**Library Management System - System Architecture**

Nilay Jasapara

**System Overview**

The Library Management System (LMS) is a library management tool that allows a library to manage their borrowers and collection of books. The LMS allows users to check-in, check-out, query for books, create new borrowers, and more. This system was implemented using Java Swing for front-end interfaces and JDBC for MySQL connections. The following document will detail the system architecture surrounding how the LMS was designed and implemented.

The source code of the LMS is broken down into functional components where each class implements a function. The “LMS.java” file is the main entry point for the Library Management System and “HomeScreen.java” is the main screen. The following .java files are then functional components that are attached to the home screen. This design was chosen because it fits the given requirements of the system most closely. Figure 1 shows the initial view of the LMS’s home screen on startup.

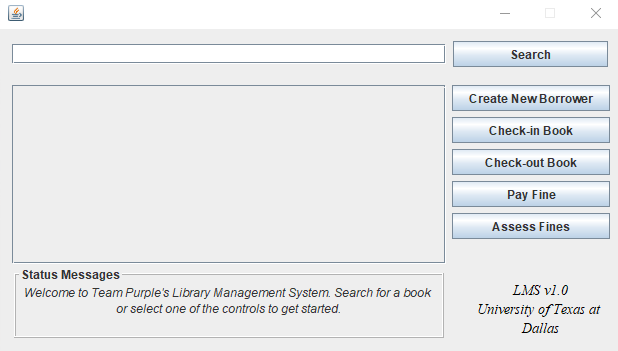


Figure 1. View of LMS home screen on startup.

As seen above, each button on the right corresponds to a requirement given in the design document. Some of these buttons will also open new windows with their own controls; these will not be discussed as they are not relevant to the system architecture. The large middle box is the query display which will display search queries given by the user in the uppermost textbox. The bottomost text pane is the status messages window which is responsible for notifying the user of pertinent information.

Next, the database schema will be discussed in detail.

**The Database**

The database LIBRARY is a relational database with six different tables: BOOK, BOOK\_AUTHORS, AUTHORS, BORROWER, BOOK\_LOANS, and FINES.

The BOOK table has two columns: Isbn (of type CHAR(10) and not null) and Title (of type VARCHAR(250) and not null). Isbn (10, not 13) is the primary key and has a CHECK constraint in which every value matches the regular expression in which the first nine characters are all digits while the tenth character can be a digit or a capital X. We assumed that every book must have a title, and if one has “no title,” the value would be the empty string.

The BOOK\_AUTHORS table has two columns: Author\_id (of type CHAR(8) and not null) and Isbn (same as BOOK.Isbn). Both columns make up the primary key and individually serve as foreign keys, with Author\_id referencing AUTHORS.Author\_id and Isbn referencing BOOK.Isbn. Because a foreign key must not have a null pointer, the AUTHORS table was created before the BOOK\_AUTHORS table (otherwise, we would have had to keep the Author\_id foreign key constraint in a separate SQL statement after the AUTHORS table was created).

The AUTHORS table has two columns: Author\_id (same as BOOK\_AUTHORS.Author\_id) and Name (of type VARCHAR(100)). Author\_id is the primary key and (like BOOK\_AUTHORS.Author\_id) has a CHECK constraint in which every value matches the regular expression in which the first two characters are “AU” (for author) and the remaining six characters are digits. When a new Author\_id is created because of a new author being added to the database, the number is one greater than the last Author\_id. Name is allowed to be null in case the author is unknown or withheld, in which Author\_id is (until the author is finally found) ‘AU000000’.

The BORROWER table has five columns: Card\_id (of type CHAR(8) and not null), Ssn (of type CHAR(11) and not null), Bname (of type VARCHAR(100) and not null), Address (same as Bname), and Phone (of type CHAR(14)). Card\_id is the primary key and has a CHECK constraint similar to that of Author\_id, only the first two characters are “ID” (to match baseline data) instead of “AU,” and like Author\_id, new Card\_id values are always one number greater. Ssn is a unique key, and its CHECK constraint ensures each value is in readable and typical Social Security Number format (XXX-XX-XXXX, where X is a digit and not necessarily the same digit). Bname holds the name of the borrower, Address hold the street address, city, and state (no zip code), and Phone has a CHECK constraint ensuring each value is in the form (XXX) XXX-XXXX, where X is a digit and not necessarily the same every time.

The BOOK\_LOANS table has six columns: Loan\_id (of type CHAR(8) and not null), Isbn (same as BOOK.Isbn; it is a foreign key), Card\_id (same as BORROWER.Card\_id; it is a foreign key), Date\_out (of type DATE and not null), Due\_date (same as Date\_out), and Date\_in (of type DATE but can be null). Loan\_id is the primary key and has a CHECK constraint similar to other ID values, but beginning with “LO” (for loan). Every new Loan\_id value is one higher than the previous, as well.

Finally, the FINES table has three columns: Loan\_id (same as BOOK\_LOANS.Loan\_id; it is a foreign key and the primary key), Fine\_amt (of type DECIMAL(4, 2) and not null), and Paid (of type BOOLEAN and not null). The amount of a fine for every overdue book is $0.25 per day late, with a maximum fine of $99.75 on late day 399 (since Fine\_amt is of type DECIMAL(4, 2)). Paid starts out as FALSE (0) until a borrower has checked in and paid fines for all books late, in which Paid becomes TRUE (1). There are no partial payments; the borrower must have paid once for all their books turned in late.

Initially, the BOOK\_LOANS and FINES tables have no tuples because no library first opens with a book already checked out. However, information for 25,000 books and 1,000 borrowers was included as appropriate from two CSV files from our instructor, Dr. Chris Davis: books.csv and borrowers.csv. They have been worked around entirely, unedited, and are included in the overall submission. Some columns from both CSV files naturally did not make our library database, and we as a team decided on the domains for Author\_id and Loan\_id.

**Library Management System Relevant Components**

As discussed briefly within the System Overview section, the LMS’s components are abstracted to separate modules to best conform to the given design document. The following are the design decisions and assumptions for the relevant components. *Note some of this is also alluded to within the LMS Quick Start Guide.*

Searching for Books

* Some books and authors' names have unicode characters within them. These characters are not allowed within the LMS but these books/authors will still appear in searches if some substring is found to match that record.

Paying Fines

* When paying fines, the user may look at the aggregate of fines for each user with unpaid fines or may view each unpaid fine on a given Card ID. If the aggregate view is desired, then no Card ID should be supplied. This way, all fines for a given user may be paid at once.

Other Design Decision and Assumptions

* All other components follow the given design requirements, therefore no other assumptions about users and how they will use the system was made.
* Design was chosen to be a single home screen and have separate controls open new windows. This was done for ease of use so that a user may use multiple tools at once.