**A smart printing service for students at HCMUT**

**I. Mô tả yêu cầu**

Hệ thống in thông minh cho sinh viên (HCMUT\_SSPS) được thiết kế để phục vụ sinh viên tại các cơ sở của trường Đại học Bách khoa TP.HCM, cho phép sinh viên in tài liệu của mình.

**1. Mô tả hệ thống**

Hệ thống bao gồm một số máy in được đặt xung quanh các cơ sở của trường. Mỗi máy in đều có thông tin như mã ID, tên hãng sản xuất, mẫu máy in, mô tả ngắn, và vị trí cụ thể (bao gồm tên cơ sở, tòa nhà và số phòng).

**2. Tính năng chính của hệ thống**

**a. In tài liệu:** Sinh viên có thể in tài liệu bằng cách tải file lên hệ thống, chọn máy in, và thiết lập các thuộc tính in như kích thước giấy, số trang cần in, in một mặt hay hai mặt, số bản in, v.v. Các loại file được phép in sẽ bị giới hạn và có thể được cấu hình bởi Quản lý dịch vụ in sinh viên (SPSO).

**b. Lưu lịch sử in ấn:** Hệ thống phải lưu lại toàn bộ các thao tác in ấn của sinh viên, bao gồm mã sinh viên, mã máy in, tên file, thời gian bắt đầu và kết thúc in, và số trang đã in cho từng kích thước giấy.

**c. Xem lịch sử in ấn:** SPSO có thể xem lịch sử in ấn của tất cả sinh viên hoặc của một sinh viên cụ thể trong một khoảng thời gian (từ ngày này đến ngày khác) và có thể xem cho tất cả hoặc chỉ một số máy in. Sinh viên cũng có thể xem lịch sử in ấn của chính mình trong một khoảng thời gian nhất định, cùng với bảng tổng kết số trang đã in cho từng kích thước giấy.

**d. Số lượng trang in mặc định:** Mỗi học kỳ, trường sẽ cấp cho mỗi sinh viên một số lượng trang in khổ A4 mặc định. Sinh viên có thể mua thêm số trang in thông qua tính năng Mua trang in của hệ thống và thanh toán qua hệ thống thanh toán trực tuyến của trường như BKPay. Hệ thống chỉ cho phép sinh viên in khi số trang in không vượt quá số dư trong tài khoản của họ. Lưu ý rằng một trang A3 sẽ được tính tương đương với hai trang A4.

**e. Quản lý máy in:** SPSO có thể quản lý các máy in như thêm mới, kích hoạt hoặc vô hiệu hóa một máy in.

**f. Quản lý cấu hình hệ thống:** SPSO có thể thay đổi các cấu hình khác của hệ thống như thay đổi số trang mặc định, thiết lập ngày mà hệ thống sẽ cấp số trang mặc định cho tất cả sinh viên, hoặc thiết lập các loại file được phép in.

**g. Báo cáo sử dụng:** Hệ thống sẽ tự động tạo các báo cáo về việc sử dụng dịch vụ in ấn vào cuối mỗi tháng và mỗi năm. Các báo cáo này sẽ được lưu trữ trong hệ thống và SPSO có thể xem bất cứ lúc nào.

**h. Xác thực người dùng:** Tất cả người dùng phải xác thực thông qua hệ thống HCMUT\_SSO trước khi sử dụng hệ thống. Hệ thống sẽ được cung cấp qua ứng dụng web và ứng dụng di động.

# **II. Task 1: Requirement elicitation**

## **1. Describe the domain context**

**Yêu cầu:** Describe the domain context of a smart printing service for students at HCMUT. Who are relevant stakeholders? What are their current needs? In your opinion, what benefits HCMUT-SSPS will be for each stakeholder?

**Hints:** Phải viết ít nhất ba đoạn văn (1) một đoạn văn về bối cảnh miền, (2) một đoạn văn về các bên liên quan và nhu cầu của họ, (3) một đoạn văn về lợi ích của HCMUT-SSPS đối với từng bên liên quan. Phần này phải dễ hiểu, tham khảo các nguồn đáng tin cậy và thông tin phải được chứng minh.

|  |
| --- |
| **Ngữ cảnh miền**  Hệ thống in thông minh dành cho sinh viên là dịch vụ cung cấp cho sinh viên khả năng in tài liệu dễ dàng tại một cơ sở của Đại học Bách Khoa Hồ Chí Minh. Hiện nay, học sinh thường phải tìm kiếm, sử dụng tài liệu in ấn bên ngoài trường học, điều này không những tốn thời gian, mà còn không đảm bảo tính bảo mật tài liệu. HCMUT-SSPS ra đời với mục đích giải quyết vấn đề này bằng cách tạo ra giải pháp nội bộ giúp học sinh in tài liệu trực tiếp tại trường thông qua các thiết bị và được giám sát. Hệ thống này sẽ tương tác với hệ thống xác thực hiện tại để đảm bảo tính bảo mật và phân cấp cho hoạt động của người dùng. |

|  |
| --- |
| **Các bên liên quan và nhu cầu của họ**  **Sinh viên:** Họ cần một giải pháp in ấn tiện lợi, nhanh chóng, an toàn mà không cần phải di chuyển hay chờ đợi nhiều. Họ cũng cần một cách để kiểm soát bản in của mình và dễ dàng thực hiện thanh toán thông qua các cổng thanh toán của trường học như BKPay. Đồng thời bảo mật được thông tin, dữ liệu của họ tránh rò rỉ ra bên ngoài trường.  **SPSO** cần một công cụ để quản lý các máy in, giám sát hoạt động in ấn, quản lý tài khoản trang in cho sinh viên, và tạo ra các báo cáo về sử dụng dịch vụ in.  **Nhà trường** cần giải pháp công nghệ để tối ưu hóa dịch vụ in ấn, đảm bảo an toàn thông tin và giảm chi phí vận hành. |

|  |
| --- |
| **Lợi ích của HCMUT-SSPS**  Đối với sinh viên, HCMUT-SSPS cung cấp một dịch vụ in ấn thuận tiện, tiết kiệm thời gian và chi phí, đồng thời hỗ trợ quản lý tài nguyên in của họ một cách dễ dàng.  Đối với SPSO, hệ thống giúp đơn giản hóa việc quản lý, từ việc giám sát các máy in, cấu hình hệ thống, cho đến việc tạo ra các báo cáo sử dụng tự động vào cuối tháng và năm.  Đối với nhà trường, hệ thống sẽ giúp nâng cao hình ảnh của trường trong việc áp dụng công nghệ thông minh vào đời sống sinh viên, cải thiện trải nghiệm của sinh viên, và tăng tính bảo mật cho các tài liệu in. Điều này cũng giảm tải cho nhân viên hành chính và cải thiện quy trình quản lý dịch vụ. |

## **2. Functional and Non-functional requirements**

**Yêu cầu:** Describe all functional and non-functional requirements that can be inferred from the project description.

**Hints:** Ít nhất 05 yêu cầu chức năng cho mỗi bên liên quan. Yêu cầu phải được viết thành câu đơn. Các yêu cầu phải đầy đủ, rõ ràng, nhất quán và chính xác. Sơ đồ trường hợp sử dụng phải đầy đủ và đúng về mặt cú pháp.

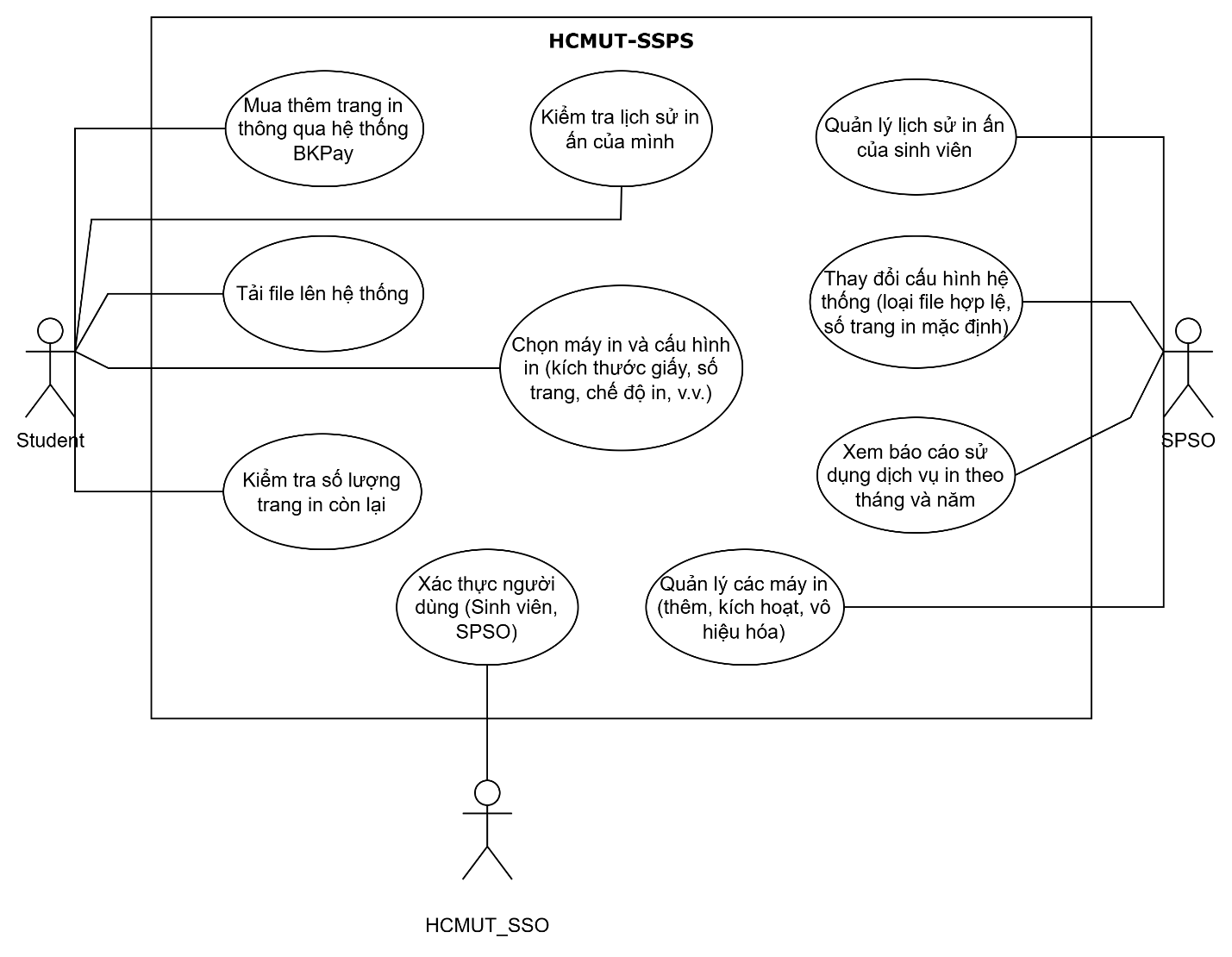
|  |
| --- |
| **Yêu cầu chức năng**  Sinh viên   * Tải file lên hệ thống để in. * Chọn máy in và thiết lập thuộc tính in (kích thước, số trang, số bản in, v.v.). * Mua thêm số trang in thông qua cổng thanh toán trực tuyến BKPay. * Kiểm tra lịch sử in ấn của mình cho một khoảng thời gian nhất định. * Hệ thống sẽ thông báo khi số trang in của sinh viên không đủ để thực hiện thao tác in.   Quản lý dịch vụ in (SPSO)   * Thêm mới, kích hoạt hoặc vô hiệu hóa các máy in. * Thay đổi các loại file được phép in. * Thiết lập và thay đổi số trang in mặc định cho mỗi sinh viên. * Xem và quản lý lịch sử in ấn của tất cả sinh viên. * Xem báo cáo sử dụng dịch vụ in theo tháng và năm.   Hệ thống chung   * Phải xác thực người dùng thông qua HCMUT\_SSO. * Phải cho phép các loại tài liệu có định dạng file hợp lệ được in. * Phải lưu lại thời gian bắt đầu và kết thúc của mỗi lệnh in. * Phải tự động gửi thông báo cho sinh viên khi số dư trang in gần hết. * Phải tạo báo cáo tự động mỗi tháng và mỗi năm về việc sử dụng dịch vụ. |

|  |
| --- |
| **Yêu cầu phi chức năng**   * Hệ thống phải hoạt động liên tục 24/7. * Hệ thống phải có khả năng xử lý tối đa 1000 lệnh in đồng thời. * Hệ thống phải bảo mật thông tin của sinh viên và các tài liệu in. * Hệ thống phải tương thích với các trình duyệt web và hệ điều hành di động phổ biến. * Giao diện của hệ thống phải thân thiện và dễ sử dụng cho sinh viên và quản lý. |

## **3.** **Use-case diagram**

**Yêu cầu:** Draw a use-case diagram for the whole system. Choose an important module and draw its use-case diagram, as well as describe the use-case using a table format.

**Hints:** Phải vẽ một sơ đồ trường hợp sử dụng. Sơ đồ trường hợp sử dụng phải hoàn chỉnh và đúng cú pháp. Một bảng mô tả trường hợp sử dụng cho mỗi trường hợp sử dụng sử dụng ví dụ từ bài giảng. Số bước trong một trường hợp sử dụng phải nhiều hơn 3. Ít nhất một trường hợp sử dụng phải có luồng ngoại lệ.



|  |  |
| --- | --- |
| Use-case Name | **Tải file lên hệ thống** |
| Use-case ID | UC01 |
| Actor | Student |
| Description | Sinh viên tải file lên để in. |
| Pre-condition | Sinh viên đã đăng nhập thông qua HCMUT\_SSO. |
| Flow | 1. Sinh viên đăng nhập vào hệ thống. 2. Sinh viên chọn chức năng tải file. 3. Sinh viên chọn file từ thiết bị cá nhân. 4. Hệ thống kiểm tra định dạng file. 5. Hệ thống tải file lên thành công và lưu trữ. |
| Exception | Nếu định dạng file không hợp lệ, hệ thống thông báo lỗi. |
| Post-condition | File đã được tải lên và sẵn sàng cho việc in. |

Bảng 1. Use-case tải file lên hệ thống

|  |  |
| --- | --- |
| Use-case Name | **Chọn máy in và cấu hình in** |
| Use-case ID | UC02 |
| Actor | Student |
| Description | Sinh viên chọn máy in và thiết lập các thuộc tính in (kích thước giấy, số trang, số bản in, ...). |
| Pre-condition | Sinh viên đã đăng nhập và tải file thành công lên hệ thống. |
| Flow | 1. Sinh viên chọn chức năng in tài liệu. 2. Sinh viên chọn máy in từ danh sách có sẵn. 3. Sinh viên thiết lập các thuộc tính in (kích thước giấy, số trang, một mặt/đôi mặt, số bản in, ...). 4. Hệ thống kiểm tra số lượng trang in còn lại của sinh viên. 5. Hệ thống xử lý lệnh in. |
| Exception | Nếu sinh viên không đủ số trang in, hệ thống thông báo lỗi. |
| Post-condition | Lệnh in được gửi đến máy in và đang chờ xử lý. |

Bảng 2. Use-case chọn máy in và cấu hình in

|  |  |
| --- | --- |
| Use-case Name | **Kiểm tra số lượng trang in còn lại** |
| Use-case ID | UC03 |
| Actor | Student |
| Description | Sinh viên kiểm tra số trang in còn lại trong tài khoản của mình. |
| Pre-condition | Sinh viên đã được cấp số trang in mặc định. |
| Flow | 1. Sinh viên đăng nhập vào hệ thống. 2. Xem số lượng trang in còn lại trong tài khoản cá nhân. |
| Exception |  |
| Post-condition | Sinh viên biết số lượng trang in còn lại trong tài khoản. |

Bảng 3. Use-case kiểm tra số lượng trang in còn lại

|  |  |
| --- | --- |
| Use-case Name | **Mua thêm số trang in** |
| Use-case ID | UC04 |
| Actor | Student |
| Description | Sinh viên mua thêm trang in thông qua BKPay. |
| Pre-condition | Sinh viên có tài khoản BKPay hợp lệ. |
| Flow | 1. Sinh viên truy cập trang mua trang in. 2. Nhập số trang cần mua. 3. Thanh toán qua hệ thống BKPay. |
| Exception | Nếu thanh toán không thành công, hệ thống thông báo lỗi. |
| Post-condition | Số trang in đã được thêm vào tài khoản của sinh viên. |

Bảng 4. Use-case mua thêm trang in

|  |  |
| --- | --- |
| Use-case Name | **Quản lý máy in** |
| Use-case ID | UC05 |
| Actor | SPSO |
| Description | Quản lý các máy in trong hệ thống. |
| Pre-condition | SPSO đã đăng nhập thông qua HCMUT\_SSO. |
| Flow | 1. SPSO truy cập mục quản lý máy in. 2. Thêm, kích hoạt hoặc vô hiệu hóa máy in. |
| Exception |  |
| Post-condition | Máy in mới đã được thêm, kích hoạt hoặc vô hiệu hóa thành công. |

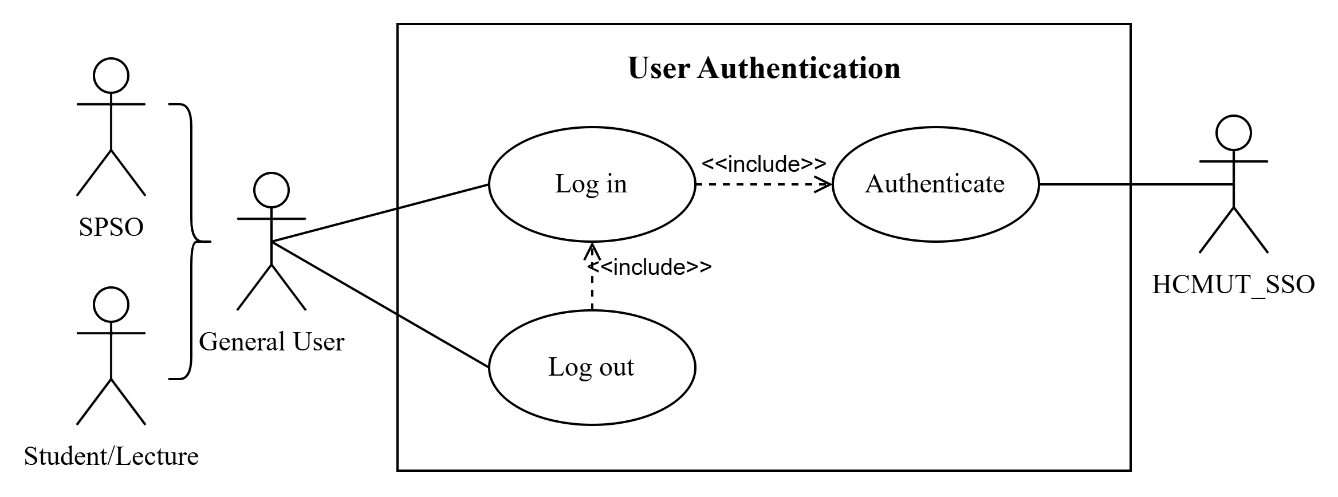
Bảng 5. Use-case quản lý máy in

|  |  |
| --- | --- |
| Use-case Name | **Quản lý lịch sử in ấn** |
| Use-case ID | UC06 |
| Actor | SPSO |
| Description | Xem và quản lý lịch sử in ấn của sinh viên. |
| Pre-condition | SPSO đã đăng nhập thông qua HCMUT\_SSO. |
| Flow | 1. SPSO truy cập mục quản lý lịch sử in ấn. 2. Lọc lịch sử in ấn theo sinh viên, máy in, khoảng thời gian. |
| Exception |  |
| Post-condition | SPSO có thể xem lịch sử in ấn và thực hiện các thao tác quản lý cần thiết. |

Bảng 6. Use-case quản lý lịch sử in ấn

|  |  |
| --- | --- |
| Use-case Name | **Thay đổi cấu hình hệ thống** |
| Use-case ID | UC07 |
| Actor | SPSO |
| Description | Thay đổi số trang mặc định, định dạng file hợp lệ, ngày cấp trang. |
| Pre-condition | SPSO có quyền quản lý hệ thống. |
| Flow | 1. SPSO truy cập mục quản lý cấu hình hệ thống. 2. Thay đổi các thiết lập như số trang in mặc định, định dạng file. |
| Exception |  |
| Post-condition | Cấu hình hệ thống đã được thay đổi thành công. |

Bảng 7. Use-case thay đổi cấu hình hệ thống



|  |  |
| --- | --- |
| Use-case Name | **Xác thực người dùng** |
| Use-case ID | UC08 |
| Actor | HCMUT\_SSO |
| Description | Xác thực thông tin tài khoản và mật khẩu người dùng. |
| Pre-condition | Người dùng có thông tin đăng nhập hợp lệ trong hệ thống HCMUT\_SSO. |
| Flow | 1. Người dùng nhập thông tin đăng nhập. 2. Hệ thống xác thực thông qua HCMUT\_SSO (Oauth). 3. Hệ thống xác thực thành công, người dùng truy cập vào trang chủ. |
| Exception | 2.1. Hệ thống hiển thị thông báo lỗi  2.2. Màn hình quay về trạng thái đăng nhập |
| Post-condition | Người dùng được đăng nhập thành công vào hệ thống. |

Bảng 8. Use-case xác thực người dùng

**III. Task 2: System modelling**

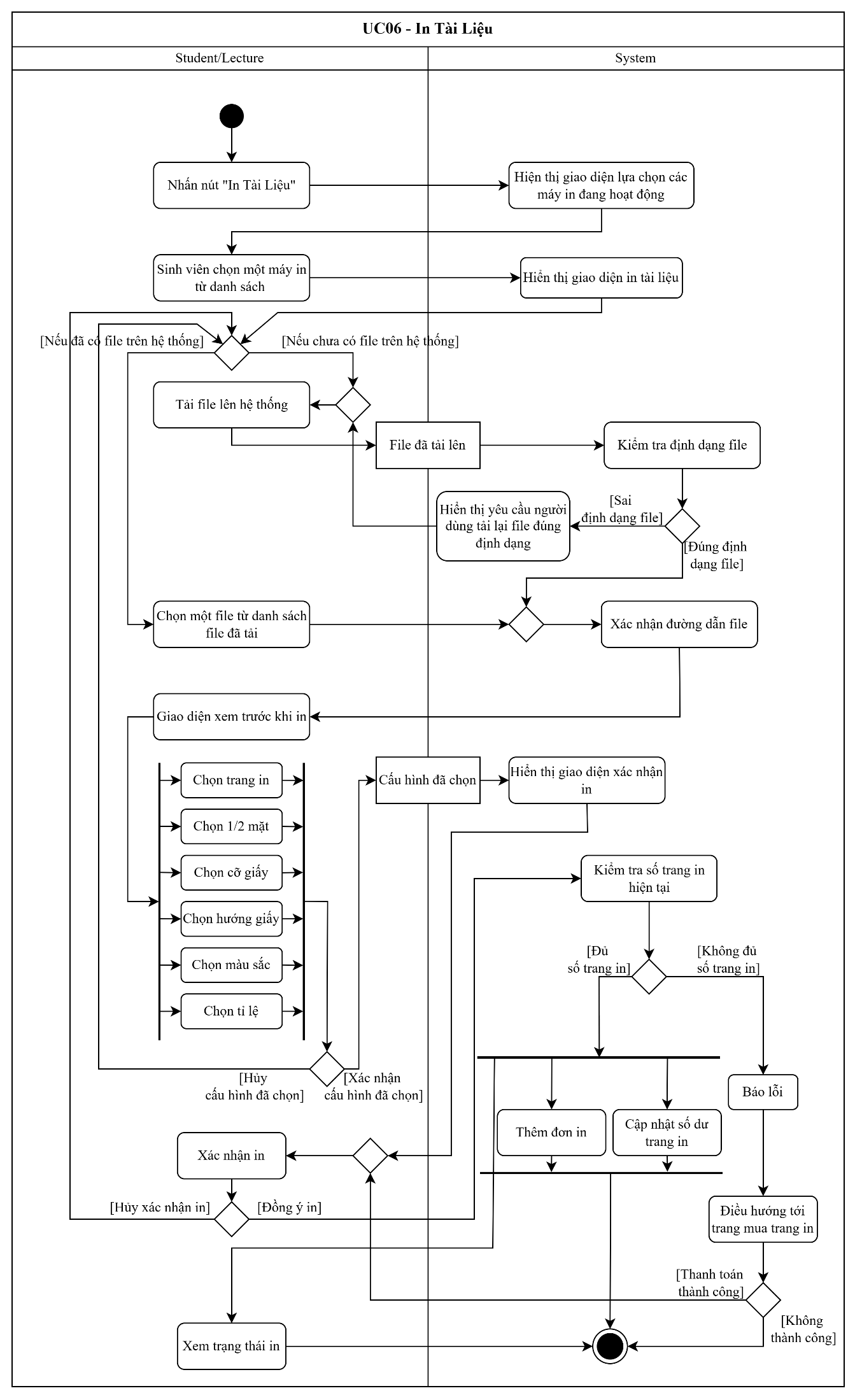
**1. Activity diagram**

**Yêu cầu:** Draw an activity diagram to capture the business process between systems and the stakeholders in a particular module (choose a module in Task 1.3)

**Hint:** draw an activity diagram for each use case with a swimlane (<https://circle.visual-paradigm.com/activity-diagram-example-swimlane/>) for different stakeholders. The number of diagrams should be maximum 5. Try to use as many notations of the diagrams as possible (<https://www.visual> paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/). Write 01 paragraph for each diagram to describe it. The diagrams should be correct and complete.

**Mô tả**

Sơ đồ mô tả quy trình in tài liệu từ góc nhìn của hai tác nhân chính: Sinh viên/Giảng viên và Hệ thống. Quy trình bắt đầu khi người dùng nhấn nút "In tài liệu", hệ thống sẽ hiển thị giao diện để người dùng lựa chọn máy in đang hoạt động. Sau đó, người dùng có thể tải file lên hệ thống nếu chưa có file hoặc chọn từ danh sách các file đã tải sẵn. Hệ thống kiểm tra định dạng file, nếu đúng, người dùng tiếp tục quy trình in, nếu sai, hệ thống sẽ yêu cầu tải lại file đúng định dạng. Sau khi chọn file, giao diện xem trước khi in được hiển thị, và người dùng có thể chọn các tùy chọn như trang in, 1 hoặc 2 mặt, kích thước giấy, hướng giấy, màu sắc, và tỉ lệ in. Người dùng xác nhận các cấu hình đã chọn hoặc hủy bỏ nếu muốn chỉnh sửa. Sau đó, hệ thống kiểm tra số trang in hiện có; nếu đủ, đơn in được thêm vào và số trang in được cập nhật. Nếu không đủ số trang, hệ thống báo lỗi và điều hướng người dùng đến trang mua thêm trang in. Sau khi hoàn tất thanh toán, quá trình in được tiếp tục, và trạng thái in sẽ được hiển thị cho người dùng xem.



**2. Sequece Diagram**

**Yêu cầu:** Draw a sequence diagram for a particular module (the same with the module used in task 2.1)

**Hint:** draw 01 sequence diagram for each use case with a swimlane for different stakeholders. The number of diagrams should be maximum 5. Try to use as many notations of the diagram as possible (<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>) Write 01 paragraph for each diagram to describe it. The diagrams should be correct and complete.

**Mô tả**

Người dùng (Sinh viên/Giảng viên) nhấn nút "In Tài Liệu" trên giao diện UI. Hành động này kích hoạt yêu cầu từ giao diện người dùng đến hệ thống để lấy danh sách các máy in đang hoạt động. Giao diện người dùng gửi yêu cầu getActivePrintersAPI() tới PrinterController. PrinterController tiếp tục gọi phương thức getActivePrinters() từ PrinterService. PrinterService sẽ yêu cầu từ Cơ sở dữ liệu để lấy danh sách máy in thông qua phương thức findAllPrinters(). Sau khi nhận được danh sách máy in từ Cơ sở dữ liệu, thông tin này sẽ được kiểm tra trạng thái bằng phương thức checkPrinterStatus(), sau đó kết quả trả về PrinterService và PrinterController. PrinterController gửi danh sách máy in trở lại giao diện người dùng.

Người dùng chọn một máy in từ danh sách máy in được hiển thị. Giao diện người dùng gửi yêu cầu selectPrinterAPI(printerId) đến PrinterController. Tại đây, PrinterController gọi phương thức validateAndFetchPrinterDetails(printerId) từ PrinterService để xác thực và lấy thông tin chi tiết của máy in. PrinterService gửi yêu cầu getPrinterDetails(printerId) tới Cơ sở dữ liệu để lấy thông tin máy in. Sau khi xác thực máy in thành công, dữ liệu được gửi lại từ PrinterService và PrinterController trở lại giao diện người dùng.

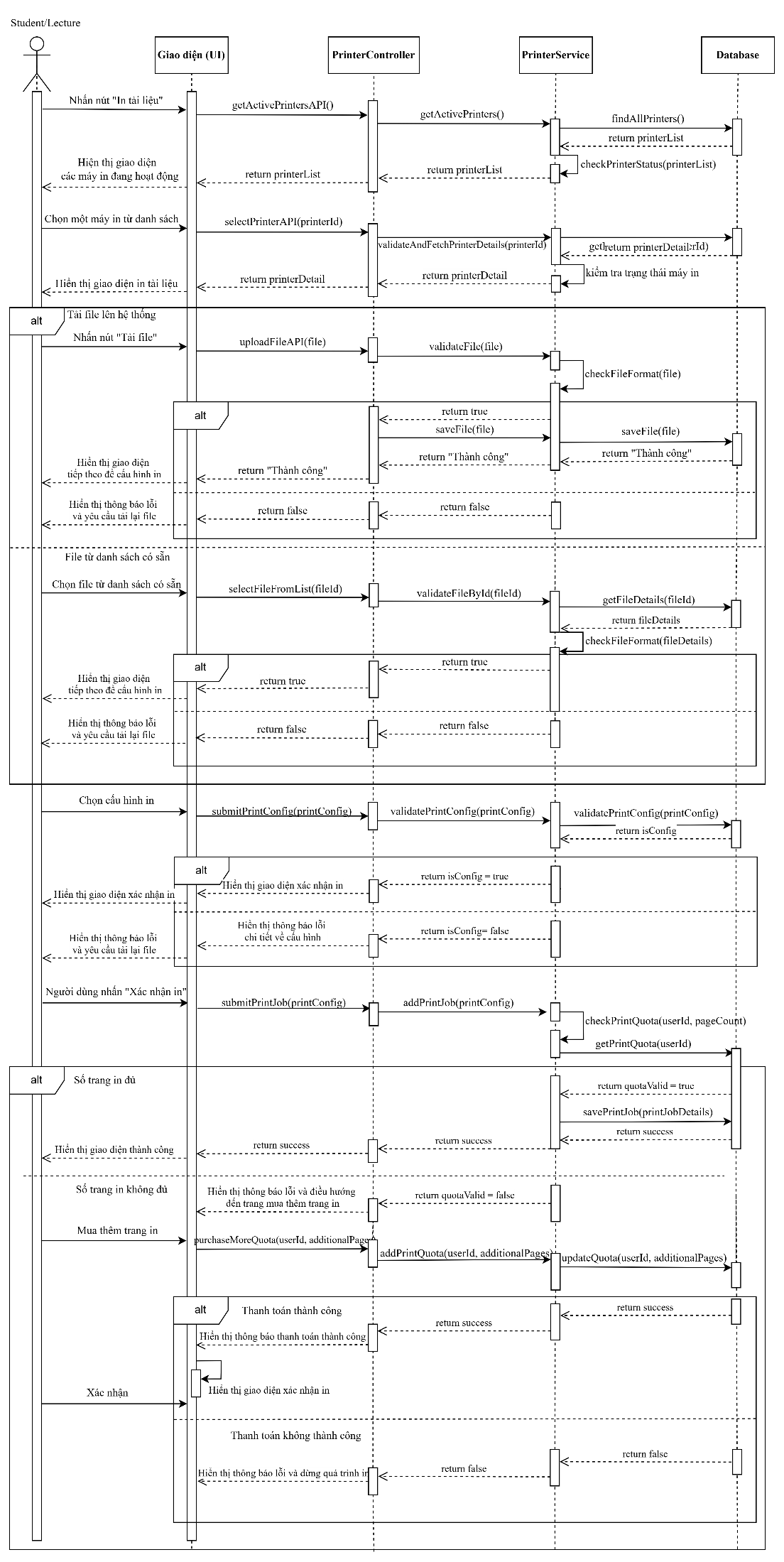
Người dùng tải file lên từ giao diện. Giao diện gọi phương thức uploadFileAPI(file) để gửi file lên hệ thống thông qua PrinterController. PrinterController gọi phương thức validateFile(file) từ PrinterService để kiểm tra định dạng và cấu trúc của file. PrinterService kiểm tra định dạng file bằng cách gọi checkFileFormat(file). Nếu file hợp lệ, file sẽ được lưu vào cơ sở dữ liệu và PrinterService trả về trạng thái "thành công". Nếu file không hợp lệ, hệ thống sẽ trả về lỗi yêu cầu người dùng tải lại file hợp lệ.

Nếu người dùng chọn một file đã tồn tại từ danh sách, giao diện sẽ gửi yêu cầu selectFileFromList(fileId) đến PrinterController. Tương tự, PrinterController sẽ gọi validateFileById(fileId) từ PrinterService, sau đó PrinterService kiểm tra thông tin chi tiết của file thông qua getFileDetails(fileId) từ Cơ sở dữ liệu. Nếu file hợp lệ, quá trình sẽ tiếp tục như với việc tải file mới, nếu không hợp lệ, người dùng sẽ nhận được thông báo yêu cầu tải lại file hợp lệ.

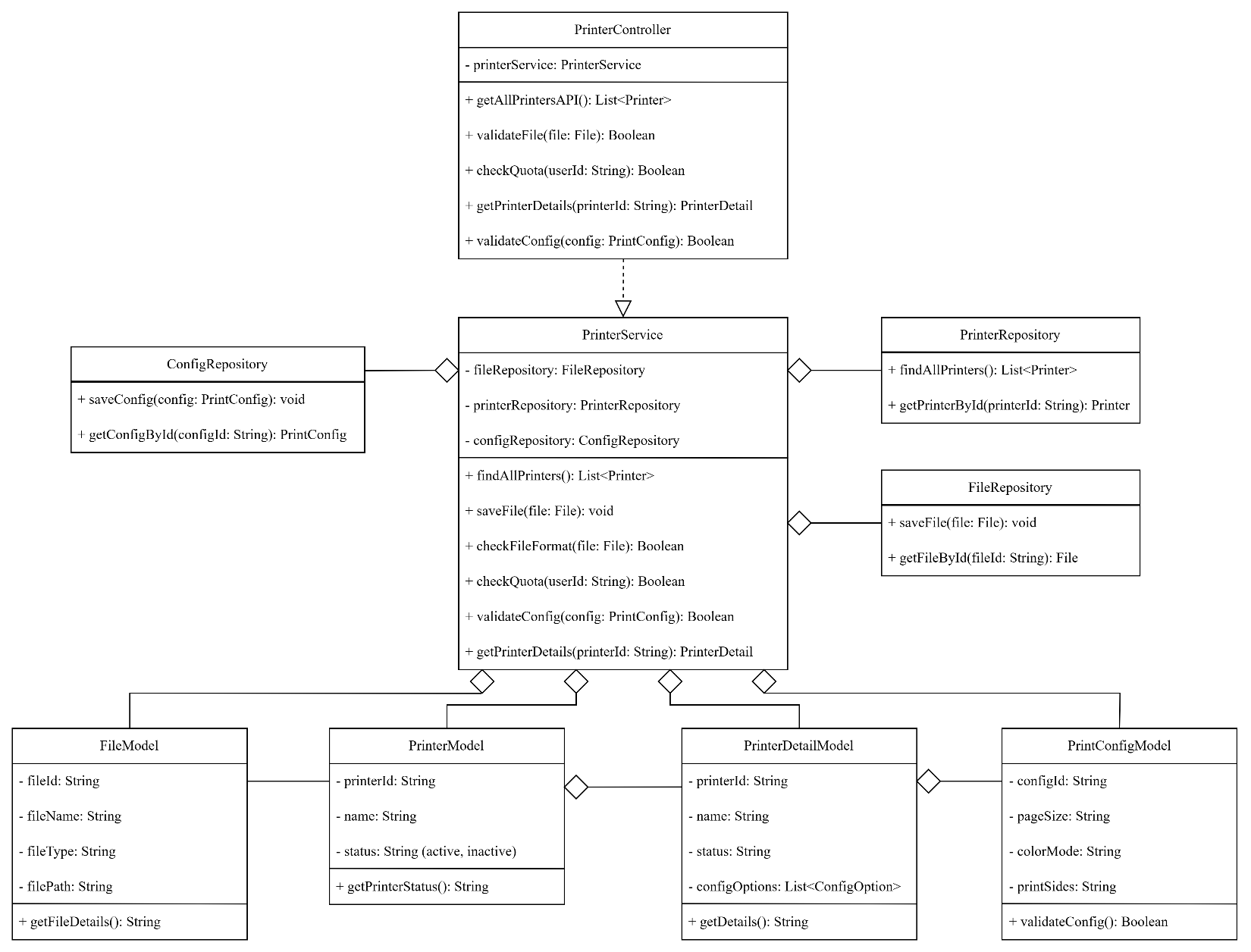
Sau khi file đã được xác thực thành công, hệ thống hiển thị giao diện cho người dùng để chọn cấu hình in (chọn trang, 1/2 mặt, cỡ giấy, hướng giấy, màu sắc). Khi người dùng xác nhận cấu hình, hệ thống sẽ kiểm tra thông qua validatePrintConfig(). Nếu cấu hình hợp lệ, quá trình in sẽ được tiếp tục. Nếu không hợp lệ, người dùng sẽ nhận được thông báo lỗi chi tiết về cấu hình.

Sau khi xác nhận cấu hình in, hệ thống sẽ kiểm tra số trang in còn lại của người dùng qua hàm getQuota(). Nếu số trang in đủ, hệ thống sẽ cho phép tiếp tục in và trả về trạng thái thành công. Nếu không đủ số trang in, người dùng sẽ được chuyển hướng đến trang thanh toán thông qua việc hiển thị thông báo lỗi.

Nếu người dùng cần mua thêm trang in, hệ thống sẽ điều hướng họ đến trang thanh toán. Sau khi hoàn tất thanh toán thành công, hệ thống cập nhật lại hạn mức trang in và tiếp tục quy trình in. Nếu thanh toán không thành công, hệ thống sẽ dừng quá trình in và thông báo lỗi.



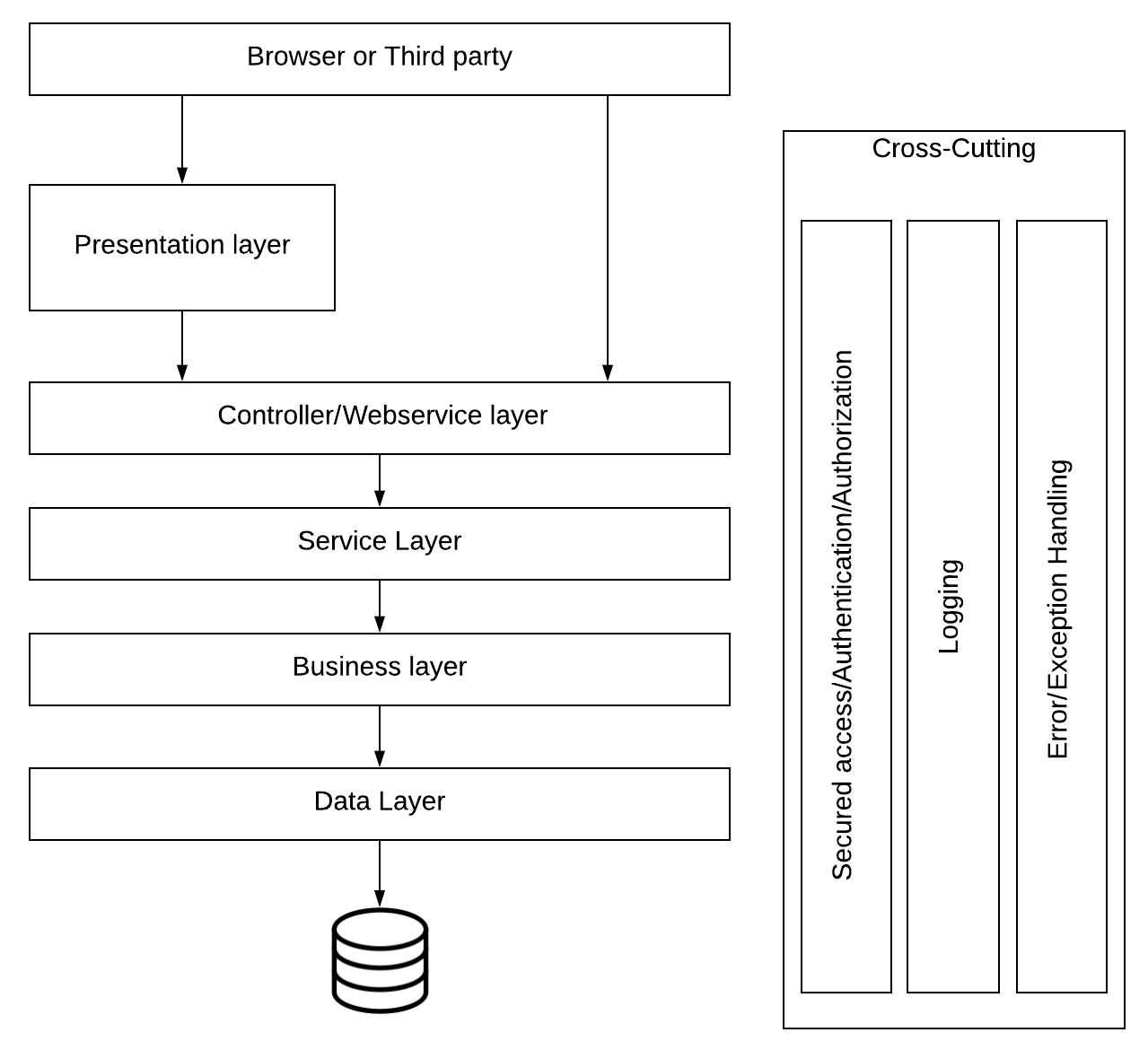
**3. Class Diagram**

****

**IV. Task 3: Architecture Design**

**1. Architectural Diagram**

Kiến trúc phân lớp (Layered Architecture) là một mô hình thiết kế phổ biến giúp tách biệt các thành phần trong hệ thống thành các tầng riêng biệt, mỗi tầng có nhiệm vụ và trách nhiệm cụ thể, đồng thời dễ dàng tích hợp các chức năng phụ trợ (Cross-Cutting Concerns). Trong hệ thống HCMUT-SSPS, kiến trúc này chia hệ thống thành các lớp chính bao gồm Presentation Layer, Business Logic Layer, Data Access Layer và Data Storage Layer. Mỗi lớp được thiết kế độc lập, giúp hệ thống dễ bảo trì, dễ mở rộng và tăng cường bảo mật. Bằng cách tách biệt các thành phần, hệ thống có thể linh hoạt thay đổi công nghệ hoặc bổ sung tính năng mới mà không ảnh hưởng đến toàn bộ cấu trúc.



**1.1. Presentation Layer**

Mô tả: Tầng Presentation là giao diện người dùng, nơi xử lý các tương tác trực tiếp với người dùng và hiển thị dữ liệu. Tầng này nhận và gửi yêu cầu từ người dùng thông qua trình duyệt hoặc các dịch vụ bên thứ ba.

Công nghệ: Sử dụng NextJS (dựa trên React) để xây dựng giao diện người dùng, giúp render nhanh chóng và thân thiện với SEO.

Chức năng:

* Hiển thị dữ liệu cho người dùng và tiếp nhận các yêu cầu từ người dùng.
* Gửi các yêu cầu HTTP đến Controller Layer của backend để lấy hoặc cập nhật dữ liệu.
* Nhận phản hồi từ backend và cập nhật giao diện một cách động.

**1.2. Controller/Webservice Layer**

Mô tả: Đây là tầng trung gian giữa Presentation Layer và các tầng xử lý nghiệp vụ, có nhiệm vụ nhận các yêu cầu từ Presentation Layer, sau đó điều hướng các yêu cầu này đến các tầng dịch vụ phù hợp.

Công nghệ: Java Spring Boot với các RESTful Controller.

Chức năng:

* Nhận các yêu cầu HTTP từ frontend (NextJS) và điều phối các yêu cầu đó đến Service Layer.
* Thực hiện các thao tác như xác thực, phân quyền người dùng, và trả về kết quả dưới dạng JSON để frontend có thể hiển thị.
* Xác định và xử lý các mã trạng thái HTTP (như 200, 404, 500) để giúp frontend hiểu rõ tình trạng của yêu cầu.

**1.3. Service Layer**

Mô tả: Tầng Service chứa các logic nghiệp vụ chính của hệ thống. Đây là nơi xử lý các quy tắc nghiệp vụ phức tạp, các kiểm tra điều kiện, và các quy trình xử lý dữ liệu trước khi gửi xuống Data Layer.

Công nghệ: Sử dụng Spring Boot Service classes.

Chức năng:

* Thực hiện logic nghiệp vụ chính và xử lý các yêu cầu từ tầng Controller.
* Gọi các repository ở Data Layer khi cần lấy hoặc cập nhật dữ liệu từ cơ sở dữ liệu.
* Đảm bảo tính nhất quán và kiểm tra các quy tắc nghiệp vụ trước khi thực hiện các thao tác CRUD.

**1.4. Business Layer**

Mô tả: Đây là lớp chính chịu trách nhiệm xử lý các quy tắc nghiệp vụ phức tạp và cung cấp các phương thức chung cho Service Layer. Business Layer có thể bao gồm các lớp nghiệp vụ độc lập và có thể tái sử dụng, giúp duy trì logic nghiệp vụ của hệ thống.

Chức năng:

* Đóng gói các quy tắc nghiệp vụ phức tạp và có thể sử dụng lại.
* Đảm bảo rằng các logic nghiệp vụ được tuân thủ trên toàn bộ hệ thống và cung cấp sự kiểm soát nhất quán cho các tầng khác.
* Hỗ trợ Service Layer trong việc xử lý các quy trình nghiệp vụ phức tạp.

**1.5. Data Layer**

Mô tả: Đây là tầng làm việc trực tiếp với cơ sở dữ liệu, cung cấp các phương thức truy cập và lưu trữ dữ liệu từ cơ sở dữ liệu SQL trên Aiven. Data Layer chịu trách nhiệm thực hiện các thao tác CRUD và chuyển đổi dữ liệu giữa các định dạng của cơ sở dữ liệu và hệ thống.

Công nghệ: Spring Data JPA để truy cập SQL trên Aiven.

Chức năng:

* Xử lý tất cả các thao tác truy xuất, lưu trữ, và cập nhật dữ liệu trong cơ sở dữ liệu.
* Chuyển đổi dữ liệu từ các bảng trong SQL thành các đối tượng mà Business và Service Layer có thể sử dụng.
* Đảm bảo dữ liệu được truy cập nhanh chóng, nhất quán và an toàn.

**1.6. Cross-Cutting Concerns**

Ngoài các lớp chính, hệ thống còn bao gồm các Cross-Cutting Concerns được áp dụng trên tất cả các tầng, bao gồm:

* Secured Access/Authentication/Authorization: Đảm bảo các yêu cầu được xác thực và người dùng được phân quyền phù hợp, ngăn chặn truy cập trái phép.
* Logging: Ghi lại các hoạt động và sự kiện của hệ thống để theo dõi và khắc phục lỗi khi cần.
* Error/Exception Handling: Xử lý các lỗi và ngoại lệ trên toàn hệ thống, đảm bảo hệ thống không bị gián đoạn và cung cấp phản hồi thích hợp đến người dùng.

**1.7. Deployment Diagram**

Sơ đồ triển khai cho thấy cách các thành phần chính trong hệ thống HCMUT-SSPS được triển khai và tương tác với nhau. Việc sử dụnggiao thức TCP/IP giữa server và database server đảm bảo độ tin cậy cho quá trình truy xuất và cập nhật dữ liệu trên Aiven.

a. Client (User Device):

* Kết nối với Server: Máy client (người dùng) sẽ kết nối với backend server thông qua giao thức HTTP.
* Hiển thị giao diện: Client sẽ nhận và render giao diện từ các file frontend NextJS thông qua trình duyệt (browser) của người dùng. NextJS là framework dựa trên React, giúp render giao diện ở cả phía server và client, tối ưu hóa tốc độ và trải nghiệm người dùng. Giao diện người dùng trên client sẽ gửi yêu cầu và nhận phản hồi từ backend server qua các API REST, từ đó hiển thị và cập nhật nội dung dựa trên tương tác của người dùng.

b. Server Backend (Java Spring Boot)

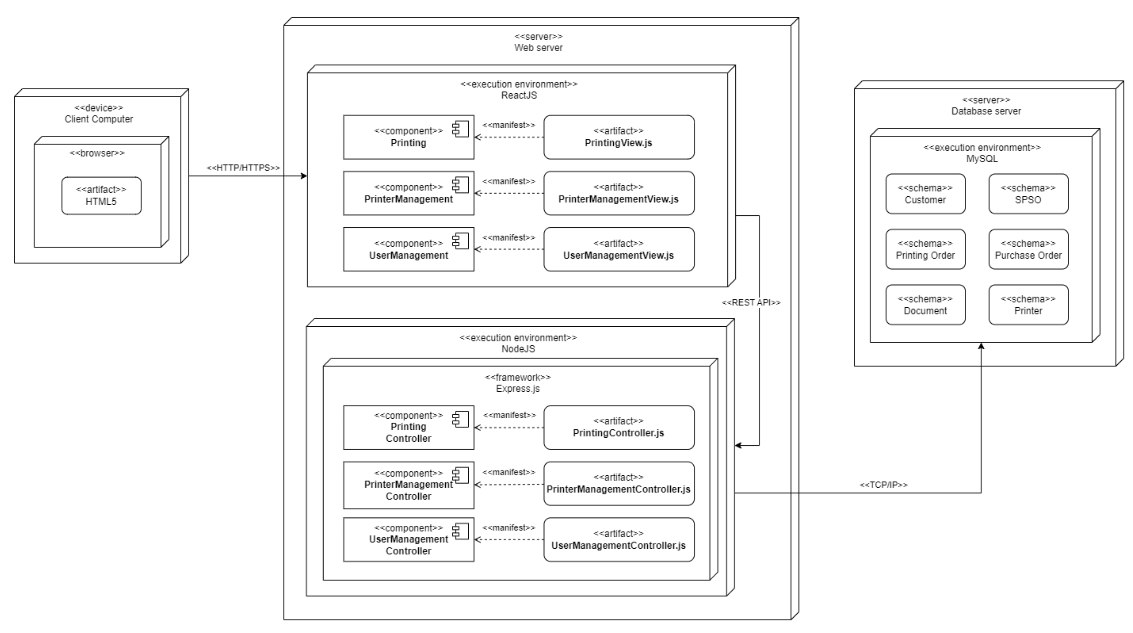
* Môi trường thực thi (Execution Environment): Backend server sử dụng Java Spring Boot, một framework mạnh mẽ để xây dựng các ứng dụng web và dịch vụ RESTful. Spring Boot xử lý tất cả các yêu cầu đến từ frontend và thực hiện các logic nghiệp vụ, chẳng hạn như xử lý thông tin người dùng, truy vấn dữ liệu, và thực hiện các thao tác cần thiết.
* Các Component chính: Spring Boot backend được chia thành các component chính phục vụ cho các nghiệp vụ khác nhau của hệ thống. Các component này sẽ bao gồm các controller (để xử lý yêu cầu HTTP), service (chứa logic nghiệp vụ), và repository (truy xuất và thao tác với cơ sở dữ liệu).
* Giao tiếp giữa frontend và backend: Frontend NextJS sẽ giao tiếp với backend Spring Boot qua các REST API. Các API này giúp frontend gửi yêu cầu để lấy dữ liệu (ví dụ: danh sách sản phẩm, thông tin người dùng) hoặc thực hiện các thao tác (như đăng nhập, thêm sản phẩm). Backend sẽ phản hồi các yêu cầu này với các dữ liệu và mã trạng thái HTTP tương ứng.

c. Database Server (SQL trên Aiven)

* Kết nối với Backend Server: Backend server sẽ kết nối với Database Server trên Aiven qua giao thức TCP/IP. Giao thức này đảm bảo dữ liệu được truyền tải hiệu quả và an toàn giữa backend và cơ sở dữ liệu.
* Cơ sở dữ liệu SQL: Database server sử dụng một cơ sở dữ liệu SQL, được lưu trữ và quản lý trên nền tảng Aiven. Aiven cung cấp các dịch vụ cơ sở dữ liệu đám mây, cho phép dễ dàng quản lý, sao lưu, và giám sát dữ liệu. Cơ sở dữ liệu này lưu trữ các thông tin có cấu trúc, ví dụ như thông tin người dùng, danh sách sản phẩm, và các mối quan hệ giữa các bảng.
* Quy trình truy xuất dữ liệu: Khi có yêu cầu truy xuất hoặc cập nhật dữ liệu từ người dùng, backend server (Spring Boot) sẽ gửi truy vấn đến database server qua Aiven, nhận dữ liệu cần thiết và trả về cho client qua API REST. Điều này giúp đảm bảo rằng dữ liệu luôn được cập nhật và nhất quán giữa các thành phần.

d. Luồng Dữ liệu và Tương tác

* Tương tác giữa Client và Backend: Máy client sẽ gửi các yêu cầu HTTP đến backend server qua giao thức HTTPS. Các yêu cầu này sẽ được xử lý bởi các controller của Spring Boot, sau đó gọi đến các service và repository để lấy dữ liệu từ database hoặc xử lý logic nghiệp vụ.
* Tương tác giữa Backend và Database: Backend server sẽ giao tiếp với cơ sở dữ liệu thông qua các truy vấn SQL để lấy, cập nhật, hoặc xóa dữ liệu, và các thay đổi này sẽ phản ánh trực tiếp lên frontend khi dữ liệu được phản hồi.
* Luồng dữ liệu phản hồi: Sau khi xử lý các yêu cầu từ client, backend sẽ trả về dữ liệu cần thiết (hoặc mã trạng thái HTTP phù hợp nếu có lỗi) qua giao thức HTTPS đến client để cập nhật giao diện hoặc thông báo trạng thái đến người dùng.



**2. Presentation Strategy**

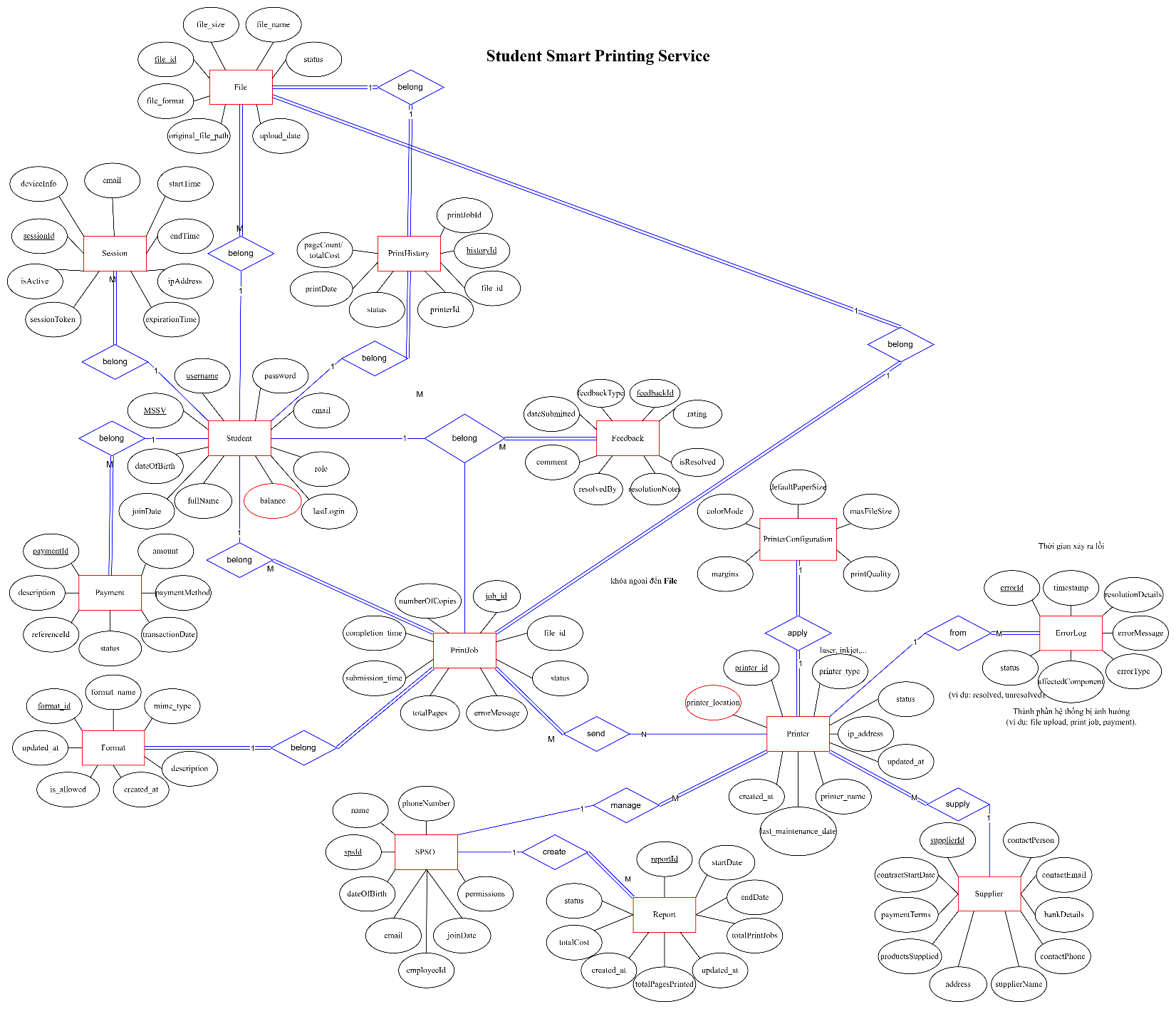
Hệ thống HCMUT-SSPS sử dụng NextJS làm framework cho tầng trình bày (Presentation Layer) vì tính linh hoạt và hiệu suất mà nó mang lại. NextJS hỗ trợ server-side rendering (SSR), static site generation (SSG) và client-side rendering (CSR), cho phép hệ thống lựa chọn phương pháp render tối ưu tùy theo yêu cầu cụ thể của từng trang. Ví dụ, với các trang có nội dung tĩnh như thông tin chung hoặc bảng điểm, SSG có thể được áp dụng để tối ưu hóa tốc độ tải trang. Đối với các trang có nội dung động và cần tải theo thời gian thực (như trang thông tin sinh viên), SSR giúp cải thiện SEO và cung cấp dữ liệu mới nhất.

Ngoài ra, khả năng tích hợp dễ dàng với React giúp NextJS phát triển giao diện người dùng theo hướng component-based, cho phép các thành phần giao diện có thể tái sử dụng và dễ dàng bảo trì. Người dùng sẽ truy cập hệ thống thông qua trình duyệt hoặc các ứng dụng di động (nếu cần) và giao diện sẽ giao tiếp với backend thông qua các REST API để tải và hiển thị dữ liệu. Điều này đảm bảo trải nghiệm người dùng nhanh chóng và hiệu quả, đồng thời cải thiện khả năng mở rộng của hệ thống khi lượng người dùng tăng.

**3. Data Storage Approach**

Hệ thống HCMUT-SSPS lựa chọn SQL Database trên Aiven cho tầng lưu trữ dữ liệu, đảm bảo khả năng quản lý dữ liệu có cấu trúc và các mối quan hệ phức tạp. SQL là một lựa chọn lý tưởng cho các ứng dụng yêu cầu tính nhất quán và toàn vẹn dữ liệu cao, như hệ thống quản lý giáo dục, trong đó dữ liệu về sinh viên, khóa học và điểm số cần được lưu trữ chặt chẽ và dễ dàng truy vấn. Aiven cung cấp một nền tảng cơ sở dữ liệu quản lý đám mây, cho phép hệ thống dễ dàng cấu hình và giám sát mà không cần đội ngũ kỹ thuật phải quản lý hạ tầng vật lý.

Ngoài ra, Aiven cung cấp các tính năng bảo mật tích hợp, bao gồm mã hóa dữ liệu và kiểm soát truy cập, giúp bảo vệ dữ liệu nhạy cảm của người dùng và tuân thủ các yêu cầu bảo mật. Khả năng tự động sao lưu và phục hồi dữ liệu của Aiven cũng là một điểm mạnh, đảm bảo rằng dữ liệu quan trọng không bị mất và có thể dễ dàng khôi phục trong trường hợp sự cố. Với khả năng mở rộng của Aiven, hệ thống có thể dễ dàng điều chỉnh dung lượng và hiệu suất của cơ sở dữ liệu khi lượng người dùng và dữ liệu tăng lên.



**4. API Management**

Trong hệ thống HCMUT-SSPS, các RESTful API được sử dụng để kết nối giữa frontend và backend, cho phép dữ liệu được trao đổi một cách nhất quán và dễ hiểu. Các API RESTful giúp frontend (NextJS) và backend (Java Spring Boot) giao tiếp thông qua các yêu cầu HTTP tiêu chuẩn như GET, POST, PUT, DELETE, giúp đảm bảo rằng các thao tác như lấy dữ liệu sinh viên, cập nhật thông tin, hoặc xóa dữ liệu đều được thực hiện có tổ chức và theo chuẩn mực.

Để đảm bảo an toàn cho API, hệ thống sử dụng JWT (JSON Web Tokens) hoặc OAuth2 cho xác thực và phân quyền. JWT giúp frontend gửi các yêu cầu kèm token xác thực, đảm bảo rằng chỉ những yêu cầu được cấp quyền mới có thể truy cập vào tài nguyên nhạy cảm. Điều này giúp bảo vệ dữ liệu người dùng và ngăn chặn truy cập trái phép. Các cơ chế xử lý lỗi và mã trạng thái HTTP cũng được thiết lập để thông báo cho frontend về các vấn đề như lỗi 400 (Bad Request) hoặc 401 (Unauthorized), giúp cải thiện khả năng quản lý và xử lý sự cố.

Ngoài ra, hệ thống có thể triển khai API Gateway để quản lý và điều phối lưu lượng truy cập, cũng như áp dụng rate limiting (giới hạn tần suất truy cập) nhằm ngăn chặn các yêu cầu quá tải hoặc các cuộc tấn công DDoS. API Gateway cũng cung cấp khả năng giám sát và phân tích hiệu suất, giúp cải thiện trải nghiệm người dùng và tối ưu hóa hiệu suất API.

**5. Component Diagram**

Các thành phần chính trong Component Diagram cho Printer Module:

UI Component (Giao diện người dùng)

Mô tả: Đây là thành phần giao diện người dùng, nơi người dùng tương tác trực tiếp để thực hiện các thao tác liên quan đến máy in, như xem danh sách máy in, xem chi tiết máy in, cấu hình máy in, và xác nhận in.

Giao diện (Interfaces):

PrinterListView: Hiển thị danh sách máy in.

PrinterDetailView: Hiển thị chi tiết thông tin của một máy in cụ thể.

PrinterConfigView: Cấu hình máy in và hiển thị giao diện khi người dùng cần cấu hình lại.

PrintConfirmationView: Xác nhận in sau khi các điều kiện in ấn thỏa mãn.

Phụ thuộc (Dependencies):

Giao tiếp với PrinterController thông qua REST API để lấy danh sách máy in, chi tiết máy in, và gửi yêu cầu in.

PrinterController Component

Mô tả: Đây là thành phần kiểm soát, xử lý các yêu cầu từ UI Component và chuyển các yêu cầu này đến PrinterService. PrinterController sẽ nhận dữ liệu từ PrinterService và trả kết quả về cho UI Component.

Giao diện (Interfaces):

getPrinterList(): Nhận yêu cầu lấy danh sách máy in và chuyển đến PrinterService.

getPrinterDetail(printerId): Nhận yêu cầu lấy thông tin chi tiết của máy in từ PrinterService.

configurePrinter(configParams): Xử lý yêu cầu cấu hình máy in và chuyển tiếp đến PrinterService.

confirmPrint(): Gửi yêu cầu xác nhận in khi người dùng đã chọn in.

Phụ thuộc (Dependencies):

Phụ thuộc vào PrinterService để thực hiện các nghiệp vụ liên quan đến in ấn và truy vấn dữ liệu.

PrinterService Component

Mô tả: Đây là thành phần xử lý nghiệp vụ, nơi chứa các logic nghiệp vụ chính liên quan đến in ấn. PrinterService sẽ nhận các yêu cầu từ PrinterController, xử lý chúng và tương tác với Database Component khi cần thiết.

Giao diện (Interfaces):

fetchPrinterList(): Truy vấn danh sách các máy in từ Database Component.

fetchPrinterDetail(printerId): Lấy thông tin chi tiết của một máy in từ Database Component.

applyPrinterConfig(configParams): Áp dụng cấu hình cho máy in và cập nhật dữ liệu vào cơ sở dữ liệu.

validateQuota(userId): Kiểm tra số lượng trang in còn lại của người dùng.

executePrint(printerId, fileData): Thực hiện lệnh in với các điều kiện đã được xác nhận.

Phụ thuộc (Dependencies):

Phụ thuộc vào Database Component để truy vấn và cập nhật dữ liệu liên quan đến máy in và cấu hình in.

Database Component

Mô tả: Đây là thành phần cơ sở dữ liệu, nơi lưu trữ dữ liệu về máy in, cấu hình, số trang in còn lại của người dùng, và các thông tin khác liên quan đến module in ấn.

Giao diện (Interfaces):

getPrinters(): Trả về danh sách các máy in.

getPrinterDetail(printerId): Trả về thông tin chi tiết của một máy in.

savePrinterConfig(printerId, configParams): Lưu cấu hình mới cho một máy in.

checkQuota(userId): Kiểm tra hạn mức in của người dùng.

logPrintJob(printerId, userId, jobDetails): Ghi nhận thông tin về công việc in.

Phụ thuộc (Dependencies):

Phụ thuộc vào PrinterService để nhận các yêu cầu thao tác với dữ liệu liên quan đến in ấn.

Các Cross-Cutting Concerns

Authentication/Authorization Component:

Kiểm tra quyền truy cập trước khi cho phép người dùng thực hiện các thao tác như cấu hình máy in hoặc gửi lệnh in.

Logging Component:

Ghi nhận các hoạt động như truy cập danh sách máy in, cấu hình máy in và yêu cầu in, nhằm mục đích giám sát và kiểm tra lỗi khi cần thiết.

Error Handling Component:

Xử lý các lỗi phát sinh trong quá trình giao tiếp giữa các component, ví dụ như lỗi kết nối cơ sở dữ liệu hoặc lỗi khi gửi yêu cầu in.