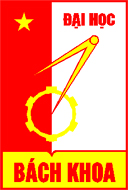
**ĐẠI HỌC BÁCH KHOA HÀ** **NỘI**

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

──────── \* ───────



**BÀI TẬP LỚN**

HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

*(Mã học phần: IT3103)*

*Đề tài số 6:*

**QUẢN LÝ CÁN BỘ MỘT TRƯỜNG HỌC**

**Sinh viên thực hiện**

| STT | Họ và tên | Mã sinh viên |
| --- | --- | --- |
| 1 | Đỗ Gia Huy | 20226109 |
| 2 | Trần Đại Hiệp | 20226081 |
| 3 | Trần Phạm Minh Đức | 20226077 |
| 4 | Nguyễn Minh Quang | 20226123 |

**Giảng viên hướng dẫn: PGS.TS Lê Đức Hậu**

***Hà Nội, tháng 12 năm 2024***

# MỤC LỤC

[**MỤC LỤC 2**](#_heading=)

[**CHƯƠNG 1. KHẢO SÁT, ĐẶC TẢ YÊU CẦU BÀI TOÁN 3**](#_heading=h.2et92p0)

[1.1. Mô tả yêu cầu bài toán 3](#_heading=h.tyjcwt)

[1.2. Biểu đồ Use case tổng quan: 4](#_heading=h.1t3h5sf)

[1.3. Biểu đồ phân rã mức 1 4](#_heading=h.852bgnpy2lru)

[1.4. Biểu đồ phân rã mức 2 6](#_heading=h.g5jsyoe6fss5)

[1.5. Đặc tả Use Case 7](#_heading=h.sqekrvtl62cn)

[**CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ BÀI TOÁN 11**](#_heading=)

[**2.1. Thiết kế Cơ sở dữ liệu 11**](#_heading=)

[2.1.1 Sơ đồ ERD 11](#_heading=h.1ksv4uv)

[**2.1.2 Phân tích cơ sở dữ liệu 11**](#_heading=)

[**2.2. Biểu đồ lớp 13**](#_heading=)

[2.2.1. Service Layer: 13](#_heading=h.puu6lhtrec2h)

[2.2.2. DAO Layer: 14](#_heading=h.4giql41kad2q)

[**CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MINH HOẠ 15**](#_heading=h.2bn6wsx)

[3.1. Kết quả chương trình minh hoạ 15](#_heading=h.2p2csry)

[3.2. Giao diện chương trình 15](#_heading=h.r6ml3fodi5gj)

[3.2.1. Giao diện admin tổng 15](#_heading=h.4wiv9e245k3q)

[3.2.2. Giao diện thêm cán bộ mới 15](#_heading=h.s2fx6q17usnz)

[3.2.3 Giao diện của giáo viên 18](#_heading=h.5p831h3bfy96)

[**TÀI LIỆU THAM KHẢO 23**](#_heading=)

# CHƯƠNG 1. KHẢO SÁT, ĐẶC TẢ YÊU CẦU BÀI TOÁN

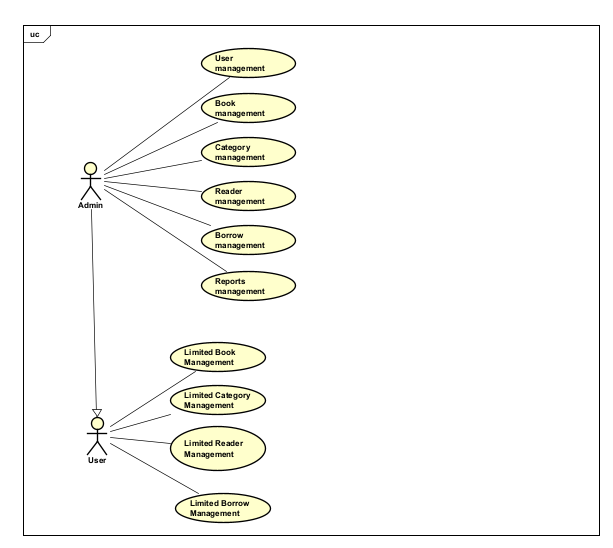
## Mô tả yêu cầu bài toán

Hệ thống quản lý thư viện được xây dựng nhằm hỗ trợ việc quản lý thông tin và hoạt động của thư viện một cách hiệu quả, chính xác và dễ dàng. Hệ thống cung cấp các chức năng cơ bản như quản lý sách, quản lý độc giả, mượn và trả sách, giúp giảm thiểu công việc thủ công và nâng cao chất lượng phục vụ của thư viện

**Mục tiêu của hệ thống:**

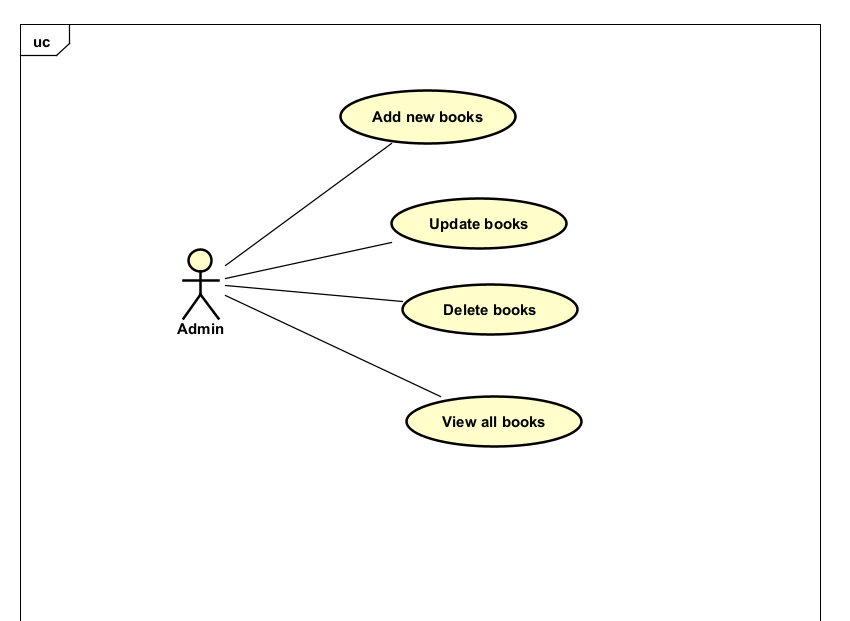
1. **Quản lý thông tin sách và thể loại sách**
   * Hệ thống tổ chức thông tin sách, bao gồm: mã sách, tên sách, tác giả, nhà xuất bản, số lượng và thể loại. Sách được phân loại rõ ràng theo thể loại (ví dụ: viễn tưởng, lịch sử, khoa học,...) để dễ dàng tìm kiếm và quản lý.
2. **Quản lý thông tin độc giả**
   * Lưu trữ thông tin độc giả như mã độc giả, tên, ngày sinh, địa chỉ,... nhằm phục vụ việc theo dõi và quản lý mượn trả sách
3. **Quản lý mượn và trả sách**
   * Hỗ trợ tạo và cập nhật thông tin mượn trả sách, bao gồm ngày mượn, ngày trả dự kiến và ngày trả thực tế. Hệ thống đảm bảo kiểm tra tình trạng sách còn lại, đồng thời theo dõi sách bị hư hỏng hoặc mất mát.
4. **Hỗ trợ tìm kiếm và thống kê**
   * Cung cấp chức năng tìm kiếm sách theo các tiêu chí như tên sách, tác giả hoặc thể loại. Hệ thống cũng cho phép thống kê sách được mượn nhiều nhất, số lượng sách hiện có, và tình trạng sách mượn.

## Biểu đồ Use case tổng quan:

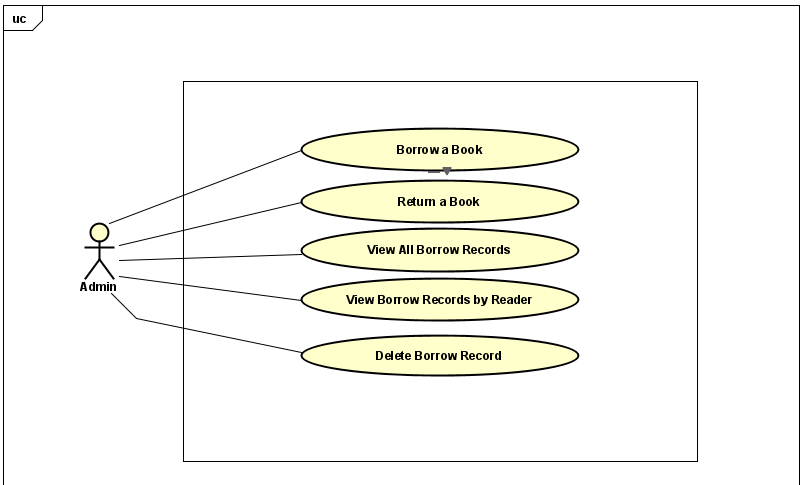


## Biểu đồ phân rã mức 1

**1.3.1. Phân rã use case “Books Management” của Actor Admin:**

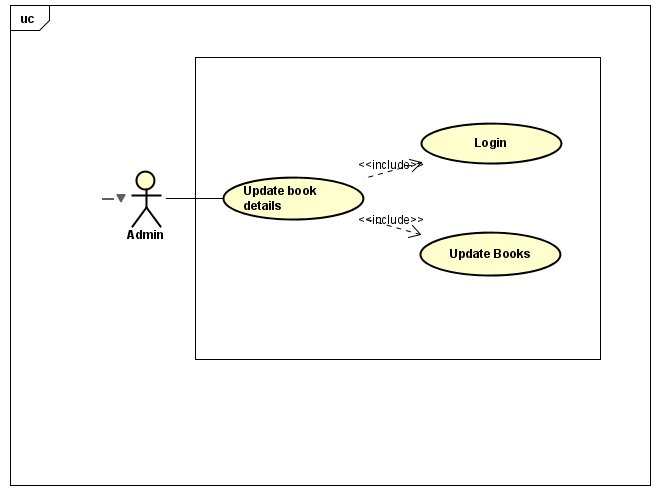


**1.3.2. Phân rã use case “Limited Borrow Management” của Actor User:**



## Biểu đồ phân rã mức 2

Phân rã mức 2 use case Borrow Book của user



Phân rã mức 2 usecase Update book details

## Đặc tả Use Case

| **Use Case “Borrow book”**   1. **Mã use case**   UC001   1. **Mô tả ngắn gọn**   Trường hợp sử dụng này mô tả sự tương tác giữa người đọc và hệ thống khi người đọc muốn mượn sách từ thư viện.   1. **Tác nhân**   **Reader (Người đọc)**  **4. Tiền điều kiện:** Reader đã đăng nhập vào hệ thống  **5. Luồng sự kiện cơ sở**   1. **Reader** chọn chức năng "Borrow Book" (Mượn sách). 2. Hệ thống yêu cầu **Reader** chọn sách muốn mượn từ danh sách sách có sẵn. 3. **Reader** chọn sách muốn mượn và nhập thông tin cần thiết (như ngày mượn, ngày trả dự kiến). 4. Hệ thống lưu thông tin mượn và thông báo mượn sách thành công, hiển thị các thông tin liên quan đến sách mượn. 5. Kết thúc UC.   **6. Luồng sự kiện thay thế**  **Bảng N-Các luồng sự kiện thay thế cho thứ tự UC Place**   | **No** | **Vị trí** | **Điều kiện** | **Hành động** | **Vị trí quay lui** | | --- | --- | --- | --- | --- | | 1 | Bước 3 | Nếu số lượng sách mượn đã hết hoặc sách không có sẵn. | Hệ thống thông báo sách không có sẵn, yêu cầu **Reader** chọn sách khác. | Bước 2 | | 2 | Bước 3 | Nếu **Reader** nhập thông tin sai (ngày mượn hoặc ngày trả dự kiến không hợp lệ). | Hệ thống yêu cầu **Reader** nhập lại thông tin hợp lệ. | Bước 2 | | 3 | Bước 3 | Nếu số lượng sách mượn của **Reader** đã vượt quá giới hạn cho phép. | Hệ thống thông báo rằng **Reader** không thể mượn thêm sách. | Kết thúc UC | | 4 | Bước 3 | Nếu thông tin mượn sách không đầy đủ hoặc không hợp lệ (ví dụ, thiếu ngày trả sách). | Hệ thống yêu cầu **Reader** nhập đầy đủ thông tin. | Bước 2 | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |

| **Use Case “Update book details”**   1. **Mã use case**   UC002   1. **Mô tả ngắn gọn**   Trường hợp sử dụng này mô tả sự tương tác giữa người quản lý và hệ thống khi người quản lý muốn cập nhật thông tin chi tiết của một cuốn sách trong thư viện.   1. **Tác nhân**   **Admin (Người quản lý)**  **4. Tiền điều kiện:** Admin đã đăng nhập vào hệ thống.  Admin có quyền cập nhật thông tin sách.  **5. Luồng sự kiện cơ sở**   1. **Admin** chọn chức năng "Update Book Details" (Cập nhật chi tiết sách). 2. **Admin** nhập mã sách cần cập nhật. 3. Hệ thống hiển thị thông tin hiện tại của sách để **Admin** chỉnh sửa. 4. **Admin** chỉnh sửa các thông tin sách như tiêu đề, tác giả, nhà xuất bản, số lượng, và thể loại. 5. Hệ thống cập nhật thông tin sách vào cơ sở dữ liệu. 6. Hệ thống thông báo cho **Admin** rằng việc cập nhật sách đã thành công. 7. Kết thúc UC.   **6. Luồng sự kiện thay thế**   | **No** | **Vị trí** | **Điều kiện** | **Hành động** | **Vị trí quay lui** | | --- | --- | --- | --- | --- | |  | Bước 2 | Nếu mã sách không tồn tại trong cơ sở dữ liệu. | Hệ thống thông báo rằng sách không tồn tại và yêu cầu **Admin** nhập lại mã sách. | Bước 1 | |  | Bước 4 | Nếu có thông tin nhập vào không hợp lệ (ví dụ: thiếu tiêu đề sách hoặc tác giả). | Hệ thống yêu cầu **Admin** nhập lại thông tin hợp lệ cho sách. | Bước 3 | | 3 | Bước 4 | Nếu có sự cố với cơ sở dữ liệu (ví dụ: không thể kết nối cơ sở dữ liệu). | Hệ thống thông báo lỗi kết nối và yêu cầu **Admin** thử lại sau. | Kết thúc UC | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |

# CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ BÀI TOÁN

## Thiết kế Cơ sở dữ liệu

### Sơ đồ quan hệ:

A screenshot of a computer

Description automatically generated

## Phân tích cơ sở dữ liệu

1. **Books**

* **Đặc điểm chính:**
  + **book\_id**: Mã định danh duy nhất của sách (khóa chính).
  + **title**: Tiêu đề của sách.
  + **author**: Tên tác giả của sách.
  + **publisher**: Nhà xuất bản của sách.
  + **quantity**: Số lượng sách còn lại trong thư viện.
  + **category\_id**: Thể loại sách, tham chiếu đến thực thể **Category** (khóa ngoại).
* **Vai trò:** Thực thể **Book** lưu trữ thông tin chi tiết của sách trong thư viện, bao gồm tiêu đề, tác giả, nhà xuất bản và số lượng còn lại. Dữ liệu này có mối quan hệ với:
* **categories**: Xác định thể loại của sách (khóa ngoại: ***Category\_id***).
* **borrows**: Theo dõi thông tin mượn trả sách (khóa ngoại: ***book\_id***).

1. **Borrows**

* **Đặc điểm chính:**
* **borrow\_id**: Mã định danh duy nhất của giao dịch mượn sách (khóa chính).
* **reader\_id**: Thông tin người đọc (tham chiếu đến thực thể *Reader*).
* **book\_id**: Thông tin sách được mượn (tham chiếu đến thực thể *Book*).
* **borrow\_date**: Ngày mượn sách.
* **expected\_return\_date**: Ngày dự kiến trả sách.
* **actual\_return\_date**: Ngày thực tế trả sách.
* **Vai trò:** Thực thể **Borrow** lưu trữ thông tin chi tiết về quá trình mượn sách trong thư viện. Dữ liệu này giúp quản lý lịch sử mượn sách, xác định người mượn, sách mượn và thời gian trả sách. **expected\_return\_date** và **actual\_return\_date** hỗ trợ theo dõi việc trả sách đúng hạn hay quá hạn. Dữ liệu này có liên kết với:
* **readers**: Xác định người mượn sách thông qua **Reader** (khóa ngoại: ***reader\_id***).
* **books**: Xác định sách được mượn thông qua **Book** (khóa ngoại: ***book\_id***).

1. **Categories**

* **Đặc điểm chính:**
* **categoryId***: Mã định danh duy nhất của thể loại sách (khóa chính).*
* **categoryName***: Tên của thể loại sách.*
* **Vai trò:** Thực thể **Category** lưu trữ thông tin về thể loại sách trong hệ thống. Dữ liệu từ bảng này giúp tổ chức và phân loại sách theo thể loại cụ thể như **Khoa học**, **Văn học**, **Lịch sử**, v.v., từ đó hỗ trợ người đọc dễ dàng tìm kiếm sách dựa trên thể loại. Nó có mối liên hệ với:
* **books**: Tham chiếu thông qua khóa ngoại ***category\_id***.

1. **Readers**

* **Đặc điểm chính:**
* **reader\_id***: Mã định danh duy nhất của độc giả (khóa chính).*
* **user\_id:** *Mã định danh liên kết với user\_id của Users*
* **full\_name***: Họ và tên đầy đủ của độc giả.*
* **birth\_date***: Ngày sinh của độc giả.*
* **address***: Địa chỉ liên lạc của độc giả.*
* **Vai trò:** Thực thể **Reader** lưu trữ thông tin chi tiết về độc giả trong hệ thống thư viện. Dữ liệu từ thực thể này giúp quản lý thông tin độc giả, bao gồm thông tin cá nhân và địa chỉ liên hệ. Nó đóng vai trò quan trọng trong việc theo dõi lịch sử mượn/trả sách và hỗ trợ quản lý người dùng trong hệ thống thư viện. Dữ liệu này có mối liên hệ với:
* **borrows**: Tham chiếu thông qua khóa ngoại ***reader\_id*** để xác định sách mà độc giả đã mượn.
* **users:** Tham chiếu thông qua khoá ngoại ***user\_id*** để cho phép người dùng xem sách mình đã mượn

1. **Users**

* **Đặc điểm chính:**
* **user\_id***: Mã định danh duy nhất của người dùng (khóa chính).*
* **username***: Tên đăng nhập của người dùng (dữ liệu unique, không trùng lặp).*
* **password***: Mật khẩu đăng nhập của người dùng (chưa được mã hóa trong đoạn code hiện tại).*
* **full\_name***: Họ và tên đầy đủ của người dùng.*
* **role***: Vai trò của người dùng trong hệ thống (ENUM gồm Admin và User).*
* **created\_at***: Thời gian tạo tài khoản của người dùng.*
* **Vai trò:** Thực thể **User** dùng để lưu trữ thông tin người dùng trong hệ thống. Nó phân loại người dùng thành ADMIN hoặc USER thông qua enum **UserRole**. Thực thể này có thể được sử dụng trong các hệ thống xác thực (authentication) và phân quyền (authorization).

## 2.2. Biểu đồ lớp

## 

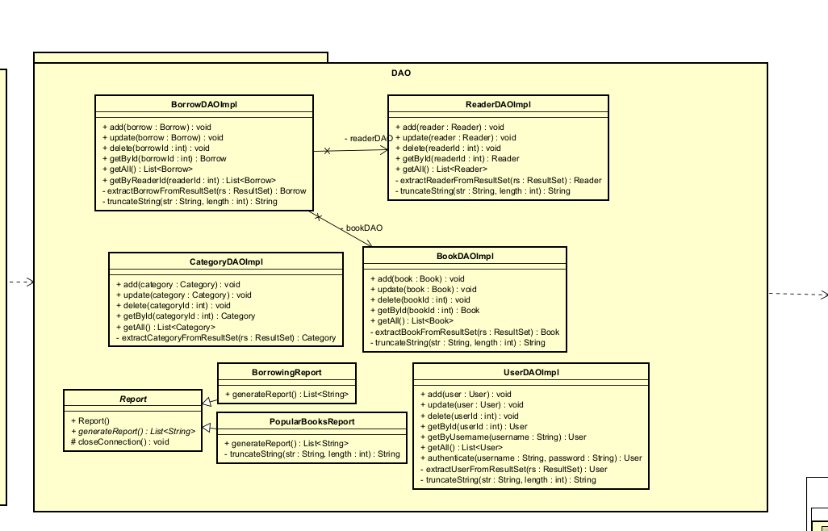
### 2.2.1. Service Layer:

A screenshot of a service

Description automatically generated

### 

### 2.2.2. DAO Layer:



**2.2.3. Entity Layer:**

A screenshot of a computer program

Description automatically generated

# CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MINH HOẠ

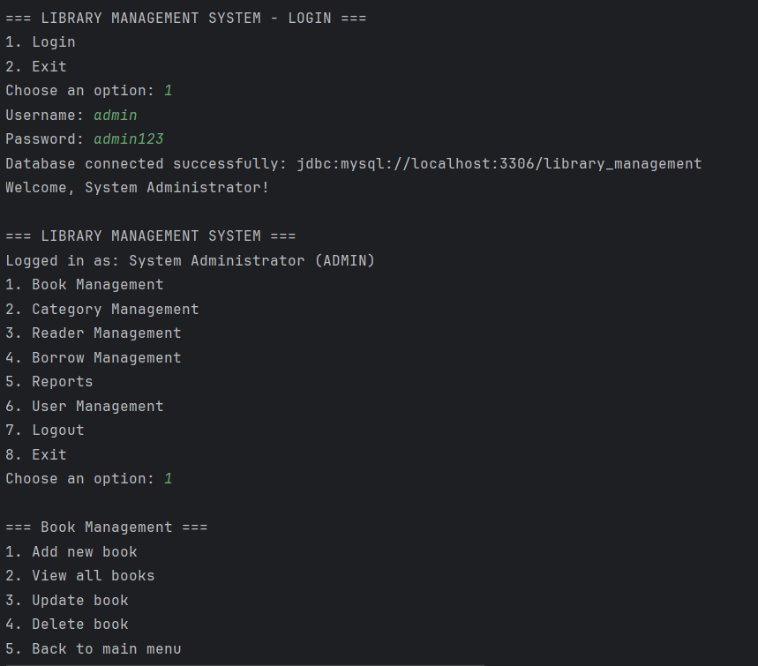
## 3.1. Kết quả chương trình minh hoạ

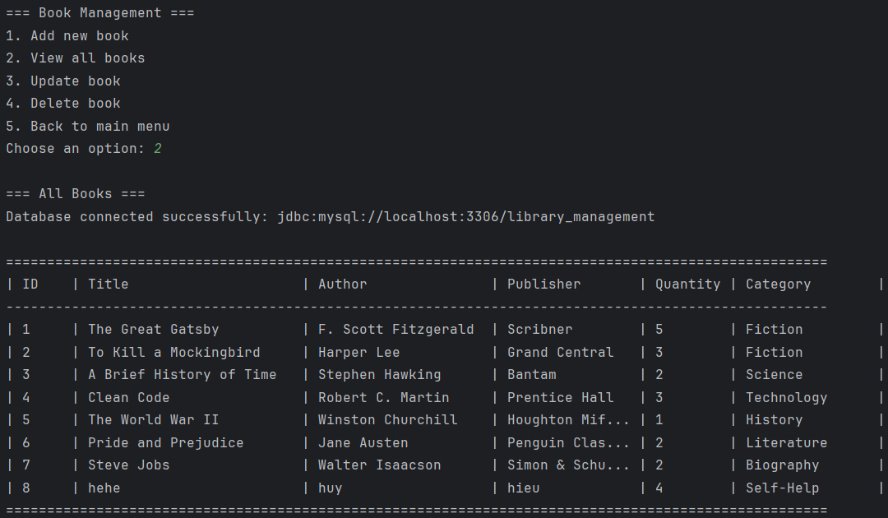
Chương trình được đặt tên là LibMana, chạy trên console log của IDEA Intellij, phục vụ việc quản lý sinh viên 1 trường học.

Chương trình được xây dựng đáp ứng đầy đủ tất cả các yêu cầu ban đầu của đề tài về chức năng

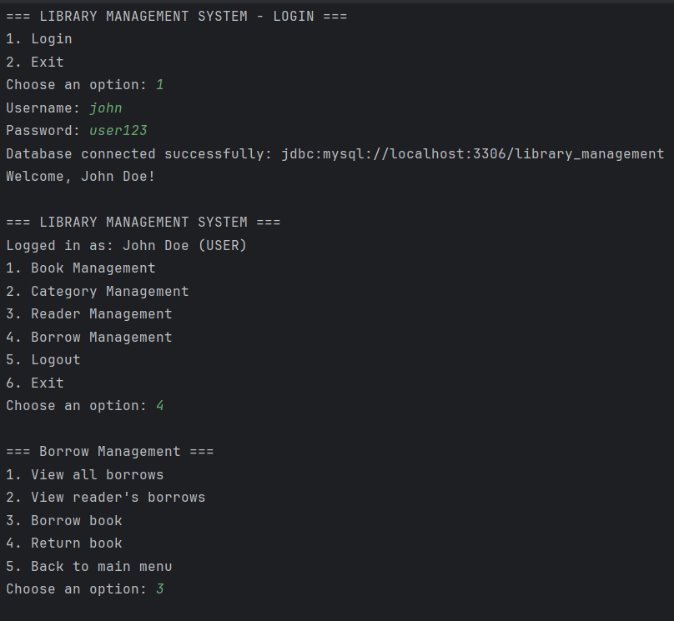
## 3.2. Giao diện chương trình

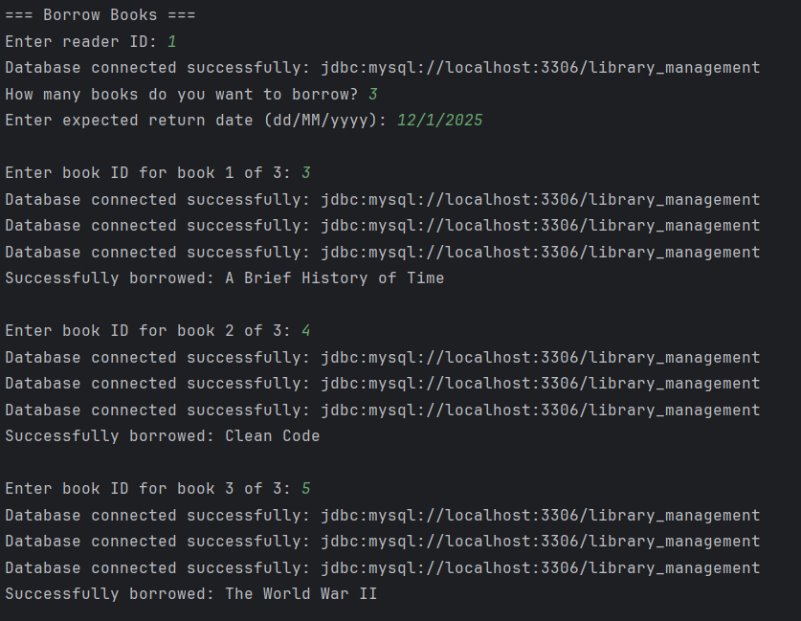
### 3.2.1. Giao diện admin - quản lý sách:





### 3.2.2. Giao diện mượn sách - User:





**CHƯƠNG 4: PHÂN TÍCH ĐẶC TÍNH HƯỚNG ĐỐI TƯỢNG**

**1. Tính đóng gói (Encapsulation)**

Nhận dạng:

* Entity Classes: Book, Reader, User, Category, Borrow đều thực hiện encapsulation
* Service Layer: Các ServiceImpl classes đóng gói business logic
* DAO Layer: Các DAOImpl classes đóng gói data access logic

Phân tích:

Tất cả các entity classes đều ẩn các thuộc tính và chỉ cho phép truy cập thông qua các methods.

* Book: Các thuộc tính bookId, title, author, quantity được khai báo là private. Truy cập thông qua các phương thức getBookId(), setTitle().
* Reader: Các thuộc tính như readerId, fullName, birthDate cũng được khai báo là private và truy cập thông qua các getter/setter.

Ví dụ:

| public class Book {  private int bookId;  private String title;  // ... other private fields  public int getBookId() {  return bookId;  }  public void setBookId(int bookId) {  this.bookId = bookId;  }  } |
| --- |

Lợi ích:

1. Bảo vệ dữ liệu: Ngăn truy cập trực tiếp vào các thuộc tính của lớp từ bên ngoài, đảm bảo tính toàn vẹn của dữ liệu.
2. Kiểm soát truy cập: Dữ liệu chỉ được thay đổi thông qua các phương thức được định nghĩa.
3. Giảm sự phụ thuộc: Các lớp khác không cần biết chi tiết dữ liệu bên trong lớp, chỉ cần sử dụng các phương thức.
4. Dễ bảo trì: Khi thay đổi cấu trúc dữ liệu, chỉ cần sửa trong lớp đó mà không ảnh hưởng đến các lớp khác.

**2. Tính kế thừa (Inheritance)**

Nhận dạng:

* Report Hierarchy: Abstract Report class với các concrete implementations
* Entity Relationships: Thể hiện quan hệ IS-A giữa các entities

Ví dụ:

| public abstract class Report {  protected Connection connection;    public abstract List<String> generateReport();  }  public class BorrowingReport extends Report {  @Override  public List<String> generateReport() {  // Specific implementation for borrowing report  }  }  public class PopularBooksReport extends Report {  @Override  public List<String> generateReport() {  // Specific implementation for popular books report  }  } |
| --- |

Lợi ích:

1. Tái sử dụng code: Giảm lặp lại code bằng cách tái sử dụng các thuộc tính và phương thức của lớp cha.
2. Mở rộng dễ dàng: Thêm các loại người dùng mới chỉ bằng cách kế thừa từ lớp User.

**3. Tính đa hình (Polymorphism)**

Thể hiện:

* Method Overriding: Trong Report classes
* Runtime Polymorphism: Xử lý các loại Report khác nhau

Phân tích:

* BorrowServiceImpl cung cấp các triển khai cụ thể cho các phương thức.

Ví dụ:

| private static void handleReportsMenu() {  // Different report types can be handled polymorphically  Report report;  switch(choice) {  case 1:  report = new BorrowingReport();  break;  case 2:  report = new PopularBooksReport();  break;  }  List<String> reportLines = report.generateReport(); // polymorphic call  } |
| --- |

Lợi ích:

1. Trừu tượng hóa: Ẩn chi tiết triển khai, chỉ tập trung vào hành động cần làm.
2. Tạo sự thống nhất: Cung cấp một giao diện chung cho các chức năng mượn và trả sách.
3. Dễ mở rộng: Khi cần thêm logic mới, chỉ cần tạo một lớp mới và triển khai interface.

**4. Tính trừu tượng (Abstraction)**

Thể hiện:

* Service Layer Abstraction: Ẩn chi tiết implementation của DAO
* Report Abstraction: Abstract Report class định nghĩa contract
* Data Access Abstraction: DAO pattern ẩn chi tiết database access

Ví dụ:

| public class BorrowServiceImpl {  private final BorrowDAOImpl borrowDAO;  public void borrowBook(Borrow borrow) {  validateBorrow(borrow); // Abstract validation  borrowDAO.add(borrow); // Abstract data access  }  } |
| --- |

Lợi ích:

1. Ẩn chi tiết database operations
2. Dễ dàng thay đổi database schema
3. Centralized data access logic

# TÀI LIỆU THAM KHẢO

1. Bài giảng học phần Lập trình hướng đối tượng của cô Đỗ Thị Ngọc Diệp.
2. Lập trình hướng đối tượng với Java – Đoàn Văn Ban – NXB Khoa học và kỹ thuật.
3. Phân tích và thiết kế hệ thống thông tin - Nguyễn Văn Ba.
4. Android cơ bản và nâng cao - vietjack.com
5. Các video, tài liệu về lập trình hướng đối tượng và lập trình android trên Youtube.