

**Book\_Authors**( BookId, Fname, Mname, Lname)

BookId References Book

**Copies(** BookId, Branch\_Id, No\_Of\_Copies)

Book\_id References Book

Branch\_Id References Library\_Branch

**Library\_Branch(** Branch\_Id, Branch\_Name, Address)

**Librarian(** EmployeeId, Branch\_Id, Reports\_To, Fname, Mname, Lname, Salary)

Branch\_Id References Library\_Branch

Reports\_To References Librarian

**Borrower (**Card\_No, Fname, Mname, Lname, Address, Phone)

**Book\_Loans (**Loan\_Id, BookId, Branch\_Id, Card\_No, Date\_Issued, Return\_date, Date\_In, Issued\_By\_EmpId)

BookId References Book

Branch\_Id References Library\_Branch

Card\_No References Borrower

Issued\_By\_EmpId References Librarian

Diagram

Description automatically generated

**4.3 Data Dictionary**

**BOOK**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| BookId | number(5) | Primary key | PK\_BOOKID |
| Title | varchar2(50) |  |  |

**BOOK\_AUTHORS**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| BookId | number(5) | Foreign Key referencing Book,  On delete cascade | FK\_Ref\_BID |
| Fname | varchar2(25) |  |  |
| Mname | varchar2(25) |  |  |
| Lname | varchar2(25) |  |  |

**LIBRARY\_BRANCH**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| Branch\_Id | number(5) | Primary key | PK\_Branch\_ID |
| Branch\_Name | varchar2(50) |  |  |
| Address | varchar2(100) |  |  |

**COPIES**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| BookId | number(5) | Foreign Key referencing Book, On delete cascade | FK\_BID\_COPIES |
| Branch\_Id | number(5) | Foreign Key referencing Library\_Branch, on delete cascade | FK\_Branch\_COPIES |
| No\_of\_Copies | Number(3) |  |  |

**BORROWER**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| Card\_No | number(5) | Primary key | PK\_CARD |
| Fname | varchar2(25) |  |  |
| Mname | varchar2(25) |  |  |
| Lname | varchar2(25) |  |  |
| Address | varchar2(100) |  |  |
| Phone | char(13) | Valid if country code: '+91', '+81', '+31', '+48', '+44’ | VALID\_PHONE |

**BOOK\_LOANS**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| Loan\_Id | number(6) | unique | UNQ\_LID |
| BookId | number(5) | Foreign key references Book, On delete cascade | FK\_BID\_LOANS |
| Branch\_Id | number(5) | Foreign key references Library\_Branch, On delete cascade | FK\_Branch\_LOANS |
| Card\_No | number(5) | Foreign key references Borrower, on delete set null | FK\_CNo |
| Date\_Issued | Date |  |  |
| Return\_Date | Date | >Date\_Issued | VALID\_RETURN\_DATE |
| Date\_In | Date |  |  |
| Issued\_By\_Empid | number(5) | Foreign key references Librarian, On delete set null | FK\_Librarian\_Loans |

**LIBRARIAN**

|  |  |  |  |
| --- | --- | --- | --- |
| *Column* | *Data Type (size)* | *Constraint* | *Constraint Name* |
| EmployeeId | number(5) | Primary key | PK\_EID |
| Branch\_Id | number(5) | Foreign key references Library\_Branch, On delete set null | FK\_BRANCHID\_librarian |
| Reports\_To | number(5) | Foreign key references  Librarian, on delete set null | FK\_EmId\_Librarian |
| Fname | Varchar2(25) |  |  |
| Mname | varchar2(25) |  |  |
| Lname | varchar2(25) |  |  |
| Salary | Number(6) |  |  |

**4.4 Relational Model Implementation**

**CREATE TABLE Book**( BookId number(5) constraint PK\_BOOKID primary key, Title varchar2(50) );

**CREATE TABLE Book\_Authors**( BookId number(5) constraint FK\_Ref\_BID references Book On Delete Cascade, Fname varchar2(25), Mname varchar2(25), Lname varchar2(25) );

**CREATE TABLE Library\_Branch**( Branch\_ID number(5) constraint PK\_Branch\_ID primary key, Branch\_Name varchar2(50), Address varchar2(100) );

**CREATE TABLE Copies**( BookID number(5) constraint FK\_BID\_COPIES references Book(BookID) On delete Cascade, Branch\_ID number(5) constraint FK\_Branch\_COPIES references Library\_Branch(Branch\_ID) On delete Cascade, No\_of\_Copies number(3) );

**CREATE TABLE Borrower**( Card\_No number(5) constraint PK\_CARD primary key, Fname varchar2(25), Mname varchar2(25), Lname varchar2(25), Address varchar2(100), Phone char(13) constraint VALID\_PHONE check (Phone Like '+91%' or Phone Like '+81%' or Phone Like '+31%' or Phone Like '+48%' or Phone Like '+44%') );

**CREATE TABLE BOOK\_LOANS**( Loan\_ID number(6) constraint UNQ\_LID unique, BookID number(5) constraint FK\_BID\_LOANS references Book On delete cascade, Branch\_ID number(5) constraint FK\_Branch\_LOANS references Library\_Branch On delete cascade, Card\_No number(5) constraint FK\_CNo references Borrower On delete set null, Date\_Issued Date, Return\_Date Date, Date\_In Date, Issued\_By\_EmpId number(5) constraint FK\_Librarian\_Loans references Librarian On delete set null);

**CREATE TABLE librarian**( EmployeeId number(5) constraint PK\_EID primary key, branch\_ID number(5) constraint FK\_BRANCHID\_librarian references Library\_Branch On delete set null, reports\_to number(5) constraint FK\_EmpId\_Librarian references Librarian on delete set null, fname varchar2(25), mname varchar2(25), Lname varchar2(25), salary number(6) );

**4.5 Queries**

List of queries used to retrieve data

**4.5.1** select count(\*) into v\_numbooksborrowed from book\_loans where card\_no = :new.card\_no and Date\_in = NULL;

**4.5.2** select return\_date into due from book\_loans where loan\_id = l\_id;

**4.5.3** select c.no\_of\_copies into V\_copies.no\_of\_copies from copies c, Librarian L

where c.bookid = bookid and L.employeeId = empid and L.branch\_Id = c.branch\_id;

**4.5.4** select \* into v\_lrow from Librarian where EmployeeId = empid;

**4.5.5** SELECT return\_date INTO retdate FROM BOOK\_LOANS WHERE LOAN\_ID=L\_ID;

**4.5.6** select \* from Copies where bookid = bid;

**4.7 Triggers**

1) Availability

CREATE or REPLACE TRIGGER availability

Before Insert or Update on Book\_Loans

For each row

Declare

no\_books exception;

Begin

case

when inserting Then

if checkavailability(:new.bookId, :new.branch\_id) = False then

raise no\_books;

end if;

Update Copies

set No\_of\_Copies = No\_of\_Copies-1

where Copies.BookId=:new.BookId and Copies.Branch\_id = :new.branch\_id;

when Updating Then

UPDATE COPIES

SET NO\_OF\_COPIES = NO\_OF\_COPIES +1

WHERE BOOKID =:OLD.BOOKID AND Branch\_ID = :new.branch\_id;

End case;

Exception

when no\_books then

raise\_application\_error(-20111, 'No copies Available');

end;

/

2)Prevents\_on\_borrows

Create or Replace trigger Prevent\_on\_borrows

before insert on book\_loans

For each row

declare

prevent\_borrow exception;

v\_numbooksborrowed number(2);

begin

select count(\*) into v\_numbooksborrowed from book\_loans where card\_no = :new.card\_no and Date\_in = NULL;

if v\_numbooksborrowed = 5 then

raise prevent\_borrow;

end if;

exception

when prevent\_borrow then

dbms\_output.put\_line('Return books before borrowing more');

End;

/

**4.8 Stored Procedures**

1) Return Book

create or replace procedure returnbook(B\_NO IN NUMBER, L\_ID IN NUMBER, indate in date )

as

fine number(20);

due date;

begin

select return\_date into due from book\_loans where loan\_id = l\_id;

if due < indate then

fine := Fine\_amount(B\_NO , L\_ID , indate );

dbms\_output.put\_line('Fine: '||fine);

else

UPDATE BOOK\_LOANS SET Date\_in = indate WHERE LOAN\_ID=L\_ID;

end if;

end;

/

2)Addbook

Create or replace procedure AddBook(V\_bookid in number,

V\_title in book.title%type, V\_authorfname in book\_authors.fname%type,

V\_authormname in book\_authors.mname%type, V\_authorlname in book\_authors.lname%type) As

Begin

Insert into Book values(V\_bookid, V\_title);

Insert into Book\_Authors values(V\_bookid, V\_authorfname, V\_authormname, V\_authorlname);

End;

/

2) AddLibrarybranch

Create or replace procedure AddBranch(

V\_branchid Library\_Branch.Branch\_Id%type,

V\_branchname Library\_Branch.branch\_name%type,

V\_address Library\_Branch.address%type)

AS

Begin

Insert into Library\_Branch values(V\_branchid, V\_branchname, V\_address);

End;

/

3)Add Librarian

Create or replace procedure AddLibrarian (

V\_empid in Librarian.EmployeeId%type,

V\_branchid in Librarian.Branch\_Id%type,

V\_reportsto in Librarian.EmployeeId%type,

V\_fname in Librarian.fname%type,

V\_mname in Librarian.mname%type,

V\_lname in Librarian.lname%type,

V\_salary in Librarian.salary%type)

AS

Begin

Insert into Librarian values(V\_empid, V\_branchid, V\_reportsto, V\_fname, V\_mname, V\_lname, V\_salary);

End;

/

4)Add Borrower

CREATE or REPLACE PROCEDURE addBorrower(

v\_Card\_No in BORROWER.Card\_no%type,

v\_Fname in BORROWER.Fname%type,

v\_Mname in BORROWER.Mname%type,

v\_Lname in BORROWER.Lname%type,

v\_address in BORROWER.Address%type,

v\_Phone in BORROWER.Phone%type)

AS

Begin

Insert into Borrower values(V\_Card\_No, v\_Fname, v\_Mname, v\_Lname, v\_address, v\_Phone);

End;

/

5)Add Availability

Create or replace procedure AddAvailability(

V\_bookid copies.bookid%type,

V\_branchid copies.Branch\_Id%type,

V\_copies copies.no\_of\_copies%type)

AS

Begin

Insert into copies values(V\_bookid, V\_branchid, V\_copies);

End;

/

6)Issue books

CREATE or REPLACE PROCEDURE issueBook(

v\_Loan\_Id in Book\_loans.Loan\_Id%type,

v\_BookId in Book\_loans.BookId%type,

V\_branchid in Library\_Branch.Branch\_id%type,

v\_Card\_no in Book\_loans.Card\_no%type,

v\_Date\_Issued in Book\_loans.Date\_Issued%type,

v\_Issued\_by\_EmpId in Book\_loans.Issued\_by\_EmpId%type)

AS

Begin

Insert into Book\_Loans values(v\_Loan\_Id, v\_BookId, v\_branchid, v\_Card\_no,

v\_Date\_Issued, v\_date\_issued + (2\*7), NULL, v\_Issued\_by\_EmpId);

End;

/

7) Search Availability

create or replace procedure searchavailability(bid in number) as

Cursor c is select \* from Copies where bookid = bid;

begin

For i in c

loop

Dbms\_output.put\_line('Bookid:'||I.bookid);

Dbms\_output.put\_line('Branchid:'||I.branch\_id);

Dbms\_output.put\_line('Number of copies :'||I.no\_of\_copies);

end loop;

end;

/

**4.9 Stored Functions**

1) Check availability

create or replace function checkavailability( bookid in number, branchid in number)

return boolean as

cursor c is select c.no\_of\_copies from copies c

where c.bookid = bookid and c.branch\_id = branchid;

V\_copies copies.no\_of\_copies%type;

flag number(1) := 0;

begin

for i in c loop

V\_copies := i.no\_of\_copies;

if V\_copies > 0 then

flag := 1;

end if;

end loop;

if flag = 1 then return true;

else

return false;

end if;

end checkavailability;

/2) return branch id

create or replace function returnbranchid (empid in Librarian.EmployeeId%type)

return Librarian.branch\_id%type

as

v\_lrow Librarian%rowtype;

begin

select \* into v\_lrow from Librarian where EmployeeId = empid;

return v\_lrow.branch\_id;

end;

/

3)Fine amount

CREATE OR REPLACE FUNCTION FINE\_AMOUNT(B\_NO IN NUMBER,L\_ID IN NUMBER, indate in date)

RETURN NUMBER IS

FINE NUMBER(20);

retdate date;

BEGIN

UPDATE BOOK\_LOANS SET Date\_in = indate WHERE LOAN\_ID=L\_ID;

SELECT return\_date INTO retdate FROM BOOK\_LOANS WHERE LOAN\_ID=L\_ID;

FINE:=(TO\_DATE(indate, 'DD-MM-YYYY') - TO\_DATE(retdate, 'DD-MM-YYYY') )\*5;

RETURN FINE;

END FINE\_AMOUNT;

/

**5. Functional Requirement Implementation**

None such requirements

**6. Result**

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Above are screenshots which show the working of all the insertion functions, triggers working and not raising any errors along with the fine amount, checkavailabitlity functions working properly;

**7.** **Conclusion and Future Work**

**7.1. Conclusion**

The Database Library System is intended to Automate the library activities such as creating a new borrower, giving books to the borrowers, maintaining the details of all the item that were available in the books. This also helps the librarians by providing information such as total copies available each book, list of books that belong to a particular category (Short, Long Loan, Reference items, etc).

**7.2. Scope for Future Work**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system area as the technology emerges; it is possible to upgrade the system to make it adaptable to the desired environment. Based on the future security issues, security can be improved using emerging technologies and a sub admin module can be added. Thus, a better demarcation or distinction between different types of users of the system can be added. Not only that, a feature for requesting the addition of books to the library can also be added to the database. A user interface can also be developed for a much more user-friendly application.