



Department Of Computer Science and Engineering

Course Title: Artificial Intelligence and Expert Systems Lab

Course Code: CSE 404

Date of Submission: 22/02/2025

Submitted To:

Noor Mairukh Khan Arnob
Lecturer,
Department Of Computer Science &
Engineering

Submitted By:

Jannatun Saumoon
ID: 21201066
Sec: B2

i) Problem Title:

Development of a Knowledge-Based “Library Book System” Using Prolog.

ii) Problem Description:

Managing library books, tracking borrowed items, and recommending relevant reads can be challenging without an efficient system. This knowledge-based system organizes books, tracks availability, manages members, and automates borrowing operations using logical rules.

Key Features

- **Book & Member Management:** Store book details, authors, and categories; verify library members.
- **Availability & Borrowing:** Track available and borrowed books, manage returns.
- **Query & Recommendations:** Search books by author/category and suggest similar reads.
- **Rule-Based Operations:** Automate availability checks, borrowing history, and returns using Prolog rules.

iii) Tools and Languages Used:

Programming Language:

- **Prolog** – Used for defining facts, rules, and queries in the knowledge base.

Development Environment:

- **SWI-Prolog** – A widely used Prolog interpreter for running and testing queries.

Database Management:

- **Prolog Knowledge Base** – Stores book details, member records, and borrowing history as facts and rules.

Diagram:

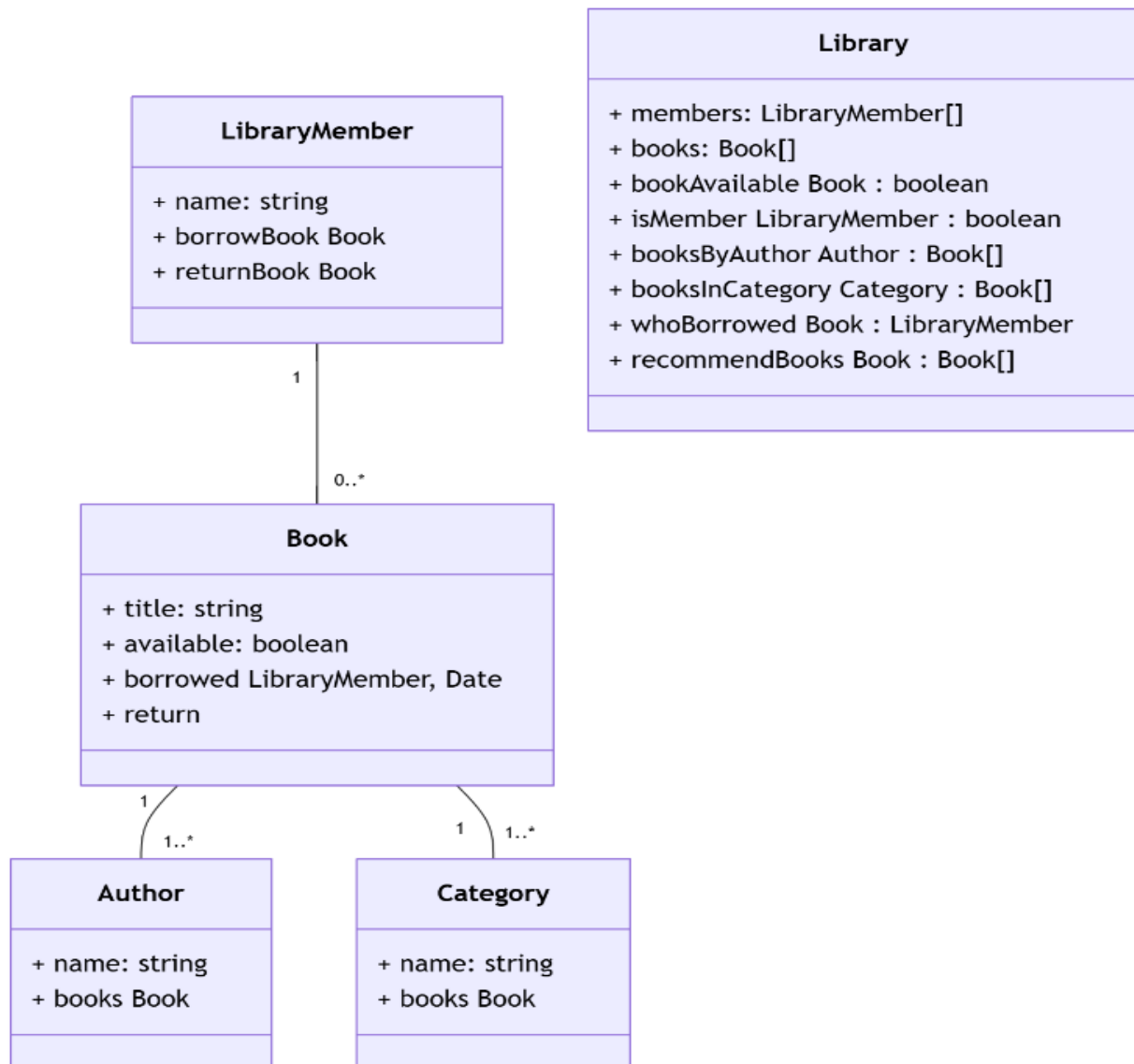
- **Draw.io**- Used to design the Tree Figure and ER diagram.

Text Editor:


- **Notepad** – Alternative lightweight text editor for Prolog scripting.

iv. Diagram:

Class Diagram:



v. Sample Source Code pl file:

 Library Book System.pl [modified]
File Edit Browse Compile Prolog Pce Help
Library Book System.pl [modified]

```
/* Facts */

/* Books and their authors */
author('Harry Potter and the Sorcerer\'s Stone', 'J.K. Rowling').
author('The Hobbit', 'J.R.R. Tolkien').
author('The Da Vinci Code', 'Dan Brown').
author('The Alchemist', 'Paulo Coelho').
author('A Game of Thrones', 'George R.R. Martin').
author('Murder on the Orient Express', 'Agatha Christie').
author('The Great Gatsby', 'F. Scott Fitzgerald').
author('To Kill a Mockingbird', 'Harper Lee').

/* Books and their categories */
category('Harry Potter and the Sorcerer\'s Stone', fantasy).
category('The Hobbit', fantasy).
category('The Da Vinci Code', thriller).
category('The Alchemist', philosophy).
category('A Game of Thrones', fantasy).
category('Murder on the Orient Express', mystery).
category('The Great Gatsby', fiction).
category('To Kill a Mockingbird', fiction).

/* Library members */
member('Alice').
member('Bob').
member('Charlie').
member('David').
```

```
member('David').

/* Book availability */
available('The Hobbit').
available('The Da Vinci Code').
available('The Alchemist').
available('A Game of Thrones').
available('The Great Gatsby').


/* Borrowed books and due dates */
borrowed('Harry Potter and the Sorcerer\'s Stone', 'Alice', '2025-03-10').
borrowed('Murder on the Orient Express', 'Bob', '2025-03-15').
borrowed('To Kill a Mockingbird', 'Charlie', '2025-03-20').

/* Rules */

/* Check if a book is available */
book_available(Book) :-
    available(Book),
    \+ borrowed(Book, _, _).

/* Check if a person is a library member */
is_member(Person) :-
    member(Person).

/* Find books by an author */
books_by_author(Author, Book) :-
    author(Book, Author).
```

 Library Book System.pl [modified]

File Edit Browse Compile Prolog Pce Help

Library Book System.pl [modified]

```
/* Find books by an author */
books_by_author(Author, Book) :-
    author(Book, Author).

/* Find books in a category */
books_in_category(Category, Book) :-
    category(Book, Category).

/* Who borrowed a book and when? */

who_borrowed(Book, Person, Date) :-
    borrowed(Book, Person, Date).

/* Return a book */
:- dynamic available/1.
:- dynamic borrowed/3.

return_book(Book, Person) :-
    borrowed(Book, Person, _),
    retract(borrowed(Book, Person, _)),
    assert(available(Book)).

/* Recommend books in the same category */
recommend(Book, SuggestedBook) :-
    category(Book, Category),
    category(SuggestedBook, Category),
    Book \== SuggestedBook.
```

vi. Sample Input/output:

1.Find books availability:

Input: ?- book_available('The Hobbit').

Output: true.

Input: ?- book_available('To Kill a Mockingbird').

Output: false.

2.Find books author:

Input: ?- books_by_author('J.K. Rowling', Book).

Output: Book = 'Harry Potter and the Sorcerer\'s Stone'.

Input: ?- books_by_author('Harper Lee', Book).

Output: Book = 'To Kill a Mockingbird'.

3.Find books category:

Input: ?- books_in_category(fantasy, Book).

Output: Book = 'Harry Potter and the Sorcerer\'s Stone' .

Input: ?- books_in_category(fiction, Book).

Output: Book = 'The Great Gatsby' ;

Book = 'To Kill a Mockingbird'.

4.Find who borrow a book:

Input: ?- who_borrowed('To Kill a Mockingbird', Person, Date).

Output: Person = 'Charlie',

Date = '2025-03-20'.

Input: ?- who_borrowed('Murder on the Orient Express', Person, Date).

Output: Person = 'Bob',

Date = '2025-03-15'.

5.Find if return a book:

Input: ?- return_book('To Kill a Mockingbird', 'Charlie').

Output: true.

6.Find recommend for a book:

Input: ?- recommend('The Hobbit', Suggestion).

Output: Suggestion = 'Harry Potter and the Sorcerer\'s Stone'

Input: ?- book_available('A Game of Thrones').

Output: true.

Input: ?- book_available('Harry Potter and the Sorcerer\'s Stone').

Output: false.

Input: ?- who_borrowed('Harry Potter and the Sorcerer\'s Stone', Person, Date).

Output: Person = 'Alice',

Date = '2025-03-10'.

vii. Conclusion:

This Prolog-based library management system efficiently tracks book availability, borrowing records, and recommendations using logical rules. It simplifies book searches, member validation, and borrowing management while ensuring seamless library operations. The rule-based approach enables dynamic data handling and intelligent decision-making for book suggestions and availability tracking.

Challenges

- **Scalability Issues:** Handling large datasets can be inefficient.
- **Dynamic Updates:** Requires explicit assertions and retractions.

- **Limited UI Support:** Needs integration for a user-friendly interface.
- **Concurrency Problems:** Managing multiple users simultaneously is complex.
- **Query Optimization:** Complex queries may need performance tuning.