KỊCH BẢN TRÌNH BÀY ĐỒ ÁN CUỐI KỲ - 15 PHÚT

Đề tài: Xây dựng hệ thống e-Library phân tán nhiều cơ sở

Thời gian: 15 phút
Cấu trúc: 3 phần chính + Demo (10 phút lý thuyết + 5 phút demo)

🎯 PHẦN 1: TỔNG QUAN VÀ BỐI CẢNH (3 phút)

Slide 1-7

Mở đầu (30 giây)

> "Kính chào thầy và các bạn! Nhóm 10 xin phép trình bày đồ án cuối kỳ môn Cơ sở dữ liệu tiên tiến với đề tài **'Xây dựng hệ thống e-Library phân tán nhiều cơ sở'**. Nhóm gồm 3 thành viên: Trương Tuấn Nghĩa, Phạm Mạnh Thắng và Lưu Anh Tú."

1.1. Bối cảnh và Giải pháp (Slide 4-5) ⏱️ 1 phút

Bối cảnh:

> "Trong thời đại số hóa hiện nay, các thư viện truyền thống đang đối mặt với 3 thách thức lớn:
>
> - **Dữ liệu lớn** với hàng nghìn đầu sách phân tán ở nhiều chi nhánh
> - **Yêu cầu nhất quán** về danh mục sách giữa các cơ sở
> - **Quản lý phức tạp** với nhiều giao dịch mượn/trả đồng thời"

Giải pháp của nhóm:

> "Chúng em đề xuất hệ thống e-Library phân tán với 4 ưu điểm vượt trội:
>
> 1. **Phân tán và đồng bộ** - Kiến trúc 4 node: 1 Central Hub + 3 chi nhánh (Hà Nội, Đà Nẵng, TP.HCM)
> 2. **Tính thống nhất nghiệp vụ** - Đồng bộ danh mục sách tự động giữa các cơ sở
> 3. **Sẵn sàng cao** - MongoDB Replica Set với automatic failover trong 10-15 giây
> 4. **Hiệu năng tối ưu** - Full-text search + 7 indexes, thời gian phản hồi trung bình 3.43ms"

1.2. Yêu cầu hệ thống (Slide 6) ⏱️ 45 giây

> "Hệ thống được thiết kế đáp ứng 5 yêu cầu cốt lõi:
>
> 1. **Quản lý đầy đủ** - Sách, người dùng, đơn mượn, lịch sử giao dịch
> 2. **Kiến trúc phân tán** - 3 node địa lý (Hà Nội – Đà Nẵng – TP.HCM) + 1 Central Hub
> 3. **Tối ưu truy vấn** - Index + Full-text search NoSQL
> 4. **Đồng bộ tự động** - Khi có giao dịch mượn/trả
> 5. **Công nghệ** - MongoDB Replica Set với replication lag chỉ 50-200ms"

1.3. Quy trình nghiệp vụ (Slide 7) ⏱️ 1 phút 15 giây

Proposal trọng điểm: Nhấn mạnh 5 quy trình chính

> "Hệ thống hỗ trợ 5 quy trình nghiệp vụ chính:

1. Đăng nhập & Phân quyền

> - JWT authentication với thời hạn 24 giờ
> - RBAC: Admin (quản lý toàn bộ) vs Customer (mượn sách)
> - Password hashing với bcrypt cost 12

2. Tìm kiếm sách

- Full-text search với TEXT index trên 3 trường: bookName + author + publisher
- Hỗ trợ tiếng Việt có dấu
- Lọc theo thể loại, chi nhánh, trạng thái

3. Quản lý người dùng

- Admin xem thông tin, lịch sử giao dịch
- Nạp tiền vào tài khoản
- Theo dõi số dư và hoạt động

4. Quản lý sách

- Central Hub: CRUD đầy đủ, đồng bộ xuống chi nhánh
- Chi nhánh: Xem, tìm kiếm, báo cáo số lượng thất thoát

5. Mượn/Trả sách - 2 luồng:

- **Luồng 1:** Đơn mượn → Thanh toán → Nhận sách → Trả sách (paid → success → returned)
- **Luồng 2:** Đơn mượn → Thanh toán → Hủy đơn (paid → cancelled)"

🏗️ PHẦN 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG (5 phút)

Slide 8-21

2.1. Yêu cầu chức năng (Slide 10) ⏱️ 45 giây

"Hệ thống triển khai 10 chức năng chính được phân quyền rõ ràng:

- **Customer:** Đăng ký, đăng nhập, tìm kiếm, giỏ hàng, đặt mượn, lịch sử
- **Admin:** Quản lý sách (CRUD), quản lý người dùng, Dashboard thống kê
- **System:** Đồng bộ dữ liệu tự động giữa các chi nhánh"

2.2. Thiết kế CSDL (Slide 11-15) ⏱️ 2 phút

Proposal trọng điểm: Giải thích chi tiết 4 collection và mối quan hệ

> "Chúng em thiết kế 4 collection chính với mối quan hệ rõ ràng:

Collection 1: users (Slide 12)

- Lưu thông tin người dùng: username (unique), password (bcrypt), role, balance
- Ví dụ: user 'ducpm' có balance 232,000đ, role customer, thuộc chi nhánh Hà Nội

Collection 2: books (Slide 13)

- Danh mục sách: bookCode (unique), bookGroup, bookName, location, quantity
- Ví dụ: Sách 'Vật Lý Lượng Tử' mã 90001, giá 2000đ/ngày, còn 50 cuốn tại Hà Nội

Collection 3: carts (Slide 14)

- Giỏ hàng: user_id + items[] (embedded array)
- Mỗi item chứa: book_id, bookCode, bookName, pricePerDay, quantity
- Session-based, tự động xóa sau khi thanh toán

Collection 4: orders (Slide 15)

- Đơn mượn: user_id, username, items[], total_amount, status
- Status workflow: paid → success → returned (hoặc paid → cancelled)
- Lưu timestamp: created_at, confirmed_at, returned_at

Mối quan hệ:

- users ↔ carts: **1:1** (một người một giỏ)
- users ↔ orders: **1:N** (một người nhiều đơn)
- carts ↔ books: **N:M** (embedded trong items[])
- orders ↔ books: **N:M** (embedded trong items[])"

2.3. Tối ưu truy vấn (Slide 16) ⏱️ 1 phút

Proposal trọng điểm: Giải thích chi tiết 7 indexes

> "Để tối ưu hiệu năng, chúng em thiết kế 7 indexes:

1. idx_bookCode_unique - UNIQUE

> - Đảm bảo mã sách duy nhất, tránh trùng lặp

2. idx_location_bookName_unique - COMPOUND UNIQUE

> - Tìm nhanh theo vị trí + tên sách
> - Ví dụ: {location: 'Hà Nội', bookName: 'Vật lý'}

3. idx_bookGroup - SINGLE FIELD

> - Lọc theo nhóm sách (Khoa học, Văn học, Lịch sử...)

4. idx_location - SINGLE FIELD

> - Chỉ lấy sách của chi nhánh hiện tại

5. idx_status - SINGLE FIELD

> - Chỉ lấy sách đang hoạt động (status: 'active')

6. idx_borrowCount_desc - DESCENDING

> - Xếp hạng sách mượn nhiều nhất

7. idx_books_text_search - TEXT INDEX

> - Full-text search trên bookName + author + publisher
> - Hỗ trợ tiếng Việt: 'vật lý' tìm được 'Vật Lý Lượng Tử'

Kết quả: Giảm thời gian truy vấn từ O(n) xuống O(log n), không cần scan toàn bộ collection"

2.4. Kiến trúc phân tán (Slide 18-21) ⏱️ 1 phút 15 giây

Proposal trọng điểm: Giải thích chi tiết Hybrid Architecture

> "Chúng em triển khai kiến trúc **Hybrid: 1 Standalone + 3-node Replica Set**

Cấu trúc:

> - **mongo1 (port 27017):** Nhasach - STANDALONE (Central Hub)
> - **mongo2 (port 27018):** NhasachHaNoi - PRIMARY (rs0)
> - **mongo3 (port 27019):** NhasachDaNang - SECONDARY (rs0)
> - **mongo4 (port 27020):** NhasachHoChiMinh - SECONDARY (rs0)

Cơ chế hoạt động (Slide 20):

> 1. **Ghi dữ liệu:** Mọi thao tác INSERT/UPDATE/DELETE thực hiện ở PRIMARY
> 2. **Đồng bộ tự động:** PRIMARY replicate sang 2 SECONDARY với lag 50-200ms
> 3. **Đọc dữ liệu:** SECONDARY phục vụ truy vấn SELECT, giảm tải cho PRIMARY
> 4. **Failover tự động:** Nếu PRIMARY chết, 1 SECONDARY được bầu làm PRIMARY trong 10-15 giây

Chiến lược đồng bộ:

> - **Replica Set (rs0):** Chỉ đồng bộ ORDERS giữa 3 chi nhánh
> - **Books & Users:** Độc lập theo từng chi nhánh, đồng bộ thủ công qua API khi cần"

🧪 PHẦN 3: CÀI ĐẶT VÀ ĐÁNH GIÁ (2 phút)

Slide 22-32

3.1. Công nghệ sử dụng (Slide 23) ⏱️ 20 giây

> "Hệ thống được xây dựng với stack công nghệ:
>
> - **Backend:** PHP 8.4 + MongoDB Driver 1.20
> - **Database:** MongoDB 8.0.16 Community Edition
> - **Container:** Docker Compose cho Replica Set
> - **Frontend:** HTML5 + Bootstrap 5 + Chart.js 4.4
> - **Security:** JWT authentication + bcrypt password hashing"

3.2. Kịch bản kiểm thử (Slide 25-30) ⏱️ 1 phút 40 giây

Proposal trọng điểm: Trình bày chi tiết 7 kịch bản test

> "Chúng em đã thực hiện 7 kịch bản kiểm thử toàn diện:

Kịch bản 1: Đăng nhập & Bảo mật (Slide 25)

- Test JWT token generation với claims đầy đủ (iss, iat, exp, nbf)
- Verify password hashing với bcrypt
- Kiểm tra RBAC: admin vs customer permissions
- **Kết quả:** Token hợp lệ 24h, password không thể reverse

Kịch bản 2: Hiển thị dữ liệu (Slide 26)

- Kiểm tra dữ liệu đúng theo từng database
- **Kết quả thực tế:**
  - Central Hub (8001): 509 sách, 42 users, 111 đơn
  - Hà Nội (8002): 200 sách, 16 users, 54 đơn
  - Đà Nẵng (8003): 163 sách, 10 users, 12 đơn
  - TP.HCM (8004): 146 sách, 10 users, 10 đơn
  - **Tổng:** 1,018 sách, 78 users, 187 đơn

Kịch bản 3: Full-text Search (Slide 27)

- Test tìm kiếm tiếng Việt có dấu
- Ví dụ: Tìm 'vật lý' → Kết quả: 'Vật Lý Lượng Tử Nhập Môn'
- **Kết quả:** TEXT index hoạt động chính xác, thời gian < 5ms

Kịch bản 4: Workflow đơn mượn (Slide 28)

- **Test 4.1:** paid → cancelled (sinh viên hủy đơn)
- **Test 4.2:** paid → success → returned (quy trình hoàn chỉnh)
- **Kết quả:** State transition chính xác, số dư được cập nhật đúng

Kịch bản 5: Ghi và Đồng bộ (Slide 29)

- Thêm sách mới tại PRIMARY (mongo2)
- Kiểm tra xuất hiện tại SECONDARY (mongo3, mongo4)

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

Kịch bản 6: Failover (Slide 30)

- Stop PRIMARY (mongo2) bằng `docker stop mongo2`
- Đợi election (10-15 giây)
- Kiểm tra SECONDARY được bầu làm PRIMARY mới
- Restart node cũ: `docker start mongo2`
- **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

Kịch bản 7: Benchmark hiệu năng (Slide 31)

- 10 loại truy vấn, mỗi loại 50 iterations
- Môi trường: MongoDB 8.0.16, 509 sách, 42 users
- **Kết quả:**
  - Truy vấn nhanh nhất: 0.880ms (Range Query + Sort)
  - Truy vấn chậm nhất: 6.460ms (Single Location Query)
  - Trung bình: 3.43ms
  - Throughput: 1,136 operations/second
  - **Đánh giá:** Phù hợp cho hệ thống quy mô vừa với vài trăm users đồng thời"

3.3. Đánh giá tổng quan (Slide 32) ⏱️ Không trình bày (chỉ tham khảo)

🎬 PHẦN 4: DEMO THỰC TẾ (5 phút)

Slide 24 + Live Demo

4.1. Chuẩn bị Demo ⏱️ 30 giây

> "Bây giờ chúng em xin phép demo hệ thống đang chạy thực tế. Hệ thống đã được khởi động với 4 node MongoDB và 4 PHP servers."

Mở trước:

- Terminal: `docker ps` để show 4 containers
- Browser tabs:
  - Tab 1: http://localhost:8001 (Central Hub)
  - Tab 2: http://localhost:8002 (Hà Nội)
  - Tab 3: MongoDB Compass
  - Tab 4: VSCode với code quan trọng

4.2. Demo Code - Kiến trúc Backend ⏱️ 1 phút 30 giây

Proposal trọng điểm: Show 3 file code quan trọng

File 1: Connection.php (30 giây)

```
// Mở VSCode: Nhasach/Connection.php
```

> "Đây là file kết nối MongoDB với 3 chế độ:
>
> - **Standalone:** Cho Central Hub (mongo1:27017)
> - **Replica Set:** Cho 3 chi nhánh (mongo2,mongo3,mongo4 với rs0)
> - **Sharded:** Dự phòng cho tương lai
>
> Chúng em cấu hình:
>
> - readPreference: 'primaryPreferred' - Ưu tiên đọc từ PRIMARY

> - writeConcern: 'majority' - Đảm bảo ghi thành công trên đa số nodes
> - journal: true - Ghi vào disk trước khi confirm"

File 2: JWTHelper.php (30 giây)

```
// Mở VSCode: Nhasach/JWTHelper.php
```

> "Class xử lý JWT authentication:
>
> - **generateToken():** Tạo token với payload chứa user_id, username, role, thời hạn 24h
> - **validateToken():** Verify token với secret key
> - **requireAuth():** Middleware bắt buộc đăng nhập, trả 401 nếu không hợp lệ
>
> Security: Secret key 'elibrary_secret_key_2025', thuật toán HS256"

File 3: api/statistics.php (30 giây)

```
// Mở VSCode: Nhasach/api/statistics.php
```

> "API thống kê sử dụng Aggregation Pipeline:
>
> - **$match:** Lọc theo điều kiện
> - **$group:** Nhóm theo bookGroup, location
> - **$lookup:** JOIN giữa orders và users
> - **$facet:** Multiple sub-pipelines trong 1 query
> - **$bucket:** Phân nhóm theo khoảng giá
>
> Ví dụ: Endpoint /api/statistics.php?type=books_by_category trả về phân bố sách theo thể loại"

4.3. Demo Giao diện - Luồng Customer ⏱️ 1 phút 30 giây

Proposal trọng điểm: Demo đầy đủ quy trình mượn sách

Bước 1: Đăng nhập (15 giây)

> "Tôi đăng nhập với tài khoản customer 'tuannghia' / '123456' tại chi nhánh Hà Nội (localhost:8002)"

- Nhập username/password
- Click "Đăng nhập"
- Redirect về trang chủ với menu customer

Bước 2: Tìm kiếm sách (20 giây)

> "Vào 'Danh sách sách', tôi tìm kiếm 'vật lý'"

- Click menu "Danh sách sách"
- Nhập "vật lý" vào ô tìm kiếm
- Kết quả hiển thị: "Vật Lý Lượng Tử Nhập Môn", "Cơ Học Cổ Điển"

Bước 3: Thêm vào giỏ hàng (15 giây)

> "Chọn sách 'Vật Lý Lượng Tử', số lượng 1, số ngày mượn 3, thêm vào giỏ"

- Click "Thêm vào giỏ"
- Notification: "Đã thêm vào giỏ hàng"

Bước 4: Xem giỏ hàng (15 giây)

> "Vào 'Giỏ hàng', kiểm tra sách đã chọn"

- Click menu "Giỏ hàng"
- Hiển thị: 1 sách, tổng tiền = 2000đ/ngày × 3 ngày = 6000đ

Bước 5: Thanh toán (25 giây)

> "Click 'Thanh toán', hệ thống kiểm tra số dư và tạo đơn mượn"

- Click "Thanh toán"
- Popup xác nhận: "Bạn có chắc muốn mượn?"
- Click "Xác nhận"
- Success: "Đặt mượn thành công! Mã đơn: #12345"
- Số dư giảm từ 50,000đ → 44,000đ

4.4. Demo Giao diện - Luồng Admin ⏱️ 1 phút 30 giây

Proposal trọng điểm: Demo Dashboard + Quản lý đơn mượn

Bước 1: Đăng nhập Admin (10 giây)

> "Logout customer, đăng nhập admin 'adminHN' / '123456'"

- Logout
- Login với adminHN
- Redirect về Dashboard

Bước 2: Dashboard thống kê (30 giây)

> "Dashboard hiển thị 6 loại biểu đồ với Chart.js"

- **Tổng quan:** 200 sách, 16 users, 54 đơn, doanh thu 1.2M
- **Biểu đồ tròn:** Phân bố sách theo thể loại (Khoa học 35%, Văn học 25%...)
- **Biểu đồ cột:** So sánh số sách giữa các chi nhánh
- **Biểu đồ đường:** Xu hướng mượn sách theo tháng (tăng dần)
- **Top 5:** Sách mượn nhiều nhất, người dùng active nhất

Bước 3: Quản lý đơn mượn (30 giây)

> "Vào 'Quản lý đơn mượn', xử lý đơn vừa tạo"

- Click menu "Quản lý đơn mượn"
- Tìm đơn #12345 với status "paid"
- Click "Xác nhận nhận sách"
- Status chuyển: paid → success
- Thông báo: "Đã xác nhận đơn mượn"

Bước 4: Cập nhật trả sách (20 giây)

> "Giả lập sinh viên trả sách"

- Tìm lại đơn #12345 với status "success"
- Click "Xác nhận trả sách"
- Status chuyển: success → returned
- Số lượng sách tăng lên: 49 → 50

4.5. Demo MongoDB Compass ⏱️ 30 giây

Proposal trọng điểm: Show dữ liệu thực tế trong database

> "Cuối cùng, tôi mở MongoDB Compass để xem dữ liệu backend"

Show:

1. **Connection:** mongodb://localhost:27018 (Hà Nội - PRIMARY)
2. **Database:** NhasachHaNoi
3. **Collection orders:**
   - Tìm đơn #12345
   - Show document với status "returned"
   - Highlight các trường: user_id, items[], total_amount, timestamps
4. **Indexes tab:**
   - Show 7 indexes đã tạo
   - Highlight TEXT index: idx_books_text_search

🎤 KẾT LUẬN (30 giây)

> "Tóm lại, nhóm em đã hoàn thành đồ án với các kết quả chính:
>
> ✅ **Kiến trúc phân tán:** 4 node MongoDB (1 Standalone + 3-node Replica Set)
>
> ✅ **Chức năng đầy đủ:** 10 chức năng nghiệp vụ với CRUD, tìm kiếm, thống kê
>
> ✅ **Hiệu năng tốt:** Thời gian phản hồi trung bình 3.43ms, throughput 1,136 ops/s
>
> ✅ **Bảo mật:** JWT + bcrypt + RBAC + CSRF protection
>
> ✅ **Sẵn sàng cao:** Failover tự động trong 10-15 giây
>
> Nhóm 10 xin cảm ơn thầy và các bạn đã lắng nghe! Em sẵn sàng trả lời câu hỏi."

📋 PHỤ LỤC: CHECKLIST TRƯỚC KHI TRÌNH BÀY

Chuẩn bị kỹ thuật:

- [ ] Khởi động hệ thống: `./start_system.sh`
- [ ] Kiểm tra 4 containers: `docker ps`
- [ ] Kiểm tra Replica Set: `docker exec mongo2 mongosh --eval "rs.status()"`
- [ ] Test 4 URLs:
  - [ ] http://localhost:8001 (Central Hub)
  - [ ] http://localhost:8002 (Hà Nội)
  - [ ] http://localhost:8003 (Đà Nẵng)
  - [ ] http://localhost:8004 (TP.HCM)

Chuẩn bị tài khoản test:

- **Customer:** tuannghia / 123456 (Hà Nội)
- **Admin:** adminHN / 123456 (Hà Nội)

Chuẩn bị browser tabs:

1. Slides PDF (fullscreen)
2. localhost:8001 (Central Hub)
3. localhost:8002 (Hà Nội)
4. MongoDB Compass (connected to mongo2:27018)
5. VSCode (mở 3 files: Connection.php, JWTHelper.php, statistics.php)
6. Terminal (docker ps)

Chuẩ bị dữ liệu:

- [ ] Có ít nhất 5 sách trong database Hà Nội
- [ ] Tài khoản tuannghia có số dư > 50,000đ
- [ ] Giỏ hàng rỗng trước khi demo

⏱️ PHÂN BỔ THỜI GIAN CHI TIẾT

Phần	Nội dung	Thời gian
Phần 1	Tổng quan & Bối cảnh	3:00
1.1	Bối cảnh & Giải pháp	1:00
1.2	Yêu cầu hệ thống	0:45
1.3	Quy trình nghiệp vụ	1:15
Phần 2	Phân tích & Thiết kế	5:00
2.1	Yêu cầu chức năng	0:45
2.2	Thiết kế CSDL	2:00
2.3	Tối ưu truy vấn	1:00
2.4	Kiến trúc phân tán	1:15
Phần 3	Cài đặt & Đánh giá	2:00
3.1	Công nghệ	0:20
3.2	Kịch bản kiểm thử	1:40
Phần 4	Demo thực tế	5:00
4.1	Chuẩn bị	0:30
4.2	Demo Code	1:30
4.3	Demo Customer	1:30
4.4	Demo Admin	1:30
4.5	MongoDB Compass	0:30
Kết luận	Tóm tắt & Q&A	0:30
TỔNG		**15:30**

💡 GỢI Ý TRẢ LỜI CÂU HỎI THƯỜNG GẶP

Q1: Tại sao dùng Hybrid (Standalone + Replica Set) thay vì toàn bộ Replica Set?

> "Em thiết kế Hybrid vì:
>
> - Central Hub (mongo1) là master data, không cần high availability
> - 3 chi nhánh (mongo2-4) cần đồng bộ ORDERS real-time → dùng Replica Set
> - Tiết kiệm tài nguyên: Standalone nhẹ hơn Replica Set
> - Linh hoạt: Có thể chuyển sang full Replica Set sau nếu cần"

Q2: Replication lag 50-200ms có ảnh hưởng gì không?

> "Lag 50-200ms là chấp nhận được vì:
>
> - Hệ thống thư viện không yêu cầu real-time tuyệt đối
> - Eventual consistency phù hợp với nghiệp vụ mượn sách
> - Nếu cần strong consistency, em có thể dùng writeConcern: 'majority' + readConcern: 'majority'"

Q3: Làm sao xử lý conflict khi 2 chi nhánh cùng mượn sách cuối cùng?

> "Em xử lý bằng:
>
> - Mỗi chi nhánh có database riêng, sách độc lập
> - Chỉ đồng bộ danh mục sách (metadata), không đồng bộ quantity
> - Nếu cần đồng bộ quantity, em sẽ dùng:
>   - Optimistic locking với version field
>   - Hoặc distributed lock với Redis"

Q4: Tại sao không dùng Sharding?

> "Em đã chuẩn bị code cho Sharding (có trong Connection.php) nhưng chưa triển khai vì:
>
> - Dataset hiện tại chỉ 1,018 sách, chưa đủ lớn để cần Sharding

> - MongoDB khuyến nghị Sharding khi data > 100GB hoặc > 1M documents
> - Nếu triển khai, em sẽ dùng shard key: {location: 1, bookCode: 1} để zone-based sharding"

Q5: Bảo mật JWT có đủ không? Có cần refresh token?

> "Hiện tại em dùng JWT với thời hạn 24h, phù hợp cho MVP. Để production, em sẽ:
>
> - Thêm refresh token với thời hạn 7 ngày
> - Lưu refresh token trong httpOnly cookie
> - Implement token rotation để tăng bảo mật
> - Thêm blacklist cho revoked tokens"

🎯 ĐIỂM NHẤN QUAN TRỌNG CẦN NHỚ

1. **Con số ấn tượng:**
   - 1,018 sách, 78 users, 187 đơn mượn
   - 7 indexes, 10 chức năng, 4 node MongoDB
   - Thời gian phản hồi: 3.43ms
   - Throughput: 1,136 ops/s
   - Failover: 10-15 giây
2. **Công nghệ nổi bật:**
   - MongoDB 8.0.16 Replica Set
   - JWT authentication + bcrypt
   - Full-text search với TEXT index
   - Docker Compose orchestration
   - Aggregation Pipeline với $facet$,bucket
3. **Ưu điểm vượt trội:**
   - Phân tán địa lý 3 vùng
   - Đồng bộ tự động với lag < 200ms
   - Failover tự động trong 10-15s
   - Bảo mật đa lớp (JWT + bcrypt + RBAC + CSRF)
   - Hiệu năng cao với indexes tối ưu

Chúc bạn trình bày thành công! 🚀