# Lab4 Văn Minh Thức 2031200067

**Library Management System Report**

## 1. Introduction

This report outlines the design and implementation of a console-based Library Management System developed in C# following SOLID principles. The system manages books and readers using in-memory sample data, supporting core functionalities like adding books, searching, lending, returning, and generating reports.

## 2. System Design

### 2.1 Core Functionalities

- **Add New Books**: Administrators add books with Title, Author, Category, and Quantity.
- **Search for Books**: Users search by Title or Category.
- **Lend Books**: Readers borrow up to 3 books if stock is available.
- **Return Books**: Readers return books, updating stock.
- **Generate Reports**: Displays readers and their borrowed books.

### 2.2 System Requirements

- Unique IDs for books and readers.
- Extensibility for new book types (e.g., eBooks, magazines) without modifying existing code.

### 2.3 Technical Design

- **Components**:
  - **Book Management**: Handles book operations.
  - **Reader Management**: Manages reader actions.
  - **Report Generation**: Produces borrowing reports.
- **SOLID Principles**:
  - **S**: Single responsibility per class (e.g., BookManager for books).
  - **O**: Open for extension via abstract Book class (e.g., PhysicalBook, EBook).
  - **L**: Substitutable book types inherit from Book.

    - **I**: Small, specific interfaces (e.g., IBook, IReader).
    - **D**: Dependencies on abstractions (e.g., IReaderManager in ReportService).
- **Structure**:
    - **Interfaces**: IBook, IBookManager, IReader, IReaderManager, IReportService.
    - **Models**: Book (abstract), PhysicalBook, EBook, Reader.
    - **Services**: BookManager, ReaderManager, ReportService.
    - **Data**: SampleData for testing.

## 3. Implementation

### 3.1 Class Design

- **Fields and Properties**: Separated private fields (e.g., _id) and public properties (e.g., Id => _id), with validation (e.g., Quantity ensures non-negative values).
- **Interfaces**: Define contracts for extensibility and communication.
- **Sample Data**: Preloaded books (e.g., "C# Basics", ID: 1, Quantity: 5) and readers (e.g., "Alice", ID: 1).

### 3.2 Key Features

- **Book Management**: BookManager adds, searches, and updates stock using List<IBook>.
- **Reader Management**: ReaderManager enforces a 3-book limit and updates stock via IBookManager.
- **Report Generation**: ReportService uses IReaderManager.GetReaders() to list borrowings.

## 4. Extensibility

- **New Book Types**: Add classes like Magazine inheriting from Book.
- **Reservations**: Implement IReservationService and integrate with ReaderManager.

## 5. Evaluation

- **SOLID Adherence**: Fully applied across design and implementation.
- **Functionality**: All requirements (add, search, lend, return, report) met.

- **Maintainability**: Modular structure with interfaces ensures easy updates.
- **Extensibility**: Open for new features without altering core code.

### 6. Conclusion

The system successfully meets its goals as a simple, extensible library management tool. Future enhancements could include a database or additional features like reservations, leveraging the current design’s flexibility.