**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO ĐỒ ÁN MÔN HỌC:**

**KIỂM THỬ PHẦN MỀM**

**Năm học 2025-2026**

**Tên đề tài:**

**Bài Tập Lab3**

**Giảng viên hướng dẫn: TS. Đỗ Như Tài**

Các thành viên trong nhóm:

1. Nguyễn Hoàn Báu - 3122411017

2. Võ Phương Liên Chi - 3122411022

3. Thái Thị Huỳnh Như - 3122411143

**Thành phố Hồ Chí Minh, năm 2025**

# **Bảng phân chia công việc**

|  |  |  |
| --- | --- | --- |
| **STT** | **Họ và tên** | **Công việc cần làm** |
| **1** | Nguyễn Hoàn Báu | **-** Bài 1. Vẽ sơ đồ  - Bài 2. Vẽ sơ đồ  - Bài 5. Vẽ sơ đồ và giải thích  - Bài tập ứng dụng:   1. Phân tích yêu cầu nghiệp vụ:  * Mô tả mục tiêu kinh doanh, tác nhân, ràng buộc, giá trị mang lại. * Xác định các use case chính (tìm kiếm sách, mượn sách, trả sách, gia hạn, thanh toán, quản lý kho).  1. Thiết kế dữ liệu (CSDL):  * Xác định các bảng chính: User, Book, Catalog, Loan, Payment. * Thiết lập mối quan hệ giữa các bảng. * Vẽ Class Diagram chi tiết. |
| **2** | Võ Phương Liên Chi | - Bài 3. Vẽ sơ đồ  - Bài 4. Vẽ sơ đồ và giải thích  - Bài 8. Vẽ sơ đồ  - Bài tập ứng dụng:   1. Thiết kế Component (Level 3):  * Phân tích và vẽ sơ đồ các module trong Web App: * Module Tìm kiếm & Danh mục sách. * Module Mượn/Trả/Gia hạn sách. * Module Quản lý tài khoản (User, Reader, Librarian, Admin). * Module Quản lý thanh toán.  1. Vẽ sơ đồ Level 2 (Container): Web App, Mobile App, Database, Payment Service. |
| **3** | Thái Thị Huỳnh Như | - Bài 6. Vẽ sơ đồ và giải thích  - Bài 7. Vẽ sơ đồ  - Bài tập ứng dụng:   1. Thiết kế Code (Level 4 – Implementation):  * Mô tả thuộc tính, phương thức, quan hệ kế thừa/association.  1. Triển khai mô phỏng:  * Viết code mô phỏng các lớp chính * Tạo một vài tình huống test nhỏ (mượn sách, trả sách, tính phí trễ hạn). * Vẽ sơ đồ Level 1 (Context): Hệ thống và các tác nhân bên ngoài. |

**MỤC LỤC**

[**Bảng phân chia công việc** 2](#_Toc209254987)

[**Bài 1: Vẽ lại sơ đồ kiến trúc SPA (Single Page Application):** 5](#_Toc209254988)

[**Bài 2: Vẽ sơ đồ triển khai CI/CD:** 6](#_Toc209254989)

[**Bài 3: Vẽ sơ đồ API của hệ thống:** 7](#_Toc209254990)

[**Bài 4: Vẽ sơ đồ C1 - System Context và giải thích:** 8](#_Toc209254991)

[**Bài 5: Vẽ sơ đồ C2 – Container và giải thích:** 9](#_Toc209254994)

[**Bài 6: Vẽ sơ đồ C3 – Component (High-Level) và giải thích:** 11](#_Toc209254999)

[**Bài 7: Vẽ sơ đồ C3 – Component (Module-Level) sau:** 13](#_Toc209255007)

[**Bài 8: Vẽ sơ đồ xử lý 1 request và giải thích:** 15](#_Toc209255008)

[**Bài Tập Ứng Dụng** 17](#_Toc209255010)

[**Sơ đồ Context:** 22](#_Toc209255021)

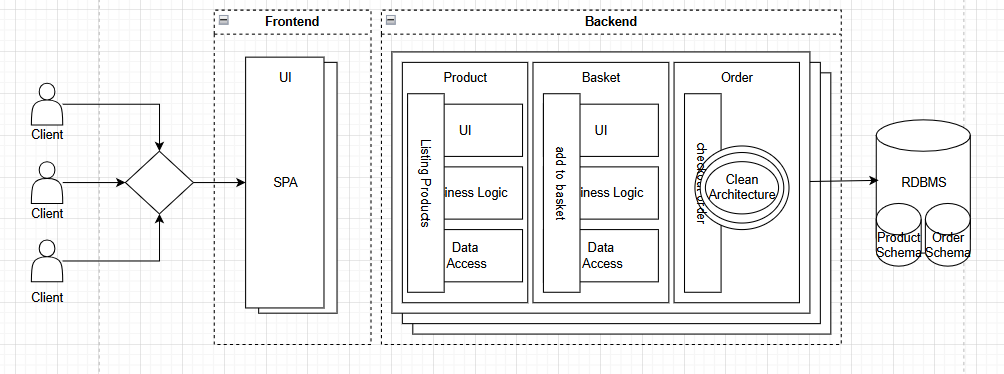
[**Sơ đồ Container:** 23](#_Toc209255022)

[**Sơ đồ Component:** 23](#_Toc209255023)

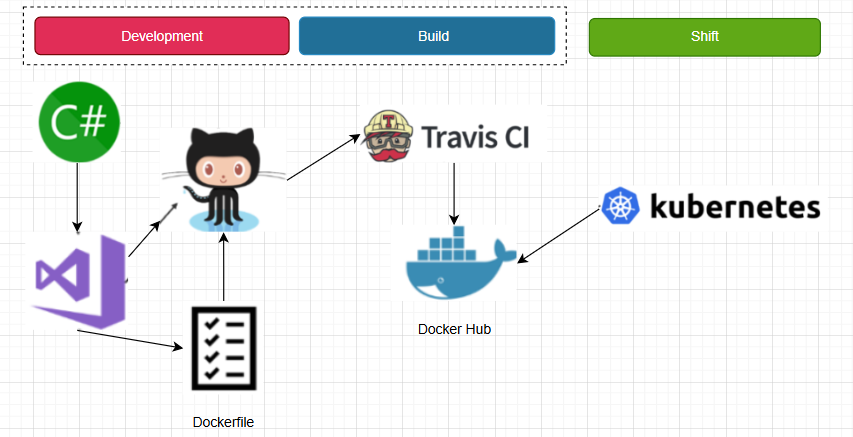
[**Class diagram:** 24](#_Toc209255024)

[**Các quyết định thiết kế chính:** 24](#_Toc209255025)

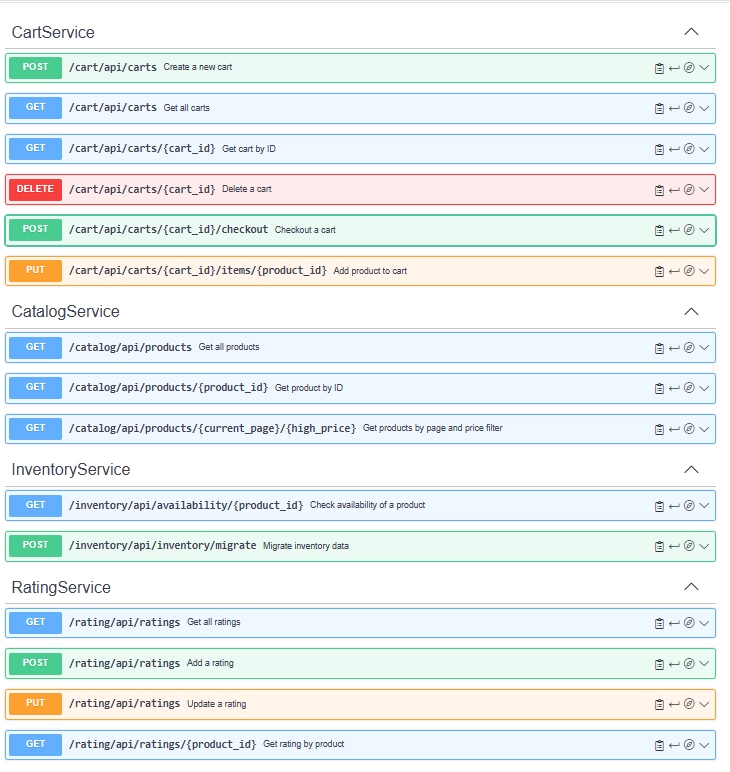
# **Bài 1: Vẽ lại sơ đồ kiến trúc SPA (Single Page Application):**

****

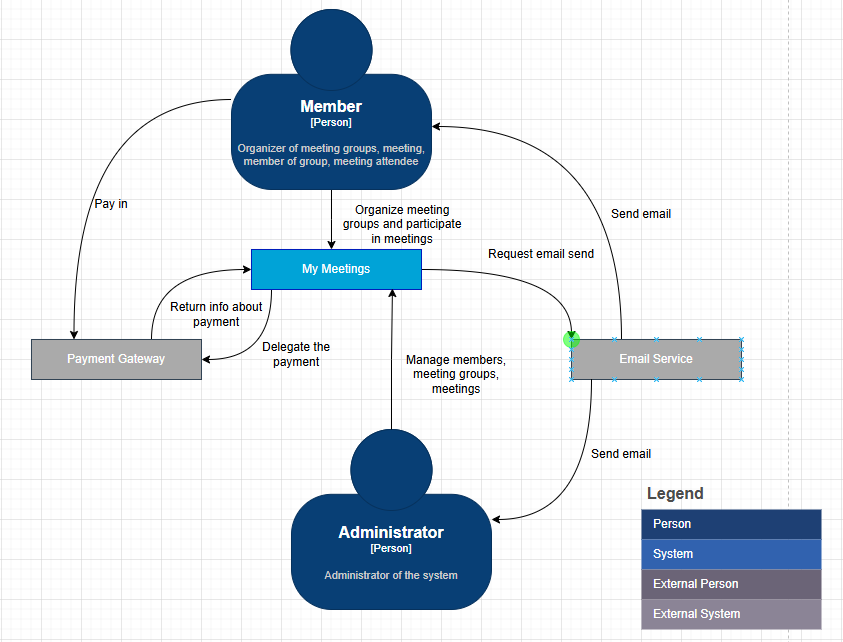
# **Bài 2: Vẽ sơ đồ triển khai CI/CD:**

****

# **Bài 3: Vẽ sơ đồ API của hệ thống:**

****

# **Bài 4: Vẽ sơ đồ C1 - System Context và giải thích:**

****

### Mục đích trong kiểm thử:

* Cho tester cái nhìn toàn cảnh: Hệ thống My Meetings sẽ tương tác với ai và hệ thống bên ngoài nào, từ đó biết được phạm vi kiểm thử tích hợp và các ràng buộc.
* Giúp xác định điểm giao tiếp (interfaces) cần kiểm thử: email, cổng thanh toán, quyền quản trị, thao tác của người dùng.

### Phân tích chi tiết:

* Member (Người dùng thông thường):
* Tổ chức nhóm họp, đăng ký/đăng nhập, tham gia cuộc họp.
* Thực hiện thanh toán (trả phí tham gia hoặc phí dịch vụ).

-> Tester phải kiểm thử: đăng ký, đăng nhập, luồng đặt/đổi lịch họp, luồng thanh toán.

* Administrator (Quản trị viên): Quản lý thành viên, nhóm họp, dữ liệu.

-> Cần kiểm thử: chức năng phân quyền, CRUD dữ liệu nhóm họp, kiểm soát email thông báo.

* Payment Gateway (Cổng thanh toán ngoài): Nhận yêu cầu thanh toán, trả thông tin kết quả.

-> Kiểm thử tích hợp: đảm bảo hệ thống gửi request/nhận response đúng định dạng, xử lý tình huống thất bại.

* Email Service (Dịch vụ email ngoài): Gửi email xác nhận, thông báo.

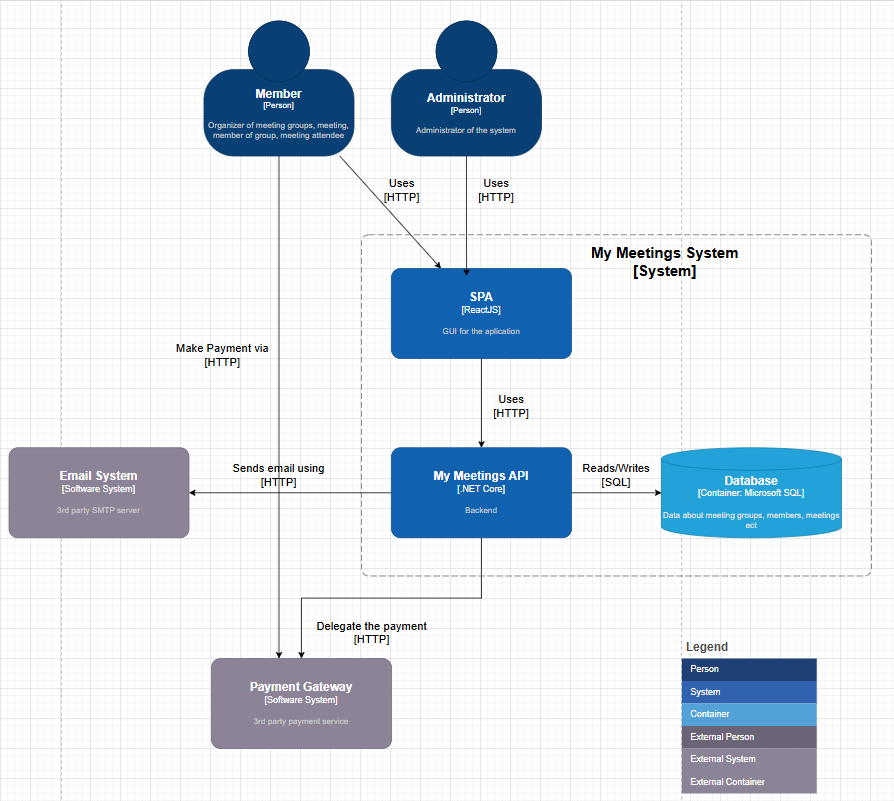
-> Kiểm thử: email được gửi đúng nội dung, đúng người nhận; xử lý lỗi khi server email lỗi.

* My Meetings (Hệ thống chính): Trung tâm giao tiếp giữa các tác nhân.

-> Kiểm thử chức năng end-to-end, cả luồng chính và luồng lỗi.

Ý nghĩa cho tester: C1 giúp lên kế hoạch kiểm thử hệ thống (System Testing) và kiểm thử tích hợp (Integration Testing), biết rõ các điểm cần mô phỏng (mock) như Email Service hay Payment Gateway.

# **Bài 5: Vẽ sơ đồ C2 – Container và giải thích:**

****

### Mục đích trong kiểm thử

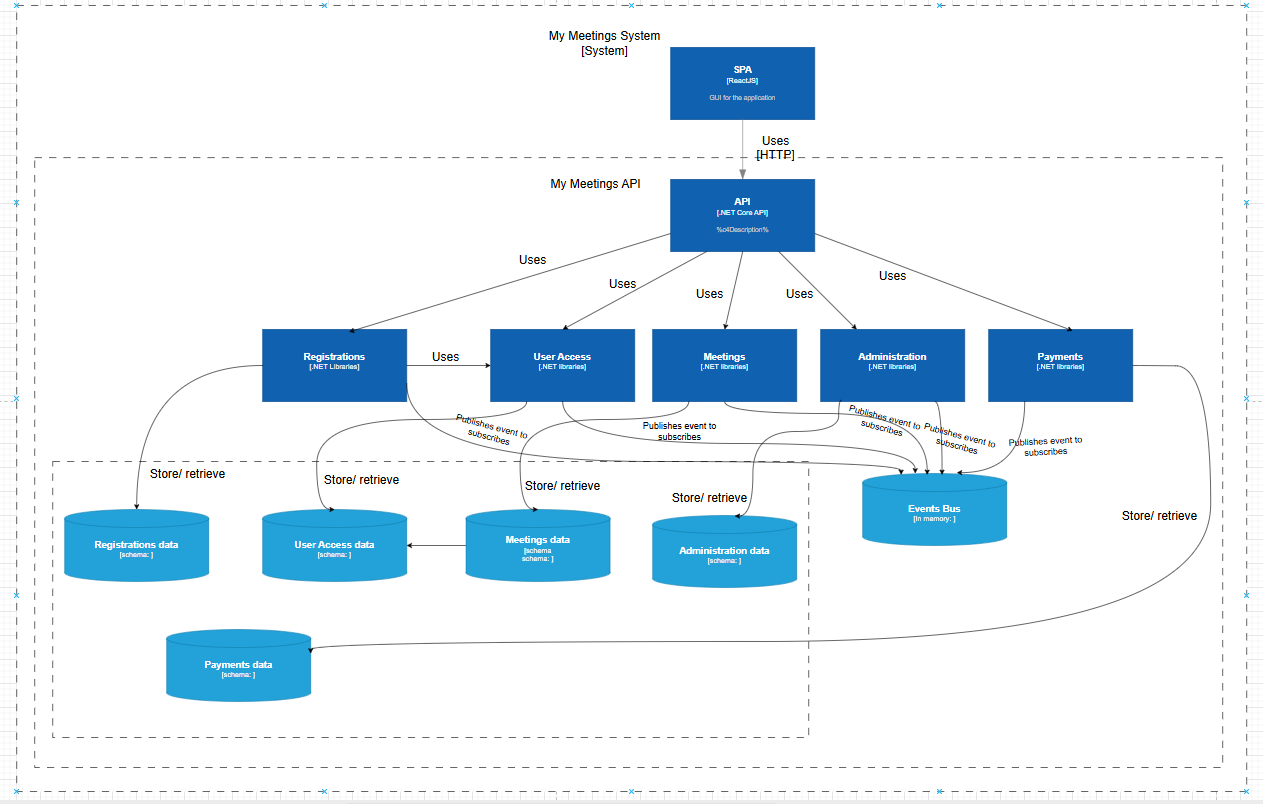
### Bước tiếp theo: đi sâu vào cấu trúc của hệ thống, xác định các khối phần mềm (container) để thiết kế test levels (unit, integration, system).

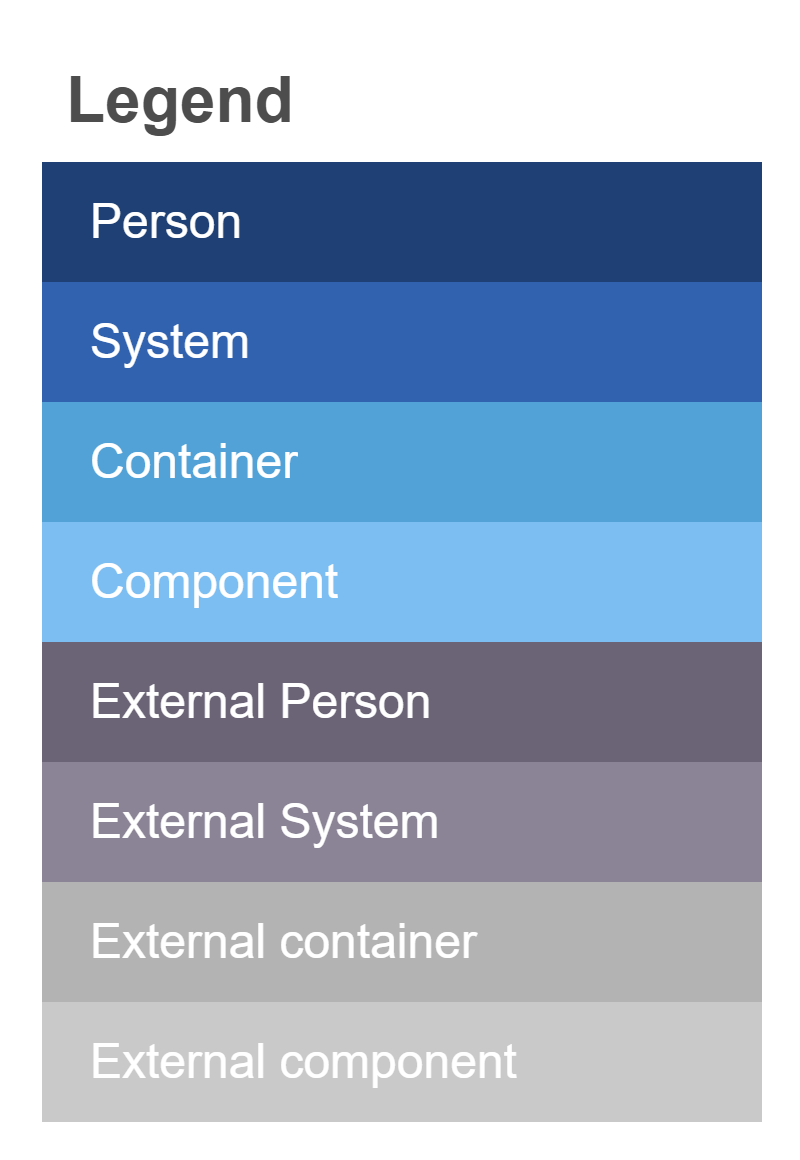
### Cho biết công nghệ và giao thức (HTTP, SQL, SMTP) => định hướng công cụ kiểm thử.

### Chi tiết các container:

* SPA (ReactJS):
* Giao diện người dùng chạy trên trình duyệt.
* Tester: kiểm thử UI/UX, kiểm thử chấp nhận (Acceptance), test client-side validation.
* My Meetings API (.NET Core):
* Xử lý nghiệp vụ, cầu nối giữa SPA và Database/Service ngoài.
* Tester: viết test API (Postman, RestAssured), test logic nghiệp vụ, test lỗi/timeout khi gọi dịch vụ ngoài.
* Database (Microsoft SQL):
* Lưu trữ dữ liệu họp, thành viên, thanh toán.
* Tester: kiểm tra tính toàn vẹn dữ liệu, ràng buộc khóa, backup/restore.
* Email System (SMTP server bên ngoài):
* Gửi email cho người dùng.
* Tester: mock server SMTP để test việc gửi mail.
* Payment Gateway (3rd party):
* Dịch vụ thanh toán.
* Tester: test API với dữ liệu giả lập, kiểm tra các trường hợp thất bại.

# **Bài 6: Vẽ sơ đồ C3 – Component (High-Level) và giải thích:**

****

****

### Mục đích trong kiểm thử

### Đi sâu vào backend quan trọng nhất: My Meetings API.

### Giúp xác định từng thành phần (component) và dữ liệu liên quan => thiết kế unit test, component test.

### Thành phần chi tiết:

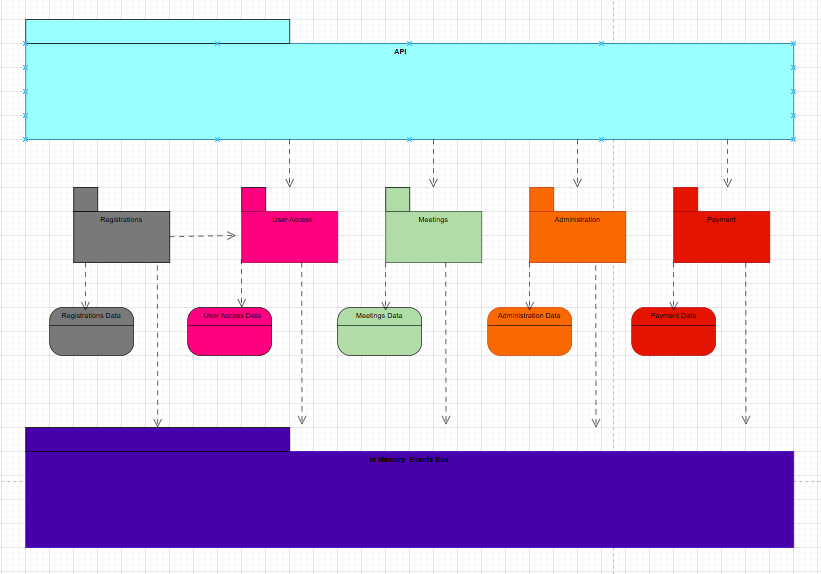
### API (.NET Core API): điểm vào cho SPA.

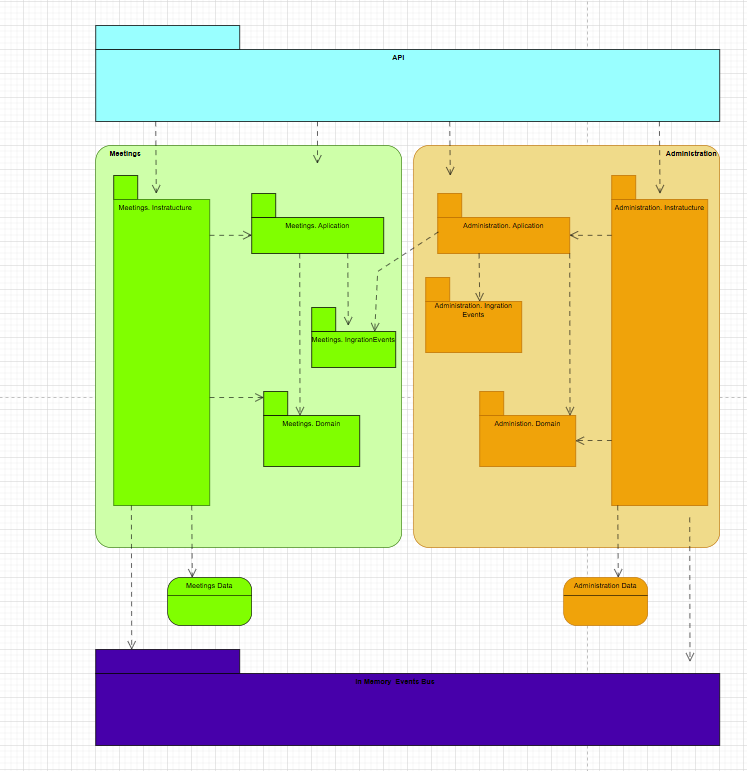
### Tester: kiểm thử API endpoints.

### Registrations (.NET Library):

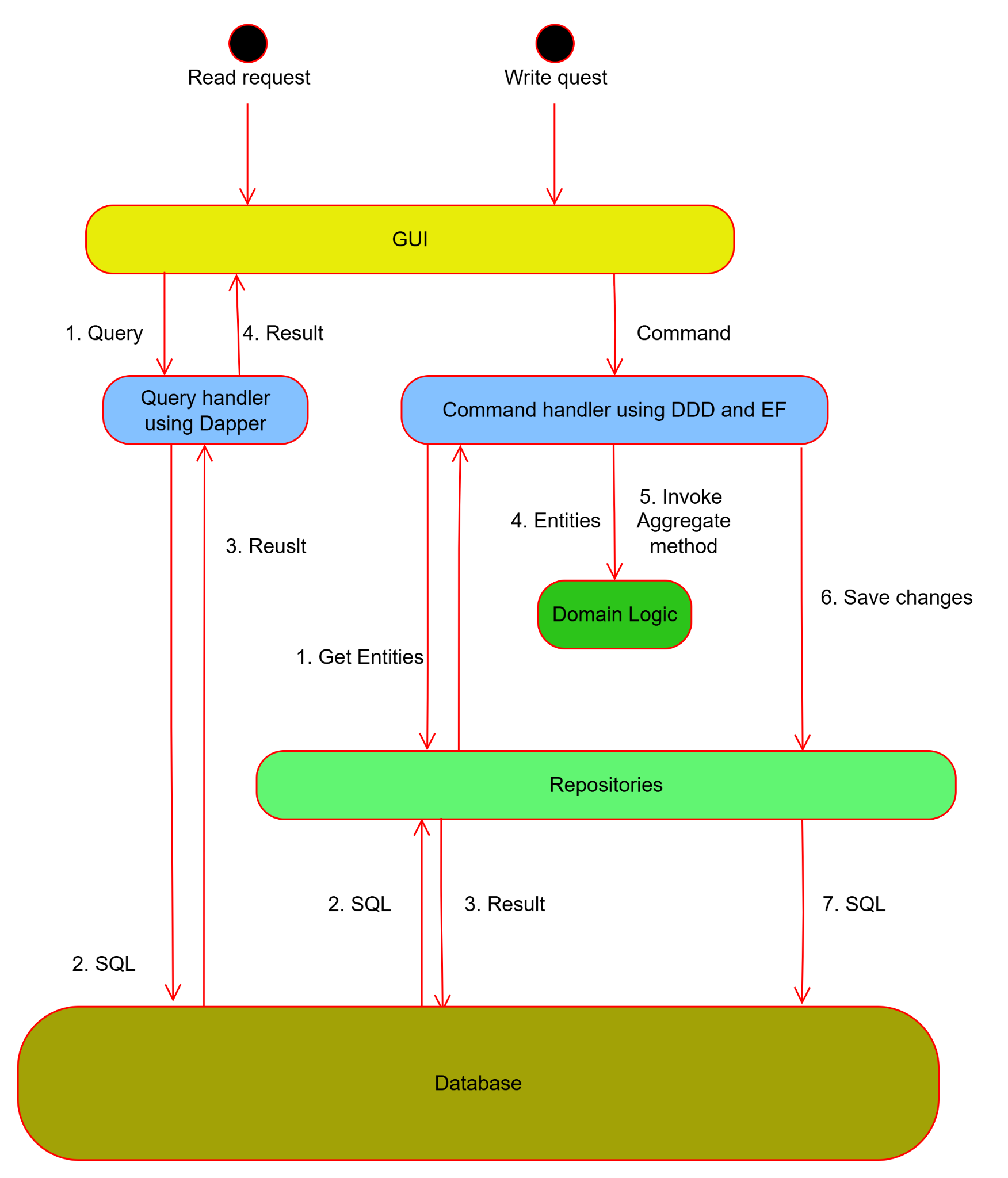
* Xử lý đăng ký, quản lý thông tin người dùng mới.
* Tester: unit test logic đăng ký, test case email trùng, password yếu.
* User Access (.NET Library):
* Quản lý quyền truy cập, xác thực.
* Tester: kiểm tra role-based access control, session, token.
* Meetings (.NET Library):
* Quản lý nhóm họp, lịch họp.
* Tester: kiểm tra tạo/sửa/xóa cuộc họp, kiểm tra giới hạn số người tham gia.
* Administration (.NET Library):
* Chức năng quản trị viên.
* Tester: test tính năng quản lý hệ thống, phân quyền admin.
* Payments (.NET Library):
* Xử lý thanh toán, giao tiếp Payment Gateway.
* Tester: kiểm tra tính đúng đắn của logic thanh toán, xử lý khi Payment Gateway lỗi.
* Events Bus:
* Gửi/nhận sự kiện nội bộ (như thông báo khi thanh toán thành công).
* Tester: test event publishing/subscribing.
* Database:
* Chứa các bảng: Registration data, User Access data, Meetings data, Administration data, Payments data.
* Tester: kiểm tra ràng buộc dữ liệu, test data migration.

# **Bài 7: Vẽ sơ đồ C3 – Component (Module-Level) sau:**

****

****

# **Bài 8: Vẽ sơ đồ xử lý 1 request và giải thích:**

****

### Mục đích trong kiểm thử:

Sơ đồ xử lý request cho thấy toàn bộ đường đi của một yêu cầu từ giao diện đến cơ sở dữ liệu và các dịch vụ liên quan. Đây là cơ sở quan trọng để thiết kế các mức kiểm thử:

* Với Read request (truy vấn): tập trung kiểm thử tính đúng đắn và hiệu năng của dữ liệu trả về.
* Với Write request (ghi dữ liệu): cần kiểm thử logic nghiệp vụ, tính toàn vẹn dữ liệu, khả năng rollback khi lỗi, và việc phát sự kiện sau khi ghi.
* Xác định rõ các điểm có thể phát sinh lỗi (API, handler, database, event bus, dịch vụ ngoài) để xây dựng kịch bản kiểm thử phù hợp.

Chi tiết các container/bước xử lý

* GUI/SPA: giao diện web nơi người dùng gửi yêu cầu. Đây là nơi kiểm thử UI/UX, hành vi gửi request, và xử lý response.
* API Layer: nhận request qua HTTP, thực hiện xác thực và ủy quyền. Tester cần kiểm thử hợp đồng API (contract), định dạng dữ liệu, và các trường hợp sai quyền.
* Query Handler (cho Read): thực hiện truy vấn cơ sở dữ liệu để trả về kết quả. Kiểm thử tập trung vào tính chính xác, thời gian phản hồi, và khả năng xử lý filter/sort.
* Command Handler (cho Write): tiếp nhận lệnh thay đổi dữ liệu, gọi domain logic để kiểm tra quy tắc. Kiểm thử gồm các tình huống đúng, sai, edge cases.
* Domain Logic: chứa các quy tắc nghiệp vụ cốt lõi. Đây là đối tượng chính của unit test, nhằm đảm bảo tuân thủ các bất biến (invariants).
* Repositories: lớp trung gian thao tác với cơ sở dữ liệu. Kiểm thử tập trung vào lưu/đọc dữ liệu, khóa toàn vẹn và rollback khi lỗi.
* Database: nơi lưu trữ dữ liệu. Tester cần kiểm thử constraint, ràng buộc khóa ngoại, và khôi phục (backup/restore).
* Event Bus: nơi publish Integration Event sau khi ghi dữ liệu thành công. Kiểm thử đảm bảo sự kiện được phát đúng, không lặp, và các module khác nhận được để xử lý tiếp (ví dụ: gửi email, tạo invoice).

# **Bài Tập Ứng Dụng**

Thiết kế kiến trúc phần mềm theo chuẩn C4 model theo 4 mức của C4:

1. Level 1 (Context Diagram): Hệ thống và các tác nhân bên ngoài.

**C1 – System Context (tóm lược)**

* Mục tiêu hệ thống: Hỗ trợ người dùng tạo và tham gia các nhóm/ cuộc họp (meeting group/ meetup), cho phép đăng ký tài khoản, đề xuất và duyệt nhóm họp, tạo cuộc họp, tham gia, bình luận và thanh toán các khoản phí (subscription hoặc event fee).
* Tác nhân bên ngoài (Actors)
* Người dùng (End User): đăng ký tài khoản, đề xuất nhóm họp, tạo cuộc họp, tham gia, bình luận, thanh toán phí.
* Cổng thanh toán (Payment Gateway): xử lý giao dịch thanh toán online (phí đăng ký hoặc phí sự kiện).
* Hệ thống email/notification: gửi thông báo cho người dùng.
* Hệ thống theo kiến trúc module hóa cao. Bao gồm các module chính:
* User Access: xác thực, ủy quyền người dùng.
* Registrations: đăng ký tài khoản người dùng mới.
* Meetings: quản lý nhóm họp, cuộc họp, bình luận, quản lý thành viên tham dự.
* Administration: quản lý và duyệt các đề xuất nhóm họp.
* Payments: xử lý thanh toán phí subscription hoặc phí sự kiện.

1. Level 2 (Container Diagram): Các container (ứng dụng web, mobile app, database, service...)

**C2 – Container View**

* Các container chính:
* API (ASP.NET Core REST): Vai trò nhận request HTTP, thực hiện xác thực/ủy quyền thông qua User Access, gửi Command/Query đến module đích.
* Đặc điểm: API mỏng, không chứa business logic.
* User Access Module
* Chức năng: xác thực (authentication) và ủy quyền (authorization).
* Registrations Module
* Chức năng: đăng ký người dùng, quản lý thông tin tài khoản mới.
* Meetings Module
* Chức năng: quản lý nhóm họp, tạo cuộc họp, quản lý thành viên tham gia, bình luận.
* Administration Module
* Chức năng: duyệt/khước từ các đề xuất nhóm họp, xử lý các tác vụ quản trị.
* Payments Module
* Chức năng: xử lý thanh toán cho subscription hoặc phí sự kiện.
* In-Memory Events Bus
* Vai trò: cung cấp cơ chế Pub/Sub cho các module tích hợp bất đồng bộ bằng sự kiện.

### Luồng giao tiếp:

### API ↔ Module qua các interface nhỏ (Command/Query).

### Module ↔ Module chỉ qua In-Memory Events Bus (không gọi trực tiếp method, không đọc chéo DB).

### Mỗi module có schema dữ liệu riêng, độc lập, có thể nâng cấp tách DB riêng.

1. Level 3 (Component Diagram): Các thành phần trong một container quan trọng.

**C3 – Component View (bên trong 1 module điển hình)**

* Application:
* Xử lý use case (Command/Query) theo CQRS.
* Quản lý domain events, integration events, internal commands.
* Domain:
* Mô hình domain theo DDD.
* Entity/Value Object giàu hành vi, encapsulation cao, persistence-ignorant (POCO).
* Tránh “primitive obsession”, đặt tên theo ngôn ngữ nghiệp vụ.
* Infrastructure:
* Kết nối database riêng của module.
* Tương tác Event Bus, nền tảng chạy nền, khởi tạo module.
* IntegrationEvents
* Định nghĩa hợp đồng sự kiện public ra Event Bus để module khác subscribe.
* Là điểm duy nhất module khác được phụ thuộc.

Dòng chảy xử lý (CQRS):

* API nhận HTTP request và xác thực qua User Access.
* API gửi Command/Query tới Application.
* Application gọi Domain thực thi nghiệp vụ, phát Domain Event nội bộ.
* Khi cần tích hợp liên-module, publish Integration Event lên In-Memory Event Bus.

1. Level 4 (Code/Implementation Diagram): Chi tiết lớp hoặc cấu trúc code.

**C4 – Code/Implementation View**

Ví dụ chi tiết cho Meetings Module:

* Application Layer
* CreateMeetingCommandHandler: xử lý yêu cầu tạo cuộc họp.
* JoinMeetingCommandHandler: xử lý tham gia cuộc họp.
* AddCommentCommandHandler: xử lý bình luận.
* MeetingQueryService: query thông tin cuộc họp.
* Domain Layer
* Entities: Meeting, MeetingGroup, MeetingParticipant, Comment.
* Value Objects: MeetingId, ParticipantId, MeetingDateRange.
* Domain Events: MeetingCreated, ParticipantJoined, CommentAdded.
* Infrastructure Layer
* Repository pattern cho MeetingRepository.
* EventPublisher kết nối với In-Memory Event Bus.
* Database context riêng cho module.
* IntegrationEvents
* MeetingCreatedIntegrationEvent.
* ParticipantJoinedIntegrationEvent.

### Nguyên tắc chung và kiểm thử

### Encapsulation cao, public tối thiểu, domain giàu hành vi.

### Mỗi module có Composition Root/IoC riêng, API gọi Initialize(...) để khởi tạo trong Startup.

### Kiểm thử: Unit test cho domain logic, integration test cho giao tiếp module, system integration testing, mutation testing và CI pipeline.

**Mô tả hệ thống:**

Hệ thống Quản lý Thư viện Trực tuyến được xây dựng nhằm hiện đại hóa quy trình mượn/trả sách trong thư viện, giúp giảm phụ thuộc vào thủ thư, tăng trải nghiệm người dùng, đồng thời quản lý kho sách và phí mượn hiệu quả.

**Các tác nhân chính:**

1. Người đọc:

* Đăng ký và đăng nhập hệ thống.
* Tìm kiếm sách theo nhiều tiêu chí (tựa đề, tác giả, thể loại).
* Mượn sách trực tuyến và theo dõi tình trạng mượn.
* Gia hạn sách khi sắp đến hạn trả.
* Trả sách trực tuyến hoặc trực tiếp.
* Xem lịch sử mượn sách và tình trạng phí trễ hạn.

2. Thủ thư / Admin:

* Quản lý kho sách: thêm mới, chỉnh sửa, xóa sách, cập nhật số lượng.
* Quản lý danh mục sách, thể loại, tác giả.
* Xử lý yêu cầu mượn, trả, gia hạn từ người dùng.
* Theo dõi và xử lý phí trễ hạn (gửi thông báo, yêu cầu thanh toán).
* Quản lý tài khoản người dùng và phân quyền.

3. Hệ thống thanh toán:

* Xử lý các giao dịch phí thành viên (membership).
* Xử lý các khoản phí trễ hạn khi người dùng trả sách muộn.
* Gửi phản hồi trạng thái giao dịch (thành công/thất bại) về hệ thống.

## **Chức năng chính của hệ thống**

1. Quản lý người dùng:

* Đăng ký, đăng nhập, xác thực.
* Quản lý vai trò (người đọc, thủ thư, admin).

2. Quản lý sách và kho sách:

* Thêm, sửa, xóa thông tin sách.
* Cập nhật số lượng sách có sẵn.
* Phân loại theo thể loại, tác giả, năm xuất bản.

3. Quản lý mượn,trả và gia hạn:

* Người dùng gửi yêu cầu mượn sách.
* Hệ thống kiểm tra tình trạng sách (còn hay hết).
* Cập nhật ngày mượn, ngày hết hạn, ngày trả.
* Cho phép gia hạn sách (nếu sách không bị đặt trước).
* Theo dõi trạng thái mượn (đang mượn, đã trả, quá hạn).

4. Quản lý phí trễ hạn và thanh toán:

* Tự động tính phí nếu trả sách quá hạn.
* Gửi thông báo nhắc nhở đến người dùng.
* Kết nối cổng thanh toán online để xử lý giao dịch.
* Ghi nhận và cập nhật trạng thái thanh toán.

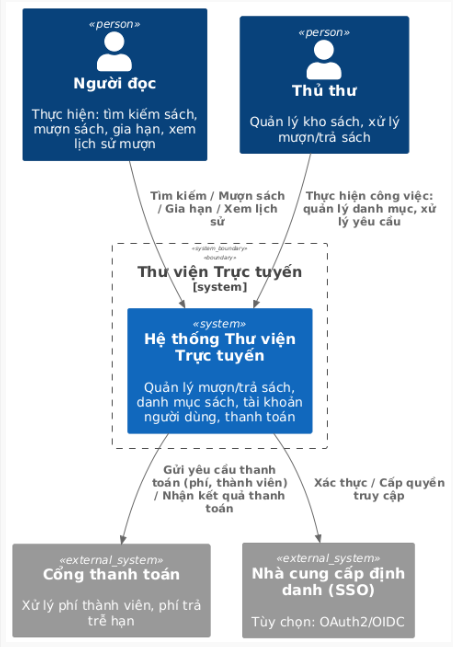
5. Lịch sử và báo cáo:

* Người dùng: xem lịch sử mượn/trả, tình trạng phí.
* Thủ thư: thống kê số lượng sách mượn/trả, người dùng vi phạm, doanh thu từ phí.

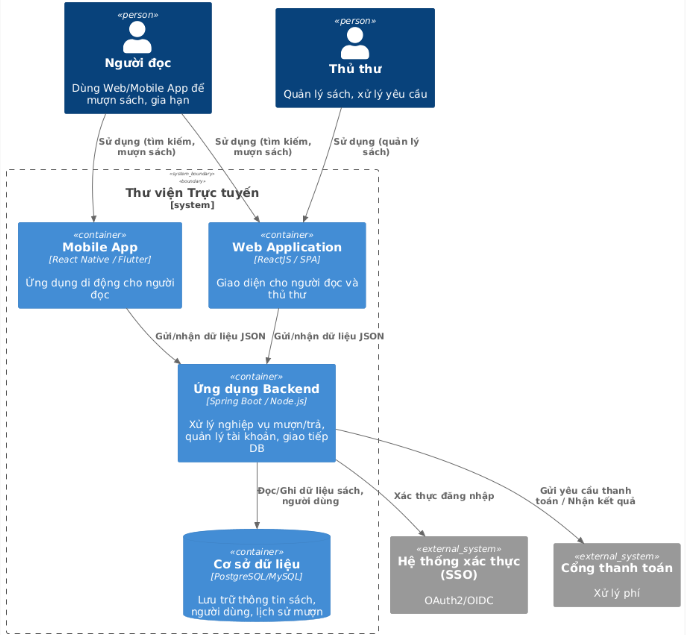
## **Ràng buộc và giá trị mang lại:**

* Tính ổn định: hệ thống hoạt động liên tục, phục vụ nhiều người dùng cùng lúc.
* Bảo mật dữ liệu người đọc: thông tin cá nhân, lịch sử mượn và giao dịch được bảo vệ.
* Khả năng mở rộng: hỗ trợ cả giao diện web và ứng dụng di động.
* Tích hợp dễ dàng với cổng thanh toán để xử lý các khoản phí trực tuyến.
* Trải nghiệm người dùng tốt: giao diện thân thiện, dễ sử dụng cho cả người đọc và thủ thư.

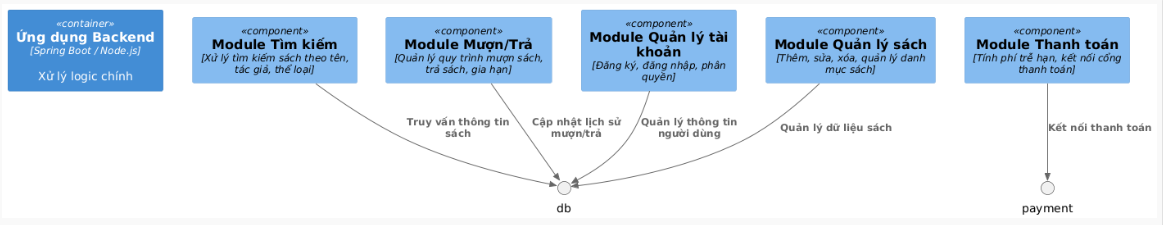
## **Sơ đồ Context:**

****

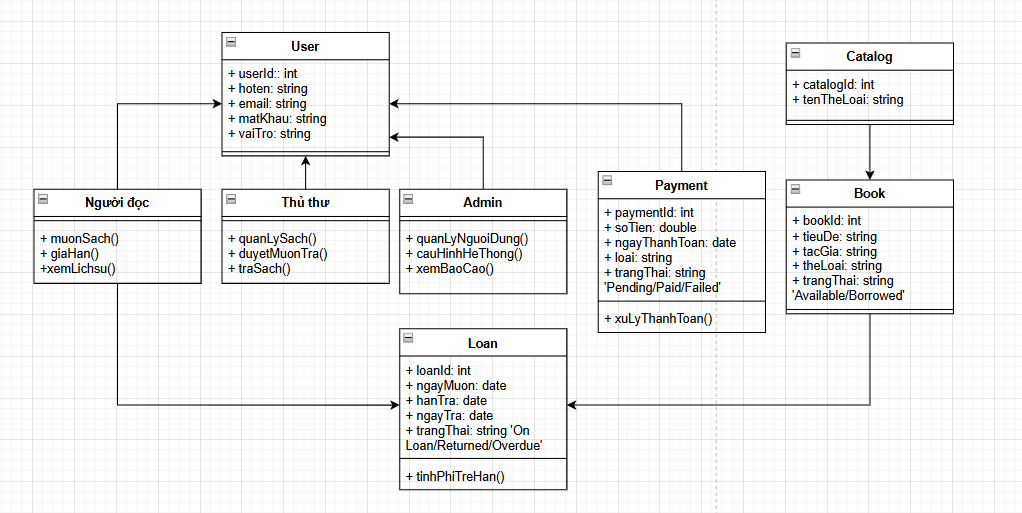
## **Sơ đồ Container:**

****

## **Sơ đồ Component:**

****

## **Class diagram:**

****

## **Các quyết định thiết kế chính:**

1. Sử dụng SPA (Single Page Application) cho Web App để:

* Trải nghiệm người dùng mượt mà, giảm tải khi chỉ cần cập nhật dữ liệu thay vì reload toàn trang.
* Cho phép tách biệt frontend–backend, hỗ trợ phát triển và kiểm thử độc lập.

2. Tách Mobile App và Web App vì:

* Web App: phục vụ đầy đủ tính năng quản trị cho thủ thư/admin.
* Mobile App: tối ưu trải nghiệm nhanh gọn cho người đọc (mượn/gia hạn, quét QR).
* Giúp hệ thống linh hoạt, phù hợp từng bối cảnh sử dụng.
* API stateless dễ mở rộng theo chiều ngang.

3. Tách riêng Backend Service:

* Chứa toàn bộ business logic: mượn – trả – gia hạn, tính phí, xử lý thanh toán.
* Đảm bảo API có thể dùng chung cho cả Web App và Mobile App.
* Có thể triển khai và mở rộng độc lập trong tương lai.

4. Quản lý cơ sở dữ liệu SQL:

* Dữ liệu có cấu trúc và quan hệ rõ ràng (sách, bản sao, người dùng, phiếu mượn).
* SQL đảm bảo tính toàn vẹn và hỗ trợ các ràng buộc dữ liệu mạnh mẽ.
* Tích hợp hệ thống thanh toán bên ngoài
* Giảm rủi ro bảo mật, tiết kiệm chi phí phát triển.

5. Mỗi module quản lý dữ liệu riêng (Data Ownership)

* Các module trao đổi dữ liệu bất đồng bộ qua sự kiện, tránh phụ thuộc chặt chẽ.
* Giúp kiểm thử luồng sự kiện và xử lý idempotent dễ dàng hơn.

6. Tích hợp cổng thanh toán ngoài (Payment Gateway)

* Không tự xây dựng chức năng thanh toán để tránh rủi ro bảo mật.
* Kết nối API của cổng thanh toán (VD: VNPay, Momo, PayPal).
* Hệ thống chỉ cần xử lý kết quả (thành công/thất bại).

7. Khả năng mở rộng và bảo trì

* Thiết kế kiến trúc mô-đun (theo tinh thần DDD + Clean Architecture).
* Các module như Tài khoản, Sách, Mượn/Trả, Thanh toán… hoạt động độc lập nhưng giao tiếp qua API hoặc Event Bus.
* Dễ dàng thay đổi hoặc nâng cấp từng module mà không ảnh hưởng toàn hệ thống.

Qua đó có thể kết luận rằng những quyết định thiết kế trên giúp hệ thống thư viện trực tuyến vừa đạt được mục tiêu kinh doanh, vừa đảm bảo kiến trúc ổn định, mở rộng được và dễ dàng kiểm thử. Nhờ có sự phân tách rõ ràng giữa frontend, backend và các module nội bộ, hệ thống có thể triển khai đầy đủ các mức kiểm thử một cách hiệu quả, từ đó nâng cao chất lượng và độ tin cậy