Library Management System

Software Design Specification

Version 2

October 19, 2023

Group 8
Allen Zhang
Alex Han
Juan Paulo Reyes

Prepared for
CS 250- Introduction to Software Systems

Instructor: Gus Hanna, Ph.D.
Fall 2023

Revision History

Date	Description	Author	Comments
10/1/23	LMS UML Diagrams	Allen Zhang Alex Han Juan Paulo Reyes	
10/19/23	No significant changes	Allen Zhang Alex Han Juan Paulo Reyes	

Document Approval

The following Software Requirements Specification has been accepted and approved by the following:

Signature	Printed Name	Title	Date
	<Your Name>	Software Eng.	
	Dr. Gus Hanna	Instructor, CS 250	

Table of Contents

1. System Description

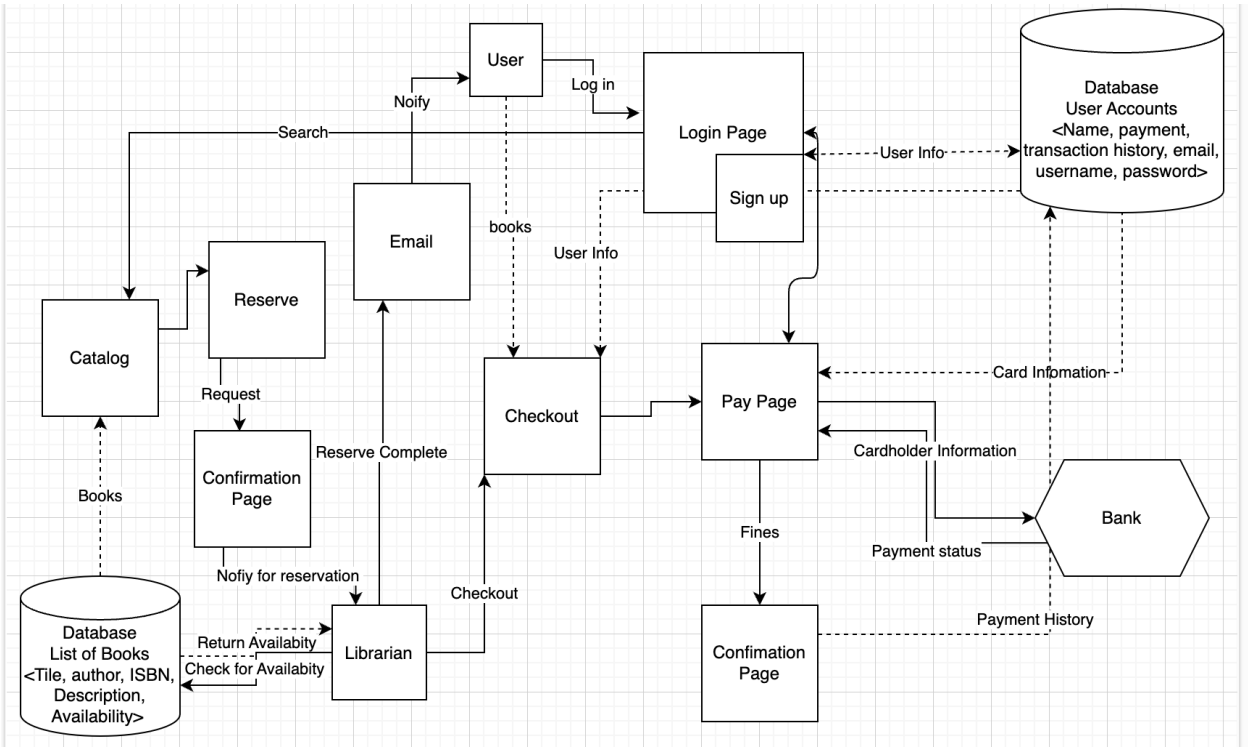
The software product is produced as a “Library Management System.” This software is intended to serve as a comprehensive digital platform for library management and member services at the BPL. The goal is to streamline and enhance library management procedures, providing library staff and users with a user-friendly and efficient way of accessing library materials and services.

The LMS will have various functionalities, allowing users to search for books. It will provide information on any books owned by the library, featuring titles, authors, descriptions, and real-time availability. The users can check out and reserve books and manage their accounts through profile customization. Also, they will be informed of due dates and reservation pick-up dates via email. This software will allow the librarians to manage their inventory, add new books, update current information, and maintain clear contact with the users. The system will not progress financial transactions or any payment for the library materials, such as late fees or fines.

The LMS has unique benefits, objectives, and goals to improve the user's experience by providing an intuitive and user-friendly interface for searching and accessing library materials. It aims to improve library staff’s efficiency in managing inventory and providing client service.

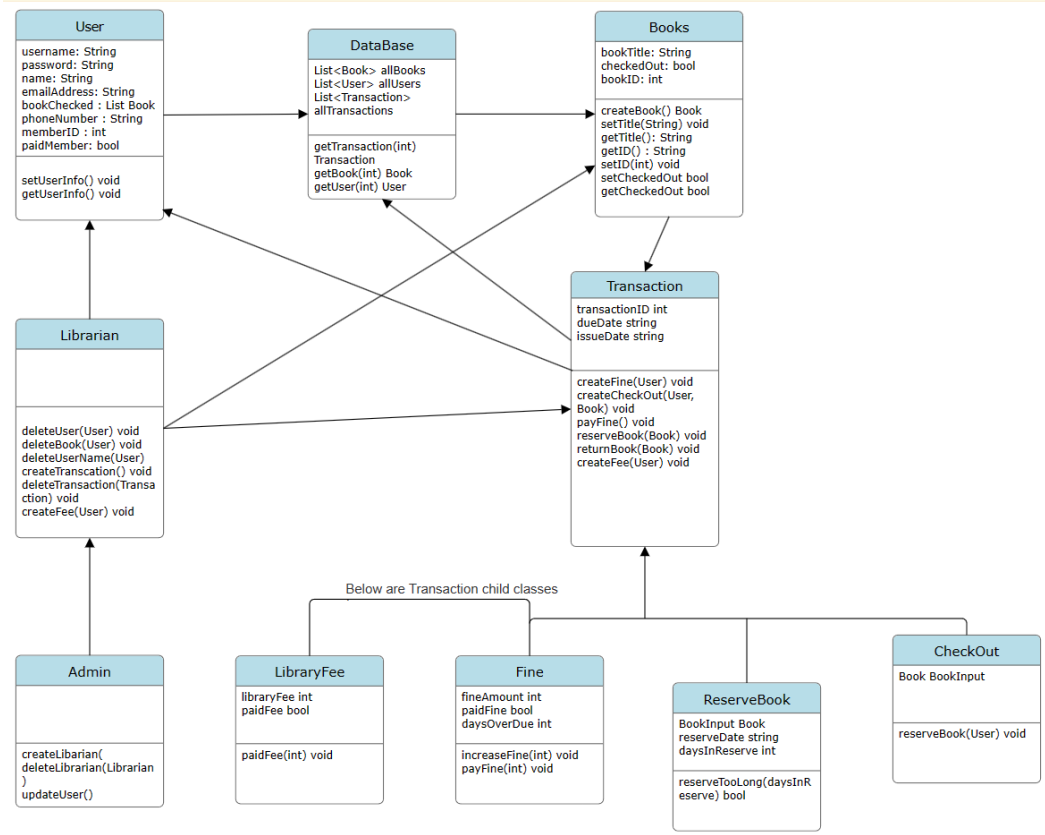
2. Software Architecture Overview

2.1. Architectural diagram of all major components



2.2. UML Class Diagram

2.2.1 UML Diagram with planned functions and attributes



2.3. Description of Classes

2.3.1 User- This class had information about the standard user which included their personal and login information. As well as what they have checked, their ID, and whether or not they are a paid member.
2.3.2 Librarian- This class inherits the traits of the user, and is given access to more methods that are related to running the library.
2.3.3 Admin, this class inherits from LibrarianInfo, which also grants them User operations and attributes. As admins they are able to delete users and librarians.
2.3.4 Database, they are supposed to store all information about the books, transactions, and users.
2.3.5 Books, this class has everything relating to books in the library.
2.3.6 Transaction, this class consists of transactions between the library and users.
2.3.7, LibraryFee, this class is supposed to be about any new members fees.
2.3.8 Fine, this class handles any fines that users have.
2.3.9, ReserveBook, this class is supposed to allow for books to be reserved.
2.3.10 CheckOut, this class is about allowing check outs of library books.

2.4. Description of Attributes

2.4.1.1 userName, this string variable consist the user name of library users, with the right password people can login to their accounts
2.4.1.2, password, this string variable is the password of an individual user's account.
2.4.1.3 name, this string is the name of the user
2.4.1.4 emailAddress, this string holds the email address the user
2.4.1.5 bookChecked, this List holds all checked out books of the user.
2.4.1.6 phoneNumber, this string holds the phone number of the user
2.4.1.7 memberID, this integer holds the member ID of the user.
2.4.1.8 paidMember, this boolean checks to see if the user has paid for library members.

2.4.4.1 bookTitle, this string contains the title of a book
2.4.4.2 checkOut, this boolean contains whether or not this book has been checked out
2.4.4.3 bookID, this integer contains the id of the book

2.4.5.1 allBooks, this list is supposed to hold all books of the library.
2.4.5.2 allUsers, this list is supposed to contain all users of the library.
2.4.5.3 allTranscations, this list is supposed to contain all transactions between user and the library.

2.4.6.1 transactionID, this integer is supposed to hold the ID of the transaction
2.4.6.2 dueDate, this string contains the due date of any transaction
2.4.6.3 issueDate, this string contains the date when the transaction occurred.

2.4.7.1 libraryFee, this contains the amount due from memberships.
2.4.7.2 paidFee, this contains a boolean if the fee was paid.

2.4.8.1 fineAmount, this integer contains the amount of money owed in fines.
2.4.8.2 paidFine, this boolean contains if the fine was paid.
2.4.8.3 daysOverDue, this integer contains the number of days

2.4.9.1 BookInput, this book contains the book that is supposed to be reserved
2.4.9.2 reserveDate, this string contains date when the book was reserved
2.4.9.3 daysInReverse, this integer contains the number of days it was reverse for

2.4.10.1 BookInput, this contains a Book, which is suppose to be the one checked out.

2.5. Description of Operations

2.5.1.1. setUserInfo() this function sets the values of user variables via by user input
2.5.1.2. getUserInto() this function gets the values of the user variable.

2.5.1.3. deleteUser(User) this function deletes Users.
2.5.1.4. deleteBook(Book) this function delete Books
2.5.1.5. deleteUserName(User) this function delete User name
2.5.1.6. createTransaction() this function creates a new transaction
2.5.1.7. deleteTransaction(Transaction) this function delete a transaction
2.5.1.8. createFee(user) this function creates a new fee for a user
2.5.1.9. createLibarian() this function creates a new Librarian
2.5.1.10. deleteLibarian(Librarian) this function deletes a Librarian
2.5.1.11. updateUsers() this gets updates on Users
2.5.1.12. getTransaction(int) this get a specific transaction based on int input
2.5.1.13. getBook(int) this gets a specific book based on int input
2.5.1.14. getUser(int) this gets a specific user based on int input
2.5.1.15. createBook() this creates a new book
2.5.1.16. setTitle(string) this sets the title of a book to the string input
2.5.1.17. getTitle() this gets the title of a book
2.5.1.18. getID() this gets the ID of a book
2.5.1.19. setID(int) this sets the ID of a book
2.5.1.20. setCheckedOut() this sets a book to being checked out or not
2.5.1.21. getCheckedOut() this checks if the book has been checked out
2.5.1.22. createFine(User) this creates a fine for a user
2.5.1.23. createCheckOut this creates a check out for a input user with a book input.
2.5.1.24. payFine() this pays the fine
2.5.1.25. reserveBook(Book) this reserves a book
2.5.1.26. returnBook(Book) this returns a book back to the library
2.5.1.27. createFee(User) this creates a fee for a user
2.5.1.28. paidFee(int) this pays a fee by an amount given by input.
2.5.1.29. increaseFine(int) this increases a fine by an amount given by input.
2.5.1.30. reserveTooLong(daysInReserve) this function checks if the a book in reserve has been in reserve for too long
2.5.1.31. reserveBook(User) this reserves a book for a user.

3. Development Plan and Timeline

Figure 3-1

Figure 3-1 Is the proposed timeline for the development of the LMS.

3.1 Partitioning of Tasks

The Development Team (DT) will be managed by a Project Manager (PM) who oversees production through the Team Leaders (TL). The TLs will be responsible for the design and functionality of any newly developed code and integration of pre-existing code with the new system. The TL will have Design Leads (DL) who will be formulating use case testing, security testing and debugging. Subsequently the Core Developers (CD) will be the ones modifying previously used code for integration and developing new code for any new use cases or features.

The development team will be working closely with the Administrative Technicians (AT) and will generate a list of requirements in the form of required use cases, features and any monitoring requirements needed by the client. Any changes or newly submitted information will be through the AT.

3.2 Team Member Responsibilities

For the client, any changes required or new features requested will be made through the Administrative Team during the development cycle.

For the DT, any updates, changes to documentation, and changes to scope of work will be determined by the PM. During the development cycle, any questions between team members should be brought up to the immediate supervisor, going higher only if the question cannot be answered. The PM shall determine the TL and the scope of their work, ensuring that each requested use case and feature is covered. The TL will be responsible for the individual development of these use cases. As the CDs complete each part of the software, the DL will generate security and unit testing for each possible security flaw or use case to ensure minimal bugs and security holes on delivery.

Once each part of the project has been finished and polished, the TLs will inform the PM and design a closed system unit test for implementation. Once testing has been completed satisfactorily, the PM will sign off for completion. The PM will then order system integration in which the TLs and DLs will work together to integrate the software with the existing database and network and the AT will train one member for system use to train the rest of the AT.

The AT will then verify all use cases and features are present and sign off on delivery along with the PM, after which the development is considered complete and the development team will continue to provide support in the form of maintenance and security updates thereafter as prescribed in the initial contract.