**Library Management System (Shawn Ellinwood) - Initial Draft**

**Project Overview**

The Library Management System is a Java program designed to manage a collection of books and library members. This initial draft demonstrates the core structure and partial functionality, including essential operations like adding members, borrowing and returning books, and listing available books.

**Current State**

The project currently includes the following primary classes and components:

1. **Book**: Manages details for each book, such as title, author, borrowing status, and due date.
2. **Member**: Handles each library member’s information and their list of borrowed books.
3. **Library**: Manages the collection of books and members, and provides methods for adding, finding, and listing books and members.
4. **LibraryManagementSystem**: A command-line interface for user interactions with the library system.

**Rationale for Choosing Serialization**

Having worked with I/O projects before, I initially considered using a text file for data storage. While I was able to create and export text files, I found that the data wasn’t well-organized or easy to read outside of an IDE like IntelliJ. I also thought about using Excel or a database but found the process confusing and beyond my current skill set, as I have no experience exporting data to Excel or working with SQL databases. Serialization ultimately proved to be the best choice, as it allowed me to save and load Java objects (like Library, Book, and Member) directly, preserving their structure without requiring complex formatting. This approach simplified data handling and fit the project’s current needs.

The program supports basic functionality such as adding books and members, borrowing and returning books, and listing available books and members. This draft provides a foundation for future functionality and enhancements.

**Features Implemented**

* A basic menu-driven interface for interacting with the library system.
* Core operations including:
  + Adding books and members.
  + Borrowing and returning books.
  + Listing available books and library members.
* Partial implementation of due dates for borrowed books and basic overdue checks.

**Known Issues and Incomplete Parts**

The current draft has several known issues and areas for improvement, which will hopefully be addressed in future versions.

1. **Partial Due Date and Overdue Check Implementation**
   * **Issue**: Basic overdue checks are implemented, but the program may not fully handle complex scenarios, such as different borrowing periods or date changes across sessions.
   * **Plan**: Implement dynamic overdue checks that assess due dates on startup and during relevant actions.
2. **Error Handling and User Input Validation**
   * **Issue**: Limited input validation for empty fields, duplicate entries, and invalid actions (e.g., attempting to borrow a non-existent book).
   * **Plan**: Improve input validation to handle empty entries, duplicates, and invalid actions, with user-friendly messages.
3. **Incomplete Data Persistence with Error Handling**
   * **Issue**: Data saving and loading are functional, but lack full error handling (e.g., corrupted or missing files).
   * **Plan**: Add error handling to file operations and provide user feedback if loading data fails, with the option to start a new library.
4. **Limited User Interface and Menu Navigation**
   * **Issue**: Menu navigation could be improved for better user experience, such as adding options to return to the main menu after each action.
   * **Plan**: Add prompts to guide users back to the main menu or continue with other actions.
5. **Handling Edge Cases**
   * **Issue**: The program currently has limited handling of edge cases, such as reaching the maximum borrowing limit or returning books that weren’t borrowed.
   * **Plan**: Add conditions to handle these scenarios, providing clear feedback for users.
6. **No Notification for Lack of Available Books**
   * **Issue**: If all books are checked out, users aren’t notified that no books are available.
   * **Plan**: Update the listAvailableBooks method to display a message when no books are available.
7. **Invalid Input for Menu Options**
   * **Issue**: Entering a non-integer value (e.g., 5.6) as a menu option throws an InputMismatchException.
   * **Plan**: Add input validation to accept only whole numbers for menu options, prompting users with an error message if input is invalid.

**Future Enhancements**

In future updates, the following enhancements will be implemented:

* Detailed overdue notifications.
* Improved persistent data handling.
* Enhanced error handling for a more robust and user-friendly experience.