Bài 1:

Các loại hệ thống bao gồm:

* **TPS (Transaction Processing System)**: Hệ thống xử lý giao dịch, tập trung vào việc ghi nhận và xử lý các giao dịch hàng ngày một cách chính xác và hiệu quả.
* **MIS (Management Information System)**: Hệ thống thông tin quản lý, cung cấp báo cáo tổng hợp và phân tích dữ liệu định kỳ để hỗ trợ quản lý trung cấp theo dõi hiệu suất.
* **DSS (Decision Support System)**: Hệ thống hỗ trợ quyết định, sử dụng dữ liệu và mô hình phân tích để hỗ trợ ra quyết định không cấu trúc, như dự báo hoặc phân tích sâu.
* **EIS (Executive Information System)**: Hệ thống thông tin điều hành, cung cấp thông tin tổng quan, dễ tiếp cận (như dashboard) cho lãnh đạo cấp cao để giám sát chiến lược.

Phân loại từng chức năng cụ thể:

1. **Giao dịch bán hàng**: Đây là chức năng xử lý các giao dịch hàng ngày như ghi nhận đơn hàng, thanh toán, cập nhật kho hàng. Loại hệ thống tương ứng là **TPS**, vì nó tập trung vào việc thu thập và xử lý dữ liệu giao dịch thời gian thực để đảm bảo hoạt động kinh doanh diễn ra suôn sẻ.
2. **Phân tích xu hướng kinh doanh**: Chức năng này liên quan đến việc phân tích dữ liệu lịch sử để xác định xu hướng, như tăng/giảm doanh số theo thời gian hoặc hành vi khách hàng. Loại hệ thống tương ứng là **DSS**, vì nó hỗ trợ ra quyết định dựa trên phân tích dữ liệu phức tạp, mô hình dự báo, giúp doanh nghiệp điều chỉnh chiến lược kịp thời.
3. **Bảng tổng quan hiệu suất hàng tháng dành cho CEO**: Đây là chức năng cung cấp cái nhìn tổng quan về các chỉ số chính (KPIs) như doanh thu, lợi nhuận, hiệu suất hàng tháng qua giao diện dễ sử dụng (dashboard). Loại hệ thống tương ứng là **EIS**, vì nó dành riêng cho lãnh đạo cấp cao, giúp CEO nhanh chóng nắm bắt tình hình mà không cần đi sâu vào chi tiết.

Bài 2:

**1. Dự án A: Phần mềm quản lý điểm cho trường cấp 2, yêu cầu rõ ràng**

* **Mô hình phù hợp**: **Waterfall**
* **Lý do**:
  + **Yêu cầu rõ ràng**: Dự án có yêu cầu được xác định rõ từ đầu (quản lý điểm, tính năng cơ bản như nhập điểm, báo cáo, v.v.), phù hợp với mô hình Waterfall, vốn yêu cầu các giai đoạn (phân tích, thiết kế, phát triển, kiểm thử, triển khai) được thực hiện tuần tự.
  + **Đơn giản và ổn định**: Một hệ thống quản lý điểm cho trường cấp 2 thường có quy mô nhỏ, ít thay đổi yêu cầu trong quá trình phát triển. Waterfall đảm bảo quy trình có cấu trúc, dễ quản lý và tiết kiệm chi phí khi yêu cầu không thay đổi.
  + **Rủi ro thấp**: Với yêu cầu rõ ràng, rủi ro về sự mơ hồ hoặc thay đổi là tối thiểu, không cần các vòng lặp phức tạp như Spiral hay tính linh hoạt cao như Agile.

**2. Dự án B: Ứng dụng mobile đặt lịch khám bệnh, yêu cầu linh hoạt**

* **Mô hình phù hợp**: **Agile**
* **Lý do**:
  + **Yêu cầu linh hoạt**: Ứng dụng mobile đặt lịch khám bệnh thường cần điều chỉnh liên tục dựa trên phản hồi từ người dùng (bệnh nhân, bác sĩ) hoặc các yêu cầu mới (như tích hợp thanh toán, thông báo đẩy). Agile hỗ trợ phát triển lặp (iteration), cho phép phát triển từng tính năng nhỏ (sprint) và điều chỉnh theo phản hồi.
  + **Tập trung vào người dùng**: Agile phù hợp với các dự án mobile, nơi trải nghiệm người dùng (UX) là yếu tố quan trọng. Các sprint ngắn cho phép thử nghiệm và cải thiện giao diện, tính năng nhanh chóng.
  + **Thích ứng với thay đổi**: Yêu cầu linh hoạt có thể dẫn đến thay đổi thường xuyên (ví dụ: thêm tính năng chat với bác sĩ). Agile cho phép đội ngũ thích ứng mà không làm gián đoạn tiến độ tổng thể.

**3. Dự án C: Hệ thống ngân hàng điện tử có bảo mật và độ phức tạp cao**

* **Mô hình phù hợp**: **Spiral**
* **Lý do**:
  + **Độ phức tạp và rủi ro cao**: Hệ thống ngân hàng điện tử đòi hỏi bảo mật cao (xác thực, mã hóa), tích hợp nhiều hệ thống (thanh toán, chuyển khoản, quản lý tài khoản), và tuân thủ quy định pháp lý. Mô hình Spiral tập trung vào quản lý rủi ro thông qua các vòng lặp, mỗi vòng bao gồm phân tích rủi ro, lập kế hoạch, phát triển, và đánh giá.
  + **Phát triển từng phần**: Spiral cho phép phát triển hệ thống theo từng giai đoạn (ví dụ: bắt đầu với module đăng nhập, sau đó thêm chuyển khoản), kiểm tra và giảm thiểu rủi ro trước khi tiến xa hơn. Điều này rất quan trọng với một hệ thống phức tạp như ngân hàng điện tử.
  + **Yêu cầu bảo mật**: Việc đánh giá rủi ro bảo mật ở mỗi vòng lặp của Spiral đảm bảo các lỗ hổng được phát hiện và khắc phục sớm, phù hợp hơn so với Waterfall (thiếu linh hoạt) hay Agile (ít tập trung vào rủi ro phức tạp).

Bài 3:

### 1. Con người

* **Mô tả**: Các bên liên quan trong hệ thống đặt món ăn online bao gồm:
  + **Khách hàng**: Người dùng đặt món ăn qua ứng dụng, chọn nhà hàng, món ăn, và thanh toán. Họ cần giao diện thân thiện và thông tin rõ ràng.
  + **Nhà hàng/Đối tác cung cấp**: Chuẩn bị món ăn, quản lý menu, và xử lý đơn hàng từ hệ thống. Họ cần công cụ quản lý đơn hàng hiệu quả.
  + **Tài xế giao hàng**: Nhận đơn từ hệ thống, giao món ăn đến khách hàng. Họ cần ứng dụng để theo dõi đơn hàng và định vị.
  + **Nhân viên hỗ trợ khách hàng**: Xử lý khiếu nại, hoàn tiền, hoặc hỗ trợ kỹ thuật.
  + **Quản trị viên hệ thống**: Quản lý nền tảng, giám sát hoạt động, và đảm bảo hệ thống hoạt động ổn định. Các nhóm này tương tác với hệ thống để đảm bảo quy trình đặt món ăn diễn ra suôn sẻ.

### 2. Dữ liệu

* **Mô tả**: Dữ liệu là nền tảng của hệ thống, bao gồm:
  + **Dữ liệu khách hàng**: Thông tin cá nhân (tên, số điện thoại, địa chỉ), lịch sử đặt hàng, phương thức thanh toán.
  + **Dữ liệu nhà hàng**: Menu, giá cả, giờ hoạt động, vị trí, đánh giá, và trạng thái đơn hàng.
  + **Dữ liệu đơn hàng**: Chi tiết món ăn, số lượng, giá, thời gian đặt, trạng thái giao hàng (đang xử lý, đang giao, hoàn tất).
  + **Dữ liệu tài xế**: Vị trí thời gian thực, lịch sử giao hàng, đánh giá hiệu suất.
  + **Dữ liệu giao dịch**: Lịch sử thanh toán, mã giảm giá, chi phí giao hàng. Dữ liệu được lưu trữ trong cơ sở dữ liệu (như SQL hoặc NoSQL) và cần bảo mật cao (mã hóa, tuân thủ GDPR/tiêu chuẩn địa phương).

### 3. Quy trình

* **Mô tả**: Quy trình là các bước vận hành hệ thống, bao gồm:
  + **Tìm kiếm và lựa chọn**: Khách hàng tìm nhà hàng, xem menu, và chọn món ăn.
  + **Đặt hàng**: Khách hàng thêm món vào giỏ hàng, chọn phương thức thanh toán, và xác nhận đơn hàng.
  + **Xử lý đơn hàng**: Nhà hàng nhận thông báo, xác nhận, và chuẩn bị món ăn.
  + **Phân phối đơn hàng**: Hệ thống gán tài xế dựa trên vị trí và tình trạng sẵn có; tài xế nhận món và giao đến khách hàng.
  + **Phản hồi và đánh giá**: Khách hàng đánh giá món ăn/tài xế, và hệ thống thu thập phản hồi để cải thiện dịch vụ. Các quy trình này được tự động hóa thông qua phần mềm nhưng có thể cần can thiệp thủ công (ví dụ: hỗ trợ khách hàng).

### 4. Phần mềm

* **Mô tả**: Phần mềm là nền tảng kỹ thuật số hỗ trợ hệ thống, bao gồm:
  + **Ứng dụng di động**: Ứng dụng dành cho khách hàng (iOS/Android) để đặt món, theo dõi đơn hàng, thanh toán, và đánh giá.
  + **Ứng dụng cho tài xế**: Hỗ trợ tài xế nhận đơn, định vị, và cập nhật trạng thái giao hàng.
  + **Hệ thống quản lý nhà hàng**: Giao diện web hoặc ứng dụng cho nhà hàng để quản lý menu, đơn hàng, và doanh thu.
  + **Backend hệ thống**: Máy chủ xử lý giao dịch, quản lý dữ liệu, và tích hợp các thành phần (API để kết nối ứng dụng, thanh toán, bản đồ).
  + **Hệ thống phân tích**: Công cụ phân tích dữ liệu (như BI tools) để dự báo nhu cầu, tối ưu hóa giao hàng, hoặc đề xuất món ăn. Ví dụ công nghệ: React Native cho ứng dụng, Node.js/Go cho backend, Google Maps API cho định vị.

### 5. Phần cứng

* **Mô tả**: Phần cứng hỗ trợ vận hành hệ thống, bao gồm:
  + **Máy chủ**: Máy chủ đám mây (như AWS, Google Cloud) để lưu trữ dữ liệu, xử lý giao dịch, và đảm bảo hệ thống hoạt động liên tục.
  + **Thiết bị người dùng**: Điện thoại thông minh hoặc máy tính bảng của khách hàng và tài xế để chạy ứng dụng.
  + **Thiết bị nhà hàng**: Máy tính, máy in hóa đơn, hoặc tablet để nhận và xử lý đơn hàng.
  + **Hạ tầng mạng**: Router, hệ thống mạng để đảm bảo kết nối nhanh, ổn định giữa các bên.
  + **Hệ thống định vị**: GPS trên điện thoại tài xế để theo dõi vị trí và tối ưu hóa lộ trình giao hàng. Phần cứng cần đảm bảo độ tin cậy cao và khả năng mở rộng để xử lý lượng lớn đơn hàng.

Bài 4:

### 1. Planning (Lập kế hoạch)

* **Việc cần làm**:
  + Xác định mục tiêu của ứng dụng: tự động hóa điểm danh, giảm thời gian kiểm tra thủ công, cung cấp báo cáo cho giáo viên/quản lý.
  + Đánh giá phạm vi dự án: quy mô (số lượng sinh viên, lớp học), ngân sách, thời gian hoàn thành.
  + Xác định nguồn lực: đội ngũ phát triển, công cụ (ngôn ngữ lập trình, cơ sở dữ liệu), và phần cứng.
  + Lập kế hoạch quản lý rủi ro: ví dụ, rủi ro về bảo mật dữ liệu sinh viên hoặc lỗi hệ thống.
  + Kết quả: Tài liệu kế hoạch dự án, bao gồm lộ trình (timeline) và phân bổ nguồn lực.

### 2. Analysis (Phân tích)

* **Việc cần làm**:
  + Thu thập yêu cầu từ các bên liên quan (giáo viên, quản lý trường, sinh viên): ví dụ, điểm danh qua QR code, nhận diện khuôn mặt, hoặc nhập tay.
  + Phân tích yêu cầu chức năng: điểm danh theo buổi học, lưu lịch sử điểm danh, xuất báo cáo.
  + Phân tích yêu cầu phi chức năng: bảo mật (mã hóa dữ liệu), hiệu suất (xử lý nhanh cho hàng trăm sinh viên), khả năng mở rộng.
  + Xác định ràng buộc: ví dụ, tích hợp với hệ thống quản lý trường học hiện có.
  + Kết quả: Tài liệu yêu cầu hệ thống (SRS) chi tiết.

### 3. Design (Thiết kế)

* **Việc cần làm**:
  + Thiết kế giao diện người dùng (UI): giao diện đơn giản cho giáo viên nhập điểm danh, giao diện sinh viên quét QR hoặc xác minh danh tính.
  + Thiết kế kiến trúc hệ thống: chọn cơ sở dữ liệu (SQL cho dữ liệu sinh viên), API cho tích hợp, và công nghệ (như React Native cho ứng dụng di động).
  + Thiết kế luồng dữ liệu: cách dữ liệu điểm danh được lưu trữ, truy xuất, và báo cáo.
  + Đảm bảo bảo mật: thiết kế xác thực người dùng (OAuth, JWT) và mã hóa dữ liệu.
  + Kết quả: Tài liệu thiết kế (bản vẽ giao diện, sơ đồ cơ sở dữ liệu, kiến trúc hệ thống).

### 4. Implementation (Triển khai)

* **Việc cần làm**:
  + Viết mã nguồn cho ứng dụng: phát triển các module như điểm danh, quản lý sinh viên, xuất báo cáo.
  + Tích hợp các công nghệ: ví dụ, QR code scanner, API kết nối với hệ thống trường học, hoặc nhận diện khuôn mặt (nếu cần).
  + Xây dựng cơ sở dữ liệu: tạo bảng cho sinh viên, lớp học, lịch sử điểm danh.
  + Đảm bảo tuân thủ thiết kế: sử dụng các framework như Flutter hoặc Java Spring cho backend.
  + Kết quả: Ứng dụng hoạt động với các tính năng cốt lõi được lập trình hoàn chỉnh.

### 5. Testing (Kiểm thử)

* **Việc cần làm**:
  + Kiểm thử chức năng: đảm bảo các tính năng (quét QR, lưu điểm danh, xuất báo cáo) hoạt động đúng.
  + Kiểm thử phi chức năng: kiểm tra hiệu suất (xử lý 500 sinh viên cùng lúc), bảo mật (ngăn truy cập trái phép), và khả năng tương thích (iOS/Android).
  + Kiểm thử người dùng: mời giáo viên/sinh viên thử nghiệm để phát hiện lỗi giao diện hoặc trải nghiệm.
  + Sửa lỗi: khắc phục các vấn đề được phát hiện trong quá trình kiểm thử.
  + Kết quả: Ứng dụng ổn định, ít lỗi, sẵn sàng triển khai.

### 6. Deployment & Maintenance (Triển khai & Bảo trì)

* **Việc cần làm**:
  + **Triển khai**: Phát hành ứng dụng lên App Store/Google Play hoặc cài đặt trên máy chủ trường học. Tích hợp với hệ thống quản lý hiện có (nếu có).
  + **Đào tạo**: Hướng dẫn giáo viên, sinh viên, và quản trị viên cách sử dụng ứng dụng.
  + **Bảo trì**: Cập nhật tính năng mới (ví dụ, thêm báo cáo phân tích), sửa lỗi phát sinh, và nâng cấp bảo mật.
  + **Hỗ trợ**: Cung cấp kênh hỗ trợ (email, hotline) để xử lý vấn đề từ người dùng.
  + Kết quả: Ứng dụng vận hành ổn định trong trường học, được cập nhật định kỳ.

Bài 5:

### 1. Planning (Lập kế hoạch)

**Việc cần làm**:

* **Xác định mục tiêu và phạm vi**: Mục tiêu là xây dựng một hệ thống điểm danh tự động bằng QR code để giảm thời gian kiểm tra thủ công, tăng độ chính xác, và cung cấp thống kê cho phòng đào tạo. Phạm vi bao gồm ứng dụng di động cho sinh viên, giao diện cho giảng viên, và bảng điều khiển (dashboard) cho phòng đào tạo.
* **Đánh giá nguồn lực**: Xác định đội ngũ (lập trình viên, nhà phân tích, thiết kế UX/UI), công cụ (React Native cho ứng dụng, Firebase cho backend, công cụ tạo QR), và ngân sách.
* **Lập lộ trình**: Dự kiến thời gian phát triển (ví dụ: 6 tháng), chia thành các mốc: 1 tháng cho phân tích, 1 tháng cho thiết kế, 3 tháng cho triển khai, và 1 tháng cho kiểm thử/triển khai.
* **Quản lý rủi ro**: Xác định rủi ro như bảo mật dữ liệu sinh viên (tuân thủ GDPR hoặc quy định địa phương), lỗi quét QR, hoặc gián đoạn tích hợp với hệ thống trường học.
* **Kết quả**: Tài liệu kế hoạch dự án, bao gồm mục tiêu, ngân sách, lộ trình, và kế hoạch giảm thiểu rủi ro.
* **Sơ đồ UML liên quan**: Chưa áp dụng sơ đồ UML ở giai đoạn này, vì đây là giai đoạn lập kế hoạch cấp cao, chưa đi vào mô hình hóa hệ thống. Tuy nhiên, một **sơ đồ ngữ cảnh (Context Diagram)** có thể được phác thảo sơ bộ để mô tả các thực thể chính (giảng viên, sinh viên, phòng đào tạo) tương tác với hệ thống.

### 2. Requirement Analysis (Phân tích yêu cầu)

**Việc cần làm**:

* **Thu thập yêu cầu**:
  + **Giảng viên**: Tạo buổi học (ngày, giờ, môn học), sinh mã QR cho mỗi buổi, quét mã QR để xác nhận điểm danh, xem danh sách sinh viên đã điểm danh.
  + **Sinh viên**: Đăng nhập ứng dụng, quét mã QR để điểm danh, xem lịch sử điểm danh cá nhân.
  + **Phòng đào tạo**: Xem thống kê điểm danh (theo lớp, môn học, hoặc sinh viên), xuất báo cáo (PDF/Excel).
  + **Yêu cầu phi chức năng**: Bảo mật (xác thực người dùng, mã hóa dữ liệu), hiệu suất (xử lý nhanh cho 500 sinh viên/buổi), tương thích (iOS/Android), và tích hợp với hệ thống quản lý trường học hiện có.
* **Phân tích yêu cầu**: Xác định các luồng chính (tạo buổi học → sinh QR → quét QR → lưu điểm danh → báo cáo). Đảm bảo yêu cầu rõ ràng, khả thi, và có thể đo lường (ví dụ: thời gian quét QR dưới 5 giây).
* **Xác minh với các bên liên quan**: Tổ chức các buổi phỏng vấn hoặc khảo sát với giảng viên, sinh viên, và phòng đào tạo để đảm bảo yêu cầu đầy đủ.
* **Kết quả**: Tài liệu yêu cầu hệ thống (SRS) chi tiết, bao gồm yêu cầu chức năng, phi chức năng, và các ràng buộc.
* **Sơ đồ UML liên quan**:
  + **Use Case Diagram**: Mô tả các tác nhân (giảng viên, sinh viên, phòng đào tạo) và các chức năng chính (tạo buổi học, quét QR, xem thống kê).
  + **Activity Diagram**: Mô tả luồng quy trình điểm danh (từ tạo mã QR đến lưu dữ liệu điểm danh). Ví dụ: Use Case Diagram sẽ có các tác nhân (Actors) là "Giảng viên", "Sinh viên", "Phòng đào tạo", với các use case như "Tạo buổi học", "Quét QR", "Xem báo cáo".

### 3. System Design (Thiết kế hệ thống)

**Việc cần làm**:

* **Thiết kế giao diện người dùng (UI)**:
  + Ứng dụng di động cho sinh viên: Giao diện đăng nhập, quét QR, lịch sử điểm danh.
  + Giao diện cho giảng viên: Web hoặc ứng dụng để tạo buổi học, quét QR, xem danh sách điểm danh.
  + Dashboard cho phòng đào tạo: Biểu đồ, bảng thống kê, và nút xuất báo cáo.
* **Thiết kế kiến trúc hệ thống**:
  + **Frontend**: Sử dụng React Native hoặc Flutter cho ứng dụng di động (tương thích iOS/Android).
  + **Backend**: Sử dụng Node.js hoặc Spring Boot, với API RESTful để xử lý yêu cầu (tạo QR, lưu điểm danh).
  + **Cơ sở dữ liệu**: SQL (như MySQL) để lưu thông tin sinh viên, buổi học, và lịch sử điểm danh.
  + **Tích hợp**: Sử dụng thư viện QR code (ZXing) và API định vị (để xác minh vị trí quét QR, nếu cần).
* **Thiết kế bảo mật**: Xác thực người dùng qua OAuth 2.0, mã hóa dữ liệu điểm danh, và giới hạn quyền truy cập (phòng đào tạo không chỉnh sửa điểm danh).
* **Thiết kế luồng dữ liệu**: Dữ liệu từ quét QR được gửi đến backend, xác minh, lưu vào cơ sở dữ liệu, và cập nhật thống kê.
* **Kết quả**: Tài liệu thiết kế hệ thống, bao gồm sơ đồ kiến trúc, giao diện mẫu (mockup), và sơ đồ cơ sở dữ liệu.
* **Sơ đồ UML liên quan**:
  + **Class Diagram**: Mô tả các thực thể chính (Student, Teacher, Attendance, Course) và mối quan hệ (ví dụ: một Course có nhiều Attendance).
  + **Sequence Diagram**: Mô tả tương tác giữa các thành phần (sinh viên quét QR → ứng dụng gửi yêu cầu → backend xác minh → lưu điểm danh).
  + **Component Diagram**: Mô tả các thành phần hệ thống (ứng dụng di động, backend, cơ sở dữ liệu).
  + **Deployment Diagram**: Mô tả cách triển khai (máy chủ đám mây, thiết bị di động, mạng).

### Sơ đồ UML tương ứng với hệ thống điểm danh

Dựa trên mô tả hệ thống, các sơ đồ UML phù hợp nhất để mô hình hóa là:

1. **Use Case Diagram**: Mô tả các tác nhân (giảng viên, sinh viên, phòng đào tạo) và chức năng chính (tạo buổi học, quét QR, xem thống kê).
   * **Lý do**: Giúp xác định rõ ràng các vai trò và tương tác chính với hệ thống.
2. **Activity Diagram**: Mô tả quy trình điểm danh (tạo mã QR → quét → xác minh → lưu dữ liệu).
   * **Lý do**: Thể hiện luồng hoạt động chi tiết, đặc biệt là quy trình quét QR và xử lý dữ liệu.
3. **Sequence Diagram**: Mô tả tương tác giữa sinh viên, ứng dụng, và backend khi quét QR.
   * **Lý do**: Làm rõ trình tự các bước và thông điệp giữa các thành phần.
4. **Class Diagram**: Mô tả các đối tượng chính (Student, Course, Attendance) và mối quan hệ.
   * **Lý do**: Cung cấp cấu trúc dữ liệu cho cơ sở dữ liệu và logic hệ thống.
5. **Component Diagram**: Mô tả các thành phần phần mềm (ứng dụng, backend, cơ sở dữ liệu).
   * **Lý do**: Hỗ trợ thiết kế kiến trúc hệ thống và tích hợp.

Bài 6:

### Tình huống A: Mô tả chức năng người dùng có thể thực hiện trên ứng dụng học tiếng Anh

* **Sơ đồ UML phù hợp**: **Use Case Diagram**
* **Lý do**:
  + Use Case Diagram được sử dụng để mô tả các chức năng mà người dùng (học viên, giảng viên, hoặc quản trị viên) có thể thực hiện trên hệ thống, như đăng nhập, học bài, làm bài kiểm tra, xem kết quả, v.v.
  + Sơ đồ này thể hiện các tác nhân (actors) và các trường hợp sử dụng (use cases), giúp làm rõ các tương tác chính giữa người dùng và ứng dụng.
  + Ví dụ: Tác nhân "Học viên" có các use case như "Đăng nhập", "Học bài", "Làm bài kiểm tra", "Xem điểm".

### Tình huống B: Mô tả lớp NguoiDung, KhoaHoc, BaiHoc và quan hệ giữa chúng

* **Sơ đồ UML phù hợp**: **Class Diagram**
* **Lý do**:
  + Class Diagram được sử dụng để mô tả các lớp (NguoiDung, KhoaHoc, BaiHoc) và mối quan hệ giữa chúng (như quan hệ một-nhiều, nhiều-nhiều, hoặc kế thừa).
  + Ví dụ: Lớp NguoiDung có thể có quan hệ "đăng ký" với nhiều KhoaHoc, và một KhoaHoc có nhiều BaiHoc. Sơ đồ này làm rõ các thuộc tính (ví dụ: tên, email của NguoiDung) và phương thức (ví dụ: registerCourse()).
  + Sơ đồ này hữu ích cho việc thiết kế cơ sở dữ liệu và cấu trúc mã nguồn.

### Tình huống C: Mô tả luồng học viên bắt đầu → vào học → làm bài → hoàn thành

* **Sơ đồ UML phù hợp**: **Activity Diagram**
* **Lý do**:
  + Activity Diagram được sử dụng để mô tả luồng hoạt động hoặc quy trình chi tiết, như các bước mà học viên thực hiện từ khi bắt đầu học đến khi hoàn thành.
  + Sơ đồ này thể hiện các bước tuần tự (bắt đầu → vào học → làm bài → hoàn thành) cùng với các quyết định (ví dụ: nếu bài kiểm tra đạt, chuyển sang bài tiếp theo).
  + Nó giúp làm rõ quy trình nghiệp vụ và các điểm phân nhánh trong luồng học tập.

### Tình huống D: Mô tả cách hệ thống triển khai trên các máy chủ, thiết bị

* **Sơ đồ UML phù hợp**: **Deployment Diagram**
* **Lý do**:
  + Deployment Diagram mô tả cách các thành phần phần mềm được triển khai trên phần cứng, như máy chủ ứng dụng, cơ sở dữ liệu, hoặc thiết bị di động của học viên.
  + Ví dụ: Ứng dụng học tiếng Anh có thể được triển khai trên máy chủ đám mây (AWS), với ứng dụng di động chạy trên điện thoại iOS/Android, và cơ sở dữ liệu lưu trên một node riêng.
  + Sơ đồ này giúp các nhà phát triển hiểu rõ kiến trúc vật lý và phân phối hệ thống.

### Tình huống E: Mô tả thứ tự tương tác giữa học viên và hệ thống khi nộp bài

* **Sơ đồ UML phù hợp**: **Sequence Diagram**
* **Lý do**:
  + Sequence Diagram mô tả thứ tự các tương tác giữa các đối tượng (học viên, ứng dụng, backend) theo thời gian, đặc biệt khi học viên nộp bài.
  + Ví dụ: Học viên nhấn nút "Nộp bài" → ứng dụng gửi yêu cầu đến backend → backend xử lý và lưu kết quả → trả về thông báo hoàn thành.
  + Sơ đồ này làm rõ các thông điệp, thời điểm, và trình tự xử lý giữa các thành phần.

Bài 7:

### 1. Planning (Lập kế hoạch)

* **Việc cần làm**:
  + Xác định mục tiêu dự án: Xây dựng hệ thống online để giảm tải thủ công, tăng tốc độ đăng ký tiêm chủng, đảm bảo an toàn dữ liệu y tế (tuân thủ HIPAA hoặc quy định Việt Nam về bảo vệ dữ liệu cá nhân).
  + Đánh giá phạm vi: Bao gồm đăng ký người dùng, xác nhận lịch tiêm, quản lý danh sách, nhưng loại trừ tích hợp thanh toán (nếu không cần).
  + Phân tích nguồn lực: Xác định đội ngũ (5-7 người: lập trình viên, nhà phân tích, tester, chuyên gia y tế làm cố vấn), ngân sách (ước tính 500-1000 triệu VND), thời gian (6-9 tháng).
  + Lập lộ trình: Chia timeline (ví dụ: 1 tháng lập kế hoạch, 2 tháng phân tích/thiết kế), quản lý rủi ro (như gián đoạn do dịch bệnh hoặc thay đổi quy định y tế).
  + Thu thập ý kiến ban đầu từ trung tâm y tế và người dân qua khảo sát.
* **Kết quả**: Tài liệu kế hoạch dự án (Project Charter), bao gồm mục tiêu, timeline, ngân sách, và kế hoạch rủi ro.

### 2. Analysis (Phân tích yêu cầu)

* **Việc cần làm**:
  + Thu thập yêu cầu chức năng: Người dân đăng ký (form nhập thông tin: CMND, địa chỉ, lịch sử bệnh, chọn vaccine); trung tâm xác nhận (kiểm tra dữ liệu, xếp lịch dựa trên nguồn vaccine sẵn có, gửi email/SMS thông báo); quản lý danh sách (lọc người đã tiêm/chưa tiêm, báo cáo thống kê theo khu vực/độ tuổi).
  + Phân tích yêu cầu phi chức năng: Bảo mật dữ liệu (mã hóa, xác thực 2 lớp), hiệu suất (xử lý 1000 đăng ký/ngày), khả năng mở rộng (tích hợp với hệ thống quốc gia nếu cần), và tính thân thiện (hỗ trợ tiếng Việt, giao diện di động).
  + Phỏng vấn các bên liên quan: Người dân (qua khảo sát online), nhân viên trung tâm (họp trực tiếp), và chuyên gia y tế để đảm bảo tuân thủ quy trình tiêm chủng.
  + Xác định ràng buộc: Tích hợp với hệ thống y tế hiện có (như cơ sở dữ liệu vaccine quốc gia).
* **Kết quả**: Tài liệu yêu cầu hệ thống (SRS - Software Requirements Specification), bao gồm mô tả chi tiết yêu cầu, use case, và ưu tiên tính năng.

### 3. Design (Thiết kế hệ thống)

* **Việc cần làm**:
  + Thiết kế giao diện người dùng (UI/UX): Form đăng ký đơn giản cho người dân (web/mobile), dashboard cho trung tâm để xác nhận lịch và theo dõi danh sách (biểu đồ, bảng lọc).
  + Thiết kế kiến trúc hệ thống: Frontend (React.js cho web, React Native cho app); Backend (Node.js hoặc Spring Boot với API RESTful); Cơ sở dữ liệu (MongoDB hoặc MySQL để lưu thông tin đăng ký và trạng thái tiêm).
  + Thiết kế bảo mật và luồng dữ liệu: Xác thực người dùng (OAuth), mã hóa dữ liệu y tế; luồng: Đăng ký → Xác nhận → Cập nhật danh sách → Báo cáo.
  + Tích hợp công cụ bên thứ ba: SMS/Email API (Twilio hoặc Viettel), tạo mã QR cho lịch tiêm nếu cần.
  + Vẽ sơ đồ UML: Use Case Diagram (chức năng chính), Class Diagram (lớp NgườiDân, LichTiem, DanhSach), Sequence Diagram (luồng đăng ký).
* **Kết quả**: Tài liệu thiết kế hệ thống (SDS), bao gồm mockup giao diện, sơ đồ kiến trúc, và cơ sở dữ liệu.

### 4. Implementation (Triển khai phát triển)

* **Việc cần làm**:
  + Phát triển mã nguồn: Xây dựng module đăng ký (form validation, lưu dữ liệu), module xác nhận (tự động xếp lịch dựa trên thuật toán ưu tiên), module quản lý danh sách (tìm kiếm, xuất báo cáo PDF/Excel).
  + Tích hợp các thành phần: Kết nối frontend-backend, thêm tính năng thông báo (push notification cho app).
  + Đảm bảo tuân thủ thiết kế: Sử dụng Git cho quản lý mã, thực hiện code review định kỳ.
  + Phát triển theo sprint (kết hợp Agile): Hoàn thành module đăng ký trước, sau đó là xác nhận và quản lý.
  + Kiểm tra sơ bộ: Unit test cho từng module (ví dụ: test đăng ký hợp lệ).
* **Kết quả**: Phiên bản phần mềm hoàn chỉnh, sẵn sàng kiểm thử, với mã nguồn được lưu trữ và tài liệu code.

### 5. Testing (Kiểm thử)

* **Việc cần làm**:
  + Kiểm thử chức năng: Đảm bảo đăng ký thành công, xác nhận lịch chính xác, danh sách cập nhật đúng (ví dụ: test đăng ký giả với 100 người dùng).
  + Kiểm thử phi chức năng: Bảo mật (penetration testing để phát hiện lỗ hổng), hiệu suất (load testing với 500 người dùng đồng thời), tương thích (web/mobile trên các thiết bị).
  + Kiểm thử người dùng (UAT): Mời nhân viên trung tâm và người dân thử nghiệm thực tế, thu thập phản hồi.
  + Sửa lỗi: Khắc phục bug phát hiện, tái kiểm thử.
  + Kiểm thử tích hợp: Đảm bảo hệ thống kết nối tốt với API bên ngoài (SMS, cơ sở dữ liệu y tế).
* **Kết quả**: Báo cáo kiểm thử, phần mềm ổn định với ít lỗi, đạt tiêu chuẩn chất lượng (ví dụ: 95% test case pass).

### 6. Deployment & Maintenance (Triển khai và bảo trì)

* **Việc cần làm**:
  + **Triển khai**: Phát hành hệ thống lên server đám mây (AWS hoặc Viettel Cloud), tích hợp với website trung tâm, và app store (Google Play/App Store). Thực hiện rollout dần dần (pilot với một khu vực trước).
  + **Đào tạo**: Tổ chức hội thảo hướng dẫn cho nhân viên trung tâm (xác nhận lịch) và hướng dẫn người dân (qua video/tutorial trên web).
  + **Bảo trì**: Giám sát hệ thống (sử dụng công cụ như Prometheus), cập nhật tính năng mới (ví dụ: thêm nhắc nhở liều thứ 2), sửa lỗi phát sinh, và nâng cấp bảo mật định kỳ.
  + **Hỗ trợ**: Thiết lập kênh hỗ trợ (hotline, chat online) để xử lý vấn đề từ người dùng.
  + **Đánh giá**: Thu thập feedback sau 3 tháng để cải tiến (ví dụ: thêm tích hợp với ứng dụng sức khỏe quốc gia).
* **Kết quả**: Hệ thống hoạt động ổn định, với kế hoạch bảo trì dài hạn và báo cáo hiệu suất

Bài 8:

### 1. Xác định các tác nhân chính và chức năng tương ứng

Các tác nhân chính (actors) trong hệ thống nền tảng học trực tuyến bao gồm ba vai trò chính dựa trên chức năng được mô tả. Tôi sẽ liệt kê từng tác nhân và các chức năng tương ứng:

* **Học viên (Student/Learner)**:
  + Đăng ký tài khoản và xác thực (sign up, login).
  + Tìm kiếm và đăng ký khóa học (enroll in courses).
  + Tham gia khóa học: Xem bài học, làm bài tập, kiểm tra (access lessons, submit assignments, take quizzes).
  + Xem điểm số cá nhân và tiến độ học tập (view grades and progress).
* **Giảng viên (Instructor/Teacher)**:
  + Quản lý bài học: Tạo, chỉnh sửa, tải lên nội dung bài học (create/edit lessons, upload materials).
  + Quản lý điểm số: Chấm bài tập, kiểm tra, cập nhật điểm (grade assignments, update scores).
  + Theo dõi học viên: Xem tiến độ lớp học, gửi phản hồi (monitor student progress, provide feedback).
* **Admin (Administrator)**:
  + Theo dõi báo cáo và thống kê: Xem dữ liệu tổng hợp về khóa học, học viên, hiệu suất (generate reports on enrollment, completion rates).
  + Phân quyền người dùng: Quản lý vai trò (assign roles, permissions for users).
  + Quản lý hệ thống: Cập nhật nội dung chung, xử lý vấn đề kỹ thuật (system maintenance, user management).

### 2. Phân loại hệ thống thông tin phù hợp

Dựa trên các loại hệ thống thông tin (Information Systems), hệ thống này có thể được phân loại như sau. Tôi sẽ phân loại từng chức năng chính để phù hợp:

* **TPS (Transaction Processing System - Hệ thống xử lý giao dịch)**: Phù hợp cho các chức năng cơ bản như đăng ký khóa học, tham gia bài học, cập nhật điểm số. Đây là hệ thống xử lý giao dịch hàng ngày, đảm bảo dữ liệu được ghi nhận thời gian thực và chính xác (ví dụ: học viên đăng ký khóa học, giảng viên cập nhật điểm).
* **MIS (Management Information System - Hệ thống thông tin quản lý)**: Phù hợp cho chức năng quản lý bài học, theo dõi báo cáo và thống kê của admin. MIS cung cấp báo cáo tổng hợp định kỳ (như thống kê số lượng học viên, tỷ lệ hoàn thành khóa học) để hỗ trợ quản lý trung cấp và cấp cao.
* **DSS (Decision Support System - Hệ thống hỗ trợ quyết định)**: Phù hợp cho các phân tích sâu hơn, như dự báo xu hướng đăng ký khóa học hoặc phân tích hiệu suất giảng viên dựa trên dữ liệu thống kê. DSS sử dụng mô hình phân tích để hỗ trợ ra quyết định (ví dụ: admin quyết định mở khóa học mới dựa trên dữ liệu).

Tổng thể, hệ thống này là sự kết hợp giữa **TPS** (nền tảng giao dịch) và **MIS** (báo cáo quản lý), với yếu tố **DSS** nếu tích hợp phân tích nâng cao. Không cần EIS (Executive Information System) vì không tập trung vào dashboard chiến lược cấp cao.

### 3. Đề xuất mô hình phát triển phần mềm phù hợp nhất

**Mô hình phù hợp nhất: Agile (cụ thể là Scrum hoặc Kanban)**.

**Lý do**:

* **Yêu cầu linh hoạt**: Hệ thống học trực tuyến thường cần điều chỉnh thường xuyên dựa trên phản hồi từ học viên/giảng viên (ví dụ: thêm tính năng học qua video hoặc tích hợp chat), phù hợp với Agile vì cho phép phát triển lặp (iterations/sprints) và thay đổi yêu cầu mà không làm gián đoạn.
* **Tập trung vào người dùng**: Agile ưu tiên phản hồi sớm (qua demo sản phẩm sau mỗi sprint), giúp đảm bảo hệ thống thân thiện và đáp ứng nhu cầu thực tế của trung tâm ngoại ngữ.
* **Quy mô vừa phải**: Không quá phức tạp như hệ thống ngân hàng (phù hợp Waterfall), nhưng cần thích ứng nhanh với thay đổi công nghệ giáo dục (không nên dùng Spiral với rủi ro cao).
* **Lợi ích**: Giảm rủi ro thất bại bằng cách phát hành phiên bản MVP (Minimum Viable Product) sớm, và dễ tích hợp với các công cụ như Jira cho quản lý task.

Nếu yêu cầu rất rõ ràng và không thay đổi, có thể dùng Waterfall, nhưng Agile phù hợp hơn cho dự án giáo dục trực tuyến.

### 4. Nêu tên 3 sơ đồ UML bạn sẽ sử dụng nếu đi sâu thiết kế hệ thống này

Nếu đi sâu vào thiết kế, tôi sẽ sử dụng các sơ đồ UML sau để mô hình hóa hệ thống:

* **Use Case Diagram**: Để mô tả các tác nhân (học viên, giảng viên, admin) và chức năng chính (đăng ký khóa học, quản lý bài học, theo dõi báo cáo), giúp làm rõ tương tác giữa người dùng và hệ thống.
* **Class Diagram**: Để mô tả cấu trúc dữ liệu, như các lớp (User, Course, Lesson, Grade) và mối quan hệ giữa chúng (ví dụ: một Course có nhiều Lesson, một User có vai trò khác nhau).
* **Sequence Diagram**: Để mô tả thứ tự tương tác giữa các thành phần khi thực hiện một chức năng cụ thể, như quy trình học viên đăng ký khóa học (học viên → hệ thống → giảng viên xác nhận).