Department of Computing
電子計算學系

THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

COMP2411 Database Systems (Fall 2022)

Group 4

Library Management System

Project Report

CHEN Derun (21098424d)

JIANG Guanlin (21093962d)

KWOK Hin Chi (20060241d)

LIU Minghao (21096308d)

YE Haowen (21098829d)

ZHANG Wengyu (21098431d)

Table of Contents

1. INTRODUCTION

A **Library Management System (LMS)** is designed to manage all the functions of a library. It helps librarians and administrators maintain the database of new books and the books that are borrowed by visitors along with their due dates. Without a computerized system to store the data, it is hard to find books, issue/reissue books, and manage all the data in an orderly and efficient way. Viewing this situation, our LMS can completely automate library activities to better maintain, organize, and handle countless books systematically. For example, it tracks the records of the number of books in the library, how many books are issued, or how many books have been returned or renewed or late fine charges. To improve the user experience, administrators can extract the reporting module of the library management system, including registration lists, book lists, circulation, and return reports, for further analysis and optimization of the system.

Hence, these modules help librarians manage their libraries in a more convenient and efficient way, and without the loss of book records or membership records, while ensuring immediate and accurate data about any type of book, thus saving a lot of time and effort.

2. SYSTEM ANALYSIS

2.1 Problems Without an Integrated LMS

- Time/Cost/Space Consuming

Storing physical documents is more expensive, and in addition to creating paper documents, maintaining them, sorting, and processing them is more costly. Archiving documents requires the labor cost of organizing and retrieving them.

- File damaged/ lost

Paper files are much more vulnerable than digital files that are backed up to multiple locations and have security. Without viable backups, disasters such as fires and floods can destroy records and make them difficult to recover.

In addition, tangible documents are not as secure as digital files because they can be easily accessed by anyone with access. Paper documents cannot have a consistent sign-in and sign-out system with no record of who has seen it made changes to it, or copied it. With digital files, it is

possible to attach an audit trail to each document so that administrators can see who looked at it and on what date and time.

2.2 System requirements

2.2.1 Functional Requirement

User character: Administrator/ Librarians

Given the information, the main functions can be divided into several parts, shown as follows:

- **Register New Books Function**

  Allows adding new books to the library.

  Function Requirements:

  1. The system should be able to validate input information.
  2. The system should be able to not allow two books to have the same Book ID.
  3. The system should be able to store more than one copy of each book, and a book can be published by different publishers that are identified by ISBN.
  4. The system should maintain the number of renting and placing of a book instance.

- **Search Books Function**

  Search books by Book ID, Book Name, Author, or Category.

  Functional requirements:

  1. The system should be able to search the database according to the selected search type.
  2. The system should be able to list all the information of the return list of searching.

- **Issuing & Returning & Placing Books Function**

  Allows issuing, returning, and placing books.

  Functional requirements:

  1. The system should be able to enter the issue information in the database.
  2. The system should be able to update the number of books.

3. The system should be able to search for the availability of books before issuing them.
4. The system should be able to enter information about the date of issuing and returning books.
5. The system should be able to provide a reserved book function to users if the desired book is not available in the library currently, no book can be reserved if it is available in the library.
6. The system should be able to deactivate a user account if s/he does not return books after a specific period of time passes.
7. The system should be able to activate a deactivated user account after s/he returns all books.
8. The system should be able to check records of books checked out and placed on hold, impling books can be reserved by a patron to make sure the book is there when he/she gets to the library to check it out.
9. The system should be able to notify the library users when the desired book becomes available and reminders that a book should be returned to the library. Both could be sent by email and/or when a patron logs in to the LMS.
10. The system should cancel a placed book if the book has been placed for more than 7 days. If some users have reserved this book, this book should be placed again for the first place user on the waiting list. Otherwise, the book should be available for free rent.
11. The system should not provide any renting and reserve function to a deactivated user until the user is activated.

- **Refresh Deactivate/Activate User Account Function**
    1. The system should be able to check all expired rent books and perform the deactivation of the corresponding user accounts frequently.
    2. The system should be able to check all deactivated users who have already returned all books, and activate the user back frequently.
- **Cancel Reserved/Placed Book Fiction**

1. The system should be able to cancel the reserved book from a user, which is side-effect free.
2. The system should be able to cancel the placed book from a user, which requires a check on the waiting list of the reserved book, and put the first place user in the placed book record.

- **Notification Function**
    1. The system should be able to send reminders to the user when any status changes on the user or associated book.
    2. The system should be able to send a notification to the librarian when any status changes to any user, book, or the whole system.
- **Generate Report Function**

    Provides an analysis report to the manager to review the system.

    Functional requirements:

    1. Generate reports such as the number of stored books, book information, and user accounts.
    2. Generate analysis report in terminal

2.2.2 Non-Functional Requirement

- Efficiency

The implementation of LMS should be easy for librarians to access the library metadata with fast book searching and transactions.

- Reliability

LMS should provide accurate membership information, report generation, book transactions, and searches.

- Usability

The system is designed to be a user-friendly environment where librarians can manage them easily and efficiently.

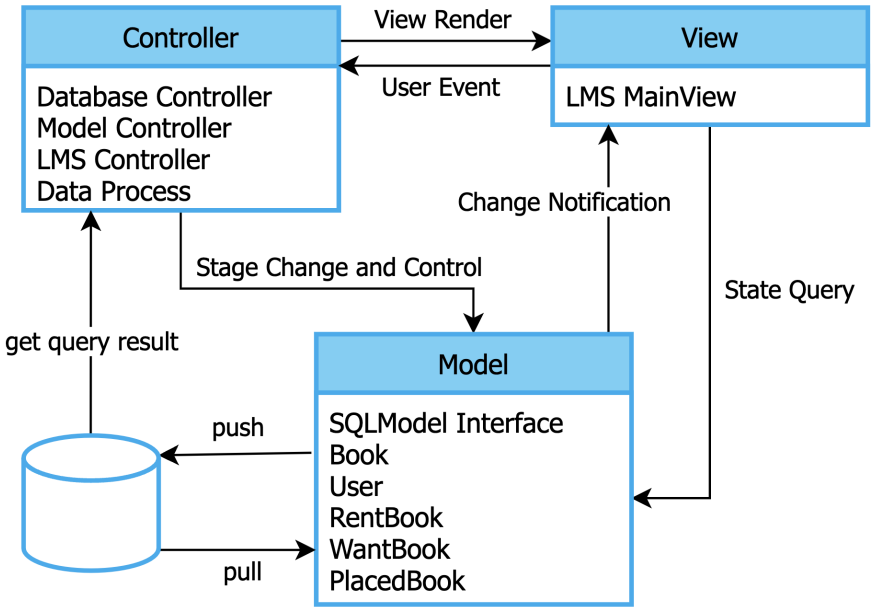
2.3 Software in Used

- IntelliJ IDEA: Implement **JAVA code** with **MVC Architecture**
- Oracle Database: Store schema **USER_ACCOUNT, BOOK, WANT_BOOK, HAS_RENT, HAS_PLACED**
- Draw.io: formulate our **ER Diagram** and **Logic Diagram**.

2.4 System Objectives

We propose our system as an integrated, user-friendly, and compatible library automation system that can fully computerize the internal operations of the library. This library management system is designed to provide instant and accurate data for any type of book, thus saving a lot of time and effort. This library management system is considered to be intuitive, efficient, and compliant so that books can be found, issued/re-issued and all data can be managed efficiently and in an organized manner.

3. SYSTEM DESIGN



Model-View-Controller (MVC)[1] specifies that an application should include a data model, presentation information, and control information.

The MVC architecture helps our development by offering high cohesion inside every part as well as low coupling between each part. This architecture also results in good work allocation among the project team for better teamwork.

We have implemented the MVC architecture to design the model, in which the application logic is separated from the user interface.

The MVC pattern architecture includes three parts:

- Model: In the business part of the application, it is an object that carries data and can also contain logic that updates the controller when the data changes.
- View: The presentation part of the application, where it visualizes the data contained in the model.
- Controller: Working on models and views, it manages the flow of the application, i.e. the flow of data in the model object, and updates the view when the data changes.

The model contains data classes, the view is used to display data, and the controller contains servlets. User requests are handled as follows.

- User interacts with the view, the view sends the user event to the controller; The view can query the update from the model directly as well;
- Controllers handle the user event and map the event to functions on models; as well as select different views. The controller can also get query result from the LMS Database
- The model implements all the functions required by the controller and update the application states; as well as notice the view changes; The model will also pull data from and push data to the Database for retrieve and update;

3.1 Database Design

3.1.1 User Table

```
CREATE TABLE USER_ACCOUNT(
   accountID NUMBER(10) NOT NULL,
   accountStatus CHAR(1) CHECK (accountStatus in ( 'T', 'F' )) NOT NULL,
   notification VARCHAR(8000) NOT NULL,
   PRIMARY KEY (accountID)
);
```

For store LMS user account entities:

- **User account ID as the primary key;**
- **User account status, T for activated, F for deactivated;**
- **User notification for store all notice to user;**

3.1.2 Book Table

```
CREATE TABLE BOOK(
   bookID NUMBER(15) NOT NULL,
   ISBN VARCHAR (15) NOT NULL,
   bookName VARCHAR(100) NOT NULL,
   author VARCHAR(100) NOT NULL,
   bookCategory VARCHAR(100) NOT NULL,
   bookRentNum NUMBER(5) NOT NULL,
   bookWantNum NUMBER(5) NOT NULL,
   PRIMARY KEY (bookID)
);
```

For store LMS book account entities:

- **Book ID as the primary key;**
- **Book ISBN;**
- **Book name;**
- **Book author name;**
- **Book category;**
- **The number of the book be rented;**
- **The number of the be placed;**

3.1.3 Has_Rent Table

```
CREATE TABLE HAS_RENT(
   accountID NUMBER(10) NOT NULL,
   bookID NUMBER(15) NOT NULL,
   rentTime VARCHAR(10) NOT NULL,
   PRIMARY KEY (bookID,accountID),
   FOREIGN KEY (bookID) REFERENCES BOOK(bookID),
   FOREIGN KEY (accountID) REFERENCES USER_ACCOUNT(accountID)
);
```

For store the rent book relation between Book and User:

- **The account ID for the rent user (foreign key);**
- **The book ID for the rent book (foreign key;**
- **The date for rent;**
- **The {account ID, book ID} is the primary key;**

3.1.4 Want_Book Table

```
CREATE TABLE WANT_BOOK(
   accountID NUMBER(10) NOT NULL,
   ISBN VARCHAR (15) NOT NULL,
   wantTime VARCHAR(10) NOT NULL,
   PRIMARY KEY (ISBN,accountID),
   FOREIGN KEY (accountID) REFERENCES USER_ACCOUNT(accountID)
);
```

For store the reserved (want) book relation between Book ISBN and User:

- **The account ID for the want user (foreign key);**
- **The ISBN for the want book ISBN;**
- **The date for want;**
- **The {account ID, ISBN} is the primary key;**

3.1.5 Has_Placed Table

```
CREATE TABLE HAS_PLACED(
   accountID NUMBER(10) NOT NULL,
   bookID NUMBER(15) NOT NULL,
   placeTime VARCHAR(10) NOT NULL,
   PRIMARY KEY (bookID,accountID),
   FOREIGN KEY (bookID) REFERENCES BOOK(bookID),
   FOREIGN KEY (accountID) REFERENCES USER_ACCOUNT(accountID)
);
```

For store the data of placed book relation between Book and User:

- **The account ID for the placed user (foreign key);**
- **The book ID for the placed book (foreign key);**
- **The date for placed;**
- **The {account ID, book ID} is the primary key;**

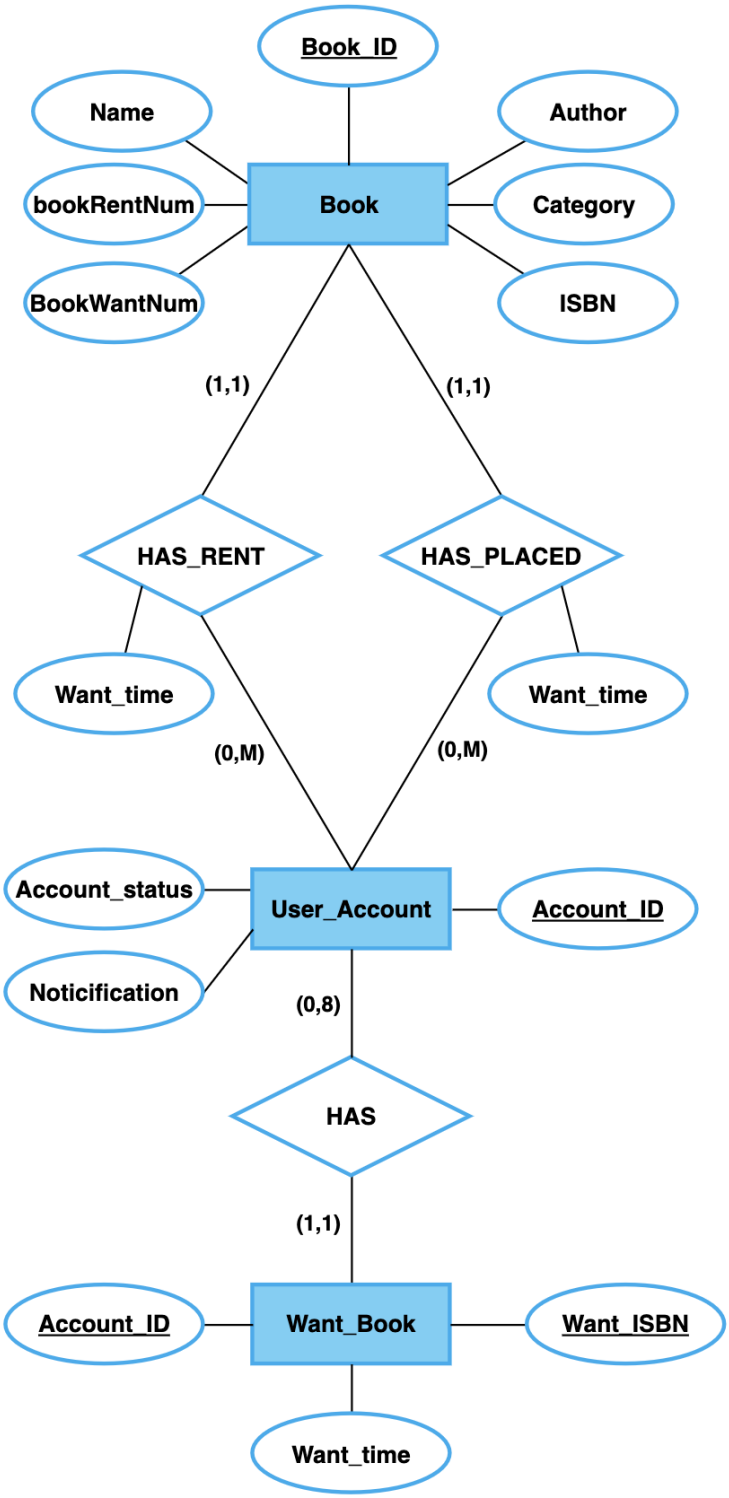
Please find the testing data in the *./testing-data.sql* file

3.1.6 Normal Form(NF) Analysis

For the Database design, it satisfies the BCNF as a perfect database design:

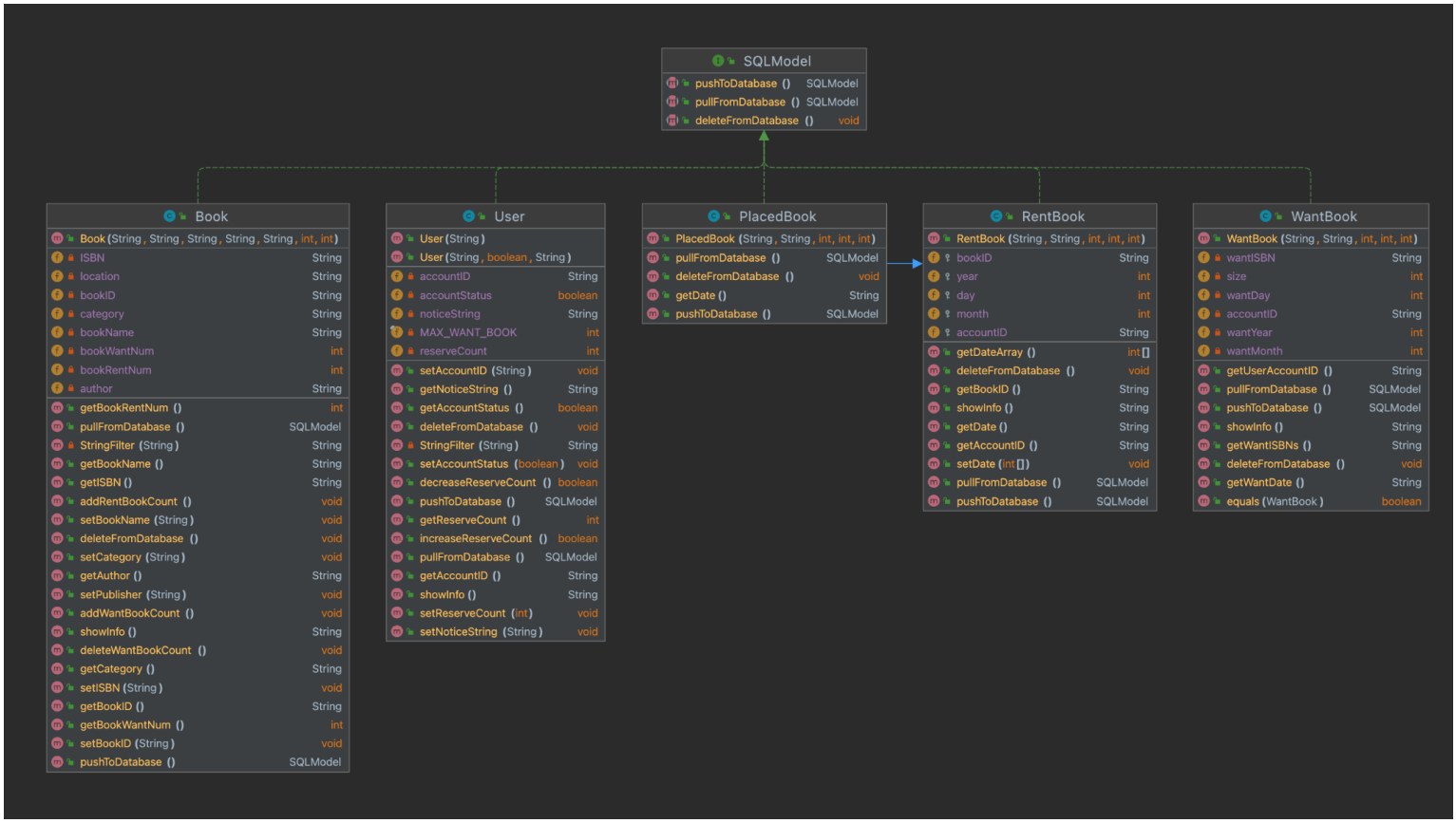
- We follow the 1NF that no attribute domain has no relations as elements, for each data record in the database has unique and non-nested attributes.
- We follow the 2NF that each column in the database relies on its primary key without reliable relation with non-primary key attributes. For each table, we have separate functions with a book list, user list, rent book function list, reserve book function list, and placed book function list to distinguish the functions as individual parts.
- We follow the 3NF that has no transitive attribute dependency among the attributes. And it also follows the BCNF that combines book and user attributes as the primary key to construct the rent book, reserved book, and placed book functions.

3.1.7 ER Diagram



3.2 Model Design

Please find the model code in the *./src/model* folder



We designed an SQLModel interface as the abstraction of all model classes, which provides three abstract methods for pull, push and delete two-way operations between model instance fields and the Database. And then let all model classes implement the interface.

3.2.1 Book.java

In this model, we set basic information about a book, which are the Book ID, Book ISBN, Book Name, Book Author, and Book Category, Number of Book Rent, and Number of Book Want in this library. We set the setter to put the data into the book and also set the getter to allow other functional classes to get the information from the book. The database pusher can push the book updates to the database, the database puller can pull the book updates from the database, and the deleter can delete some useless information or unavailable books in the database.

3.2.2 User.java

In this model, we set up basic information about the user, i.e. Account ID, Account Notice, Account Status, and Number of Want Books. Setter and getter will update the account information and fetch information from the user account. The database enabler will create or update user information in the database, the puller will fetch information from the database, and the deleter will delete useless information or the account is deleted by the user. Also, if a user rents some books, or if a book from the list of books they want to rent is available for rent, they will receive a notification from the library management system, which will also be updated in the database.

3.2.3 RentBook.java

The significance of the book rental model is to update whether the book can be rented or not. So, in this model, we set the Account ID, Book ID, and timestamp which includes the year, month, and day of the rental. In this model, we also set setters and getters for other models or controllers to use them. The database pusher, puller, and deleter are set in the model to control the extraction and manipulation of the database.

3.2.4 WantBook.java

The meaning of the wanted book model is to update whether the book is added to the wanted book list or can be rented from the wanted book list. This model maintains the Account ID, Book ISBN, and timestamp which includes the year, month, and day of the reserve. In this model, it is necessary to operate on the database to get the status of the book and set the getter for other functions and update the wanted book table from the database. In addition, setting and getting date data is necessary for the desired book model to calculate the date of the savings.

3.2.5 PlacedBook.java

The point of the put book model is to update the book that has been returned to the library and can be sent to the user who wants the book. This model maintains the Account ID, Book ID and timestamp which includes the year, month, and day of the placing. In this model, we need to get the list of wanted books from the want book table in the database. The operations in this model

are all related to the database, including extracting from the database, pushing to the database, and deleting some useless information from the database.

3.3 Data Structures

3.3.1 ArrayList

This data structure will store the intermediate data obtained from the model controller. For example, the search function by Book Name, Book ISBN, Book Author, and Book Category will return a list of book information, so it will be better to use ArrayList to store it here, also ArrayList is based on a dynamic array and we do not need to provide the number of books.

3.3.2 HashMap

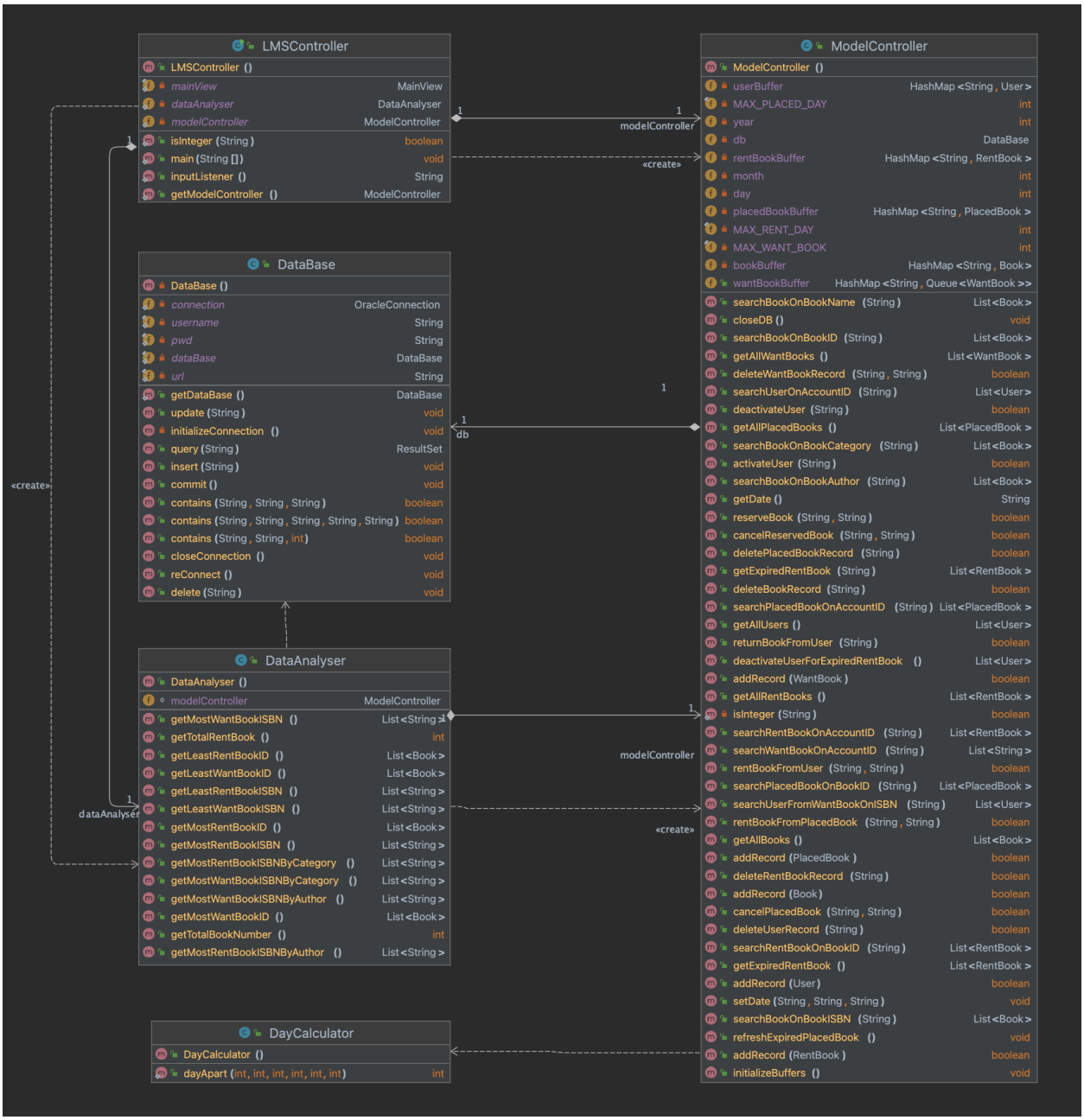
This data structure will store temporary data from the database side to the application side or from the application side to the database side, more like a buffer between the model and the database, which connects the two sides of the data transfer. For example, in our model controller, one of the HashMap – book buffers handle the data from the database to the model – book and also handles the data from the model to the database for updating the book table.

3.3.3 Queue

This data structure is used in the function that has a list of wantBook instances, and the Queue can take out the user's first order when the book is available to the user. In this project, the queue is implemented by a Linked List, where it is better to use remove the first and insert the last.

3.4 Controller Design

Please find the controller code in the *./src/controller* folder



3.4.1 DataBase controller with Singleton design pattern

In this DataBase.java controller, we use **JDBC**[2] to connect to the Oracle Database Server. Based on the database connection, we encapsulate initialization and closing methods, as well as several SQL queries and update template statements into methods, such as updater, deleter, inserter, container, committer, and searcher, which can be used as a utility by other controllers and model classes to communicate with the database.

Moreover, we apply the *Singleton design pattern*[3] in this Database Controller by setting the constructor as private and encapsulating an instance of itself as its field, which ensures that only one exists of this class instance, and one and only one access to this instance. This design pattern guarantees and protects the connection and access to the database within a particular period of time.

3.4.2 Model Controller

In this controller, we use 5 HashMaps as buffers to handle temporary data pushed into the database tables or pulled out of the database and returned to the model. This controller uses the functionality of the Database controller to connect to the **Oracle Database Server** and serve as a bridge between the two parties. The functions included are, searching for books and user information, determining the availability of some status, manipulating records in the database tables, and the list of rented and wanted books, and other utility functions as listed below.

3.4.2.1 Account Activation/Deactivation Function

For account status, we stipulate that users who didn't return the book back to the library within 14 days after renting the book will be banned automatically with account deactivation. And the deactivation only could be canceled when users return all the expired books back to the library.

3.4.2.2 Notification Function

For the notification function, we stipulate that for every rental, reservation, placement, cancellation of the reservation, cancellation of placement, and remaining rental by the user, we record it in the notification as a personal history check.

3.4.2.3 Add Record and Delete Record

For all book operations on rentals, reservations, and placements, we have a function that adds a record of the operation to each buffer so that the current state can be changed to another state when needed by the corresponding record operation.

3.4.2.4 Search books function

The search function is to retrieve book information by book ID, book title, author, category, and book ISBN to achieve the correct search. In addition, it also supports searching all books and users' information.

3.4.2.5 Books Operation

For operational books, users can rent from the library, reserve a book, rent a book from a placed book, and cancel a reserved book, whether it is in the rented book list or the placed book list.

3.4.2.6 Refresh Expired Placed Book

For placed-on-hold books for users after a book is placed on hold more than the maximum placed day - 7 days, the book is no longer placed for that user. The model controller will check whether any user is on the waiting list of the book, if yes, then the user in the first place on the list will be assigned to that book and put the book placed for that new holder user. If no user is on the waiting list, then the book will be set as available and put back into the library for free rent.

3.4.2.6 Get Expired Rent Book & Deactivate User For Expired RentBook

For expired rent books for users after the book is rented for more than the maximum rent day - 14 days, the books will be set to the expired rent book list with deactivation punishment for users that they couldn’t rent any books during this period until they return back the book. After returning the book, the users’ status will automatically be set to activate and remove the book record from expired rent books, and users could rent other books.

3.4.2.7 Set Current Date

The model controller provides a function for setting the current date, including year, month, and day.

3.4.2.8 Reset From Database

The model controller provides a set of functions to maintain the buffer of intermediate data between the LMS and the database. The model controller is able to refresh the newly queried data from the database and update buffers to maintain normal operation in LMS.

3.4.3 LMS Controller

The LMS Controller is the main entrance of the entire LMS.

The LMS controller is mainly responsible for the operation of the library system as a CLI operation for the librarians to operate for the users. We have input listeners to monitor the inputs and determine the corresponding functions for the operation by several input choices, and we have encapsulated these methods in the Model Controller section.

3.4.4 Day Calculator

In Day Calculator[4], we have implemented a date calculation function between two days to detect if a book rented by a user is expired and if a book already placed in the list of placed books is expired and not rented by the user. It is also possible to calculate the time between months and even years.

3.4.5 Analysis Report

Please find the analysis report code in the *./src/controller/DataAnalyser.java* file

In terms of analysis report, we implemented an analysis report function to generate the time when the report was created, analyze the overall number of books, the overall number of books rented, and the number of most and least popular books selected from different search sorts to present the popular and unpopular books from different sorting methods. We can also get the most rented book with its bookID to assume it as the maximum loss that needs to be maintained by librarians. (Its view design can be found in **3.6.3 View of Analysis Report** section)

3.4.5.1 Analysis Report Design Rationale

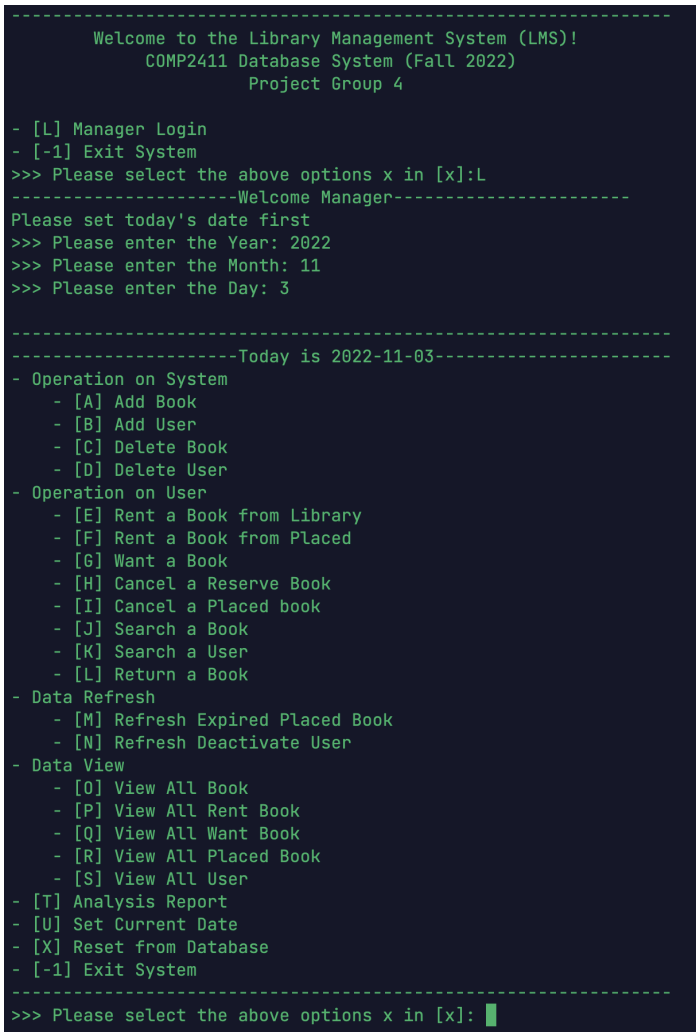
The report contains with several **analyses**, where:

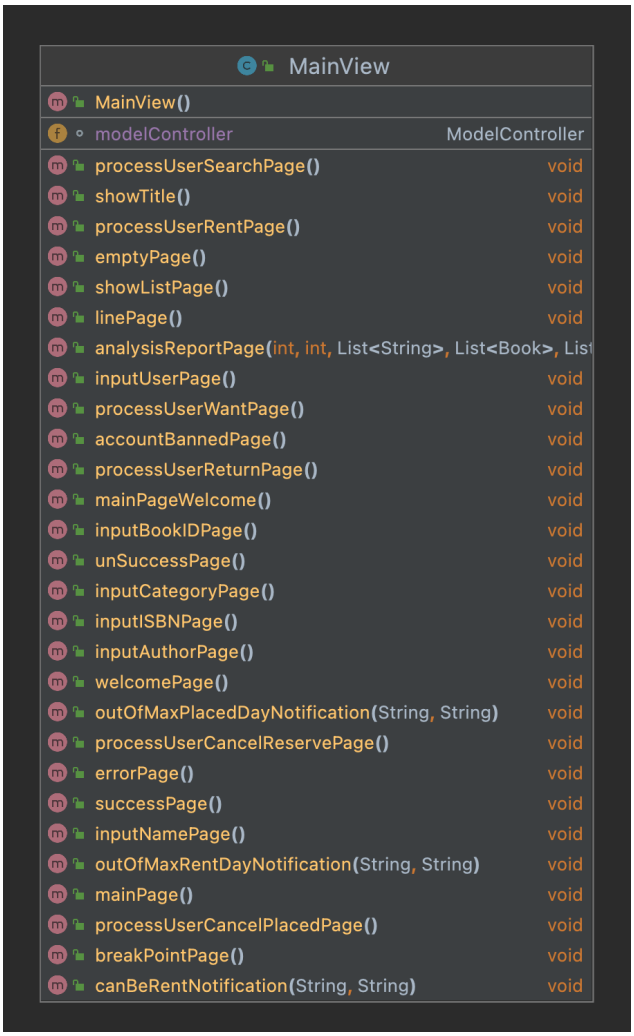
- ***Total book & total rent book*** is used to count the total number of the books in the library and the total number of books rented out, which allows users to get the most basic information about the library.
- ***Most rent book & wanted book*** is used to analyze the most rented out or most wanted books in the library, reflecting the rental or wanting of the most popular books.
- ***Least rent book & want book*** is used to analyze the least rented out or least wanted books in the library, reflecting the rental or wanting of the least popular books.
- ***Most rent book instances*** are used to analyze the specific book most rented out in the library, reminding the administrator that the book needs to be checked for possible damage.
- ***Most want book instances*** are used to analyze the specific book most wanted by the user in the library.
- ***Least rent book instances*** are used to analyze the specific book least rented out in the library, implying that the book may not be very well received by readers, or the position of the book in the library is not so convenient for users to find, or the book may have a slightly damaged appearance.
- ***Least want book instances*** are used to analyze the specific book least wanted by the user in the library.
- ***Most rent & want book by author*** is used to analyze the most rented or most wanted books by the same author in the library, reflecting the most popular books by this author.
- ***Most rent & want book by category*** is used to analyze the most rented or most wanted books by the same category in the library, reflecting the most popular books by this category.

3.5 View Design

Please find the view data in the *./src/view* folder

Overview of the CLI Page and MainView Class





3.5.1 Layout

- In order to show the whole Command Line User Interface (CLI) Page directly, we use the function **System.out.println()** to print related information line by line, which consists of the title, the guide of operation on the LMS system and descriptive statements.
- To avoid visual confusion, a string of "-" is used to separate the different sections and serve as a subheading for the same types of functions.

- Also, every function consists of a specific format, which is the **"[option]"** combined with the corresponding description statement, such as the **option "[A]"** and **the statement "Add Book"** under **"Operation on System"**.
- At the same time, indentation is used under each operation statement for a fast and efficient operation, which allows the manager to clearly recognize the specific function that belongs to each operation.
- **">>>"** always leads to an input statement, which asks the manager to enter an option to do operations.

3.5.2 Abstraction

The view is designed with several **Pages/ Sections,** where:

- ***welcomePage*** *with welcome words* in the middle of the page to indicate that the librarian is welcome to the LMS and that s/he can successfully open the system.
- ***loginPage*** *with manager login* reminder.
- ***mainPage*** *with all operations guides (i.e. Operation on System, Operation on User, Data Refresh, Data View)*, and each page is combined with some sub-pages. When the manager enters an option, it will jump to and show the corresponding page.
- ***otherPage*** *with other features guides (i.e. Analysis Report, Set Current Date, Reset from Database)* provides options to show the corresponding page.

3.5.3 View of Analysis Report

```
------------------------------------------------------------
                ANALYSIS REPORT
             Create Time: 2022-11-05
------------------------------------------------------------
             Books Status Information
------------------------------------------------------------
Total Number of Books: 17
Total Number of Books Already Borrowed: 184
Most Rent Book:
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Author]: Jane Austen[Category]: Novel [Times of Rent]: 37
Most Rent Book Instance:
  - [Book ID]:10002 [Book ISBN]: 0-02 [Book Name]: Harry Potter and the Philosopher Stone [Author]: J.K. Rowling [Category]: Fantasy [Time of Rent]: 31 [Time of Want]: 2
Least Rent Book:
  - [Book Name]: The Lion, the Witch and the Wardrobe [Book ISBN]: 0-05 [Author]: C.S. Lewis[Category]: Fantasy [Times of Rent]: 1
Least Rent Book Instance:
  - [Book ID]:10005 [Book ISBN]: 0-05 [Book Name]: The Lion, the Witch and the Wardrobe [Author]: C.S. Lewis [Category]: Fantasy [Time of Rent]: 1 [Time of Want]: 10
Most Want Book:
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Author]: Jane Austen[Category]: Novel [Times of Rent]: 15
Most Want Book Instance:
  - [Book ID]:10003 [Book ISBN]: 0-03 [Book Name]: The Hobbit [Author]: J.R.R. Tolkien [Category]: Fantasy [Time of Rent]: 3 [Time of Want]: 12
Least Want Book
  - [Book Name]: The Chronicles of Narnia [Book ISBN]: 0-04 [Author]: C.S. Lewis[Category]: Fantasy [Times of Want]: 0
Least Want Book Instance:
  - [Book ID]:10004 [Book ISBN]: 0-04 [Book Name]: The Chronicles of Narnia [Author]: C.S. Lewis [Category]: Fantasy [Time of Rent]: 5 [Time of Want]: 0
Most Rent Book By Category:
  - [Book Name]: The Adventures of Tom Sawyer [Book ISBN]: 1-10 [Category]: Story [Times of Rent]: 23
  - [Book Name]: Harry Potter and the Philosopher Stone [Book ISBN]: 0-02 [Category]: Fantasy [Times of Rent]: 31
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Category]: Novel [Times of Rent]: 37
Most Rent Book By Author:
  - [Book Name]: The Hobbit [Book ISBN]: 0-03 [Author]: J.R.R. Tolkien [Times of Rent]: 26
  - [Book Name]: Butterball [Book ISBN]: 1-08 [Author]: Henry-Albert-Guy de Maupassant [Times of Rent]: 24
  - [Book Name]: The Chronicles of Narnia [Book ISBN]: 0-04 [Author]: C.S. Lewis [Times of Rent]: 5
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Author]: Jane Austen [Times of Rent]: 37
  - [Book Name]: The Little Prince [Book ISBN]: 0-06 [Author]: Antoine de Saint [Times of Rent]: 5
  - [Book Name]: Harry Potter and the Philosopher Stone [Book ISBN]: 0-02 [Author]: J.K. Rowling [Times of Rent]: 31
  - [Book Name]: The Adventures of Tom Sawyer [Book ISBN]: 1-10 [Author]: Mark Twain [Times of Rent]: 23
  - [Book Name]: The Adventures of Sherlock Holmes [Book ISBN]: 1-09 [Author]: Arthur Conan Doyle [Times of Rent]: 13
Most Want Book By Category:
  - [Book Name]: The Adventures of Tom Sawyer [Book ISBN]: 1-10 [Category]: Story [Times of Want]: 13
  - [Book Name]: The Hobbit [Book ISBN]: 0-03 [Category]: Fantasy [Times of Want]: 13
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Category]: Novel [Times of Want]: 15
Most Want Book By Author:
  - [Book Name]: The Hobbit [Book ISBN]: 0-03 [Author]: J.R.R. Tolkien [Times of Want]: 13
  - [Book Name]: Butterball [Book ISBN]: 1-08 [Author]: Henry-Albert-Guy de Maupassant [Times of Want]: 8
  - [Book Name]: Pride and Prejudice [Book ISBN]: 1-07 [Author]: Jane Austen [Times of Want]: 15
  - [Book Name]: The Little Prince [Book ISBN]: 0-06 [Author]: Antoine de Saint [Times of Want]: 5
  - [Book Name]: Harry Potter and the Philosopher Stone [Book ISBN]: 0-02 [Author]: J.K. Rowling [Times of Want]: 2
  - [Book Name]: The Lion, the Witch and the Wardrobe [Book ISBN]: 0-05 [Author]: C.S. Lewis [Times of Want]: 10
  - [Book Name]: The Adventures of Tom Sawyer [Book ISBN]: 1-10 [Author]: Mark Twain [Times of Want]: 13
  - [Book Name]: The Adventures of Sherlock Holmes [Book ISBN]: 1-09 [Author]: Arthur Conan Doyle [Times of Want]: 5
------------------------------------------------------------
```

- In order to enable the administrator to query the analysis data of the LMS, we use a separate ***Analysis Report*** page to display the information.
- At the beginning of the analysis report, we print the name of the report and the date this report was generated, distinguished from the following body information by a single horizontal line.
- To avoid typographical problems caused by differences in length between data, which can lead to errors when reading the information, use indentation and a string of "-" to separate different sections as subheadings for the same type of report.
- In the same line of information, use **"[data name]"** to specify the type of information being printed, such as the **data name "[Book Name]"** and **the name of the book “Pride and Prejudice”**.
- There will be an option at the end of the analysis report function where users can enter to return to the main page when they are done viewing the analysis report.

4. SYSTEM TESTING

The purpose of the system testing process is to show that the program does what it is intended to do and to discover program defects before it is put into use by using artificial data to test. Our project went through two levels of testing.

4.1 Unit Test

We mainly perform the unit test for the DataBase.java class in the controller package.

- This unit test is aimed to test database connection, an encapsulated method for general query, contains, creates tables, updates, inserts, deletes, and searches.
- This unit test is used to find the potential exceptions in methods.
- Moreover, it can also verify the Singleton design of the database class.

4.2 Component Test

We mainly perform the unit test for the ModelController.java class in the controller package.

- This component test is aimed to test all utility functions of the LMS by invoking all models and other controllers.
- The component test includes tests for
    - initializing the data buffer,
    - refreshing the buffer from the database,
    - adding records,
    - deleting records,
    - searching on the records,
    - activated or deactivated the user,
    - reserve the book,
    - canceling the reserved book,
    - cancel the placed book,
    - get expired rent book,
    - return book from the user,
    - refresh expired placed book
    - rent book from user

- rent book from the placed book,
- activate user after return book and deactivate User For Expired Rent Book

4.3 System Test

For system testing, we created the testing data on users and books, as well as the default situation with several functions of rent, reserve and placed records in the database. After several rounds of testing on the LMS as a whole system, we observed the correct outputs from all functions designed for the LMS, as well as some hidden errors and unconsidered situations of the system, followed by corresponding fixing and updates.

5. LIMITATIONS

- We have a limitation on the bookID generation is manual operation instead of automatically generated by the LMS System which needs a librarian to notice the different bookID input while adding books to the library.
- We don’t have any login system with passwords for librarians to operate which means it may have low security for library management that anyone could use the LMS System without any certification check.
- On account of that we don't have an account login system for users to connect the notification part for sending the email to users’ email remaining, so it could only be used by librarians for checking records when users need to check their own status and operation records.
- Due to unsatisfactory performance for the online database of apollo, we actually have a long delay in initializing the connection with DBMS which may cause an unsatisfactory experience for LMS System users.
- We don’t provide specification about the error input, which only will display a repetitive input box for librarians to input again.

6. CONCLUSIONS & FUTURE SCOPE

For improvement, we plan to add the Security System with account login for librarians to have higher security of library management. We also plan to add the extra part for users to have self-service for renting, and reserving books on their own in library devices or even online.

7. REFERENCES

[1] Wikipedia Contributors, “Model–view–controller,” *Wikipedia*, Jan. 21, 2019.

https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

[2] “Lesson: JDBC Basics (The Java™ Tutorials > JDBC(TM) Database Access),” *docs.oracle.com*.

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

[3] Wikipedia Contributors, “Singleton pattern,” *Wikipedia*, Jul. 01, 2019.

https://en.wikipedia.org/wiki/Singleton_pattern

[4] “Assignment 1 Question 4 Answer”, *COMP 1011 Programming Fundamental*, Jan, 2022.

End of Library Management System Report

COMP2411 Group 4