**PROBLEM STATEMENT**

**Description**This SQL programming project involves the creation of a database host application that  
interfaces with a backend SQL database implementing a Library Management System. Users of  
the system are understood to be librarians (not book borrowers).

**Functional Requirements**

1. **Graphical User Interface (GUI) and Overall Design**  
   All interface with the Library (queries, updates, deletes, etc.) must be done from a graphical user interface of your original design. Your GUI application will interface with the Library database via an appropriate MySQL connector. Initial database creation and population may be done from command line or other admin tool. Overall design will be judged on usability first, look-and-feel secondary.
2. **Book Search and Availability**Using your GUI, be able to search for a book, given any combination of ISBN, title, and/  
   or Author(s).Your query should support substring matching. You should then display the  
   following in your search results:  
   • ISBN  
   • Book title  
   • Book author(s) (If displaying search results in table columns, author should display a  
   single derived value, even though author names are stored as composite attributes.). Nonperson authors (i.e. institute names, group names) should use the AUTHOR.Lname field.  
   • Branch info  
   - branch\_id  
   - branch\_name  
   - How many copies are owned/inventoried by a specified branch  
   - Book availability at each branch (i.e. How many copies not already checked out?).  
   Each book at a given branch should display on a single line.
3. **Book Loans  
   Checking Out Books**  
   • Using your GUI, be able to check out a book, given the combination of BOOK\_COPIES(Isbn, branch\_id) and BORROWER(Card\_no), i.e. create a new tuple in BOOK\_LOANS. Generate a new unique primary key for loan\_id. The date\_out should  
   be today’s date. The due\_date should be 14 days after the date\_out.  
   • Each BORROWER is permitted a maximum of 3 BOOK\_LOANS. If a BORROWER  
   already has 3 BOOK\_LOANS, then the checkout (i.e. create new BOOK\_LOANS tuple)  
   should fail and return a useful error message.  
   • If the number of BOOK\_LOANS for a given book at a branch already equals the  
   No\_of\_copies (i.e. There are no more book copies available at your library\_branch), then  
   the checkout should fail and return a useful error message.   
   **Checking In Books**  
   • Using your GUI, be able to check in a book. Be able to locate BOOK\_LOANS tuples by  
   searching on any of book\_id, Card\_no, and/or any part of BORROWER name. Once  
   located, provide a way of selecting one of potentially multiple results and a button (or  
   menu item) to check in (i.e. enter a value for date\_in in corresponding BOOK\_LOANS  
   tuple).
4. **Borrower Management**• Using your GUI, be able to create new borrowers in the system.  
   • All name, SSN, and address attributes are required to create a new account (i.e. value  
   must be not null).  
   • You must devise a way to automatically generate new card\_no primary keys for each new tuple that uses a compatible format with the existing borrower IDs.  
   • Borrowers are allowed to possess exactly one library card. If a new borrower is attempted with same SSN, then your system should reject and return a useful error message.
5. **Fines**• fine\_amt attribute is a dollar amount that should have two decimal places.  
   • paid attribute is a boolean value (or integer 0/1) that idicates whether a fine has been  
   paid.  
   • Fines are assessed at a rate of $0.25/day (twenty-five cents per day).  
   • You should provide a button, menu item, etc. that updates/refreshes entries in the FINES table. In reality, this would occur as a cron/batch script that executed daily.  
   • There are two scenarios for late books  
   1. Late books that have been returned — the fine will be [(the difference in days  
   between the due\_date and date\_in) \* $0.25].  
   2. Late book that are still out — the estimated fine will be [(the difference between the  
   due\_date and TODAY) \* $0.25].  
   • If a row already exists in FINES for a particular late BOOK\_LOANS record, then  
   - If paid == FALSE, do not create a new row, only update the fine\_amt if different than current value.  
   - If paid == TRUE, do nothing.  
   • Provide a mechanism for librarians to enter payment of fines (i.e. to update a FINES  
   record where paid == TRUE)  
   - Do not allow payment of a fine for books that are not yet returned.  
   - Display of Fines should be grouped by card\_no. i.e. SUM the fine\_amt for each  
   Borrower.  
   - Display of Fines should provide a mechanism to filter out previously paid fines (either  
   by default or choice).