BVRIT HYDERABAD

College of Engineering for Women

Department of CSE(AIML)

CASE STUDY I

LIBRARY MANAGEMENT SYSTEM

TEAM 1

G. Nikhita - 22WH1A6601

G. Revathi - 22WH1A6606

G. Tejaswini - 22WH1A6608

N. Divya Saahithya - 22WH1A6616

N. Keerthi - 22WH1A6629

N. Jijnasa - 22WH1A6635

A. Prashanthi - 22WH1A6638

K. Vijaya Rajasree22WH1A6648

K. Sreeja - 22WH1A6657

P. Harshini - 22WH1A6660

Under the guidance of

Ms V. Asha(Asst.Professor)

INDEX

S.no	Content	Pg.No
1	Introduction	3
2	Requirement Analysis	4-7
3	Database Design	8
4	Database Implementation	9-14
5	Testing and Results	15-21

INTRODUCTION

A Library Management System (LMS) is a software application that automates and manages the operations of a library. It simplifies tasks such as cataloging, circulation, inventory management, and user management. By providing features like catalog organization, book loan management, and user access control, an LMS enhances efficiency, improves user experience, and enables libraries to maintain accurate records and data analytics for informed decision-making. Overall, it transforms traditional library functions into streamlined, accessible processes that benefit both staff and library patrons.

Benefits:

- Efficiency: Automates routine tasks, reducing manual effort and errors.
- Accessibility: Provides easy access to library resources for users and staff.
- Enhanced User Experience: Improves user satisfaction through streamlined processes and services.
- Inventory Control: Facilitates accurate tracking and management of library collections.
- Data-Driven Decisions: Provides insights through data analytics and reporting.

REQUIREMENT ANALYSIS

1. Introduction

The Library Management System is designed to help libraries manage their collections of books and track borrowing activities. It includes functionalities for managing books, authors, genres, borrowers, and transactions.

2. Functional Requirements

Functional requirements describe the interactions between the system and its users, and the system's behavior in response to user inputs.

- **User Management**
  - **Register Borrowers:** The system should allow registering new borrowers with details such as first name, last name, email, and phone number.
  - **Update Borrower Information:** The system should allow updating borrower details.
  - **View Borrower Details:** The system should allow viewing details of registered borrowers.
- **Book Management**
  - **Add New Books**: The system should allow adding new books with details such as title, author, genre, publication year, and copies available.
  - **Update Book Details:** The system should allow updating details of existing books.
  - **View Book Details:** The system should allow viewing details of books**.**
  - **Delete Books**: The system should allow deleting books from the inventory.

- **Author Management**
  - **Add New Authors:** The system should allow adding new authors with details such as name and bio.

- **Update Author Details:** The system should allow updating author details.
- **View Author Details:** The system should allow viewing details of authors.

➢ **Genre Management**

- **Add New Genres**: The system should allow adding new genres with details such as name and description.
- **Update Genre Details**: The system should allow updating genre details.
- **View Genre Details:** The system should allow viewing details of genres.

➢ **Transaction Management**

- **Issue Books:** The system should allow issuing books to borrowers with details such as borrower ID, book ID, issue date, and due date.
- **Return Books**: The system should allow recording the return of books with details such as return date and status.
- **View Transactions:** The system should allow viewing transaction details including borrowed, returned, and overdue books.

➢ **Reports and Queries**

- **Generate Reports:** The system should generate reports on book availability, borrowed books, overdue books, and borrower histories.
- **Execute Queries**: The system should support executing predefined queries to retrieve specific information, such as the most borrowed book or books that have never been borrowed.

3. Non-Functional Requirements

Non-functional requirements define the quality attributes of the system, including performance, usability, and security.

➢ **Performance** - The system should respond to user queries and actions within 2 seconds. The system should handle concurrent access by multiple users without significant performance degradation.

- **Usability** - The system should have an intuitive user interface that is easy to navigate. The system should provide clear error messages and guidance for correcting user input errors.
- **Security -** The system should ensure that only authorized personnel can perform administrative actions such as adding or deleting books and authors. User data should be stored securely to prevent unauthorized access.
- **Reliability -** The system should be available 99% of the time during library hours. The system should provide regular backups to prevent data loss.
- **Maintainability -** The system should be designed in a modular way to allow easy updates and maintenance. The code should be well-documented to facilitate future enhancements.

4. Use Cases

Use cases describe the interactions between a user (actor) and the system to achieve a specific goal.

Register Borrower

- **Actor:** Librarian
- **Goal:** Register a new borrower in the system.
- **Steps:**

1. Librarian enters borrower details.
2. System validates and stores the details.
3. System confirms the registration**.**

Add Book

- **Actor:** Librarian
- **Goal:** Add a new book to the library inventory**.**
- **Steps:**

1. Librarian enters book details.
2. System validates and stores the details.
3. System confirms the addition.

Issue Book

- **Actor**: Librarian

- **Goal**: Issue a book to a borrower.
- **Steps:**

1. Librarian enters borrower ID and book ID.
2. System validates the details and updates the transaction.
3. System confirms the issuance.

Return Book

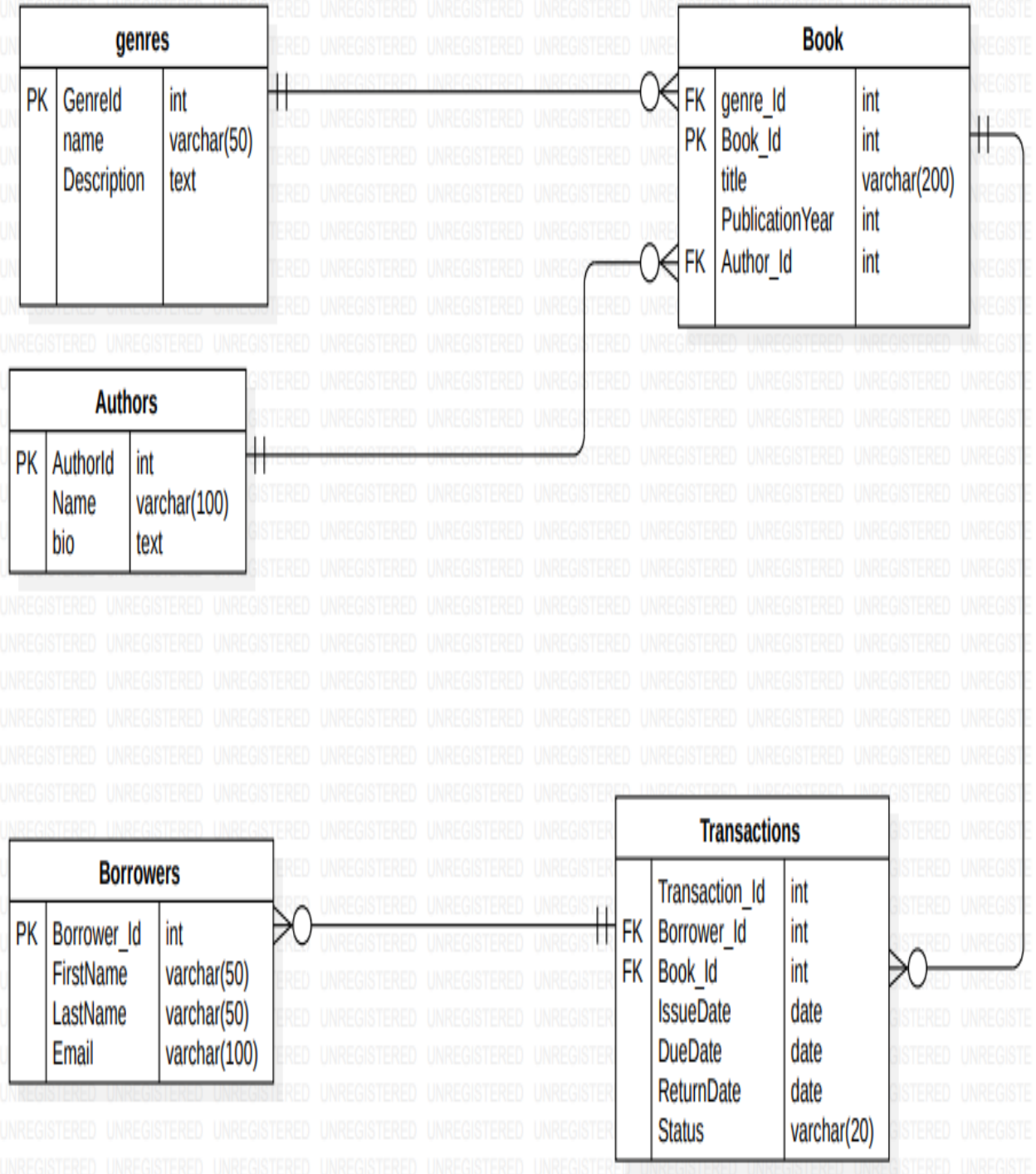
- **Actor:** Librarian
- **Goal**: Record the return of a borrowed book.
- **Steps:**

1. Librarian enters transaction ID and return date.
2. System updates the transaction status.
3. System confirms the return.

5. User Stories

- User stories are short, simple descriptions of a feature told from the perspective of the user.
- As a librarian, I want to register new borrowers so that they can borrow books from the library.
- As a librarian, I want to add new books to the inventory so that the library's collection is updated.
- As a borrower, I want to search for books by title, author, or genre so that I can find books of interest.
- As a librarian, I want to issue books to borrowers so that they can read them at home.
- As a librarian, I want to record the return of books so that the inventory is accurate.
- As a librarian, I want to view reports on overdue books so that I can follow up with borrowers.
- As a borrower, I want to view my borrowing history so that I can keep track of the books I have read.

DATABASE DESIGN



DATABASE IMPLEMENTATION

Creation of Books table

```
CREATE TABLE Books (
    BookID INT AUTO_INCREMENT PRIMARY KEY,
    Title VARCHAR(200) NOT NULL,
    AuthorID INT,
    GenreID INT,
    PublicationYear INT,`
    CopiesAvailable INT DEFAULT 0,
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID),
    FOREIGN KEY (GenreID) REFERENCES Genres(GenreID)
);


INSERT INTO Books (Title, AuthorID, GenreID, PublicationYear,
CopiesAvailable) VALUES
('Harry Potter', 1, 1,  1997, 5),
('A Game of Thrones', 2, 1, 1996, 3),
('Naruto vol2', 4, 4, 2010, 6),
('Demon slayer vol1',5,4,2000,10),
('The Alchemist',3,5,2005,8),
('Age of vice',2,3,2015,11),
('The Cruel Prince',5,5,1998,15),
('Oliver Twist',9,5,1990,10),
```

('Origin Of species',10,3,2000,4),

('Malgudi Days',7,4,1995,20);

SELECT* FROM Books;

BookID	Title	AuthorID	GenreID	PublicationYear	CopiesAvailable
1	Harry Potter	1	1	1997	5
2	A Game of Thrones	2	1	1996	3
3	Naruto vol2	4	4	2010	6
4	Demon slayer vol1	5	4	2000	10
5	The Alchemist	3	5	2005	8
6	Age of vice	2	3	2015	11
7	The Cruel Prince	5	5	1998	15
8	Oliver Twist	9	5	1990	10
9	Origin Of species	10	3	2000	4
10	Malgudi Days	7	4	1995	20

Creation of Borrowers table

```
create table Borrowers (
    Borrower_ID int primary key,
    FirstName varchar(50),
    LastName varchar(50),
    Email varchar(100),
    Phone int
);


insert into Borrowers  values
(101, 'John', 'Doe', 'john@gmail.com', 1234567890),
(102, 'Jane', 'Smith', 'jane@gmail.com', 2345678901),
(103, 'Emily', 'Jones', 'emily@gmail.com', 3456789012),
(104, 'Michael', 'Brown', 'michael@gmail.com', 4567890123),
```

(105, 'Jessica', 'Williams', 'jessica@gmail.com', 5678901234),

(106, 'Daniel', 'Taylor', 'daniel@gmail.com', 6789012345),

(107, 'Sarah', 'Miller', 'sarah@gmail.com', 7890123456),

(108, 'David', 'Wilson', 'david@gmail.com', 8901234567);

SELECT* FROM Borrowers;

Borrower_ID	FirstName	LastName	Email	Phone
101	John	Doe	john@gmail.com	1234567890
102	Jane	Smith	jane@gmail.com	2345678901
103	Emily	Jones	emily@gmail.com	3456789012
104	Michael	Brown	michael@gmail.com	4567890123
105	Jessica	Williams	jessica@gmail.com	5678901234
106	Daniel	Taylor	daniel@gmail.com	6789012345
107	Sarah	Miller	sarah@gmail.com	7890123456
108	David	Wilson	david@gmail.com	8901234567

Creation of Transactions table

CREATE TABLE Transactions(

TransactionID INT AUTO_INCREMENT PRIMARY KEY,

Borrower_ID INT,

BookID INT,

IssueDate DATE NOT NULL,

DueDate DATE NOT NULL,

ReturnDate DATE,

Status VARCHAR(20) CHECK (Status IN ('Borrowed','Returned','Overdue')),

FOREIGN KEY (Borrower_ID) REFERENCES Borrowers(Borrower_ID),

```
    FOREIGN KEY (BookID) REFERENCES Books(BookID)
    );
INSERT INTO Transactions (Borrower_ID, BookID, IssueDate, DueDate,
ReturnDate, Status) VALUES
    (101, 1, '2023-06-01', '2023-06-15', NULL, 'Borrowed'),
    (102, 2, '2023-06-02', '2023-06-16', '2023-06-10', 'Returned'),
    (103, 3, '2023-06-03', '2023-06-17', NULL, 'Borrowed'),
    (104, 4, '2023-06-04', '2023-06-18', NULL, 'Overdue'),
    (105, 5, '2023-06-05', '2023-06-19', '2023-06-18', 'Returned'),
    (106, 6, '2023-06-06', '2023-06-20', NULL, 'Borrowed'),
    (107, 7, '2023-06-07', '2023-06-21', '2023-06-20', 'Returned'),
    (108, 8, '2023-06-08', '2023-06-22', NULL, 'Borrowed');
SELECT* FROM Transactions;
```

TransactionID	Borrower_ID	BookID	IssueDate	DueDate	ReturnDate	Status
9	101	1	2023-06-01	2023-06-15	NULL	Borrowed
10	102	2	2023-06-02	2023-06-16	2023-06-10	Returned
11	103	3	2023-06-03	2023-06-17	NULL	Borrowed
12	104	4	2023-06-04	2023-06-18	NULL	Overdue
13	105	5	2023-06-05	2023-06-19	2023-06-18	Returned
14	106	6	2023-06-06	2023-06-20	NULL	Borrowed
15	107	7	2023-06-07	2023-06-21	2023-06-20	Returned
16	108	8	2023-06-08	2023-06-22	NULL	Borrowed

Creation of Authors table

```
CREATE TABLE Authors (
    AuthorID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(100) NOT NULL,
    Bio TEXT
```

);

insert into Authors values(1,"Dallas Clayton","American author"),(2,"Demi","British author"),(3,"Arundhati Roy","American novelist"),(4,"Vanshika Mishra","British author"),(5,"Aravind Adiga","Indian novelist"),(6,"Jhumpa Lahiri","British novelist"),(7,"R.K.Narayan","Indian author"),(8,"RabindraNathTagore","Indian author"),(9,"Charles Dickens","American author"),(10,"Charles Darwin","British author");

select * from Authors;

AuthorID	Name	Bio
1	Dallas Clayton	American author
2	Demi	British author
3	Arundhati Roy	American novelist
4	Vanshika Mishra	British author
5	Aravind Adiga	Indian novelist
6	Jhumpa Lahiri	British novelist
7	R.K.Narayan	Indian author
8	RabindraNathTagore	Indian author
9	Charles Dickens	American author
10	Charles Darwin	British author

Creation of Genres table

CREATE TABLE Genres (

GenreID INT AUTO_INCREMENT PRIMARY KEY,

Name VARCHAR(50) NOT NULL,

Description TEXT

);

INSERT INTO Genres (Name, Description) VALUES

('Fantasy', 'Fiction with magical elements'),

('Science Fiction', 'Fiction with futuristic elements'),

('Mystery','Uncovering secrets'),

('Picture Books','Engaging visuals'),

('Thriller','Psychological drama');

select * from Genres;

GenreID	Name	Description
1	Fantasy	Fiction with magical elements
2	Science Fiction	Fiction with futuristic elements
3	Mystery	Uncovering secrets
4	Picture Books	Engaging visuals
5	Thriller	Psychological drama

TESTING AND RESULTS

1. List all books with their genres.

```
SELECT b.Title, g.Name AS Genre
FROM Books b
JOIN Genres g ON b.GenreID = g.GenreID;
```

OUTPUT:

Title	Genre
Harry Potter	Fantasy
A Game of Thrones	Fantasy
Age of vice	Mystery
Origin Of species	Mystery
Naruto vol2	Picture Books
Demon slayer vol1	Picture Books
Malgudi Days	Picture Books
The Alchemist	Thriller
The Cruel Prince	Thriller
Oliver Twist	Thriller

2. Find the total number of copies available for each genre.

```
SELECT g.Name AS Genre, SUM(b.CopiesAvailable) AS TotalCopies
FROM Books b
JOIN Genres g ON b.GenreID = g.GenreID
GROUP BY g.Name;
```

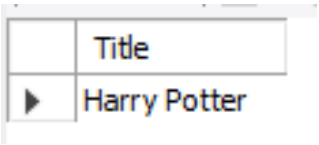
OUTPUT:

Genre	TotalCopies
Fantasy	8
Picture Books	36
Thriller	33
Mystery	15

3. List all books borrowed by 'John Doe'.

```
SELECT b.Title
FROM Transactions t
JOIN Books b ON t.BookID = b.BookID
JOIN Borrowers br ON t.Borrower_ID = br.Borrower_ID
WHERE br.FirstName = 'John' AND br.LastName = 'Doe';
```

OUTPUT:



	Title
▶	Harry Potter

4. List overdue books with borrower details.

```
SELECT b.Title, br.FirstName, br.LastName, t.DueDate
FROM Transactions t
JOIN Books b ON t.BookID = b.BookID
JOIN Borrowers br ON t.Borrower_ID = br.Borrower_ID
WHERE t.Status = 'Overdue';
```

OUTPUT:

	Title	FirstName	LastName	DueDate
▶	Demon slayer vol1	Michael	Brown	2023-06-18

5. Find the most borrowed book.

```
SELECT b.Title, COUNT(t.TransactionID) AS BorrowCount
FROM Transactions t
JOIN Books b ON t.BookID = b.BookID
GROUP BY b.Title
ORDER BY BorrowCount DESC
LIMIT 1;
```

OUTPUT:

	Title	BorrowCount
▶	Harry Potter	1

6. List authors who have written more than one book.

```
SELECT a.Name, COUNT(b.BookID) AS BookCount
FROM Authors a
JOIN Books b ON a.AuthorID = b.AuthorID
GROUP BY a.Name
HAVING COUNT(b.BookID) > 1;
```

OUTPUT:

	Name	BookCount
▶	Demi	2
	Aravind Adiga	2

7. Find books published before the year 2000.

```
SELECT Title, PublicationYear
FROM Books
WHERE PublicationYear < 2000;
```

OUTPUT:

	Title	PublicationYear
▶	Harry Potter	1997
	A Game of Thrones	1996
	The Cruel Prince	1998
	Oliver Twist	1990
	Malgudi Days	1995

8. Count the number of books in each status (Borrowed, Returned, Overdue).

```
SELECT Status, COUNT(TransactionID) AS StatusCount
FROM Transactions
GROUP BY Status;
```

OUTPUT:

	Status	StatusCount
▶	Borrowed	4
	Returned	3
	Overdue	1

9. Find the author of the book 'The Alchemist'.

```
SELECT a.Name
FROM Authors a
JOIN Books b ON a.AuthorID = b.AuthorID
WHERE b.Title = 'The Alchemist';
```

OUTPUT:

	Name
▶	Arundhati Roy

10. List all books along with the borrower's name if borrowed.

```
SELECT b.Title, br.FirstName, br.LastName

FROM Books b

LEFT JOIN Transactions t ON b.BookID = t.BookID

LEFT JOIN Borrowers br ON t.Borrower_ID = br.Borrower_ID;
```

OUTPUT:

Title	FirstName	LastName
Harry Potter	John	Doe
A Game of Thrones	Jane	Smith
Naruto vol2	Emily	Jones
Demon slayer vol1	Michael	Brown
The Alchemist	Jessica	Williams
Age of vice	Daniel	Taylor
The Cruel Prince	Sarah	Miller
Oliver Twist	David	Wilson
Origin Of species	NULL	NULL
Malgudi Days	NULL	NULL

11. Find the book with the most available copies.

```
SELECT Title, CopiesAvailable

FROM Books

ORDER BY CopiesAvailable DESC

LIMIT 1;
```

OUTPUT:

Title	CopiesAvailable
Malgudi Days	20

12. List all authors from India.

```
SELECT Name

FROM Authors

WHERE Bio LIKE '%Indian%';
```

OUTPUT:

	Name
▶	Aravind Adiga
	R.K.Narayan
	RabindraNathTagore

13. Find the books that have never been borrowed.

```
SELECT b.Title

FROM Books b

LEFT JOIN Transactions t ON b.BookID = t.BookID

WHERE t.TransactionID IS NULL;
```

OUTPUT:

	Title
▶	Origin Of species
	Malgudi Days

14. Find the number of books written by each author.

```
SELECT a.Name, COUNT(b.BookID) AS BookCount

FROM Authors a

LEFT JOIN Books b ON a.AuthorID = b.AuthorID

GROUP BY a.Name;
```

OUTPUT:

Name	BookCount
Dallas Clayton	1
Demi	2
Arundhati Roy	1
Vanshika Mishra	1
Aravind Adiga	2
Jhumpa Lahiri	0
R.K.Narayan	1
RabindraNathTagore	0
Charles Dickens	1
Charles Darwin	1

15. List all books along with their authors and genres.

```
SELECT b.Title, a.Name AS Author, g.Name AS Genre
FROM Books b
JOIN Authors a ON b.AuthorID = a.AuthorID
JOIN Genres g ON b.GenreID = g.GenreID;
```

OUTPUT:

Title	Author	Genre
Harry Potter	Dallas Clayton	Fantasy
A Game of Thrones	Demi	Fantasy
Naruto vol2	Vanshika Mishra	Picture Books
Demon slayer vol1	Aravind Adiga	Picture Books
The Alchemist	Arundhati Roy	Thriller
Age of vice	Demi	Mystery
The Cruel Prince	Aravind Adiga	Thriller
Oliver Twist	Charles Dickens	Thriller
Origin Of species	Charles Darwin	Mystery
Malgudi Days	R.K.Narayan	Picture Books