University of Science and Technology Chittagong

Project Proposal and Development

Submitted by,

Name- **Ikhoanus Safa Houqe**

Id:- **22070113**

Batch- **43(b)**

Course id:- **CSE 124**

Submitted to,

Debabarata Mallick

Lecturer,

University of Science and Technology Chattagram

Project title: Library Management System in Java.

Purpose of the project: The *Library Management System* is designed to automate the management of books in a library, including functionalities like adding new books, issuing books to users, returning books, and maintaining records of issued/returned books. The system helps reduce manual work and errors, and it provides an efficient way to organize and access library information.

Problem the Project Aims to Solve: Managing a library manually can be time-consuming and prone to errors. This project aims to address the following key issues:

1. Difficulty in Tracking Books
    - Manually keeping records of borrowed and available books can lead to misplacement or loss of books.
    - There is no efficient way to know who has borrowed a book and when it should be returned.
2. Time-Consuming Manual Processes
    - Librarians spend a lot of time logging book transactions, which slows down operations.
    - Searching for a book in a physical log or register is inefficient.
3. Lack of Availability Information
    - Users do not have an immediate way to check if a book is available or borrowed.
    - Miscommunication can occur, leading to frustration when a book is not found.
4. Errors in Book Management
    - Manually updating records increases the chance of errors, such as incorrect borrower details.
    - Books may be misplaced or recorded incorrectly, making it difficult to locate them.

How the Project Solves These Issues

✅ Automates book tracking – The system records books when borrowed and returned, reducing manual errors.
✅ Quick book search – Users can search for a book by title instead of manually checking records.
✅ Displays real-time availability – Users can instantly see which books are available or borrowed.
✅ Efficient inventory management – Helps librarians maintain an organized book collection with ease.

Importance and relevance of using OOP concepts: Object-oriented programming (OOP) plays a crucial role in system design due to its ability to organize complex systems into manageable units. This approach promotes modularity, reusability, and encapsulation, making systems easier to understand, maintain, and modify over time.

The main goals of this Library Management System project:

Book Management – Add, display, and maintain a collection of books in the library.

Borrowing and Returning Books – Allow users to borrow and return books while tracking the borrowing status.

User Interaction – Display messages for successful and unsuccessful operations (e.g., borrowing an already borrowed book).

Book Search – Enable searching for books by title.

Inventory Tracking – Provide the ability to count the total number of books in the library.

User-Friendly Console Interface – Ensure ease of use with clear output messages and interactions.

The key functionalities implemented in the Library Management System:

1. Book Management
    - Add books to the library (addBook method).
    - Display all available books (displayBooks method).
2. Borrowing & Returning Books
    - Borrow a book (borrowBook method) while ensuring it's not already borrowed.
    - Return a borrowed book (returnBook method) and update its status.
3. Book Search
    - Search for a book by title (searchBook method) and display its details.
4. Inventory Tracking
    - Count the total number of books in the library (countBooks method).

Programming language(Java) :

```
package project1;


import java.util.ArrayList;
import java.util.List;


class Book {
   private String title;
   private String author;
   private boolean isBorrowed;
   private String borrowedBy;
public Book(String title, String author) {
      this.title = title;
      this.author = author;
```

```
        this.isBorrowed = false;
        this.borrowedBy = "";               }

    public String getTitle() {
        return title; }

    public String getAuthor() {
        return author; }

    public boolean isBorrowed() {
        return isBorrowed; }

    public String getBorrowedBy() {
        return borrowedBy; }

    public void borrowBook(String borrower) {
        if (!isBorrowed) {
            isBorrowed = true;
            borrowedBy = borrower;
            System.out.println(borrower + " borrowed " + title);}
        else {
            System.out.println("Book is already borrowed by " + borrowedBy); }
                                    }
public void returnBook() {
        if (isBorrowed) {
            System.out.println(borrowedBy + " returned " + title);
```

```
            isBorrowed = false;
            borrowedBy = "";}
        else {
            System.out.println("Book is not borrowed."); }
                        }
}
class Library {
    private List<Book> books;

    public Library() {
        books = new ArrayList<>(); }

    public void addBook(Book book) {
        books.add(book);
         System.out.println("Added: " + book.getTitle()); }

    public void borrowBook(String title, String borrower) {
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
                book.borrowBook(borrower);
                return; }               }
        System.out.println("Book not found!");
                                                                    }
public void returnBook(String title) {
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
```

```
            book.returnBook();
            return; }
        }
        System.out.println("Book not found!");
    }
public void displayBooks() {
        System.out.println("Library Books:");
        for (Book book : books) {
            System.out.println("Title: " + book.getTitle() + ", Author: " +
book.getAuthor() +
                    (book.isBorrowed() ? " (Borrowed by: " + book.getBorrowedBy() + ")"
: " (Available)"));
        }
    }
    public void searchBook(String title) {
        for (Book book : books) {
            if (book.getTitle().equalsIgnoreCase(title)) {
                System.out.println("Book Found: " + book.getTitle() + " by " +
book.getAuthor());
                return;
            }
        }
        System.out.println("Book not found in the library.");
    }
   public void countBooks() {
        System.out.println("Total books in the library: " + books.size());
    }
```

```
}
public class LibraryManagementSystem {
    public static void main(String[] args) {
        Library library = new Library();

        library.addBook(new Book("Harry Potter", "J.K. Rowling"));
        library.addBook(new Book("The Hobbit", "J.R.R. Tolkien"));
        library.addBook(new Book("1984", "George Orwell"));
        library.borrowBook("Harry Potter", "Alice");
        library.displayBooks();
        library.returnBook("Harry Potter");
        library.displayBooks();
        library.searchBook("1984");
        library.countBooks();
    }
}
```

Output:

```
Added: Harry Potter
Added: The Hobbit
Added: 1984
Alice borrowed Harry Potter
Library Books:
Title: Harry Potter, Author: J.K. Rowling (Borrowed by: Alice)
Title: The Hobbit, Author: J.R.R. Tolkien (Available)
Title: 1984, Author: George Orwell (Available)
Alice returned Harry Potter
```

```
Library Books:
Title: Harry Potter, Author: J.K. Rowling (Available)
Title: The Hobbit, Author: J.R.R. Tolkien (Available)
Title: 1984, Author: George Orwell (Available)
Book Found: 1984 by George Orwell
Total books in the library: 3
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

OOP Principles in the Library Management System:

1. Encapsulation:
    - Book Class: The attributes title, author, isBorrowed, and borrowedBy are private. Only the public methods like borrowBook() and returnBook() can modify these attributes, ensuring controlled access to data.
    - Library Class: The list of books (books) is encapsulated, preventing direct manipulation from outside the Library class. Access to books is only possible through public methods like addBook(), borrowBook(), and displayBooks().
2. Abstraction:
    - Book Class: The user interacts with the methods borrowBook() and returnBook(), abstracting away the complexity of how borrowing and returning are managed internally. The user doesn't need to know about the isBorrowed or borrowedBy fields directly.
    - Library Class: Operations like adding books, borrowing, and returning are abstracted into simple methods (addBook(), borrowBook(), returnBook()), hiding the internal implementation from the user.
3. Polymorphism:
    - Method Overloading: The borrowBook() method is overloaded with different parameters, such as title and borrower, showing polymorphic behavior by allowing different ways to borrow a book.

- Example of Polymorphism (future): You could implement method overriding in a subclass (e.g., DigitalBook) that provides a custom displayDetails() method.

4. Composition:
    - The Library class is composed of Book objects. It contains a list of Book instances (books), demonstrating composition where one object (Library) is composed of multiple instances of another object (Book).

Project Phases with Estimated Completion Times:

Phase 1: Requirement Analysis & Planning (1 Week)

- Define project scope, objectives, and key functionalities.
- Identify necessary classes, methods, and attributes.
- Plan the overall system architecture.

Phase 2: Core Development (2 Weeks)

- Implement Book class with encapsulation (1-2 days).
- Implement Library class for book management (3-4 days).
- Implement borrowing and returning book functionality (3 days).
- Implement book search and inventory tracking (2 days).

Phase 3: Testing & Debugging (1 Week)

- Test all features (borrowing, returning, searching, and inventory tracking).
- Fix any identified bugs or logical errors.

Final Product Achievements:-

The final Library Management System will achieve the following:

1. Efficient Book Management – Users can add, search, and display books in the library.

2. Borrowing & Returning System – Books can be borrowed and returned while keeping track of availability.
3. User-Friendly Interaction – Clear messages will notify users of successful and failed operations.
4. Search Functionality – Users can search for books by title.
5. Inventory Tracking – The system can count and display the total number of books.
6. Scalability & Maintainability – The code structure follows OOP principles, making it easy to extend with more features.

How it will be useful to users:-

This Library Management System will be useful to users—such as librarians, students, or book lovers—in several practical ways:

💡 Key Usefulness:

1. Easy Book Tracking
   Users can see which books are available and which ones are currently borrowed, including who borrowed them.
2. Efficient Borrowing and Returning
   The system allows quick check-in/check-out of books with clear messages if a book is already borrowed.
3. Search Functionality
   Users can quickly find a specific book by title without manually scanning a list.
4. Inventory Overview
   The system shows the total number of books in the library, helping manage the collection size.
5. Simple and Extendable
   The structure is clean and can be easily extended with new features like due dates, book categories, or user accounts.

Summary:- The Library Management System is a simple yet powerful Java-based application designed to streamline the management of books in a library. It enables users to add, borrow, return, search, and view books, making library operations efficient and user-friendly. This system promotes organized, efficient, and modern library management—benefiting institutions, librarians, and readers alike.

Reference:

1/ https://chatgpt.com/canvas/shared/67ed1ec0c09481918b1e64b2694a184e

2/ *Programming with Java* (4th Edition), written by E.Balagurusamy, page 421

3/ www.geeksforgeeks.org

Screenshot and explanation how much and how many features are developed:-

Developed Features

📚 Book Class

- Attributes:
  - title
  - author
  - isBorrowed
  - borrowedBy
- Methods:
  - borrowBook(String borrower)
  - returnBook()
  - Getter methods for all attributes

🏛️ **Library Class**

- Attributes:
  - List<Book> books

- Methods:
    - addBook(Book book) – Adds a book to the library
    - borrowBook(String title, String borrower) – Borrows a book
    - returnBook(String title) – Returns a book
    - displayBooks() – Shows current books and their status
    - searchBook(String title) – Searches a book by title (New Feature)
    - countBooks() – Counts total number of books (New Feature)

🚀 **Main Class (LibraryManagementSystem)**

- Demonstrates:
    - Adding books
    - Borrowing and returning books
    - Displaying books
    - Searching for a book
    - Counting total books

Features in use (in main method):

```
108         library.addBook(new Book("Harry Potter", "J.K. Rowling"));
109         library.borrowBook("Harry Potter", "Alice");
110         library.displayBooks();
111         library.returnBook("Harry Potter");
112         library.displayBooks();
113         library.searchBook("1984");
114         library.countBooks();
```

Remaining Features to Implement

1. **Update Book Information**
    - Add functionality to edit the title or author of a book after it has been added.

2. **Delete Book**
    - Implement the ability to remove a book from the library.
3. **Borrow/Return Date Tracking**
    - Record the date a book was borrowed and returned.
4. **Due Date and Fine System**
    - Implement a system for due dates and calculate fines for overdue books.
5. **Borrowing History**
    - Maintain a log of who borrowed each book and when.

New Learnings So Far

- Learned to design and implement object-oriented programs using Java.
- Gained hands-on experience with classes, constructors, and methods.
- Practiced using ArrayList and basic Java collections.
- Applied conditional statements and loops for logical flow.

Problems and Challenges Faced

- No check for duplicate book titles when adding books.
- No persistent storage — all data is lost after the program exits.
- No fine or due-date system for book returns.
- No history or log to track previous borrow and return activity.

Thank You