ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH

BÁO CÁO BÀI TẬP LỚN

KIẾN TRÚC PHẦN MỀM

ĐỀ TÀI:
HỆ THỐNG GIA SƯ THÔNG MINH (ITS)

Giảng viên hướng dẫn: Trần Trương Tuấn Phát

Sinh viên thực hiện:

Nguyễn Anh Khoa	2211614
Phạm Quang Minh	2212075
Đoàn Ngọc Hoàng Sơn	2212935
Nguyễn Văn Sơn	2212949
Trần Nam Sơn	2212956
Nguyễn Hiệp Tài	2212985
Huỳnh Cảnh Thịnh	2213272

TP. HỒ CHÍ MINH, THÁNG 12 NĂM 2025

Mục lục

Danh sách hình vẽ

Danh sách bảng

1 Giới thiệu

1.1 Đặt vấn đề

Trong môi trường giáo dục truyền thống, việc giảng dạy chủ yếu diễn ra theo phương pháp đồng bộ, nơi giáo viên truyền đạt kiến thức chung cho cả lớp. Điều này tạo ra một số thách thức lớn:

Giới hạn trong học tập cá nhân hóa

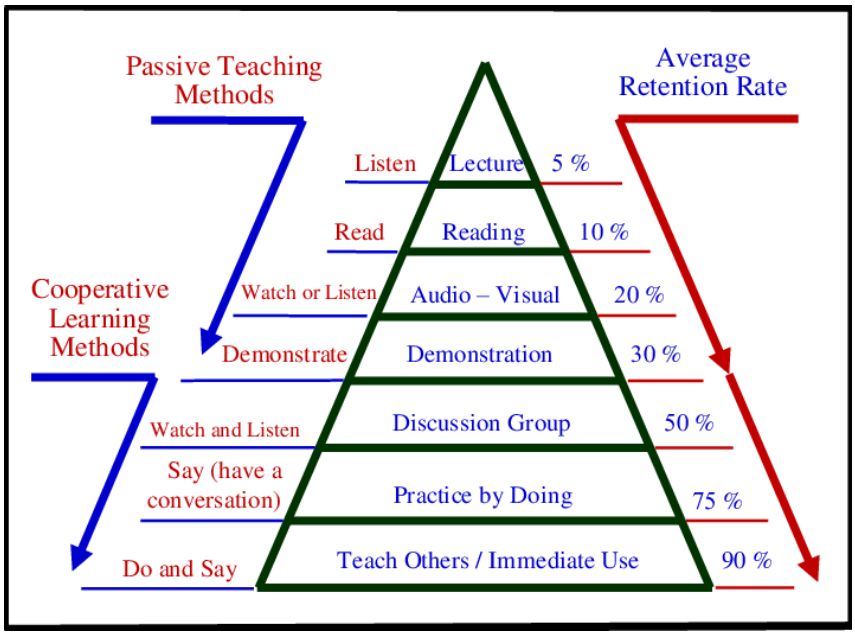
Việc giảng dạy truyền thống khiến giáo viên gặp khó khăn trong việc đáp ứng nhu cầu, tốc độ và phong cách học tập đa dạng của từng học viên.

- Tốc độ và nhu cầu: Mỗi học viên có tốc độ học khác nhau, nhưng thời gian và nguồn lực hạn chế khiến việc cung cấp trải nghiệm học tập cá nhân hóa, phản hồi tức thì và hỗ trợ thích ứng theo thời gian thực trở thành một thách thức lớn.
- Thiếu hiệu quả: Điều này dẫn đến sự thiếu hiệu quả trong việc phát huy tối đa tiềm năng của mỗi người học.

Sự thống trị của phương pháp giảng dạy thụ động

Các phương pháp giảng dạy truyền thống, nơi thông tin chảy một chiều từ người hướng dẫn đến học viên, thường dựa trên các hình thức thụ động.

- Tỷ lệ lưu giữ kiến thức thấp: Theo nguyên lý của Tam giác Học tập (Learning Pyramid)[1], các phương pháp thụ động như Listen hay Read có tỷ lệ lưu giữ trung bình rất thấp (chỉ 5% - 10%).



Hình 1: *Tam giác học tập*

- Thiếu kỹ năng mềm: Các phương pháp này không tạo điều kiện cho học viên tham gia vào các hoạt động học tập chủ động như giải quyết vấn đề, làm việc nhóm, hoặc giao tiếp , vốn là những kỹ năng mềm thiết yếu mà các chương trình kỹ thuật cần phải nâng cao.

Hạn chế trong đánh giá và theo dõi tiến độ học tập

Hệ thống giáo dục truyền thống chủ yếu dựa vào các bài kiểm tra định kỳ như giữa kỳ và cuối kỳ, dẫn đến việc đánh giá không phản ánh đầy đủ quá trình học tập của học viên.

- Thiếu dữ liệu thời gian thực: Giáo viên không thể theo dõi tiến độ của từng học viên theo từng buổi học, dẫn đến khó phát hiện sớm những lỗ hổng kiến thức.
- Quá trình bị gián đoạn: Học viên chỉ nhận được phản hồi sau khi hoàn thành bài kiểm tra, nên khó điều chỉnh chiến lược học tập ngay lập tức.
- Không khuyến khích sự phát triển liên tục: Khi đánh giá chỉ tập trung vào điểm số, việc hình thành năng lực lâu dài và tư duy phản biện bị xem nhẹ.

[1] https://www.educationcorner.com/the-learning-pyramid/

Thiếu đa dạng trong nội dung học tập và phương tiện giảng dạy

Phần lớn nội dung trong mô hình truyền thống được thiết kế theo chuẩn chương trình chung, không phân tầng theo trình độ và không linh hoạt theo nhu cầu cá nhân.

- Ít mức độ truy cập: Học viên không có cơ hội lựa chọn nội dung phù hợp với khả năng—ví dụ: nâng cao, bổ sung cơ bản hoặc nội dung mở rộng.
- Thiếu phương tiện tương tác: Việc phụ thuộc vào sách giáo khoa và bài giảng trên lớp hạn chế việc tiếp cận tài liệu multimedia hoặc công cụ mô phỏng trực quan.
- Cản trở mô hình học tập tự định hướng: Học viên khó tiếp cận tài liệu ở mọi lúc, mọi nơi, dẫn đến sự bị động trong việc xây dựng lộ trình học riêng.

1.2 Mô tả ý tưởng

Để giải quyết những hạn chế của giáo dục truyền thống và tối đa hóa hiệu quả học tập, Hệ thống Gia sư Thông minh (ITS) đã được nghiên cứu và phát triển. ITS là một ứng dụng phần mềm mạnh mẽ sử dụng các kỹ thuật Trí tuệ Nhân tạo (AI) để mô phỏng và cung cấp hướng dẫn cá nhân hóa. ITS tự điều chỉnh linh hoạt theo tốc độ, khả năng, điểm mạnh, điểm yếu và sở thích của từng người học thông qua các tính năng tiên tiến, tập trung vào học tập chủ động:

- Tăng cường thực hành: ITS cung cấp bài tập thực hành thiết kế theo thời gian thực, kèm theo phản hồi thích ứng để đảm bảo học viên liên tục áp dụng kiến thức và phát triển kỹ năng giải quyết vấn đề.
- Hỗ trợ kỹ năng mềm: ITS tích hợp các module hỗ trợ thảo luận nhóm, giúp học viên luyện tập giao tiếp và hợp tác.
- Tự đánh giá và giải thích: Để đạt mức độ lưu giữ kiến thức cao nhất, ITS yêu cầu học viên giải thích các khái niệm thiết kế cho gia sư AI, buộc họ phải trở thành "người dạy", từ đó củng cố khả năng tự điều chỉnh (Self-Regulation) và hiểu sâu.
- Lộ trình học tập tùy chỉnh: AI của hệ thống phân tích chi tiết hoạt động học tập của học viên để tự động điều chỉnh lộ trình, gợi ý các tài liệu (như video, bài đọc) và chủ đề tiếp theo có độ khó phù hợp, thay thế cho chương trình giảng dạy chung cứng nhắc.
- Đánh giá liên tục: Thay vì chỉ dựa vào kiểm tra định kỳ, ITS cung cấp các bài kiểm tra ngắn sau mỗi nội dung của chương. Hệ thống AI có khả năng tự động tạo ra các câu hỏi đánh giá mới, đa dạng, phân tầng theo độ khó và tập trung vào các lỗ hổng kiến thức cụ thể của từng học viên.
- Phát hiện sớm lỗ hổng kiến thức: Mọi tương tác của học viên đều được ghi lại và phân tích chi tiết. Điều này giúp phát hiện sớm các lĩnh vực mà học viên đang gặp khó khăn, cho phép can thiệp và hỗ trợ kịp thời, đảm bảo không bỏ sót bất kỳ lỗ hổng kiến thức nào.

Nhờ những tính năng này, ITS cung cấp phản hồi, gợi ý và lộ trình học tập tùy chỉnh, mang lại hiệu quả tương đương với mô hình gia sư một kèm một, giúp người học phát huy tối đa tiềm năng của mình.

2 Cơ sở lý thuyết và công nghệ

Hệ thống được xây dựng trên nền tảng Java/Spring Boot kết hợp với kiến trúc phân tán Spring Cloud và giao diện người dùng React. Việc lựa chọn bộ công nghệ này nhằm tận dụng khả năng phát triển nhanh, tính ổn định cao và sự hỗ trợ mạnh mẽ cho kiến trúc Service-Based Architecture – SBA.

2.1 Công nghệ Lõi và Backend

Các dịch vụ nghiệp vụ và hỗ trợ (Domain & Infrastructure Services) đều sử dụng hệ sinh thái Spring.

- Spring Boot: Đây là framework phát triển lõi cho toàn bộ các dịch vụ backend, bao gồm Identity Access Management, Quiz, System, Learning, Course, AI, và Service Registry. Spring Boot cung cấp môi trường chạy độc lập và cấu hình tự động (auto-configuration), đảm bảo mỗi module là một đơn vị triển khai rời rạc và dễ dàng quản lý.
- Spring Cloud Gateway: Công nghệ này được sử dụng để triển khai module API Gateway. Nó đóng vai trò là điểm truy cập duy nhất cho client, chịu trách nhiệm chính trong việc định tuyến yêu cầu tới dịch vụ tương ứng và thực hiện lọc yêu cầu (filtering) trước khi đến các dịch vụ nghiệp vụ.
- Spring Cloud Discovery (Eureka Server): Đây là công nghệ nền tảng cho module Service Registry. Nó đóng vai trò là máy chủ đăng ký dịch vụ, lưu trữ metadata và danh sách các instance đang chạy của các dịch vụ.
- Spring Cloud Discovery (Eureka Client): Tất cả các Service nghiệp vụ (IAM, Quiz, System, Learning, Course, AI) và API Gateway đều được cấu hình là Eureka Client. Điều này cho phép chúng tự động đăng ký với Eureka Server và tận dụng cơ chế khám phá dịch vụ (Service Discovery) để xác định instance hợp lệ tại thời điểm chạy (runtime).
- Spring AI: Đây là thư viện được sử dụng trong AI Service. Nó hỗ trợ việc tích hợp các mô hình Trí tuệ Nhân tạo (AI Models) bên ngoài một cách dễ dàng. Spring AI giúp AI Service thực hiện các chức năng như phân tích hành vi và tiến trình học, cũng như đề xuất lộ trình học tập phù hợp dựa trên dữ liệu phân tích.

Hình 2: *Các công nghệ Back-end*

2.2 Công nghệ Frontend

React là một thư viện JavaScript mã nguồn mở do Facebook phát triển, được sử dụng rộng rãi để xây dựng giao diện người dùng (User Interface – UI), đặc biệt là các ứng dụng web có tính tương tác cao. React cho phép nhà phát triển xây dựng giao diện thông qua các thành phần (components) độc lập, có khả năng tái sử dụng, giúp tổ chức mã nguồn rõ ràng và dễ bảo trì hơn.

Một trong những ưu điểm nổi bật của React là sử dụng Virtual DOM – một cấu trúc DOM ảo giúp tối ưu hóa hiệu năng. Khi giao diện thay đổi, React chỉ cập nhật những phần tử cần thiết thay vì render lại toàn bộ trang, nhờ đó cải thiện tốc độ xử lý và trải nghiệm của người dùng.

React hỗ trợ mô hình lập trình declarative (khai báo), cho phép mô tả giao diện theo trạng thái của ứng dụng. Khi trạng thái thay đổi, React tự động cập nhật UI tương ứng, giảm thiểu lỗi và đơn giản hóa việc quản lý vòng đời giao diện.

Ngoài ra, React có một hệ sinh thái phát triển rộng lớn bao gồm:

- React Hooks: Cung cấp cách tiếp cận mới để quản lý state và vòng đời trong function components.
- React Router: Hỗ trợ điều hướng nhiều trang trong ứng dụng single-page.

Hình 3: *React*

Nhờ vào cấu trúc linh hoạt, hiệu năng cao và cộng đồng lớn mạnh, React trở thành một trong những công nghệ phổ biến nhất được lựa chọn trong phát triển các ứng dụng web hiện đại, đặc biệt trong các dự án cần giao diện tương tác, trải nghiệm mượt mà và khả năng mở rộng tốt

2.3 Nền tảng Cloud: Amazon Web Services (AWS)

Amazon Web Services (AWS) là nền tảng điện toán đám mây hàng đầu thế giới, cung cấp hơn 200 dịch vụ toàn diện bao phủ hầu hết mọi nhu cầu về hạ tầng và ứng dụng. Với các trung tâm dữ liệu được phân bố trên toàn cầu, AWS mang đến khả năng mở rộng linh hoạt, độ sẵn sàng cao, bảo mật nhiều lớp, cùng với mô hình chi phí tối ưu theo nhu cầu sử dụng.

Hình 4: *Amazon Web Services*

AWS cung cấp đầy đủ bốn nhóm dịch vụ cốt lõi trong kiến trúc cloud hiện đại:

- Compute: EC2, ECS, Lambda cho phép triển khai ứng dụng dưới nhiều mô hình từ máy ảo, container đến serverless.
- Storage: S3, EBS, EFS hỗ trợ lưu trữ đối tượng, khối và file system với độ bền dữ liệu cao.
- Networking: VPC, Elastic Load Balancing, Route 53 cho phép kiểm soát mạng nội bộ, định tuyến và phân phối tải.
- Database: RDS, DynamoDB, Aurora cung cấp hệ quản trị cơ sở dữ liệu quan hệ và NoSQL hiệu năng cao.

Một điểm mạnh nổi bật của AWS là khả năng tự động mở rộng (auto scaling) và cân bằng tải (load balancing) giúp hệ thống đáp ứng linh hoạt khi lưu lượng tăng đột biến mà không gián đoạn hoạt động. Đồng thời, AWS áp dụng mô hình bảo mật theo nhiều lớp (multi-layer security) gồm mã hoá dữ liệu, kiểm soát truy cập theo IAM, network isolation qua VPC và giám sát liên tục với CloudWatch.

Nhờ các đặc điểm này, việc triển khai hệ thống trên AWS giúp giảm tải phần vận hành hạ tầng, tăng độ tin cậy, đồng thời cho phép nhóm phát triển tập trung vào logic ứng dụng thay vì lo lắng về máy chủ hay cấu hình mạng. Đây là nền tảng phù hợp cho các hệ thống IoT, AI và xử lý dữ liệu theo thời gian thực, đặc biệt khi yêu cầu khả năng mở rộng cao và vận hành ổn định.

2.4 Docker và Amazon Elastic Container Service (ECS)

Để đảm bảo tính nhất quán trong quá trình phát triển và triển khai, toàn bộ các thành phần backend của hệ thống được đóng gói dưới dạng Docker container. Docker cho phép đóng gói mã nguồn kèm theo môi trường chạy (runtime, thư viện, cấu hình) vào một Docker image duy nhất. Nhờ vậy, ứng dụng có thể vận hành giống nhau ở mọi môi trường — từ máy lập trình viên, môi trường staging, cho đến môi trường production. Điều này giúp giảm thiểu xung đột phụ thuộc (dependency conflict), đơn giản hoá quá trình triển khai và tăng tốc độ phát hành phiên bản mới.

Để quản lý và vận hành các container này, hệ thống sử dụng Amazon Elastic Container Service (ECS), một dịch vụ điều phối container (container orchestration) mạnh mẽ của AWS. ECS cho phép triển khai các container lên một ECS Cluster, theo dõi trạng thái của chúng và tự động khởi động lại khi xảy ra lỗi. ECS hỗ trợ hai mô hình chạy: EC2 (tự quản lý EC2 instances) và Fargate (serverless, không cần quản lý máy chủ), giúp linh hoạt tối ưu chi phí và hiệu năng.

Hình 5: *Docker và Amazon Elastic Container Service*

Thông qua ECS, hệ thống được hưởng các cơ chế vận hành tự động như:

- Tự động mở rộng (Auto Scaling): tăng hoặc giảm số lượng container dựa trên tải.
- Health Check và tự phục hồi: container lỗi sẽ được thay thế ngay lập tức để duy trì tính sẵn sàng.
- Load Balancing: tích hợp với Elastic Load Balancer (ELB) để phân phối đều lưu lượng.
- Triển khai phiên bản mới không gián đoạn: thông qua cơ chế rolling update.

Việc áp dụng Docker và ECS giúp hệ thống đạt được khả năng mở rộng linh hoạt, ổn định trong vận hành, dễ dàng bảo trì và phù hợp với kiến trúc microservices. Đồng thời, nền tảng cloud-native của ECS giúp giảm tối đa công sức quản lý hạ tầng, cho phép nhóm phát triển tập trung hoàn toàn vào logic ứng dụng.

2.5 Spring Cloud Gateway

Spring Cloud Gateway đóng vai trò là tầng định tuyến trung tâm (API Gateway) trong kiến trúc của hệ thống. Đây là một giải pháp hiện đại được xây dựng trên nền tảng Spring WebFlux và mô hình lập trình phản ứng (reactive programming), cho phép xử lý lượng lớn yêu cầu với hiệu năng cao và độ trễ thấp. Spring Cloud Gateway chịu trách nhiệm tiếp nhận toàn bộ lưu lượng từ người dùng hoặc từ API Gateway bên ngoài, sau đó định tuyến chúng đến đúng dịch vụ phía backend trong hệ thống.

Về mặt chức năng, Spring Cloud Gateway cung cấp nhiều khả năng phù hợp cho hệ thống:

- **Định tuyến linh hoạt (Dynamic Routing):** cho phép điều hướng yêu cầu tới các service khác nhau dựa trên URL, header, tham số truy vấn hoặc quy tắc tùy chỉnh.
- **Cân bằng tải (Load Balancing):** tích hợp liền mạch với Spring Cloud LoadBalancer hoặc Service Discovery để phân phối yêu cầu đều giữa các bản sao container.
- **Lọc và xử lý trước/sau (Pre/Post Filters):** hỗ trợ chèn logic xử lý như xác thực, ghi log, giới hạn tốc độ (rate limiting), sửa đổi header hoặc payload trước khi chuyển đến service đích.
- **Hỗ trợ bảo mật:** dễ dàng tích hợp với Spring Security để triển khai cơ chế xác thực, phân quyền, JWT, OAuth2.
- **Khả năng mở rộng:** được thiết kế theo kiến trúc phi đồng bộ, phù hợp cho phân tán tải cao trong môi trường container.

Với vai trò cổng vào duy nhất của hệ thống, Spring Cloud Gateway giúp đơn giản hóa giao tiếp, giảm độ phức tạp giữa client và các microservice, đồng thời tăng tính bảo mật bằng cách che giấu cấu trúc nội bộ. Nhờ hoạt động trên nền phản ứng (reactive), gateway đạt hiệu năng tốt ngay cả khi số lượng kết nối và yêu cầu tăng cao.

2.6 Spring Cloud Discovery và Eureka

Trong hệ thống, việc các dịch vụ có thể tìm thấy và giao tiếp với nhau một cách tự động là một yêu cầu quan trọng. Spring Cloud Discovery cùng với Eureka cung cấp một giải pháp hoàn chỉnh cho cơ chế Service Registry và Service Discovery, đảm bảo sự linh hoạt và khả năng mở rộng của hệ thống phân tán.

2.6.1 Eureka Server – Service Registry

Eureka Server hoạt động như một trung tâm đăng ký dịch vụ (Service Registry). Nó duy trì một danh sách động các service instance đang hoạt động trong hệ thống cùng với metadata liên quan (như địa chỉ, port, trạng thái...). Các dịch vụ khi khởi động sẽ gửi yêu cầu đăng ký (registration) đến Eureka Server và định kỳ gửi heartbeat để thông báo rằng chúng vẫn còn hoạt động.

Nhờ đó, hệ thống có thể:

- **Xóa bỏ cấu hình tĩnh**: Không cần khai báo cứng địa chỉ IP hay endpoint của từng service.
- **Hỗ trợ mở rộng tự động**: Khi có service instance mới xuất hiện hoặc bị hỏng, registry sẽ phản ánh ngay lập tức.
- **Tăng tính linh hoạt**: Cho phép dịch chuyển container, scaling theo tải mà không ảnh hưởng tới cấu trúc hệ thống.

2.6.2 Eureka Client – Service Discovery

Các dịch vụ trong hệ thống đóng vai trò là Eureka Client, nghĩa là chúng sẽ:

- **Tự động đăng ký** với Eureka Server khi khởi động.
- **Lấy danh sách các service khác** từ registry để biết cách giao tiếp mà không cần cấu hình thêm.
- **Cập nhật danh sách instance theo thời gian thực**, bảo đảm rằng mỗi yêu cầu được định tuyến đến service đang khả dụng.

Cơ chế Service Discovery giúp các dịch vụ giao tiếp theo kiểu "tên dịch vụ" thay vì địa chỉ cố định, tạo ra:

- **Khả năng chịu lỗi cao**: Khi một instance ngừng hoạt động, client sẽ tự động chuyển sang instance khác.
- **Tự động cân bằng tải**: Kết hợp với các công cụ như Spring Cloud LoadBalancer hoặc Gateway.
- **Tối ưu hoá vận hành trong môi trường container**: Các địa chỉ container thay đổi liên tục nhưng không ảnh hưởng đến toàn hệ thống.

2.7 Giao tiếp giữa các Service: REST API trên mạng nội bộ ECS

REST API (Representational State Transfer Application Programming Interface) là phương thức giao tiếp phổ biến nhờ tính đơn giản, linh hoạt và khả năng mở rộng. REST tuân theo các nguyên tắc của kiến trúc hướng tài nguyên, trong đó mỗi tài nguyên được biểu diễn bằng một định danh duy nhất (URI) và được thao tác thông qua các phương thức chuẩn của giao thức HTTP như `GET`, `POST`, `PUT` và `DELETE`. Cách tiếp cận này giúp các dịch vụ duy trì tính độc lập, giảm sự phụ thuộc lẫn nhau và tạo điều kiện thuận lợi cho việc phát triển và triển khai riêng biệt từng thành phần.

Trong hệ thống, các dịch vụ giao tiếp với nhau thông qua REST API truyền qua mạng nội bộ của ECS Cluster. Giao tiếp nội bộ mang lại nhiều lợi ích: (1) tăng cường bảo mật do không cần công khai endpoint ra Internet, (2) giảm thiểu độ trễ vì dữ liệu truyền trong hạ tầng mạng tối ưu hoá của cụm container, và (3) đảm bảo tính nhất quán trong cách các dịch vụ trao đổi dữ liệu. Việc sử dụng REST trong môi trường tách biệt cũng giúp việc giám sát, kiểm thử và mở rộng dịch vụ trở nên dễ dàng hơn, đồng thời cho phép từng dịch vụ có thể phát triển bằng công nghệ hoặc ngôn ngữ lập trình khác nhau mà không ảnh hưởng đến tổng thể hệ thống.

2.8 AWS Amplify

AWS Amplify là một nền tảng phát triển ứng dụng web và di động được xây dựng nhằm đơn giản hóa quá trình triển khai, quản lý và vận hành frontend trong môi trường đám mây. Amplify cung cấp một tập hợp các công cụ và dịch vụ giúp tự động hóa toàn bộ vòng đời phát triển của ứng dụng, từ xây dựng (build), kiểm thử (test), đến triển khai (deploy) và phân phối (hosting).

Một trong những điểm mạnh của Amplify là khả năng hỗ trợ CI/CD tích hợp, cho phép ứng dụng tự động được build và triển khai mỗi khi có thay đổi mới trên các nhánh mã nguồn. Điều này giảm thiểu sai sót thủ công, đồng thời đảm bảo rằng bản phát hành luôn nhất quán với mã nguồn mới nhất.

Ngoài ra, Amplify cung cấp hạ tầng Hosting tĩnh với CDN phân tán toàn cầu, giúp tối ưu hóa tốc độ tải trang cho người dùng ở nhiều khu vực địa lý khác nhau. Bằng việc sử dụng mạng phân phối nội dung (Content Delivery Network), các tệp frontend có thể được truy xuất nhanh chóng, giảm độ trễ và cải thiện trải nghiệm người dùng.

Về mặt bảo mật và vận hành, Amplify hỗ trợ cấu hình dễ dàng các domain tùy chỉnh, chứng chỉ HTTPS, ghi log, theo dõi hiệu năng và khả năng rollback về phiên bản trước khi gặp lỗi. Nhờ sự tích hợp chặt chẽ với các dịch vụ AWS khác, Amplify mang đến môi trường triển khai mạnh mẽ, linh hoạt và phù hợp cho các ứng dụng web hiện đại.

Hình 6: *AWS Amplify*

3 Yêu cầu về hệ thống

3.1 Xác định người dùng

Do hệ thống sẽ được sử dụng trong môi trường đại học, các phân loại người dùng cũng sẽ dựa trên môi trường thực tế. Điều này đảm bảo hệ thống được vận hành trơn tru khi áp dụng vào một trường đại học bất kỳ, cũng như không phải thuê thêm người điều hành riêng. Phân loại người dùng bao gồm:

- Admin: Người phụ trách vận hành, giám sát hệ thống sau khi triển khai. Nhiệm vụ của người dùng này còn bao gồm cài đặt, điều chỉnh cách vận hành của một số tính năng. Admin cũng có thể là một giảng viên trong trường.
- Trưởng khoa: Người phụ trách quản lý một số môn học của một khoa. Vai trò chủ yếu của người dùng này là kiểm duyệt các yêu cầu về quyền hạn người dùng, nội dung của các khóa học.
- Giảng viên: Là người phụ trách đăng tải tài liệu, xây dựng bài học, quiz,... Giảng viên cũng là người sẽ trả lời các câu hỏi trên diễn đàn của sinh viên, theo dõi tiến độ học tập,...
- Trợ giảng (TA): Vai trò chủ yếu của người dùng này là trợ giúp các giảng viên về các vấn đề liên quan tới một khóa học, như đề xuất chỉnh sửa nội dung, thảo luận trên diễn đàn, theo dõi tiến độ sinh viên,...
- Sinh viên: Là người dùng trực tiếp sử dụng các tài nguyên được đăng tải trên hệ thống. Sinh viên có quyền được truy cập các khóa học, xem nội dung khóa học, làm bài tập,...

3.2 Yêu cầu chức năng

Biểu diễn các yêu cầu chức năng dưới dạng User story, ta có bảng sau:

Mã	Ai	Muốn gì	Để làm gì
A-US01	Admin	Quản lý người dùng (tạo, xóa, sửa)	Đảm bảo hệ thống vận hành ổn định và đúng đối tượng sử dụng
A-US02	Admin	Quản lý quyền (thêm, xóa)	Đảm bảo hệ thống vận hành ổn định và đúng đối tượng sử dụng
A-US03	Admin	Cấu hình các tham số hệ thống	Tinh chỉnh hoạt động hệ thống theo yêu cầu thực tế
A-US04	Admin	Theo dõi hoạt động hệ thống (logs, thống kê)	Giám sát, phát hiện lỗi và tối ưu hiệu năng
D-US01	Trưởng khoa	Xác thực yêu cầu cấp quyền cho giảng viên hoặc trợ giảng	Đảm bảo đúng người đúng vai trò trong khoa
D-US02	Trưởng khoa	Kiểm duyệt nội dung các khóa học của khoa	Đảm bảo chất lượng và tính phù hợp chuyên môn
L-US01	Giảng viên	Tạo và quản lý nội dung khóa học (bài giảng, bài tập, quiz)	Xây dựng tài liệu giúp sinh viên học tập hiệu quả
L-US02	Giảng viên	Theo dõi tiến độ học tập của sinh viên	Đánh giá và hỗ trợ học tập kịp thời
L-US03	Giảng viên	Trả lời câu hỏi trên diễn đàn khóa học	Giải đáp thắc mắc cho sinh viên
T-US01	Trợ giảng	Hỗ trợ giảng viên chỉnh sửa hoặc đề xuất nội dung khóa học	Giảm tải công việc và nâng cao chất lượng khóa học
T-US02	Trợ giảng	Tham gia thảo luận trên diễn đàn	Hỗ trợ giải đáp câu hỏi của sinh viên
T-US03	Trợ giảng	Theo dõi tiến độ sinh viên	Hỗ trợ giảng viên trong đánh giá và nhắc nhở sinh viên
S-US01	Sinh viên	Truy cập và xem nội dung khóa học	Học tập theo chương trình giảng viên cung cấp
S-US02	Sinh viên	Làm bài tập, quiz và xem kết quả	Đánh giá mức độ hiểu bài và cải thiện kiến thức
S-US03	Sinh viên	Đặt câu hỏi và thảo luận trên diễn đàn	Trao đổi với giảng viên, trợ giảng và bạn học
S-US04	Sinh viên	Theo dõi tiến độ học tập của bản thân	Quản lý quá trình học tập và hoàn thành yêu cầu khóa học

Bảng 1: *Bảng User Stories*

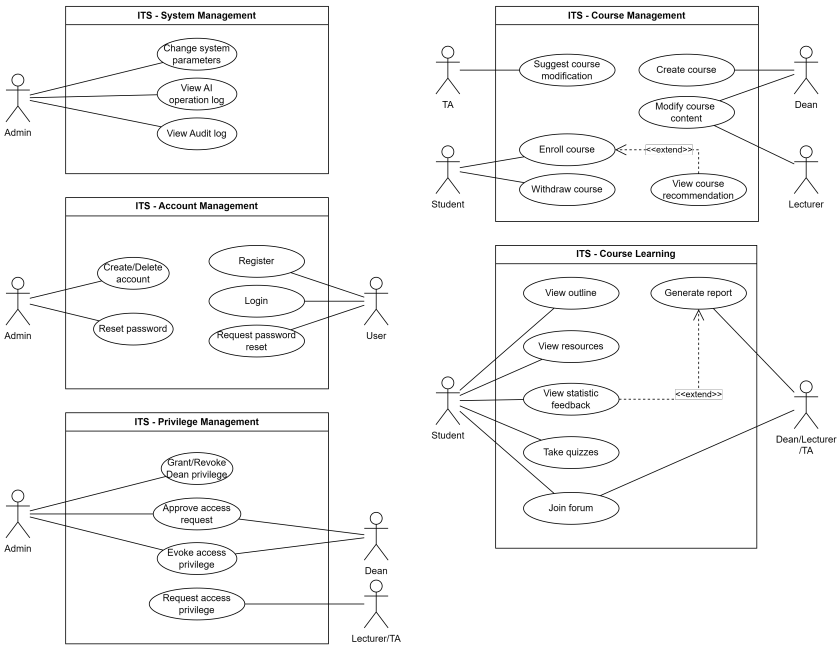
3.3 Yêu cầu phi chức năng

Từ kết quả thăm dò ý kiến người dùng, kết hợp brainstorm trong các buổi họp nhóm, các yêu cầu phi chức năng của hệ thống được miêu tả như sau:

Mã NFR	Mô tả yêu cầu phi chức năng
NFR01	Hệ thống phải cho phép thay đổi hành vi vận hành mà không cần chỉnh sửa mã nguồn, bao gồm bật/tắt tính năng và điều chỉnh tham số hoạt động.
NFR02	Hệ thống phải hỗ trợ thay đổi mô hình AI mà không gây gián đoạn dịch vụ (zero-downtime model update).
NFR03	Hệ thống phải cho phép cấu hình loại tài liệu được phép đăng tải trong khóa học mà không cần triển khai lại.
NFR04	Hệ thống phải cho phép chỉnh sửa layout hoặc cấu trúc giao diện khóa học thông qua cơ chế cấu hình.
NFR05	Module tạo báo cáo phải hỗ trợ cấu hình layout của báo cáo.
NFR06	Module tạo báo cáo phải cho phép cấu hình phạm vi dữ liệu được sử dụng trong báo cáo.
NFR07	Module tạo quiz bằng AI phải hỗ trợ cấu hình số lượng câu hỏi, độ khó và phân bố độ khó.
NFR08	Module gợi ý hint phải cho phép cấu hình tần suất và mức độ chi tiết của gợi ý.
NFR09	Module gợi ý chủ đề phải cho phép thêm hoặc loại bỏ chủ đề quan tâm thông qua cấu hình.
NFR10	Hệ thống phải đảm bảo mức độ sẵn sàng không dưới 99% trong thời gian có lưu lượng truy cập cao.
NFR11	Khi xảy ra sự cố downtime, hệ thống phải tự phục hồi trong thời gian không vượt quá 5 phút.
NFR12	Các thành phần quan trọng phải được triển khai theo mô hình nhiều vùng sẵn sàng (multi-AZ) hoặc tương đương.
NFR13	Dữ liệu bài giảng đăng tải phải được kiểm tra tính hợp lệ (định dạng, nội dung đầy đủ).
NFR14	Mọi dữ liệu tải lên phải được lưu trữ an toàn, bao gồm mã hóa khi lưu trữ và khi truyền tải.
NFR15	Mọi dữ liệu hiển thị phải đảm bảo nhất quán giữa các người dùng và các phiên xử lý.
NFR16	Các thao tác quan trọng phải đảm bảo tính toàn vẹn giao dịch (transactional integrity).
NFR17	Lịch sử chỉnh sửa nội dung khóa học phải được ghi lại để phục vụ truy vết.
NFR18	Các API liên quan đến AI phải có thời gian phản hồi trung bình dưới 2 giây.
NFR19	Thời gian tải nội dung khóa học không vượt quá 1 giây trên kết nối mạng phổ thông.
NFR20	Hệ thống phải sử dụng cơ chế caching để cải thiện tốc độ truy xuất dữ liệu.
NFR21	Hệ thống phải hỗ trợ ít nhất X sinh viên truy cập đồng thời (giá trị X được xác định sau khi ước tính tải).
NFR22	Hệ thống phải tránh xung đột dữ liệu (race condition) khi nhiều người dùng thao tác song song.
NFR23	Hệ thống phải tự động mở rộng tài nguyên khi tải tăng cao, đặc biệt trong mùa thi.
NFR24	Hệ thống phải tự động thu hẹp tài nguyên khi tải giảm để tối ưu chi phí vận hành.
NFR25	Các dịch vụ AI và quiz phải hỗ trợ autoscaling dựa trên CPU/GPU hoặc số lượng yêu cầu mỗi giây.

Bảng 2: *Danh sách yêu cầu phi chức năng (NFR)*

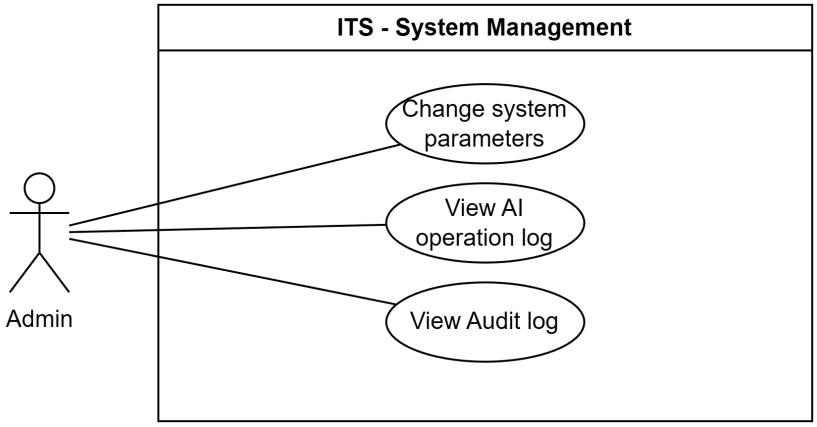
4 Đặc tả chi tiết các use-case



Hình 7: *Tổng quan các Use-case*

4.1 Quản lý tài khoản

4.1.1 Tổng quan



Hình 8: *Các use-case về Quản lý hệ thống*

4.1.2 Đặc tả chi tiết

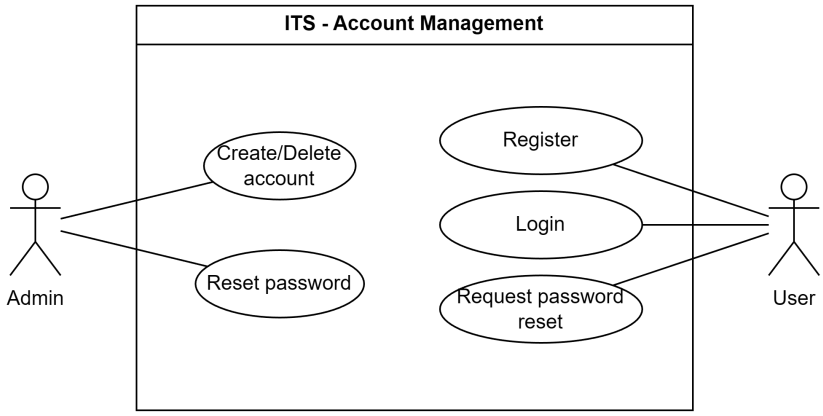
Bảng 3: *Usecase «Tải file lên hệ thống»*

Usecase	**Tải file lên hệ thống**
Use-case ID	ST.02
Tổng quan	Sinh viên tải file cần in lên hệ thống trước khi thực hiện thao tác in.
Actor	Sinh viên.
Tiền điều kiện	• Hệ thống SSPS đang hoạt động. • Sinh viên đã đăng nhập.

Usecase	Tải file lên hệ thống
Điều kiện kích hoạt	Sinh viên ấn vào nút "Tải lên file".
Các bước	1. Hệ thống hiển thị giao diện chọn file từ máy sinh viên. 2. Sinh viên chọn file và ấn Chọn. 3. Hệ thống kiểm tra loại file hợp lệ. 4. Hệ thống lưu file.
Hậu điều kiện	• File sinh viên chọn được tải lên hệ thống. • Hệ thống lưu trữ file thành công.
Xử lý ngoại lệ	ex1. Bước 3 - Nếu file được chọn không thuộc một trong các định dạng đã được SPSO chỉ định: (a) Hệ thống hiển thị thông báo lỗi "Sai định dạng file". (b) Hệ thống hiển thị danh sách các loại file được phép tải lên. (c) Sinh viên chọn hủy thao tác. (d) Use-case trở lại bước 2. ex2. Bước 4 - Hệ thống không thể lưu trữ file do các lỗi hệ thống như thiếu bộ nhớ, hoặc đường truyền không ổn định gây lỗi file tải lên không nguyên vẹn: (a) Hệ thống hiển thị thông báo lỗi "Không thể tải lên file, vui lòng thử lại sau". (b) Sinh viên chọn hủy thao tác. (c) Use-case kết thúc.

4.2 Quản lý quyền chỉnh sửa khóa học

4.2.1 Tổng quan

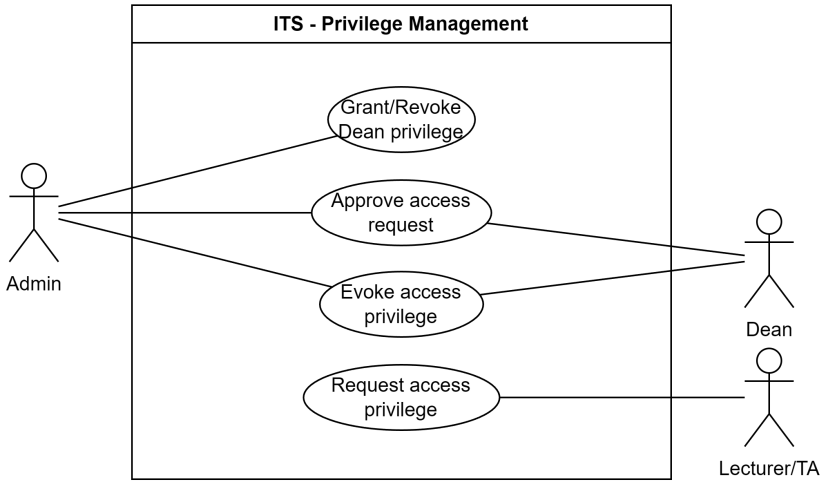


Hình 9: *Các use-case về Quản lý tài khoản*

4.2.2 Đặc tả chi tiết

4.3 Quản lý hệ thống

4.3.1 Tổng quan

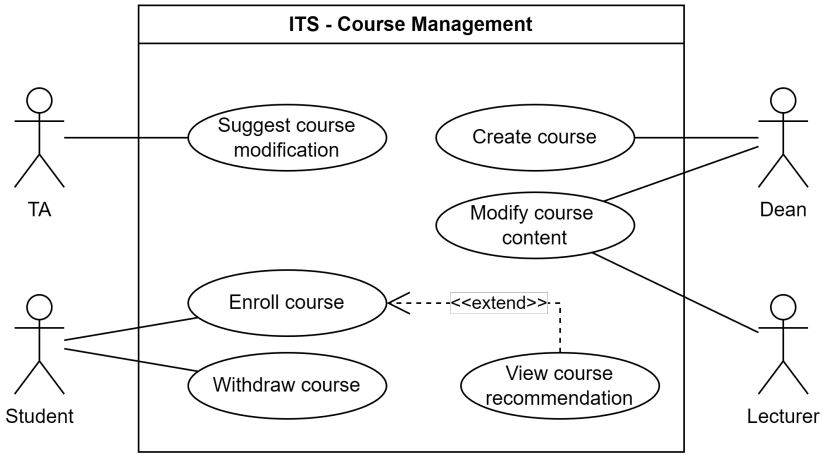


Hình 10: *Các use-case về Quản lý quyền truy cập*

4.3.2 Đặc tả chi tiết

4.4 Quản lý khóa học

4.4.1 Tổng quan

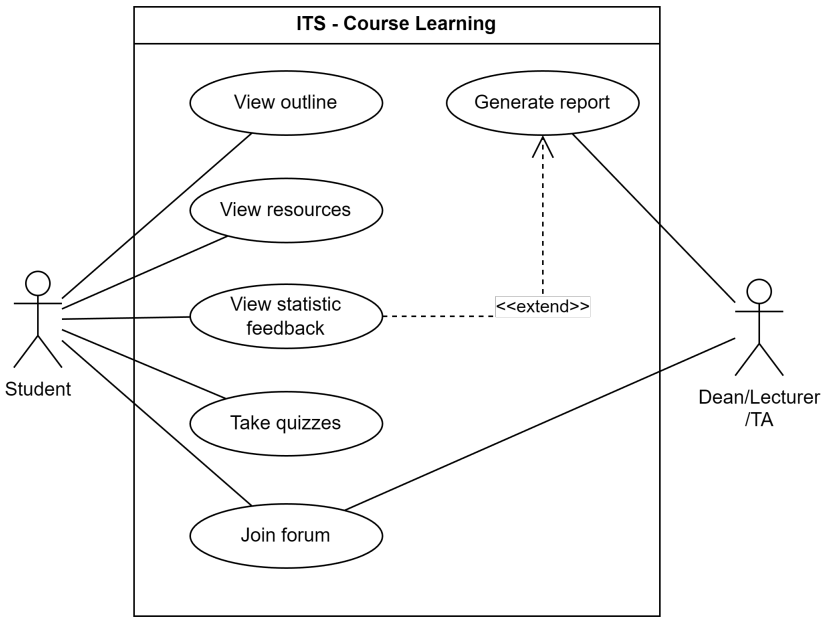


Hình 11: *Các use-case về Quản lý khóa học*

4.4.2 Đặc tả chi tiết

4.5 Học khóa học

4.5.1 Tổng quan



Hình 12: *Các use-case về Học khóa học*

4.5.2 Đặc tả chi tiết

5 Phân tích & Thiết kế hệ thống

5.1 Entity Relationship Diagram (ERD)

5.2 Kiến trúc hệ thống

6 Hiện thực

7 Triển khai

8 Tổng kết

8.1 Nhận xét

8.2 Hướng phát triển

9 Tài liệu tham khảo

10 Phụ lục