**Câu 4: bài tập ứng dụng**

**1.Tại sao tách Web App và Mobile App?**

**- Về mặt người sử dụng:** Sinh viên, độc giả, thủ thư có thể dùng laptop/PC hoặc smartphone.

- Web App phù hợp cho thủ thư/admin → giao diện bảng, quản lý nhiều dữ liệu (catalog, báo cáo, kho sách).

- Mobile App phù hợp cho độc giả → tìm kiếm, mượn/gia hạn sách nhanh, quét QR code trong thư viện.

**2. Tại sao dùng SPA (Single Page Application) cho Web App?**

**Hiệu năng & UX:** SPA (Angular/React/Vue) cho phép chuyển trang nhanh, trải nghiệm mượt.

**Real-time update:** SPA dễ tích hợp WebSocket/SignalR để cập nhật trạng thái mượn sách hoặc phí trễ hạn theo thời gian thực.

**Khả năng tích hợp:** SPA gọi RESTful API hoặc GraphQL từ backend dễ dàng.

**3. Tại sao tách Payment Service riêng?**

**Lý do bảo mật:** Không nên để backend thư viện trực tiếp xử lý thẻ tín dụng → cần một dịch vụ tích hợp (PayPal, MoMo, Stripe).

**Khả năng tái sử dụng:** Sau này có thể dùng Payment Service chung cho các hệ thống khác (ví dụ: đóng góp cho thư viện).

**Giảm rủi ro:** Dễ thay đổi cổng thanh toán (MoMo → VNPay → Stripe) mà không ảnh hưởng core system.

**4. Tại sao chọn Database tập trung (SQL/NoSQL)?**

**Tính toàn vẹn dữ liệu:** Quản lý kho sách, trạng thái mượn/trả cần quan hệ chặt chẽ → **RDBMS (SQL Server, PostgreSQL, MySQL)**.

**Khả năng mở rộng:** Nếu cần quản lý log tìm kiếm hoặc hành vi người dùng, có thể bổ sung **NoSQL (MongoDB/ElasticSearch)**.

**5. Tại sao phân tách Component trong Web App?**

**Module tìm kiếm (Search Module):** tách riêng để dễ tích hợp ElasticSearch hoặc API tìm kiếm nâng cao.

**Module mượn/trả (Borrow/Return Module):** core của hệ thống, có logic về hạn mượn, phí trễ hạn.

**Module quản lý tài khoản (Account Management):** phục vụ xác thực, phân quyền (Người đọc vs Thủ thư).

**6. Bảo mật & hiệu năng**

**Authentication:** JWT/OAuth2 để quản lý session người dùng.

**Authorization:** Role-based Access Control (RBAC):

Người đọc chỉ được tìm kiếm, mượn sách.

Thủ thư/Admin có quyền duyệt mượn/trả, quản lý sách.

**Caching:** Redis cache để giảm tải truy vấn tìm kiếm.

**Scalability:** Backend chạy theo microservices hoặc modular monolith, dễ mở rộng bằng Docker/Kubernetes.