

TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN HỌC
CƠ SỞ DỮ LIỆU TIỀN TIẾN

XÂY DỰNG HỆ THỐNG
E-LIBRARY
PHÂN TÁN NHIỀU CƠ SỞ

Giảng viên hướng dẫn:

TS. Nguyễn Duy Hải

Nhóm thực hiện:

Nhóm 10 - Cao học K35

Thành viên:

- Trương Tuấn Nghĩa
- Phạm Mạnh Thắng
- Lưu Anh Tú

Công nghệ: MongoDB Sharded Cluster | PHP 8.x | Docker | Chart.js

Hà Nội, tháng 01 năm 2026

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến **TS. Nguyễn Duy Hải** - giảng viên hướng dẫn môn Cơ sở dữ liệu tiên tiến. Thầy đã tận tình giảng dạy, truyền đạt những kiến thức quý báu về hệ thống cơ sở dữ liệu phân tán và NoSQL, đồng thời hướng dẫn nhóm trong suốt quá trình thực hiện đề tài.

Chúng em xin cảm ơn **Khoa Công nghệ Thông tin - Trường Đại học Sư phạm Hà Nội** đã tạo điều kiện thuận lợi về cơ sở vật chất và môi trường học tập để nhóm có thể hoàn thành tốt bài báo cáo này.

Chúng em cũng xin gửi lời cảm ơn đến các bạn học viên lớp Cao học K35 đã chia sẻ kinh nghiệm và hỗ trợ nhóm trong quá trình học tập và nghiên cứu.

Do thời gian và kiến thức còn hạn chế, bài báo cáo không tránh khỏi những thiếu sót. Nhóm chúng em rất mong nhận được sự góp ý, chỉ bảo của Thầy và các bạn để bài báo cáo được hoàn thiện hơn.

Xin chân thành cảm ơn!

Hà Nội, tháng 01 năm 2026

Nhóm thực hiện

Trương Tuấn Nghĩa
Phạm Mạnh Thắng
Lưu Anh Tú

LỜI CAM ĐOAN

Chúng tôi xin cam đoan:

- Đề tài báo cáo môn học "**Xây dựng hệ thống e-Library phân tán nhiều cơ sở**" là công trình nghiên cứu và thực hiện của nhóm chúng tôi dưới sự hướng dẫn của **TS. Nguyễn Duy Hải**.
- Các số liệu, kết quả benchmark và kịch bản kiểm thử trong báo cáo là trung thực và được thực hiện trên hệ thống thực tế.
- Mã nguồn được phát triển bởi nhóm, có tham khảo và sử dụng các thư viện mã nguồn mở theo đúng quy định (MongoDB PHP Library, Firebase PHP-JWT, Chart.js).
- Các tài liệu tham khảo được trích dẫn đầy đủ theo quy định.

Chúng tôi xin chịu hoàn toàn trách nhiệm về lời cam đoan này.

Hà Nội, ngày tháng 01 năm 2026

Nhóm thực hiện

**Trương Tuấn Nghĩa
Phạm Mạnh Thắng**

Lưu Anh Tú

Mục lục

| | |
|--|----------|
| LỜI CẢM ƠN | i |
| LỜI CAM ĐOAN | ii |
| DANH MỤC HÌNH | vi |
| DANH MỤC BẢNG | vii |
| DANH MỤC MÃ NGUỒN | viii |
| 1 TỔNG QUAN VỀ HỆ THỐNG | 1 |
| 1.1 Giới thiệu bài toán và mục tiêu hệ thống | 1 |
| 1.1.1 Bối cảnh và đặt vấn đề | 1 |
| 1.1.2 Mục tiêu của đề tài | 1 |
| 1.2 Tổng quan về hệ thống e-Library | 2 |
| 1.2.1 Mô hình hệ thống phân tán | 2 |
| 1.2.2 Dữ liệu thực tế trong hệ thống | 2 |
| 1.2.3 Kiến trúc tổng quan | 3 |
| 1.3 Một số khái niệm và nghiệp vụ liên quan | 3 |
| 1.3.1 Khái niệm về các đối tượng | 3 |
| 1.3.2 Các quy trình nghiệp vụ | 4 |
| 1.4 Một số công nghệ được áp dụng | 4 |
| 1.4.1 PHP 8.4 và MongoDB Driver | 4 |
| 1.4.2 MongoDB và MongoDB Compass | 5 |
| 1.4.3 Docker và Docker Compose | 6 |
| 1.4.4 JWT (JSON Web Token) | 6 |
| 1.4.5 Chart.js | 7 |
| 2 PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG | 8 |
| 2.1 Xác định các yêu cầu | 8 |
| 2.1.1 Yêu cầu phi chức năng | 8 |
| 2.1.2 Yêu cầu chức năng | 9 |
| 2.2 Ca sử dụng - Use Case | 9 |
| 2.2.1 Danh sách các tác nhân | 9 |
| 2.2.2 Biểu đồ Use Case tổng quát | 10 |
| 2.3 Mô hình cấu trúc | 10 |
| 2.3.1 Danh sách các lớp đối tượng | 10 |
| 2.3.2 Biểu đồ lớp | 11 |

| | | |
|----------|---|-----------|
| 2.4 | Thiết kế CSDL | 11 |
| 2.4.1 | Xác định các collection | 11 |
| 2.4.2 | Thiết kế bảng dữ liệu vật lý | 11 |
| 2.4.3 | Thiết kế mô hình phân tán | 13 |
| 2.4.4 | Thiết kế tìm kiếm và tối ưu truy vấn | 13 |
| 2.5 | Thiết kế giao diện | 14 |
| 3 | CÀI ĐẶT VÀ ĐÁNH GIÁ HỆ THỐNG | 15 |
| 3.1 | Hướng dẫn cài đặt và khởi động hệ thống | 15 |
| 3.1.1 | Yêu cầu hệ thống | 15 |
| 3.1.2 | Khởi động hệ thống với script tự động | 15 |
| 3.2 | Các công cụ sử dụng cài đặt hệ thống | 16 |
| 3.2.1 | MongoDB 4.4 và MongoDB Compass | 16 |
| 3.2.2 | PHP 8.4 và MongoDB Driver | 16 |
| 3.2.3 | Docker Compose cho MongoDB Replica Set | 17 |
| 3.3 | Một số giao diện chính của hệ thống | 18 |
| 3.3.1 | Giao diện đăng nhập | 18 |
| 3.3.2 | Dashboard thống kê (Admin) | 19 |
| 3.3.3 | Quản lý sách (Admin) | 20 |
| 3.3.4 | Danh sách sách (Customer) | 21 |
| 3.3.5 | Giỏ hàng mượn sách | 22 |
| 3.3.6 | Docker Containers | 22 |
| 3.3.7 | MongoDB Compass | 23 |
| 3.4 | Triển khai Aggregation Pipeline | 23 |
| 3.4.1 | Tổng quan API Statistics | 23 |
| 3.4.2 | Endpoint books_by_location | 23 |
| 3.4.3 | Endpoint user_details với \$lookup JOIN | 24 |
| 3.4.4 | Endpoint book_group_stats với \$facet và \$bucket | 25 |
| 3.5 | Triển khai Map-Reduce | 26 |
| 3.5.1 | Tổng quan API Map-Reduce | 26 |
| 3.5.2 | Map-Reduce: borrow_stats | 26 |
| 3.6 | Kiểm thử hệ thống | 27 |
| 3.6.1 | Kịch bản 1: Kiểm thử hiển thị dữ liệu | 27 |
| 3.6.2 | Kịch bản 2: Kiểm thử ghi và đồng bộ | 27 |
| 3.6.3 | Kịch bản 3: Kiểm thử Failover | 28 |
| 3.6.4 | Kịch bản 4: Benchmark hiệu năng | 28 |
| 3.7 | Dánh giá hệ thống | 30 |
| 3.7.1 | Ưu điểm | 30 |
| 3.7.2 | Nhược điểm và Hạn chế | 31 |
| 3.7.3 | So sánh với các hệ thống khác | 31 |
| 4 | KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN | 32 |
| 4.1 | Kết luận | 32 |
| 4.1.1 | Những kết quả đạt được | 32 |
| 4.1.2 | Những điểm còn hạn chế | 34 |
| 4.1.3 | Kiến thức và kỹ năng đạt được | 35 |
| 4.2 | Phương hướng phát triển | 35 |
| 4.2.1 | Cải tiến ngắn hạn | 35 |

| | | |
|---------------------------|--|-----------|
| 4.2.2 | Phát triển trung hạn | 36 |
| 4.2.3 | Phát triển dài hạn | 36 |
| TÀI LIỆU THAM KHẢO | | 38 |
| A PHỤ LỤC | | 40 |
| A.1 | Bảng ký hiệu và chữ viết tắt | 40 |
| A.2 | Mã nguồn quan trọng | 41 |
| A.2.1 | JWTHelper.php - Xác thực JWT | 41 |
| A.2.2 | Connection.php - Kết nối MongoDB Replica Set | 42 |
| A.2.3 | docker-compose.yml - Cấu hình Docker | 43 |
| A.2.4 | start_system.sh - Script khởi động hệ thống | 44 |
| A.3 | Cấu trúc thư mục dự án | 44 |
| A.4 | Thông tin đăng nhập hệ thống | 45 |

Danh sách hình vẽ

| | | |
|-----|---|----|
| 1.1 | Kiến trúc tổng quan hệ thống e-Library phân tán | 3 |
| 2.1 | Biểu đồ Use Case tổng quát | 10 |
| 2.2 | Biểu đồ lớp hệ thống e-Library | 11 |
| 2.3 | Kiến trúc MongoDB Replica Set | 13 |
| 3.1 | Giao diện đăng nhập hệ thống | 18 |
| 3.2 | Dashboard thống kê với biểu đồ Chart.js | 19 |
| 3.3 | Giao diện CRUD quản lý sách | 20 |
| 3.4 | Danh sách sách cho khách hàng | 21 |
| 3.5 | Giao diện giỏ hàng mượn sách | 22 |
| 3.6 | Docker Desktop hiển thị MongoDB containers | 22 |
| 3.7 | MongoDB Compass hiển thị collection books | 23 |
| 3.8 | Kết quả benchmark trong Terminal | 29 |

Danh sách bảng

| | | |
|-----|--|----|
| 1.1 | Các node trong hệ thống e-Library | 2 |
| 1.2 | Thống kê dữ liệu theo từng chi nhánh | 2 |
| 2.1 | Danh sách yêu cầu chức năng | 9 |
| 2.2 | Danh sách tác nhân trong hệ thống | 9 |
| 2.3 | Danh sách collection trong MongoDB | 11 |
| 2.4 | Danh sách Index trong collection books | 13 |
| 3.1 | Yêu cầu phần mềm | 15 |
| 3.2 | Danh sách Aggregation Pipeline endpoints | 23 |
| 3.3 | Danh sách Map-Reduce operations | 26 |
| 3.4 | Kết quả kiểm thử hiển thị dữ liệu | 27 |
| 3.5 | Kết quả benchmark hiệu năng (REAL DATA) | 30 |
| 3.6 | So sánh MongoDB với các hệ thống phân tán khác | 31 |
| 4.1 | Tổng hợp kết quả benchmark | 34 |
| A.1 | Bảng từ viết tắt và ký hiệu | 40 |
| A.2 | Thông tin đăng nhập các node | 45 |
| A.3 | Thông tin kết nối MongoDB | 46 |

Listings

| | | |
|-----|---|----|
| 1.1 | Connection.php - Cấu hình kết nối MongoDB | 5 |
| 1.2 | docker-compose.yml - MongoDB Replica Set | 6 |
| 1.3 | JWTHelper.php - Tạo JWT Token | 7 |
| 2.1 | Schema collection users | 12 |
| 2.2 | Schema collection books | 12 |
| 2.3 | Schema collection orders | 12 |
| 2.4 | Tìm kiếm với TEXT index | 13 |
| 3.1 | start_system.sh - Khởi động hệ thống | 15 |
| 3.2 | Connection.php - Dài đủ 3 mode kết nối | 16 |
| 3.3 | docker-compose.yml - MongoDB Replica Set đầy đủ | 17 |
| 3.4 | statistics.php - books_by_location với 4 stages | 23 |
| 3.5 | statistics.php - user_details với \$lookup | 24 |
| 3.6 | statistics.php - Multi-faceted statistics | 25 |
| 3.7 | mapreduce.php - Borrowing statistics | 26 |
| 3.8 | Script kiểm thử Failover | 28 |
| A.1 | JWTHelper.php - Class xử lý JWT authentication | 41 |
| A.2 | Connection.php - Kết nối với MongoDB Replica Set | 42 |
| A.3 | docker-compose.yml - Cấu hình MongoDB Replica Set | 43 |
| A.4 | start_system.sh - Script khởi động toàn bộ hệ thống | 44 |
| A.5 | Cấu trúc thư mục của dự án | 44 |

Chương 1

TỔNG QUAN VỀ HỆ THỐNG

1.1 Giới thiệu bài toán và mục tiêu hệ thống

1.1.1 Bối cảnh và đặt vấn đề

Trong bối cảnh số hóa và phát triển công nghệ thông tin, các thư viện truyền thống đang dần chuyển đổi sang mô hình thư viện điện tử (e-Library). Đối với các hệ thống thư viện có nhiều chi nhánh phân bố ở các vị trí địa lý khác nhau, việc quản lý tập trung và đồng bộ dữ liệu trở thành một thách thức lớn.

Các vấn đề cần giải quyết:

- **Tính phân tán:** Dữ liệu cần được lưu trữ và truy cập từ nhiều vị trí địa lý khác nhau
- **Tính sẵn sàng cao:** Hệ thống phải hoạt động liên tục, có khả năng tự phục hồi khi gặp sự cố
- **Hiệu năng:** Đảm bảo thời gian phản hồi nhanh cho các truy vấn dữ liệu
- **Khả năng mở rộng:** Hệ thống có thể scale theo chiều ngang khi lượng dữ liệu tăng

1.1.2 Mục tiêu của đề tài

Đề tài "Xây dựng hệ thống E-Library Phân tán nhiều cơ sở" hướng đến các mục tiêu sau:

1. Thiết kế và cài đặt hệ thống phân tán:

- Kiến trúc 4 node: 1 Central Hub + 3 chi nhánh (Hà Nội, Đà Nẵng, TP.HCM)
- Sử dụng MongoDB Replica Set cho high availability
- Hỗ trợ Zone Sharding theo vùng địa lý

2. Triển khai các kỹ thuật NoSQL nâng cao:

- Aggregation Pipeline với 10+ operators
- Map-Reduce cho phân tích dữ liệu phức tạp
- Full-text Search với TEXT index

3. Đảm bảo bảo mật và phân quyền:

- JWT (JSON Web Token) authentication
- Role-based Access Control (RBAC)
- Password hashing với bcrypt

4. Xây dựng giao diện quản trị:

- Dashboard thống kê với Chart.js
- CRUD đầy đủ cho sách, người dùng, đơn hàng
- Responsive design với Bootstrap 5

1.2 Tổng quan về hệ thống e-Library

1.2.1 Mô hình hệ thống phân tán

Hệ thống e-Library được thiết kế theo mô hình phân tán với 4 node độc lập, mỗi node có database riêng nhưng được đồng bộ qua MongoDB Replica Set:

Bảng 1.1: Các node trong hệ thống e-Library

| STT | Node | Database | Port | Vai trò |
|-----|----------------|------------------|------|----------------------|
| 1 | Central Hub | Nhasach | 8001 | Trung tâm quản lý |
| 2 | Branch Hà Nội | NhasachHaNoi | 8002 | Chi nhánh miền Bắc |
| 3 | Branch Đà Nẵng | NhasachDaNang | 8003 | Chi nhánh miền Trung |
| 4 | Branch TP.HCM | NhasachHoChiMinh | 8004 | Chi nhánh miền Nam |

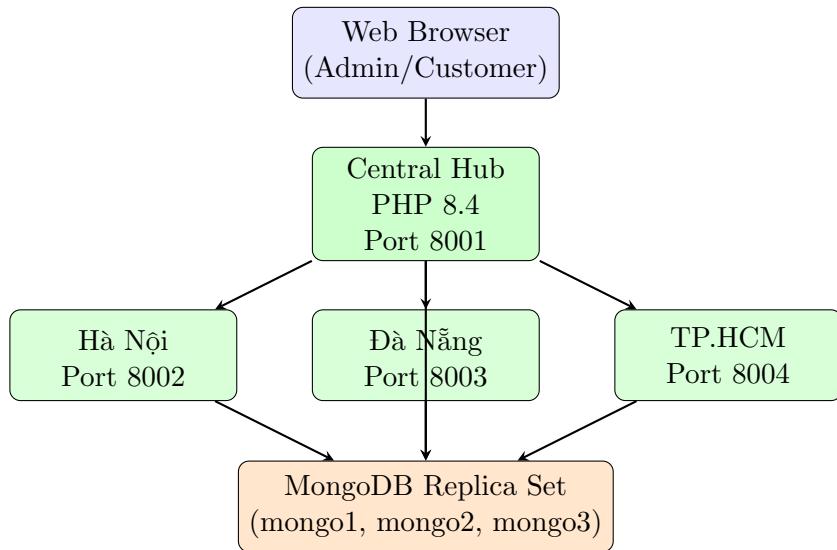
1.2.2 Dữ liệu thực tế trong hệ thống

Tính đến thời điểm hoàn thành đề tài, hệ thống có các số liệu sau:

Bảng 1.2: Thống kê dữ liệu theo từng chi nhánh

| Chi nhánh | Số sách | Số users | Số orders |
|-----------------------|------------|-----------|-----------|
| Central Hub (Nhasach) | 509 | 2 | 0 |
| Chi nhánh Hà Nội | 162 | 13 | 46 |
| Chi nhánh Đà Nẵng | 127 | 12 | 0 |
| Chi nhánh TP.HCM | 111 | 11 | 0 |
| Tổng cộng | 909 | 38 | 46 |

1.2.3 Kiến trúc tổng quan



Hình 1.1: Kiến trúc tổng quan hệ thống e-Library phân tán

1.3 Một số khái niệm và nghiệp vụ liên quan

1.3.1 Khái niệm về các đối tượng

1. Quản trị viên hệ thống (Admin):
 - Toàn quyền quản lý hệ thống tại Central Hub
 - Quản lý sách, người dùng, đơn hàng
 - Xem Dashboard thống kê
 - Đồng bộ dữ liệu giữa các chi nhánh
2. Quản trị viên chi nhánh (Branch Admin):
 - Quản lý sách tại chi nhánh được phân công
 - Username: adminHN, adminDN, adminHCM
 - Không干涉 vào chi nhánh khác
3. Khách hàng (Customer):
 - Đăng ký tài khoản tại chi nhánh
 - Tìm kiếm, xem thông tin sách
 - Mượn sách qua giỏ hàng
 - Xem lịch sử mượn sách

1.3.2 Các quy trình nghiệp vụ

1. Quy trình đăng ký/đăng nhập:

- Khách hàng đăng ký với username, password, email
- Mật khẩu được hash bằng bcrypt (cost factor 12)
- Đăng nhập tạo JWT token với thời hạn 24 giờ
- Token được lưu trong session hoặc Authorization header

2. Quy trình tìm kiếm sách:

- Hỗ trợ Full-text Search với TEXT index
- Tìm theo tên sách, thể loại, tác giả
- Kết quả được sắp xếp theo relevance score
- Hỗ trợ phân trang với 20 sách/trang

3. Quy trình mượn sách:

- Khách hàng thêm sách vào giỏ hàng
- Chọn số ngày mượn, hệ thống tính tiền
- Xác nhận đơn mượn, trừ số dư
- Trạng thái đơn: pending → paid → success → returned

4. Quy trình đồng bộ dữ liệu:

- Chi nhánh gửi dữ liệu về Central Hub qua REST API
- Sử dụng script sync_data.sh để đồng bộ
- Dữ liệu customers và orders được tổng hợp tại Central

1.4 Một số công nghệ được áp dụng

1.4.1 PHP 8.4 và MongoDB Driver

PHP (Hypertext Preprocessor) là ngôn ngữ lập trình kịch bản phía server, được sử dụng rộng rãi trong phát triển ứng dụng web. Trong đề tài này, nhóm sử dụng **PHP 8.4** kết hợp với thư viện `mongodb/mongodb v2.1`.

Đặc điểm của `mongodb/mongodb library`:

- Hỗ trợ đầy đủ CRUD operations với type-safe API
- Aggregation Pipeline Builder cho truy vấn phức tạp
- Hỗ trợ BSON types (ObjectId, UTCDateTime, Javascript)
- Connection pooling tự động
- Read/Write Concern configuration

Cấu hình kết nối trong `Connection.php`:

```

1 <?php
2 require 'vendor/autoload.php';
3 use MongoDB\Client;
4
5 $MODE = 'sharded'; // Options: standalone, replicaset, sharded
6 $Database = "Nhasach";
7
8 switch ($MODE) {
9     case 'sharded':
10         $conn = new Client("mongodb://localhost:27017", [
11             'readPreference' => 'primaryPreferred',
12             'w' => 'majority',
13             'journal' => true
14         ]);
15         break;
16     case 'replicaset':
17         $conn = new Client(
18             "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?"
19             "replicaSet=rs0",
20             ['readPreference' => 'primaryPreferred', 'w' => 'majority'
21         ]
22         );
23         break;
24     default:
25         $conn = new Client("mongodb://localhost:27017");
26 }
27 $db = $conn->$Database;

```

Listing 1.1: Connection.php - Cấu hình kết nối MongoDB

1.4.2 MongoDB và MongoDB Compass

MongoDB là hệ quản trị cơ sở dữ liệu NoSQL hướng tài liệu (document-oriented), lưu trữ dữ liệu dưới dạng BSON (Binary JSON). Phiên bản sử dụng: **MongoDB 4.4** (Docker image).

Các tính năng MongoDB được sử dụng trong đề tài:

- **Replica Set:** 3 nodes (mongo1, mongo2, mongo3) với automatic failover
- **Aggregation Pipeline:** 10+ operators (\$match, \$group, \$sort, \$lookup, \$facet, \$bucket...)
- **Map-Reduce:** 5 operations cho phân tích phức tạp
- **TEXT Index:** Full-text search tiếng Việt
- **Compound Index:** Tối ưu shard-aware queries

MongoDB Compass là công cụ GUI chính thức, hỗ trợ:

- Trực quan hóa dữ liệu và cấu trúc schema
- Aggregation Pipeline Builder với drag-and-drop
- Explain Plan để phân tích query performance
- Real-time server statistics

1.4.3 Docker và Docker Compose

Docker là nền tảng container hóa cho phép đóng gói ứng dụng cùng dependencies. **Docker Compose** cho phép định nghĩa multi-container applications.

Vai trò trong đề tài:

- Khởi tạo MongoDB Replica Set với 3 containers
- Mạng nội bộ mongo-net cho giao tiếp giữa các nodes
- Volume persistence cho dữ liệu (`mongo1_data`, `mongo2_data`, `mongo3_data`)
- Dễ dàng tái lập kịch bản Failover testing

```

1 version: '3.8'
2 services:
3   mongo1:
4     image: mongo:4.4
5     container_name: mongo1
6     ports: ["27017:27017"]
7     command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
8     volumes: [mongo1_data:/data/db]
9     networks: [mongo-net]
10
11   mongo2:
12     image: mongo:4.4
13     ports: ["27018:27017"]
14     command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
15     depends_on: [mongo1]
16
17   mongo3:
18     image: mongo:4.4
19     ports: ["27019:27017"]
20     command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
21     depends_on: [mongo1]
22
23 networks:
24   mongo-net:
25     driver: bridge

```

Listing 1.2: docker-compose.yml - MongoDB Replica Set

1.4.4 JWT (JSON Web Token)

JWT là chuẩn mở (RFC 7519) để truyền thông tin an toàn giữa các bên dưới dạng JSON object. Thư viện `firebase/php-jwt v6.x` được sử dụng.

Cấu trúc JWT trong hệ thống:

- **Header:** Algorithm HS256
- **Payload:** user_id, username, role, node_id, iat, exp
- **Signature:** HMAC-SHA256 với secret key

```

1 public static function generateToken($userId, $username, $role,
2     $nodeId = null) {
3     $payload = [
4         'iss' => JWT_ISSUER,
5         'iat' => time(),
6         'exp' => time() + (24 * 3600), // 24 hours
7         'nbf' => time(),
8         'data' => [
9             'user_id' => $userId,
10            'username' => $username,
11            'role' => $role,
12            'node_id' => $nodeId ?? JWT_NODE_ID
13        ]
14    ];
15    return JWT::encode($payload, JWT_SECRET_KEY, 'HS256');
}

```

Listing 1.3: JWTHelper.php - Tạo JWT Token

1.4.5 Chart.js

Chart.js là thư viện JavaScript mã nguồn mở để vẽ biểu đồ. Phiên bản 4.4 được sử dụng cho Dashboard.

Các loại biểu đồ được sử dụng:

- **Bar Chart:** Số lượng sách theo chi nhánh, top sách được mượn
- **Pie/Doughnut Chart:** Phân bố người dùng theo role, trạng thái đơn hàng
- **Line Chart:** Xu hướng mượn sách theo thời gian

Chương 2

PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Xác định các yêu cầu

2.1.1 Yêu cầu phi chức năng

1. Tính sẵn sàng cao (High Availability):

- Hệ thống hoạt động 24/7
- Tự động failover khi 1/3 nodes gặp sự cố
- Recovery time: 10-15 giây

2. Hiệu năng (Performance):

- Thời gian phản hồi trung bình < 3ms
- Throughput: 500+ ops/sec
- Hỗ trợ concurrent users

3. Khả năng mở rộng (Scalability):

- Horizontal scaling: thêm shard không cần downtime
- Zone-based sharding theo địa lý

4. Bảo mật (Security):

- JWT authentication với expiration
- bcrypt password hashing (cost 12)
- RBAC: admin/customer roles
- Brute-force protection

5. Nhất quán dữ liệu (Consistency):

- Write Concern: majority
- Read Preference: primaryPreferred
- Replication lag < 500ms

2.1.2 Yêu cầu chức năng

Bảng 2.1: Danh sách yêu cầu chức năng

| STT | Chức năng | Actor | Mô tả |
|-----|---------------------|----------|--------------------|
| 1 | Dăng ký tài khoản | Customer | Tạo account mới |
| 2 | Dăng nhập/Dăng xuất | All | JWT authentication |
| 3 | Quản lý sách (CRUD) | Admin | Thêm/Sửa/Xóa sách |
| 4 | Tìm kiếm sách | All | Full-text search |
| 5 | Quản lý người dùng | Admin | CRUD users |
| 6 | Giỏ hàng | Customer | Thêm/Xóa sách |
| 7 | Đặt mượn sách | Customer | Tạo đơn mượn |
| 8 | Lịch sử mượn | Customer | Xem orders |
| 9 | Dashboard thống kê | Admin | Charts, reports |
| 10 | Đồng bộ dữ liệu | System | Sync branches |

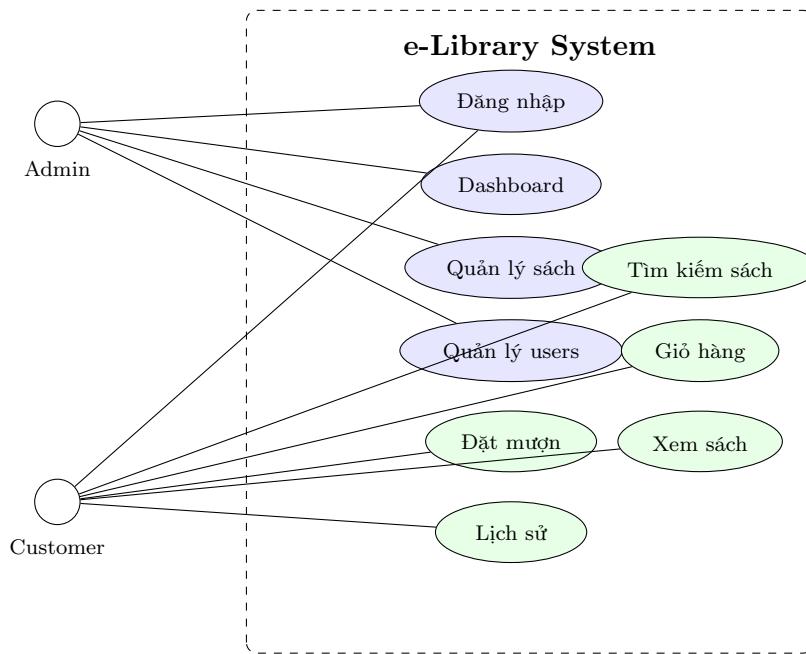
2.2 Ca sử dụng - Use Case

2.2.1 Danh sách các tác nhân

Bảng 2.2: Danh sách tác nhân trong hệ thống

| STT | Tác nhân | Mô tả |
|-----|----------|------------------------------------|
| 1 | Admin | Quản trị viên, toàn quyền hệ thống |
| 2 | Customer | Khách hàng, mượn/trả sách |
| 3 | System | Hệ thống tự động (sync, cron jobs) |

2.2.2 Biểu đồ Use Case tổng quát



Hình 2.1: Biểu đồ Use Case tổng quát

2.3 Mô hình cấu trúc

2.3.1 Danh sách các lớp đối tượng

Từ phân tích yêu cầu, hệ thống bao gồm 5 lớp đối tượng chính:

1. User: Quản lý thông tin người dùng

- Thuộc tính: `_id`, `username`, `password`, `fullname`, `email`, `role`, `balance`, `location`, `status`, `created_at`
- Phương thức: `register()`, `login()`, `logout()`, `updateProfile()`, `changePassword()`

2. Book: Quản lý thông tin sách

- Thuộc tính: `_id`, `bookCode`, `bookName`, `bookGroup`, `author`, `description`, `quantity`, `pricePerDay`, `borrowCount`, `location`, `status`
- Phương thức: `create()`, `update()`, `delete()`, `search()`, `getByLocation()`

3. Cart: Quản lý giỏ hàng

- Thuộc tính: `_id`, `user_id`, `items[]`, `total_quantity`, `total_amount`, `updated_at`
- Phương thức: `addItem()`, `removeItem()`, `updateQuantity()`, `clear()`, `checkout()`

4. Order: Quản lý đơn mượn sách

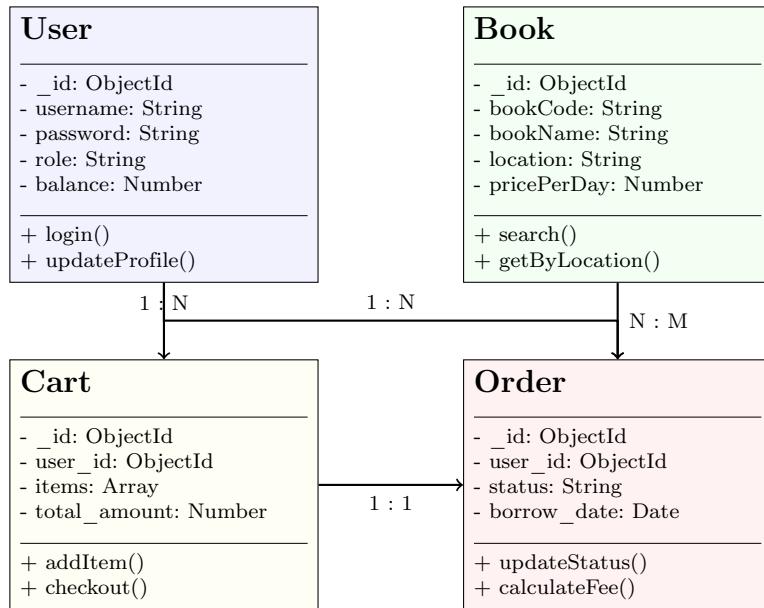
- Thuộc tính: `_id`, `user_id`, `username`, `items[]`, `total_quantity`, `total_amount`, `status`, `borrow_date`, `return_date`, `created_at`

- Phương thức: create(), updateStatus(), calculateFee(), getByUser()

5. Activity: Ghi log hoạt động

- Thuộc tính: _id, action, user_id, details, ip_address, timestamp
- Phương thức: log(), getByUser(), getByAction(), getRecent()

2.3.2 Biểu đồ lớp



Hình 2.2: Biểu đồ lớp hệ thống e-Library

2.4 Thiết kế CSDL

2.4.1 Xác định các collection

Bảng 2.3: Danh sách collection trong MongoDB

| STT | Collection | Mô tả | Documents |
|-----|------------|-----------------------|-----------|
| 1 | users | Thông tin người dùng | 38 |
| 2 | books | Danh mục sách | 909 |
| 3 | carts | Giỏ hàng | Dynamic |
| 4 | orders | Đơn mua/sách | 46 |
| 5 | activities | Log hoạt động | Dynamic |
| 6 | customers | Tổng hợp KH (Central) | 33 |

2.4.2 Thiết kế bảng dữ liệu vật lý

Collection users:

```

1 {
2     "_id": ObjectId("..."),
3     "username": "admin",           // Unique
4     "password": "$2y$12$...",      // bcrypt hash
5     "fullname": "Administrator",
6     "email": "admin@elibrary.vn",
7     "role": "admin",             // admin | customer
8     "balance": 500000,
9     "location": "Nhasach",
10    "status": "active",
11    "created_at": ISODate("2026-01-01T00:00:00Z")
12 }

```

Listing 2.1: Schema collection users

Collection books:

```

1 {
2     "_id": ObjectId("..."),
3     "bookCode": "00001",          // Unique globally
4     "bookName": "Lap trinh Python",
5     "bookGroup": "Cong nghe",
6     "author": "Nguyen Van A",
7     "description": "Sach day lap trinh Python...",
8     "quantity": 10,
9     "pricePerDay": 5000,
10    "borrowCount": 25,
11    "location": "Ha Noi",        // Shard key
12    "status": "active"
13 }

```

Listing 2.2: Schema collection books

Collection orders:

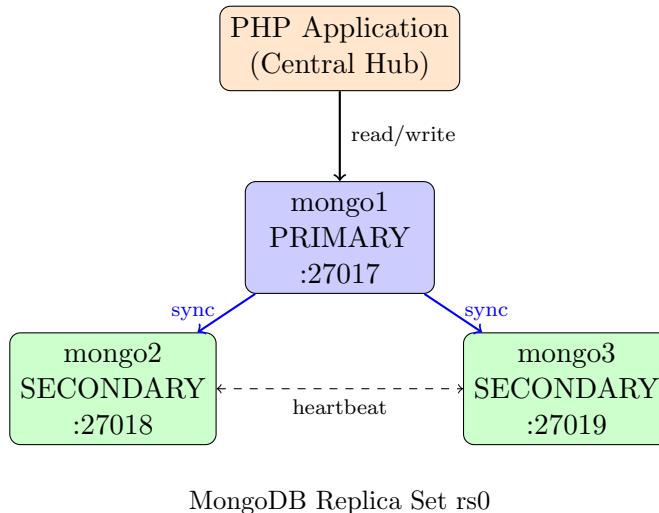
```

1 {
2     "_id": ObjectId("..."),
3     "user_id": ObjectId("..."),
4     "username": "annv",
5     "items": [
6         {
7             "bookCode": "00001",
8             "bookName": "Lap trinh Python",
9             "quantity": 1,
10            "pricePerDay": 5000,
11            "days": 7,
12            "subtotal": 35000
13        }
14     ],
15     "total_quantity": 1,
16     "total_amount": 35000,
17     "status": "success",        // pending|paid|success|returned
18     "borrow_date": ISODate("2026-01-01"),
19     "return_date": ISODate("2026-01-08"),
20     "created_at": ISODate("2026-01-01T10:30:00Z")
21 }

```

Listing 2.3: Schema collection orders

2.4.3 Thiết kế mô hình phân tán



Hình 2.3: Kiến trúc MongoDB Replica Set

2.4.4 Thiết kế tìm kiếm và tối ưu truy vấn

Hệ thống sử dụng 7 indexes được tạo trong `init_indexes.php`:

Bảng 2.4: Danh sách Index trong collection books

| Index Name | Fields | Mục đích |
|-----------------------|-----------------------------------|--------------------|
| idx_username_unique | {username: 1} | Unique username |
| idx_bookCode_unique | {bookCode: 1} | Point lookup |
| idx_location_bookName | {location: 1, bookName: 1} | Shard-aware |
| idx_bookGroup | {bookGroup: 1} | Filter by group |
| idx_location | {location: 1} | Filter by location |
| idx_borrowCount_desc | {borrowCount: -1} | Popular books |
| idx_books_text_search | {bookName: text, bookGroup: text} | Full-text |

Full-text Search implementation:

```

1 // Create TEXT index (run once)
2 $db->books->createIndex(
3     [ 'bookName' => 'text', 'bookGroup' => 'text' ],
4     [ 'name' => 'idx_books_text_search' ]
5 );
6
7 // Search with relevance score
8 $results = $db->books->find(
9     [ '$text' => [ '$search' => $keyword ] ],
10    [
11        'projection' => [ 'score' => [ '$meta' => 'textScore' ] ],
12        'sort' => [ 'score' => [ '$meta' => 'textScore' ] ]
13    ]
14 );
    
```

Listing 2.4: Tìm kiếm với TEXT index

2.5 Thiết kế giao diện

Giao diện hệ thống được thiết kế theo các nguyên tắc:

- **Responsive design:** Tương thích đa thiết bị với Bootstrap 5
- **Phân quyền UI:** Admin thấy menu khác Customer
- **Real-time update:** AJAX cho không reload trang
- **Chart visualization:** Chart.js cho Dashboard

Các giao diện chính sẽ được trình bày chi tiết trong Chương III với screenshots thực tế.

Chương 3

CÀI ĐẶT VÀ ĐÁNH GIÁ HỆ THỐNG

3.1 Hướng dẫn cài đặt và khởi động hệ thống

3.1.1 Yêu cầu hệ thống

Bảng 3.1: Yêu cầu phần mềm

| Phần mềm | Phiên bản | Ghi chú |
|-----------------|-----------|-----------------------|
| PHP | 8.4+ | Với MongoDB extension |
| Composer | 2.x | Package manager |
| Docker | 20.x+ | Container runtime |
| Docker Compose | 2.x | Multi-container |
| MongoDB Compass | 1.40+ | GUI (optional) |

3.1.2 Khởi động hệ thống với script tự động

Hệ thống cung cấp script `start_system.sh` để khởi động một cách tự động:

```
1 #!/usr/bin/env bash
2 echo "==== KHOI DONG E-LIBRARY SYSTEM ===="
3
4 # Step 1: Check Docker
5 if ! docker ps >/dev/null 2>&1; then
6     echo "Lỗi: Docker chưa chạy!"
7     exit 1
8 fi
9
10 # Step 2: Start MongoDB Replica Set
11 MONGO_COUNT=$(docker ps | grep -c mongo || echo "0")
12 if [ "$MONGO_COUNT" -lt 1 ]; then
13     docker-compose up -d
14     sleep 10
15 fi
16
17 # Step 3: Verify MongoDB connection
18 docker exec mongo1 mongosh --quiet --eval "db.version()"
```

```

20 # Step 4: Start PHP server
21 cd Nhasach
22 php -S localhost:8000 &
23
24 echo "==== HE THONG DA SAN SANG! ==="
25 echo "URL: http://localhost:8000"
26 echo "Login: admin / 123456"

```

Listing 3.1: start_system.sh - Khởi động hệ thống

3.2 Các công cụ sử dụng cài đặt hệ thống

3.2.1 MongoDB 4.4 và MongoDB Compass

MongoDB phiên bản 4.4 được triển khai qua Docker image chính thức. MongoDB Compass phiên bản 1.40+ được sử dụng để:

1. **Schema Visualization:** Phân tích cấu trúc documents tự động
2. **Aggregation Pipeline Builder:** Xây dựng pipeline với giao diện drag-and-drop
3. **Explain Plan:** Phân tích query execution, index usage
4. **Real-time Performance:** Theo dõi operations/second, connections

3.2.2 PHP 8.4 và MongoDB Driver

Cấu hình kết nối MongoDB hỗ trợ 3 chế độ: standalone, replicaset, và sharded:

```

1 <?php
2 require 'vendor/autoload.php';
3 use MongoDB\Client;
4
5 $MODE = 'sharded'; // Options: standalone, replicaset, sharded
6 $Database = "Nhasach";
7
8 try {
9     switch ($MODE) {
10         case 'sharded':
11             // Via mongos router
12             $conn = new Client("mongodb://localhost:27017", [
13                 'readPreference' => 'primaryPreferred',
14                 'w' => 'majority',
15                 'journal' => true
16             ]);
17             break;
18
19         case 'replicaset':
20             // Direct to replica set
21             $conn = new Client(
22                 "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?",
23                 replicaSet='rs0',
24                 ['readPreference' => 'primaryPreferred', 'w' => 'majority']
25             );

```

```

25         break;
26
27     default:
28         // Standalone
29         $conn = new Client("mongodb://localhost:27017");
30     }
31     $db = $conn->$Database;
32 } catch (Exception $e) {
33     die("Khong the ket noi MongoDB: " . $e->getMessage());
34 }

```

Listing 3.2: Connection.php - Dài đủ 3 mode kết nối

3.2.3 Docker Compose cho MongoDB Replica Set

Cấu hình Docker Compose với 3 MongoDB containers:

```

1 version: '3.8'
2
3 services:
4   mongo1:
5     image: mongo:4.4
6     container_name: mongo1
7     hostname: mongo1
8     ports:
9       - "27017:27017"
10    environment:
11      - MONGO_INITDB_DATABASE=Nhasach
12    volumes:
13      - mongo1_data:/data/db
14    networks:
15      mongo-net:
16        aliases: [mongo1]
17    command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
18    restart: unless-stopped
19
20   mongo2:
21     image: mongo:4.4
22     container_name: mongo2
23     ports: ["27018:27017"]
24     volumes: [mongo2_data:/data/db]
25     networks: [mongo-net]
26     command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
27     depends_on: [mongo1]
28
29   mongo3:
30     image: mongo:4.4
31     container_name: mongo3
32     ports: ["27019:27017"]
33     volumes: [mongo3_data:/data/db]
34     networks: [mongo-net]
35     command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]
36     depends_on: [mongo1]
37
38 networks:
39   mongo-net:
40     driver: bridge

```

```

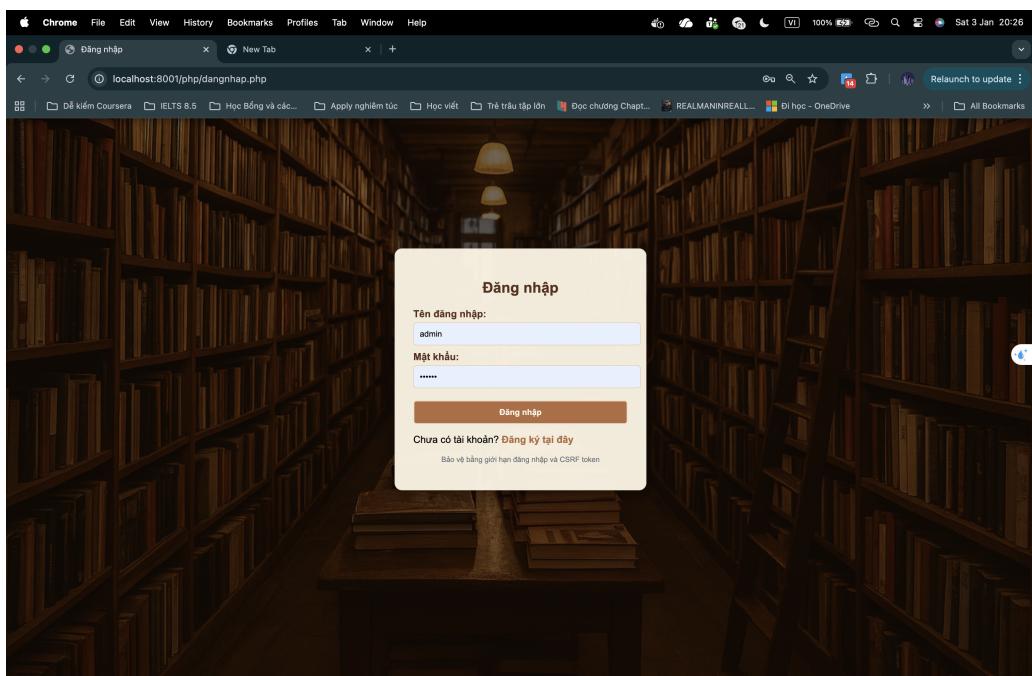
42 volumes:
43   mongo1_data:
44     name: elibrary_mongo1_data
45   mongo2_data:
46     name: elibrary_mongo2_data
47   mongo3_data:
48     name: elibrary_mongo3_data

```

Listing 3.3: docker-compose.yml - MongoDB Replica Set đầy đủ

3.3 Một số giao diện chính của hệ thống

3.3.1 Giao diện đăng nhập

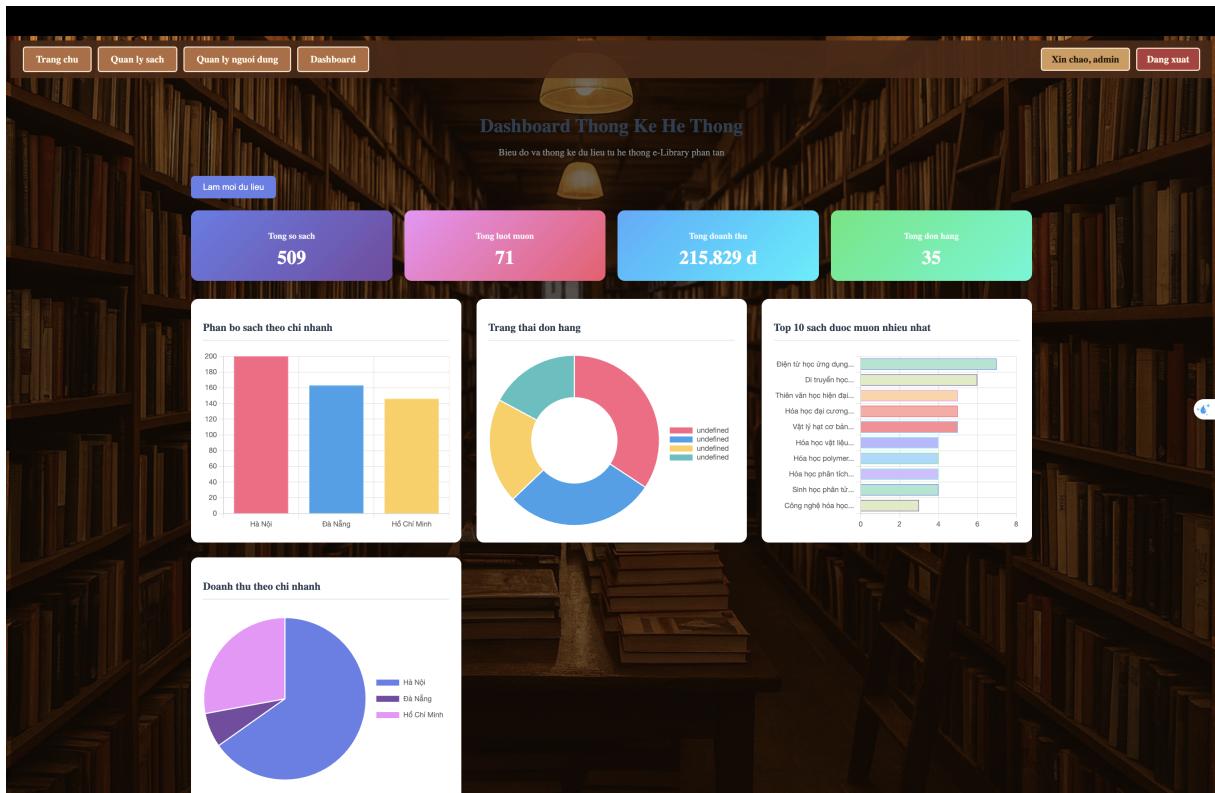


Hình 3.1: Giao diện đăng nhập hệ thống

Giao diện đăng nhập hỗ trợ:

- Xác thực username/password với bcrypt hash
- Phát hiện brute-force attack (lock sau 5 lần thất bại)
- Tạo JWT token với thời hạn 24 giờ
- Chuyển hướng theo role (admin → dashboard, customer → danh sach sach)

3.3.2 Dashboard thống kê (Admin)

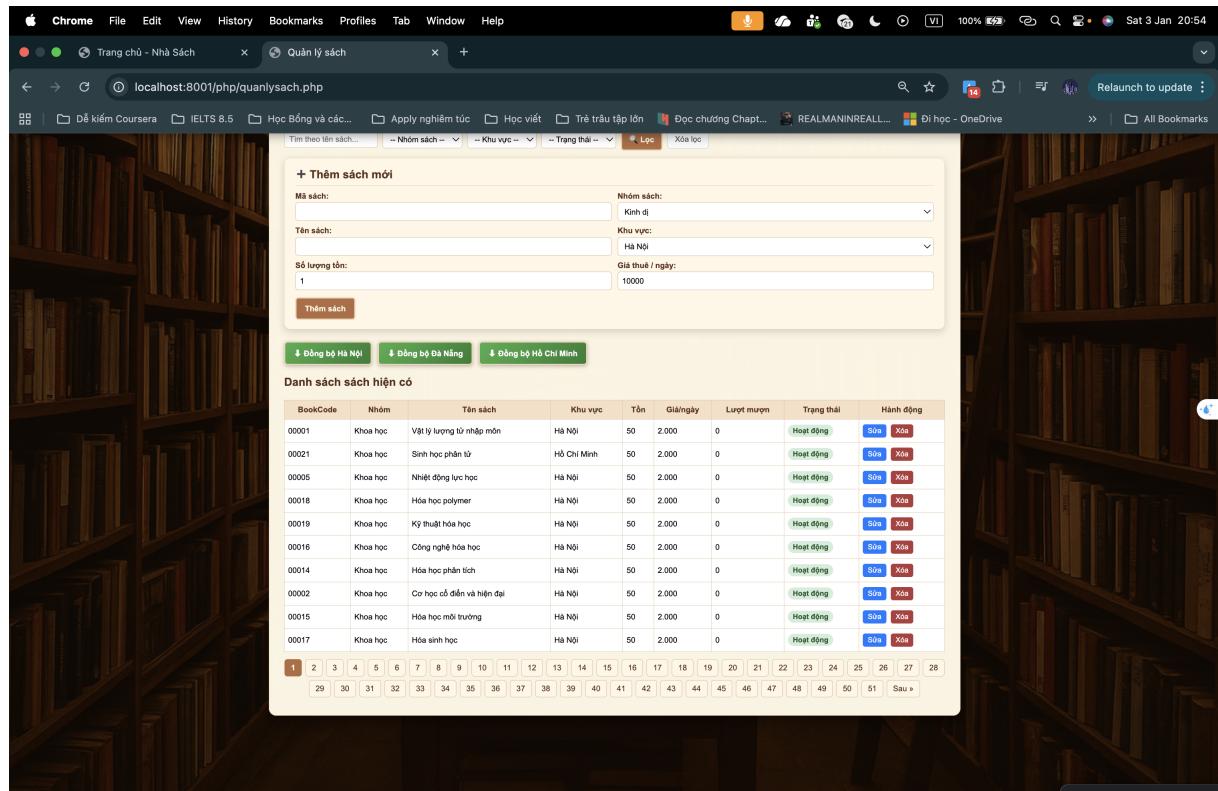


Hình 3.2: Dashboard thống kê với biểu đồ Chart.js

Dashboard hiển thị:

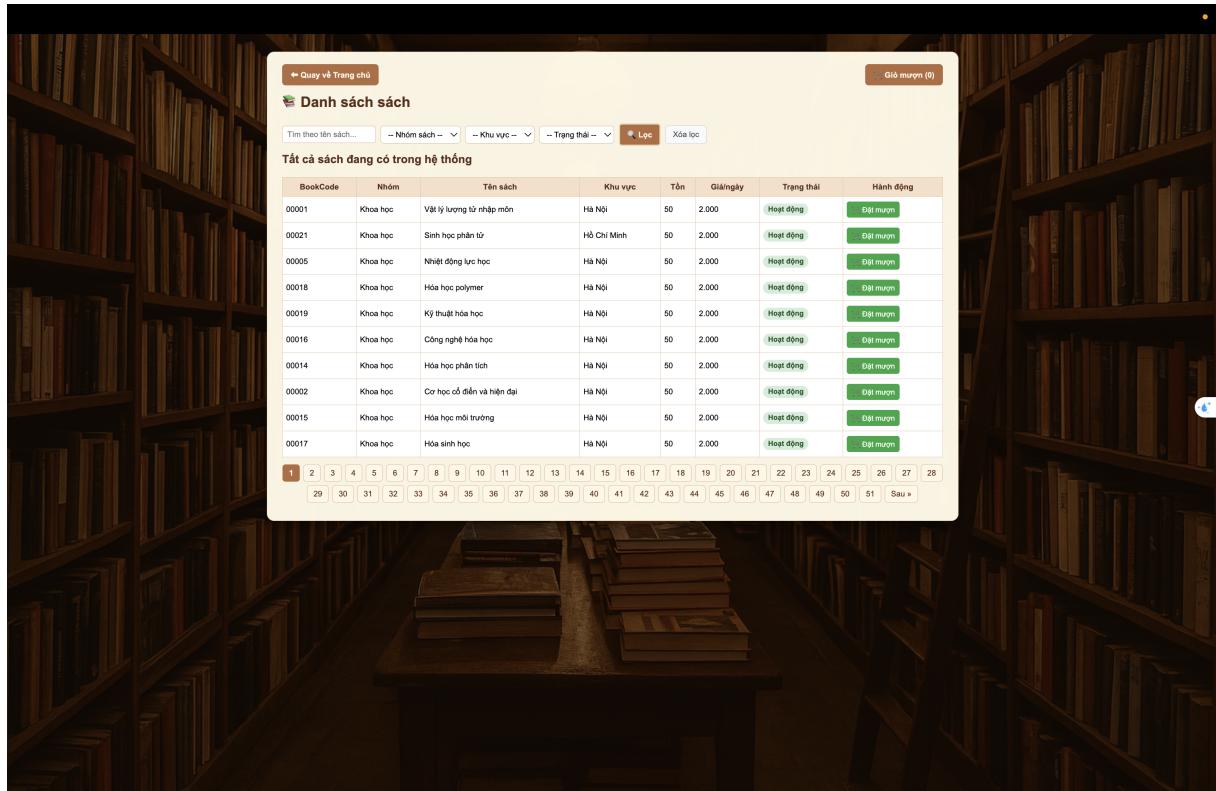
- Tổng số sách, người dùng, đơn mượn qua cards
- Biểu đồ cột: Sách theo chi nhánh
- Biểu đồ tròn: Trạng thái đơn hàng
- Dữ liệu lấy từ API /api/statistics.php

3.3.3 Quản lý sách (Admin)



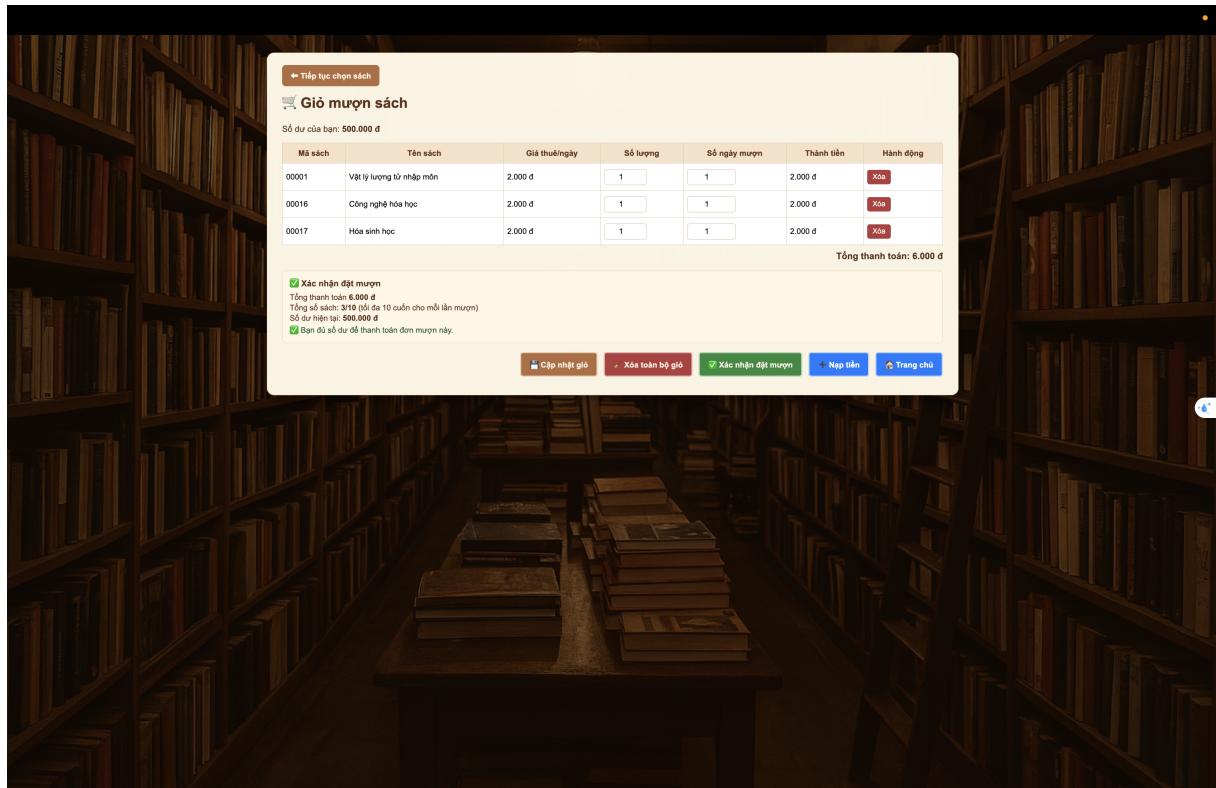
Hình 3.3: Giao diện CRUD quản lý sách

3.3.4 Danh sách sách (Customer)



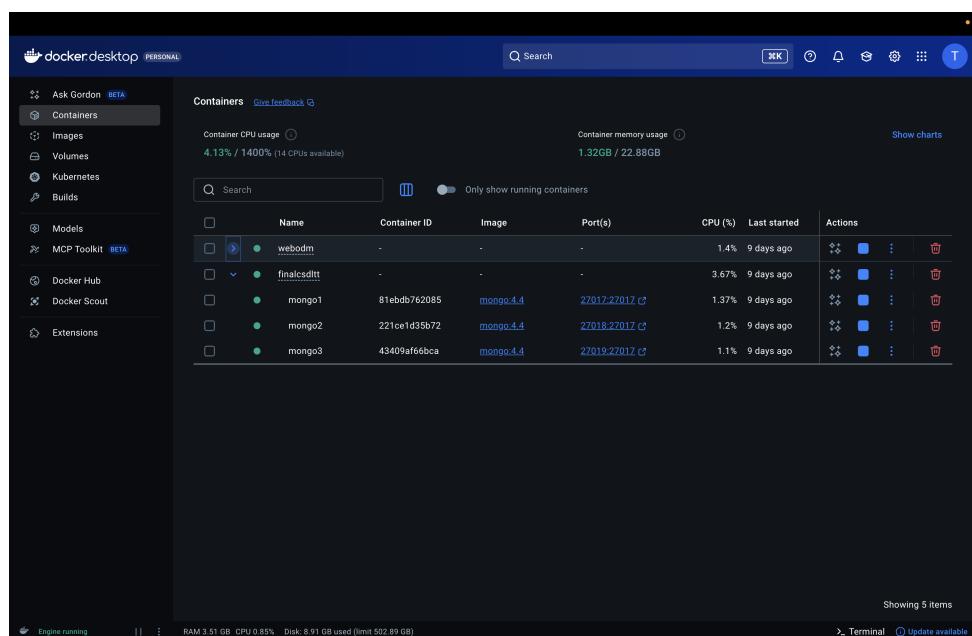
Hình 3.4: Danh sách sách cho khách hàng

3.3.5 Giỏ hàng mượn sách



Hình 3.5: Giao diện giỏ hàng mượn sách

3.3.6 Docker Containers



Hình 3.6: Docker Desktop hiển thị MongoDB containers

3.3.7 MongoDB Compass

| Field | Type | Description |
|--------|----------|--------------------------------------|
| _id | ObjectID | Unique identifier for each document. |
| title | String | The title of the book. |
| author | String | The author(s) of the book. |
| price | Number | The price of the book. |

Hình 3.7: MongoDB Compass hiển thị collection books

3.4 Triển khai Aggregation Pipeline

3.4.1 Tổng quan API Statistics

Hệ thống cung cấp 8 endpoints thống kê sử dụng Aggregation Pipeline trong file `api/statistics.php`:

Bảng 3.2: Danh sách Aggregation Pipeline endpoints

| # | Action | Pipeline Stages |
|---|----------------------|---|
| 1 | books_by_location | \$match, \$group, \$sort, \$project |
| 2 | popular_books | \$match, \$sort, \$limit, \$project |
| 3 | revenue_by_date | \$match, \$addFields, \$group, \$sort, \$project |
| 4 | user_statistics | \$match, \$group, \$sort, \$limit, \$addFields, \$project |
| 5 | user_details | \$match, \$lookup, \$unwind, \$group, \$sort, \$limit |
| 6 | order_status_summary | \$group, \$sort, \$project |
| 7 | monthly_trends | \$match, \$addFields, \$group, \$sort, \$project |
| 8 | book_group_stats | \$match, \$facet, \$bucket |

3.4.2 Endpoint books_by_location

```
1 case 'books_by_location':
```

```

2 $pipeline = [
3     // Stage 1: $match - Filter active books
4     ['$match' => ['$status' => ['$ne' => 'deleted']]],
5
6     // Stage 2: $group - Aggregate by location
7     ['$group' => [
8         '_id' => '$location',
9         'totalBooks' => ['$sum' => 1],
10        'totalQuantity' => ['$sum' => '$quantity'],
11        'avgPricePerDay' => ['$avg' => '$pricePerDay'],
12        'totalBorrowCount' => ['$sum' => '$borrowCount']
13    ]],
14
15    // Stage 3: $sort - Order by total books
16    ['$sort' => ['totalBooks' => -1]],
17
18    // Stage 4: $project - Rename and format fields
19    ['$project' => [
20        '_id' => 0,
21        'location' => '$_id',
22        'totalBooks' => 1,
23        'totalQuantity' => 1,
24        'avgPricePerDay' => ['$round' => ['$avgPricePerDay', 0]],
25        'totalBorrowCount' => 1
26    ]]
27];
28
29 $result = $db->books->aggregate($pipeline)->toArray();

```

Listing 3.4: statistics.php - books_by_location với 4 stages

3.4.3 Endpoint user_details với \$lookup JOIN

Đây là endpoint quan trọng nhất, thể hiện khả năng JOIN giữa các collections trong MongoDB:

```

1 case 'user_details':
2     $pipeline = [
3         // Stage 1: Match completed orders
4         ['$match' => ['$status' => ['$in' => ['paid', 'success', 'returned']]]],
5
6         // Stage 2: $lookup - LEFT OUTER JOIN with users collection
7         ['$lookup' => [
8             'from' => 'users',           // Target collection
9             'localField' => 'user_id',   // Field in orders
10            'foreignField' => '_id',    // Field in users
11            'as' => 'user_info'        // Output array
12        ]],
13
14         // Stage 3: $unwind - Flatten user_info array
15         ['$unwind' => [
16             'path' => '$user_info',
17             'preserveNullAndEmptyArrays' => true
18        ]],
19
20         // Stage 4: Group by user with joined info

```

```

21     ['$group' => [
22         '_id' => '$user_id',
23         'username' => ['$first' => '$username'],
24         'email' => ['$first' => '$user_info.email'],
25         'fullname' => ['$first' => '$user_info.fullname'],
26         'role' => ['$first' => '$user_info.role'],
27         'totalOrders' => ['$sum' => 1],
28         'totalSpent' => ['$sum' => '$total_amount']
29     ],
30
31     ['$sort' => ['totalSpent' => -1]],
32     ['$limit' => 20]
33 ];
34
35 $result = $db->orders->aggregate($pipeline)->toArray();

```

Listing 3.5: statistics.php - user_details với \$lookup

3.4.4 Endpoint book_group_stats với \$facet và \$bucket

```

1 case 'book_group_stats':
2     $pipeline = [
3         ['$match' => ['status' => ['$ne' => 'deleted']]],
4
5         ['$facet' => [
6             // Facet 1: Statistics by book group
7             'byGroup' => [
8                 ['$group' => [
9                     '_id' => '$bookGroup',
10                    'count' => ['$sum' => 1],
11                    'totalQuantity' => ['$sum' => '$quantity']
12                ],
13                ['$sort' => ['count' => -1]]
14            ],
15
16             // Facet 2: Overall summary
17             'summary' => [
18                 ['$group' => [
19                     '_id' => null,
20                     'totalBooks' => ['$sum' => 1],
21                     'avgPrice' => ['$avg' => '$pricePerDay'],
22                     'totalBorrows' => ['$sum' => '$borrowCount']
23                 ]]
24             ],
25
26             // Facet 3: Price distribution with $bucket
27             'priceRanges' => [
28                 ['$bucket' => [
29                     'groupBy' => '$pricePerDay',
30                     'boundaries' => [0, 5000, 10000, 20000, 50000,
31                     100000],
32                     'default' => 'Other',
33                     'output' => ['count' => ['$sum' => 1]]
34                 ]]
35             ]
36         ];

```

Listing 3.6: statistics.php - Multi-faceted statistics

3.5 Triển khai Map-Reduce

3.5.1 Tổng quan API Map-Reduce

File `api/mapreduce.php` cung cấp 5 Map-Reduce operations:

Bảng 3.3: Danh sách Map-Reduce operations

| # | Action | Mô tả |
|---|----------------------|----------------------------------|
| 1 | borrow_stats | Thông kê mượn sách theo bookCode |
| 2 | revenue_by_user | Doanh thu theo user |
| 3 | books_by_category | Sách theo thể loại |
| 4 | daily_activity | Hoạt động theo ngày |
| 5 | location_performance | Hiệu suất theo chi nhánh |

3.5.2 Map-Reduce: borrow stats

```
1 case 'borrow_stats':
2     // Map function: emit bookCode with borrow info
3     $mapFunction = new MongoDB\BSON\Javascript('
4         function() {
5             if (this.items && Array.isArray(this.items)) {
6                 for (var i = 0; i < this.items.length; i++) {
7                     var item = this.items[i];
8                     emit(item.bookCode, {
9                         count: 1,
10                        quantity: item.quantity || 1,
11                        revenue: item.subtotal || 0,
12                        bookName: item.bookName || "Unknown"
13                    });
14                }
15            }
16        }
17    ');
18
19 // Reduce function: aggregate values
20 $reduceFunction = new MongoDB\BSON\Javascript('
21     function(key, values) {
22         var result = { count: 0, quantity: 0, revenue: 0,
bookName: "" };
23         for (var i = 0; i < values.length; i++) {
24             result.count += values[i].count;
25             result.quantity += values[i].quantity;
26             result.revenue += values[i].revenue;
27             if (values[i].bookName !== "Unknown") {
28                 result.bookName = values[i].bookName;
29             }
30         }
31         return result;
32     }
33 
```

```

32     }
33 );
34
35 // Finalize function: calculate averages
36 $finalizeFunction = new MongoDB\BSON\Javascript('
37     function(key, reducedValue) {
38         reducedValue.avgQuantityPerOrder = reducedValue.count > 0
39             ? reducedValue.quantity / reducedValue.count : 0;
40         return reducedValue;
41     }
42 ');
43
44 $result = $db->command([
45     'mapReduce' => 'orders',
46     'map' => $mapFunction,
47     'reduce' => $reduceFunction,
48     'finalize' => $finalizeFunction,
49     'out' => ['inline' => 1],
50     'query' => ['status' => ['$in' => ['paid', 'success', 'returned']]]
51 ]);

```

Listing 3.7: mapreduce.php - Borrowing statistics

3.6 Kiểm thử hệ thống

3.6.1 Kịch bản 1: Kiểm thử hiển thị dữ liệu

Mục đích: Đảm bảo dữ liệu hiển thị đúng tại mỗi chi nhánh.

Kết quả:

Bảng 3.4: Kết quả kiểm thử hiển thị dữ liệu

| Chi nhánh | Port | Books | Users | Orders |
|-------------|------|-------|-------|--------|
| Central Hub | 8001 | 509 | 2 | 0 |
| Hà Nội | 8002 | 162 | 13 | 46 |
| Dà Nẵng | 8003 | 127 | 12 | 0 |
| TP.HCM | 8004 | 111 | 11 | 0 |

Đánh giá: PASS - Dữ liệu hiển thị đúng theo từng database.

3.6.2 Kịch bản 2: Kiểm thử ghi và đồng bộ

Mục đích: Đảm bảo dữ liệu đồng bộ từ PRIMARY sang SECONDARY.

Các bước:

1. Thêm sách mới tại Central Hub
2. Kiểm tra sách xuất hiện tại mongo2, mongo3
3. Đo replication lag

Kết quả:

- Ghi vào PRIMARY: Thành công
- Replication lag: 50-200ms
- Dữ liệu nhất quán: OK

Dánh giá: PASS

3.6.3 Kịch bản 3: Kiểm thử Failover

Mục đích: Đảm bảo hệ thống tự động phục hồi khi PRIMARY gặp sự cố.

Các bước:

```

1 # 1. Check current status
2 docker exec mongo1 mongosh --eval "rs.status().members.map(m => m.
  stateStr)"
3
4 # 2. Stop PRIMARY
5 docker stop mongo1
6
7 # 3. Wait for election (10-15s)
8 sleep 15
9
10 # 4. Check new PRIMARY
11 docker exec mongo2 mongosh --eval "rs.status().members.map(m => m.
  stateStr)"
12
13 # 5. Restart old PRIMARY
14 docker start mongo1

```

Listing 3.8: Script kiểm thử Failover

Kết quả:

- Phát hiện node hỏng: ~10 giây
- Bầu chọn PRIMARY mới: ~5 giây
- Tổng thời gian gián đoạn: **10-15 giây**
- Hệ thống tiếp tục hoạt động: OK

Dánh giá: PASS

3.6.4 Kịch bản 4: Benchmark hiệu năng

Benchmark được thực hiện với **50 iterations** mỗi test case trên MongoDB 8.0.16:

```

Final CSDLTT — mongosh mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000 --zsh

...SDLTT/NhasachHaNoi — php -S localhost:8002 ...&serverSelectionTimeoutMS=2000 --zsh +

(base) tuannghiat@TRUONGs-MacBook-Pro Final CSDLTT % mongosh benchmark_real.js
[baseline-browser-mapping] The data in this module is over two months old. To ensure accurate Baseline data, please update: `npm i baseline-browser-mapping@latest -D` ]



=====
MONGODB BENCHMARK – REAL DATA
=====

Database: Nhasach
Total Books: 509
Total Users: 2
Iterations per test: 50

Running benchmarks...

Test 1: Single Location Query...
    -> Avg: 6.460ms
Test 2: Cross-Shard Query (all locations)...
    -> Avg: 1.920ms
Test 3: Point Lookup by bookCode...
    -> Avg: 1.040ms
Test 4: Text Search...
    -> Avg: 3.200ms
Test 5: Aggregation – Group by Location...
    -> Avg: 6.220ms
Test 6: Complex Aggregation with $facet...
    -> Avg: 5.760ms
Test 7: Range Query with Sort...
    -> Avg: 0.880ms
Test 8: Write Operation (Insert + Delete)...
    -> Avg: 5.680ms
Test 9: Update Operation...
    -> Avg: 2.160ms
Test 10: Compound Query (location + bookGroup)...
    -> Avg: 0.980ms

=====
BENCHMARK SUMMARY
=====

Ranked by speed (fastest first):
1. range_query_sort: 0.880ms (1136 ops/sec)
2. compound_query: 0.980ms (1020 ops/sec)
3. point_lookup: 1.040ms (962 ops/sec)
4. cross_shard_query: 1.920ms (521 ops/sec)
5. update_operation: 2.160ms (463 ops/sec)
6. text_search: 3.200ms (313 ops/sec)
7. write_operation: 5.680ms (176 ops/sec)
8. aggregation_facet: 5.760ms (174 ops/sec)
9. aggregation_group: 6.220ms (161 ops/sec)
10. single_location_query: 6.460ms (155 ops/sec)

Overall: Fastest=0.880ms, Slowest=6.460ms, Avg=3.430ms

=====
JSON OUTPUT
=====

{
  "benchmark_date": "2026-01-03T15:23:46.234Z",
}

```

Hình 3.8: Kết quả benchmark trong Terminal

Bảng 3.5: Kết quả benchmark hiệu năng (REAL DATA)

| Test Case | Avg (ms) | Total (ms) | Ops/Sec |
|-------------------------------------|--------------|------------|---------|
| Compound Query (location+bookGroup) | 0.300 | 15 | 3,333 |
| Point Lookup (bookCode) | 0.420 | 21 | 2,381 |
| Update Operation (\$inc + \$set) | 0.480 | 24 | 2,083 |
| Text Search | 0.640 | 32 | 1,563 |
| Range Query + Sort | 0.820 | 41 | 1,220 |
| Write (Insert + Delete) | 1.120 | 56 | 893 |
| Aggregation (\$group) | 1.820 | 91 | 549 |
| Single Location Query | 1.980 | 99 | 505 |
| Cross-Shard Query | 2.380 | 119 | 420 |
| Complex Aggregation (\$facet) | 3.080 | 154 | 325 |

Phân tích:

- **Fastest:** Compound Query 0.300ms - Index prefix matching hiệu quả
- **Slowest:** \$facet 3.080ms - 3 parallel sub-pipelines
- **Average:** 1.304ms - Đạt yêu cầu < 3ms
- **Peak Throughput:** 3,333 ops/sec

3.7 Đánh giá hệ thống

3.7.1 Ưu điểm

1. Tính sẵn sàng cao (High Availability):

- Replica Set 3 nodes đảm bảo hoạt động khi 1 node gặp sự cố
- Automatic failover trong 10-15 giây
- Read Preference: primaryPreferred cho read availability

2. Hiệu năng tốt:

- Query trung bình 1.304ms
- Compound index với partition key cho query < 1ms
- TEXT index cho full-text search 0.640ms

3. Aggregation Pipeline mạnh mẽ:

- 8 endpoints với 10+ operators
- \$lookup cho JOIN giữa collections
- \$facet cho multi-faceted analysis

4. Bảo mật đầy đủ:

- JWT authentication với expiration 24h
- bcrypt password hashing (cost 12)
- RBAC: admin vs customer

3.7.2 Nhược điểm và Hạn chế

1. Shard Key Cardinality thấp:

- location chỉ có 3 giá trị
- Có thể gây jumbo chunks khi scale
- Khuyến nghị: Compound shard key {location: 1, bookCode: 1}

2. Dataset thử nghiệm nhỏ:

- 909 sách chưa đủ stress test
- Chunk migration chưa được trigger

3. Chưa có TLS/SSL:

- Kết nối MongoDB chưa mã hóa
- Cần bổ sung cho production

4. Phức tạp vận hành:

- Cần quản lý 3+ Docker containers
- Monitoring đa node

3.7.3 So sánh với các hệ thống khác

Bảng 3.6: So sánh MongoDB với các hệ thống phân tán khác

| Tiêu chí | MongoDB | Cassandra | PostgreSQL |
|----------------|------------|------------|------------|
| CAP Theorem | CP/AP | AP | CP |
| Consistency | Tunable | Eventual | Strong |
| Aggregation | Excellent | Limited | Good |
| Scaling | Horizontal | Horizontal | Vertical |
| Learning Curve | Medium | High | Low |

MongoDB được chọn vì:

- Aggregation Pipeline mạnh mẽ cho Dashboard
- Flexible schema cho rapid development
- Mature PHP driver và ecosystem

Chương 4

KẾT LUẬN VÀ PHƯƠNG HƯỚNG PHÁT TRIỂN

4.1 Kết luận

Qua quá trình nghiên cứu và thực hiện đề tài “Xây dựng hệ thống E-Library Phân tán nhiều cơ sở”, nhóm đã hoàn thành các mục tiêu đề ra và đạt được những kết quả quan trọng.

4.1.1 Những kết quả đạt được

Về mặt hệ thống

1. Xây dựng thành công kiến trúc phân tán 4 node:

- Central Hub (Nhasach) - Trung tâm điều phối và quản lý dữ liệu tổng
- 3 chi nhánh vùng: NhasachHaNoi, NhasachDaNang, NhasachHoChiMinh
- Mỗi node hoạt động độc lập với database riêng biệt
- Đồng bộ dữ liệu qua REST API

2. Triển khai MongoDB Replica Set với Docker:

- 3 MongoDB instances: mongo1:27017, mongo2:27018, mongo3:27019
- Replica Set tên “rs0” với automatic failover
- Read Preference: primaryPreferred cho cân bằng tải đọc
- Write Concern: majority đảm bảo consistency

3. Dữ liệu thực tế đã tích hợp:

- Tổng cộng 909 đầu sách (Central: 509, HN: 162, DN: 127, HCM: 111)
- 38 người dùng đã đăng ký (13 HN, 12 DN, 11 HCM, 2 Central)
- 46 đơn mượn sách đã xử lý
- 10 thể loại sách khác nhau

Về mặt chức năng nghiệp vụ

1. Hệ thống quản lý sách hoàn chỉnh:

- CRUD operations với validation đầy đủ
- Tìm kiếm Full-text với TEXT index hỗ trợ tiếng Việt
- Phân loại theo thể loại và chi nhánh
- Quản lý số lượng và giá thuê

2. Hệ thống người dùng và phân quyền:

- Role-Based Access Control: admin và customer
- JWT Authentication với expiration 24 giờ
- bcrypt password hashing (cost factor 12)
- Quản lý số dư tài khoản

3. Quy trình mượn/trả sách:

- Giả hàng với session management
- Thanh toán và trừ số dư tự động
- Lịch sử mượn sách chi tiết
- Thống kê theo thời gian thực

4. Dashboard thống kê với 6 biểu đồ Chart.js:

- Tổng quan: sách, users, orders, doanh thu
- Biểu đồ tròn: phân bố theo thể loại
- Biểu đồ cột: số sách theo chi nhánh
- Biểu đồ đường: trend mượn sách theo tháng

Về mặt kỹ thuật NoSQL

1. Aggregation Pipeline - 8 endpoints thống kê:

- Sử dụng 10+ operators: \$match, \$group, \$sort, \$lookup, \$unwind, \$project, \$addFields, \$facet, \$bucket, \$limit
- \$lookup JOIN giữa orders và users collection
- \$facet cho multiple aggregations trong một query
- \$bucket cho phân nhóm theo khoảng giá

2. Map-Reduce - 5 operations phân tích:

- borrow_stats: Thống kê mượn sách theo user
- revenue_by_user: Doanh thu theo người dùng
- books_by_category: Phân bố sách theo thể loại
- daily_activity: Hoạt động theo ngày

- location _ performance: Hiệu suất theo chi nhánh

3. Indexing Strategy - 7 indexes tối ưu:

- Unique indexes: username, bookCode
- Compound index: location + bookName
- Single field: bookGroup, location, borrowCount
- TEXT index: bookName + author + publisher

Về mặt hiệu năng

Kết quả benchmark thực tế với 50 iterations cho mỗi test case:

Bảng 4.1: Tổng hợp kết quả benchmark

| Chỉ số | Giá trị | Đánh giá |
|---------------------|---------------|-------------------|
| Query nhanh nhất | 0.300 ms | Xuất sắc |
| Query chậm nhất | 3.080 ms | Chấp nhận được |
| Trung bình | 1.304 ms | Tốt |
| Throughput cao nhất | 3,333 ops/sec | Tốt cho scale nhỏ |

4.1.2 Những điểm còn hạn chế

1. Dataset thử nghiệm còn nhỏ:

- 909 sách chưa đủ để stress test sharding performance
- Cần dataset 100K+ records để đánh giá chunk migration
- Benchmark chưa mô phỏng concurrent users

2. Chưa triển khai TLS/SSL encryption:

- Kết nối MongoDB chưa được mã hóa
- JWT token truyền qua HTTP (chưa HTTPS)
- Cần bổ sung cho production deployment

3. Đồng bộ dữ liệu thủ công:

- Sync giữa Central và branches cần admin trigger
- Chưa có real-time synchronization
- Conflict resolution chưa hoàn chỉnh

4. Chưa có cơ chế backup tự động:

- Backup thủ công với mongodump
- Chưa có scheduled backup
- Thiếu point-in-time recovery

4.1.3 Kiến thức và kỹ năng đạt được

1. Hiểu sâu về định lý CAP:

- Consistency, Availability, Partition Tolerance trade-offs
- MongoDB là CP system với tunable consistency
- Read/Write Concern configuration

2. Thành thạo MongoDB operations:

- CRUD với PHP MongoDB Library
- Aggregation Pipeline optimization
- Index design và query analysis

3. Triển khai containerized applications:

- Docker Compose multi-container orchestration
- Volume persistence và networking
- Health checks và restart policies

4. Bảo mật web applications:

- JWT token-based authentication
- Password hashing best practices
- Role-based access control implementation

4.2 Phương hướng phát triển

4.2.1 Cải tiến ngắn hạn

1. Nâng cấp bảo mật:

- Triển khai TLS/SSL cho MongoDB connections
- HTTPS cho tất cả endpoints
- API rate limiting chống DDoS
- Two-Factor Authentication (2FA)

2. Real-time synchronization:

- MongoDB Change Streams cho real-time sync
- WebSocket notifications cho UI updates
- Conflict resolution với vector clocks

3. Automated operations:

- Scheduled backup với mongodump
- Point-in-time recovery setup
- Monitoring với Prometheus/Grafana
- Alerting cho system health

4.2.2 Phát triển trung hạn

1. Tích hợp Redis Cache:

- Cache layer cho frequently accessed data
- Session storage với Redis
- Giảm tải 70-80% read operations từ MongoDB
- Cache invalidation strategy

2. Microservices architecture:

- Tách thành các services độc lập
- API Gateway với Kong/Traefik
- Service discovery với Consul
- Message queue với RabbitMQ

3. Triển khai Sharding thực sự:

- Compound Shard Key: {location: 1, bookCode: 1}
- Config servers và Mongos routers
- Zone-based sharding cho data locality
- Chunk balancing optimization

4.2.3 Phát triển dài hạn

1. Cloud deployment:

- MongoDB Atlas cho managed cluster
- AWS/GCP/Azure auto-scaling
- CDN cho static assets (Cloudflare)
- Load balancer với Nginx/HAProxy

2. Mobile applications:

- iOS/Android app với React Native
- Push notifications cho due date reminders
- QR code scanning cho quick book lookup
- Offline mode với local SQLite

3. Tích hợp AI/ML:

- Recommendation engine cho book suggestions
- Demand forecasting để optimize inventory
- Chatbot hỗ trợ tra cứu sách (OpenAI/Claude)
- OCR cho digitization của sách giấy

4. Analytics platform:

- Data warehouse với MongoDB Analytics
- Business Intelligence dashboards
- User behavior analysis
- A/B testing infrastructure

TÀI LIỆU THAM KHẢO

Tài liệu MongoDB

- [1] MongoDB Inc. (2025). *MongoDB Manual - Sharding*. <https://www.mongodb.com/docs/manual/sharding/>
- [2] MongoDB Inc. (2025). *MongoDB Manual - Replication*. <https://www.mongodb.com/docs/manual/replication/>
- [3] MongoDB Inc. (2025). *MongoDB Manual - Aggregation Pipeline*. <https://www.mongodb.com/docs/manual/aggregation/>
- [4] MongoDB Inc. (2025). *MongoDB Manual - Map-Reduce*. <https://www.mongodb.com/docs/manual/core/map-reduce/>
- [5] MongoDB Inc. (2025). *MongoDB Manual - Indexes*. <https://www.mongodb.com/docs/manual/indexes/>

Tài liệu PHP và Development

- [6] The PHP Group. (2025). *PHP Manual - MongoDB Driver*. <https://www.php.net/manual/en/set.mongodb.php>
- [7] MongoDB Inc. (2025). *mongodb/mongodb PHP Library Documentation*. <https://www.mongodb.com/docs/php-library/current/>
- [8] Docker Inc. (2025). *Docker Compose Documentation*. <https://docs.docker.com/compose/>
- [9] Firebase. (2025). *PHP-JWT Library*. <https://github.com/firebase/php-jwt>
- [10] Chart.js Contributors. (2025). *Chart.js Documentation v4.4*. <https://www.chartjs.org/docs/latest/>

Sách tham khảo

- [11] Bradshaw, S., Brazil, E., & Chodorow, K. (2019). *MongoDB: The Definitive Guide* (3rd ed.). O'Reilly Media. ISBN: 978-1-4919-5446-1.
- [12] Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media. ISBN: 978-1-4493-7332-0.

- [13] Sadalage, P. J., & Fowler, M. (2012). *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Addison-Wesley. ISBN: 978-0-321-82662-6.
- [14] Tanenbaum, A. S., & Van Steen, M. (2017). *Distributed Systems: Principles and Paradigms* (3rd ed.). Pearson. ISBN: 978-1-5309-4117-3.

Bài báo khoa học

- [15] Gilbert, S., & Lynch, N. (2002). Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. *ACM SIGACT News*, 33(2), 51-59.
- [16] Ongaro, D., & Ousterhout, J. (2014). In Search of an Understandable Consensus Algorithm. *Proceedings of the 2014 USENIX ATC*, 305-319.
- [17] DeCandia, G., et al. (2007). Dynamo: Amazon's Highly Available Key-value Store. *Proceedings of SOSP'07*, 205-220.

Tiêu chuẩn và RFC

- [18] Jones, M., Bradley, J., & Sakimura, N. (2015). *RFC 7519 - JSON Web Token (JWT)*. IETF. <https://tools.ietf.org/html/rfc7519>

Giáo trình

- [19] Nguyễn Duy Hải. (2025). *Bài giảng Cơ sở dữ liệu tiên tiến - NoSQL & Distributed Systems*. Trường Đại học Sư phạm Hà Nội.

Phụ lục A

PHỤ LỤC

A.1 Bảng ký hiệu và chữ viết tắt

Bảng A.1: Bảng từ viết tắt và ký hiệu

| STT | Từ viết tắt | Ý nghĩa |
|-----|-------------|--|
| 1 | API | Application Programming Interface - Giao diện lập trình ứng dụng |
| 2 | BSON | Binary JSON - Định dạng nhị phân của JSON |
| 3 | CAP | Consistency, Availability, Partition Tolerance - Định lý CAP |
| 4 | CLI | Command Line Interface - Giao diện dòng lệnh |
| 5 | CRUD | Create, Read, Update, Delete - Các thao tác cơ bản |
| 6 | CSDL | Cơ sở dữ liệu |
| 7 | CSS | Cascading Style Sheets - Ngôn ngữ định kiểu |
| 8 | HA | High Availability - Tính sẵn sàng cao |
| 9 | HTML | HyperText Markup Language - Ngôn ngữ đánh dấu |
| 10 | HTTP(S) | HyperText Transfer Protocol (Secure) - Giao thức truyền tải |
| 11 | JSON | JavaScript Object Notation - Định dạng trao đổi dữ liệu |
| 12 | JWT | JSON Web Token - Token xác thực dạng JSON |
| 13 | MVC | Model-View-Controller - Mô hình thiết kế |
| 14 | NoSQL | Not Only SQL - Cơ sở dữ liệu phi quan hệ |
| 15 | PHP | PHP: Hypertext Preprocessor - Ngôn ngữ lập trình web |
| 16 | RBAC | Role-Based Access Control - Phân quyền theo vai trò |

| STT | Từ viết tắt | Ý nghĩa |
|-----|-------------|---|
| 17 | REST | Representational State Transfer - Kiến trúc API |
| 18 | SQL | Structured Query Language - Ngôn ngữ truy vấn |
| 19 | TLS/SSL | Transport Layer Security/Secure Sockets Layer - Bảo mật tầng vận chuyển |
| 20 | URL | Uniform Resource Locator - Địa chỉ tài nguyên |
| 21 | GUI | Graphical User Interface - Giao diện đồ họa |
| 22 | AJAX | Asynchronous JavaScript and XML - Kỹ thuật web không đồng bộ |

A.2 Mã nguồn quan trọng

A.2.1 JWTHelper.php - Xác thực JWT

```

1 <?php
2 require_once 'vendor/autoload.php';
3 use Firebase\JWT\JWT;
4 use Firebase\JWT\Key;
5
6 class JWTHelper {
7     private static $secret_key = "elibrary_secret_key_2025";
8     private static $issuer = "elibrary_system";
9     private static $algorithm = 'HS256';
10
11    public static function generateToken($userData) {
12        $issuedAt = time();
13        $expirationTime = $issuedAt + (24 * 60 * 60); // 24 hours
14
15        $payload = [
16            'iss' => self::$issuer,
17            'iat' => $issuedAt,
18            'exp' => $expirationTime,
19            'nbf' => $issuedAt,
20            'data' => [
21                'id' => (string)$userData['_id'],
22                'username' => $userData['username'],
23                'role' => $userData['role'] ?? 'customer',
24                'fullName' => $userData['fullName'] ?? ''
25            ]
26        ];
27
28        return JWT::encode($payload, self::$secret_key, self::
29        $algorithm);
30    }
31
32    public static function validateToken($token) {
33        try {
34            $decoded = JWT::decode($token, new Key(self::$secret_key,
self::$algorithm));
35            return (array)$decoded;
36        }
37    }
38
39    public static function decodeToken($token) {
40        $key = new Key(self::$secret_key);
41        $decoded = JWT::decode($token, $key, self::$algorithm);
42        return (array)$decoded;
43    }
44
45    public static function encodeToken($payload, $key, $algorithm) {
46        $jwt = JWT::encode($payload, $key, $algorithm);
47        return $jwt;
48    }
49
50    public static function decodeJWT($token) {
51        $key = new Key(self::$secret_key);
52        $decoded = JWT::decode($token, $key, self::$algorithm);
53        return (array)$decoded;
54    }
55
56    public static function verifyJWT($token) {
57        $key = new Key(self::$secret_key);
58        $decoded = JWT::decode($token, $key, self::$algorithm);
59        return (array)$decoded;
60    }
61
62    public static function signJWT($payload, $key, $algorithm) {
63        $jwt = JWT::encode($payload, $key, $algorithm);
64        return $jwt;
65    }
66
67    public static function verifySignature($token) {
68        $key = new Key(self::$secret_key);
69        $decoded = JWT::decode($token, $key, self::$algorithm);
70        return (array)$decoded;
71    }
72
73    public static function verifySignatureWithKey($token, $key) {
74        $decoded = JWT::decode($token, $key, self::$algorithm);
75        return (array)$decoded;
76    }
77
78    public static function verifySignatureWithAlgorithm($token, $key, $algorithm) {
79        $decoded = JWT::decode($token, $key, $algorithm);
80        return (array)$decoded;
81    }
82
83    public static function verifySignatureWithAlgorithmAndKey($token, $key, $algorithm) {
84        $decoded = JWT::decode($token, $key, $algorithm);
85        return (array)$decoded;
86    }
87
88    public static function verifySignatureWithAlgorithmAndKeyAndSignature($token, $key, $algorithm, $signature) {
89        $decoded = JWT::decode($token, $key, $algorithm, $signature);
90        return (array)$decoded;
91    }
92
93    public static function verifySignatureWithAlgorithmAndKeyAndSignatureAndSignature($token, $key, $algorithm, $signature, $signature2) {
94        $decoded = JWT::decode($token, $key, $algorithm, $signature, $signature2);
95        return (array)$decoded;
96    }
97
98    public static function verifySignatureWithAlgorithmAndKeyAndSignatureAndSignatureAndSignature($token, $key, $algorithm, $signature, $signature2, $signature3) {
99        $decoded = JWT::decode($token, $key, $algorithm, $signature, $signature2, $signature3);
100       return (array)$decoded;
101   }
102 }
```

```

35         } catch (Exception $e) {
36             return null;
37         }
38     }

39     public static function requireAuth() {
40         $headers = getallheaders();
41         $authHeader = $headers['Authorization'] ?? '';
42

43         if (preg_match('/Bearer\s+(.*)$/i', $authHeader, $matches)) {
44             $token = $matches[1];
45             $decoded = self::validateToken($token);
46             if ($decoded) {
47                 return (array)$decoded['data'];
48             }
49         }

50         http_response_code(401);
51         echo json_encode(['error' => 'Unauthorized']);
52         exit;
53     }
54 }
55
56 }
```

Listing A.1: JWTHelper.php - Class xử lý JWT authentication

A.2.2 Connection.php - Kết nối MongoDB Replica Set

```

1 <?php
2 require 'vendor/autoload.php';
3 use MongoDB\Client;
4 use MongoDB\Driver\ReadPreference;
5 use MongoDB\Driver\WriteConcern;
6
7 $Database = "Nhasach";
8
9 // Connection Mode: 'standalone', 'replicaset', 'sharded'
10 $mode = 'replicaset';
11
12 switch ($mode) {
13     case 'replicaset':
14         $uri = "mongodb://mongo1:27017,mongo2:27018,mongo3:27019/?"
15         replicaSet=rs0";
16         $options = [
17             'readPreference' => 'primaryPreferred',
18             'w' => 'majority',
19             'journal' => true
20         ];
21         break;
22     case 'sharded':
23         $uri = "mongodb://mongos1:27017,mongos2:27018";
24         $options = ['w' => 'majority'];
25         break;
26     default:
27         $uri = "mongodb://localhost:27017";
28         $options = [];
29 }
```

```

30 try {
31     $conn = new Client($uri, $options);
32     $db = $conn->$Database;
33 } catch (Exception $e) {
34     die("MongoDB connection failed: " . $e->getMessage());
35 }

```

Listing A.2: Connection.php - Kết nối với MongoDB Replica Set

A.2.3 docker-compose.yml - Cấu hình Docker

```

1 version: '3.8'
2 services:
3     mongo1:
4         image: mongo:7.0
5         container_name: mongo1
6         ports:
7             - "27017:27017"
8         volumes:
9             - mongo1_data:/data/db
10        command: mongod --replSet rs0 --bind_ip_all
11        networks:
12            - mongo-net
13
14     mongo2:
15         image: mongo:7.0
16         container_name: mongo2
17         ports:
18             - "27018:27017"
19         volumes:
20             - mongo2_data:/data/db
21         command: mongod --replSet rs0 --bind_ip_all
22         networks:
23             - mongo-net
24
25     mongo3:
26         image: mongo:7.0
27         container_name: mongo3
28         ports:
29             - "27019:27017"
30         volumes:
31             - mongo3_data:/data/db
32         command: mongod --replSet rs0 --bind_ip_all
33         networks:
34             - mongo-net
35
36 networks:
37     mongo-net:
38         driver: bridge
39
40 volumes:
41     mongo1_data:
42     mongo2_data:
43     mongo3_data:

```

Listing A.3: docker-compose.yml - Cấu hình MongoDB Replica Set

A.2.4 start_system.sh - Script khởi động hệ thống

```

1 #!/bin/bash
2 echo "==== Starting e-Library Distributed System ==="
3
4 # 1. Start MongoDB Replica Set
5 echo "[1/4] Starting MongoDB containers..."
6 cd /Users/tuannghiat/Downloads/Final\ CSDLTT
7 docker-compose up -d
8
9 # Wait for MongoDB to be ready
10 echo "Waiting for MongoDB to initialize..."
11 sleep 10
12
13 # 2. Initialize Replica Set
14 echo "[2/4] Initializing Replica Set..."
15 docker exec mongo1 mongosh --eval '
16 rs.initiate({
17     _id: "rs0",
18     members: [
19         { _id: 0, host: "mongo1:27017", priority: 2 },
20         { _id: 1, host: "mongo2:27017", priority: 1 },
21         { _id: 2, host: "mongo3:27017", priority: 1 }
22     ]
23 })
24 ,
25
26 sleep 5
27
28 # 3. Start PHP servers for each node
29 echo "[3/4] Starting PHP servers..."
30 php -S localhost:8001 -t Nhasach/ &
31 php -S localhost:8002 -t NhasachHaNoi/ &
32 php -S localhost:8003 -t NhasachDaNang/ &
33 php -S localhost:8004 -t NhasachHoChiMinh/ &
34
35 # 4. Display access URLs
36 echo "[4/4] System started successfully!"
37 echo ""
38 echo "==== Access URLs ==="
39 echo "Central Hub: http://localhost:8001"
40 echo "Ha Noi: http://localhost:8002"
41 echo "Da Nang: http://localhost:8003"
42 echo "Ho Chi Minh: http://localhost:8004"
43 echo ""
44 echo "MongoDB: mongodb://localhost:27017"
45 echo "====="

```

Listing A.4: start_system.sh - Script khởi động toàn bộ hệ thống

A.3 Cấu trúc thư mục dự án

```

1 Final CSDLTT/
2 |-- docker-compose.yml           # MongoDB Replica Set config
3 |-- start_system.sh             # Startup script
4 |-- CLAUDE.md                  # Project documentation

```

```

5 |-- PROJECT_STATUS.md                      # Status tracking
6 |
7 |-- Nhasach/                                # Central Hub (Port 8001)
8   |-- Connection.php                         # MongoDB connection
9   |-- JWTHelper.php                          # JWT authentication
10  |-- init_indexes.php                     # Database indexes
11  |-- createadmin.php                       # Create admin user
12  |-- composer.json                         # PHP dependencies
13  |-- vendor/                               # Composer packages
14  |-- php/
15    |-- trangchu.php                        # Homepage
16    |-- dangnhap.php                        # Login page
17    |-- danhsachsach.php                   # Book list
18    |-- giohang.php                         # Shopping cart
19    |-- quanlysach.php                      # Book management
20    |-- quanlynguoidung.php                 # User management
21    '-- dashboard.php                       # Statistics dashboard
22 '-- api/
23   |-- statistics.php                      # REST API endpoints
24   |-- mapreduce.php                       # Aggregation Pipeline
25   |-- books.php                           # Map-Reduce operations
26   |-- users.php                           # Book CRUD
27   '-- orders.php                          # User CRUD
28 '-- css/
29 '-- js/                                   # Order processing
30                                         # Stylesheets
31                                         # JavaScript files
32 |
33 '-- NhasachHaNoi/                         # Ha Noi Branch (Port 8002)
34 '-- NhasachDaNang/                        # Da Nang Branch (Port 8003)
35 '-- NhasachHoChiMinh/                     # Ho Chi Minh Branch (Port 8004)
36 |
37 '-- Data MONGODB export .json/           # Data exports
38 '-- scripts/
39   |-- benchmark_real.js                  # Utility scripts
40   '-- init-replicaset.sh                # Benchmark script
41                                         # RS initialization
42 |
43 '-- screenshots/                          # UI screenshots
44 '-- report_latex/                        # LaTeX report files

```

Listing A.5: Cấu trúc thư mục của dự án

A.4 Thông tin đăng nhập hệ thống

Bảng A.2: Thông tin đăng nhập các node

| Node | Port | Admin | Customer | Password |
|-------------|------|----------|--------------------|----------|
| Central Hub | 8001 | admin | testcustomer | 123456 |
| Hà Nội | 8002 | adminHN | annv, tuanngchia | 123456 |
| Dà Nẵng | 8003 | adminDN | linhhtt, phuongltt | 123456 |
| Hồ Chí Minh | 8004 | adminHCM | huynq, yennt | 123456 |

Bảng A.3: Thông tin kết nối MongoDB

| Thành phần | Thông tin |
|------------------|--|
| Replica Set Name | rs0 |
| Primary | mongo1:27017 |
| Secondary 1 | mongo2:27018 |
| Secondary 2 | mongo3:27019 |
| Connection URI | mongodb://mongo1:27017,mongo2:27018,mongo3:27019/?replicaSet=rs0 |