

LIBRARY MANAGEMENT SYSTEM

Details

The Library Management System offers five functions:

1. Viewing available books
2. Borrowing a book
3. Returning a book
4. Adding a book
5. Removing a book

Program

```
import java.sql.*;
import java.util.Scanner;

public class LibraryManagementSystem {

    private static final String DB_URL = "jdbc:mysql://localhost:3306/library_mgmt";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "";

    public static void main(String[] args) {

        try {

            Connection connection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
```

```
System.out.println("Connected to database.");  
Scanner scanner = new Scanner(System.in);  
while (true) {  
    System.out.println("\n1. Display available books");  
    System.out.println("2. Borrow a book");  
    System.out.println("3. Return a book");  
    System.out.println("4. Add a book");  
    System.out.println("5. Remove a book");  
    System.out.println("6. Exit");  
    System.out.print("Enter your choice: ");  
    int choice = scanner.nextInt();  
    switch (choice) {  
        case 1:  
            displayAvailableBooks(connection);  
            break;  
        case 2:  
            borrowBook(connection, scanner);  
            break;  
        case 3:  
            returnBook(connection, scanner);  
            break;  
        case 4:  
            addBook(connection, scanner);  
            break;
```

```

        case 5:

            removeBook(connection, scanner);

            break;

        case 6:

            System.out.println("Exiting...");

            connection.close();

            System.exit(0);

            break;

        default:

            System.out.println("Invalid choice. Please try again.");

    }

}

} catch (SQLException e) {

    System.out.println("Database connection error: " + e.getMessage());

}

}

private static void displayAvailableBooks(Connection connection) throws
SQLException {

    PreparedStatement preparedStatement = connection.prepareStatement(

        "SELECT * FROM books WHERE book_id NOT IN (SELECT book_id FROM
        borrowed_books WHERE return_date IS NULL)"

    );

    ResultSet resultSet = preparedStatement.executeQuery();

    System.out.println("\nAvailable books:");

    while (resultSet.next()) {

```

```

        System.out.println(
            resultSet.getInt("book_id") + " | " +
                resultSet.getString("book_name") + " | " +
                resultSet.getString("book_author") + " | " +
                resultSet.getDouble("price") + " | " +
                resultSet.getDate("published_date")
        );
    }
    resultSet.close();
    preparedStatement.close();
}

private static void borrowBook(Connection connection, Scanner scanner) throws
SQLException {
    System.out.print("Enter the ID of the book you want to borrow: ");
    int bookId = scanner.nextInt();
    scanner.nextLine(); // Consume newline
    // Check if the book is available
    PreparedStatement preparedStatement = connection.prepareStatement(
        "SELECT * FROM borrowed_books WHERE book_id = ? AND return_date IS
        NULL"
    );
    preparedStatement.setInt(1, bookId);
    ResultSet resultSet = preparedStatement.executeQuery();
    if (resultSet.next()) {
        System.out.println("This book is already borrowed.");
    }
}

```

```
        return;
    }

    resultSet.close();

    preparedStatement.close();


    System.out.print("Enter your name: ");

    String borrowerName = scanner.nextLine();

    // Get current date

    Date borrowedDate = new Date(System.currentTimeMillis());


    preparedStatement = connection.prepareStatement(

        "INSERT INTO borrowed_books (book_id, book_name, borrowed_date,
        borrowed_person) VALUES (?, ?, ?, ?)"

    );

    preparedStatement.setInt(1, bookId);

    // Get book details for the borrowed book

    PreparedStatement bookDetailsStatement = connection.prepareStatement(

        "SELECT book_name FROM books WHERE book_id = ?"

    );

    bookDetailsStatement.setInt(1, bookId);

    ResultSet bookDetailsResultSet = bookDetailsStatement.executeQuery();

    if (bookDetailsResultSet.next()) {

        preparedStatement.setString(2, bookDetailsResultSet.getString("book_name"));

    }

}
```

```

else {
    System.out.println("Book not found.");
    return;
}
bookDetailsResultSet.close();
bookDetailsStatement.close();
preparedStatement.setDate(3, borrowedDate);
preparedStatement.setString(4, borrowerName);
int rowsAffected = preparedStatement.executeUpdate();
if (rowsAffected > 0) {
    System.out.println("Book borrowed successfully.");
}
else {
    System.out.println("Failed to borrow book.");
}
preparedStatement.close();
}

private static void returnBook(Connection connection, Scanner scanner) throws
SQLException {
    System.out.print("Enter the ID of the book you want to return: ");
    int bookId = scanner.nextInt();
    // Get current date
    Date returnDate = new Date(System.currentTimeMillis());
    PreparedStatement preparedStatement = connection.prepareStatement(

```

```
"UPDATE borrowed_books SET return_date = ? WHERE book_id = ? AND  
return_date IS NULL"
```

```
);
```

```
preparedStatement.setDate(1, returnDate);
```

```
preparedStatement.setInt(2, bookId);
```

```
int rowsAffected = preparedStatement.executeUpdate();
```

```
if (rowsAffected > 0) {
```

```
    System.out.println("Book returned successfully.");
```

```
}
```

```
else {
```

```
    System.out.println("Failed to return book. Either the book was not borrowed or  
the ID is incorrect.");
```

```
}
```

```
preparedStatement.close();
```

```
}
```

```
private static void addBook(Connection connection, Scanner scanner) throws  
SQLException {
```

```
    scanner.nextLine(); // Consume newline
```

```
    System.out.print("Enter book name: ");
```

```
    String bookName = scanner.nextLine();
```

```
    System.out.print("Enter author name: ");
```

```
    String authorName = scanner.nextLine();
```

```
    System.out.print("Enter price: ");
```

```
    double price = scanner.nextDouble();
```

```
    System.out.print("Enter published date (YYYY-MM-DD): ");
```

```
String publishedDateStr = scanner.next();

try {

    Date publishedDate = Date.valueOf(publishedDateStr);

    PreparedStatement preparedStatement = connection.prepareStatement(
        "INSERT INTO books (book_name, book_author, price, published_date)
VALUES (?, ?, ?, ?)"
    );

    preparedStatement.setString(1, bookName);
    preparedStatement.setString(2, authorName);
    preparedStatement.setDouble(3, price);
    preparedStatement.setDate(4, publishedDate);

    int rowsAffected = preparedStatement.executeUpdate();

    if (rowsAffected > 0) {

        System.out.println("Book added successfully.");

    }

} else {

    System.out.println("Failed to add book.");

    preparedStatement.close();

} catch (IllegalArgumentException e) {

    System.out.println("Invalid date format. Please use the format YYYY-MM-DD.");

}

}
```



```
private static void removeBook(Connection connection, Scanner scanner) throws
SQLException {

    System.out.print("Enter the ID of the book you want to remove: ");

    int bookId = scanner.nextInt()

    PreparedStatement preparedStatement = connection.prepareStatement(

        "DELETE FROM books WHERE book_id = ? AND book_id NOT IN (SELECT
book_id FROM borrowed_books WHERE return_date IS NULL)"

    );

    preparedStatement.setInt(1, bookId);

    int rowsAffected = preparedStatement.executeUpdate();

    if (rowsAffected > 0) {

        System.out.println("Book removed successfully.");

    }

    else {

        System.out.println("Failed to remove book. Either the book ID is incorrect or the
book is borrowed.");

    }

    preparedStatement.close();

}

}
```

OUTPUT

Created a Database : library_mgmt

Table Creation

```
CREATE TABLE books (  
    book_id INT AUTO_INCREMENT PRIMARY KEY,  
    book_name VARCHAR(255) NOT NULL,  
    book_author VARCHAR(255) NOT NULL,  
    Price DECIMAL(10, 2) NOT NULL,  
    Published_date DATE NOT NULL  
);
```

```
CREATE TABLE borrowed_books (  
    book_id INT,  
    book_name VARCHAR(255),  
    borrowed_date DATE NOT NULL,  
    return_date DATE,  
    borrowed_person VARCHAR(255) NOT NULL,  
    FOREIGN KEY (book_id) REFERENCES books(book_id)  
);
```

Database : library_mgmt and Tables : books and borrowed_books

The screenshot displays the phpMyAdmin web interface in a browser window. The address bar shows the URL: `localhost/phpmyadmin/index.php?route=/database/structure&db=library_mgmt`. The interface is for the `library_mgmt` database on a server at `127.0.0.1`.

On the left sidebar, the database structure is shown, including `information_schema`, `library_mgmt` (selected), `mysql`, `performance_schema`, `phpmyadmin`, and `test`. Under `library_mgmt`, there are two tables: `books` and `borrowed_books`.

The main panel shows the 'Structure' tab for the `library_mgmt` database. It includes a 'Filters' section with a search box. Below this is a table listing the database's tables:

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> books	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
<input type="checkbox"/> borrowed_books	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	32.0 KiB	-
2 tables	Sum	6	InnoDB	utf8mb4_general_ci	48.0 KiB	0 B

Below the table list, there are options to 'Check all' and 'With selected:'. At the bottom of the main panel, there is a 'Create new table' section with a 'Table name' input field, a 'Number of columns' input field (set to 4), and a 'Create' button.

The bottom of the image shows a Windows taskbar with various application icons, the system clock showing 6:54 PM on 2/25/2024, and the language set to ENG IN.

1. To display available books

1. Display available books

2. Borrow a book

3. Return a book

4. Add a book

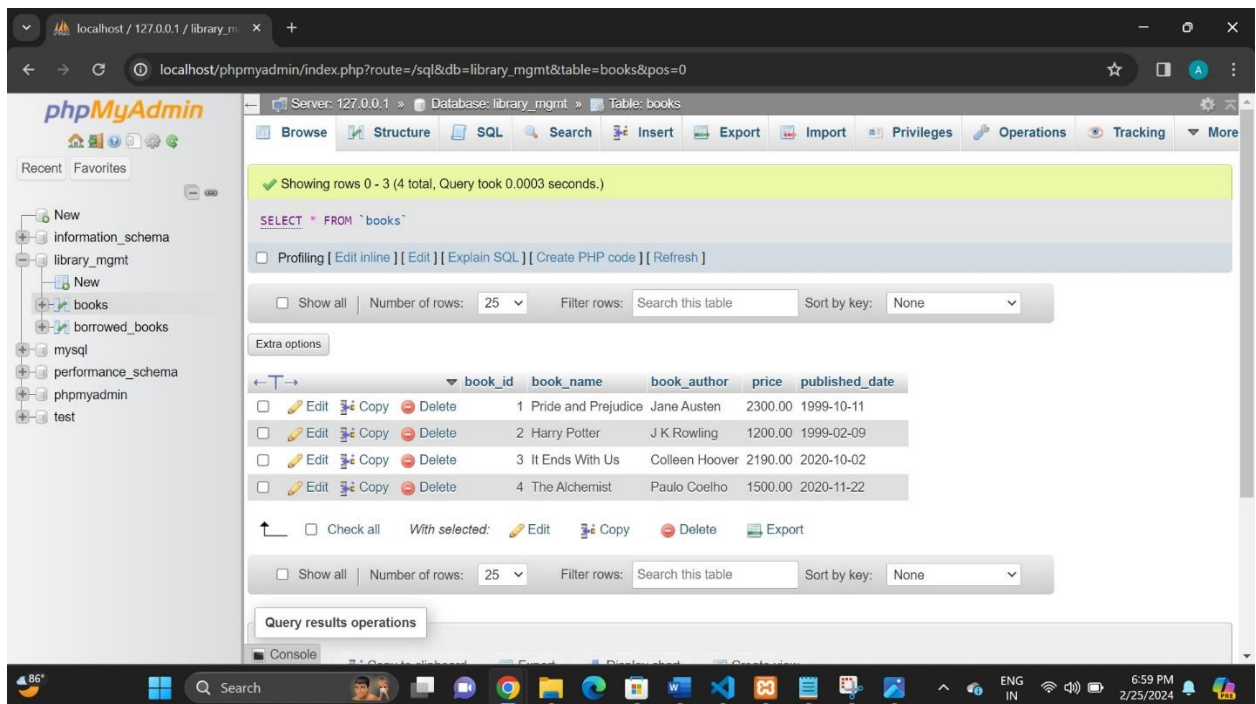
5. Remove a book

6. Exit

Enter your choice: 1

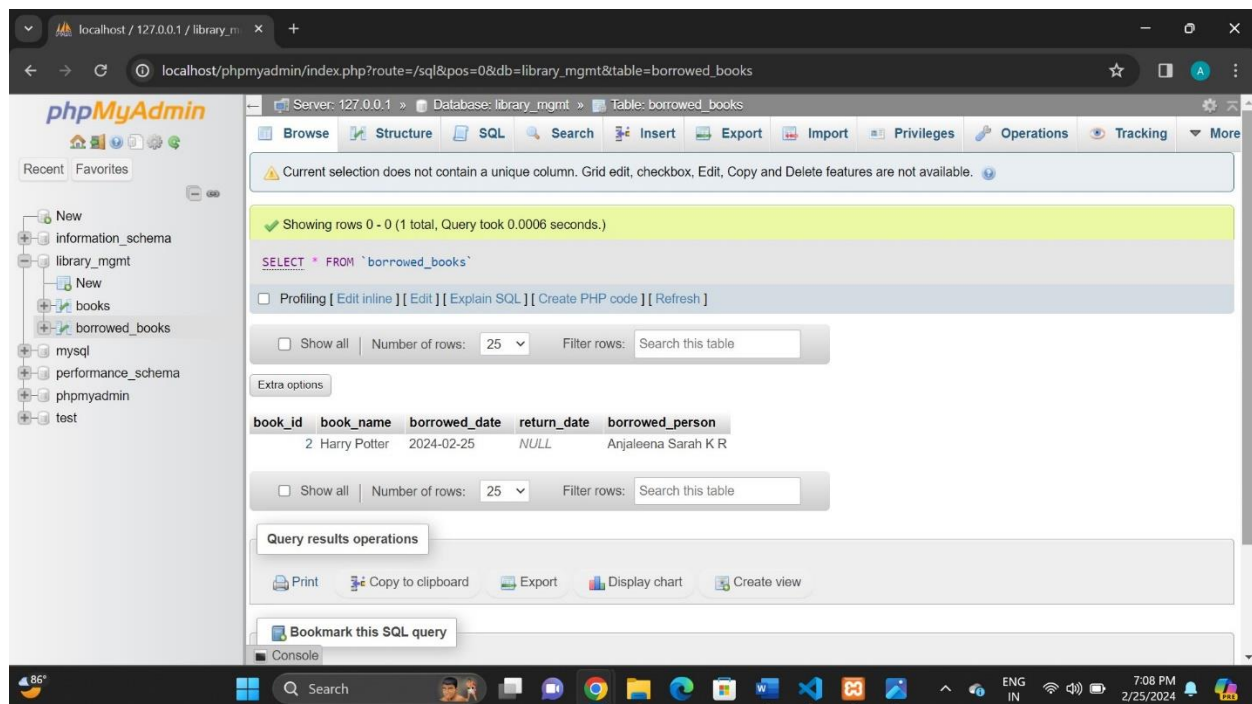
Available books:

1		Pride and Prejudice		Jane Austen		2300.0		1999-10-11
2		Harry Potter		J K Rowling		1200.0		1999-02-09
3		It Ends With Us		Colleen Hoover		2190.0		2020-10-02
4		The Alchemist		Paulo Coelho		1500.0		2020-11-22



2. To borrow a book

```
1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 2
Enter the ID of the book you want to borrow: 2
Enter your name: Anjaleena Sarah K R
Book borrowed successfully.
```



3. To return a book

```
1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 3
Enter the ID of the book you want to return: 2
Book returned successfully.
```

After returning the book – return_date is updated

The screenshot shows the phpMyAdmin interface for the 'library_mgmt' database. The 'borrowed_books' table is selected, and the SQL query 'SELECT * FROM `borrowed_books`' is executed. The results show one row with the following data:

book_id	book_name	borrowed_date	return_date	borrowed_person
2	Harry Potter	2024-02-25	2024-02-25	Anjaleena Sarah K R

The interface also shows the 'Query results operations' section with options like Print, Copy to clipboard, Export, Display chart, and Create view. The console at the bottom shows the command prompt output from the previous steps.

4. To add a book

```
1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 4
Enter book name: The Great Gatsby
Enter author name: F Scott
Enter price: 2500
Enter published date (YYYY-MM-DD): 2020-09-11
Book added successfully.
```

The screenshot shows the phpMyAdmin web interface in a browser. The address bar indicates the URL: `localhost/phpmyadmin/index.php?route=/sql&pos=0&db=library_mgmt&table=borrowed_books`. The left sidebar shows the database structure with 'library_mgmt' selected, containing 'books' and 'borrowed_books' tables. The main panel displays the 'borrowed_books' table structure and a single row of data.

Database: library_mgmt » Table: borrowed_books

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

`SELECT * FROM `borrowed_books``

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table

Extra options

book_id	book_name	borrowed_date	return_date	borrowed_person
2	Harry Potter	2024-02-25	2024-02-25	Anjaleena Sarah K R

Show all | Number of rows: 25 | Filter rows: Search this table

Query results operations

Print | Copy to clipboard | Export | Display chart | Create view

Bookmark this SQL query

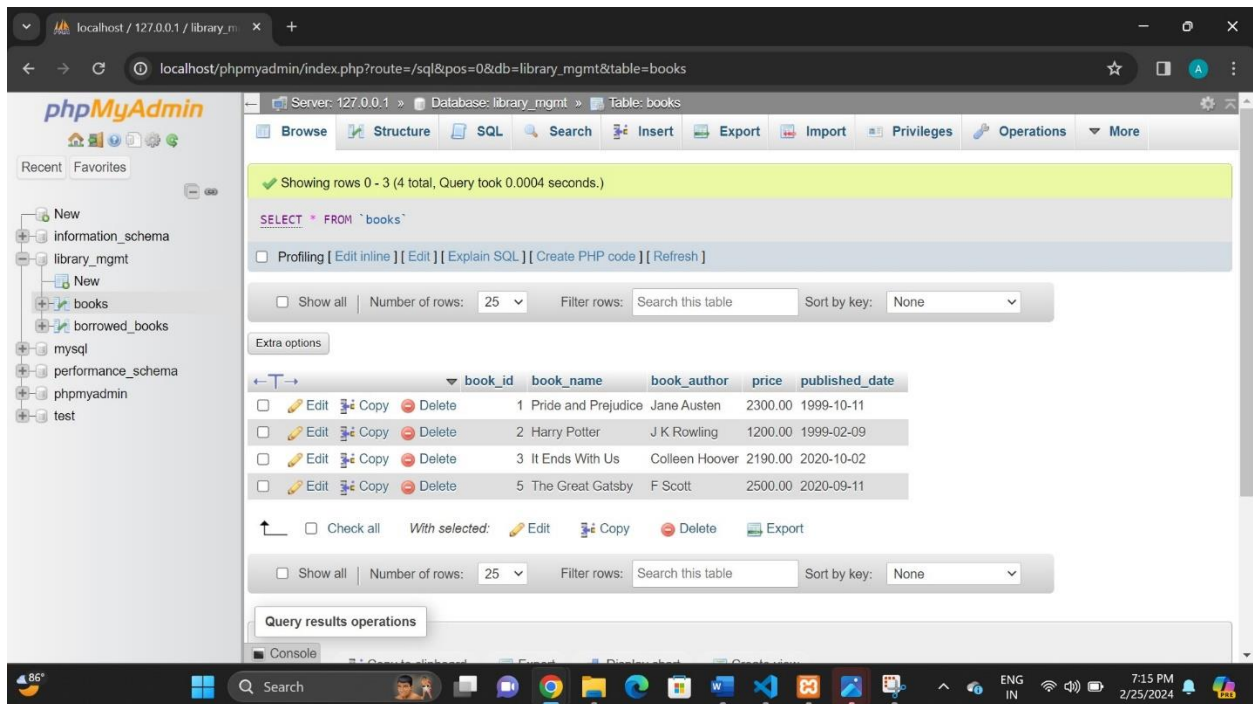
Console

5. To remove a book

```
1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 5
Enter the ID of the book you want to remove: 4
Book removed successfully.

1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 1

Available books:
1 | Pride and Prejudice | Jane Austen | 2300.0 | 1999-10-11
2 | Harry Potter | J K Rowling | 1200.0 | 1999-02-09
3 | It Ends With Us | Colleen Hoover | 2190.0 | 2020-10-02
5 | The Great Gatsby | F Scott | 2500.0 | 2020-09-11
```



6. Exiting from the program

```
1. Display available books
2. Borrow a book
3. Return a book
4. Add a book
5. Remove a book
6. Exit
Enter your choice: 6
Exiting...
PS C:\java_pjrt> █
```