LMS Project Report

User Authentication Code:

```java
import java.util.HashMap;
import java.util.Map;

public class AuthenticationService {  4 usages
    private Map<String, User> users; // This should be replaced by a proper user management system.  4 usages
    private User currentUser;  3 usages

    public AuthenticationService() {  1 usage
        this.users = new HashMap<>();
        // In a real application, replace this with a database or another secure storage method
        this.users.put("admin", new User( username: "admin",  password: "adminpass",  role: "admin", id: 100));
        this.users.put("borrower", new User( username: "borrower",  password: "borrowerpass",  role: "borrower", id: 200));
    }

    public boolean login(String username, String password) {  1 usage
        // Authentication logic - in a real-world app, securely check password with hashed value.
        User user = users.get(username);
        if (user != null && user.getPassword().equals(password)) {
            this.currentUser = user;
            return true;
        }
        return false;
    }

    public void logout() {  no usages
        this.currentUser = null;
    }

    public User getCurrentUser() {  1 usage
        return currentUser;
    }
}
```

User Login Screenshots:

Admin Output-

```
/Users/udaybhaskarvalapadasu/Library/Java/JavaVirtualMach
Welcome to the Library Management System!
1. Login
2. Exit
Choose an option: 1
Username: admin
Password: admin
Login failed!
Welcome to the Library Management System!
1. Login
2. Exit
Choose an option: 1
Username: admin
Password: adminpass

Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option:
```

Borrower Ouput-

```
Welcome to the Library Management System!
1. Login
2. Exit
Choose an option: 1
Username: borrower
Password: borrowerpass

Borrower Menu:
1. Search Books
2. View My Borrowing History
3. Logout
Choose an option:
```

Add Book Operation Code :

```java
public void addBook() {  1 usage
    System.out.println("Add a new book");
    System.out.print("Enter ISBN: ");
    String isbn = scanner.nextLine();

    System.out.print("Enter Title: ");
    String title = scanner.nextLine();

    System.out.print("Enter Author: ");
    String author = scanner.nextLine();

    System.out.print("Enter Genre: ");
    String genre = scanner.nextLine();

    System.out.print("Enter Publication Year: ");
    int publicationYear;
    try {
        publicationYear = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Publication year needs to be an integer.");
        return;
    }

    boolean bookAdded = addNewBook(isbn, title, author, genre, publicationYear);
    if (bookAdded) {
        System.out.println("Book successfully added to the system.");
    } else {
        System.out.println("Failed to add the book to the system.");
    }
}
```

```java
public boolean addNewBook(String isbn, String title, String author, String genre, int publicationYear) {  1 usage
    String sql = "INSERT INTO Books (ISBN, Title, Author, Genre, PublicationYear) VALUES (?, ?, ?, ?, ?)";
    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setString( parameterIndex: 1, isbn);
        statement.setString( parameterIndex: 2, title);
        statement.setString( parameterIndex: 3, author);
        statement.setString( parameterIndex: 4, genre);
        statement.setInt( parameterIndex: 5, publicationYear);

        int rowsInserted = statement.executeUpdate();
        return rowsInserted > 0;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

Add Book Output Screenshots:

```
Welcome to the Library Management System!
1. Login
2. Exit
Choose an option: 1
Username: admin
Password: adminpass

Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 1
Add a new book
Enter ISBN: 1234567890129
Enter Title: The Seventh Title
Enter Author: Uday Bhaskar
Enter Genre: Fantasy
Enter Publication Year: 2001
Book successfully added to the system.
```

```
mysql> select * from books
    -> ;
+---------------+-------------------+-------------+-------------+-----------------+
| ISBN          | Title             | Author      | Genre       | PublicationYear |
+---------------+-------------------+-------------+-------------+-----------------+
| 1234567890123 | The First Title   | Author A    | Fantasy     |            2000 |
| 1234567890124 | The Second Title  | Author B    | Sci-Fi      |            2001 |
| 1234567890125 | The Third Title   | Author C    | Mystery     |            2002 |
| 1234567890126 | The Fourth Title  | Author D    | Non-fiction |            2003 |
| 1234567890127 | The Fifth Title   | Author E    | Romance     |            2004 |
| 1234567890128 | The Six Title     | Author F    | Sci-Fi      |            2005 |
| 1234567890129 | The Seventh Title | Uday Bhaskar | Fantasy     |            2001 |
+---------------+-------------------+-------------+-------------+-----------------+
7 rows in set (0.00 sec)
```

Add Book Copies Operation Code:

```java
public void addBookCopies() {  1 usage
    System.out.println("Add copies of a book");
    System.out.print("Enter ISBN of the book: ");
    String isbn = scanner.nextLine();

    System.out.print("Enter the number of copies to add: ");
    int numberOfCopies;
    try {
        numberOfCopies = Integer.parseInt(scanner.nextLine());
    } catch (NumberFormatException e) {
        System.out.println("Number of copies needs to be an integer.");
        return;
    }

    boolean copiesAdded = addBookCopies(isbn, numberOfCopies);
    if (copiesAdded) {
        System.out.println(numberOfCopies + " copies successfully added for ISBN " + isbn);
    } else {
        System.out.println("Failed to add copies for ISBN " + isbn);
    }
}
```

```java
public boolean addBookCopies(String isbn, int numberOfCopies) {  1 usage
    String sql = "INSERT INTO BookCopies (ISBN, AvailabilityStatus) VALUES (?, ?)";
    try (PreparedStatement statement = connection.prepareStatement(sql)) {
        int rowsInserted = 0;
        for (int i = 0; i < numberOfCopies; i++) {
            statement.setString( parameterIndex: 1, isbn);
            statement.setBoolean( parameterIndex: 2,  x: true); // New copies are assumed to be
            rowsInserted += statement.executeUpdate();
        }
        return rowsInserted == numberOfCopies;
    } catch (SQLException e) {
        e.printStackTrace();
        return false;
    }
}
```

Add Book Copies Operation Output:

```
Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 2
Add copies of a book
Enter ISBN of the book: 1234567890129
Enter the number of copies to add: 2
2 copies successfully added for ISBN 1234567890129
```

```
mysql> select * from bookcopies
    -> ;
+--------------+---------------+--------------------+
| SerialNumber | ISBN          | AvailabilityStatus |
+--------------+---------------+--------------------+
|            1 | 1234567890123 |                  1 |
|            2 | 1234567890123 |                  0 |
|            3 | 1234567890123 |                  1 |
|            4 | 1234567890124 |                  1 |
|            5 | 1234567890124 |                  1 |
|            6 | 1234567890124 |                  0 |
|            7 | 1234567890125 |                  1 |
|            8 | 1234567890125 |                  1 |
|            9 | 1234567890125 |                  0 |
|           10 | 1234567890126 |                  1 |
|           11 | 1234567890126 |                  0 |
|           12 | 1234567890126 |                  1 |
|           13 | 1234567890127 |                  1 |
|           14 | 1234567890127 |                  0 |
|           15 | 1234567890127 |                  1 |
|           16 | 1234567890123 |                  1 |
|           17 | 1234567890124 |                  1 |
|           18 | 1234567890125 |                  1 |
|           19 | 1234567890126 |                  1 |
|           20 | 1234567890127 |                  1 |
|           23 | 1234567890128 |                  1 |
|           24 | 1234567890128 |                  1 |
|           25 | 1234567890129 |                  1 |
|           26 | 1234567890129 |                  1 |
+--------------+---------------+--------------------+
24 rows in set (0.00 sec)
```

Add Borrower Operation Code:

```java
public void addBorrower() {  1 usage
    System.out.println("Add a new borrower to the system");

    System.out.print("Enter borrower's name: ");
    String name = scanner.nextLine();

    System.out.print("Enter borrower's email: ");
    String email = scanner.nextLine();

    System.out.print("Enter borrower's contact number: ");
    String contactNumber = scanner.nextLine();

    boolean isRegistered = registerNewBorrower(name, email, contactNumber);
    if (isRegistered) {
        System.out.println("New borrower added successfully.");
    } else {
        System.out.println("Failed to add the new borrower.");
    }
}
```

```java
public boolean registerNewBorrower(String name, String email, String contactNumber) {  1 usage
        String sql = "INSERT INTO Borrowers (Name, Email, ContactNumber) VALUES (?, ?, ?)";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {
            statement.setString( parameterIndex: 1, name);
            statement.setString( parameterIndex: 2, email);
            statement.setString( parameterIndex: 3, contactNumber);

            int rowsInserted = statement.executeUpdate();
            return rowsInserted > 0;
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
}
```

Add Borrower Operation Output:

```
Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 3
Add a new borrower to the system
Enter borrower's name: Manish
Enter borrower's email: manish@gmail.com
Enter borrower's contact number: 9407628855
New borrower added successfully.
```

```
mysql> Select * from borrowers;
+------------+---------------+------------------------+---------------+
| BorrowerID | Name          | Email                  | ContactNumber |
+------------+---------------+------------------------+---------------+
|          1 | Alice Johnson | alice.johnson@email.com | 123-456-7890  |
|          2 | Bob Harris    | bob.harris@email.com    | 098-765-4321  |
|          3 | Cindy Lou     | cindy.lou@email.com     | 543-210-9876  |
|          4 | David Green   | david.green@email.com   | 210-543-6789  |
|          5 | Eva Stone     | eva.stone@email.com     | 678-901-2345  |
|          6 | Uday          | uday@gmail.com          | 9406298875    |
|          7 | Manish        | manish@gmail.com        | 9407628855    |
+------------+---------------+------------------------+---------------+
7 rows in set (0.00 sec)
```

Add Borrower Book Operation Code:

```java
public void borrowBook() {  1 usage
    System.out.println("Process a book borrowing");

    System.out.print("Enter Borrower ID: ");
    int borrowerId = scanner.nextInt();
    scanner.nextLine();  // Flush the newline

    System.out.print("Enter Book ISBN: ");
    String isbn = scanner.nextLine();

    System.out.print("Enter Borrowing Date (YYYY-MM-DD): ");
    String borrowingDate = scanner.nextLine();

    boolean isBorrowed = processBookBorrowing(borrowerId, isbn, borrowingDate);
    if (isBorrowed) {
        System.out.println("Book borrowing processed successfully.");
    } else {
        System.out.println("Failed to process book borrowing.");
    }
}
```

```java
public boolean processBookBorrowing(int borrowerId, String isbn, String borrowingDate) {  1 usage
        // Start a transaction
        try {
            connection.setAutoCommit(false); // Disable auto-commit for transaction

            // Step 1: Check if there is an available copy
            String checkCopySQL = "SELECT SerialNumber FROM BookCopies WHERE ISBN = ? AND AvailabilityStatus = TRUE LIMIT 1";
            int serialNumber = -1;
            try (PreparedStatement checkCopyStmt = connection.prepareStatement(checkCopySQL)) {
                checkCopyStmt.setString( parameterIndex: 1, isbn);
                ResultSet resultSet = checkCopyStmt.executeQuery();
                if (resultSet.next()) {
                    serialNumber = resultSet.getInt( columnLabel: "SerialNumber");
                } else {
                    connection.rollback(); // No available copy found, rollback transaction
                    return false;
                }
            }

            // Step 2: Insert a new row into Transactions
            String transactionSQL = "INSERT INTO Transactions (BorrowerID, SerialNumber, BorrowingDate) VALUES (?, ?, ?)";
            try (PreparedStatement transactionStmt = connection.prepareStatement(transactionSQL)) {
                transactionStmt.setInt( parameterIndex: 1, borrowerId);
                transactionStmt.setInt( parameterIndex: 2, serialNumber);
                transactionStmt.setDate( parameterIndex: 3, Date.valueOf(borrowingDate));
                transactionStmt.executeUpdate();
            }
```

```java
            // Step 3: Update the book copy's availability status
            String updateCopySQL = "UPDATE BookCopies SET AvailabilityStatus = FALSE WHERE SerialNumber = ?";
            try (PreparedStatement updateCopyStmt = connection.prepareStatement(updateCopySQL)) {
                updateCopyStmt.setInt( parameterIndex: 1, serialNumber);
                updateCopyStmt.executeUpdate();
            }

            connection.commit(); // Commit the transaction
            return true;
        } catch (SQLException e) {
            e.printStackTrace();
            try {
                connection.rollback(); // Rollback the transaction on error
            } catch (SQLException rollbackEx) {
                rollbackEx.printStackTrace();
            }
            return false;
        } finally {
            try {
                connection.setAutoCommit(true); // Restore auto-commit mode
            } catch (SQLException finalEx) {
                finalEx.printStackTrace();
            }
        }
    }
```

Add Borrower Book Operation Output:

```
Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 4
Process a book borrowing
Enter Borrower ID: 7
Enter Book ISBN: 1234567890129
Enter Borrowing Date (YYYY-MM-DD): 2024-04-10
Book borrowing processed successfully.
```

```
mysql> select * from transactions;
+---------------+------------+--------------+---------------+--------------+
| TransactionID | BorrowerID | SerialNumber | BorrowingDate | ReturnDate   |
+---------------+------------+--------------+---------------+--------------+
|             1 |          1 |            1 | 2024-01-01    | 2024-01-15   |
|             2 |          2 |            2 | 2024-01-05    | NULL         |
|             3 |          3 |            3 | 2024-02-01    | 2024-02-15   |
|             4 |          4 |            4 | 2024-02-10    | NULL         |
|             5 |          5 |            5 | 2024-03-01    | 2024-03-15   |
|             6 |          6 |           13 | 2024-04-10    | 2024-04-11   |
|             7 |          7 |           25 | 2024-04-10    | NULL         |
+---------------+------------+--------------+---------------+--------------+
7 rows in set (0.00 sec)
```

Add Return Book Operation Code:

```java
public void returnBook() {  1 usage
    System.out.println("Process a book return");

    System.out.print("Enter Transaction ID of the borrowing record: ");
    int transactionId = scanner.nextInt(); // Make sure to handle InputMismatchException
    scanner.nextLine(); // Flush the newline

    System.out.print("Enter Return Date (YYYY-MM-DD): ");
    String returnDate = scanner.nextLine();

    boolean isReturned = processBookReturning(transactionId, returnDate);
    if (isReturned) {
        System.out.println("Book return processed successfully.");
    } else {
        System.out.println("Failed to process book return.");
    }
}

public boolean processBookReturning(int transactionId, String returnDate) {  1 usage
    // Start a transaction
    try {
        connection.setAutoCommit(false); // Disable auto-commit for transaction

        // Step 1: Retrieve the SerialNumber of the book copy based on the transactionId
        String getCopySQL = "SELECT SerialNumber FROM Transactions WHERE TransactionID = ?";
        int serialNumber = -1;
        try (PreparedStatement getCopyStmt = connection.prepareStatement(getCopySQL)) {
            getCopyStmt.setInt( parameterIndex: 1, transactionId);
            ResultSet resultSet = getCopyStmt.executeQuery();
            if (resultSet.next()) {
                serialNumber = resultSet.getInt( columnLabel: "SerialNumber");
            } else {
                connection.rollback(); // No borrowing record found, rollback transaction
                return false;
            }
        }

        // Step 2: Update the return date in Transactions
        String updateTransactionSQL = "UPDATE Transactions SET ReturnDate = ? WHERE TransactionID = ?";
        try (PreparedStatement updateTransactionStmt = connection.prepareStatement(updateTransactionSQL)) {
            updateTransactionStmt.setDate( parameterIndex: 1, Date.valueOf(returnDate));
            updateTransactionStmt.setInt( parameterIndex: 2, transactionId);
            updateTransactionStmt.executeUpdate();
        }

        // Step 3: Update the book copy's availability status
        String updateCopySQL = "UPDATE BookCopies SET AvailabilityStatus = TRUE WHERE SerialNumber = ?";
        try (PreparedStatement updateCopyStmt = connection.prepareStatement(updateCopySQL)) {
            updateCopyStmt.setInt( parameterIndex: 1, serialNumber);
            updateCopyStmt.executeUpdate();
        }

        connection.commit(); // Commit the transaction
        return true;
    } catch (SQLException e) {
        e.printStackTrace();
        try {
            connection.rollback(); // Rollback the transaction on error
        } catch (SQLException rollbackEx) {
            rollbackEx.printStackTrace();
        }
        return false;
    } finally {
        try {
            connection.setAutoCommit(true); // Restore auto-commit mode
        } catch (SQLException finalEx) {
            finalEx.printStackTrace();
        }
    }
}
```

Add Return Book Operation Output:

```
Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 5
Process a book return
Enter Transaction ID of the borrowing record: 7
Enter Return Date (YYYY-MM-DD): 2024-04-11
Book return processed successfully.
```

```
mysql> select * from transactions;
+---------------+------------+--------------+---------------+------------+
| TransactionID | BorrowerID | SerialNumber | BorrowingDate | ReturnDate |
+---------------+------------+--------------+---------------+------------+
|             1 |          1 |            1 | 2024-01-01    | 2024-01-15 |
|             2 |          2 |            2 | 2024-01-05    | NULL       |
|             3 |          3 |            3 | 2024-02-01    | 2024-02-15 |
|             4 |          4 |            4 | 2024-02-10    | NULL       |
|             5 |          5 |            5 | 2024-03-01    | 2024-03-15 |
|             6 |          6 |           13 | 2024-04-10    | 2024-04-11 |
|             7 |          7 |           25 | 2024-04-10    | 2024-04-11 |
+---------------+------------+--------------+---------------+------------+
7 rows in set (0.00 sec)
```

Add View Borrowing History Code:

```
public List<Map<String, Object>> viewBorrowingHistorySQL(int borrowerId) {  1 usage
        List<Map<String, Object>> history = new ArrayList<>();
        String sql = "SELECT t.TransactionID, t.SerialNumber, t.BorrowingDate, t.ReturnDate, b.Title " +
                "FROM Transactions t " +
                "JOIN BookCopies bc ON t.SerialNumber = bc.SerialNumber " +
                "JOIN Books b ON bc.ISBN = b.ISBN " +
                "WHERE t.BorrowerID = ?";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {...} catch (SQLException e) {
            e.printStackTrace();
        }
        return history;
    }

public void viewBorrowingHistory() {  2 usages
    System.out.print("Enter Borrower ID to view borrowing history: ");
    int borrowerId = scanner.nextInt();
    scanner.nextLine();  // to consume the rest of the line

    List<Map<String, Object>> history = viewBorrowingHistorySQL(borrowerId);
    if (history.isEmpty()) {
        System.out.println("No borrowing history found for the borrower with ID " + borrowerId);
    } else {
        System.out.println("Borrowing history for borrower ID " + borrowerId + ":");
        for (Map<String, Object> transaction : history) {
            System.out.println("Transaction ID: " + transaction.get("TransactionID") +
                    ", Serial Number: " + transaction.get("SerialNumber") +
                    ", Book Title: " + transaction.get("Title") +
                    ", Borrowing Date: " + transaction.get("BorrowingDate") +
                    ", Return Date: " + transaction.get("ReturnDate"));
        }
    }
}
```

Add View Borrowing History Output:

```
Admin Menu:
1. Add Book
2. Add BookCopies
3. Add Borrower
4. Borrow Book
5. Return Book
6. Search Books
7. View Borrowing History
8. Logout
Choose an option: 7
Enter Borrower ID to view borrowing history: 7
Borrowing history for borrower ID 7:
Transaction ID: 7, Serial Number: 25, Book Title: The Seventh Title, Borrowing Date: 2024-04-10, Return Date: 2024-04-11
```

Borrower Access of History:

```
Welcome to the Library Management System!
1. Login
2. Exit
Choose an option: 1
Username: borrower
Password: borrowerpass

Borrower Menu:
1. Search Books
2. View My Borrowing History
3. Logout
Choose an option: 2
Enter Borrower ID to view borrowing history: 7
Borrowing history for borrower ID 7:
Transaction ID: 7, Serial Number: 25, Book Title: The Seventh Title, Borrowing Date: 2024-04-10, Return Date: 2024-04-11

Borrower Menu:
1. Search Books
2. View My Borrowing History
3. Logout
Choose an option:
```

Search Logic:

```
    }

    public void searchBooks() {  2 usages
        System.out.println("Search for books");

        System.out.print("Enter search keyword: ");
        String keyword = scanner.nextLine();

        List<Map<String, Object>> foundBooks = searchBooksLogic(keyword);
        if (foundBooks.isEmpty()) {
            System.out.println("No books found matching the criteria.");
        } else {
            System.out.println("Found books:");
            for (Map<String, Object> book : foundBooks) {
                System.out.println("ISBN: " + book.get("ISBN") +
                        ", Title: " + book.get("Title") +
                        ", Author: " + book.get("Author") +
                        ", Genre: " + book.get("Genre") +
                        ", Publication Year: " + book.get("PublicationYear") +
                        ", Serial Number: " + book.get("SerialNumber") +
                        ", Available: " + book.get("AvailabilityStatus"));
            }
        }
    }
```

```java
public List<Map<String, Object>> searchBooksLogic(String keyword) {  1 usage
        List<Map<String, Object>> results = new ArrayList<>();
        String sql = "SELECT b.*, bc.SerialNumber, bc.AvailabilityStatus " +
                "FROM Books b " +
                "JOIN BookCopies bc ON b.ISBN = bc.ISBN " +
                "WHERE b.ISBN LIKE ? OR b.Title LIKE ? OR b.Author LIKE ? " +
                "OR b.Genre LIKE ? OR CAST(b.PublicationYear AS CHAR) LIKE ?";
        try (PreparedStatement statement = connection.prepareStatement(sql)) {
            String searchKeyword = "%" + keyword + "%";
            statement.setString( parameterIndex: 1, searchKeyword);
            statement.setString( parameterIndex: 2, searchKeyword);
            statement.setString( parameterIndex: 3, searchKeyword);
            statement.setString( parameterIndex: 4, searchKeyword);
            statement.setString( parameterIndex: 5, searchKeyword);

            ResultSet rs = statement.executeQuery();
            while (rs.next()) {
                Map<String, Object> bookData = new HashMap<>();
                bookData.put("ISBN", rs.getString( columnLabel: "ISBN"));
                bookData.put("Title", rs.getString( columnLabel: "Title"));
                bookData.put("Author", rs.getString( columnLabel: "Author"));
                bookData.put("Genre", rs.getString( columnLabel: "Genre"));
                bookData.put("PublicationYear", rs.getInt( columnLabel: "PublicationYear"));
                bookData.put("SerialNumber", rs.getInt( columnLabel: "SerialNumber"));
                bookData.put("AvailabilityStatus", rs.getBoolean( columnLabel: "AvailabilityStatus"));
                results.add(bookData);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return results;
    }
```

main > java > LibraryService > processBookReturning

```
Search for books
Enter search keyword: fifith
No books found matching the criteria.

Borrower Menu:
1. Search Books
2. View My Borrowing History
3. Logout
Choose an option: 1
Search for books
Enter search keyword: fifth
Found books:
ISBN: 1234567890127, Title: The Fifth Title, Author: Author E, Genre: Romance, Publication Year: 2004, Serial Number: 13, Available: true
ISBN: 1234567890127, Title: The Fifth Title, Author: Author E, Genre: Romance, Publication Year: 2004, Serial Number: 14, Available: false
ISBN: 1234567890127, Title: The Fifth Title, Author: Author E, Genre: Romance, Publication Year: 2004, Serial Number: 15, Available: true
ISBN: 1234567890127, Title: The Fifth Title, Author: Author E, Genre: Romance, Publication Year: 2004, Serial Number: 20, Available: true

Borrower Menu:
1. Search Books
```