JAVA PROJECT ON

" LIBRARY MANAGEMENT SYSTEM "

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

BY

Section- K21RB

Gyanendra Nath Shukla	12113311	Database, return book & CSV File
Ayush Kumar	12106375	Admin,User Login & Terminate
Roshan Kumar	12106308	Add,Issue & Search Book

SUBMITTED TO:

Dr.Bhimsen Moharana Sir (28066) ASST. PROFESSOR

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

LOVELY PROFESSIONAL UNIVERSITY

Phagwara, Punjab (India)

-: ABSTRACT :-

A library management system is a software application that helps libraries to manage their resources, such as add books, delete books, issue books, return books, search for book and other materials. This project was developed in Java programming language with a database backend to provide an efficient and reliable solution to manage library resources.

The system allows librarians to manage the entire library, including user information, book information, and managing all the books. The user can search for books, check the availability of the book, reserve a book, and issue a book. This also includes manager view who can terminate librarian from job and add books.

The database used in this project is SQLite, which is a widely used open-source database management system. The system uses JDBC (Java Database Connectivity) to establish a connection between Java and the database.

The system's interface is user-friendly and straightforward, making it easy for librarians to use. The system's features are well-organized, and the system can handle large amounts of data.

The system file include a batch file which makes project more user-friendly to run on a single click. It runs the file in terminal on one click on the file, as to handle efficiency and least delay during the run time.

Overall, this project provides an efficient and reliable solution for managing library resources and is a useful tool for libraries of any size.

Existing System:

The existing system of library management mostly involves manual methods, including maintaining a paper-based record of books, borrowers, and transactions. This manual system is time-consuming, prone to errors, and inefficient. It is difficult for librarians to manage many books and borrowers, and the chances of losing data are high.

It is very difficult to manage lots of data in pages. According to technology, it is very offensive way to manage. Organisations need a greater number of librarians to manage this task. Librarians need to check the paper/register, most of the time for book return. There is chance of data inconsistency while writing.

Proposed System:

The proposed system for library management is a computerized system that automates various tasks involved in managing a library. The proposed system is built on the Java programming language and uses a database management system to store data. The system is designed to be user-friendly, efficient, and reliable, providing librarians with the tools they need to manage library resources effectively.

The proposed system is capable of performing various tasks such as book search, issue books, return books, add books, delete books and other manager works. The system also generates reports that librarians can use to monitor the library's performance.

The proposed system is scalable, meaning that it can grow and adapt to meet the changing needs of the library. The system is also designed to handle large amounts of data, making it suitable for libraries of any size.

In summary, the proposed system is an essential tool for libraries that want to manage their resources effectively. It is a reliable, efficient, and user-

friendly system that provides librarians with the tools they need to manage their library's resources effectively.

Advantages of Library Management System:

The library management system project has several advantages, some of which are:

- Improved Efficiency: The proposed system is designed to automate various library management tasks, reducing the time and effort required to perform these tasks manually. This will improve the efficiency of the library and allow librarians to focus on more critical tasks.

- Accurate Record-keeping: The computerized system will maintain accurate records of all library resources, including books, borrowers, and transactions. This will reduce the chances of errors and inconsistencies that are likely to occur in a manual system.

- Easy Access: The proposed system is designed to be user-friendly and accessible to all authorized library staff. This will enable librarians to access information quickly and efficiently, improving the quality of service they provide to library users.

- Better Resource Management: The system will enable librarians to monitor and manage library resources more effectively. They can use the generated reports to track the library's performance, including books issued, books returned.

- Scalability: The proposed system is scalable, meaning that it can grow and adapt to meet the changing needs of the library. The system can handle large amounts of data, making it suitable for libraries of any size.

Overall, the library management system project has several advantages that make it a valuable tool for libraries that want to manage their resources effectively. The proposed system is efficient, reliable, user-friendly, and scalable, providing librarians with the tools they need to manage library resources effectively.

System Requirements :

Software Requirement:

- The following are the software requirements for the library management system:
- Operating System: The system is compatible with Windows, Linux, and macOS operating systems.
- Java Development Kit (JDK): The system requires JDK 8 or higher to be installed on the computer.
- Integrated Development Environment (IDE): The system can be developed using any IDE, such as JDSE, NetBeans, or IntelliJ IDEA.
- SQLite Database: The system uses SQLite as the database management system to store data.

Hardware Requirements:

- The following are the hardware requirements for the library management system:

- Processor: The system requires a 1 GHz or higher processor.

- RAM: The system requires a minimum of 2 GB of RAM.

- Hard Disk Space: The system requires a minimum of 100 MB of free hard disk space.

- Display: The system requires a monitor with a minimum resolution of 1024x768.

- Input Devices: The system requires a keyboard and mouse or TouchPad for input.

Overall, the system requirements for the library management system are not very demanding, and most modern computers should meet these requirements without any issues. The software requirements are straightforward, and the hardware requirements are relatively modest, making the system accessible to a wide range of users.

Data Requirements

ENTITIES:

- Book
- Admin
- User

ATTRIBUTES:

- **BOOK**:
  - Title
  - Author
  - Book Id
  - Issue Date
  - Due Date

- **ADMIN**:
  - User_id
  - User Name
  - Address
  - Password

- **User Login**:
  - User_id
  - User_name
  - Password

- **Issued Books**:
  - Book_id
  - Book_name
  - Issue_date
  - Due_date

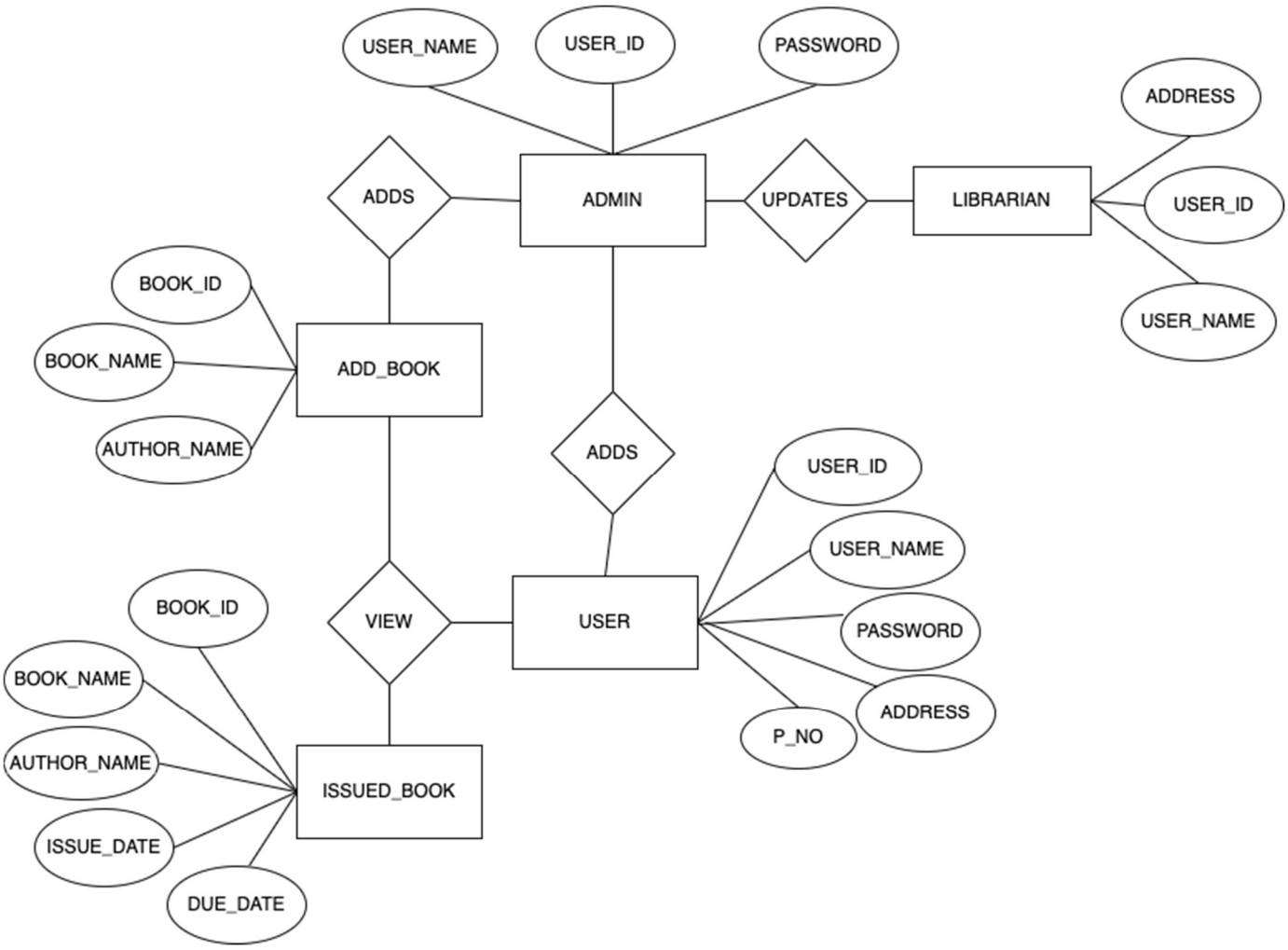
Relationships – Cardinality

- ADMIN adds BOOK
- ADMIN adds User_details
- ADMIN updates librarian_details
- USER view issued_books

Entity-Relationship :

Entity Relationship Diagram is used to model database software engineering to illustrate logical structure of the database. It is a relational schema database modelling approach. This approach commonly used in database design. The diagram created using this method is called E-R Diagram.

The E-R Diagram depicts the various relationships among entities, considering each object as entity. Entities are represented with a rectangle shape and relationships are represented with a diamond shape. It depicts the relationship between data objects. The ER Diagram is the notation that is used to conduct the data modelling activity.



Entity-Relationship Diagram

CODE:

This code is to link/connect to DataBase:

```
import java.sql.Connection; import
java.sql.DatabaseMetaData; import
java.sql.DriverManager; import
java.sql.SQLException; import
java.sql.Statement; import
java.sql.ResultSet; public class
sqlitehandler{
   public static Connection create_database(String database_name){        String
databasepath="jdbc:sqlite:/D:\\javaProject\\database\\"+database_name+".db";

      try {
         Connection conn = DriverManager.getConnection(databasepath);             if (conn !=
null) {                DatabaseMetaData meta = conn.getMetaData();
           System.out.println("The driver name is " + meta.getDriverName());
          //  System.out.println("A new database has been created.");
         }
         return conn;

      } catch (SQLException e) {
         System.out.println(e.getMessage());              return null;
      }
          }        public static ResultSet runStatement(Connection c1,String sqlcommand) {
         // SQLite connection string              try{
           // Connection conn = DriverManager.getConnection(url);
            Statement stmt = c1.createStatement();
            ResultSet rs=stmt.executeQuery(sqlcommand);              System.out.println("command
executed Succesfully");              return rs;
         } catch (SQLException e) {
            System.out.println(e.getMessage());              return null;
         }         }
       public static void main(String args[]){
           Connection c1 =  create_database("anand");
           runStatement(c1,"create table t1(n int, bookname)");         }

}
```

It is used to generate ResultSet and runStatement.

Code for final work:

```
import java.sql.*; import java.util.*; import
java.time.*; import
java.text.SimpleDateFormat;
 public class app extends sqlitehandler{

    public static void addUser(Connection admConnection){
        Scanner sc = new Scanner(System.in);
System.out.println("Enter user ID: ");         int userid = sc.nextInt();
        System.out.println("Enter name: ");
        String name = sc.next();
        System.out.println("Enter address: ");
        String address = sc.next();
        System.out.println("Enter phone no: ");         long phoneno =
sc.nextLong();
        System.out.println("Enter password: ");
        String password = sc.next();
        //Adding to database named Library_Management
        String sql = "insert into user values("+ userid +", '"+ name +"','"+ address
+"','"+ phoneno +"','"+ password +"');";         //System.out.println(sql);
runStatement(admConnection,sql);     sc.close();
    }
    public static void deleteLibrarianInfo(Connection admConnection){
   Scanner sc=new Scanner(System.in);
   System.out.println("Enter the Librarian's User Id whose details you want to delete"); int delLib=sc.nextInt();
sc.close();

   //deleting the user whose employee id is given
   String sql="delete from user where userid = +"+ delLib+ "";    runStatement(admConnection, sql);
}



    public static void addBook(Connection admConnection){
    Scanner sc = new Scanner(System.in);
System.out.println("Enter book id: ");     int bookid =
sc.nextInt();
    sc.nextLine(); // Consume the newline character left behind by nextInt()
    System.out.println("Enter book name: ");
    String bookname = sc.nextLine();

    System.out.println("Enter author name: ");
    String authorname = sc.nextLine();
```

```
            adminLogin(c1);                break;
case 2:                 userlogin(c1);
break;                 default:
            System.out.println("INVALID CHOICE!");
          }
    }

    public static void adminLogin(Connection c1){         Scanner sc = new
Scanner(System.in);
        System.out.println("Enter username: ");
        String username = sc.next();
        System.out.println("Enter your password: ");
        String password = sc.next();
        String sql = "select * from user";         ResultSet out =
runStatement(c1,sql);        try{
          while(out.next()){
            String username1 = out.getString("username");                String password1 =
out.getString("password");
            if(username1.equals(username) && password1.equals(password)){
                System.out.println("WELCOME" + " "+ username);
                System.out.println("You have logged in as admin successfully! ");
                System.out.println();
                System.out.println("press any key to move further...");                    sc.next();
                System.out.println();
                System.out.println("1. Add User Details");
                System.out.println("2. Terminate from job ");
System.out.println("Enter your choice...");                    int choice = sc.nextInt();
switch(choice){                        case 1:
                    addUser(c1);                          break;
case 2:
                    deleteLibrarianInfo(c1);                          break;
default:
                    System.out.println("INVALID CHOICE!");
                }                   return;
            }
          }
          System.out.println("Incorrect username or password");
        }        catch (SQLException e){
          e.printStackTrace();
          System.out.println(e);
        }
    }
```

```
public static void userlogin(Connection c1){
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter username: ");
      String username = sc.next();
      System.out.println("Enter your password: ");
      String password = sc.next();
      String sql = "select * from user";
      ResultSet out = runStatement(c1,sql);
      try{
         while(out.next()){
            String username1 = out.getString("username");              String password1 =
out.getString("password");
            if(username1.equals(username) && password1.equals(password)){
               System.out.println("WELCOME" + " "+ username);
               System.out.println("You have logged in as admin successfully! ");
               System.out.println();
               System.out.println("press any key to move further...");                  sc.next();
               System.out.println();
               System.out.println("1. Add Book Details");
               System.out.println("2. Search books ");                  System.out.println("3. Issue books ");
               System.out.println("4. Return books ");
System.out.println("Enter your choice...");                  int choice = sc.nextInt();
switch(choice){                     case 1:
                     addBook(c1);
break;                        case 2:
                     bookSearch(c1);                        break;
case 3:
                     issueBook(c1);                        break;
case 4:
                     returnBook(c1);                        break;
default:
                     System.out.println("INVALID CHOICE!");                        break;
               }                  return;
            }
         }
         System.out.println("Incorrect username or password");
      }
      catch (SQLException e){
         e.printStackTrace();
         System.out.println(e);
      }
   }
```

//TODO_____ISSUE AND RETURN_____

```
----------------------------------------
   public static void issueBook(Connection c1) {        Scanner sc = new
Scanner(System.in);
       System.out.println("Enter book id: ");
       String bookId = sc.next();
       System.out.println("Enter issue date (yyyy-mm-dd): ");
       String issueDate = sc.next();
       System.out.println("Enter due date (yyyy-mm-dd): ");        String dueDate
= sc.next();     try{
       String check = "Select bookid from bookTable where bookid = '"+ bookId +"' ";        ResultSet checkit =
runStatement(c1,check);        if(checkit.next()){
           String sql = "INSERT INTO issuedbooks (bookId, bookname, authorname,
issueDate, dueDate) SELECT bookId, bookname, authorname, '" + issueDate + "', '" + dueDate
+ "' FROM bookTable WHERE bookId = '" + bookId + "'";
           String sql1 = "DELETE FROM bookTable WHERE bookid='" + bookId + "'";            ResultSet
result = runStatement(c1, sql);            runStatement(c1,sql1);            if (result == null) {
               System.out.println("Book issued successfully!");
           } else {
               System.out.println("Failed to issue book!");
           }
       }else{
           System.out.println("Wrong book id.");
       }    }
catch(SQLException e){
       // e.printStackTrace();
       System.out.println("Error occurred while executing SQL statement: " + e.getMessage());
   }
   }
   public static void returnBook(Connection c1) {
   Scanner sc = new Scanner(System.in);
   System.out.println("Enter book id: ");
   String bookId = sc.next();

   try {
       //fetch bookname and authorname from bookTable using bookId
       String sqlSelect = "select bookname, authorname from issuedbooks where bookid='" + bookId + "'";
       ResultSet rs = runStatement(c1, sqlSelect);
       String bookname = "";        String
authorname = "";        if (rs.next()) {
           bookname = rs.getString("bookname");            authorname =
rs.getString("authorname");
       }
       String check = "select bookid from issuedbooks where bookid = '"+ bookId +"'";
```

```
        // insert book details into bookTable
        String sqlInsert = "insert into bookTable(bookid, bookname, authorname) values
('" + bookId + "','" + bookname + "','" + authorname + "')";            ResultSet insertResult =
runStatement(c1, sqlInsert);            if (insertResult == null) {
            System.out.println("Book details inserted into bookTable!");            } else {
            System.out.println("Failed to insert book details into bookTable!");            }
        //delete book from issued_books
        String sqlDelete = "delete from issuedbooks where bookid='" + bookId + "'";            ResultSet deleteResult =
runStatement(c1, sqlDelete);            if (deleteResult == null) {
            System.out.println("Book returned successfully!");
        } else {
            System.out.println("Failed to return book!");
        }
   }else{
      System.out.println("Wrong book id");
   }
   } catch (SQLException e) {
      System.out.println("Error occurred while executing SQL statement: " + e.getMessage());
   }
}
public static void exportBooksToCSV(Connection admConnection) {
    String sql = "SELECT * FROM bookTable;"; // SQL query to fetch all books details
    ResultSet resultSet = null;
    FileWriter cs = null;

    try {
        resultSet = runStatement(admConnection, sql);
        cs = new FileWriter("books.csv"); // CSV file name

        // Writing CSV file headers
        cs.append("Book ID,Book Name,Author Name");
        cs.append("\n");

        // Writing books details to CSV file
        while (resultSet.next()) {
            int bookId = resultSet.getInt("bookid");
            String bookName = resultSet.getString("bookname");
            String authorName = resultSet.getString("authorname");

            cs.append(String.valueOf(bookId));
            cs.append(",");
            cs.append(bookName);
            cs.append(",");
            cs.append(authorName);
            cs.append("\n");
```

```
        }

        System.out.println("Books details exported to CSV file successfully!");
    } catch (SQLException e) {
        e.printStackTrace();
        System.out.println("Error occurred in SQL statement: " + e.getMessage());
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Error occurred while writing to CSV file: " + e.getMessage());
    } finally {
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (cs != null) {
                cs.flush();
                cs.close();
            }
        } catch (IOException | SQLException e) {
            e.printStackTrace();
        }
    }
}

  public static void main(String args[]){
    Connection c1 =  create_database("Library_Management");
    // Connection c1 =  create_database("library");
    // String sql = "CREATE TABLE IF NOT EXISTS login ("
    //          + " username text,"
    //          + " password text NOT NULL"
    //          + ");";
    // runStatement(c1,sql);

    String sql3 = "CREATE TABLE IF NOT EXISTS user("
          + " userid int primary key not null,"
          + " name text,"
          + " address text NOT NULL,"
          + " phoneno text,"
          + " password text "
          + ");";         runStatement(c1,sql3);
    String sql4 = "CREATE TABLE IF NOT EXISTS bookTable("
          + " bookid int primary key not null,"
          + " bookname text,"
          + " authorname text NOT NULL "
          + ");";         runStatement(c1,sql4);
    String sql6="CREATE TABLE IF NOT EXISTS issuedbooks("
          +"bookid int primary key not null,"
          +"bookname text,"
```

```
ResultSet checkit = runStatement(c1, check);          if(checkit.next())
        +"dueDate text"
              +");";
        ResultSet res =  runStatement(c1,sql6);
        // addUser(c1);
        // addBook(c1);
        // bookSearch(c1);          // deleteLibrarianInfo(c1);          login(c1);
        // issueBook(c1);
        // returnBook(c1);
    }
}
```

CODE FOR RUNNING CODE (BATCH FILE):

```
@echo off javac app.java java -cp ".;sqlite-jdbc-
3.41.0.0.jar" app timeout 10
```

CONCLUSION:

In conclusion, the library management system project is an essential tool for libraries that want to manage their resources effectively. The proposed system is designed to automate various library management tasks, reducing the time and effort required to perform these tasks manually. It provides librarians with the tools they need to manage library resources effectively, including accurate record-keeping, easy access to information, and better resource management.

The proposed system is scalable, reliable, efficient, and user-friendly, making it suitable for libraries of any size. It is developed using Java programming language and SQLite database management system, and it requires modest hardware requirements that are easily met by most modern computers.

Overall, the library management system project is a valuable tool for libraries that want to improve their efficiency, accuracy, and service quality. It is an excellent example of how technology can be used to enhance traditional library management practices, making it easier for librarians to provide quality service to library users. With the implementation of the proposed system, the library can better manage its resources, improve its services, and provide a better experience for its users.

REFERENCES:

1. SQL, PL/SQL : The programming Language of oracle by Ivan Bayross, BPB Publication.
2. SIMPLIFIED Approach to DBMS by Prateek Bhatia and Gurvinder Singh, Kalyani Publishers.
3. JAVA THE COMPLETE REFERENCE by Herbert Schildt, McGraw Hill Education.