**Library Management System Requirement Specification**

**Objective**

Design and implement a Library Management System (LMS) to manage books, library members, and borrowing/lending activities. The system must incorporate core Object Oriented Programming (OOP) principles such as encapsulation, inheritance, polymorphism, abstraction, and composition. Additionally, the system will include implementations of sorting and searching algorithms to enhance functionality and algorithmic understanding.

**System Requirements**

**1. Classes to Implement**

1. **Book**

o **Attributes**:

▪ bookId (unique identifier)

▪ title

▪ author

▪ genre

▪ isAvailable (boolean to check if the book is available for borrowing) o **Methods**:

▪ markAsBorrowed() - Marks the book as not available.

▪ markAsReturned() - Marks the book as available.

▪ displayDetails() - Displays book details.

2. **Member**

o **Attributes**:

▪ memberId (unique identifier)

▪ name

▪ email

1

▪ borrowedBooks (list of books currently borrowed by the member) o **Methods**:

▪ borrowBook(book) - Adds a book to the borrowedBooks list if it is available.

▪ returnBook(book) - Removes a book from the borrowedBooks list and updates its availability.

▪ displayDetails() - Displays member details along with the list of borrowed books.

3. **Librarian (inherits from Member)**

o **Additional Attributes**:

▪ employeeId

o **Additional Methods**:

▪ addBook(book) - Adds a new book to the library's catalog.

▪ removeBook(book) - Removes a book from the catalog.

▪ manageMember(member) - Manages members (e.g., add, remove, or update member information).

4. **Library**

o **Attributes**:

▪ libraryName

▪ catalog (list of all books in the library)

▪ members (list of all members)

o **Methods**:

▪ registerMember(member) - Adds a new member to the members list. ▪ unregisterMember(member) - Removes a member from the list.

▪ searchBookByTitle(title) - Searches for a book in the catalog by its title.

▪ searchBookByAuthor(author) - Searches for books by an author. ▪ displayCatalog() - Displays the details of all books in the catalog.

2

▪ displayMembers() - Displays the details of all members.

**2. Functional Requirements**

1. **Book Management**:

o Librarians can add, remove, and update book details.

o Books can be searched by title, author, or genre.

o Each book must have a unique identifier.

2. **Member Management**:

o Members can borrow and return books.

o Borrowing is only allowed if the book is available.

o A member can borrow up to 5 books at a time.

o The system should display a warning if a member tries to borrow more than the limit.

3. **Borrowing and Returning**:

o When a book is borrowed, its availability is updated.

o When a book is returned, its availability is restored.

4. **Reporting**:

o Display all books borrowed by a specific member.

o Display all currently borrowed books in the library.

**Additional Requirements: Sorting and Searching Algorithms**

**5. Sorting Features**

The LMS must support sorting functionalities for books and members using manually implemented algorithms.

1. **Book Sorting**

o Implement sorting by:

▪ **Title** (alphabetical order).

3

▪ **Author** (alphabetical order).

▪ **Genre** (group books by genre in alphabetical order). o Use **Bubble Sort** or **Selection Sort** for these functionalities.

**Method**:

sortBooksBy(criteria) in the Library class to sort the catalog. 2. **Member Sorting**

o Implement sorting by:

▪ **Name** (alphabetical order).

▪ **Number of Borrowed Books** (descending order).

o Use **Insertion Sort** or **Quick Sort** for these functionalities.

**Method**:

- sortMembersBy(criteria) in the Library class to sort members.

**6. Searching Features**

The LMS must support efficient searching for books and members. 1. **Book Searching**

o **Search Criteria**:

▪ By Title: Partial or exact match.

▪ By Author: List all books by a specific author.

▪ By Genre: List all books in a specific genre.

o **Algorithms**:

▪ Use **Linear Search** for partial matches.

▪ Use **Binary Search** for exact matches (after sorting).

**Method**:

- searchBookByTitle(title)

- searchBookByAuthor(author),

- searchBookByGenre(genre) in the Library class.

2. **Member Searching**

4

o **Search Criteria**:

▪ By Name: Partial or exact match.

▪ By Member ID: Exact match.

o **Algorithms**:

▪ Use **Linear Search** for partial matches.

▪ Use **Binary Search** for exact matches (after sorting).

**Method**:

- searchMemberByName(name)

- searchMemberById(memberId) in the Library class.

**7. Performance Considerations**

• Analyze and document the **time complexity** of the implemented sorting and searching algorithms.

• Compare the implemented sorting algorithms and explain their suitability for this project.

**Deliverables**

1. **Code**:

o Complete implementation of the system using OOP principles.

o Sorting and searching algorithms must be implemented manually. o Use proper comments to explain key sections of the code.

2. **Documentation**:

o Explanation of the design decisions and how OOP principles were applied. o A flow diagram showing the relationships between the classes.

o Detailed analysis of the time complexity for sorting and searching algorithms.

3. **Test Cases**:

5

o Test the system with at least 5 members, 10 books, and multiple borrowing and returning transactions.

o Verify sorting and searching functionalities for books and members.

6