BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC GIA ĐỊNH**

**KHOA: CÔNG NGHỆ THÔNG TIN**

Logo, company name

Description automatically generated

**TIỂU LUẬN**

**TÊN ĐỀ TÀI: WEBSITE GIỚI THIỆU, BÁN SÁCH TRỰC TUYẾN**

**MÔN: LẬP TRÌNH NODEJS**

Ngành: **CÔNG NGHỆ THÔNG TIN**

Chuyên ngành: **KỸ THUẬT PHẦN MỀM**

Giảng viên hướng dẫn: **LÊ HUỲNH PHƯỚC**

Sinh viên thực hiện: **ĐẶNG QUỐC DINH**

MSSV: **2108110031**

Lớp: **K15DCPM04**

TP. Hồ Chí Minh, tháng 4 năm 2024

**Khoa/Viện: Công nghệ thông tin**

**NHẬN XÉT VÀ CHẤM ĐIỂM CỦA GIẢNG VIÊN**

**TIỂU LUẬN MÔN: LẬP TRÌNH NODEJS**

1. **Họ và tên sinh viên: Đặng Quốc Dinh**
2. **Tên đề tài**: **Website giới thiệu, bán sách trực tuyến**
3. **Nhận xét**:

***Những kết quả đạt được:***

***Những hạn chế:***

1. **Điểm đánh giá** *(theo thang điểm 10, làm tròn đến 0.5):*

Sinh viên: Đặng Quốc Dinh

Điểm số: ……….…… Điểm chữ: ……………………………………………………..

|  |  |
| --- | --- |
|  | *TP. HCM, ngày … tháng … năm 20……*  **Giảng viên chấm thi**  *(Ký và ghi rõ họ tên)* |

MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN VỀ NODEJS 5](#_Toc164156859)

[1.1. Giới thiệu về NodeJS 5](#_Toc164156860)

[1.1.1. Khái niệm cơ bản về NodeJS 5](#_Toc164156861)

[1.1.2. Lịch sử phát triển 5](#_Toc164156862)

[1.1.3. Đặc điểm nổi bật 6](#_Toc164156863)

[1.1.4. Ứng dụng và phổ biến 8](#_Toc164156864)

[1.1.5. Ưu điểm và nhược điểm 9](#_Toc164156865)

[1.2. Cài đặt và Cấu hình 10](#_Toc164156866)

[1.2.1. Cài đặt NodeJS 10](#_Toc164156867)

[1.2.2. Quản lý phiên bản với Node Version Manager 11](#_Toc164156868)

[1.2.3. Sử dụng npm (Node Package Manager) 12](#_Toc164156869)

[1.3. Cơ bản về Lập trình với NodeJS 13](#_Toc164156870)

[1.3.1. Cú pháp cơ bản 13](#_Toc164156871)

[1.3.2. Quản lý Module và Dependency 14](#_Toc164156872)

[1.3.3. Asynchronous Programming và Callbacks 15](#_Toc164156873)

[1.4. ExpressJS và Frameworks liên quan 16](#_Toc164156874)

[1.4.1. Giới thiệu về ExpressJS 16](#_Toc164156875)

[1.4.2. Cài đặt và sử dụng ExpressJS 17](#_Toc164156876)

[1.4.3. Middleware và Routing 17](#_Toc164156877)

[1.5. Xử lý Dữ liệu 18](#_Toc164156878)

[1.5.1. Kết nối và thao tác với cơ sở dữ liệu (MongoDB, MySQL, ...) 18](#_Toc164156879)

[1.5.2. Xử lý JSON 18](#_Toc164156880)

[1.5.3. Sử dụng RESTful APIs 19](#_Toc164156881)

[1.6. Tài nguyên và Cộng đồng 20](#_Toc164156882)

[1.6.1. Tài liệu học và tài nguyên tham khảo 20](#_Toc164156883)

[1.6.2. Cộng đồng NodeJS và các diễn đàn trao đổi kiến thức 21](#_Toc164156884)

[1.6.3. Các dự án mã nguồn mở và thư viện phổ biến 22](#_Toc164156885)

[1.7. Tổng kết và các xu hướng mới, tiềm năng phát triển của NodeJS 23](#_Toc164156886)

[1.7.1. Các xu hướng mới và tiềm năng phát triển của NodeJS 23](#_Toc164156887)

[1.7.2. Tổng kết 24](#_Toc164156888)

[CHƯƠNG 2: Website giới thiệu, bán sách trực tuyến 25](#_Toc164156889)

[2.1. Giao diện trang chủ 25](#_Toc164156890)

[2.2. Giao diện đăng nhập 25](#_Toc164156891)

[2.3. Giao diện Admin 26](#_Toc164156892)

[2.4. Giao diện sinh viên 29](#_Toc164156893)

[TÀI LIỆU THAM KHẢO 30](#_Toc164156894)

# CHƯƠNG 1: TỔNG QUAN VỀ NODEJS

## 1.1. Giới thiệu về NodeJS

NodeJS là một nền tảng phát triển ứng dụng mạng được xây dựng dựa trên JavaScript runtime của Chrome (V8 Engine). Điều này cho phép các nhà phát triển sử dụng JavaScript để xây dựng cả phần máy chủ và phần khách hàng của ứng dụng, tạo ra một môi trường lập trình đồng nhất và linh hoạt.

Ban đầu ra mắt vào năm 2009 bởi Ryan Dahl, NodeJS nhanh chóng thu hút sự quan tâm của cộng đồng lập trình với việc giải quyết các vấn đề liên quan đến việc xử lý đa luồng và I/O không đồng bộ. Một trong những điểm đặc biệt của NodeJS là sự kết hợp giữa mô hình non-blocking I/O và single-threaded event loop, giúp tối ưu hóa hiệu suất và sự linh hoạt trong xử lý các yêu cầu mạng.

NodeJS được sử dụng rộng rãi trong nhiều loại ứng dụng, từ các ứng dụng web real-time, API và microservices, đến ứng dụng IoT và server-side rendering. Sự mở rộng của nó cũng được hỗ trợ thông qua cộng đồng lớn và sôi động, với hàng ngàn các module và framework được cung cấp thông qua npm (Node Package Manager).

### 1.1.1. Khái niệm cơ bản về NodeJS

NodeJS là một môi trường runtime JavaScript mã nguồn mở và đa nền tảng được sử dụng để xây dựng các ứng dụng web. Nó được phát triển vào năm 2009 bởi Ryan Dahl và dựa trên Chrome V8 JavaScript engine. NodeJS cho phép các lập trình viên viết mã JavaScript cho cả ứng dụng front-end và back-end.

### 1.1.2. Lịch sử phát triển

NodeJS được tạo ra bởi Ryan Dahl vào năm 2009, với mục tiêu ban đầu là tạo ra một trang web có khả năng push như trong các ứng dụng web như Gmail. Sau khi thử nghiệm với một số ngôn ngữ khác nhau, Dahl chọn JavaScript vì nó có API Nhập/Xuất không đầy đủ.

- Dưới đây là một số mốc lịch sử quan trọng trong quá trình phát triển của NodeJS:

+ 2009: Ryan Dahl giới thiệu ý tưởng về mô hình lập trình phi đồng bộ cho các ứng dụng mạng.

+ 2010: Phiên bản NodeJS đầu tiên được phát hành, hỗ trợ Linux và Mac OS X.

+ 2011: Phiên bản NodeJS đầu tiên hỗ trợ Windows được phát hành.

+ 2012: Dahl từ chức và trao quyền quản lý dự án cho Isaac Schlueter, đồng thời NodeJS được InfoWorld bình chọn là "Công nghệ của năm".

+ 2014: Phiên bản v0.10.0 được phát hành, đánh dấu sự trưởng thành và ổn định của NodeJS.

+ 2015: The NodeJS Foundation được thành lập để thúc đẩy sự phát triển và áp dụng của NodeJS.

+ 2016: Phiên bản v6.0.0 được phát hành, giới thiệu nhiều tính năng mới và cải tiến hiệu suất.

+ 2017: Phiên bản v8.0.0 được phát hành, đánh dấu sự trưởng thành của NodeJS như một nền tảng doanh nghiệp.

+ 2018: Phiên bản v10.0.0 được phát hành, tập trung vào cải thiện hiệu suất và độ tin cậy.

+ 2019: Phiên bản v12.0.0 được phát hành, giới thiệu nhiều tính năng mới và cải tiến bảo mật.

+ 2020: Phiên bản v14.0.0 được phát hành, tập trung vào cải thiện hiệu suất và hỗ trợ các tính năng mới của JavaScript.

+ 2021: Phiên bản v16.0.0 được phát hành, giới thiệu nhiều tính năng mới và cải thiện độ tin cậy.

+ 2022: Phiên bản v18.0.0 được phát hành, tập trung vào cải thiện hiệu suất và hỗ trợ các tính năng mới của JavaScript.

+ 2023: Phiên bản v19.0.0 được phát hành, giới thiệu nhiều tính năng mới và cải thiện độ bảo mật.

Ngày nay, NodeJS là một trong những nền tảng phát triển phổ biến nhất cho các ứng dụng web và mạng. Nó được sử dụng bởi các công ty lớn như PayPal, Netflix, Uber và eBay. NodeJS cũng có một cộng đồng lập trình viên lớn và tích cực, cung cấp nhiều tài nguyên và hỗ trợ cho người dùng.

### 1.1.3. Đặc điểm nổi bật

NodeJS là nền tảng phát triển mã nguồn mở đa nền tảng được xây dựng dựa trên trình thông dịch JavaScript V8 của Chrome. Nó cho phép các lập trình viên viết mã JavaScript cho cả ứng dụng front-end và back-end, mang lại nhiều lợi ích như:

- Hiệu quả:

+ Sử dụng mô hình lập trình phi đồng bộ (asynchronous) và không chặn (non-blocking) dựa trên sự kiện, giúp xử lý nhiều yêu cầu cùng lúc, tăng hiệu quả ứng dụng.

+ Khả năng I/O không đồng bộ giúp NodeJS xử lý nhiều kết nối mạng cùng lúc mà không cần chờ đợi kết quả từ từng kết nối, dẫn đến hiệu suất cao hơn so với các ngôn ngữ truyền thống như PHP.

- Dễ học:

+ Sử dụng JavaScript, ngôn ngữ lập trình phổ biến và dễ học, giúp các lập trình viên web dễ dàng tiếp cận và bắt đầu phát triển ứng dụng NodeJS.

+ Cộng đồng NodeJS lớn và hoạt động mạnh mẽ, cung cấp nhiều tài liệu và hướng dẫn học tập miễn phí.

- Linh hoạt:

+ Chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, mang lại sự linh hoạt cho các nhà phát triển trong việc triển khai ứng dụng.

+ Hỗ trợ nhiều module và thư viện JavaScript phong phú, giúp đơn giản hóa việc phát triển các ứng dụng phức tạp.

- Mở rộng:

+ Kiến trúc đơn luồng phi đồng bộ giúp NodeJS dễ dàng mở rộng để đáp ứng nhu cầu lưu lượng truy cập tăng cao.

+ Có thể sử dụng các công nghệ đám mây như AWS, Google Cloud Platform và Azure để triển khai các ứng dụng NodeJS một cách dễ dàng và hiệu quả.

- Cộng đồng:

+ Cộng đồng lập trình viên NodeJS lớn và tích cực, luôn sẵn sàng hỗ trợ và chia sẻ kiến thức với nhau.

+ Nhiều tài liệu, hướng dẫn và công cụ học tập miễn phí được cung cấp bởi cộng đồng, giúp người mới bắt đầu dễ dàng tiếp cận và học NodeJS.

- Ngoài những đặc điểm nổi bật trên, Node.js còn có một số ưu điểm khác như:

+ Phát triển nhanh chóng: Mô hình lập trình phi đồng bộ giúp đơn giản hóa việc phát triển ứng dụng và tiết kiệm thời gian.

+ Mã thống nhất: Sử dụng JavaScript cho cả front-end và back-end giúp giảm bớt sự phức tạp và dễ dàng bảo trì code hơn.

+ Phù hợp cho các ứng dụng thời gian thực: Khả năng xử lý phi đồng bộ và không chặn của + NodeJS giúp nó lý tưởng cho các ứng dụng thời gian thực như trò chuyện trực tuyến và game trực tuyến.

Nhìn chung, NodeJS là nền tảng phát triển mạnh mẽ, linh hoạt và dễ sử dụng, phù hợp cho nhiều loại ứng dụng khác nhau. Nó là lựa chọn lý tưởng cho các nhà phát triển muốn xây dựng ứng dụng nhanh chóng, hiệu quả và dễ dàng mở rộng.

### 1.1.4. Ứng dụng và phổ biến

NodeJS là nền tảng phát triển mã nguồn mở đa nền tảng được xây dựng dựa trên trình thông dịch JavaScript V8 của Chrome. Nhờ những đặc điểm nổi bật như hiệu quả, dễ học, linh hoạt, mở rộng và cộng đồng lớn, NodeJS được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau, bao gồm:

- Phát triển web:

+ Ứng dụng web động: NodeJS là lựa chọn phổ biến cho việc xây dựng các ứng dụng web động như trang web thương mại điện tử, mạng xã hội và ứng dụng web một trang (SPA).

+ API web: NodeJS được sử dụng để xây dựng API web cung cấp dữ liệu và chức năng cho các ứng dụng khác.

+ Máy chủ web: NodeJS có thể được sử dụng để xây dựng máy chủ web hiệu quả và có thể mở rộng cao.

- Ứng dụng mạng:

+ Máy chủ chat: NodeJS là lựa chọn lý tưởng cho việc xây dựng máy chủ chat thời gian thực như WebSocket.

+ Ứng dụng nhắn tin: NodeJS được sử dụng để xây dựng các ứng dụng nhắn tin như Skype và Facebook Messenger.

+ Ứng dụng chơi game trực tuyến: NodeJS có thể được sử dụng để xây dựng các ứng dụng chơi game trực tuyến đòi hỏi hiệu suất cao.

- Công cụ dòng lệnh:

+ Trình thu thập dữ liệu web: NodeJS được sử dụng để viết các công cụ thu thập dữ liệu web tự động hóa việc thu thập dữ liệu từ các trang web.

+ Công cụ quản lý hệ thống: NodeJS có thể được sử dụng để viết các công cụ quản lý hệ thống tự động hóa các tác vụ quản trị hệ thống.

- Ứng dụng IoT:

+ Hệ thống nhà thông minh: NodeJS được sử dụng để xây dựng hệ thống nhà thông minh cho phép điều khiển các thiết bị gia dụng từ xa.

+ Thiết bị đeo: NodeJS có thể được sử dụng để phát triển phần mềm cho các thiết bị đeo như đồng hồ thông minh và vòng tay thông minh.

- Các ứng dụng khác:

+ Ứng dụng xử lý ảnh: NodeJS được sử dụng để xây dựng các ứng dụng xử lý ảnh như chỉnh sửa ảnh và chuyển đổi ảnh.

+ Ứng dụng học máy: NodeJS có thể được sử dụng để phát triển các mô hình học máy và triển khai chúng vào ứng dụng.

NodeJS là một trong những nền tảng phát triển phổ biến nhất hiện nay. Theo khảo sát của NodeJS Foundation: https://nodejs.org/en/, NodeJS được sử dụng bởi hơn 40% nhà phát triển web. Nó được sử dụng bởi nhiều công ty lớn như PayPal, Netflix, Uber, eBay và Microsoft.

### 1.1.5. Ưu điểm và nhược điểm

- Ưu điểm:

+ Hiệu quả: NodeJS sử dụng mô hình lập trình phi đồng bộ (asynchronous) và không chặn (non-blocking) dựa trên sự kiện, giúp xử lý nhiều yêu cầu cùng lúc, tăng hiệu quả ứng dụng. Khả năng I/O không đồng bộ giúp NodeJS xử lý nhiều kết nối mạng cùng lúc mà không cần chờ đợi kết quả từ từng kết nối, dẫn đến hiệu suất cao hơn so với các ngôn ngữ truyền thống như PHP.

+ Dễ học: NodeJS sử dụng JavaScript, ngôn ngữ lập trình phổ biến và dễ học, giúp các lập trình viên web dễ dàng tiếp cận và bắt đầu phát triển ứng dụng NodeJS. Cộng đồng NodeJS lớn và hoạt động mạnh mẽ, cung cấp nhiều tài liệu và hướng dẫn học tập miễn phí.

+ Linh hoạt: NodeJS chạy trên nhiều hệ điều hành khác nhau như Windows, macOS và Linux, mang lại sự linh hoạt cho các nhà phát triển trong việc triển khai ứng dụng. Hỗ trợ nhiều module và thư viện JavaScript phong phú, giúp đơn giản hóa việc phát triển các ứng dụng phức tạp.

+ Mở rộng: Kiến trúc đơn luồng phi đồng bộ giúp NodeJS dễ dàng mở rộng để đáp ứng nhu cầu lưu lượng truy cập tăng cao. Có thể sử dụng các công nghệ đám mây như AWS, Google Cloud Platform và Azure để triển khai các ứng dụng NodeJS một cách dễ dàng và hiệu quả.

+ Cộng đồng: Cộng đồng lập trình viên NodeJS lớn và tích cực, luôn sẵn sàng hỗ trợ và chia sẻ kiến thức với nhau. Nhiều tài liệu, hướng dẫn và công cụ học tập miễn phí được cung cấp bởi cộng đồng, giúp người mới bắt đầu dễ dàng tiếp cận và học NodeJS.

+ Phát triển nhanh chóng: Mô hình lập trình phi đồng bộ giúp đơn giản hóa việc phát triển ứng dụng và tiết kiệm thời gian.

+ Mã thống nhất: Sử dụng JavaScript cho cả front-end và back-end giúp giảm bớt sự phức tạp và dễ dàng bảo trì code hơn.

+ Phù hợp cho các ứng dụng thời gian thực: Khả năng xử lý phi đồng bộ và không chặn của NodeJS giúp nó lý tưởng cho các ứng dụng thời gian thực như trò chuyện trực tuyến và game trực tuyến.

- Nhược điểm:

+ Độ cong học tập: Mặc dù NodeJS tương đối dễ học, nhưng vẫn có một số khái niệm mới cần được hiểu, đặc biệt cho người mới bắt đầu lập trình.

+ Hệ sinh thái non trẻ: Hệ sinh thái NodeJS vẫn còn tương đối non trẻ so với các nền tảng khác như Java hoặc PHP. Điều này có nghĩa là có thể ít tài nguyên và hỗ trợ hơn cho một số trường hợp sử dụng cụ thể.

+ Vấn đề hiệu suất: NodeJS có thể gặp vấn đề hiệu suất với các ứng dụng đòi hỏi tính toán chuyên sâu. Tuy nhiên, hiệu suất của NodeJS đã được cải thiện đáng kể trong những năm gần đây và nó có thể xử lý hầu hết các trường hợp sử dụng một cách hiệu quả.

+ Lỗi callback: Viết mã phi đồng bộ có thể dẫn đến lỗi callback, có thể khó gỡ lỗi. Tuy nhiên, có một số kỹ thuật và công cụ có thể giúp giảm thiểu rủi ro mắc lỗi callback.

+ Khả năng bảo mật: NodeJS là một nền tảng mã nguồn mở, điều này có nghĩa là nó dễ bị tấn công hơn các nền tảng độc quyền. Tuy nhiên, có một số biện pháp phòng ngừa có thể được thực hiện để bảo mật ứng dụng NodeJS.

## 1.2. Cài đặt và Cấu hình

Cài đặt và cấu hình NodeJS là bước quan trọng để bắt đầu phát triển ứng dụng với nền tảng này.

### 1.2.1. Cài đặt NodeJS

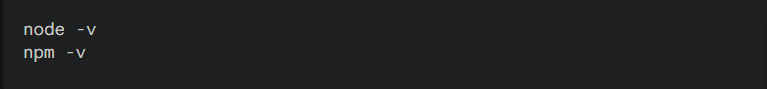
- Dưới đây là hướng dẫn cách cài đặt NodeJS trên các hệ điều hành phổ biến:

+ Cài đặt NodeJS trên Windows:

Bước 1: Tải xuống NodeJS từ trang web chính thức: https://nodejs.org/en/download

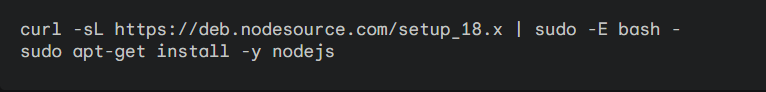
Bước 2: Chạy trình cài đặt đã tải xuống và làm theo hướng dẫn trên màn hình.

Bước 3: Kiểm tra cài đặt bằng cách mở Command Prompt và gõ lệnh sau:

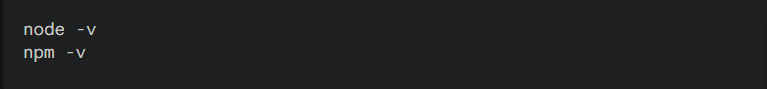


+ Cài đặt NodeJS trên macOS:

Bước 1: Mở Terminal và gõ lệnh sau:



Bước 2: Kiểm tra cài đặt bằng cách mở Terminal và gõ lệnh sau:

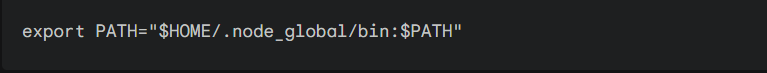


+ Cài đặt NodeJS trên Linux:

Bước 1: Tải xuống NodeJS từ trang web chính thức: https://nodejs.org/en/download

Bước 2: Giải nén tệp đã tải xuống.

Bước 3: Thêm đường dẫn đến thư mục bin của NodeJS vào biến môi trường PATH bằng cách chỉnh sửa tệp ~/.bashrc hoặc ~/.zshrc và thêm dòng sau:

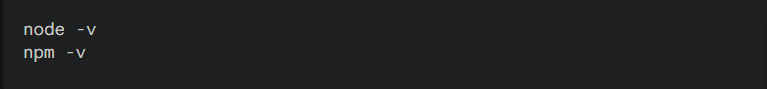


Bước 4: Cập nhật tệp ~/.bashrc hoặc ~/.zshrc bằng cách gõ lệnh sau:



Bước 5: Khởi động lại máy tính.

Bước 6: Kiểm tra cài đặt bằng cách mở Terminal và gõ lệnh sau:



### 1.2.2. Quản lý phiên bản với Node Version Manager

NVM (Node Version Manager) là một công cụ quản lý phiên bản NodeJS phổ biến giúp bạn dễ dàng cài đặt, chuyển đổi và sử dụng nhiều phiên bản NodeJS khác nhau trên cùng một hệ thống. NVM giúp bạn linh hoạt hơn trong việc phát triển dự án, cho phép bạn sử dụng phiên bản NodeJS phù hợp nhất cho từng dự án cụ thể.

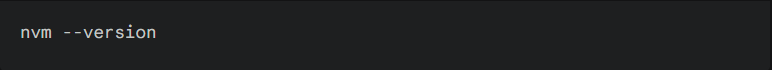
- Cài đặt NVM:

Bước 1: Tải xuống trình cài đặt NVM từ trang web chính thức: github.com/coreybutler/nvm-windows/releases

Bước 2: Chạy trình cài đặt đã tải xuống và làm theo hướng dẫn trên màn hình.

Bước 3: Khởi động lại máy tính.

Bước 4: Kiểm tra cài đặt bằng cách mở Command Prompt và gõ lệnh sau:

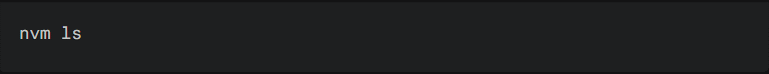


- Sử dụng NVM:

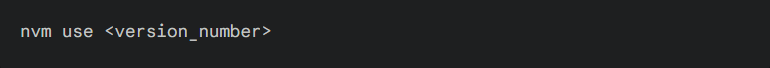
+ Cài đặt phiên bản NodeJS:



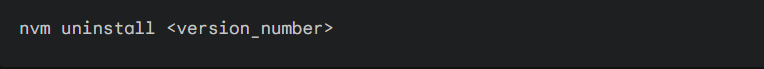
+ Danh sách các phiên bản NodeJS đã cài đặt:



+ Chuyển đổi phiên bản NodeJS mặc định:



+ Xóa phiên bản NodeJS:



### 1.2.3. Sử dụng npm (Node Package Manager)

npm (Node Package Manager) là công cụ quản lý gói (package manager) mặc định cho NodeJS, giúp bạn dễ dàng cài đặt, quản lý và cập nhật các thư viện (library) và công cụ (tool) cần thiết cho dự án NodeJS của bạn. npm hoạt động dựa trên kho lưu trữ gói trực tuyến khổng lồ gọi là npm Registry, nơi chứa hàng nghìn gói NodeJS được phát triển bởi cộng đồng.

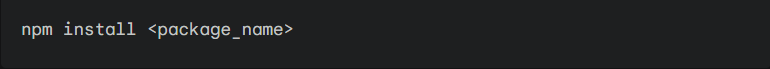
- Cài đặt npm: npm được cài đặt cùng với NodeJS. Do đó, nếu bạn đã cài đặt NodeJS trên hệ thống của mình, bạn đã có npm.

- Sử dụng npm:

+ Tìm kiếm gói:



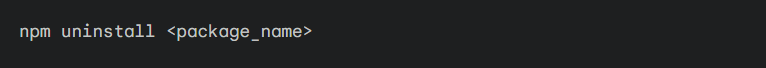
+ Cài đặt gói:



+ Cài đặt gói cục bộ:



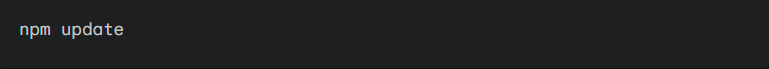
+ Gỡ cài đặt gói:



+ Cập nhật gói:



+ Cập nhật tất cả các gói:



## 1.3. Cơ bản về Lập trình với NodeJS

Để bắt đầu lập trình với NodeJS, bạn có thể tham khảo các ví dụ đơn giản và thực hành viết các chương trình nhỏ. Sau khi đã nắm vững các kiến thức cơ bản, bạn có thể tham gia các dự án thực tế để nâng cao kỹ năng và kinh nghiệm của mình.

### 1.3.1. Cú pháp cơ bản

NodeJS sử dụng cú pháp JavaScript, do đó bạn cần nắm vững kiến thức cơ bản về JavaScript để có thể lập trình với NodeJS. Dưới đây là một số cú pháp cơ bản thường được sử dụng trong lập trình NodeJS:

- Khai báo biến:

+ Sử dụng từ khóa var, let hoặc const để khai báo biến.

+ `var` là từ khóa khai báo biến truyền thống, phạm vi toàn cục.

+ `let` là từ khóa khai báo biến có phạm vi khối.

+ `const` là từ khóa khai báo biến hằng số, không thể thay đổi giá trị sau khi khai báo.

- Gán giá trị: Sử dụng toán tử `=` để gán giá trị cho biến.

- Kiểu dữ liệu: NodeJS hỗ trợ các kiểu dữ liệu cơ bản như số nguyên (number), chuỗi (string), boolean (boolean), và đối tượng (object).

- Toán tử: NodeJS hỗ trợ các toán tử toán học, so sánh, logic, v.v. giống như JavaScript.

- Cấu trúc điều khiển: NodeJS hỗ trợ các cấu trúc điều khiển như if, else, switch, for, while, v.v. để điều khiển luồng thực thi chương trình.

- Hàm: Hàm là khối mã được sử dụng để thực hiện một nhiệm vụ cụ thể.

- Mô-đun: Mô-đun là tập tin JavaScript chứa mã có thể được sử dụng lại trong các chương trình khác.

- Xử lý lỗi: NodeJS sử dụng khối try-catch để xử lý lỗi.

### 1.3.2. Quản lý Module và Dependency

Module và Dependency là hai khái niệm quan trọng trong lập trình NodeJS. Hiểu rõ cách quản lý module và dependency giúp bạn phát triển dự án NodeJS hiệu quả và dễ dàng hơn.

- Module:

+ Module là một tập tin JavaScript đóng gói một hoặc nhiều chức năng có thể được sử dụng lại trong các chương trình khác.

+ Module giúp chia nhỏ mã thành các phần nhỏ hơn, dễ quản lý và tái sử dụng.

+ Có hai loại module chính trong NodeJS:

* Module nội: Module được tích hợp sẵn trong NodeJS.
* Module bên ngoài: Module được cài đặt từ thư viện bên ngoài hoặc do bạn tự viết.

- Dependency:

+ Dependency là module mà một module khác phụ thuộc vào để hoạt động.

+ Khi bạn cài đặt một module, các dependency của nó cũng được cài đặt tự động.

+ Quản lý dependency hiệu quả giúp đảm bảo dự án NodeJS của bạn hoạt động chính xác và không bị xung đột.

- Công cụ quản lý module và dependency:

+ npm: Công cụ quản lý gói mặc định cho NodeJS. npm giúp bạn cài đặt, cập nhật và gỡ cài đặt module và dependency dễ dàng.

+ yarn: Lựa chọn thay thế npm, cung cấp một số tính năng nâng cao như tốc độ cài đặt nhanh hơn và khả năng giải quyết xung đột dependency tốt hơn.

- Cách quản lý module và dependency:

+ Xác định dependency: Xác định các module và dependency cần thiết cho dự án của bạn.

+ Cài đặt dependency: Sử dụng npm hoặc yarn để cài đặt các dependency.

+ Khai báo dependency: Trong tệp package.json của dự án, khai báo các dependency đã cài đặt.

+ Cập nhật dependency: Khi có phiên bản mới của dependency được phát hành, hãy sử dụng npm hoặc yarn để cập nhật.

+ Gỡ cài đặt dependency: Khi không còn cần thiết, hãy sử dụng npm hoặc yarn để gỡ cài đặt dependency.

### 1.3.3. Asynchronous Programming và Callbacks

Asynchronous Programming (Lập trình phi đồng bộ) là một mô hình lập trình cho phép một chương trình thực hiện nhiều tác vụ cùng lúc mà không cần đợi một tác vụ khác hoàn thành. Điều này đặc biệt hữu ích trong NodeJS, nơi các tác vụ I/O (Input/Output) như đọc/ghi file, truy vấn database thường tốn nhiều thời gian.

- Ưu điểm của Asynchronous Programming:

+ Hiệu quả: Cho phép xử lý nhiều yêu cầu đồng thời, cải thiện hiệu suất ứng dụng.

+ Không chặn: Tránh tình trạng code bị treo đợi kết quả của các tác vụ I/O.

+ Scalability: Dễ dàng mở rộng để xử lý nhiều kết nối đồng thời.

Callbacks là một trong những kỹ thuật cơ bản được sử dụng trong Asynchronous Programming của NodeJS. Callback là một hàm được truyền như tham số cho một hàm khác, và được gọi sau khi hàm thứ hai hoàn thành nhiệm vụ của mình.

- Cơ chế hoạt động của Callbacks:

+ Gọi một hàm thực hiện tác vụ I/O (ví dụ: đọc file).

+ Hàm I/O nhận một hàm callback làm tham số.

+ Hàm I/O bắt đầu thực hiện tác vụ.

+ Khi tác vụ I/O hoàn thành, hàm I/O gọi hàm callback và truyền kết quả (nếu có) cho callback.

+ Hàm callback xử lý kết quả trả về từ hàm I/O.

- Nhược điểm của Callbacks:

+ Callback Hell: Khi nhiều tác vụ I/O lồng nhau, code có thể trở nên khó đọc và bảo trì, được gọi là "Callback Hell".

+ Error Handling: Truyền lỗi qua callback phức tạp và dễ bị bỏ sót.

- Các giải pháp thay thế Callbacks:

+ Promises: Cung cấp cú pháp đơn giản hơn để xử lý các tác vụ bất đồng bộ.

+ Async/Await: Cung cấp cú pháp giống như đồng bộ để viết code bất đồng bộ, cải thiện khả năng đọc.

Mặc dù Callbacks là một kỹ thuật cơ bản, việc sử dụng Promises và Async/Await ngày càng được ưa chuộng do tính dễ đọc và quản lý lỗi tốt hơn.

## 1.4. ExpressJS và Frameworks liên quan

ExpressJS là framework web được ưa chuộng nhất cho NodeJS, nổi tiếng với sự đơn giản, linh hoạt và hiệu quả. Tuy nhiên, ExpressJS không phải là framework MVC hoàn chỉnh. Để khắc phục hạn chế này, nhiều frameworks MVC khác được xây dựng dựa trên ExpressJS, cung cấp cấu trúc MVC đầy đủ và hỗ trợ nhiều tính năng nâng cao.

### 1.4.1. Giới thiệu về ExpressJS

ExpressJS là một framework web nhẹ, linh hoạt và mạnh mẽ được xây dựng trên nền tảng NodeJS. Nó cung cấp một API đơn giản và hiệu quả để xử lý routing, HTTP requests, rendering template, và middleware. Nhờ những ưu điểm này, ExpressJS trở thành lựa chọn phổ biến cho việc phát triển các ứng dụng web hiện đại với NodeJS.

- Ưu điểm của ExpressJS:

+ Nhẹ và linh hoạt: ExpressJS có kích thước nhỏ gọn và dễ dàng tích hợp vào các dự án khác nhau.

+ Mạnh mẽ: ExpressJS cung cấp nhiều tính năng mạnh mẽ để xây dựng các ứng dụng web phức tạp.

+ Dễ học và sử dụng: API của ExpressJS đơn giản và dễ hiểu, giúp các developer có thể nhanh chóng bắt đầu sử dụng.

+ Cộng đồng lớn: ExpressJS có cộng đồng người dùng và developer lớn, cung cấp nguồn tài liệu phong phú và hỗ trợ nhiệt tình.

- Tính năng chính của ExpressJS:

+ Routing: ExpressJS cung cấp một hệ thống routing mạnh mẽ để định hướng các HTTP requests đến các controller phù hợp.

+ HTTP requests: ExpressJS cho phép xử lý các HTTP requests một cách dễ dàng, bao gồm GET, POST, PUT, DELETE, v.v.

+ Rendering template: ExpressJS hỗ trợ nhiều template engine khác nhau như Jade, EJS, Pug, v.v. để render giao diện người dùng.

+ Middleware: ExpressJS sử dụng middleware để xử lý các tác vụ chung như logging, authentication, authorization, v.v.

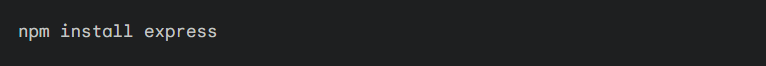
### 1.4.2. Cài đặt và sử dụng ExpressJS

Để bắt đầu sử dụng ExpressJS, bạn cần thực hiện các bước sau:

- Cài đặt NodeJS và npm:

- Cài đặt ExpressJS:

+ Mở Command Prompt hoặc Terminal và sử dụng lệnh sau để cài đặt ExpressJS:



+ Lệnh này sẽ cài đặt ExpressJS và các dependency cần thiết vào thư mục dự án của bạn.

- Tạo ứng dụng ExpressJS: Tạo một thư mục mới cho dự án của bạn và di chuyển đến thư mục đó.

### 1.4.3. Middleware và Routing

ExpressJS là một framework mạnh mẽ cho Node.js, cung cấp nhiều tính năng để xây dựng các ứng dụng web. Trong đó, Middleware và Routing là hai khái niệm quan trọng giúp xử lý các request của người dùng một cách hiệu quả và linh hoạt.

- Routing (Định Tuyến):

+ Routing xác định mối quan hệ giữa các URL và các hàm xử lý tương ứng (handler functions).

+ Khi người dùng gửi một request đến server với một URL cụ thể, ExpressJS sẽ sử dụng routing để tìm handler function phù hợp để xử lý request đó.

+ Routing trong ExpressJS được thực hiện thông qua các phương thức HTTP (GET, POST, PUT, DELETE, v.v.).

- Middleware (Phần Mềm Trung Gian):

+ Middleware là một hàm được thực thi ở giữa quá trình xử lý request của ExpressJS.

+ Middleware có thể thực hiện nhiều tác vụ khác nhau như:

* Kiểm tra xác thực (authentication) và phân quyền (authorization) của người dùng.
* Ghi log các request và response.
* Xử lý dữ liệu request (ví dụ: parsing JSON data).
* Cung cấp các tính năng chung cho nhiều route.

+ Middleware có thể được sử dụng cục bộ cho một route cụ thể hoặc toàn cục cho toàn bộ ứng dụng.

- Sử dụng Middleware và Routing kết hợp: Middleware và Routing thường được sử dụng kết hợp với nhau để xây dựng các ứng dụng ExpressJS hiệu quả. Routing giúp xác định handler function phù hợp, còn middleware có thể được sử dụng trước hoặc sau handler function để thực hiện các tác vụ chung hoặc kiểm tra cần thiết.

## 1.5. Xử lý Dữ liệu

Xử lý dữ liệu là một phần quan trọng trong việc phát triển ứng dụng web và API với ExpressJS. ExpressJS cung cấp các phương tiện để bạn có thể xử lý và truy cập dữ liệu được gửi từ client thông qua các yêu cầu HTTP như GET, POST, PUT và DELETE.

### 1.5.1. Kết nối và thao tác với cơ sở dữ liệu (MongoDB, MySQL, ...)

ExpressJS là framework web mạnh mẽ cho NodeJS, tuy nhiên nó không cung cấp sẵn tính năng kết nối và thao tác với cơ sở dữ liệu. Để thực hiện các tác vụ này, bạn cần sử dụng các thư viện (library) của bên thứ ba. Dưới đây là một số thư viện phổ biến:

- Kết nối với MongoDB:

+ Mongoose: [https://mongoosejs.com/] là một Object Document Mapper (ODM) phổ biến cho MongoDB, giúp đơn giản hóa việc tương tác với database.

+ MongoDB Native Driver: [https://www.mongodb.com/docs/drivers/node/] cung cấp API JavaScript gốc để kết nối và thao tác với MongoDB.

- Kết nối với MySQL:

+ MySQL Driver for NodeJS: [https://github.com/mysqljs/mysql] là driver chính thức cho MySQL, cho phép bạn kết nối và thực thi các câu lệnh SQL.

+ Sequelize: [https://sequelize.org/] là một ORM mạnh mẽ cho NodeJS, hỗ trợ nhiều database khác nhau, bao gồm MySQL, PostgreSQL, SQLite, v.v.

### 1.5.2. Xử lý JSON

ExpressJS cung cấp nhiều phương thức và API để xử lý dữ liệu JSON trong các request và response của ứng dụng web. Dưới đây là một số ví dụ:

- Parse JSON từ Request Body: Để parse dữ liệu JSON từ body request, bạn có thể sử dụng middleware `express.json`. Middleware này sẽ tự động parse JSON body và gán dữ liệu vào object `req.body`.

- Gửi Response JSON: Để gửi response JSON, bạn có thể sử dụng phương thức `res.json`. Phương thức này sẽ tự động chuyển đổi object JavaScript sang định dạng JSON và gửi đến người dùng.

- Sử dụng `JSON.stringify` và `JSON.parse`: Bạn cũng có thể sử dụng các hàm JSON.stringify() và JSON.parse() để chuyển đổi dữ liệu JavaScript sang JSON và ngược lại.

### 1.5.3. Sử dụng RESTful APIs

RESTful API (Representational State Transfer Application Programming Interface) là một tập hợp các nguyên tắc thiết kế cho các API web nhằm tạo ra các giao diện dễ sử dụng, thống nhất và có thể mở rộng. ExpressJS cung cấp nhiều tính năng và API để bạn dễ dàng xây dựng các RESTful API trong ứng dụng web của mình.

- Nguyên tắc RESTful:

+ Tài nguyên (Resource): Mọi thứ trong hệ thống đều được biểu diễn như một tài nguyên, có thể được xác định bằng một URL và có các phương thức HTTP để truy cập và thao tác.

+ Không trạng thái (Stateless): Mỗi request đều độc lập và không phụ thuộc vào trạng thái của các request trước đó. Server không lưu trữ trạng thái của các request trước đó.

+ Cacheable: Các response của API nên có thể cache được để cải thiện hiệu suất.

+ Phân tầng (Layered): Hệ thống API nên được phân tầng thành các thành phần độc lập, dễ dàng bảo trì và mở rộng.

+ Có thể khám phá được (Discoverable): Các API nên cung cấp các tài liệu và metadata để người dùng có thể dễ dàng khám phá và sử dụng.

- Sử dụng ExpressJS để xây dựng RESTful API:

+ Sử dụng phương thức HTTP: ExpressJS cung cấp các phương thức HTTP như GET, POST, + PUT, DELETE để tương ứng với các thao tác CRUD (Create, Read, Update, Delete) trên tài nguyên.

+ Xác định URL: URL của API nên phản ánh rõ ràng tài nguyên và thao tác được thực hiện. Ví dụ: /users (lấy danh sách users), /users/:id (lấy thông tin user với ID cụ thể), /users/:id/posts (lấy danh sách bài viết của user với ID cụ thể).

+ Sử dụng Middleware: ExpressJS cung cấp các middleware để xử lý các tác vụ chung như parse JSON, xác thực người dùng, ghi log, v.v.

+ Gửi Response JSON: ExpressJS cung cấp phương thức res.json() để gửi response JSON một cách dễ dàng.

+ Lập trình hướng đối tượng: Sử dụng lập trình hướng đối tượng để tổ chức code API của bạn một cách logic và dễ quản lý.

## 1.6. Tài nguyên và Cộng đồng

Tài nguyên và cộng đồng là hai yếu tố quan trọng đối với sự phát triển và thành công của NodeJS.

### 1.6.1. Tài liệu học và tài nguyên tham khảo

NodeJS là một nền tảng JavaScript mã nguồn mở, được sử dụng rộng rãi cho phát triển web và các ứng dụng mạng. Nhờ khả năng xử lý phi đồng bộ, hiệu quả cao và linh hoạt, NodeJS ngày càng trở nên phổ biến và thu hút nhiều người học tập.

Để giúp bạn bắt đầu với NodeJS, tôi đã tổng hợp một số tài liệu học tập và tài nguyên tham khảo hữu ích:

- Tài liệu chính thức:

+ Trang web chính thức của NodeJS: https://nodejs.org/en

* Cung cấp thông tin chi tiết về NodeJS, bao gồm hướng dẫn cài đặt, sử dụng, các tính năng, API, v.v.
* Tài liệu được cập nhật thường xuyên và là nguồn thông tin chính thức cho NodeJS.

+ NodeJS Documentation: https://nodejs.org/en/docs

* Cung cấp tài liệu chi tiết về API của NodeJS, bao gồm các module, class, function, v.v.
* Tài liệu được tổ chức khoa học và dễ hiểu, giúp bạn tìm kiếm thông tin cần thiết một cách nhanh chóng.

+ NodeJS Guides: https://nodejs.org/en/docs

* Cung cấp các hướng dẫn và ví dụ thực tế về cách sử dụng NodeJS cho các tác vụ khác nhau, như xây dựng web server, ứng dụng web, CLI tools, v.v.
* Hướng dẫn được viết rõ ràng, dễ hiểu và phù hợp cho người mới bắt đầu học NodeJS.

- Sách:

+ Getting Started with NodeJS 2nd Edition by Alexander Lindley: Cuốn sách giới thiệu toàn diện về NodeJS, bao gồm các khái niệm cơ bản, API, module, xây dựng ứng dụng web, v.v.

+ NodeJS Design Patterns by Mario Casciaro: Cuốn sách hướng dẫn cách áp dụng các mẫu thiết kế trong lập trình NodeJS để xây dựng ứng dụng hiệu quả, dễ bảo trì và mở rộng.

+ NodeJS in Action by Mike North: Cuốn sách cung cấp hướng dẫn thực tế về cách sử dụng NodeJS cho các tác vụ phát triển phổ biến, như xây dựng API, xử lý dữ liệu, và quản lý socket.

- Khóa học online:

+ NodeJS - The Complete Beginner's Guide by Udemy: Khóa học toàn diện dành cho người mới bắt đầu học NodeJS, bao gồm các khái niệm cơ bản, API, module, xây dựng ứng dụng web, v.v.

+ The Complete JavaScript Course: Master JavaScript & NodeJS by Udemy: Khóa học toàn diện về JavaScript và NodeJS, giúp bạn học lập trình JavaScript từ cơ bản đến nâng cao và áp dụng vào NodeJS để xây dựng ứng dụng web.

+ NodeJS Tutorial for Beginners by freeCodeCamp: Khóa học miễn phí giới thiệu về NodeJS, bao gồm các khái niệm cơ bản, API, xây dựng ứng dụng web đơn giản, v.v.

### 1.6.2. Cộng đồng NodeJS và các diễn đàn trao đổi kiến thức

NodeJS sở hữu cộng đồng lập trình viên đông đảo và nhiệt tình, luôn sẵn sàng hỗ trợ và chia sẻ kiến thức với nhau. Tham gia vào cộng đồng NodeJS là cách tuyệt vời để bạn học hỏi, nâng cao kỹ năng và phát triển bản thân trong lĩnh vực lập trình. Dưới đây là một số diễn đàn và nhóm trao đổi kiến thức NodeJS uy tín mà bạn có thể tham khảo:

- Diễn đàn:

+ NodeJS Forums: https://github.com/NodeBB/NodeBB. Diễn đàn chính thức của NodeJS, nơi bạn có thể thảo luận về bất kỳ chủ đề nào liên quan đến NodeJS, từ các vấn đề kỹ thuật đến chia sẻ kinh nghiệm và tìm kiếm trợ giúp.

+ Stack Overflow: https://stackoverflow.com/. Trang web hỏi đáp lập trình phổ biến, nơi bạn có thể đặt câu hỏi về NodeJS và nhận trợ giúp từ cộng đồng lập trình viên toàn cầu.

+ Reddit r/node: https://www.reddit.com/r/node/. Subreddit dành cho NodeJS trên Reddit, nơi bạn có thể tham gia các cuộc thảo luận, chia sẻ tin tức và cập nhật mới nhất về NodeJS.

- Nhóm Facebook:

+ Cộng đồng NodeJS Việt Nam: Nhóm Facebook dành cho cộng đồng NodeJS Việt Nam, nơi bạn có thể kết nối với các lập trình viên NodeJS Việt Nam, chia sẻ kinh nghiệm và học hỏi lẫn nhau.

+ NodeJS Vietnam Group: Facebook dành cho cộng đồng NodeJS Việt Nam, nơi bạn có thể tham gia các cuộc thảo luận, chia sẻ kiến thức và tìm kiếm trợ giúp về NodeJS.

- Kênh Youtube:

+ NodeJS Tutorials: youtube.com/watch?v=f2EqECiTBL8. Kênh Youtube cung cấp các hướng dẫn và bài học về NodeJS cho người mới bắt đầu và những người có kinh nghiệm.

+ The Net Ninja: youtube.com/channel/UCW5YeuERMmlnqo4oq8vwUpg. Kênh Youtube cung cấp các video hướng dẫn về nhiều ngôn ngữ lập trình, bao gồm NodeJS.

+ Traversy Media: youtube.com/watch?v=hdI2bqOjy3c. Kênh Youtube cung cấp các khóa học và hướng dẫn về nhiều ngôn ngữ lập trình, bao gồm NodeJS.

- Các kênh khác:

+ Meetup: meetup.com/topics/nodejs/. Tham gia các buổi meetup của cộng đồng NodeJS để gặp gỡ các lập trình viên khác, chia sẻ kinh nghiệm và học hỏi những điều mới.

+ Blogs: Theo dõi các blog về NodeJS để cập nhật tin tức, bài viết chia sẻ kinh nghiệm và các mẹo lập trình hữu ích.

+ Sách và tài liệu: Đọc sách và tài liệu về NodeJS để học hỏi kiến thức chuyên sâu và nâng cao kỹ năng lập trình của bạn.

### 1.6.3. Các dự án mã nguồn mở và thư viện phổ biến

- Dự Án Mã Nguồn Mở:

+ ExpressJS: ExpressJS là một framework web NodeJS mạnh mẽ và linh hoạt, được sử dụng rộng rãi để xây dựng các ứng dụng web và API.

+ Socket.IO: Socket.IO là một thư viện cho phép truyền thông thời gian thực giữa client và server thông qua WebSocket và các giao thức khác.

+ Node-RED: Node-RED là một công cụ dựa trên NodeJS cho việc kết nối các thiết bị IoT, dịch vụ mạng và các ứng dụng khác theo cách trực quan và linh hoạt.

+ Puppeteer: Puppeteer là một thư viện NodeJS cho việc kiểm tra và kiểm soát các trình duyệt web tự động thông qua giao diện người dùng.

+ NestJS: NestJS là một framework NodeJS dựa trên ExpressJS, được thiết kế để xây dựng các ứng dụng web và API có cấu trúc, dễ bảo trì và dễ mở rộng.

- Thư Viện Phổ Biến:

+ Lodash: Lodash là một thư viện tiện ích cho JavaScript, cung cấp nhiều chức năng hữu ích cho việc xử lý dữ liệu.

+ Moment.js: Moment.js là một thư viện cho việc xử lý, hiển thị và định dạng thời gian trong JavaScript.

+ Axios: Axios là một thư viện HTTP client cho NodeJS và trình duyệt, giúp thực hiện các yêu cầu HTTP một cách dễ dàng và linh hoạt.

+ Mongoose: Mongoose là một thư viện MongoDB ODM (Object Data Modeling) cho NodeJS, cung cấp một cách dễ dàng để tương tác với cơ sở dữ liệu MongoDB.

+ Jest: Jest là một framework kiểm thử JavaScript phổ biến, được sử dụng rộng rãi trong các dự án NodeJS để viết và chạy các bài kiểm thử.

## 1.7. Tổng kết và các xu hướng mới, tiềm năng phát triển của NodeJS

### 1.7.1. Các xu hướng mới và tiềm năng phát triển của NodeJS

NodeJS tiếp tục là một trong những nền tảng phát triển phần mềm phổ biến và được ưa chuộng nhất hiện nay. Nhờ khả năng xử lý phi đồng bộ, hiệu suất cao và cộng đồng phát triển mạnh mẽ, NodeJS đã được sử dụng để xây dựng nhiều ứng dụng web và server thành công. Vậy, xu hướng phát triển của NodeJS trong tương lai như thế nào?

Dưới đây là một số xu hướng mới và tiềm năng phát triển của NodeJS trong năm 2024 và hơn thế nữa:

- Sử dụng TypeScript ngày càng phổ biến: TypeScript là một siêu tập hợp của JavaScript, bổ sung thêm hệ thống kiểu tĩnh cho ngôn ngữ. Việc sử dụng TypeScript giúp cải thiện độ tin cậy, bảo trì và khả năng đọc code NodeJS, đồng thời giảm thiểu lỗi runtime. Nhờ những ưu điểm này, TypeScript đang được áp dụng ngày càng rộng rãi trong các dự án NodeJS lớn.

- Nhu cầu về các framework full-stack gia tăng: Các framework full-stack như Next.js, Nest.js, và Strapi đang ngày càng phổ biến trong cộng đồng NodeJS. Các framework này cung cấp giải pháp tích hợp cho cả phần front-end và back-end, giúp đơn giản hóa việc phát triển ứng dụng web toàn diện.

- Serverless functions ngày càng được ưa chuộng: Với sự ra đời và phát triển của các nền tảng serverless như AWS Lambda và Azure Functions, NodeJS được ứng dụng rộng rãi để xây dựng các serverless functions. Serverless functions giúp giảm thiểu chi phí vận hành và bảo trì cơ sở hạ tầng, đồng thời mang lại khả năng mở rộng linh hoạt cho ứng dụng.

- Tiếp tục phát triển các ứng dụng thời gian thực: NodeJS vẫn là lựa chọn hàng đầu cho phát triển các ứng dụng thời gian thực như chat, game online, và bảng tin tức. Nhờ khả năng xử lý phi đồng bộ và WebSocket, NodeJS đảm bảo hiệu suất cao và độ trễ thấp cho các ứng dụng này.

- Tăng cường áp dụng trí tuệ nhân tạo (AI) và học máy (ML): NodeJS được sử dụng để xây dựng các mô hình AI và ML, xử lý dữ liệu lớn và cung cấp các dịch vụ thông minh cho ứng dụng. Sự kết hợp giữa NodeJS, AI và ML mở ra nhiều tiềm năng phát triển mới cho các lĩnh vực như chatbot, phân tích dữ liệu, và tự động hóa.

- NodeJS trên các thiết bị IoT: NodeJS đang dần được sử dụng để phát triển các ứng dụng cho Internet vạn vật (IoT). Nhờ khả năng xử lý phi đồng bộ và hiệu quả năng lượng, NodeJS phù hợp cho các thiết bị IoT có tài nguyên hạn chế.

- Cộng đồng NodeJS ngày càng phát triển và mạnh mẽ: Cộng đồng NodeJS là một trong những cộng đồng phát triển phần mềm lớn nhất và tích cực nhất hiện nay. Với sự đóng góp của các lập trình viên trên toàn thế giới, NodeJS liên tục được cải tiến và cập nhật các tính năng mới, đảm bảo sự phát triển bền vững cho nền tảng này.

### 1.7.2. Tổng kết

NodeJS đã trở thành một trong những công nghệ phát triển phần mềm quan trọng nhất trong thập kỷ qua, và tiếp tục là một lực lượng mạnh mẽ trong cộng đồng phát triển phần mềm. Với sự linh hoạt, hiệu quả và khả năng mở rộng của mình, NodeJS đã thúc đẩy sự phát triển của các ứng dụng web, dịch vụ mạng và ứng dụng thời gian thực trên toàn thế giới.

Một trong những điểm mạnh của NodeJS là khả năng xử lý đồng thời và sự kiện trong một môi trường không đồng bộ, giúp tăng cường hiệu suất và tốc độ cho các ứng dụng của mình. Điều này làm cho NodeJS trở thành một lựa chọn lý tưởng cho việc xây dựng các hệ thống có khả năng mở rộng và linh hoạt, đặc biệt là trong các ứng dụng web thời gian thực và microservices.

NodeJS cũng đóng vai trò quan trọng trong việc khuyến khích sự đổi mới và tiến bộ trong cộng đồng phát triển phần mềm. Các công nghệ phụ trợ như TypeScript, GraphQL và Serverless đã mở ra những cơ hội mới cho NodeJS, giúp tạo ra các ứng dụng mạng phức tạp và hiệu quả hơn.

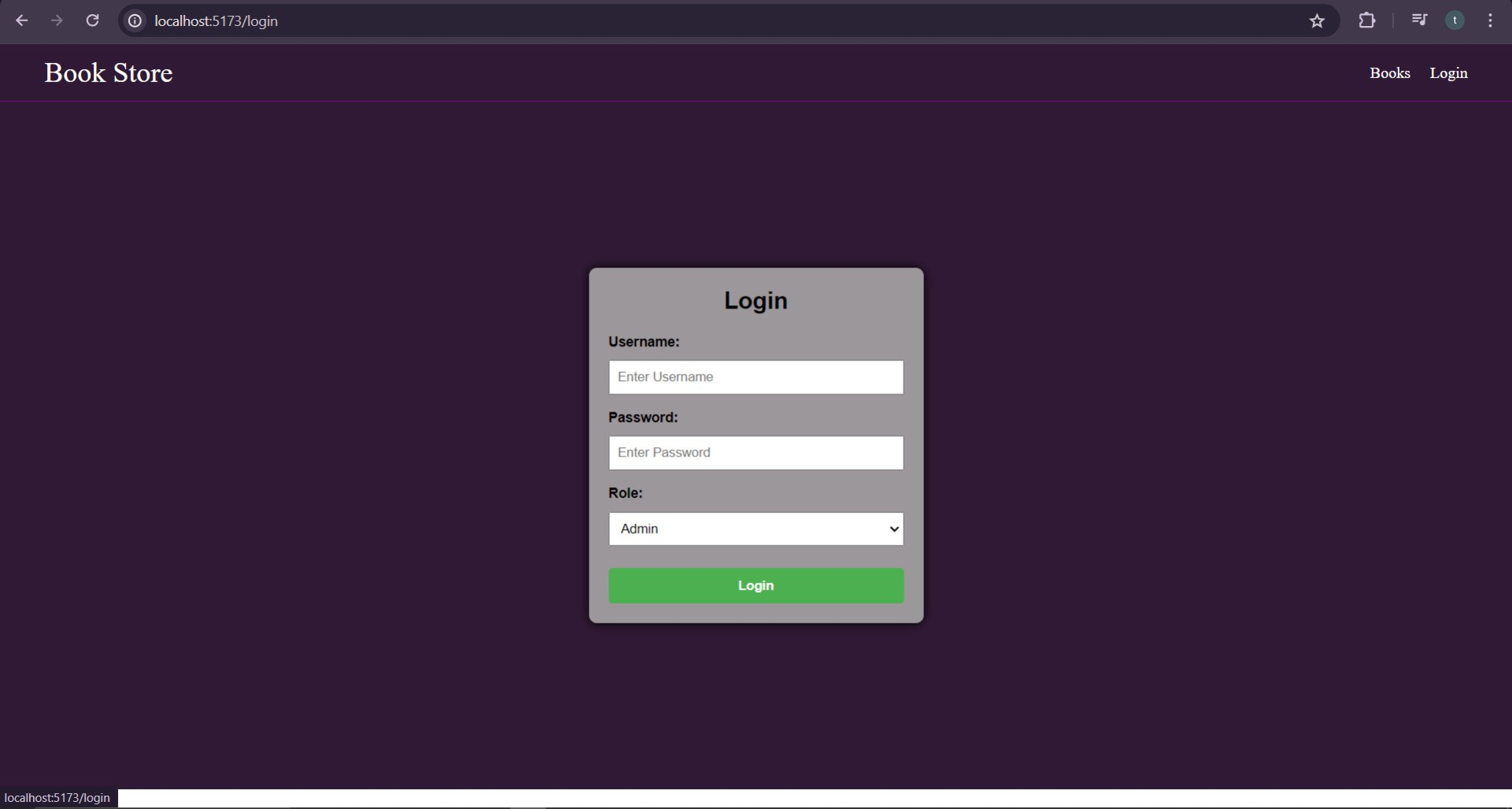
Tuy nhiên, NodeJS không chỉ là về công nghệ. Nó còn là về cộng đồng - một cộng đồng đam mê và sáng tạo, luôn sẵn sàng chia sẻ kiến thức và kinh nghiệm, giúp mọi người học hỏi và phát triển. Sức mạnh của cộng đồng này đã đóng góp quan trọng vào sự thành công và phổ biến của NodeJS.

# CHƯƠNG 2: Website giới thiệu, bán sách trực tuyến

## 2.1. Giao diện trang chủ

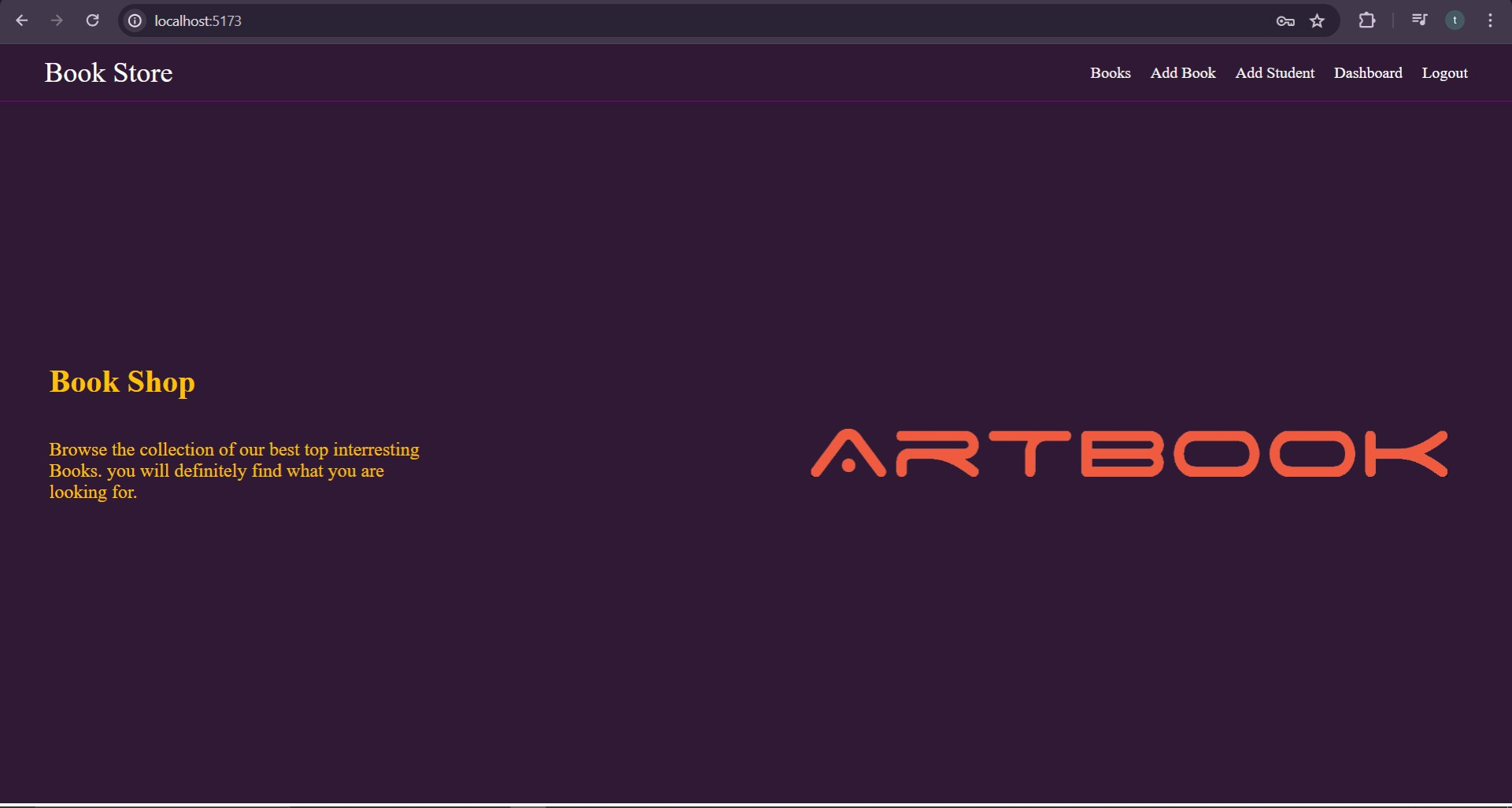


## 2.2. Giao diện đăng nhập

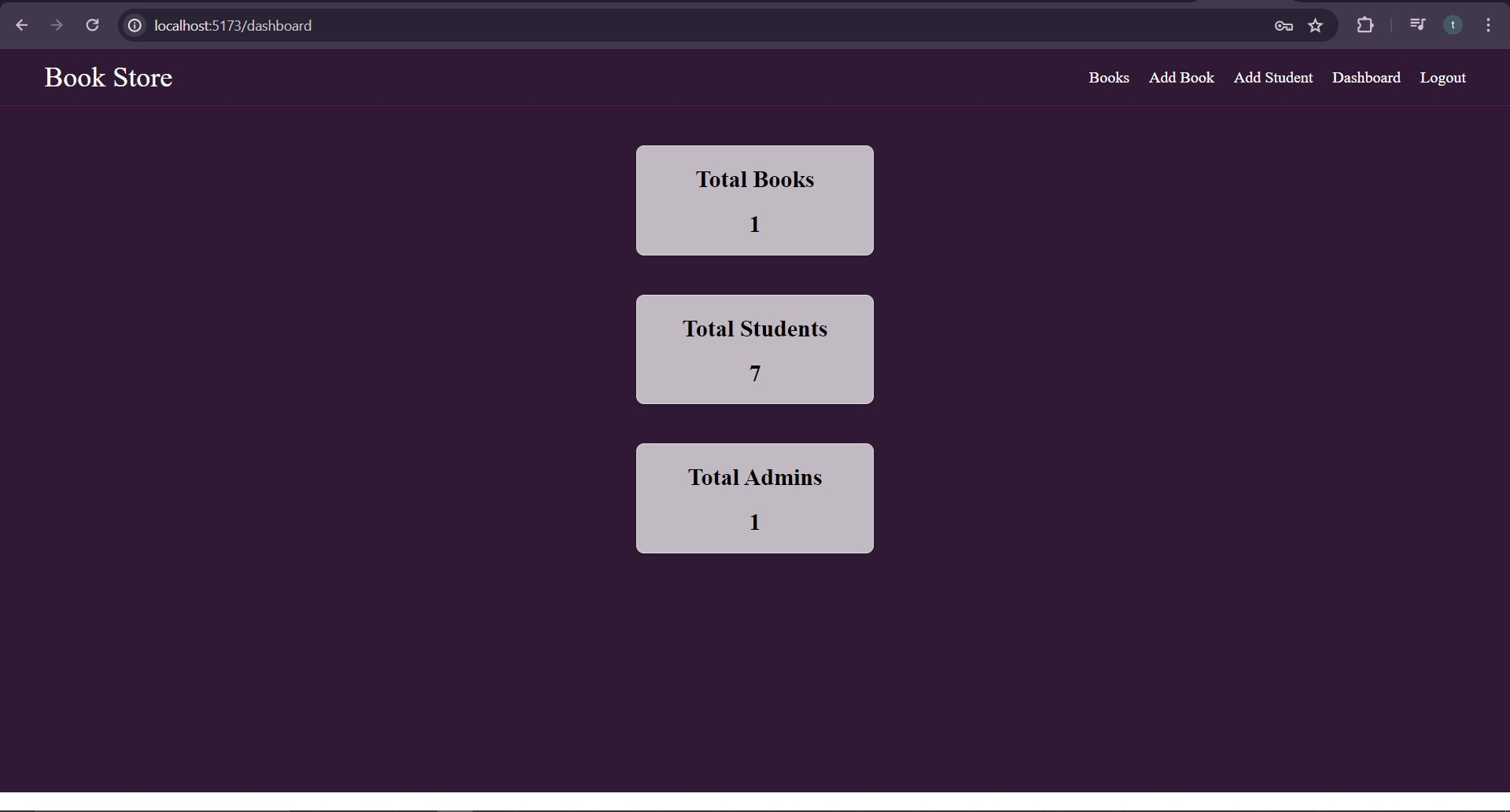


## 2.3. Giao diện Admin

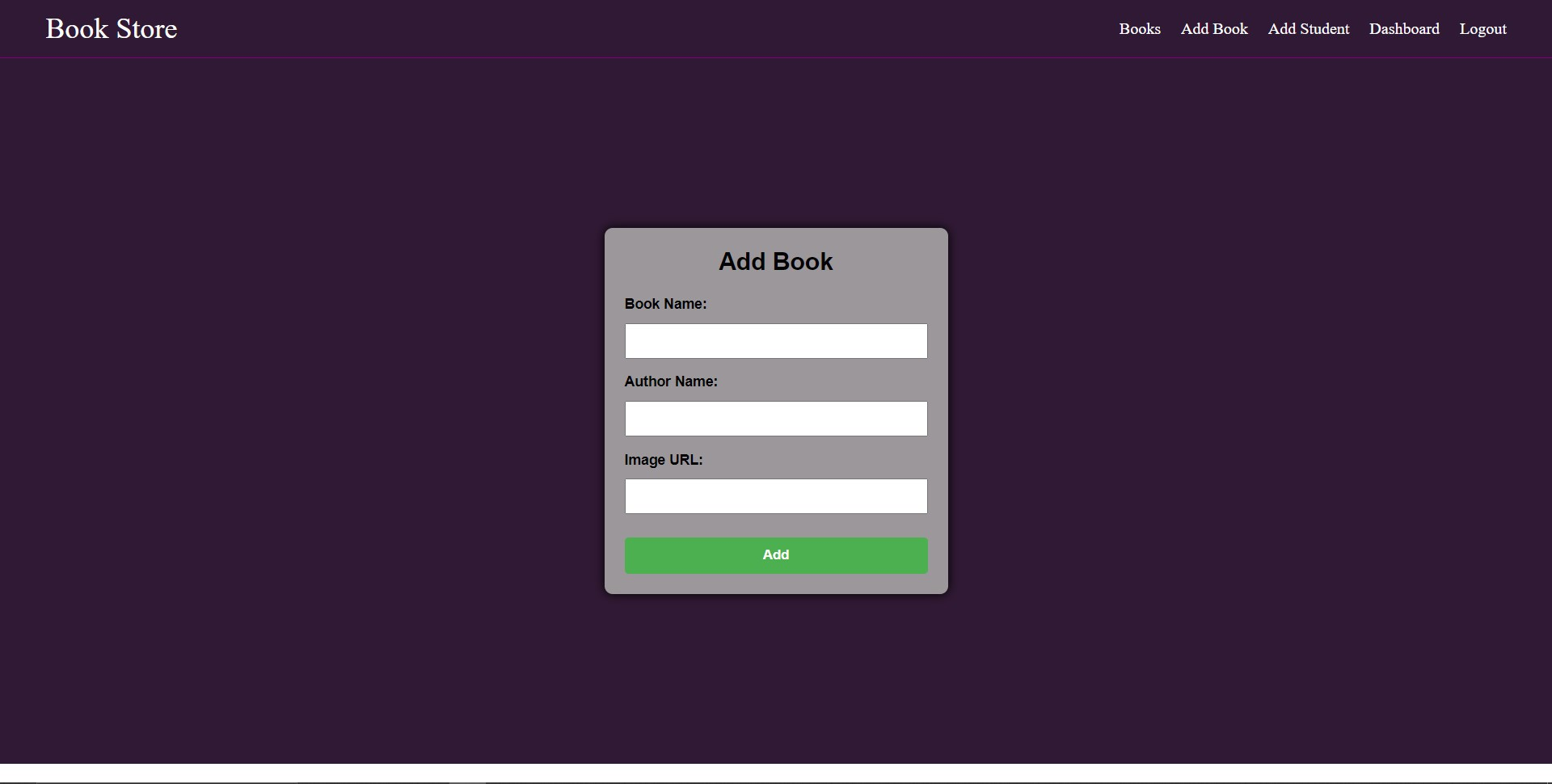
Giao diện trang chủ Admin:

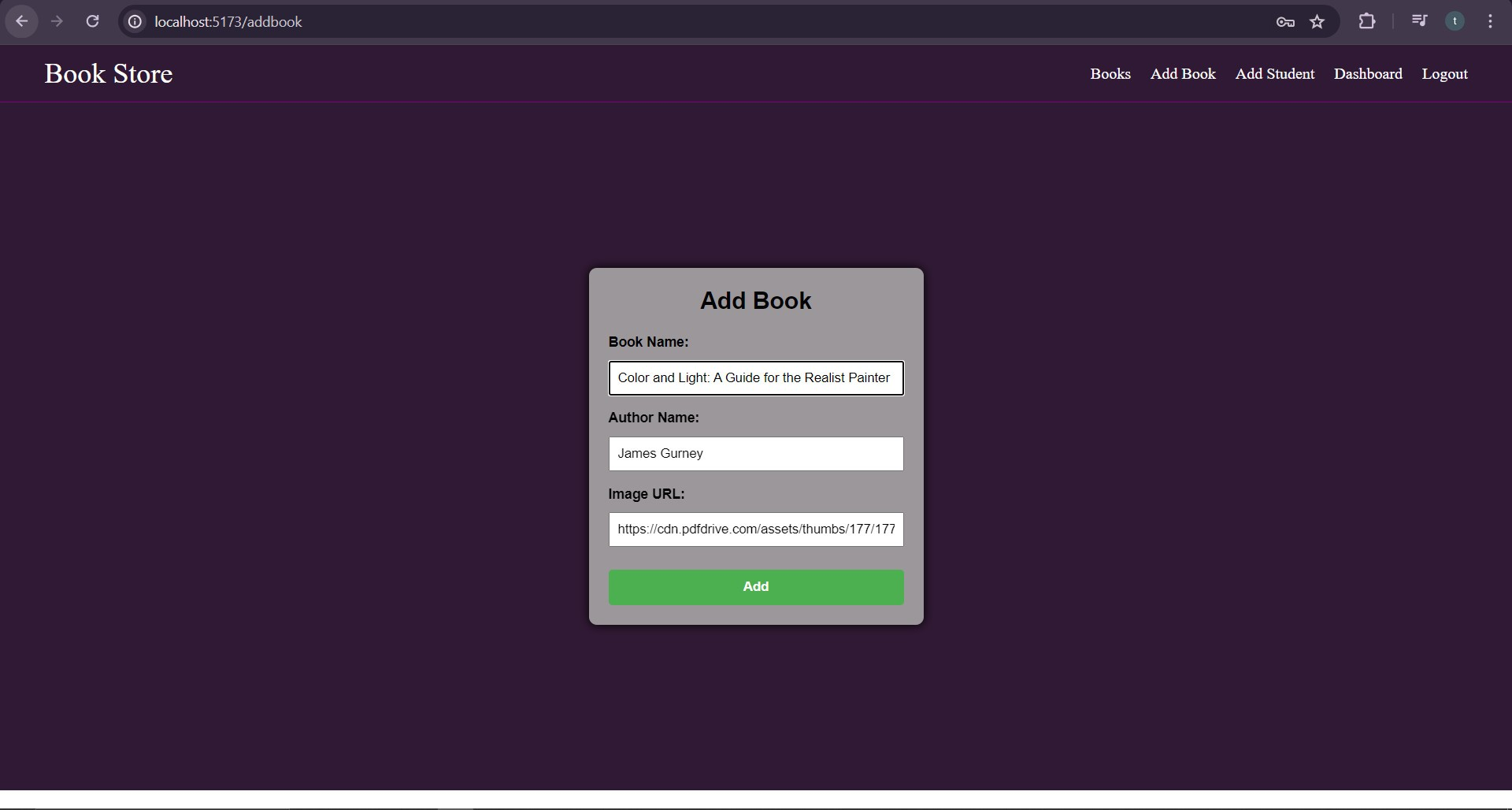


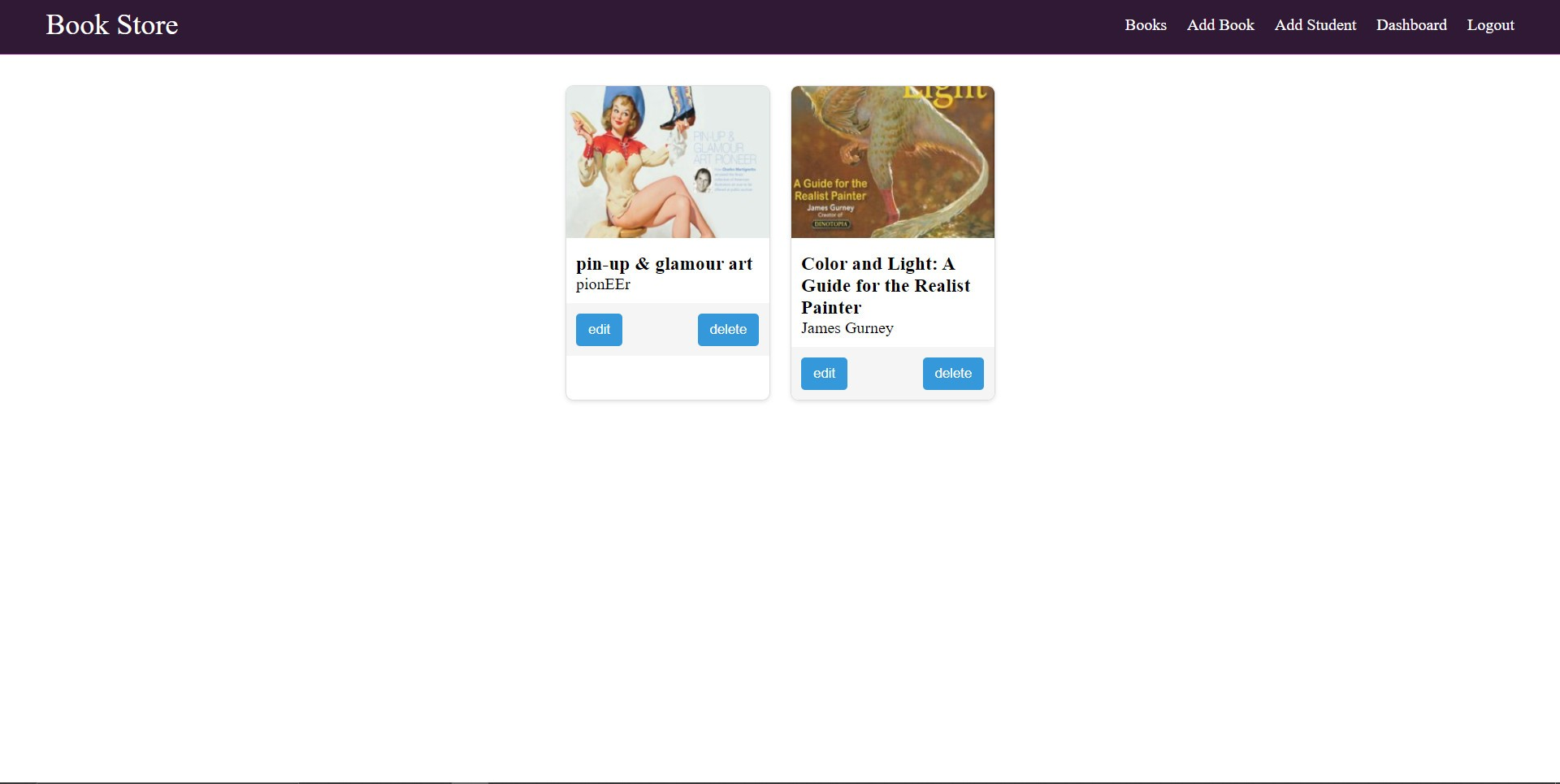
Giao diện Dashboard Admin:



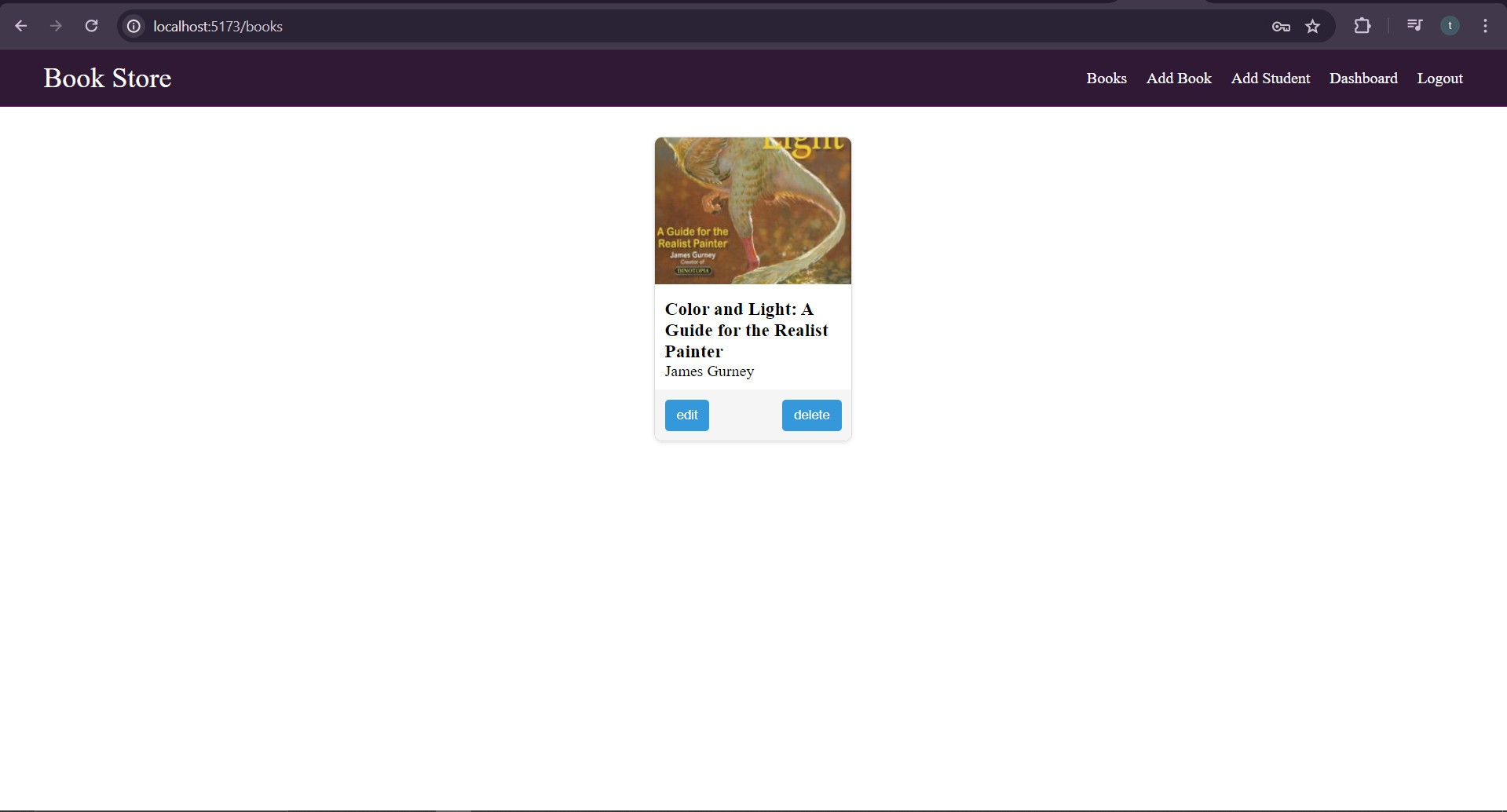
Chức năng thêm sách:

****

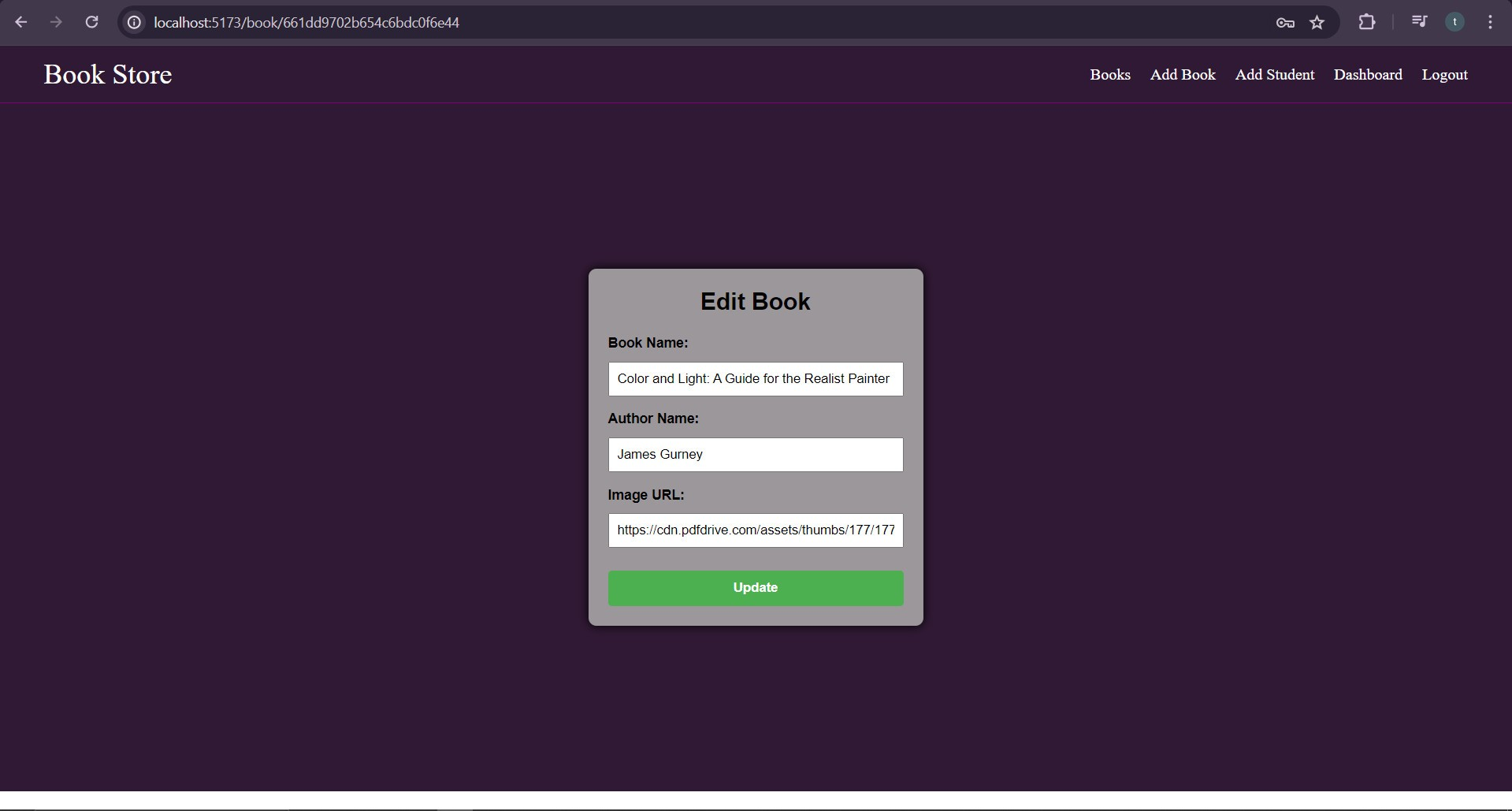
****

****

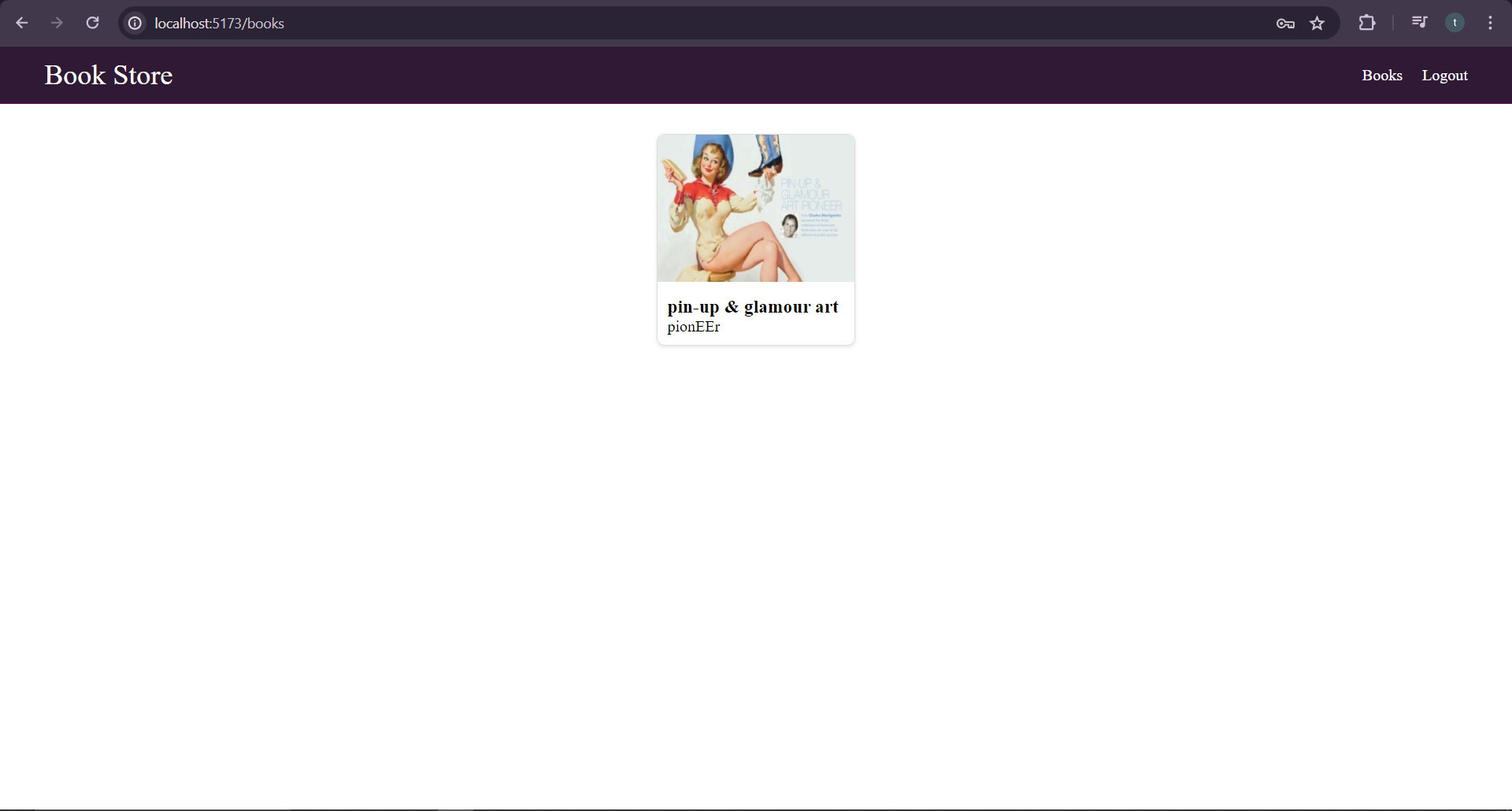
Chức năng xóa sách:

****

Chức năng chỉnh sửa sách:

****

## 2.4. Giao diện sinh viên



# TÀI LIỆU THAM KHẢO

1. chat.openai.com

2. gemini.google.com

3. nodejs.org/en

4. nodejs.org/en/docs

5. nodejs.org/en/about/get-involved