Data Structures Project

Library Management System

Ebrahim Khodadadi

January 2026

Project Objective

The goal of this project is to design and implement a simple library management system in C++ that supports the core operations of a real-world library, including adding, removing, searching, and sorting books, managing book loan requests, and maintaining an operation history.

The program has been developed with a strong focus on key concepts from the Data Structures course. It utilizes dynamic data structures such as a **doubly linked list**, **queue**, and **stack** to ensure efficiency, flexibility, and proper memory management. The implementation follows object-oriented programming (OOP) principles and features an interactive menu-based interface that allows users to easily interact with the system.

Overall Program Structure

The program consists of several main classes:

- **Class Book**: Represents a single book with attributes including a unique code, title, author, publication year, and availability status (available / borrowed).
- **Class Library**: Manages the collection of books using a **doubly linked list**.
- **Class LoanQueue**: Handles loan requests using a **queue** implemented with a singly linked list.
- **Class HistoryStack**: Maintains a history of the most recent operations using a **stack** with a fixed capacity (maximum 5 operations).

Main Components and Functionality

1. **Book Storage and Management** All books are stored in a doubly linked list. This structure enables fast insertion and deletion of books and allows the collection size to grow dynamically (unlike fixed-size arrays). Key operations include:
    - Adding a new book
    - Removing a book by its unique code
    - Searching for a book by its unique code
    - Displaying all books
2. **Book Sorting** The system provides sorting capabilities based on two criteria:
    - Sorting by book code (ascending)
    - Sorting by publication year (ascending) Sorting is performed directly on the linked list using a straightforward and understandable algorithm (Bubble Sort).
3. **Loan Management** Loan requests are stored in a **queue** following the First-In-First-Out (FIFO) principle. When the user chooses to process the next request:
    - The first request is removed from the queue.
    - If the requested book is available, its status is changed to "borrowed".
    - If the book is not available, the request is returned to the end of the queue for later processing. Book returns are also supported, changing the status back to "available".

Implementing the queue with a linked list eliminates any size limitation on the number of requests and avoids the overflow issues common in array-based queues.

4. **Operation History** To meet the requirement of displaying recent operations, a **stack** with a capacity of 5 entries is used. After most operations (e.g., adding, removing, loaning, sorting, etc.), a brief description of the action is automatically pushed onto the stack. When the stack reaches its limit, the oldest entry is automatically discarded to make room for the new one. Users can view the history at any time, with the most recent operations displayed first (LIFO order).