Library Management System Design Document

High Level Structure:

The Library Management System is written in Web Programming Languages and executed on a local web server. The server environment is provided by MAMP, which includes Apache web server, MySQL database server and PHP.

The user interface is a web page, which contains menu navigator and corresponding modules, the homepage.html file is linked to a style.css file and a script.js file, which is responsible for the display style and webpage behaviors. The structure of different modules are nearly the same. They all have buttons for submission, blanks for input, initially hidden section for displaying message and result from the server.

The webpage calls AJAX function to communicate with the server via json data. The server calls PHP MySQLi functions to access MySQL data server.

Load Data Module:

File create_library.sql is responsible for creating the whole database, tables, column names and constraints. However, it does not contain any tuples initially. The create_library.sql file is imported to the database server via phpMyAdmin provided by MAMP to create the library database.

The users are supposed to click the ‘load data’ button on the home page to call AJAX on loadData.php file to upload book and borrower data to the database server. The main function of loadData.php is to read the given books.csv and borrowers.csv files line by line, extract attributes needed, map these attributes to corresponding tables in the given schema and upload data. Here table BOOK, BOOK_AUTHORS, AUTHORS and BORROWER are filled with given data. These tables are exactly the same as the description.

Book Search Module:

User is supposed to input any combination of book isbn, title and/or author(s) in the search bar to locate a book. The input key words should be separated by ‘*’, so that it would be easier to search tuples. Assume all the books having one of these key words matched their isbn, title and authors would be listed in the result.

First tuples in BOOKS is fetched, then using the isbn to fetch multiple authors, finally search whether there exists a tuple where data_in column is null, which indicates that this book is currently not available. These attributes are combined as a table, sent to and display on the webpage via AJAX.

Book Loans Module:

This module is divided into 2 submodules:

Check Out Module:

Only book isbn and borrower's card id are needed to check out a book. After these 2 strings are sent to the server, first check number of tuples in table BOOK_LOANS where the date_in column is null with the card id, which indicates the number of books borrowed by this borrower that are not returned yet. If the borrower owned 3 books not returned, he or she is not permitted to borrow new books any more. Second check whether the book having given isbn is available right now. Fetch tuples in BOOK_LOANS with null date_in attribute could yield the result. If previous 2 constraints are met, a new tuple in BOOK_LOANS is inserted. Assume the loan_id is accumulated from 1. So the newest tuple's loan_id is the number of existing tuples add 1. After the new loan record is inserted, the detail information including loan_id, isbn, card_id, date_out and due_date will be listed in a table in the message division. The date_out is the current date and due_date is current date add 14 days, the date_in column is default null until the book is returned.

Check In Module:

User is supposed to input key words including isbn, card id and/or borrower names in the search bar to fetch all related loan records where the book is not return yet (date_in is null). Assume that all tuples with attribute matching at least one of these key words will be fetched, and listed in a table with related attributes. The user once can select only one book by clicking on the check box at the end of a row to check in. In order to simulate the real life situation, the day elapse function is added along with the check in button. Click the 'add days' button and then click the 'check in' button, the check in date would be the number in the input box of days after current date. After checking in, the detail information including date_in would show as a table in the message section, and the record would disappear in the search result at the same time, the date_in column for this loan tuple is updated to current date add elapsing days as well.

Fines Module:

The display of fines is grouped by card_id, which should be the key word to fetch FINES records. User is supposed to input the card_id then click 'refresh' button to fetch records, current date, card_id and number of records would show in the message section. The sum of unpaid fines would show above detail loan records in the result section. The refresh mechanism is as follow: insert new fines records where books are returned but late; update existing fines records where books are late but just returned (date_in change from null to a date); update existing fines records where books are late and still not returned; insert new fines records where books are late and not returned; do nothing with the paid fines records.

To simulate a real-life situation, a day elapse function is added. When several days are added, some loan records where books are not returned will be late and appear in the result when click 'refresh' button. The user once can select one unpaid record to pay its fine. When the 'pay fines' button is clicked, the selected record would disappear and the total fine would be reduced by the paid fine, the paid column would be set to true and detail paid record information will show as a table in the message section. Notice that fine records for books that are not returned cannot be selected to pay, as the check boxes are disabled if the book is not returned. Once a fines record is paid, it would be filter out for this borrower.

Notice that whenever there is a date roll back (i.e. add days number reset to 0 or become less, which is impossible in real life situations), wrong simulation records would appear in the fines table, so click the 'reset' button to delete all records in fines and refresh again in this case.

Borrower Management Module:

Used for creating new borrowers in table BORROWER. The input fields are validated using the script as follow: first name and last name cannot be blank; SSN should not be blank and should have a format like XXX-XX-XXXX; address fields should not be blank; phone number should have a format like (XXX) XXX-XXXX or blank. When these inputs are validated and sent to the server, the server would check whether there exists a borrower having the same SSN first, the existing borrower will show as a table in the message section. Otherwise a new borrower is created. Assume the card_id has 6 digits and the first borrower has a card_id '000001', then the new borrower's card_id should be number of existing borrowers add 1. Once created, detail information including card_id will show as a table in the message section.