LIBRARY MANAGEMENT SYSTEM – PROJECT DOCUMENTATION

1. PROJECT OVERVIEW

The Library Management System is a complete Java console application for managing library operations. It allows librarians to add books and register members, supports book borrowing/returning with availability tracking, provides search functionality by title/author/genre, and displays all books or available books only. Built with proper OOP principles, the system uses encapsulation, ArrayList for dynamic storage, and input validation throughout.

Primary Goals:

Manage book inventory with ISBN, title, author, genre, and availability status

Track library members and their borrowed books

Enable borrow/return operations with availability validation

Provide case-insensitive search across multiple book attributes

Offer intuitive menu-driven interface for all operations

2. PROJECT OBJECTIVES

The main goals of this project are:

1. To understand and implement Object-Oriented Programming (OOP) concepts in Java
2. To design classes using encapsulation with private data members and public methods
3. To manage books and library members efficiently
4. To implement borrowing and returning of books with proper validation
5. To provide search functionality for books by title, author, or genre
6. To build a menu-driven console application
7. To improve problem-solving and Java programming skills
8. To simulate a real-world library system using Java

3. SETUP & INSTALLATION INSTRUCTIONS

1. Install Java Development Kit (JDK) on the system
2. Download JDK from the official Oracle or OpenJDK website
3. Verify Java installation by running the command java -version in the command prompt
4. Install a Java-supported IDE such as Eclipse, IntelliJ IDEA, or Visual Studio Code
5. Open the IDE and create a new Java project
6. Name the project as LibraryManagementSystem
7. Create a package (optional) named library or INTERNSHIP
8. Inside the package, create the following Java class files:
    - Book.java
    - Member.java
    - Library.java
    - LibrarySystem.java
9. Copy the respective source code into each file and save them
10. Compile the project to ensure there are no errors
11. Run the LibrarySystem.java file which contains the main() method
12. The menu-driven Library Management System will start in the console

4. CODE STRUCTURE EXPLANATION

BOOK CLASS

The Book class represents a single book in the library. It stores book details such as ISBN, title, author, genre, and availability status. Encapsulation is implemented using private variables and public getter and setter methods. The class also provides a method to display complete book information.

```
package Library_Management_System;

public class Book {
   private String isbn;
   private String title;
   private String author;
   private String genre;
   private boolean isAvailable;

   public Book(String isbn, String title, String author, String genre) {
      this.isbn = isbn;
      this.title = title;
      this.author = author;
      this.genre = genre;
      this.isAvailable = true;
   }
```

```
    public String getIsbn() { return isbn; }
    public void setIsbn(String isbn) { this.isbn = isbn; }

    public String getTitle() { return title; }
    public void setTitle(String title) { this.title = title; }

    public String getAuthor() { return author; }
    public void setAuthor(String author) { this.author = author; }

    public String getGenre() { return genre; }
    public void setGenre(String genre) { this.genre = genre; }

    public boolean isAvailable() { return isAvailable; }
    public void setAvailable(boolean available) { isAvailable = available; }

    public void displayInfo() {
        System.out.println("ISBN: " + isbn);
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Genre: " + genre);
        System.out.println("Status: " + (isAvailable ? "Available" : "Borrowed"));
        System.out.println();
    }
}
```

MEMBER CLASS

The Member class represents a library member ,It stores member details like member ID, name, contact information, and borrowed books. It allows members to borrow and return books with proper availability checks. An ArrayList is used to manage the list of borrowed books dynamically.

```
package Library_Management_System;

import java.util.ArrayList;

public class Member {
    private String memberId;
    private String name;
    private String contact;
    private ArrayList<Book> borrowedBooks;

    public Member(String memberId, String name, String contact) {
        this.memberId = memberId;
        this.name = name;
```

```
        this.contact = contact;
        this.borrowedBooks = new ArrayList<>();
    }

    public String getMemberId() { return memberId; }
    public void setMemberId(String memberId) { this.memberId = memberId; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public String getContact() { return contact; }
    public void setContact(String contact) { this.contact = contact; }

    public ArrayList<Book> getBorrowedBooks() { return borrowedBooks; }

    public boolean borrowBook(Book book) {
        if (book.isAvailable()) {
            borrowedBooks.add(book);
            book.setAvailable(false);
            return true;
        }
        return false;
    }

    public boolean returnBook(Book book) {
        if (borrowedBooks.contains(book)) {
            borrowedBooks.remove(book);
            book.setAvailable(true);
            return true;
        }
        return false;
    }

    public void displayInfo() {
        System.out.println("Member ID: " + memberId);
        System.out.println("Name: " + name);
        System.out.println("Contact: " + contact);
        System.out.println("Books Borrowed: " + borrowedBooks.size());

        if (!borrowedBooks.isEmpty()) {
            System.out.println("Borrowed Books:");
            for (Book book : borrowedBooks) {
                System.out.println("  - " + book.getTitle());
            }
        }
        System.out.println();
    }
}
```

LIBRARY CLASS

The Library class acts as the main management system of the library. It maintains collections of books and members using ArrayLists. It provides functionalities to add books, register members, search books, and find members. The class ensures centralized control of library operations.

```
package Library_Management_System;

import java.util.ArrayList;

public class Library {
    private ArrayList<Book> books;
    private ArrayList<Member> members;

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

    public void addBook(Book book) { books.add(book); }
    public void addMember(Member member) { members.add(member); }

    public Book findBookByIsbn(String isbn) {
        for (Book book : books) {
            if (book.getIsbn().equals(isbn)) return book;
        }
        return null;
    }

    public Member findMemberById(String memberId) {
        for (Member member : members) {
            if (member.getMemberId().equals(memberId)) return member;
        }
        return null;
    }

    public ArrayList<Book> searchBooksByTitleOrAuthor(String keyword) {
        ArrayList<Book> results = new ArrayList<>();
        keyword = keyword.toLowerCase();

        for (Book book : books) {
            if (book.getTitle().toLowerCase().contains(keyword) ||
                book.getAuthor().toLowerCase().contains(keyword)) {
                results.add(book);
            }
        }
        return results;
    }
```

```
    public void displayAllBooks() {
        System.out.println("=== ALL BOOKS ===");
        if (books.isEmpty()) {
            System.out.println("No books in the library!");
            return;
        }
        for (Book book : books) book.displayInfo();
    }

    public void displayAvailableBooks() {
        System.out.println("=== AVAILABLE BOOKS ===");
        boolean found = false;
        for (Book book : books) {
            if (book.isAvailable()) {
                book.displayInfo();
                found = true;
            }
        }
        if (!found) System.out.println("No books available at the moment!");
    }
}
```

LIBRARYSYSTEM CLASS

The LibrarySystem class is the main class containing the main() method. It provides a menu-driven console interface for user interaction. This class connects all other classes and controls the overall program flow. It handles user input, validation, and execution of library operations.

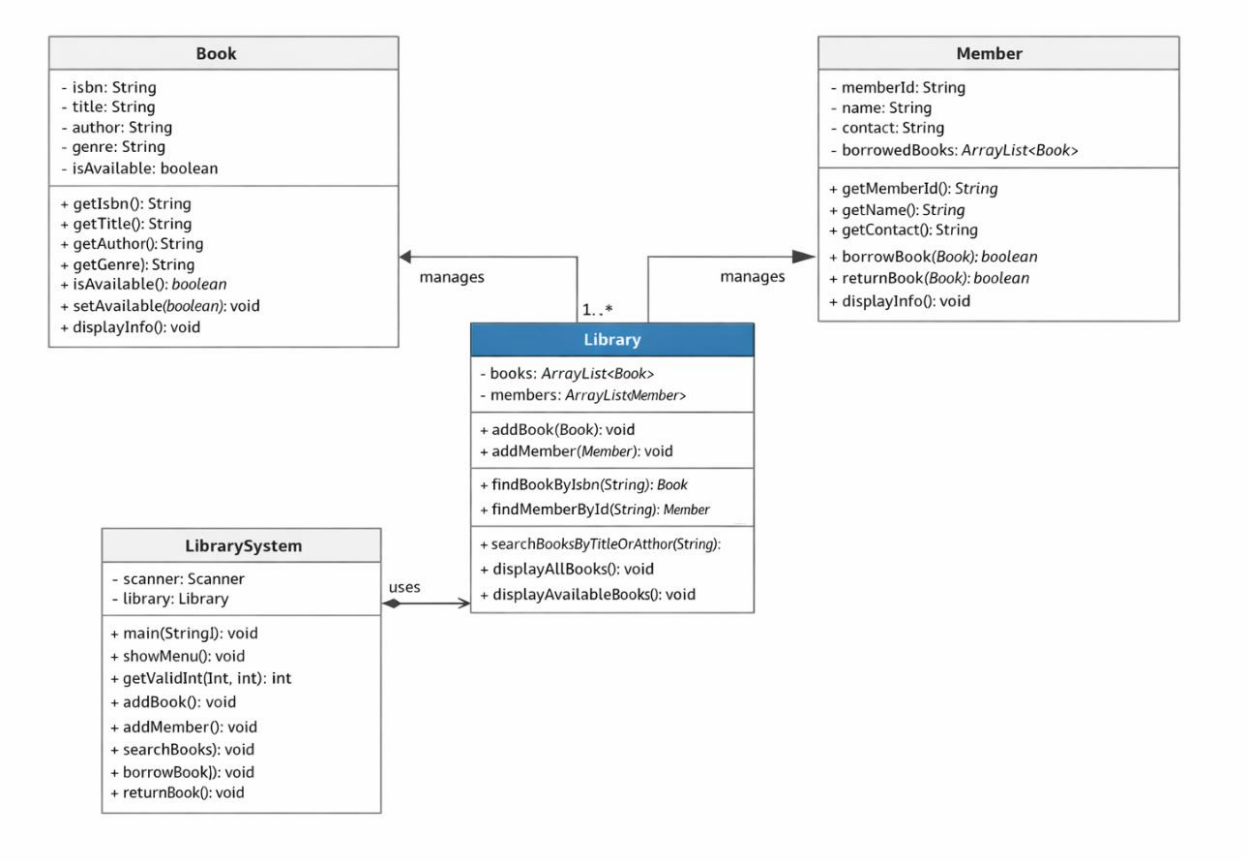
```
package Library_Management_System;

import java.util.Scanner;
import java.util.ArrayList;

public class LibrarySystem {
    private static Scanner scanner = new Scanner(System.in);
    private static Library library = new Library();

    public static void main(String[] args) {
        initializeLibrary();
        boolean running = true;

        while (running) {
            showMenu();
            int choice = getValidInt(1, 8);

            switch (choice) {
```

```
            case 1 -> addBook();
            case 2 -> addMember();
            case 3 -> library.displayAllBooks();
            case 4 -> library.displayAvailableBooks();
            case 5 -> searchBooks();
            case 6 -> borrowBook();
            case 7 -> returnBook();
            case 8 -> {
                running = false;
                System.out.println("Thank you for using Library Management System!");
            }
        }
    }
    scanner.close();
  }

  private static void initializeLibrary() {
    library.addBook(new Book("978-3-16-148410-2", "The Magic of Lost Temple", "Sudha Murthy",
"Story"));
    library.addBook(new Book("978-0-262-03384-8", "Never Never", "Collin Grover", "Fiction"));
    library.addBook(new Book("978-0-13-468599-1", "The God of Small Things", "Arundhati Roy",
"Novel"));
  }

  private static void showMenu() {
    System.out.println("\n=== LIBRARY MANAGEMENT SYSTEM ===");
    System.out.println("1. Add New Book");
    System.out.println("2. Register New Member");
    System.out.println("3. Display All Books");
    System.out.println("4. Display Available Books");
    System.out.println("5. Search Books by Title/Author");
    System.out.println("6. Borrow Book");
    System.out.println("7. Return Book");
    System.out.println("8. Exit");
    System.out.print("Enter your choice: ");
  }

  private static int getValidInt(int min, int max) {
    while (true) {
      try {
        int value = Integer.parseInt(scanner.nextLine());
        if (value >= min && value <= max) return value;
        System.out.print("Please enter number between " + min + " and " + max + ": ");
      } catch (Exception e) {
        System.out.print("Invalid input! Enter a number: ");
      }
    }
  }
```

```
private static void addBook() {
    System.out.println("\n=== ADD NEW BOOK ===");
    System.out.print("Enter ISBN: "); String isbn = scanner.nextLine();
    System.out.print("Enter Title: "); String title = scanner.nextLine();
    System.out.print("Enter Author: "); String author = scanner.nextLine();
    System.out.print("Enter Genre: "); String genre = scanner.nextLine();

    Book book = new Book(isbn, title, author, genre);
    library.addBook(book);
    System.out.println("Book added successfully!");
}

private static void addMember() {
    System.out.println("\n=== REGISTER NEW MEMBER ===");
    System.out.print("Enter Member ID: "); String id = scanner.nextLine();
    System.out.print("Enter Name: "); String name = scanner.nextLine();
    System.out.print("Enter Contact: "); String contact = scanner.nextLine();

    Member member = new Member(id, name, contact);
    library.addMember(member);
    System.out.println("Member registered successfully!");
}

private static void searchBooks() {
    System.out.println("\n=== SEARCH BOOKS ===");
    System.out.print("Enter search keyword: ");
    String keyword = scanner.nextLine();

    ArrayList<Book> results = library.searchBooksByTitleOrAuthor(keyword);
    if (results.isEmpty()) System.out.println("No books found!");
    else {
        System.out.println("Search Results:");
        for (Book book : results) book.displayInfo();
    }
}

private static void borrowBook() {
    System.out.println("\n=== BORROW BOOK ===");
    System.out.print("Enter Member ID: "); String memberId = scanner.nextLine();
    Member member = library.findMemberById(memberId);
    if (member == null) { System.out.println("Member not found!"); return; }

    System.out.print("Enter Book ISBN: "); String isbn = scanner.nextLine();
    Book book = library.findBookByIsbn(isbn);
    if (book == null) { System.out.println("Book not found!"); return; }

    if (member.borrowBook(book)) System.out.println("Book borrowed successfully!");
```

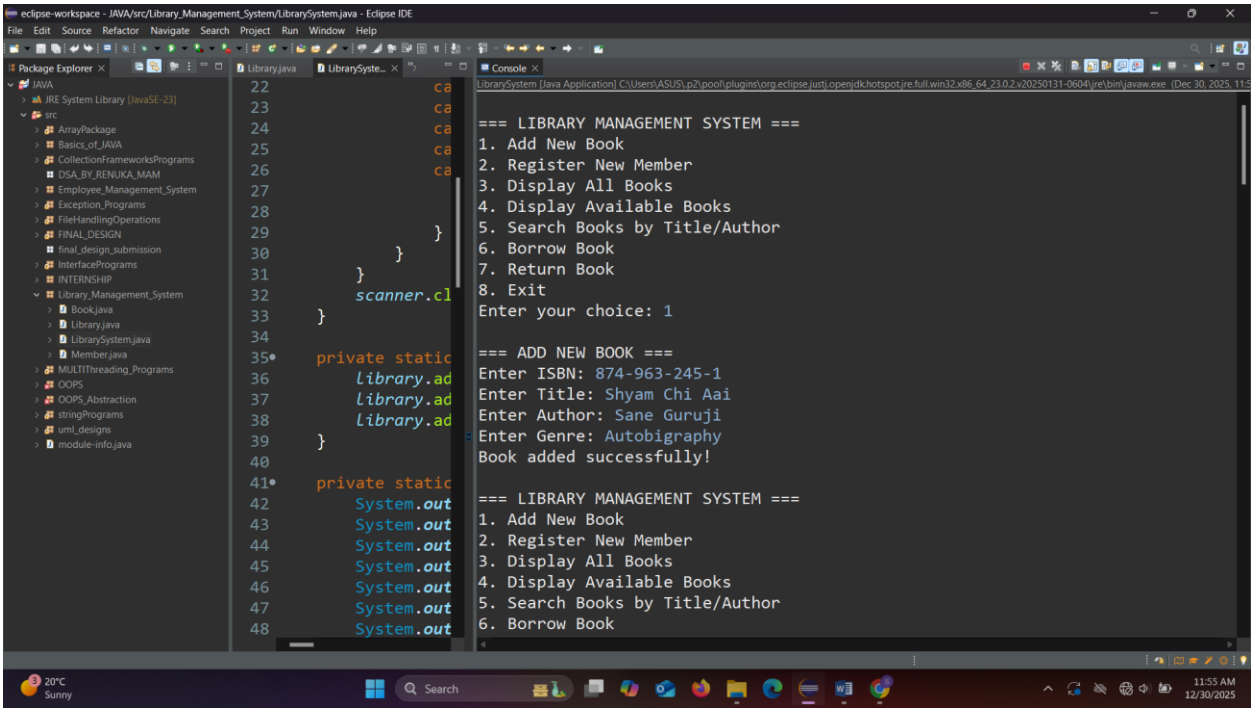
```
        else System.out.println("Book is not available!");
    }

    private static void returnBook() {
        System.out.println("\n=== RETURN BOOK ===");
        System.out.print("Enter Member ID: "); String memberId = scanner.nextLine();
        Member member = library.findMemberById(memberId);
        if (member == null) { System.out.println("Member not found!"); return; }

        System.out.print("Enter Book ISBN: "); String isbn = scanner.nextLine();
        Book book = library.findBookByIsbn(isbn);
        if (book == null) { System.out.println("Book not found!"); return; }

        if (member.returnBook(book)) System.out.println("Book returned successfully!");
        else System.out.println("Return failed! The member did not borrow this book.");
    }
}
```

5. UML DIAGRAM

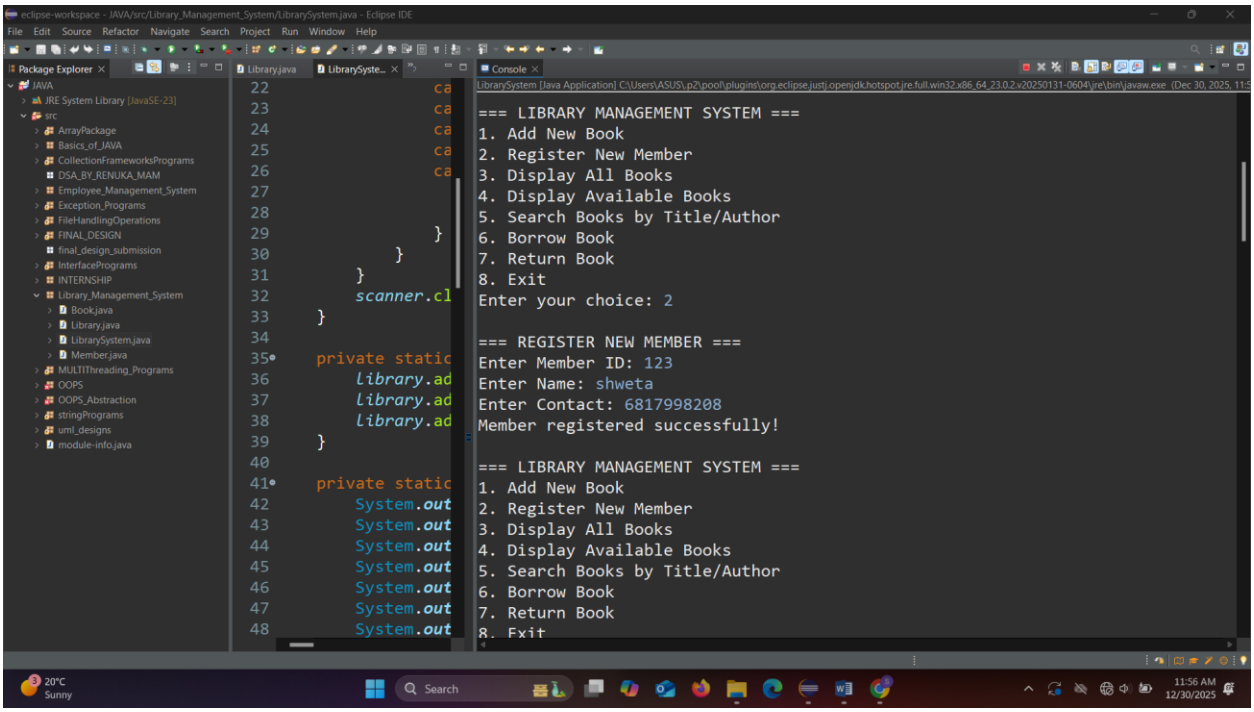


6. SCREENSHOTS OF WORKING APPLICATION

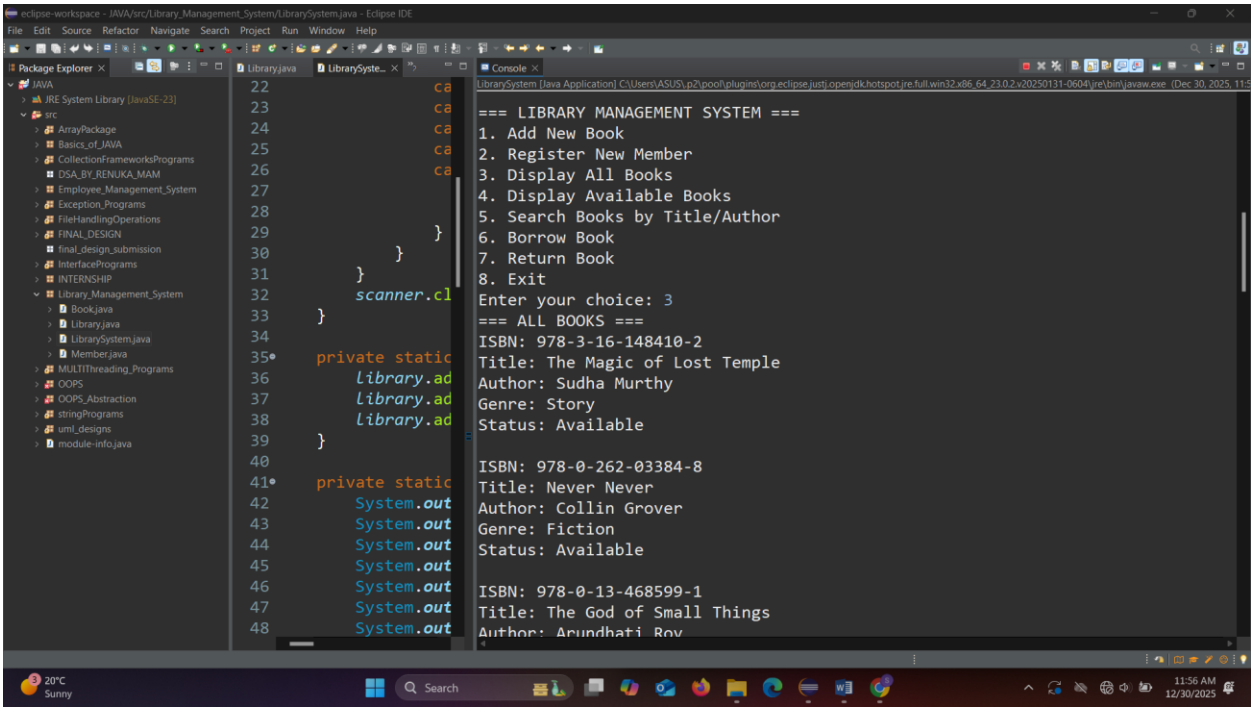
ADD BOOK

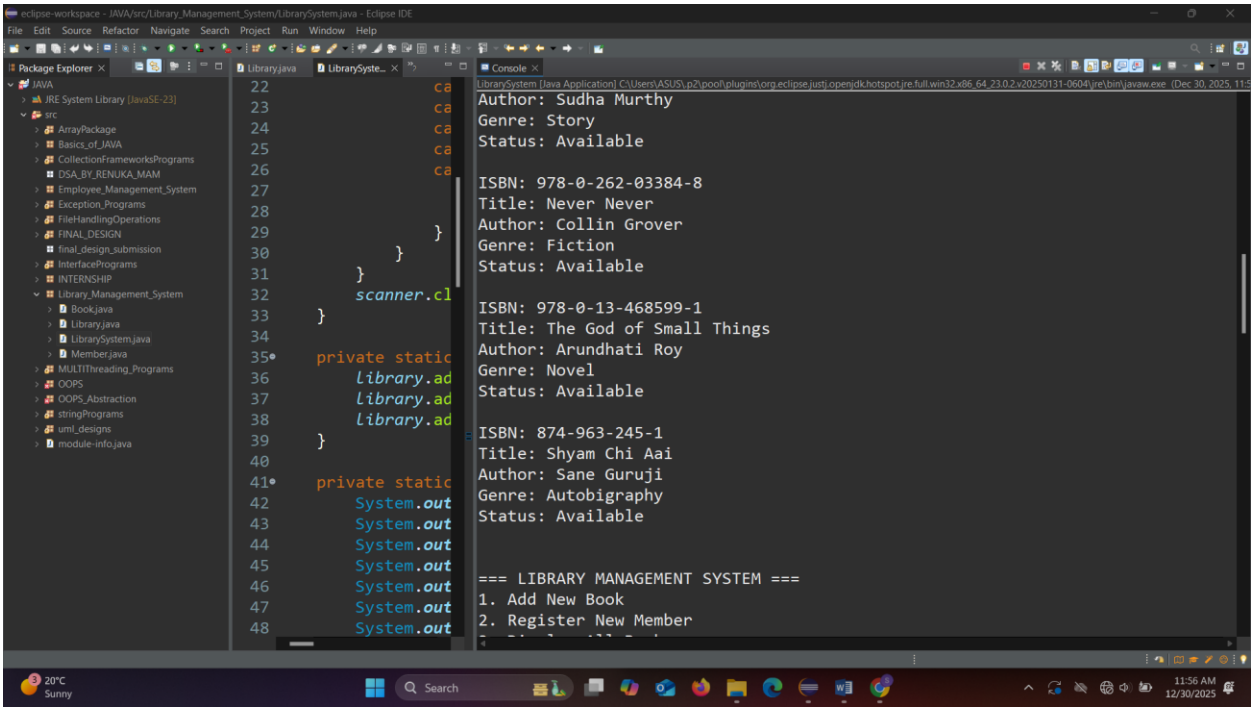


ADD MEMBER

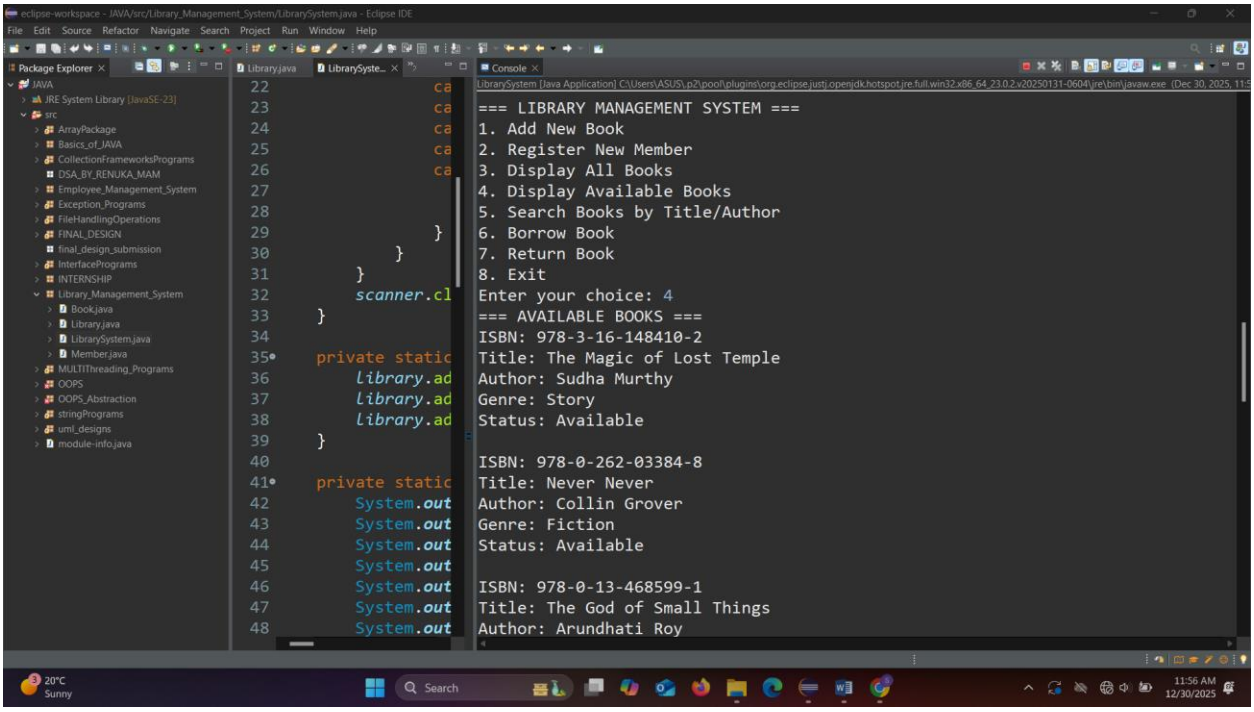


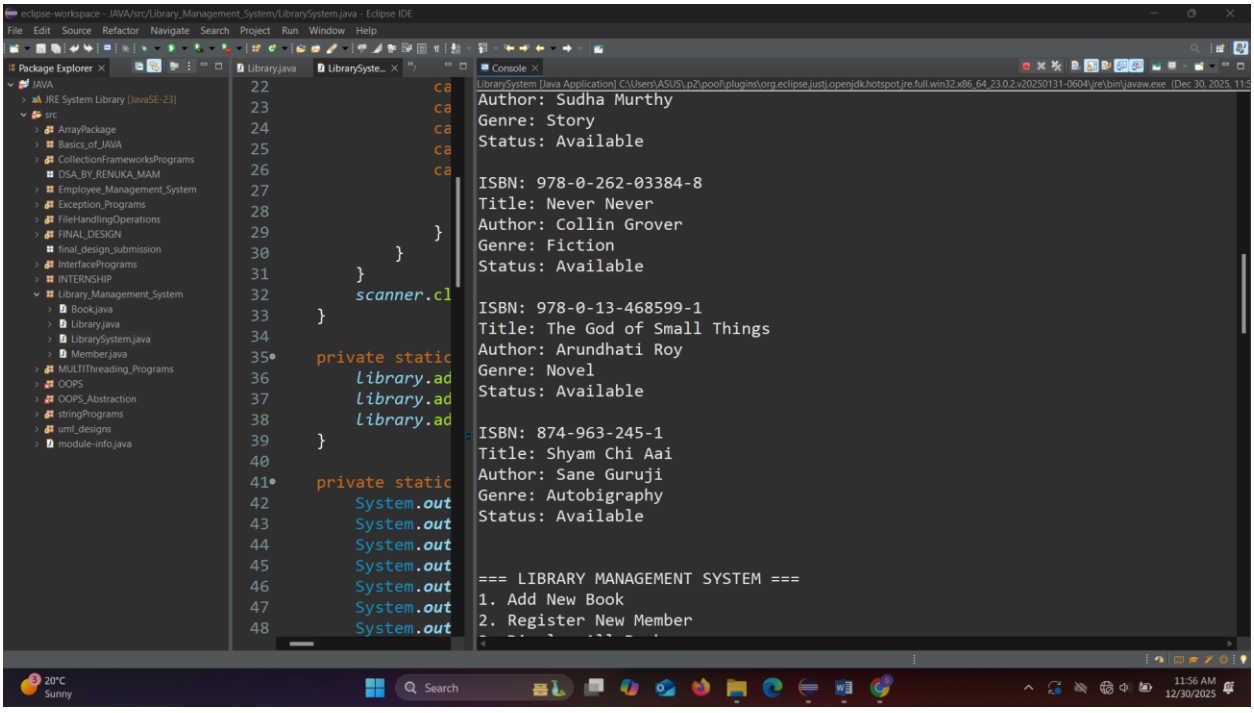
DISPLAY ALL BOOKS



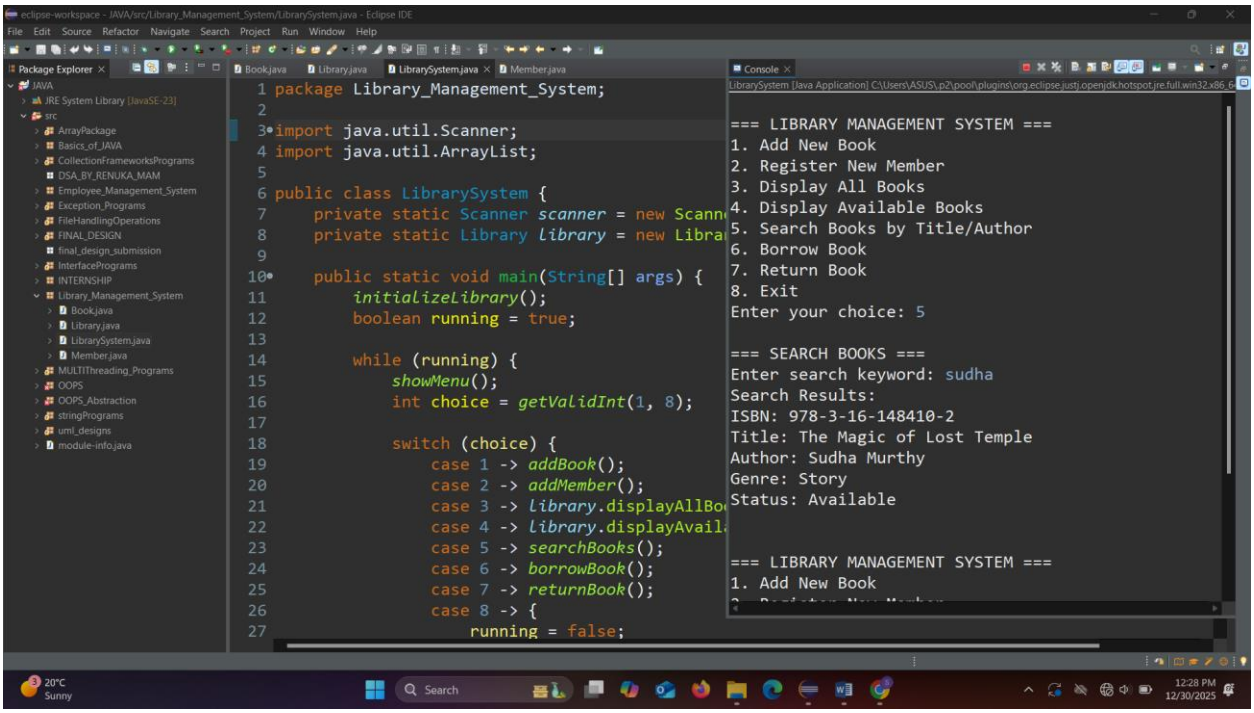


DISPLAY AVAILABLE BOOKS

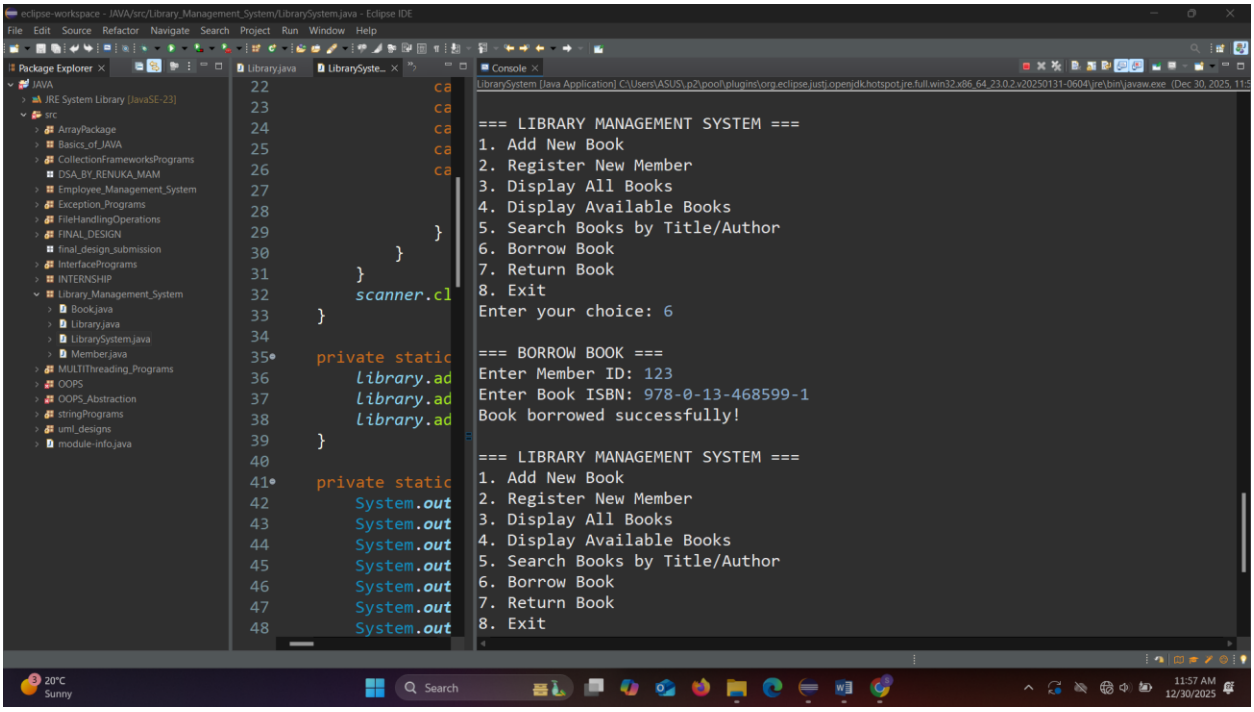




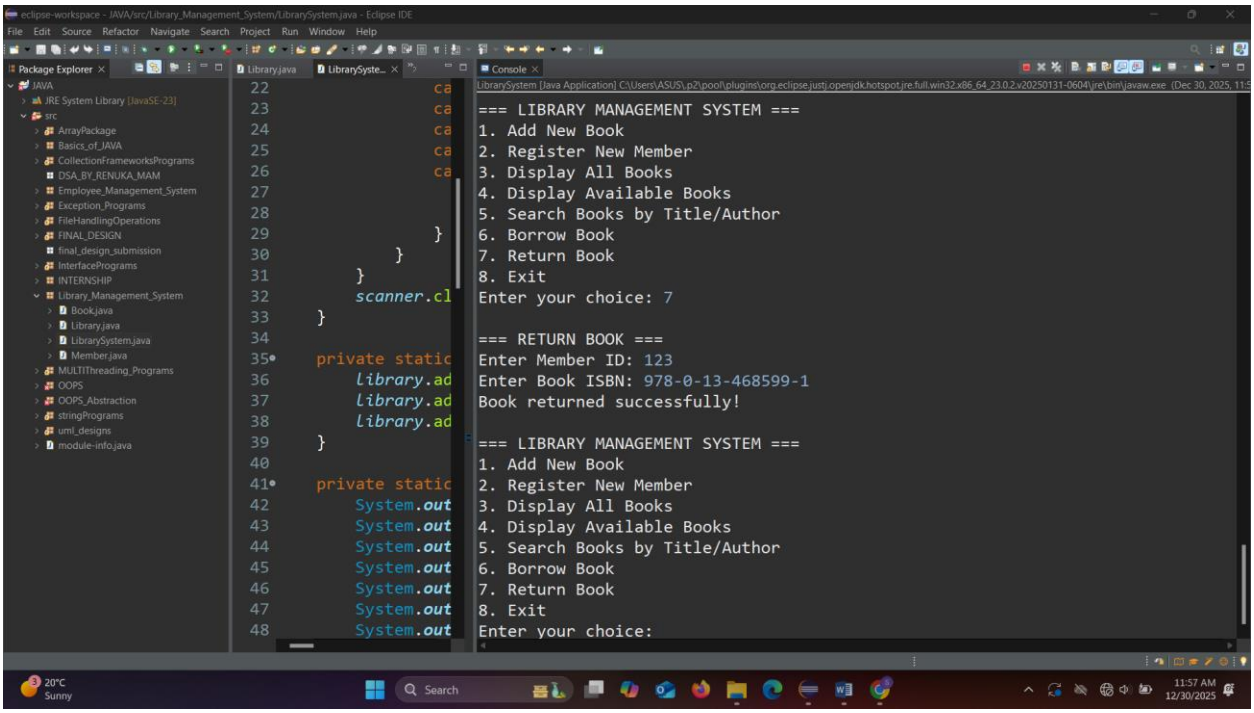
SEARCH BOOKS BY TITLE/AUTHOR



BORROW BOOK



RETURN BOOK



7. SAMPLE EXAMPLE

```
ISBN: 978-3-16-148410-2
Title: The Magic of Lost Temple
Author: Sudha Murthy
Genre: Story
Status: Available

ISBN: 978-0-262-03384-8
Title: Never Never
Author: Collin Grover
Genre: Fiction
Status: Available

ISBN: 978-0-13-468599-1
Title: The God of Small Things
Author: Arundhati Roy
Genre: Novel
```

```
Status: Available
```

8. HOW THE PROJECT MEETS TECHNICAL REQUIREMENTS

- Book Class Creation
  - The Book class includes ISBN, title, author, genre, and availability attributes.
  - It represents individual books in the library.
- Member Class Creation
  - The Member class includes memberId, name, contact, and borrowedBooks.
  - It stores member details and tracks borrowed books.
- Encapsulation Implementation
  - All class variables are declared private.
  - Public getter and setter methods are used to access and modify data safely.
- Borrow and Return Functionality
  - Members can borrow books only if they are available.
  - Returned books are marked available again.
- Library Class for Management
  - The Library class manages collections of books and members.
  - It performs add, search, and retrieval operations.
- Search Functionality
  - Books can be searched using keywords matching title, author
  - Search is case-insensitive for better usability.
- Book Availability Validation
  - A book cannot be borrowed if it is already borrowed.
  - Availability status is updated automatically.
- Menu-Driven Interface
  - A console-based menu allows users to perform operations easily.
  - User input is validated to avoid incorrect entries.

9 . CONCLUSION

The Library Management System efficiently manages books and members, allowing adding, searching, borrowing, and returning books with proper availability validation. It demonstrates core object-oriented principles like encapsulation and modularity, and provides a user-friendly menu interface. The project lays a strong foundation for further enhancements, such as reporting and database integration.