Final LMS Project

CEN 3024C Software Development

Fiodor Culacovschi

12.04.2023

TABLE OF CONTENTS

Introduction

In the world of managing information and technology, creating software applications is super important for any project to succeed. People usually follow a set plan called the Software Development Life Cycle (SDLC) to develop software from the beginning to when it's ready to use and maintain. The SDLC has different steps like planning, analysis, design, implementation, testing, deployment, and maintenance, and each step is crucial for making sure the software works well.

One specific project that's important is the Library Management System (LMS). This project aims to make things easier for libraries, which are places full of knowledge. The goal is to create software that's easy to use and helps manage all the resources in a library efficiently.

My Library Management System Project goes beyond the usual manual way of running libraries. It is intended to make things easier by automating and improving different tasks. These tasks include adding, removing, checking in, checking out, updating availability, and setting due dates, all connected to an SQL database.

The existing LMS is good at basic book management. It combines a user-friendly screen with a strong SQL database in the background. The SQL database table acts as the main place for book information, making it easy for users to do things through the screen.

Module 3

Requirements document for Library Management System

1. Introduction

1.1 Purpose

The purpose of this document is to represent the requirements for LMS development plan.

As a potential developer I must create the LMS, it's like a blueprint for each project with detailed notes, sketches, diagrams, equations for algorithms, etc.

1.2 Project scope

LMS outline:

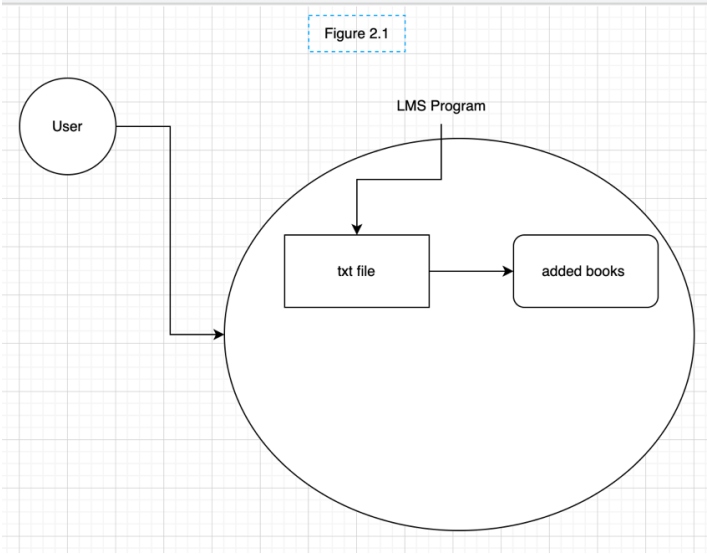
- The program should be able to read and store book information from comma-delimited txt file.
- User(librarian) can remove a book from the catalog by barcode or title.
- Check out the book from LMS, status will change to "checked out" and the date must be set for a 4-week period from the current date. Display an error message if the book has been already checked out.
- Checking in book from LMS, program should be able to change the status to "checked in" and remove the due date or represent as "null".
- Program will display the contents of DB on the screen.

2. Functional Requirements

2.1 Adding books from txt file to LMS (use phase 1 & 2):

`User story:` The program will allow user to add books to the LMS from an earlier created txt file with comma-delimited book info.

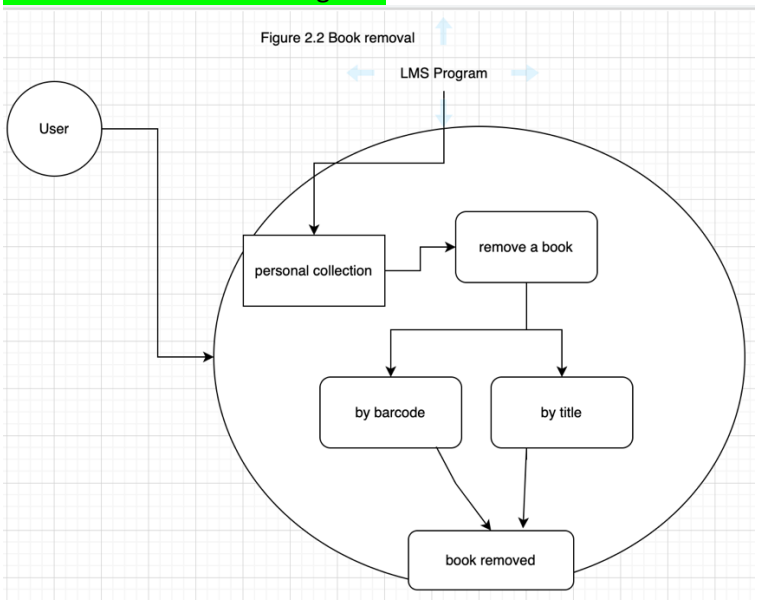
Add books from text file use case diagram:



2.2 Removing a book:

`User story:` As a librarian, I want to have the ability to remove any book(s) from LMS using barcode or title.

Book removal use case diagram:

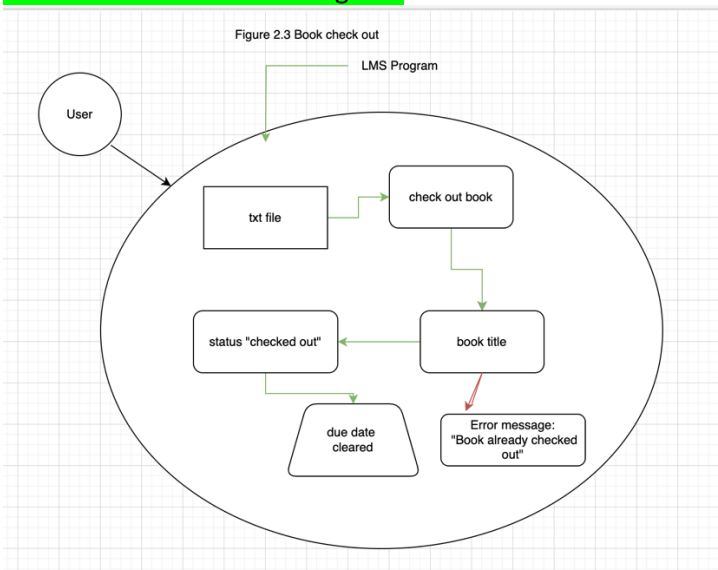


2.3 Book check out process:

`User story:` To check out the book user will have to enter the book title. The program should change the status to “checked out” and clear the due date. and the date must

be set for a 4-week period from the current date. Display an error message if the book has been already checked out.

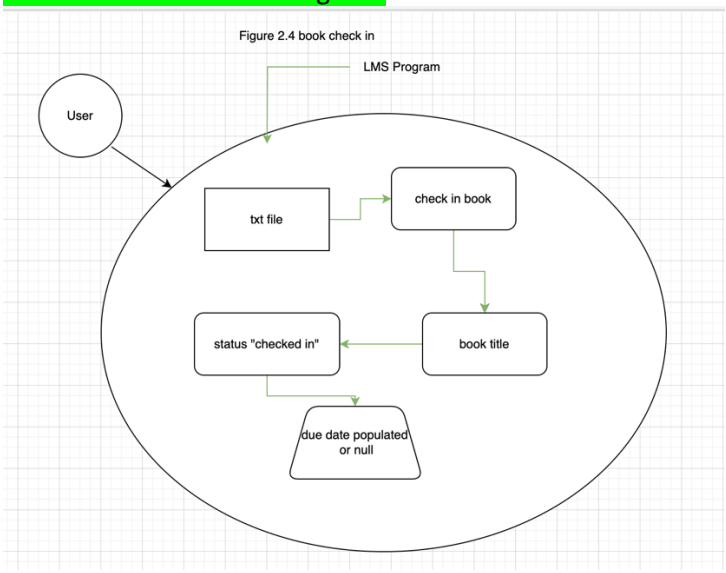
Book check out use case diagram:



2.4 Book check in process:

`User story:` To check in the book user will have to type or select the book title. The program should change the status to “checked in” and set the due date.

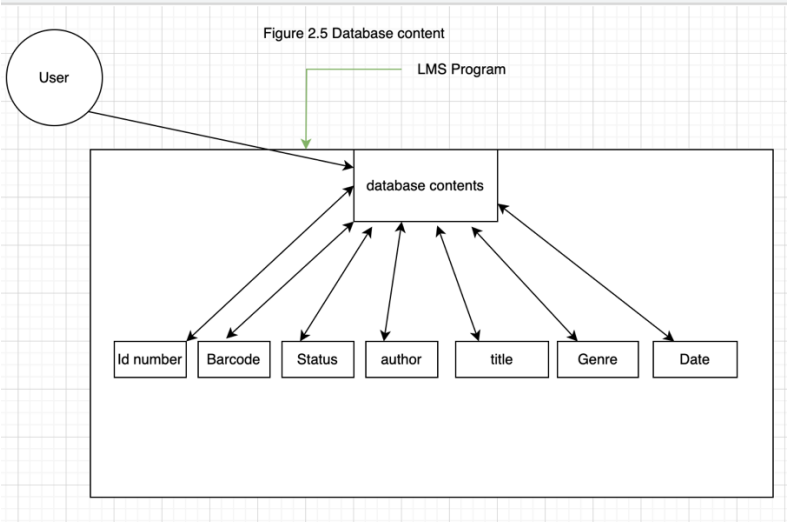
Book check in use case diagram:



2.5 To display database contents:

`User story:` The program will display the database contents so user can see what books are available at this moment.

Database content use case diagram:



3. Non-Functional Requirements

3.1 Data Requirements:

The program will contain the following:

- Information about each book including barcode number, author, title, genre, status, and due date.
- Information about user for example if user decide to create multiple accounts or other potential expansions.
- Log data for auditing and troubleshooting.

3.2 User Interface Design:

GUI for LMS:

- Fields and text boxes for entering book info.
- Buttons for adding or removing, checking out or in books.
- A list or a table to display database content.

4. Constraints

- The program should be able to handle enough data without errors.
- Program compatibility with different OS or Java versions.
- User-friendly GUI.
- In the final version of this project (phase 3) program should be able to support SQL queries for data retrieval or other manipulations.

5. Glossary/ Appendices

- Provide definitions of technical terms used in the project description.
- Any additional information, resources relevant to the project.

Module 4

Implementation Plan for Library Management System (LMS) using MVC Architectural Pattern

1. Introduction

1.1 Purpose

The purpose of this document is to demonstrate the development plan for the Library Management System (LMS) project. It is a blueprint for the development team, providing detailed notes, sketches, diagrams, and algorithmic equations for the LMS implementation.

1.2 Project Scope

LMS Outline

The Library Management System (LMS) is designed to enable efficient management of library resources. The project's scope includes the following key functionalities:

1.3 Importing Book Information:

- The system should be capable of reading and storing book information from a comma-delimited text file.

Book Removal:

Authorized users (librarians) should be able to remove a book from the catalog using either the book's barcode or title.

Book Checkout:

Users should be able to check out books from the LMS.
The system should change the book's status to "checked out" and set a due date for four weeks from the current date.
An error message should be displayed if a user attempts to check out a book that is already checked out.

Book Check-in:

Users should be able to check in books to the LMS.
The system should change the book's status to "checked in" and remove the due date or represent it as "null."

Display of Library Contents:

The LMS should provide a user-friendly interface to display the contents of the library catalog on the screen.

2. Architectural Pattern

I have chosen the Model-View-Controller (MVC) architectural pattern for the development of my Library Management System (LMS).

MVC will separate the application into three interconnected components:

Model:
Represents the data and business and LMS logic.

View:
Manages the presentation and the UI.

Controller:
This handles user input and coordinates the Model and View.

I decide to use the MVC architectural pattern because it aligns with my project goals of modularity, separation of concerns, and maintainability. It allows me to maintain and develop all components independently, enabling future enhancements and updates.

3. Development Plan
3.1 Model Component
Tasks:

- Design the data model to represent books, users, and transactions.
- Implement algorithms for book checkout and check-in.
- Create methods for reading and storing book information from a text file.
- Develop a database schema for persistent data storage.

3.2 View Component
Tasks:

- Design the user UI for interacting with the LMS.
- Create windows for browsing the catalog, checking out, and checking in books.
- Implement error message displays for different issues.

3.3 Controller Component
Tasks:

- Create controllers to handle user input and interactions.
- Develop logic for checkout, and check-in, and book removal processes.
- Prompt for proper communication between the Model and View components.

3.4 Integration and Testing
Tasks:

- Integrate the Model, View, and Controller components into comprehensive LMS.
- Implement testing for individual components.
- To ensure seamless functionality, effectuate integration testing.
- Report and solve any issues or bugs.

3.5 User Documentation

Tasks:

- Create user documentation and user guides for librarians and end-users.
- Instructions should include importing book information, managing the catalog, and using the LMS features effectively.

4. For this project I opted to follow the **MVC architectural pattern design.**
The following is an overview for each component in the MVC pattern and how it will be used to generate the Library Management System (LMS):

4.1 Model:

Purpose: The Model represents the main function of the application. It encapsulates the data and the business logic. It manages and processes data, as well as enforces the application's rules and behaviors.

LMS Implementation: In this LMS, the Model component will handle tasks such as:

- Define the data structures to represent books, users, and transactions.
- Implementing algorithms for removal, checkout, check-in of books.
- Managing the consistency of data, for using a database to store information about books and users.

4.2 View:

Purpose: The View is responsible for presenting data to the user in a way that is comprehendible and visually appealing. It handles the user interface (UI) components and displays information to the user.

LMS Implementation:

In the LMS, the View component will do the following:

- Design the user interface windows where librarians and users can interact with the system.
- Display the catalog of books, user profiles, and transaction history.
- Display error messages and notifications to users when necessary.

4.3 Controller:

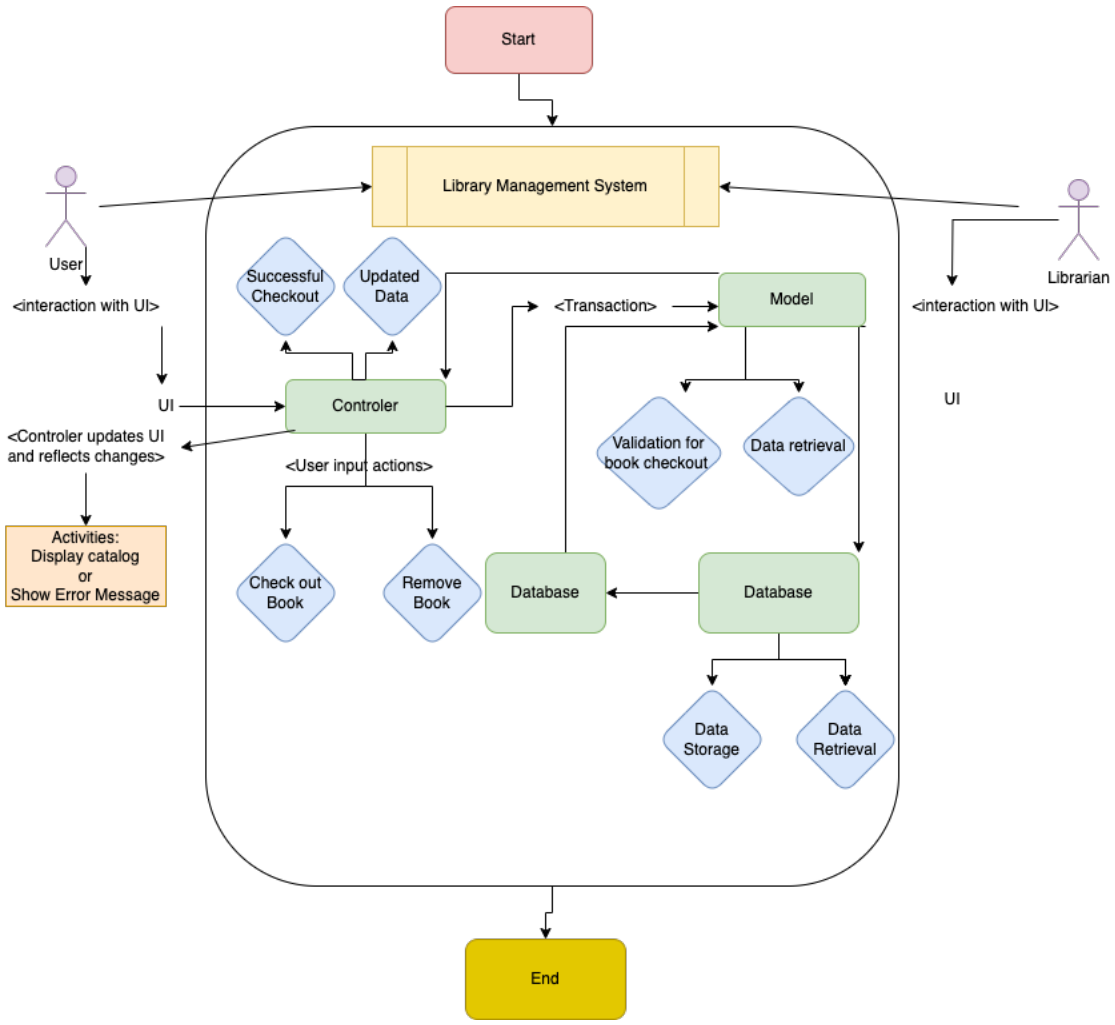
Purpose: The Controller is an intermediary between the Model and the View.

- It will receive user input and translate it into actions that the Model should perform.
- In addition, it will handle the flow of the application and controls the interactions between Model and View.

LMS Implementation: For the LMS, the Controller component will do the following:

- Receive user input, for example to check out or remove a book.
- Call the correct methods in the Model to perform these actions.
- It updates the View to reflect the changes in the data or provide feedback to the user.

5. Activity Diagram



6. Coding Strategy: In my first phase of writing the code for this project I used Java to build the Library Management System (LMS) functionality. Using Model-View-Controller (MVC) architecture I have effectively separating concerns. I broke down into small steps of the coding strategy and identified the Java classes and object-oriented principles used for each layer:

6.1 Model (Data Layer):

Purpose: The Model layer represents the core data and business logic of the application.
Java Classes/Object-Oriented Principles were used:

- *Book Class:* Represents a book and encapsulates its properties (id, title, author).
- *userCollection:* A list (collection) of Book objects representing the user's book collection.
- *Data Encapsulation:* The Book class uses encapsulation by providing private fields (id, title, author) and public getter methods.
- *Data Abstraction:* The Book class abstracts the concept of a book with properties and behaviors being well-defined.

6.2 View (User Interface Layer):
Purpose: The View layer is responsible for displaying data to the user and receiving user input.
Java Classes/Object-Oriented Principles were used:

- *LibraryManagementApp Class:* contains methods for displaying information to the user and receiving user input.
- *UI Interaction:* The Scanner class is used to read user input.
- *Menu Display:* A menu is shown to the user to facilitate interaction.
- *Error Handling:* The code includes error handling for file reading and writing, giving an engaging user experience.

6.3 Controller (Logic Layer):
Purpose: The Controller layer takes user input, coordinates the Model and View, and contains application logic.
Java Classes/Object-Oriented Principles Used:
LibraryManagementApp Class: Performs as the controller by implementing actions based on user input.
Method Calls: Methods *in LibraryManagementApp* conducts interactions between the Model and View components.
Iterators: An iterator is implemented to search for and remove books from the user's collection.
Data Manipulation: Methods like *addToUserCollection* and *removeBookFromCollection* manipulate the data and update the Model and View as required.

7. Project Additional Considerations
At this phase of my project my code is a simple program that runs in IDE, it reads the information from a “txt” file, user can do most of the things my project requires. However, as the project is moving to another stage, I must do the following:

- My next step is going to be to do the transition from a console-based app to GUI implementing SQL database.
- I will refactor my code and will separate GUI from MVC.
- GUI for user interactions.
- MVC will interact with database and other data operations.

7.1 GUI Framework
For my project I will use JavaFX for building the UI’s.

7.2 Database Set up
Using JDBC I will create a database connection.

7.3 GUI Design
I will create multiple windows for different tasks, for example displaying book catalog, then user can add books to his own collection, checking out books, and removing them.

- To display the catalog with books available the program will pull the data from the database.
- To add books to the collection: I will create a window where user will be prompted to input book information by barcode or book ID. The application will be able to insert appropriate information into database.
- To check out books: I will create a window where user selects the book from the catalog and initiate check out. In this case the database will be updated and will show the information that the book has been checked out.
- To remove a book from user collection: I will create a window where user can see all added books and remove books no longer needed and the database should be updated.

7.4 Database interactions

- Use JDBS create a connection to the database.
- Implement SQL queries, for example: create, read, update, delete.
- Make sure that all connections were made properly, and all information id displayed appropriately.

7.5 Event handling

- Make sure components like text fields and buttons are responsive.
- Test application thoroughly for compatibility, functionality, interactivity and responsiveness.

Module 5

Testing plan for Library Management System Project

1. Introduction:

The Library Management System project scope is to create an application that will run smoothly on any platform and device. The goal is to develop a software solution to efficiently manage library recourses, that will include the books, the librarians, and other administrative tasks. My testing plan is to test all classes and methods and the interaction between them, test user input and output, and to make sure that the systems functionality, security, performance, and reliability meet projects' quality objectives.

2. Test Objectives:

- ***Functionality validation of LMS:*** make sure meet all necessary requirements
- ***System performance:*** test the system and make sure it handles expected loads
- ***Security:*** make sure all user data and libraries are protected
- ***Report any defects or inconsistences*****:** find quick methods to eliminate them
- ***Make sure*****:** that the system works well and won't crash during its use

3. Scope:

3.1 Functional testing:

- Creating an account
- Log in/out process
- Catalog management: add a book, remove or extend due date
- Library administration: new user registration, updating their profiles, borrow and returning process management
- Search and retrieval of library recourses

3.2 Performance Testing:

- ***Load testing:*** test how many users can access the system at the same time without any inconveniences
- ***Stress testing:*** test the program to its limits, see if it can handle multiple operations running simultaneously for example if the library has big sales event going on
- ***Response time testing:*** test how quick the program responds for example a book search or check out process

3.3 Security Testing:

- Authentication and authorization process, make sure that the right users can access certain parts of the library system
- Protect all user data, implement encryption method

4. Testing Strategy

4.1 Functional testing:

- ***Unit testing:*** make sure that the search engine within LMS works properly
- ***Integration testing:*** make sure all different parts of the library work correctly
- ***System testing:*** make sure the entire library system works as a whole, all processes running properly

4.2 Performance testing:

- ***Load testing:*** test how many users can access the system at the same time without any inconveniences
- ***Stress testing:*** test the program to its limits, see if it can handle multiple operations running simultaneously for example if the library has big sales event going on
- ***Scalability testing:*** think of how much more the program can handle, this test will help you understand if It can add more functions/operations to the library without causing any problems

4.3 Security Testing:

- Authentication and authorization process, make sure that the right users can access certain parts of the library system
- Protect all user data, implement encryption method

5. Test Deliverables

5.1 Test plan:

- It is like a guidebook that tells step by step how everything should be done, what parts of the program have to be tested and how to test them

5.2 Test cases:

- It is like instruction which tells what exactly has to be done, check if everything works as intended

5.3 Test scripts:

- It is like an automated tool that does all test cases automatically which speeds up the whole process and the results are more accurate

5.4 Test data sets:

- These data sets are made-up information that user is going to use to check system reaction

5.5 Defect report:

- Take a note of all detected defects that program causes during testing for future improvements

6. Test execution

- In this case I am going to be holding this role as a developer and tester however, in real world there will be a testing team that will perform these various tests, they will take notes of any defects that were spotted, send it back to developers to fix all bugs, then testers will share the report with other stakeholders.
- All test cases will be following an earlier guideline this will include functional tests, performance test, security test, user acceptance test and others.

Building and deploying the software

The project was divided into different phases before it was launched. First, I designed a plan and came up with ideas for how I wanted my project to look. I then created a simple program that could read information about books from a file. The next step was to write tests for the project and check each part of the application. Testing is important, as it helps ensure that the software does what it's supposed to do, preventing bugs, reducing development costs, and improving performance.

Then, the project was split into additional phases.
One of them involved creating a user interface (UI). Once that part was done and tested to make sure it worked as intended,
the next step was to connect everything to an SQL database.
The final stage of the project was to ensure that the application would run smoothly, fetching data from the database table, and displaying it in the UI. After completing these steps, I had to create a JAR file and include the entire project along with Javadoc.

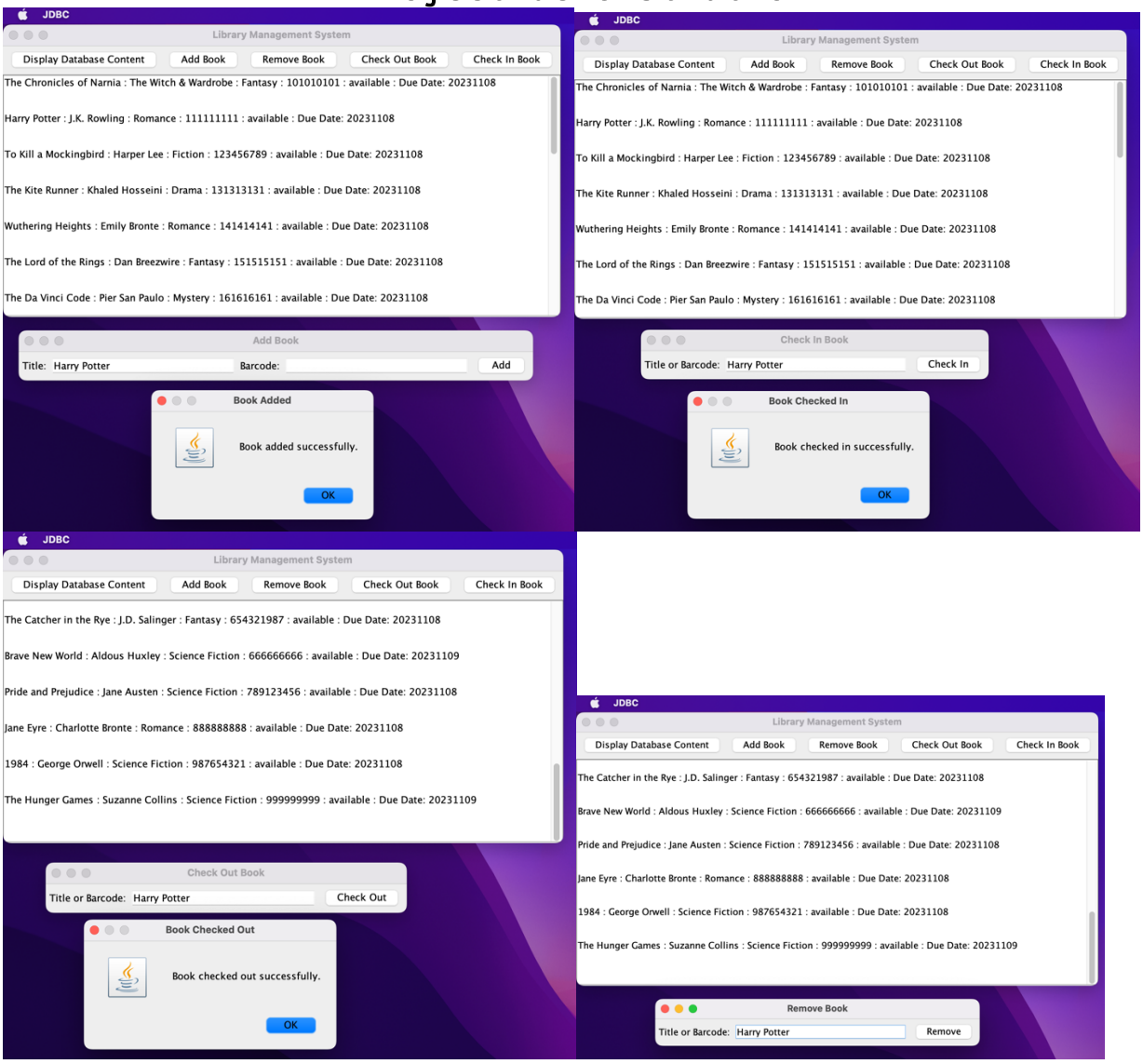
Appendix A

```java
/**
 * @author Fiodor Culacovschi
 * @version 12.04.2023
 */
/**
 * The JDBC class contains the main method and GUI components for the Library Management System.
 * It allows users to display, add, remove, check out, and check in books.
 * It provides methods for interacting with a MySQL database.
 */
public class JDBC {

    // Variables for GUI components and book due date information
    private static JTextArea textArea;
    private static String dueDate;
    private static String formattedDueDate;


    /**
     * Main method to launch the Library Management System GUI.
     *
     * @param args Command line arguments (not used in this application).
     */
    public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
            @Override
            public void run() {
                createAndShowGUI();
            }
        });
    }
```

Appendix B

```
/**
 * The BookManager class is responsible for managing books in the Library Management System.
 * It includes methods for adding, removing, checking out, and checking in books.
 */
class BookManager {
    // ... (Include methods for managing books as per your project requirements)
}

/**
 * The DatabaseManager class is responsible for handling interactions with the MySQL database.
 * It provides methods for establishing connections, executing queries, and managing database operations.
 */
class DatabaseManager {
    // ... (Include methods for database operations as per your project requirements)
}

/**
 * The BookDialog class provides a generic template for creating dialogs related to book operations.
 * It includes common components like labels, text fields, and buttons that can be reused for various operations.
 */
class BookDialog {
    // ... (Include components and methods for creating book-related dialogs)
}

/**
 * The DateUtil class provides utility methods for handling date-related operations in the Library Management System.
 * It includes methods for calculating due dates, formatting dates, etc.
 */
class DateUtil {
    // ... (Include methods for date-related operations as per your project requirements)
}

/**
 * The Book class represents a book in the Library Management System.
 * It includes properties like title, author, barcode, etc., and methods for accessing and modifying these properties.
 */
class Book {
    // ... (Include properties and methods for representing a book)
}

/**
 * The BookDatabase class is responsible for managing the collection of books stored in the MySQL database.
 * It includes methods for retrieving, updating, and manipulating book data in the database.
 */
class BookDatabase {
    // ... (Include methods for managing book data in the database)
}
```

Project demonstration



[Final LMS Fiodor Culacovschi YouTube link]

[Final LMS GitHub link]