Library Management System - Project Report

📌 Introduction

Libraries require efficient book management systems to track available and borrowed books. This project, **Library Management System**, is implemented using Python in a Jupyter Notebook environment. It leverages **arrays** and **linked lists** to manage books efficiently, ensuring smooth search, borrow, and return operations.

🛠️ Implementation Details

The system consists of the following components:

1. **Book Class**: Represents a book with attributes like `book_id`, `title`, `author`, and `availability status`.
2. **Library Class**: Manages book records using an array for storage and a linked list for tracking borrowed books.
3. **Linked List Class**: Helps manage borrowed books dynamically, ensuring optimized book return operations.
4. **Functions**:
   - `add_book()`: Adds a new book to the library.
   - `search_book()`: Finds a book by title.
   - `borrow_book()`: Marks a book as borrowed and adds it to the linked list.
   - `return_book()`: Removes a book from the borrowed list and marks it as available.
   - `display_books()`: Shows the current status of all books.

📂 Data Structures Used

- **Arrays**: Used to store all books, ensuring direct access by index.
- **Linked Lists**: Used for tracking borrowed books dynamically, reducing the overhead of shifting elements in case of deletions.

⚡ Performance Analysis

Operation	Data Structure Used	Time Complexity
Add Book	Array	O(1)
Search Book	Linear Search (Array)	O(n)
Borrow Book	Linked List Append	O(1)
Return Book	Linked List Delete	O(n)
Display Books	Array Traversal	O(n)

📊 Conclusion

This **Library Management System** efficiently handles book records using fundamental **data structures**. By combining **arrays** for book storage and **linked lists** for tracking borrowed books, it ensures efficient search, borrowing, and return functionalities. The system can be extended further with a database for persistent storage.