 **TRƯỜNG ĐẠI HỌC ĐÀ LẠT**

**KHOA CÔNG NGHỆ THÔNG TIN**

**--🙞🕮🙜--**



**BÁO CÁO MẪU THIẾT KẾ**

**ĐỀ TÀI: ỨNG DỤNG QUẢN LÝ THƯ VIỆN SÁCH BOOKVERSE  
(Phát triển bằng HTML, CSS, JavaScript, Vite theo mô hình MVC và các mẫu thiết kế phần mềm Singleton, Factory, Observer)**

**Giảng viên hướng dẫn:** ThS. Đoàn Minh Khuê

**Sinh viên thực hiện:** 2212494 - Đặng Hoài Vũ

**Đà Lạt, tháng 10 năm 2025**

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

Đà Lạt, ngày …… tháng …… năm ……

**Giảng viên hướng dẫn**

(Ký và ghi rõ họ tên)

**LỜI MỞ ĐẦU**

Trong thời đại công nghệ số phát triển mạnh mẽ, việc quản lý và truy cập thông tin ngày càng trở nên quan trọng, đặc biệt trong lĩnh vực giáo dục và đọc sách. Sách là nguồn tri thức quý giá, nhưng cách quản lý và lưu trữ truyền thống còn nhiều hạn chế như tốn thời gian, khó tìm kiếm và thiếu tính tương tác. Chính vì vậy, em lựa chọn thực hiện đề tài “Ứng dụng quản lý thư viện sách BookVerse” nhằm xây dựng một giải pháp số hóa quá trình quản lý và tra cứu sách một cách tiện lợi, hiện đại và thân thiện với người dùng.

Dự án BookVerse được phát triển với công nghệ HTML5, CSS3 và JavaScript (ES6), kết hợp mô hình MVC và các mẫu thiết kế phần mềm như Singleton, Factory, Observer, giúp hệ thống dễ mở rộng, bảo trì và tái sử dụng. Ứng dụng hỗ trợ các chức năng cơ bản như thêm, sửa, xóa, tìm kiếm, lọc và thống kê sách; đồng thời cung cấp giao diện glassmorphism hiện đại, hiệu ứng chuyển đổi Dark/Light Mode, và khả năng lưu trữ dữ liệu cục bộ (LocalStorage) mà không cần máy chủ.

Thông qua dự án này, em mong muốn củng cố kiến thức về lập trình web, rèn luyện kỹ năng thiết kế giao diện người dùng, đồng thời học cách áp dụng các nguyên lý thiết kế phần mềm vào thực tế. BookVerse không chỉ là sản phẩm học tập, mà còn là bước khởi đầu cho việc xây dựng các ứng dụng web có tính ứng dụng cao và trải nghiệm người dùng tốt hơn trong tương lai.

MỤC LỤC

[PHẦN 1. GIỚI THIỆU DỰ ÁN 2](#_Toc212221488)

[1.1 Tên dự án và mục tiêu 2](#_Toc212221489)

[1.2 Vấn đề thực tế giải quyết 2](#_Toc212221490)

[1.3 Đối tượng người dùng 2](#_Toc212221491)

[PHẦN 2. PHÂN TÍCH HỆ THỐNG 3](#_Toc212221492)

[2.1 Yêu cầu chức năng 3](#_Toc212221493)

[2.2 Yêu cầu phi chức năng 4](#_Toc212221494)

[2.3 Mô tả nghiệp vụ 4](#_Toc212221495)

[2.4 Mô hình dữ liệu 6](#_Toc212221496)

[2.5 Giới hạn và phạm vi 7](#_Toc212221497)

[PHẦN 3. THIẾT KẾ VÀ KIẾN TRÚC PHẦN MỀM 8](#_Toc212221498)

[3.1 Cấu trúc thư mục 8](#_Toc212221499)

[3.2 Design Patterns được áp dụng 8](#_Toc212221500)

[3.2.1 MVC Pattern: Đây là kiến trúc chủ đạo của dự án, giúp tách biệt rõ ràng 3 thành phần: 9](#_Toc212221501)

[3.2.2 Singleton Pattern: Em dùng pattern này để đảm bảo chỉ có một instance (thể hiện) duy nhất của các lớp quản lý tài nguyên chung: 9](#_Toc212221502)

[3.2.3 Factory Pattern: 9](#_Toc212221503)

[3.2.4 Observer Pattern: 9](#_Toc212221504)

[3.3 Luồng hoạt động Model-View-Controller 10](#_Toc212221505)

[3.4 Cách dữ liệu di chuyển và lưu trữ 10](#_Toc212221506)

[PHẦN 4. CHỨC NĂNG CHI TIẾT VÀ GIAO DIỆN 12](#_Toc212221507)

[4.1 Mô tả cụ thể các tính năng 12](#_Toc212221508)

[4.2 Mô tả các component UI 16](#_Toc212221509)

[4.3 Giao diện tổng thể 17](#_Toc212221510)

[4.3.1 Header: 17](#_Toc212221511)

[4.3.2 Main Content: Được chia làm 3 khu vực: 17](#_Toc212221512)

[4.3.3 Modal Overlay: 18](#_Toc212221513)

[4.4 Phong cách thiết kế 18](#_Toc212221514)

[4.5 Flow người dùng 19](#_Toc212221515)

[PHẦN 5. CÔNG NGHỆ SỬ DỤNG 20](#_Toc212221516)

[5.1 Ngôn ngữ và framework 20](#_Toc212221517)

[5.2 Thư viện và công cụ 20](#_Toc212221518)

[5.3 Cấu hình và chạy dự án 21](#_Toc212221519)

[5.4 Cơ chế build và deploy 21](#_Toc212221520)

[PHẦN 6. KIỂM THỬ VÀ KẾT QUẢ 22](#_Toc212221521)

[6.1 Kiểm thử các chức năng chính 22](#_Toc212221522)

[6.2 Ảnh hưởng lên LocalStorage 23](#_Toc212221523)

[6.3 Hiệu năng và tương thích 23](#_Toc212221524)

[PHẦN 7. ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN 24](#_Toc212221525)

[7.1 Ưu điểm 24](#_Toc212221526)

[7.2 Hạn chế 25](#_Toc212221527)

[7.3 Hướng phát triển tương lai 25](#_Toc212221528)

[PHẦN 8. KẾT LUẬN 27](#_Toc212221529)

[8.1 Giá trị kỹ thuật 27](#_Toc212221530)

[8.2 Giá trị học thuật 27](#_Toc212221531)

[8.3 Ý nghĩa đối với sinh viên (bản thân em) 28](#_Toc212221532)

[8.4 Kết luận cuối cùng 28](#_Toc212221533)

[TÀI LIỆU THAM KHẢO 29](#_Toc212221534)

# 

# PHẦN 1. GIỚI THIỆU DỰ ÁN

## **Tên dự án và mục tiêu**

Dự án của em có tên là **BookVerse**. Đây là một ứng dụng web do em phát triển với mục tiêu giúp người dùng quản lý thư viện sách cá nhân của họ.

Về mặt kỹ thuật, dự án được xây dựng bằng Vanilla JavaScript ES6+ và áp dụng mô hình kiến trúc MVC. Mục tiêu cốt lõi của dự án là giải quyết nhu cầu quản lý bộ sưu tập sách cá nhân một cách hiệu quả, đồng thời cung cấp một giao diện trực quan cho người dùng.

## **Vấn đề thực tế giải quyết**

Trong thực tế, cá nhân em nhận thấy rằng trong thời đại số hóa, việc quản lý sách cá nhân ngày càng trở nên quan trọng. Người dùng hiện nay cần một công cụ chuyên dụng để:

* Tổ chức và quản lý bộ sưu tập sách của riêng họ.
* Tìm kiếm và lọc sách một cách linh hoạt theo nhiều tiêu chí.
* Xem thống kê về bộ sưu tập, ví dụ như tổng số sách và sự phân bố của các thể loại.
* Lưu trữ dữ liệu một cách bền bỉ giữa các phiên làm việc.
* Trải nghiệm một giao diện đẹp mắt và dễ sử dụng.

## **1.3 Đối tượng người dùng**

Ứng dụng BookVerse mà em xây dựng hướng đến nhiều nhóm đối tượng người dùng cụ thể, bao gồm:

* **Sinh viên**: Dùng để quản lý sách giáo khoa và các tài liệu học tập.
* **Người đọc (độc giả nói chung)**: Dùng để quản lý thư viện sách cá nhân và theo dõi tiến độ đọc.
* **Thư viện cá nhân**: Hỗ trợ quản lý bộ sưu tập sách tại nhà.

# PHẦN 2. PHÂN TÍCH HỆ THỐNG

## **Yêu cầu chức năng**

Để đáp ứng các vấn đề đã nêu ở Phần 1, em đã xác định các yêu cầu chức năng chính cho ứng dụng BookVerse như sau:

* **Quản lý sách (CRUD Operations)**:
  + Cho phép thêm sách mới với các thông tin: tiêu đề, tác giả, thể loại, ảnh bìa, và mô tả.
  + Cho phép chỉnh sửa thông tin của sách hiện có.
  + Cho phép xóa sách khỏi thư viện.
  + Hiển thị danh sách tất cả các cuốn sách.
* **Tìm kiếm và lọc**:
  + Người dùng có thể tìm kiếm sách theo tên, tác giả, thể loại, hoặc mô tả.
  + Người dùng có thể lọc sách theo thể loại.
  + Chức năng tìm kiếm phải là real-time và có áp dụng kỹ thuật debouncing.
* **Thống kê và phân tích**:
  + Hiển thị tổng số sách hiện có trong thư viện.
  + Thống kê tổng số lượng thể loại sách.
  + Xác định và hiển thị thể loại phổ biến nhất.
  + Sử dụng biểu đồ tròn để hiển thị phân bố các thể loại sách.
* **Quản lý dữ liệu**:
  + Cung cấp chức năng xuất (export) dữ liệu thư viện ra file JSON.
  + Cung cấp chức năng nhập (import) dữ liệu từ file JSON.
  + Tự động tải dữ liệu mẫu (demo data) khi thư viện trống.

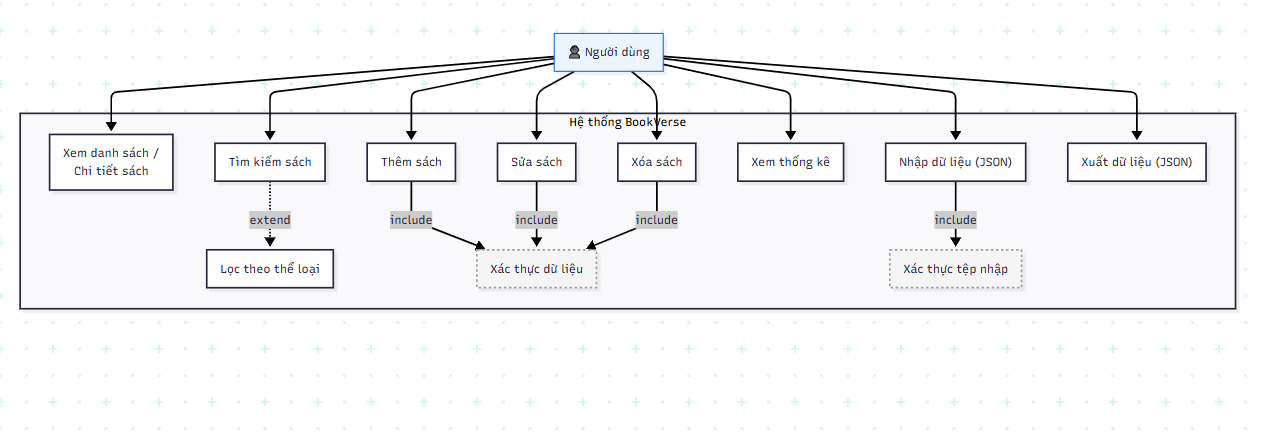
## **Yêu cầu phi chức năng**

Bên cạnh các yêu cầu về chức năng, dự án cũng cần đáp ứng các yêu cầu phi chức năng sau:

* **Hiệu suất**: Ứng dụng phải phản hồi nhanh. Chức năng tìm kiếm phải được tối ưu với debouncing 300ms để tránh xử lý thừa.
* **Khả năng sử dụng**: Giao diện phải được thiết kế trực quan, dễ sử dụng. Ứng dụng phải hỗ trợ điều hướng bằng bàn phím (keyboard navigation).
* **Khả năng mở rộng**: Mã nguồn (code) phải được tổ chức theo các design patterns để dễ bảo trì và mở rộng trong tương lai.
* **Tương thích**: Ứng dụng phải hoạt động ổn định trên các trình duyệt web hiện đại.
* **Responsive**: Giao diện phải có khả năng tương thích tốt trên nhiều thiết bị như mobile, tablet, và desktop.

## **Mô tả nghiệp vụ**

Dựa trên các yêu cầu, em mô tả các nghiệp vụ chính và sơ đồ hoạt động của hệ thống:

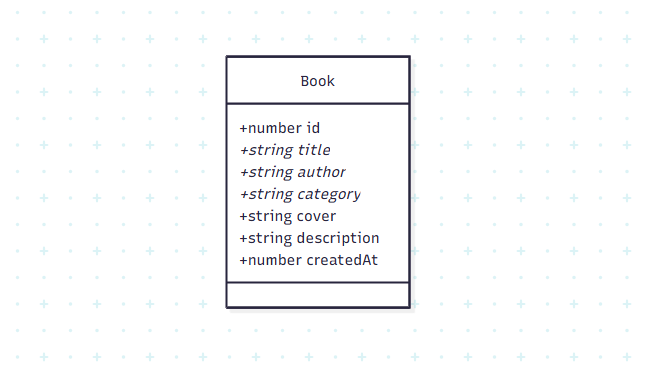


Sơ đồ Use Case trên mô tả các nghiệp vụ chính của hệ thống:

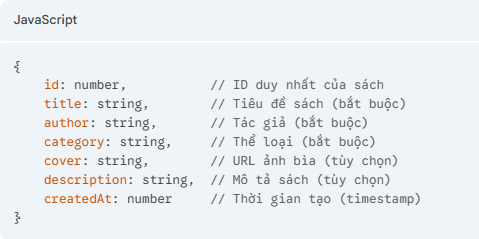
* **Tác nhân (Actor)**: Người dùng là tác nhân duy nhất tương tác với hệ thống.
* **Các Use Case chính**: Người dùng có thể trực tiếp thực hiện các nghiệp vụ:
  + Xem danh sách / Chi tiết sách
  + Tìm kiếm sách
  + Thêm sách
  + Sửa sách
  + Xóa sách
  + Xem thống kê
  + Nhập dữ liệu (JSON)
  + Xuất dữ liệu (JSON)
* **Mối quan hệ «extend»**:
  + Chức năng Tìm kiếm sách có thể được mở rộng (extend) với chức năng Lọc theo thể loại.
* **Mối quan hệ «include»**:
  + Các nghiệp vụ Thêm sách, Sửa sách, và Xóa sách đều bắt buộc bao gồm (include) quy trình Xác thực dữ liệu.
  + Nghiệp vụ Nhập dữ liệu (JSON) bao gồm (include) quy trình Xác thực tệp nhập.
* **Sơ đồ hoạt động (Luồng tổng quan)**: Người dùng mở ứng dụng → Hệ thống tải dữ liệu từ LocalStorage → Hệ thống hiển thị danh sách sách và các số liệu thống kê → Người dùng thực hiện thao tác (thêm/sửa/xóa/tìm kiếm) → Hệ thống cập nhật dữ liệu → Hệ thống lưu dữ liệu mới vào LocalStorage → Hệ thống cập nhật lại giao diện

## **Mô hình dữ liệu**

Để phục vụ cho các nghiệp vụ trên, em thiết kế mô hình dữ liệu như sau:



* **Cấu trúc đối tượng Book**: Mỗi cuốn sách sẽ là một đối tượng JavaScript với cấu trúc:



* **Kiểu dữ liệu lưu trữ**:
  + **LocalStorage**: Được sử dụng để lưu trữ dữ liệu sách và cài đặt theme của người dùng.
  + **JSON Format**: Là định dạng dữ liệu được sử dụng cho chức năng import và export.
  + **Session Data**: Là dữ liệu tạm thời được xử lý trong phiên làm việc của ứng dụng.

## **Giới hạn và phạm vi**

Trong khuôn khổ đồ án môn học này, em xác định rõ phạm vi và giới hạn của dự án:

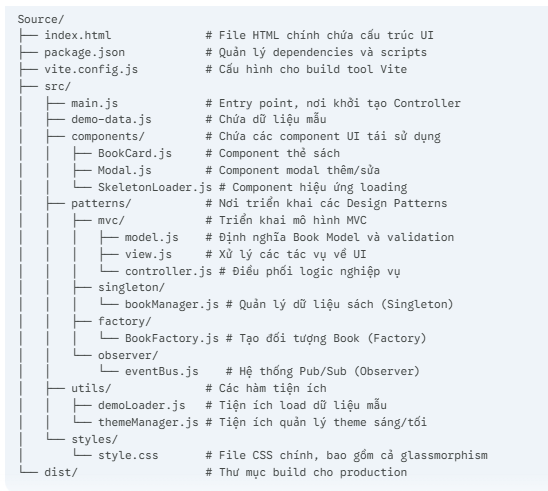
* **Phạm vi trong (In-scope)**:
  + Quản lý sách cá nhân (CRUD).
  + Tìm kiếm và lọc sách.
  + Thống kê cơ bản.
  + Import/Export dữ liệu định dạng JSON.
  + Giao diện responsive.
* **Phạm vi ngoài (Out-of-scope)**:
  + Dự án không có hệ thống đăng nhập hay xác thực người dùng.
  + Không có backend server (máy chủ).
  + Không sử dụng cơ sở dữ liệu thực sự (như SQL hay NoSQL), chỉ dùng LocalStorage.
  + Không có các tính năng nâng cao như đánh giá sách (rating).
  + Không hỗ trợ đồng bộ dữ liệu qua đám mây (cloud sync).

# PHẦN 3. THIẾT KẾ VÀ KIẾN TRÚC PHẦN MỀM

Để đảm bảo code dễ bảo trì và mở rộng, em đã tổ chức dự án và áp dụng các mẫu thiết kế (design patterns) một cách có hệ thống.

## **Cấu trúc thư mục**

Em tổ chức cấu trúc thư mục dự án theo hướng module hóa, tách biệt rõ ràng các thành phần:



## **Design Patterns được áp dụng**

Trong dự án này, em đã áp dụng 4 design patterns chính:

### **MVC Pattern**: Đây là kiến trúc chủ đạo của dự án, giúp tách biệt rõ ràng 3 thành phần:

* + **Model (Book)**: Định nghĩa cấu trúc dữ liệu của cuốn sách, bao gồm cả các quy tắc validation và serialization (chuyển đổi dữ liệu).
  + **View**: Chịu trách nhiệm hiển thị giao diện người dùng, lắng nghe các sự kiện (event listeners), và tích hợp với Chart.js để vẽ biểu đồ.
  + **Controller**: Đóng vai trò trung gian, điều phối hoạt động giữa Model và View, và xử lý toàn bộ logic nghiệp vụ (business logic).

### **Singleton Pattern**: Em dùng pattern này để đảm bảo chỉ có một instance (thể hiện) duy nhất của các lớp quản lý tài nguyên chung:

* + **BookManager**: Đảm bảo chỉ có một đối tượng duy nhất quản lý toàn bộ dữ liệu sách.
  + **EventBus**: Là một singleton cho hệ thống publish/subscribe (Observer).

### **Factory Pattern:**

* + Em sử dụng **BookFactory** để tạo ra các đối tượng Book. Pattern này giúp đóng gói logic khởi tạo, và cho phép mở rộng để tạo ra nhiều loại sách khác nhau trong tương lai (ví dụ: comic, novel, textbook) một cách dễ dàng.

### **Observer Pattern:**

* + Em triển khai pattern này thông qua **EventBus**, một hệ thống publish/subscribe (pub/sub).
  + Nó cho phép các component giao tiếp với nhau một cách lỏng lẻo (loosely coupled). Khi có một sự kiện xảy ra (ví dụ: BOOKS\_CHANGED, BOOK\_ADDED), EventBus sẽ thông báo cho tất cả các component đã đăng ký lắng nghe sự kiện đó.

## **Luồng hoạt động Model-View-Controller**

Luồng hoạt động chính của ứng dụng khi người dùng tương tác tuân thủ nghiêm ngặt theo mô hình MVC như sau:

1. **User Interaction**: Người dùng tương tác với **View** (ví dụ: click nút "Thêm sách", gõ vào ô tìm kiếm).
2. **View → Controller**: **View** không tự xử lý mà phát ra (emit) một sự kiện thông qua **EventBus** (ví dụ: BOOK\_ADD\_REQUESTED).
3. **Controller Processing**: **Controller** lắng nghe sự kiện này và bắt đầu xử lý business logic (ví dụ: lấy dữ liệu từ form).
4. **Controller → Model**: **Controller** gọi đến **BookManager** (Model) để thao tác dữ liệu (ví dụ: thêm sách mới).
5. **Model Update**: **BookManager** cập nhật dữ liệu và sau đó phát ra một sự kiện (ví dụ: BOOKS\_CHANGED) qua EventBus để thông báo rằng dữ liệu đã thay đổi.
6. **Controller → View**: **Controller** (hoặc chính **View**) lắng nghe sự kiện BOOKS\_CHANGED và yêu cầu **View** cập nhật lại giao diện.
7. **View Rendering**: **View** render lại giao diện (ví dụ: hiển thị thêm sách mới, cập nhật biểu đồ thống kê) với dữ liệu mới nhất.

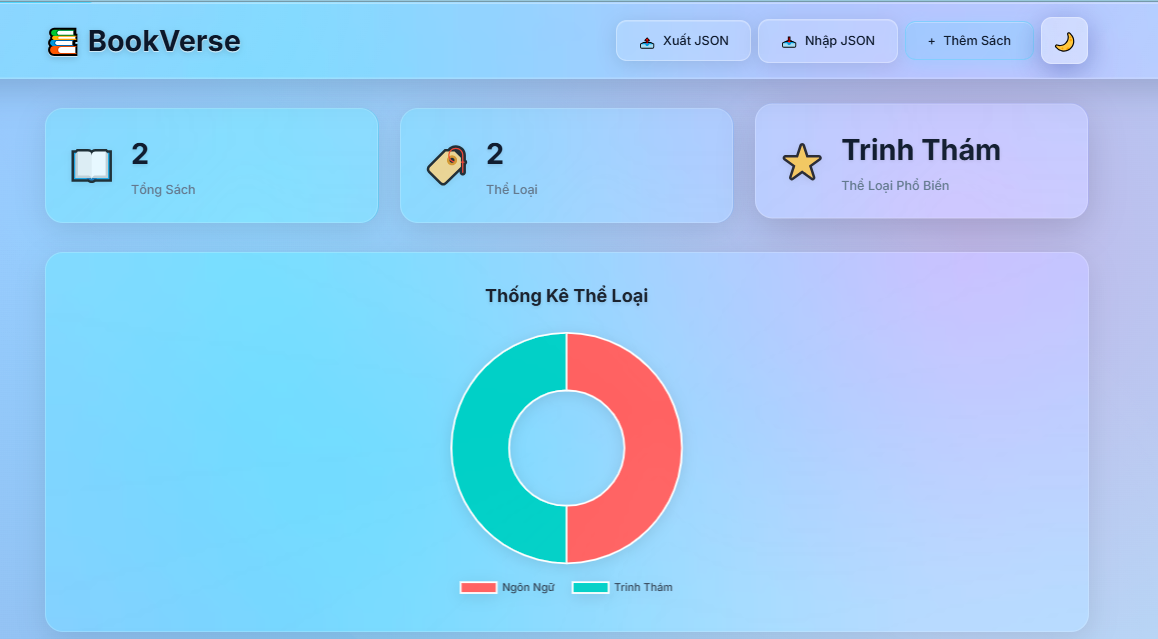
## **Cách dữ liệu di chuyển và lưu trữ**

* **LocalStorage**: Đây là nơi lưu trữ chính của ứng dụng.
  + Key bookverse\_books: Dùng để lưu trữ một mảng JSON chứa tất cả các đối tượng sách.
  + Key theme: Dùng để lưu cài đặt theme (light/dark) của người dùng.
* **JSON Format**: Dữ liệu sách khi import/export sẽ có dạng một mảng các đối tượng sách theo đúng cấu trúc đã định nghĩa ở Phần 2.
* **Import/Export**:
  + **Export**: Luồng xử lý là BookManager.getAll() → JSON.stringify() → tạo file và cho phép người dùng download.
  + **Import**: Luồng xử lý là Người dùng upload file → JSON.parse() → BookFactory.createMultiple() (để validate) → BookManager.importBooks().

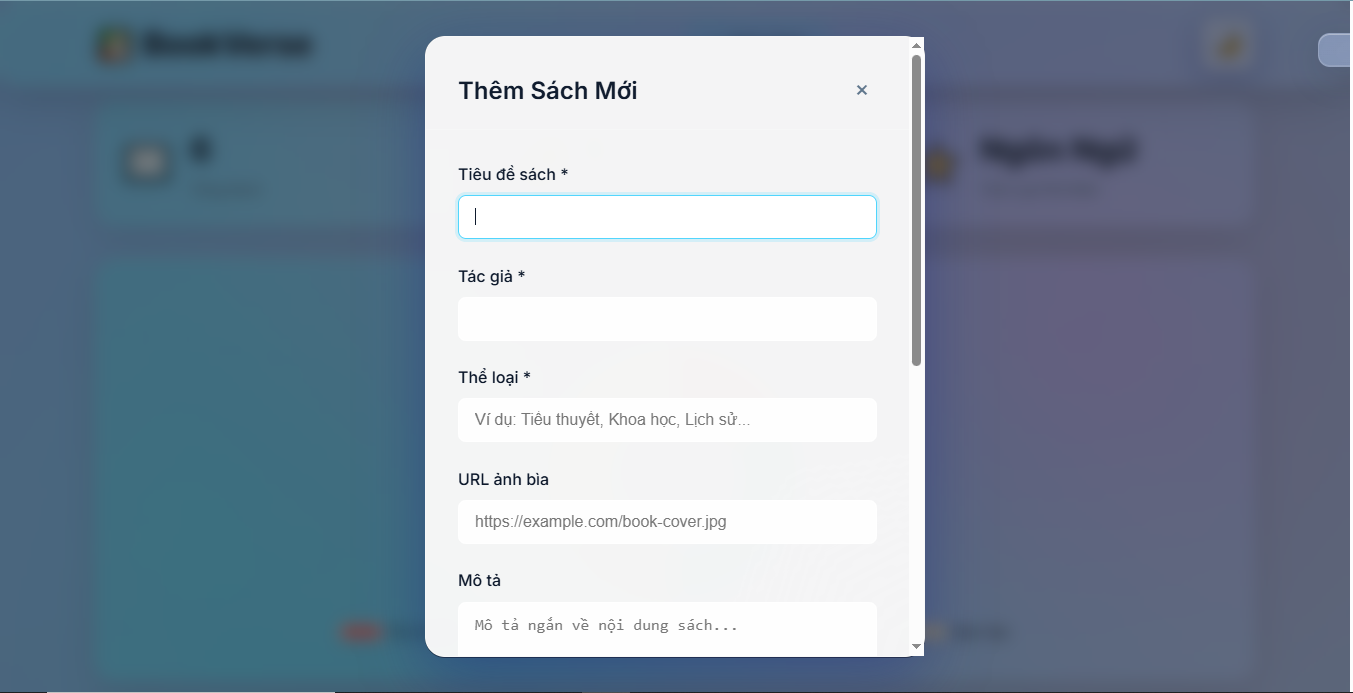
# PHẦN 4. CHỨC NĂNG CHI TIẾT VÀ GIAO DIỆN

## **Mô tả cụ thể các tính năng**

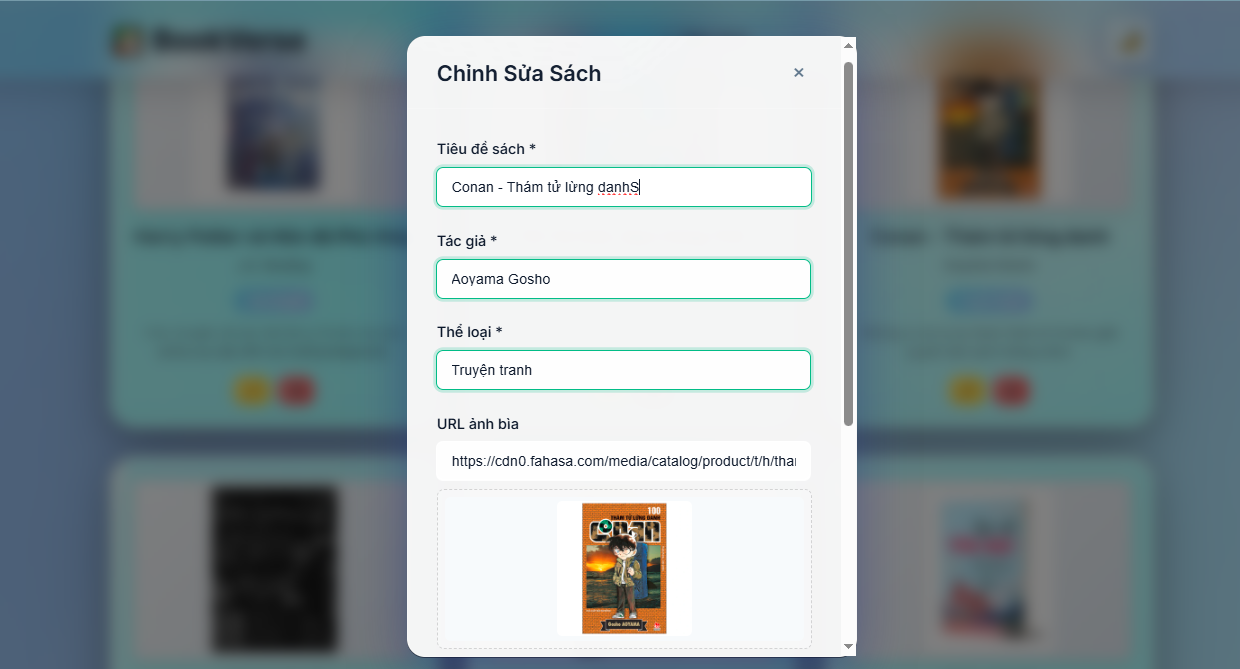
Em đã hiện thực hóa các yêu cầu chức năng đã phân tích ở Phần 2 thành các tính năng cụ thể như sau:



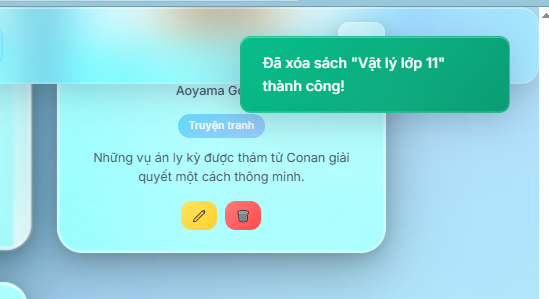
* **Thêm sách:**



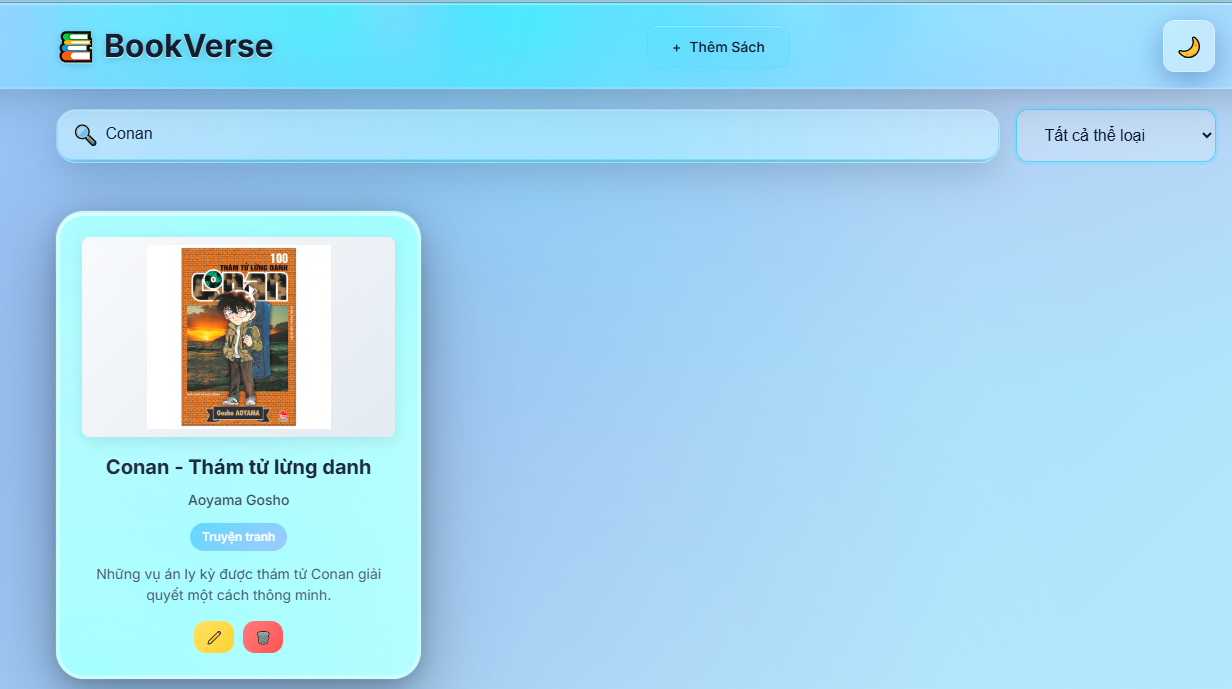
* + Sử dụng một Modal popup (cửa sổ bật lên) để hiển thị form nhập liệu.
  + Form có validation (kiểm tra hợp lệ) real-time cho các trường bắt buộc như tiêu đề, tác giả, thể loại.
  + Có chức năng preview ảnh bìa ngay khi người dùng dán URL vào.
  + Modal cũng được trang bị các tính năng hỗ trợ tiếp cận (accessibility) như focus trap, giúp người dùng điều khiển bằng bàn phím không bị "lạc" ra ngoài modal.
* **Sửa sách:**



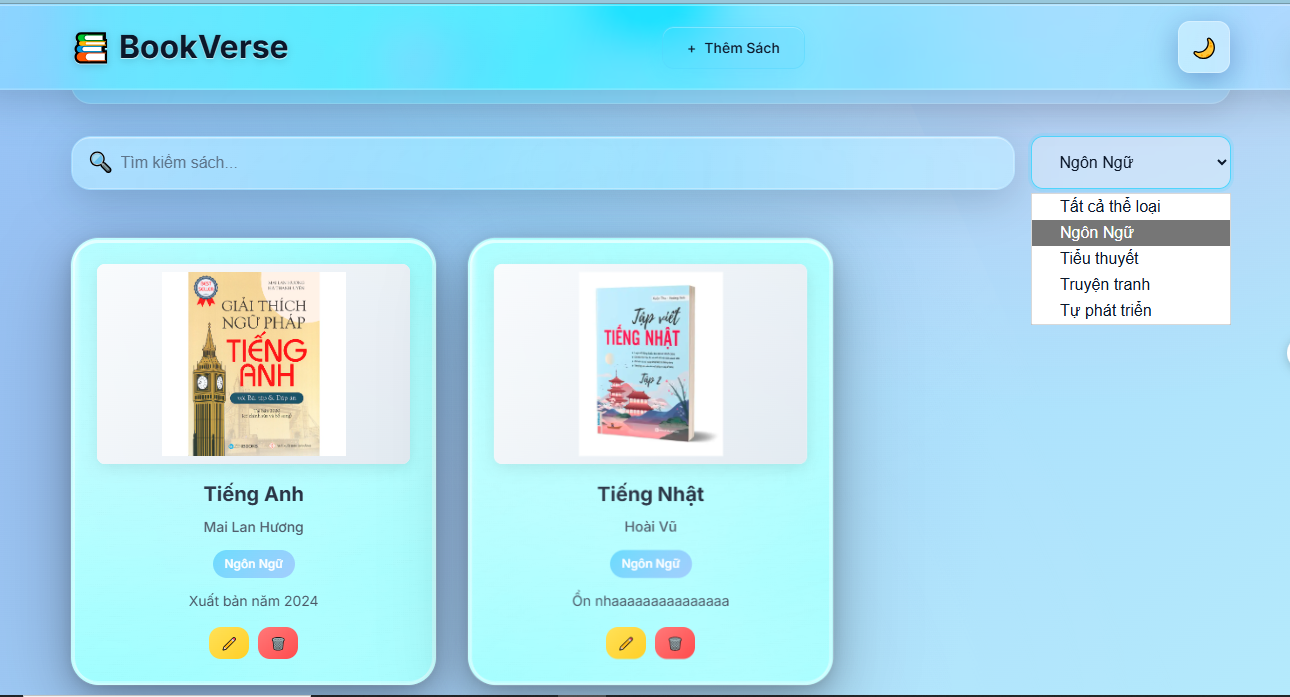
* + Người dùng nhấn nút "Chỉnh sửa" trên mỗi thẻ sách (book card).
  + Modal sẽ mở ra và tự động điền thông tin hiện có của cuốn sách đó vào form.
  + Các tính năng validation và preview ảnh bìa hoạt động tương tự như khi thêm sách.
* **Xóa sách:**



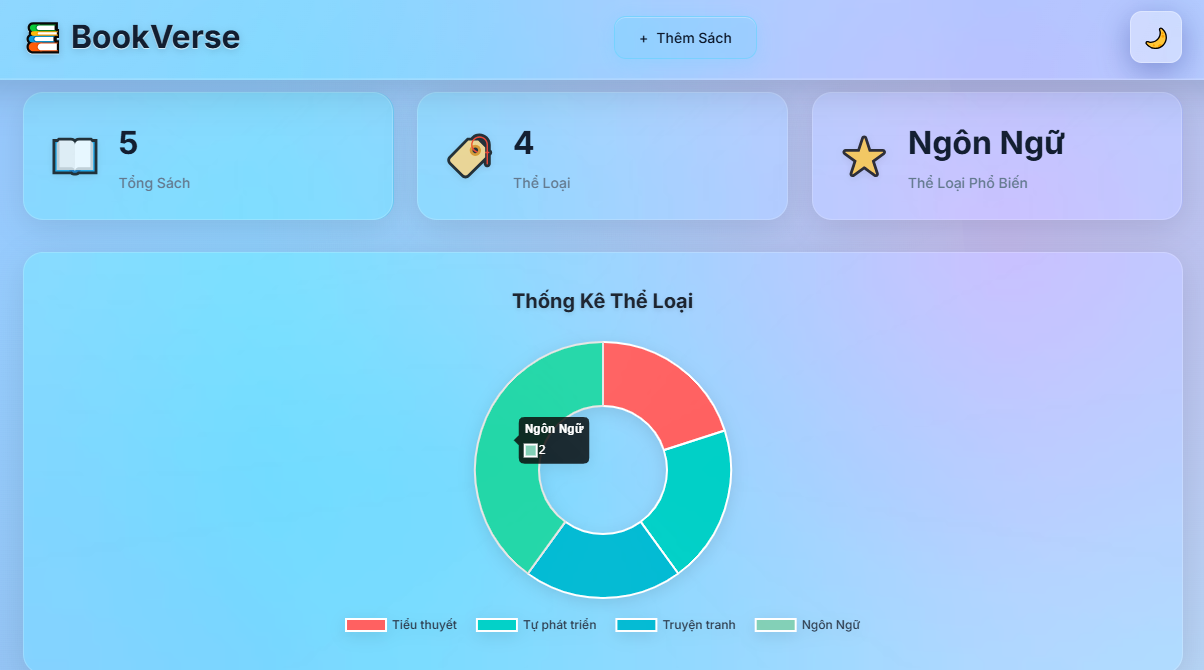
* + Người dùng nhấn nút "Xóa" trên thẻ sách.
  + Một hộp thoại xác nhận (confirmation dialog) sẽ xuất hiện để yêu cầu người dùng xác nhận, tránh xóa nhầm.
  + Sau khi xác nhận, giao diện và LocalStorage được cập nhật ngay lập tức.
* **Tìm kiếm:**



* + Em sử dụng một ô input để người dùng nhập từ khóa.
  + Tính năng này được tối ưu bằng kỹ thuật **debouncing 300ms**, nghĩa là hệ thống chỉ bắt đầu tìm kiếm sau khi người dùng ngừng gõ 300ms, giúp giảm tải xử lý.
  + Việc tìm kiếm sẽ dựa trên các trường: title (tiêu đề), author (tác giả), category (thể loại), và description (mô tả).
* **Lọc theo thể loại:**



* + Em sử dụng một dropdown (menu thả xuống) để hiển thị danh sách các thể loại sách.
  + Danh sách này được tạo tự động (dynamic) dựa trên dữ liệu sách hiện có.
* **Thống kê:**



* + Các số liệu thống kê được tính toán từ BookManager: Tổng số sách, Số thể loại, và Thể loại phổ biến nhất.
  + Em tích hợp thư viện **Chart.js** để vẽ biểu đồ tròn (doughnut chart), trực quan hóa sự phân bố của các thể loại sách.
  + Em cũng code thêm một fallback display (hiển thị dự phòng) trong trường hợp Chart.js không tải được, đảm bảo giao diện không bị vỡ.

## **Mô tả các component UI**

Để dễ dàng tái sử dụng và quản lý, em đã chia giao diện thành các component chính:

* BookCard Component:
  + Là component hiển thị thông tin tóm tắt của một cuốn sách (ảnh bìa, tiêu đề, tác giả, thể loại).
  + Chứa 2 nút chức năng là "Chỉnh sửa" và "Xóa".
  + Có hiệu ứng hover và được tối ưu accessibility bằng ARIA labels.
* Modal Component:
  + Component đa dụng dùng cho cả việc Thêm và Sửa sách.
  + Như đã mô tả ở trên, nó bao gồm form, validation, preview ảnh, và focus trap.
* Toast Notifications:
  + Là các thông báo nhỏ trượt ra ở góc màn hình để phản hồi hành động của người dùng (ví dụ: "Thêm sách thành công", "Xóa sách thành công").
  + Thông báo sẽ tự động ẩn sau 3 giây.
* SkeletonLoader:
  + Là hiệu ứng loading (còn gọi là shimmer effect) hiển thị khung xương của các BookCard và StatsCard trong khi chờ dữ liệu được tải.
  + Điều này giúp cải thiện trải nghiệm người dùng, cho cảm giác ứng dụng tải nhanh hơn.

## **Giao diện tổng thể**

Bố cục chung của ứng dụng được em chia thành 3 phần rõ ràng:

### Header:

* + Chứa logo "📚 BookVerse" và tên dự án.
  + Chứa các nút hành động chính: "Xuất JSON", "Nhập JSON", "Thêm Sách", và nút chuyển đổi theme (sáng/tối).

### Main Content: Được chia làm 3 khu vực:

* + **Stats Section**: Nằm ở trên cùng, hiển thị 3 thẻ thống kê (tổng sách, tổng thể loại, thể loại phổ biến) và biểu đồ tròn Chart.js.
  + **Controls Section**: Khu vực chứa các bộ điều khiển, bao gồm ô tìm kiếm (Search input) và bộ lọc thể loại (Category filter).
  + **Books Section**: Khu vực chính, sử dụng CSS Grid để hiển thị danh sách các BookCard.

### Modal Overlay:

* + Là lớp phủ mờ (backdrop blur effect) xuất hiện khi Modal được kích hoạt, chứa form thêm/sửa sách.
  + Người dùng có thể đóng modal bằng phím Escape hoặc click ra bên ngoài vùng form.

## **Phong cách thiết kế**

Để tạo nên sự hiện đại và thu hút cho dự án, em đã tập trung vào các kỹ thuật thiết kế giao diện sau:

* Glassmorphism (Thiết kế kính mờ):
  + Đây là phong cách chủ đạo của toàn bộ ứng dụng. Em sử dụng thuộc tính CSS backdrop-filter: blur(40px) saturate(200%) kết hợp với background: rgba(255, 255, 255, 0.1) để tạo hiệu ứng kính mờ, có chiều sâu.
* Liquid Glass Effect (Hiệu ứng kính lỏng):
  + Đây là một phiên bản nâng cao của glassmorphism mà em áp dụng. Bằng cách sử dụng radial-gradients và transform 3D khi hover, em tạo ra hiệu ứng ánh sáng di chuyển trên bề mặt "kính", mang lại cảm giác cao cấp giống như hiệu ứng trên iPhone.
* Dark/Light Theme:
  + Ứng dụng hỗ trợ 2 chế độ hiển thị sáng và tối.
  + Em sử dụng **CSS Custom Properties** (biến CSS) để quản lý màu sắc, giúp việc chuyển đổi theme mượt mà và dễ bảo trì code.
  + Lựa chọn theme của người dùng được lưu vào LocalStorage.
* Responsive Design:
  + Em áp dụng phương pháp **Mobile-first**, thiết kế cho màn hình di động trước, sau đó mở rộng lên tablet và desktop.
  + Em sử dụng kết hợp CSS Grid và Flexbox để tạo layout linh hoạt, đảm bảo trải nghiệm tốt trên mọi kích thước màn hình.

## **Flow người dùng**

Cuối cùng, em xin mô tả một luồng sử dụng điển hình của người dùng:

1. **Mở trang**: Người dùng truy cập ứng dụng. Hệ thống sẽ load theme (sáng/tối) từ LocalStorage, khởi tạo Controller, sau đó tải dữ liệu sách.
2. **Hiển thị**: Giao diện render các BookCard, hiển thị các số liệu thống kê và khởi tạo biểu đồ Chart.js.
3. **Thêm sách**: Người dùng nhấn "Thêm Sách". Modal mở ra. Người dùng nhập thông tin và nhấn "Submit". Dữ liệu được lưu vào LocalStorage, và giao diện được cập nhật ngay lập tức.
4. **Tìm kiếm**: Người dùng gõ từ khóa vào ô tìm kiếm. Sau 300ms (nhờ debounce), danh sách sách tự động được lọc và hiển thị lại.
5. **Thống kê**: Mỗi khi dữ liệu (thêm/sửa/xóa) thay đổi, hệ thống sẽ tự động tính toán lại số liệu thống kê và cập nhật lại biểu đồ.

# PHẦN 5. CÔNG NGHỆ SỬ DỤNG

Để xây dựng dự án BookVerse, em đã lựa chọn các công nghệ hoàn toàn là "Vanilla" (thuần túy), không phụ thuộc vào các framework lớn như React hay Vue, nhằm tập trung vào kiến thức JavaScript cốt lõi và kiến trúc phần mềm.

## **Ngôn ngữ và framework**

* JavaScript ES6+: Em sử dụng các tính năng hiện đại của JavaScript như Modules, Classes, cú pháp Async/Await, và Destructuring để tổ chức code sạch sẽ và hiệu quả.
* HTML5: Em sử dụng các thẻ HTML5 có ý nghĩa (Semantic elements) và các thuộc tính hỗ trợ tiếp cận (accessibility attributes) để xây dựng cấu trúc trang web.
* CSS3: Em đã dùng các tính năng nâng cao của CSS3 như Custom Properties (biến CSS) để quản lý theme, CSS Grid và Flexbox để dựng layout, cùng với các hiệu ứng animations và backdrop-filter cho hiệu ứng glassmorphism.

## **Thư viện và công cụ**

Dự án chỉ sử dụng một số thư viện và công cụ hỗ trợ tối thiểu:

* Vite: Em chọn Vite làm công cụ build và máy chủ phát triển (development server). Vite cung cấp tốc độ HMR (Hot Module Replacement) cực nhanh, giúp quá trình code và debug hiệu quả hơn.
* Chart.js: Đây là thư viện duy nhất em dùng cho việc trực quan hóa dữ liệu, cụ thể là để vẽ biểu đồ tròn (doughnut chart) cho phần thống kê.
* LocalStorage API: Em sử dụng API có sẵn của trình duyệt này để lưu trữ dữ liệu sách và cài đặt theme của người dùng một cách bền bỉ.
* Terser: Công cụ này được Vite tích hợp sẵn để nén (minify) code JavaScript khi build-production, giúp giảm dung lượng file và tăng tốc độ tải trang.

## **Cấu hình và chạy dự án**

Để cài đặt và chạy dự án, người dùng chỉ cần thực hiện các lệnh npm cơ bản:

* Cài đặt dependencies:

**npm install**

* Chạy development server:

**npm run dev**

(Dự án sẽ chạy trên http://localhost:3000)

* Build bản production:

**npm run build**

(Lệnh này sẽ tạo ra thư mục dist/ chứa các file đã được tối ưu)

* Xem trước bản production:

**npm run preview**

## **Cơ chế build và deploy**

* **Vite Configuration**: Em cấu hình Vite để trỏ vào thư mục Source/ làm gốc, và thư mục build output là dist/. Em cũng tắt sourcemap cho bản production để tối ưu.
* **Build Process**:
  1. **Development**: Sử dụng npm run dev với máy chủ HMR của Vite.
  2. **Build**: Chạy npm run build, Vite sẽ tự động tối ưu, nén code JavaScript/CSS.
  3. **Output**: Toàn bộ file tĩnh (static files) cần thiết cho website sẽ nằm trong thư mục dist/.
  4. **Deploy**: Vì đây là ứng dụng frontend-only, em chỉ cần upload toàn bộ nội dung thư mục dist/ lên bất kỳ dịch vụ web server tĩnh nào (như GitHub Pages, Netlify, Vercel) là có thể chạy được.

# PHẦN 6. KIỂM THỬ VÀ KẾT QUẢ

## **Kiểm thử các chức năng chính**

Em đã thực hiện kiểm thử thủ công (manual testing) cho các kịch bản sử dụng chính và ghi nhận kết quả như sau:

* **Thêm sách**:
  + Form validation (kiểm tra hợp lệ) hoạt động chính xác, báo lỗi khi thiếu trường bắt buộc.
  + Chức năng preview ảnh bìa hiển thị đúng ảnh khi dán URL.
  + Dữ liệu được lưu thành công vào LocalStorage sau khi thêm.
  + Giao diện (UI) được cập nhật ngay lập tức, hiển thị sách mới.
* **Tìm kiếm**:
  + Kỹ thuật Debouncing hoạt động đúng (chỉ tìm kiếm sau 300ms ngừng gõ).
  + Tìm kiếm hoạt động chính xác trên tất cả các trường: tiêu đề, tác giả, thể loại, mô tả.
  + Tìm kiếm không phân biệt chữ hoa/thường (Case-insensitive).
* **Xóa sách**:
  + Hộp thoại xác nhận (Confirmation dialog) hiển thị đúng khi nhấn nút xóa.
  + Dữ liệu được xóa vĩnh viễn khỏi LocalStorage sau khi xác nhận.
  + Giao diện cập nhật lại, đồng thời các số liệu thống kê cũng được tính toán lại chính xác.
  + Toast notification (thông báo) xóa thành công hiển thị đúng.
* **Biểu đồ (Chart.js)**:
  + Biểu đồ Chart.js tải và hiển thị (render) đúng dữ liệu phân bố thể loại.
  + Giao diện dự phòng (fallback display) hiển thị chính xác khi cố tình chặn không cho Chart.js tải.
  + Biểu đồ tự động cập nhật khi dữ liệu sách thay đổi.

## **Ảnh hưởng lên LocalStorage**

Em đã kiểm tra LocalStorage của trình duyệt và xác nhận các hành vi sau:

* **Thêm sách**: Một item mới được push vào mảng bookverse\_books.
* **Sửa sách**: Item tương ứng trong mảng được cập nhật (update).
* **Xóa sách**: Item tương ứng được xóa (remove) khỏi mảng.
* **Import**: Toàn bộ mảng bookverse\_books được thay thế (replace) bằng dữ liệu từ file JSON.
* **Export**: Dữ liệu trong mảng được serialize (tuần tự hóa) thành chuỗi JSON để tải về, không làm thay đổi LocalStorage.

## **Hiệu năng và tương thích**

* **Tốc độ phản hồi**: Các thao tác CRUD (Thêm, Sửa, Xóa) đều cho phản hồi ngay lập tức (dưới 100ms) vì chỉ xử lý ở phía client.
* **Tìm kiếm**: Nhờ có debouncing, ứng dụng không bị giật (lag) ngay cả khi gõ phím nhanh.
* **Tương thích trình duyệt**: Ứng dụng hoạt động ổn định trên các trình duyệt hiện đại em đã kiểm thử, bao gồm Chrome, Firefox, Safari (trên macOS), và Edge.
* **Hiệu năng di động**: Giao diện responsive hoạt động tốt, các hiệu ứng và cuộn trang mượt mà trên thiết bị di động.

# PHẦN 7. ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

Sau quá trình thực hiện và kiểm thử, em xin đưa ra một số đánh giá chủ quan về những ưu điểm, hạn chế của dự án và đề xuất hướng phát triển trong tương lai.

## **Ưu điểm**

Dự án BookVerse đã đạt được một số ưu điểm nổi bật sau:

* **Kiến trúc gọn gàng (Clean Architecture)**:
  + Code được tổ chức rất rõ ràng theo các design patterns đã đề cập (MVC, Singleton, Factory, Observer).
  + Việc tách biệt rõ ràng các thành phần (Separation of Concerns) giúp dự án dễ bảo trì và mở rộng về sau.
* **Giao diện đẹp và hiện đại**:
  + Việc áp dụng phong cách thiết kế **Glassmorphism** và hiệu ứng **Liquid Glass** nâng cao đã mang lại trải nghiệm thị giác rất tốt.
  + Giao diện responsive tốt trên nhiều thiết bị.
  + Việc hỗ trợ đầy đủ Dark/Light theme cũng là một điểm cộng lớn.
* **Code rõ ràng (Clean Code)**:
  + Em đã cố gắng viết JSDoc comments đầy đủ cho các hàm và lớp.
  + Quy tắc đặt tên (naming conventions) nhất quán và có cơ chế xử lý lỗi (error handling) tốt.
  + Dự án cũng tuân thủ các tiêu chuẩn về hỗ trợ tiếp cận (Accessibility compliance).

## **Hạn chế**

Bên cạnh các ưu điểm, dự án vẫn còn tồn tại nhiều hạn chế mang tính cố hữu do phạm vi của đồ án môn học:

* **Không có backend**:
  + Đây là hạn chế lớn nhất. Dữ liệu hiện chỉ được lưu cục bộ tại trình duyệt người dùng (Local Storage).
  + Không có API, không có xác thực người dùng (authentication).
  + Không thể đồng bộ hóa dữ liệu (data synchronization) giữa nhiều thiết bị.
* **Giới hạn của LocalStorage**:
  + LocalStorage có dung lượng lưu trữ rất giới hạn (chỉ khoảng 5-10MB).
  + Dữ liệu không bền vững, có thể bị mất nếu người dùng xóa cache trình duyệt và không có cơ chế backup tự động.

## **Hướng phát triển tương lai**

Từ những hạn chế trên, em xin đề xuất các hướng phát triển để hoàn thiện BookVerse thành một sản phẩm thực thụ:

* **Tích hợp Backend (Backend Integration)**:
  + Xây dựng một API server bằng Node.js/Express.
  + Sử dụng cơ sở dữ liệu thực sự như MongoDB hoặc PostgreSQL để lưu trữ dữ liệu.
  + Triển khai hệ thống xác thực người dùng bằng JWT (JSON Web Tokens).
* **Phát triển tính năng nâng cao (Advanced Features)**:
  + Xây dựng hệ thống tài khoản người dùng (user profiles).
  + Thêm các tính năng xã hội như đánh giá (ratings), bình luận (reviews) sách.
  + Cho phép người dùng theo dõi tiến độ đọc (reading progress tracking) và gợi ý sách (recommendations).
* **Tích hợp Cloud và PWA**:
  + Sử dụng các dịch vụ đám mây như Firebase hoặc AWS để đồng bộ dữ liệu real-time và lưu trữ ảnh bìa sách.
  + Phát triển ứng dụng thành một Progressive Web App (PWA) để hỗ trợ chạy offline.
* **Phát triển ứng dụng di động (Mobile App)**:
  + Xây dựng phiên bản app native cho di động bằng React Native hoặc Flutter để tận dụng các tính năng như push notification hay camera (để quét mã vạch sách).

# PHẦN 8. KẾT LUẬN

## **Giá trị kỹ thuật**

Qua dự án BookVerse, em đã áp dụng thành công các mẫu thiết kế (design patterns) quan trọng trong thực tế, đây là giá trị kỹ thuật cốt lõi của đồ án:

* **MVC Pattern**: Giúp em tách biệt rõ ràng logic (Model), giao diện (View), và phần điều khiển (Controller).
* **Singleton Pattern**: Đảm bảo chỉ có một thể hiện duy nhất cho BookManager và EventBus, giúp quản lý tài nguyên tập trung.
* **Factory Pattern**: Giúp việc tạo các đối tượng Book trở nên linh hoạt.
* **Observer Pattern**: Thông qua EventBus, em xây dựng được một kiến trúc hướng sự kiện (event-driven), giúp các component giao tiếp mà không phụ thuộc trực tiếp vào nhau.

## **Giá trị học thuật**

Về mặt học thuật, dự án này là cơ hội để em rèn luyện và chứng minh khả năng:

* **Lập trình JavaScript hiện đại**: Sử dụng thành thạo các tính năng ES6+ như modules, classes, và async/await.
* **Thiết kế UI/UX**: Áp dụng các kỹ thuật thiết kế giao diện nâng cao như Glassmorphism, responsive design, và đảm bảo tính hỗ trợ tiếp cận (accessibility).
* **Kiến trúc phần mềm**: Tư duy về "clean code", "separation of concerns" để xây dựng một cấu trúc code có thể mở rộng.
* **Tối ưu hiệu năng**: Áp dụng các kỹ thuật như debouncing, lazy loading, và rendering hiệu quả.

## **Ý nghĩa đối với sinh viên (bản thân em)**

Đối với cá nhân em, đồ án này mang lại ý nghĩa thực tiễn rất lớn:

* **Ứng dụng lý thuyết Design Pattern**: Em đã hiểu sâu sắc cách áp dụng các pattern từ sách vở vào một dự án thực tế và thấy rõ lợi ích của chúng trong việc tổ chức code.
* **Làm chủ Frontend hiện đại**: Việc chọn Vanilla JavaScript thay vì dùng framework giúp em hiểu rõ bản chất của DOM manipulation, event handling, và các kỹ thuật CSS3 nâng cao.
* **Kỹ năng quản lý phần mềm**: Em đã học được cách tổ chức cấu trúc file, sử dụng build tools (Vite), và quy trình làm việc (workflow) từ development đến testing và debugging.

## **Kết luận cuối cùng**

BookVerse là một dự án đồ án hoàn chỉnh, thể hiện khả năng của em trong việc áp dụng kiến thức lý thuyết của môn Công nghệ phần mềm vào thực tế. Dự án không chỉ đáp ứng đầy đủ các yêu cầu kỹ thuật về kiến trúc và design patterns, mà còn có giao diện đẹp mắt và trải nghiệm người dùng tốt. Mặc dù chỉ là một ứng dụng frontend-only, BookVerse đã cung cấp cho em một nền tảng vững chắc để phát triển các ứng dụng web phức tạp hơn trong tương lai.

# TÀI LIỆU THAM KHẢO

* Design Patterns: Elements of Reusable Object-Oriented Software - Gang of Four
* JavaScript: The Good Parts - Douglas Crockford
* CSS Grid Layout - Rachel Andrew
* Chart.js Documentation
* Vite Documentation