RiCon

RiCourse
Software Architecture Document

Version 1.5

Revision History

Date	Version	Description	Author
07/12/2024	1.0	Khởi tạo, Introduction, Architectural Goals and Constraints	Lê Đình Hoàng Vũ
12/12/2024	1.1	Use-Case Model	Nguyễn Anh Hào
13/12/2024	1.2	Logical View	Nguyễn Anh Hào, Lê Anh Khôi, Nguyễn Trung Kiên - 169, Nguyễn Trung Kiên - 170
28/12/2024	1.4	Deployment	Nguyễn Trung Kiên - 170 Lê Đình Hoàng Vũ
28/12/2024	1.4	Implement View Front-end	Nguyễn Trung Kiên - 169
28/12/2024	1.5	Implement View Back-end	Nguyễn Trung Kiên - 170

Table of Contents

Software Architecture Document

1. Introduction

1.1 Mục đích:

Tài liệu Kiến trúc Phần mềm này cung cấp cái nhìn tổng quan về kiến trúc của Hệ thống Quản lý khóa học RiCourse. Mục đích của nó là nắm bắt và truyền đạt các quyết định kiến trúc quan trọng đã được đưa ra trên hệ thống.

1.2 Phạm vi:

Tài liệu Kiến trúc Phần mềm này áp dụng cho Hệ thống Quản lý Khóa học RiCourse, được phát triển bởi RiCon. Nó mô tả kiến trúc cấp cao của toàn bộ hệ thống, bao gồm các thành phần chính, mối quan hệ của chúng và các nguyên tắc và hướng dẫn chi phối thiết kế và sự phát triển của chúng.

1.3 Tài liệu tham khảo:

- Software Architecture Document. (2001). California State University, Northridge. Truy cập ngày 1 tháng 8 năm 2024, từ https://www.ecs.csun.edu/~rlingard/COMP684/Example2SoftArch.htm
- Tuan Thanh HO. (2020, Ngày 28 tháng 11). PA3 PA4 [Video]. YouTube. Truy cập ngày 1 tháng 8 năm 2024, từ https://www.youtube.com/watch?v=pcM9xQiUt5g
- Slides được cung cấp bởi Cô Nguyễn Thị Minh Tuyền, Thầy Hồ Tuấn Thanh và Thầy Mai Anh Tuấn trong Khóa học Giới thiệu về Kỹ thuật Phần mềm.

1.4 Tổng quan:

Tài liệu Kiến trúc Phần mềm chứa các thông tin sau:

- Architectural Goals and Constraints: Liệt kê các yêu cầu và ràng buộc chính có tác động đáng kể đến kiến trúc.
- Use-Case Model: Mô tả các Use-Case và kịch bản quan trọng về mặt kiến trúc.
- Logical View: Mô tả các phần quan trọng về mặt kiến trúc của mô hình thiết kế.
- Deployment: Mô tả việc triển khai vật lý của hệ thống.
- Implementation View: Mô tả cấu trúc tổng thể của mô hình triển khai.

2. Architectural Goals and Constraints

2.1 Công nghệ:

- Front-end: Ứng dụng front-end của hệ thống được xây dựng bằng Next.js – một framework mạnh mẽ của React. RiCourse có sử dụng Material UI để tạo ra giao diện đẹp mắt, hiện đại, thân thiện với người dùng.
- Back-end: NestJS được chọn làm framework phát triển back-end, với cấu trúc dựa trên mô hình module. Ngoài ra, nhóm còn sử dụng Python để xử lý những vấn đề phức tạp hơn nhận diện người thi có gian lận hay không.
- Database: Nhóm sử dụng PostgreSQL và Prisma ORM giúp giảm thiểu việc viết SQL trực tiếp và dễ dàng tương tác với cơ sở dữ liệu qua các model.
- Môi trường lập trình: Visual Studio Code.
- Môi trường ứng dụng: Website.

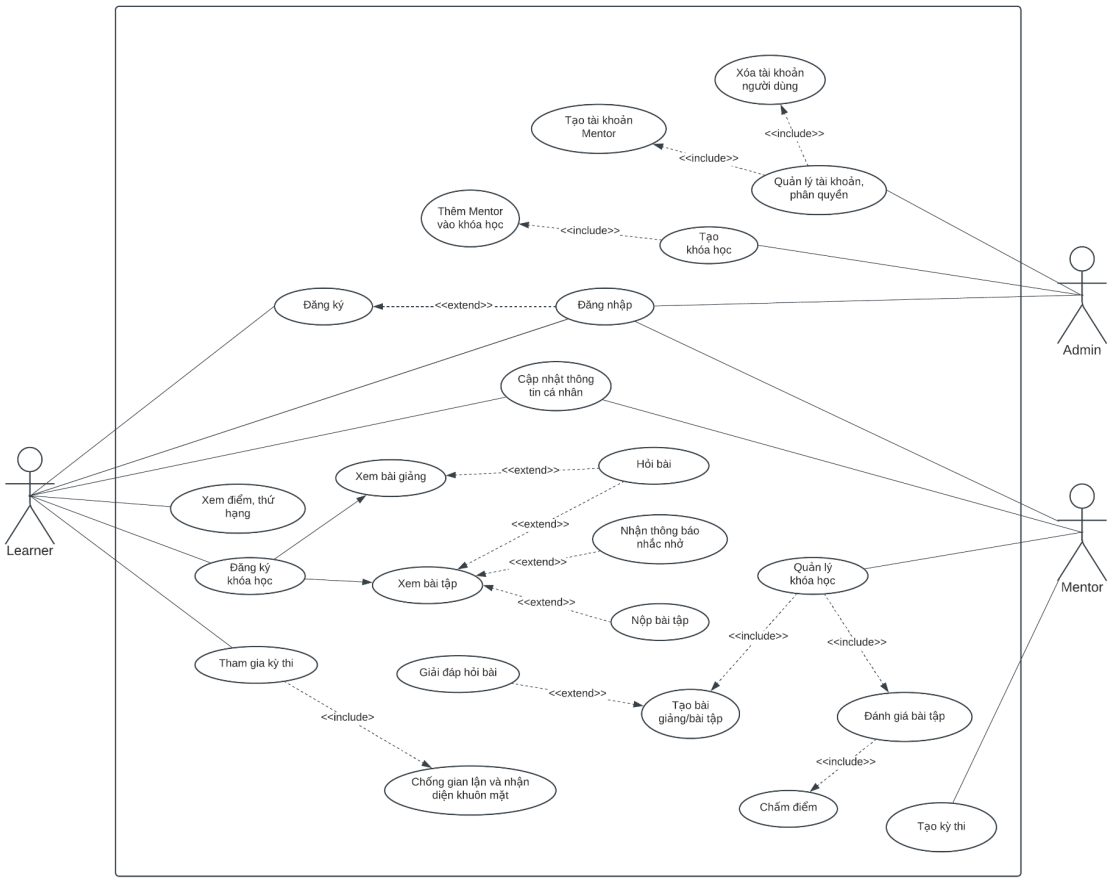
2.2 Yêu cầu chung:

- Yêu cầu về thiết bị: Người dùng cần sử dụng một thiết bị có trình duyệt web hiện đại, chẳng hạn như Google Chrome, Microsoft Edge, Safari, hoặc các trình duyệt phổ biến khác. Thiết bị cần có khả năng kết nối Internet để truy cập vào hệ thống.
- Yêu cầu về hiệu suất:
    - Thời gian tải ban đầu: Hệ thống phải đảm bảo rằng thời gian tải trang đầu tiên khi người dùng truy cập vào website không vượt quá 10 giây.
    - Phản hồi yêu cầu: Mọi yêu cầu của người dùng cần được xử lý nhanh chóng, và hệ thống sẽ trả về kết quả trong vòng dưới 3 giây.
    - Khả năng chịu tải: Hệ thống cần đảm bảo rằng có thể xử lý ít nhất 300 người dùng đồng thời mà không bị giảm hiệu suất.
    - Quản lý tài nguyên và tối ưu hóa backend: Để duy trì hiệu suất ổn định, hệ thống sẽ được tối ưu hóa để xử lý nhanh chóng các truy vấn đến cơ sở dữ liệu và tính toán, giảm thiểu tối đa các thao tác không cần thiết.

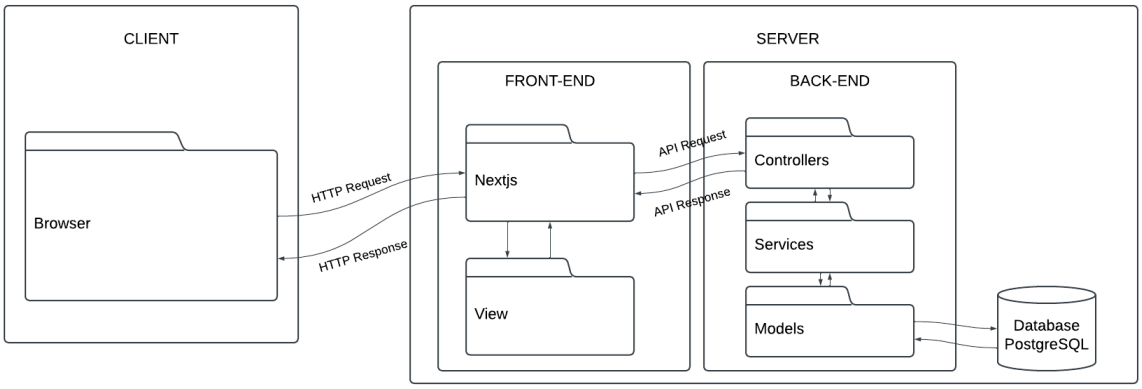
2.3 Sự phụ thuộc, Ràng buộc và Bảo mật:

- Ràng buộc:
    - Tên người dùng: Phải dài ít nhất 8 ký tự.
    - Mật khẩu: Phải dài ít nhất 8 ký tự và bao gồm ít nhất 1 chữ cái, 1 số và 1 ký tự đặc biệt.
    - Định danh tài khoản: Tài khoản sẽ được xác định bằng tên người dùng.
    - Quyền truy cập của khách: Nếu bạn không đăng ký tài khoản, bạn vẫn có thể xem trang web với tư cách khách.
    - Ngôn ngữ mặc định của nền tảng của chúng tôi sẽ là tiếng Việt.
- Bảo mật:
    - Xác thực và phân quyền người dùng: Hệ thống sử dụng JWT (JSON Web Token) để xác thực và duy trì phiên người dùng. Sau khi người dùng đăng nhập, một token sẽ được cấp phát và yêu cầu trong các yêu cầu tiếp theo để xác thực danh tính và bảo vệ dữ liệu.
    - Mã hóa thông tin nhạy cảm: Mọi mật khẩu của người dùng đều được mã hóa trước khi lưu trữ trong cơ sở dữ liệu, sử dụng các thuật toán mã hóa mạnh mẽ như bcrypt.
    - Chống các cuộc tấn công phổ biến: Hệ thống được bảo vệ khỏi các cuộc tấn công như SQL Injection, XSS (Cross-Site Scripting) và CSRF (Cross-Site Request Forgery).
    - Quản lý và giám sát liên tục: Hệ thống sẽ thực hiện giám sát liên tục đối với tất cả các hoạt động và sự kiện liên quan đến bảo mật thông qua Audit Log.
    - Đảm bảo tính bảo mật của API: Tất cả các API giao tiếp giữa front-end và back-end đều sẽ được bảo vệ bằng các biện pháp bảo mật mạnh mẽ, bao gồm xác thực API bằng token.

3. Use-Case Model



4. Logical View

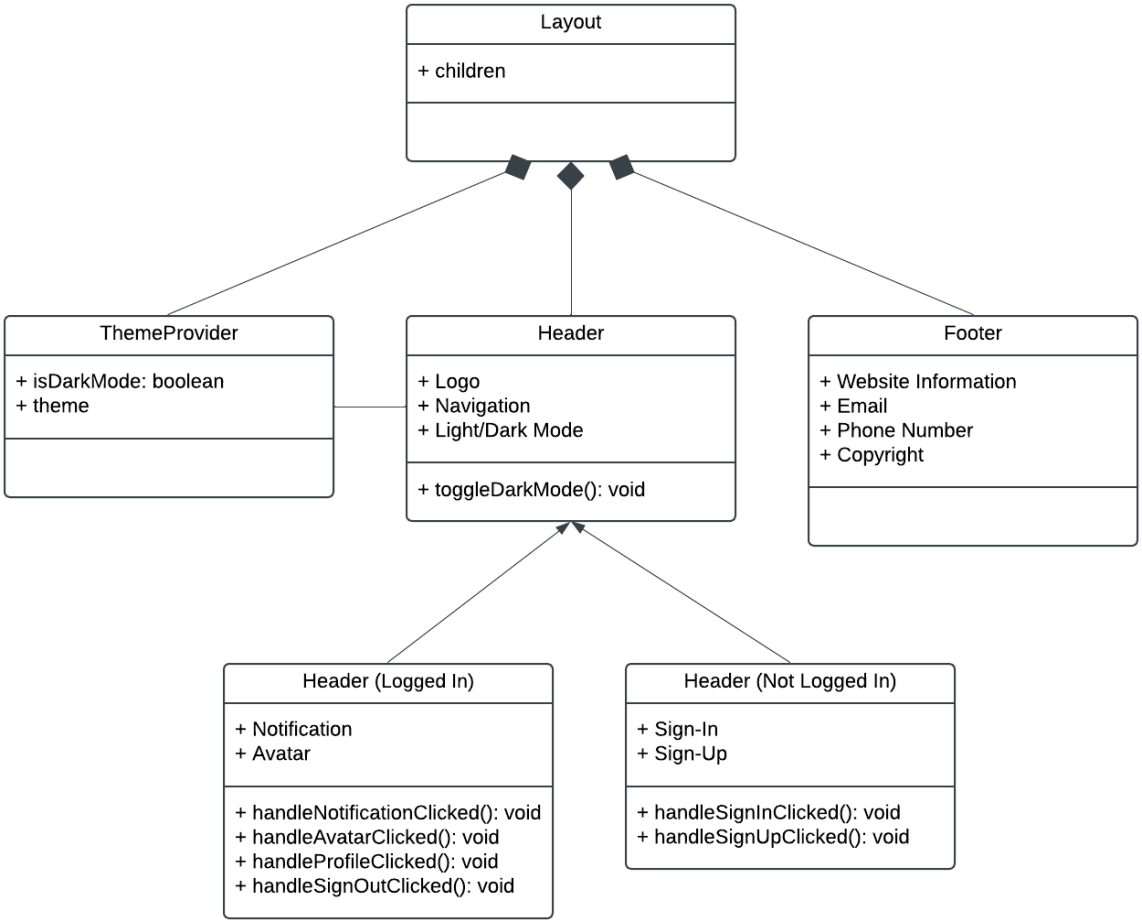


RiCourse sử dụng kiến trúc Client - Server. Ở phía Client là Browser của người dùng, giao tiếp với hệ thống thông qua giao thức HTTP. Ở phía Server, cụ thể sử dụng kiến trúc microservices, nghĩa là các phần của hệ thống phía server sẽ được phân tách các thành phần chức năng thành các dịch vụ độc lập, mỗi dịch vụ có thể sử dụng ngôn ngữ và công nghệ riêng. Trong hệ thống này, front-end được triển khai bằng framework Nextjs và thư

viện MUI, còn back-end được triển khai bằng NestJS, viết bằng ngôn ngữ lập trình Typescript, và cuối cùng là AI Services (một service của back-end) được viết bằng Python, đây đều là các dịch vụ riêng biệt, không phụ thuộc vào nhau, chúng được giao tiếp thông qua API.

4.1 Component: View

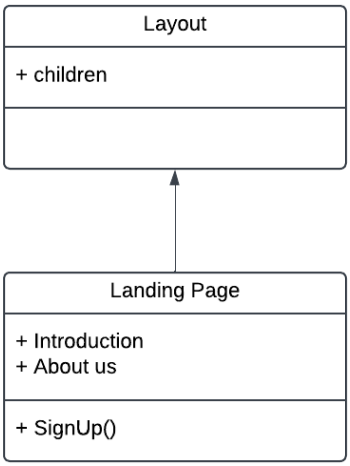
4.1.1 Layout



Mô tả: Trong front-end Next.js, layout là một cách để tái sử dụng các phần tử giao diện (UI) chung như header, footer,,... mà không cần phải viết lại mã cho mỗi trang. Layout giúp tổ chức và cấu trúc trang web theo một cách rõ ràng và dễ duy trì hơn. Layout thường được sử dụng để bọc nội dung của các trang cụ thể (children, ví dụ Auth Page, Admin Page,...) trong một cấu trúc cố định muốn chia sẻ trên toàn bộ ứng dụng.

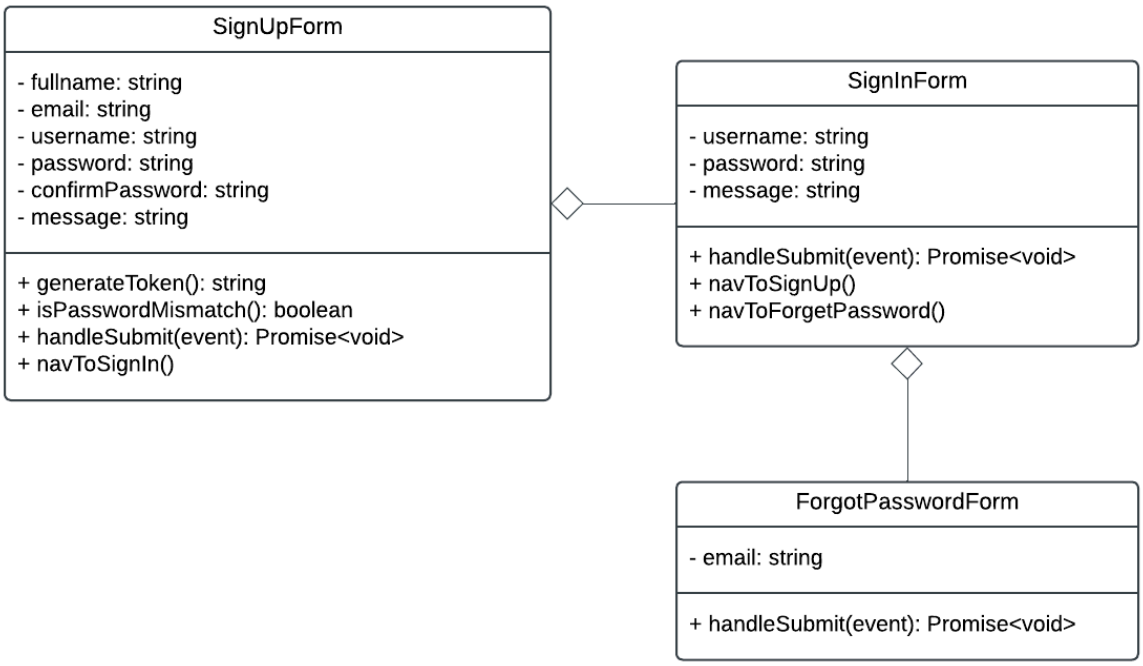
Với website RiCourse, Layout chung sẽ gồm có Header và Footer, trong đó Header sẽ có sự thay đổi một chút ở trước và sau khi người dùng đăng nhập. Ngoài ra, layout còn có thể được thay đổi được giao diện nhờ có ThemeProvider.

4.1.2 Home Page



Mô tả: Trang Home Page, là trang dạng một Landing Page mà người dùng sẽ thấy khi truy cập vào trang web. Trang này đóng vai trò như một điểm bắt đầu, cung cấp cái nhìn tổng quan về mục đích và tính năng của hệ thống. Trên trang chủ, người dùng sẽ tìm thấy các thông tin giới thiệu về nền tảng, các dịch vụ mà hệ thống cung cấp, cũng như các liên kết dễ dàng dẫn tới các trang chức năng khác như đăng nhập, đăng ký hoặc tìm hiểu thêm về các khóa học và các tính năng của hệ thống. Trang này sẽ được thiết kế đơn giản nhưng thu hút, với mục tiêu tạo ấn tượng tốt và dễ dàng dẫn dắt người dùng vào các hành động tiếp theo.

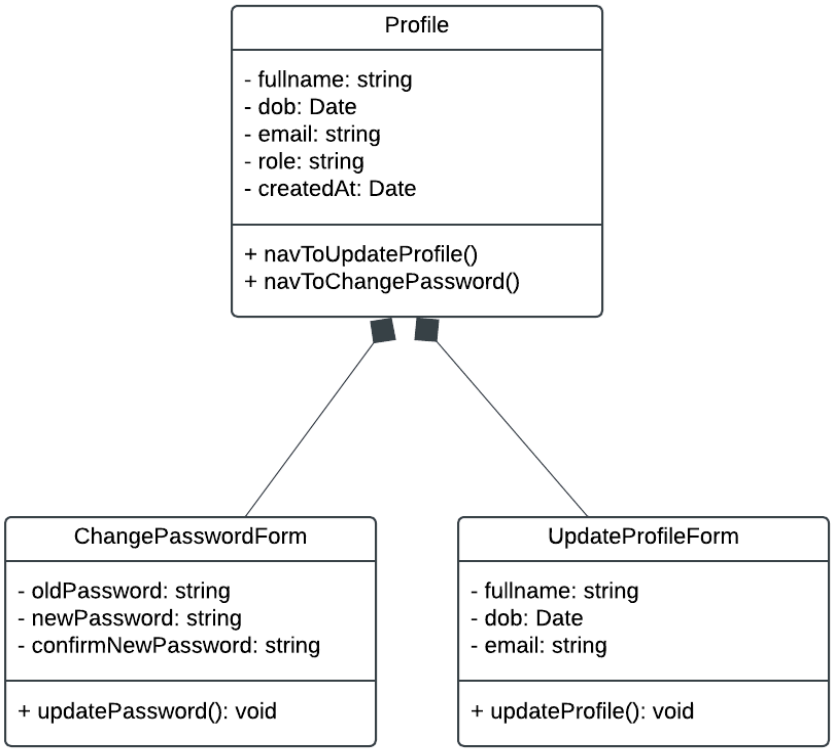
4.1.3 Authentication Page



Mô tả: Trang xác thực là nơi người dùng thực hiện các thao tác đăng nhập hoặc đăng ký tài khoản khi truy cập vào hệ thống lần đầu. Đây là bước đầu tiên mà người dùng sẽ phải thực hiện trước khi truy cập vào bất kỳ tính năng nào của hệ thống. Trang này yêu cầu người dùng nhập thông tin tài khoản của mình, bao gồm tên đăng nhập và

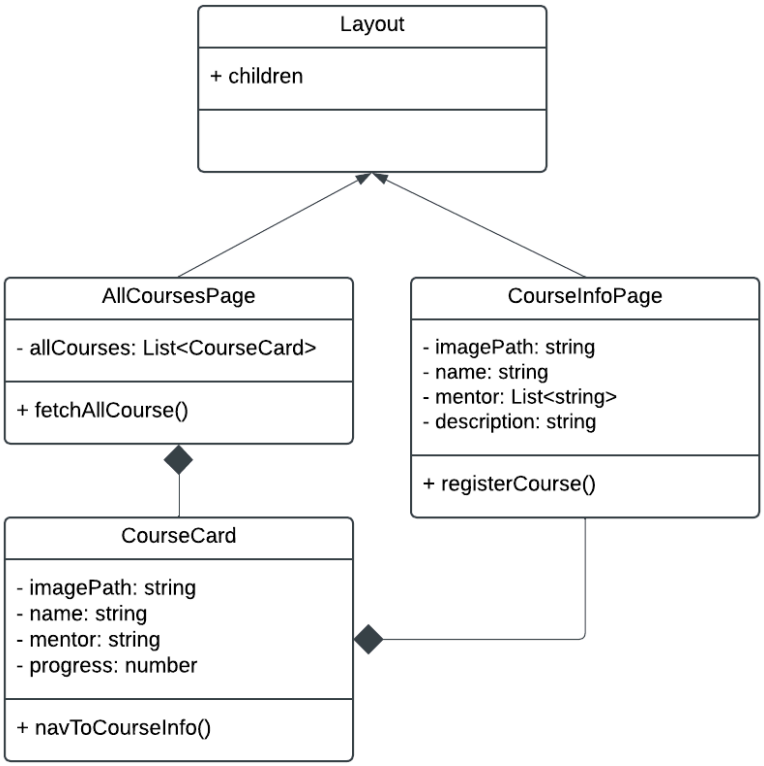
mật khẩu, để hệ thống có thể xác thực quyền truy cập. Nếu người dùng chưa có tài khoản, hệ thống cũng cung cấp tùy chọn đăng ký tài khoản mới. Quyền truy cập vào hệ thống sẽ được phân loại theo các vai trò người dùng, bao gồm Learner (Học viên), Mentor (Giảng viên), và Admin (Quản trị viên), với mỗi vai trò có những quyền hạn và chức năng khác nhau.

4.1.4 Profile Page



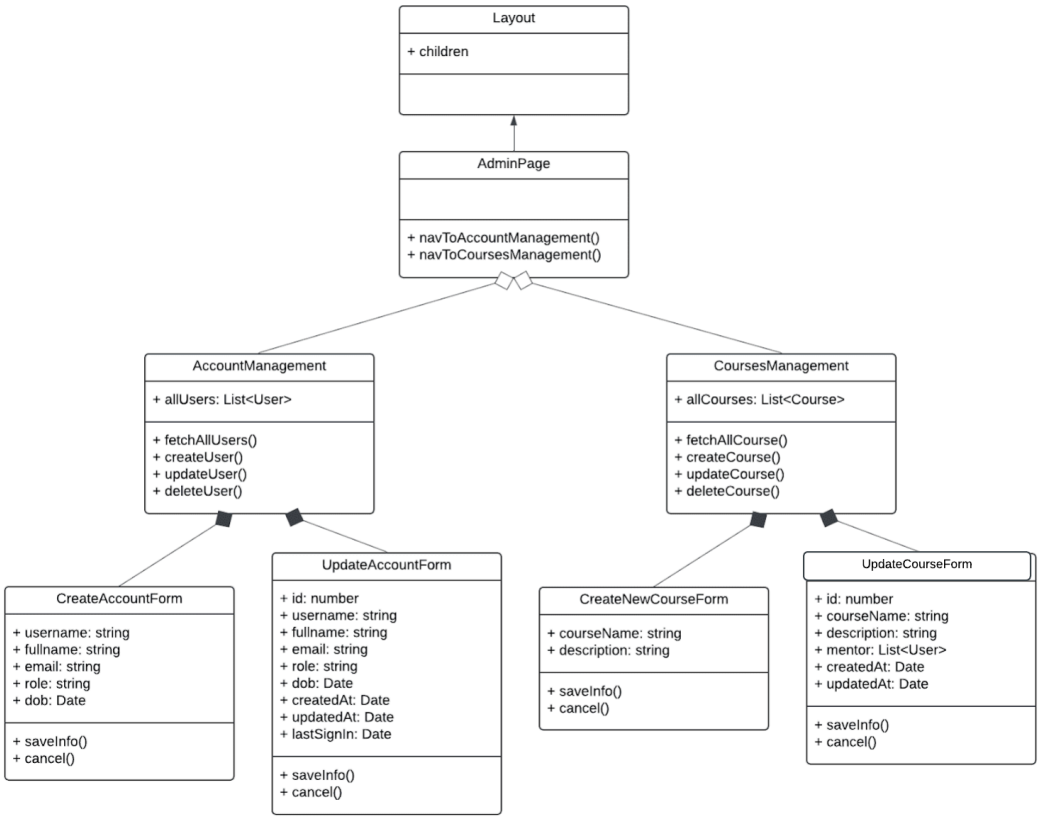
Mô tả: Trang thông tin người dùng là nơi người dùng có thể xem và quản lý các thông tin cá nhân của mình, bao gồm tên, email, số điện thoại và các thông tin liên quan khác. Người dùng cũng có thể cập nhật các thông tin cá nhân này nếu cần thiết. Trang này cung cấp các chức năng để người dùng thay đổi mật khẩu của mình, cũng như cấu hình các tùy chọn liên quan đến tài khoản, giúp đảm bảo tính bảo mật và cập nhật của thông tin tài khoản. Ngoài ra, trên trang này, người dùng có thể xem lịch sử hoạt động của mình, chẳng hạn như các khóa học đã tham gia, các kỳ thi đã hoàn thành, hoặc các hoạt động khác liên quan đến tài khoản.

4.1.5 Course Page



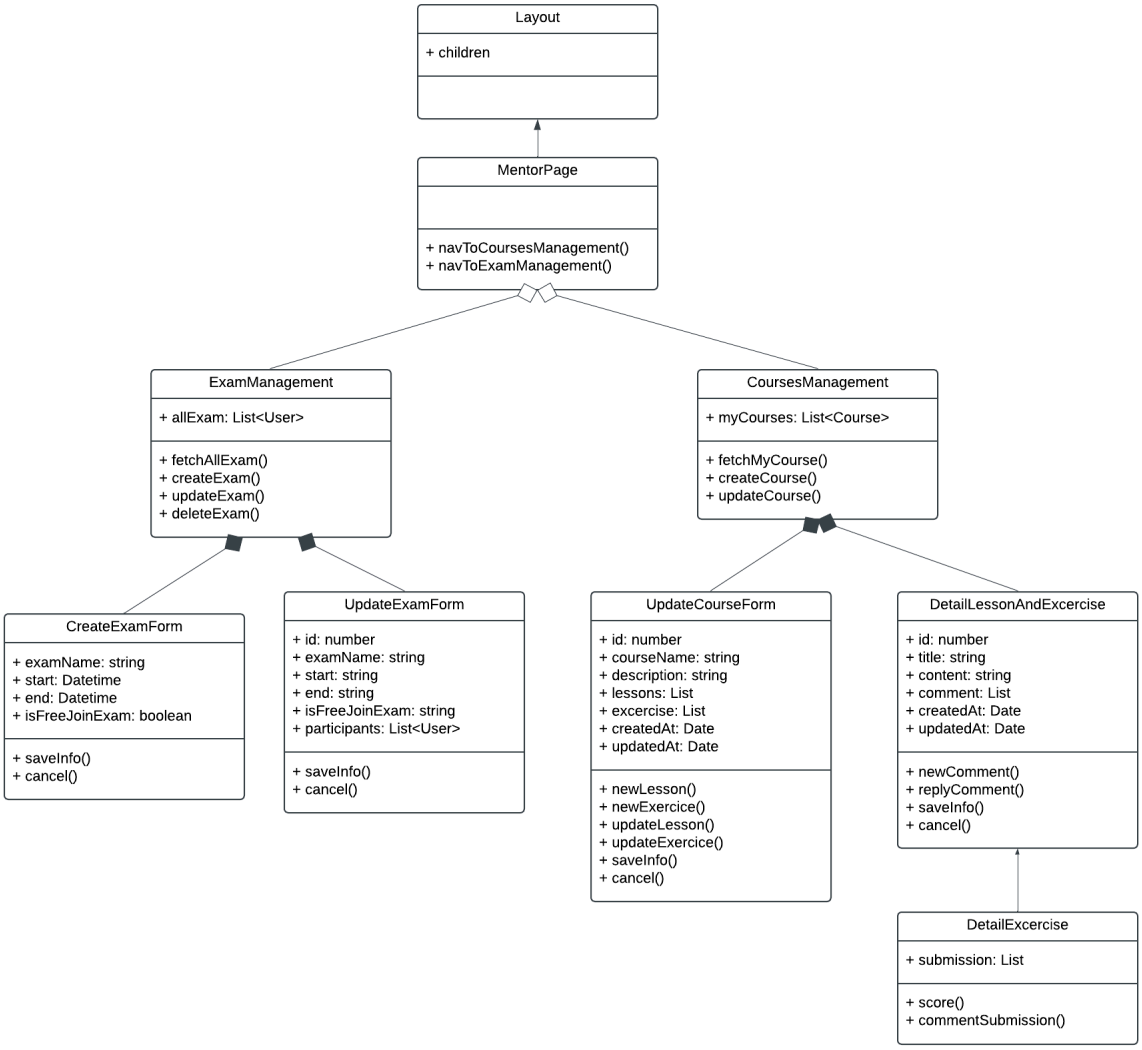
Mô tả: Trang khóa học là một phần quan trọng trong hệ thống của website, nơi người dùng có thể khám phá các khóa học chi tiết, từ các thông tin cơ bản đến các tài liệu học tập. Đây là nơi giúp người học (Learner) tìm hiểu và quyết định tham gia các khóa học dựa trên các yếu tố như mô tả tên khóa học, mô tả chi tiết về nội dung và mục tiêu của khóa học, các bài học cụ thể, cũng như thời gian dự kiến hoàn thành. Trang cũng cung cấp các chức năng đăng ký khóa học dễ dàng, cho phép Learner đăng ký tham gia khóa học ngay lập tức, hoặc thêm khóa học vào danh sách yêu thích nếu họ muốn quay lại sau này. Danh sách khóa học được thể hiện dưới dạng các Card để hiển thị hình ảnh trực quan, sinh động.

4.1.6 Admin Page



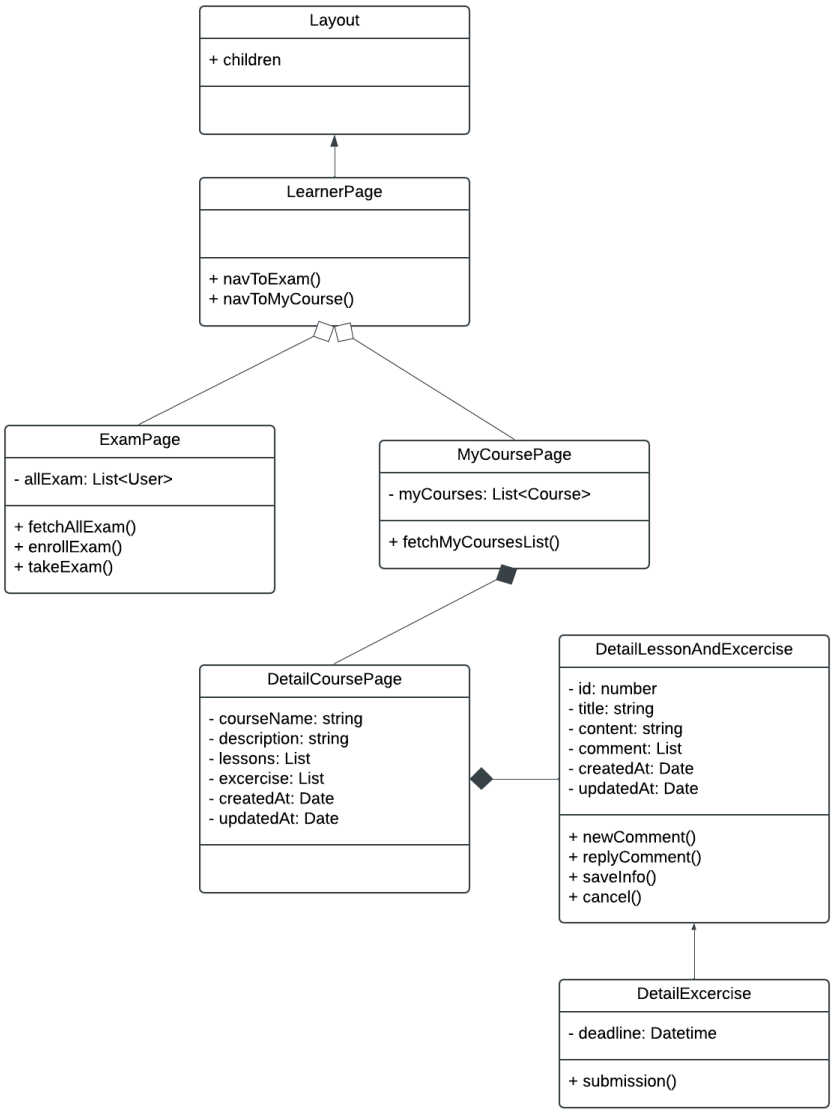
Mô tả: Trang quản trị dành cho Admin là nơi quản lý toàn bộ hệ thống. Đây là giao diện mà các quản trị viên sử dụng để theo dõi và điều phối mọi hoạt động của hệ thống. Admin có quyền quản lý người dùng, bao gồm các thao tác như xem thông tin người dùng, thay đổi quyền hạn của người dùng, cập nhật thông tin tài khoản người dùng, hoặc xóa người dùng khỏi hệ thống. Bên cạnh đó, Admin cũng có quyền quản lý các khóa học trên nền tảng, bao gồm việc xem danh sách các khóa học hiện có, chỉnh sửa thông tin của các khóa học, cấp quyền quản lý khóa học cho các Mentor, hoặc xóa các khóa học khỏi hệ thống nếu cần thiết. Trang này cung cấp các công cụ mạnh mẽ giúp quản lý hiệu quả người dùng và các nội dung khóa học trên nền tảng.

4.1.7 Mentor Page



Mô tả: Trang dành cho Mentor cung cấp các công cụ cần thiết để giảng viên quản lý và duy trì các khóa học của mình. Các Mentor có thể sử dụng trang này để chỉnh sửa nội dung bài giảng, thêm hoặc sửa bài tập cho học viên, quản lý các bài nộp của học viên và chấm điểm các bài tập. Ngoài ra, Mentor còn có thể trả lời các câu hỏi của học viên được đăng dưới các bài học hoặc bài tập của mình, giúp tạo ra môi trường học tập tương tác và hỗ trợ. Trang này giúp Mentor quản lý các hoạt động giảng dạy, theo dõi tiến độ học tập của học viên, và tạo ra trải nghiệm học tập chất lượng cao hơn cho người học.

4.1.8 Learner Page



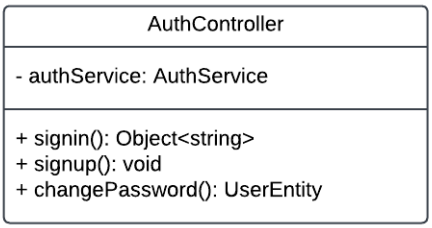
Mô tả: Các trang dành cho Learner (Học viên) cung cấp các tính năng giúp học viên tham gia vào các khóa học và kỳ thi một cách dễ dàng. Về khóa học, Learner có thể xem thông tin chi tiết về các khóa học hiện có trên nền tảng, đăng ký tham gia các khóa học, cũng như theo dõi danh sách các khóa học mà mình đã đăng ký. Ngoài ra, Learner cũng có thể đặt câu hỏi ngay dưới các bài học hoặc bài tập của mình nếu có bất kỳ thắc mắc nào. Về các kỳ thi, Learner có thể xem danh sách các kỳ thi mà mình đã đăng ký tham gia (hoặc được Mentor đăng ký cho), tham gia kỳ thi khi kỳ thi bắt đầu, cũng như kiểm tra kết quả của các kỳ thi đã hoàn thành. Tất cả các tính năng này giúp Learner dễ dàng tham gia vào quá trình học tập và kiểm tra của mình.

4.2 Component: Controller

Controller là một thành phần quan trọng trong kiến trúc backend của hệ thống, sử dụng NestJS để xử lý các yêu cầu HTTP đến từ client và trả về phản hồi cho người dùng. Controller có vai trò chính trong việc nhận các yêu cầu từ frontend (NextJS), sau đó gọi các Services (xem mục 4.3) tương ứng để xử lý dữ liệu, thực hiện các thao tác

nghiệp vụ, và cuối cùng trả về kết quả dưới dạng JSON hoặc các định dạng dữ liệu khác. Mỗi controller trong NestJS sẽ gắn với các route cụ thể và các phương thức HTTP (GET, POST, PUT, DELETE, v.v.) để xử lý các yêu cầu từ client. Controller không tự mình xử lý các logic nghiệp vụ mà sẽ ủy quyền cho các Service, giúp việc duy trì mã nguồn dễ dàng và sạch sẽ hơn. Controller sẽ nhận dữ liệu đầu vào, chuyển đến Service để xử lý và trả lại kết quả cho người dùng qua API, giúp frontend có thể dễ dàng truy xuất và sử dụng dữ liệu đó.

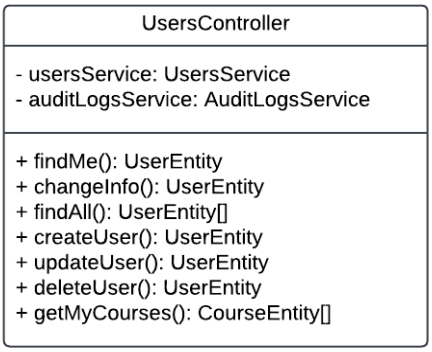
4.2.1 Auth Controller



Mục đích: Xử lý những thao tác liên quan đến việc đăng nhập, đăng ký, đổi mật khẩu.
Các phương thức:

- signin(): đăng nhập
- signup(): đăng ký
- changePassword(): đổi mật khẩu

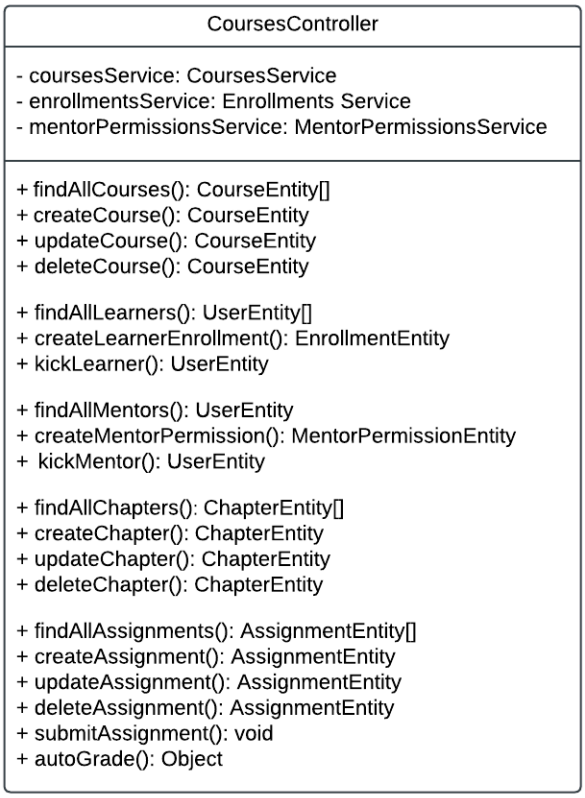
4.2.2 Users Controller



Mục đích: Xử lý những thao tác liên quan đến người dùng
Các phương thức:

- findMe(): tìm thông tin của người dùng hiện tại
- changeInfo(): thay đổi thông tin của người dùng hiện tại
- getMyCourses(): tìm thông tin về các khóa học của người dùng hiện tại
- findAll(): tìm thông tin của tất cả người dùng trong hệ thống (có thể đi kèm với vài điều kiện tìm kiếm)
- createUser(): tạo ra người dùng mới với các thông tin cơ bản (chỉ Admin)
- updateUser(): cập nhật thông tin của một người dùng
- deleteUser(): xóa một người dùng

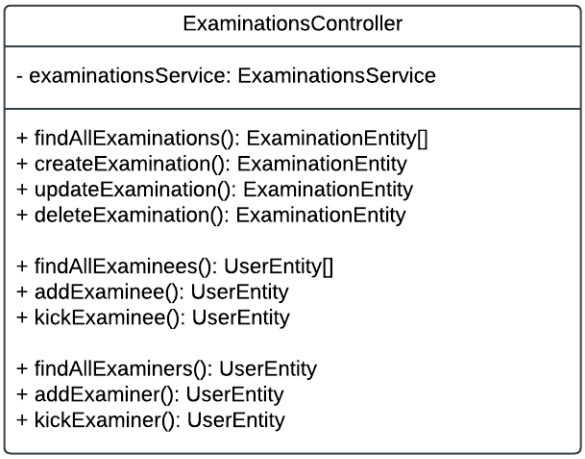
4.2.3 Courses Controller



Mục đích: Xử lý những thao tác liên quan đến các khóa học
Các phương thức:

- findAllCourses(): tìm thông tin của tất cả các khóa học trên hệ thống
- createCourse(): tạo một khóa học (chỉ Admin)
- updateCourse(): cập nhật thông tin một khóa học
- deleteCourse(): xóa khóa học
- findAllLearners(): tìm thông tin của tất cả các học viên trong một khóa học
- createLearnerEnrollment(): cho một học viên tham gia (được quyền truy cập vào khóa học)
- kickLearner(): kết thúc quyền được truy cập một khóa học của một học viên
- findAllMentors(): tìm thông tin của tất cả các người hướng dẫn trong một khóa học
- createMentorPermission(): cho một người hướng dẫn được tham gia giảng dạy trong khóa học
- kickMentor(): kết thúc quyền được giảng dạy của người hướng dẫn trong một khóa học
- findAllChapters(): tìm thông tin của tất cả các chương trong một khóa học
- createChapter(): tạo chương mới trong một khóa học
- updateChapter(): cập nhật thông tin của một chương trong khóa học
- deleteChapter(): xóa một chương trong khóa học
- findAllAssignments(): tìm thông tin của tất cả các bài tập trong một chương của khóa học
- createAssignment(): tạo bài tập trong một chương của khóa học
- updateAssignment(): cập nhật thông tin của bài tập trong một chương của khóa học\
- deleteAssignment(): xóa một bài tập trong một chương của khóa học
- submitAssignment(): nộp bài làm
- autoGrade(): tự động chấm điểm

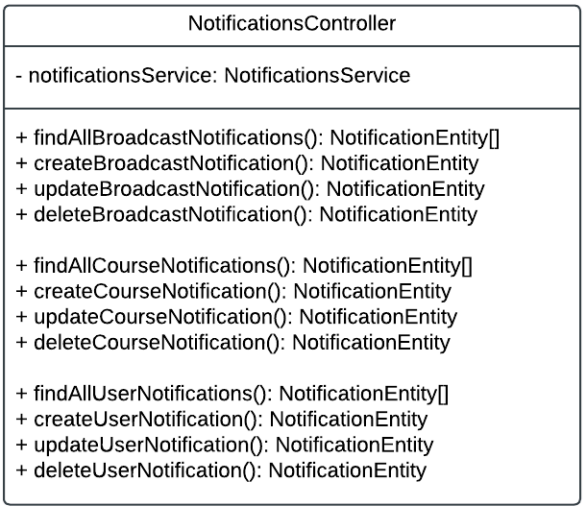
4.2.4 Examinations Controller



Mục đích: Xử lý những thao tác liên quan đến kỳ kiểm tra
Các phương thức:

- findAllExaminations(): tìm thông tin của tất cả các kỳ kiểm tra
- createExamination(): tạo kỳ kiểm tra
- updateExamination(): cập nhật thông tin kỳ kiểm tra
- deleteExamination(): xóa kỳ kiểm tra
- findAllExaminees(): tìm thông tin của tất cả các thí sinh trong kỳ kiểm tra
- addExaminee(): thêm thí sinh (học viên) vào kỳ kiểm tra
- kickExaminee(): xóa thí sinh ra khỏi kỳ kiểm tra
- findAllExaminers(): tìm thông tin của tất cả các giám khảo trong kỳ kiểm tra
- addExaminer(): thêm giám khảo (người hướng) vào kỳ kiểm tra
- kickExaminer(): xóa thí sinh ra khỏi kỳ kiểm tra

4.2.5 Notifications Controller



Mục đích: Xử lý những thông bảo của hệ thống
Các phương thức:

- findAllBroadcastNotifications(): tìm thông tin của các thông báo trên phạm vi toàn ứng dụng

- createBroadcastNotification(): tạo thông báo trên phạm vi toàn ứng dụng
- updateBroadcastNotification(): cập nhật thông tin của các thông báo trên phạm vi toàn ứng dụng
- deleteBroadcastNotification(): xóa thông báo trên phạm vi toàn ứng dụng
- findAllCourseNotifications(): tìm thông tin của các thông báo trên phạm vi khóa học
- createCourseNotification(): tạo thông báo trên phạm vi khóa học
- updateCourseNotification(): cập nhật thông tin của thông báo trên phạm vi khóa học
- deleteCourseNotification(): xóa thông báo trên phạm vi khóa học
- findAllUserNotifications(): tìm thông tin của thông báo được gửi cho người dùng cụ thể
- createUserNotification(): tạo thông báo được gửi cho người dùng cụ thể
- updateUserNotification(): cập nhật thông tin của thông báo được gửi cho người dùng cụ thể
- deleteUserNotification(): xóa thông báo được gửi cho người dùng cụ thể

4.3 Component: Services

4.3.1 Auth Service

- Chịu trách nhiệm xử lý thao tác liên quan đến xác thực và phân quyền cho người dùng.
- Bao gồm đăng ký, đăng nhập và đổi mật khẩu tài khoản.

4.3.2 Audit logs Service

- Chịu trách nhiệm ghi lại những sự thay đổi thông tin tài khoản của Admin đã thực hiện trên người dùng khác.

4.3.3 Users Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến tài khoản và các yếu tố liên quan đến người dùng.
- Bao gồm tìm, thêm, xóa, sửa người dùng, xem danh sách khóa học đã đăng ký, thay đổi thông tin tài khoản.

4.3.4 Courses Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến khóa học.
- Bao gồm tìm, thêm, xóa, sửa thông tin của khóa học.

4.3.5 Enrollments Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến việc tham gia khóa học của học viên
- Bao gồm tìm sự tham gia theo Id học viên, thêm, xóa, các sự tham gia của học viên đối với một khóa học, hoặc tìm các sự tham gia của học viên theo Id khóa học.

4.3.6 Mentor Permissions Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến việc quyền tham gia giảng dạy khóa học của người hướng dẫn
- Bao gồm thêm, xóa quyền tham gia của người hướng dẫn, tìm tất cả các quyền tham gia khóa học của người hướng dẫn.

4.3.7 Chapters Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến thông tin của chương trong khóa học
- Bao gồm tìm, thêm, xóa chương theo Id khóa học, cập nhật thông tin chương trong khóa học.

4.3.8 Assignments Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến bài tập của chương trong khóa học.
- Bao gồm tìm, thêm, xóa bài tập của chương theo Id khóa học và Id chương, cập nhật thông tin bài tập, gợi ý chấm điểm bằng AI cũng như nhận bài nộp của học viên.

4.3.9 Examinations Service

- Chịu trách nhiệm xử lý các thao tác liên quan kỳ kiểm tra.
- Bao gồm tìm, thêm, xóa, sửa thông tin của kỳ kiểm tra, thêm học viên như là thí sinh vào kỳ kiểm tra, thêm

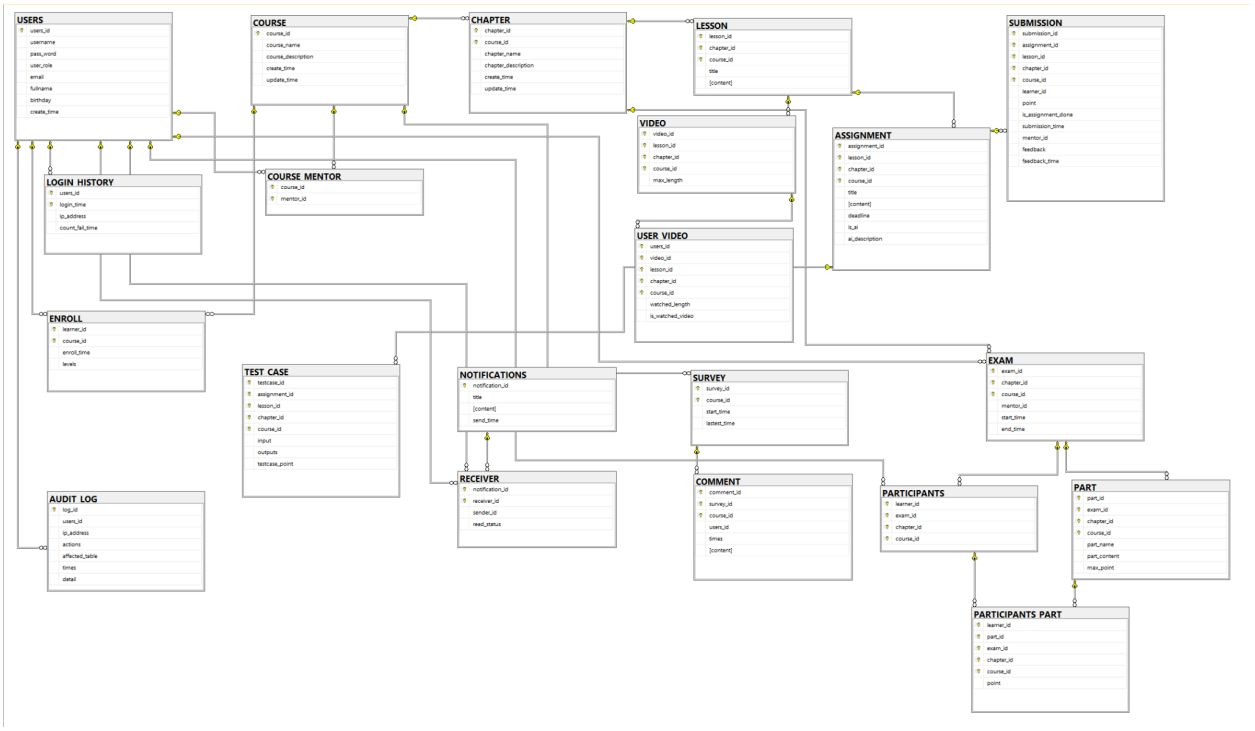
người hướng dẫn như là giám khảo vào kỳ kiểm tra, hủy tư cách tham gia của thí sinh/giám khảo, gợi ý chấm bài bằng AI.

4.3.10 Notifications Service

- Chịu trách nhiệm xử lý các thao tác liên quan đến thông báo của hệ thống.
- Bao gồm tìm, thêm, xóa, sửa các thông báo ở các phạm vi hệ thống/khóa học/người dùng cụ thể.

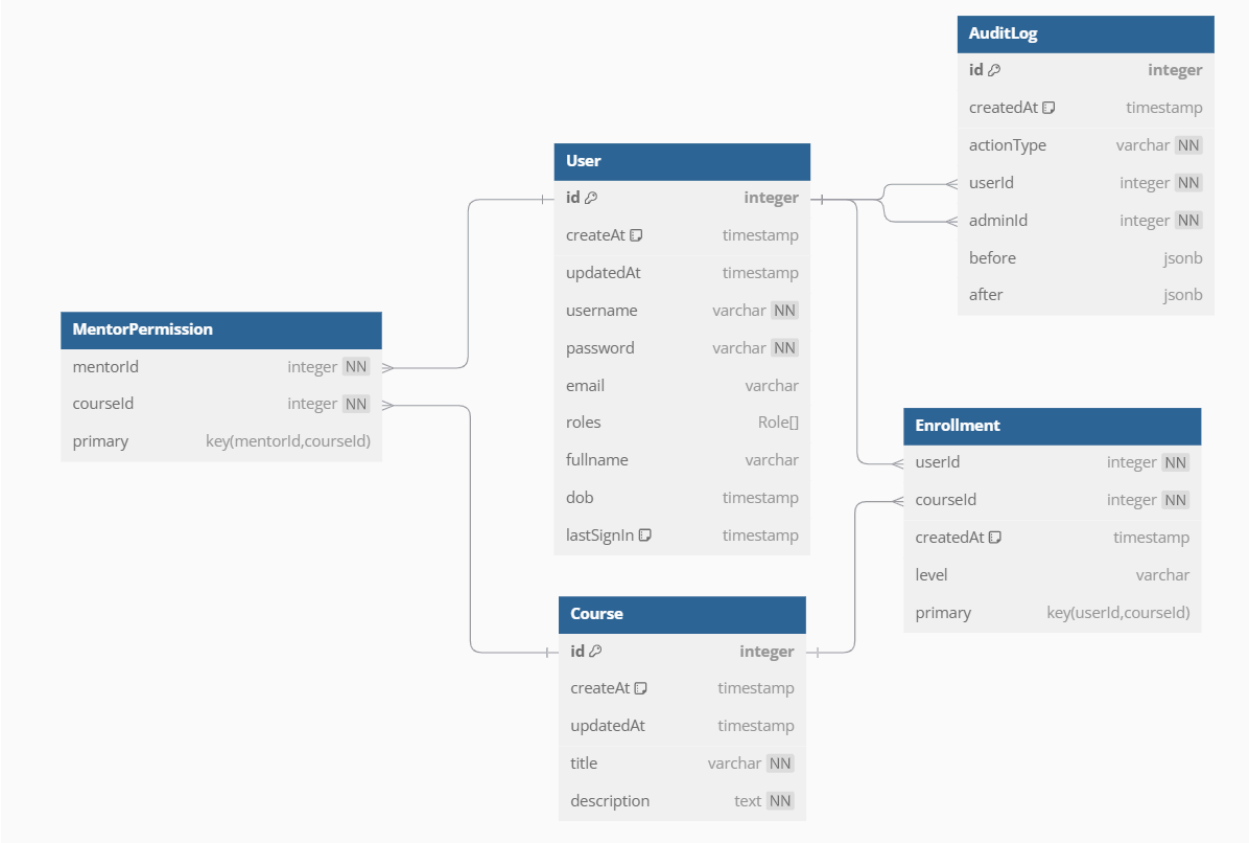
4.4 Component: Model

Model đóng vai trò quan trọng trong việc mô tả cấu trúc dữ liệu và cung cấp các phương thức để tương tác với cơ sở dữ liệu. Model thường được sử dụng để biểu diễn các thực thể (Entities) trong cơ sở dữ liệu, giúp các dịch vụ (Services) dễ dàng thao tác với dữ liệu thông qua ORM (Object-Relational Mapping). Trong ứng dụng này, Prisma ORM được sử dụng để kết nối và tương tác với cơ sở dữ liệu PostgreSQL, giúp tối ưu hóa việc quản lý và truy vấn dữ liệu.



Bản nháp thiết kế cơ sở dữ liệu, xem ảnh rõ nét hơn tại *Database.png*

Nhìn chung, có khá nhiều bảng và được liên kết với nhau tương đối phức tạp, nhưng trong đó có một số model đáng quan tâm sau đây:



Một số Model đáng chú ý

4.4.1 User model

Mục đích: Quản lý thông tin người dùng. Quản lý xác thực và phân quyền.
Các trường:

+ id: Khóa chính của người dùng.
+ username: Tên đăng nhập, duy nhất.
+ password: Mật khẩu được mã hóa.
+ email: Địa chỉ email người dùng.
+ roles: Danh sách các vai trò (Admin, Learner, Mentor).
+ fullname: Họ và tên đầy đủ.
+ dob: Ngày sinh.
+ createAt: Ngày tạo tài khoản (mặc định là thời điểm hiện tại).
+ updatedAt: Ngày cập nhật gần nhất.
+ lastSignIn: Lần đăng nhập cuối.

4.4.2 Course model

Mục đích: Quản lý thông tin khóa học. Lưu trữ mô tả chi tiết khóa học.
Các trường:

+ id: Khóa chính của khóa học.
+ title: Tên khóa học.
+ description: Mô tả nội dung khóa học.
+ createAt: Ngày tạo khóa học (mặc định là thời điểm hiện tại).
+ updatedAt: Ngày cập nhật gần nhất.

4.4.3 Enrollment model

Mục đích: Quản lý đăng ký của người dùng vào khóa học. Lưu thông tin mức độ tham gia và ngày đăng ký.
Các trường**:**

+ userId: Khóa ngoại tham chiếu User.id.
+ courseId: Khóa ngoại tham chiếu Course.id.
+ createdAt: Ngày đăng ký.
+ level: Cấp độ tham gia của người dùng trong khóa học.

4.4.4 AuditLog model

Mục đích: Lưu trữ các hành động của hệ thống để phục vụ mục đích kiểm tra và xử lý sự cố. Lưu giữ dữ liệu trước và sau khi thay đổi.
Các trường:

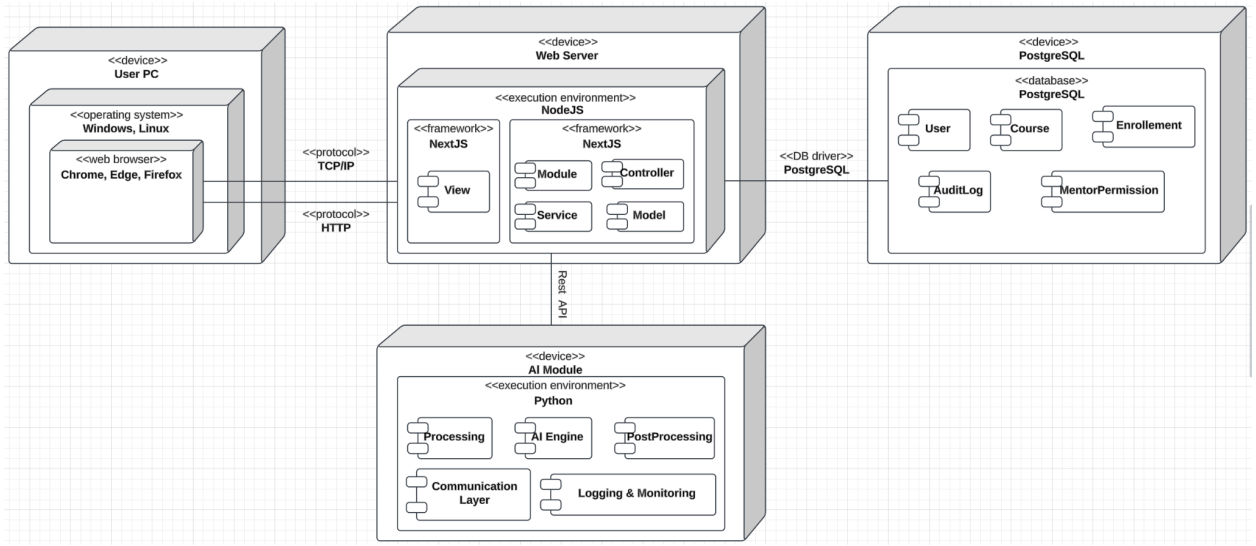
+ id: Khóa chính của bản ghi.
+ createdAt: Ngày thực hiện hành động.
+ actionType: Loại hành động (thêm, sửa, xóa,...).
+ userId: Người thực hiện hành động, tham chiếu User.id.
+ adminId: Quản trị viên liên quan, tham chiếu User.id.
+ before: Dữ liệu trước khi thay đổi (dạng JSON).
+ after: Dữ liệu sau khi thay đổi (dạng JSON).

4.4.5 MentorPermission model

Mục đích: Quản lý quyền truy cập của giảng viên vào các khóa học. Xác định các khóa học mà giảng viên được phép giảng dạy.
Các trường:

+ mentorId: Khóa ngoại tham chiếu User.id.
+ courseId: Khóa ngoại tham chiếu Course.id.

5. Deployment



5.1 Client-side:

Truy cập: Người dùng có thể truy cập website hệ thống quản lý khóa học trực tuyến từ máy tính cá nhân.

Tương thích: Hệ thống hỗ trợ các trình duyệt phổ biến như Chrome, Microsoft Edge, Firefox trên các hệ điều hành Windows, Linux. Người dùng chỉ cần nhập URL hoặc địa chỉ IP của website vào trình duyệt để sử dụng.

Kết nối: Trình duyệt gửi yêu cầu HTTP đến server, thiết lập kết nối TCP giữa trình duyệt và máy chủ.

5.2 Server-side:

Công nghệ: Hệ thống back-end được xây dựng bằng NestJS với ngôn ngữ TypeScript. Giao diện web được tải bởi framework NextJS kết hợp với MUI.

Hosting: Hệ thống web được lưu trữ trên các máy chủ chạy hệ điều hành Windows và triển khai qua Docker, xử lý yêu cầu từ client và phản hồi bằng HTTP.

Components:

- View: Hiển thị giao diện người dùng.
- Module: NestJS được chia thành các module để quản lý và phát triển hệ thống một cách rõ ràng và dễ bảo trì: AuthModule, Users Module, Courses Module,...
- Controller: Quản lý dữ liệu từ phía người dùng, xử lý logic và kết nối với mô hình dữ liệu để truy xuất hoặc cập nhật thông tin.
- Routes: Định nghĩa các đường dẫn URL và liên kết chúng đến controller tương ứng.
- Models: Xác định cấu trúc và tương tác với cơ sở dữ liệu để lưu trữ và truy xuất.

Giao tiếp với cơ sở dữ liệu: Website giao tiếp với cơ sở dữ liệu PostgreSQL thông qua Prisma ORM để thực hiện các thao tác lưu trữ và truy vấn dữ liệu.

5.3 Database:

Lựa chọn cơ sở dữ liệu: Hệ thống sử dụng PostgreSQL - 1 hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, mạnh mẽ, hỗ trợ khả năng mở rộng và bảo mật cao.

Schema:

- Users: Lưu trữ thông tin người dùng bao gồm ID, tên, email, mật khẩu,...
- Courses: Lưu trữ thông tin về các khóa học, bao gồm tên khóa học, mô tả, ngày bắt đầu, ngày kết thúc.
- Enrollments: Quản lý thông tin đăng ký của người dùng vào các khóa học.
- AuditLog: Lưu trữ các hành động của hệ thống để phục vụ kiểm tra và xử lý sự cố.
- MentorPermission: Quản lý quyền truy cập của giảng viên vào các khóa học.

5.4 AI Module:

Hệ thống sử dụng AI để nhận diện gian lận trong thi cử thông qua phân tích video giám sát từ camera.

Công nghệ: Sử dụng ngôn ngữ lập trình Python với thư viện PyTorch.

Component:

- Processing: Xử lý video đầu vào.
- AI Engine: Chạy mô hình AI đã huấn luyện. Mô hình được huấn luyện bằng PyTorch với tập dữ

liệu liên quan đến các hành vi bất thường trong thi cử.

- PostProcessing: Xử lý kết quả đầu ra từ AI Engine để tạo báo cáo.
- Communication Layer: Cung cấp API (REST) để gửi, nhận dữ liệu với Web Server.
- Logging & Monitoring: Ghi lại nhật ký hoạt động của AI Module.

6. Implementation View

6.1 Front-end:

Thư mục gốc:

- **cypress**: Thư mục chứa các tập tin cấu hình và test case cho Cypress - một công cụ tự động hóa test end-to-end phổ biến.
- **node_modules**: Thư mục chứa tất cả các thư viện và phụ thuộc được cài đặt bởi npm hoặc yarn. Đây là nơi lưu trữ các package mà dự án frontend sử dụng.
- **public**: Thư mục chứa các tài sản tĩnh của ứng dụng, chẳng hạn như hình ảnh, file CSS, HTML, JavaScript mà không cần build. Các file trong thư mục này sẽ được copy trực tiếp vào thư mục output khi build dự án.
- **src**: Thư mục nguồn chính của dự án, chứa toàn bộ mã nguồn của ứng dụng.

Thư mục src:

- **test**: Chứa các test case, thường sử dụng các framework như Jest để kiểm thử đơn vị và tích hợp.
- **app**: Có thể là thư mục chứa các thành phần chính của ứng dụng, như các container hoặc page.
- **components**: Chứa các component tái sử dụng được, là các khối xây dựng cơ bản của giao diện người dùng.
- **config:** Chứa các file cấu hình của ứng dụng, chẳng hạn như các biến môi trường, các thiết lập liên quan đến API hoặc các thư viện khác.
- **context:** Có thể chứa các context (context trong React) để quản lý trạng thái toàn cục của ứng dụng.
- **interfaces:** Chứa các interface để định nghĩa kiểu dữ liệu cho các component và function.
- **stories:** Thư mục này thường được sử dụng với các thư viện như Storybook để tạo các stories (ví dụ) về các component, giúp cho việc phát triển và thiết kế giao diện trở nên trực quan hơn.
- **styles:** Chứa các file CSS hoặc các module CSS để định dạng giao diện.
- **utils:** Chứa các function hoặc helper utility để thực hiện các tác vụ chung.
- **middleware.ts:** Có thể chứa các middleware function để xử lý các request trước khi chúng đến component.

Các file cấu hình khác:

- **.dockerignore:** Chỉ định các file hoặc thư mục không cần đưa vào image Docker.
- **.env:** Chứa các biến môi trường của dự án.
- **.eslintrc.json:** Cấu hình cho ESLint, một công cụ để kiểm tra chất lượng code.

- **.gitignore:** Chỉ định các file hoặc thư mục không cần đưa vào hệ thống quản lý phiên bản Git.
- **Dockerfile:** File cấu hình để xây dựng image Docker.
- **jest.config.js:** Cấu hình cho Jest, một framework để kiểm thử JavaScript.
- **package-lock.json:** Lưu trữ thông tin chi tiết về các phiên bản của các dependency đã cài đặt.
- **package.json:** File mô tả dự án, bao gồm các dependency, script, và các thông tin khác.
- **tsconfig.json:** File cấu hình cho TypeScript.
- **README.md:** File chứa thông tin về dự án, hướng dẫn sử dụng, và các thông tin khác.
- **docker-entrypoint.sh:** Script được thực thi khi container Docker khởi động.
- **cypress.config.ts:** File cấu hình cho Cypress.
- **next-env.d.ts:** File khai báo kiểu cho các biến môi trường trong Next.js.

6.2 Back-end:

Thư mục gốc:

- **dist**: chứa mã JavaScript đã được biên dịch từ TypeScript, sẵn sàng để triển khai.
- **node_module**: Bao gồm tất cả các thư viện và phụ thuộc được cài đặt thông qua npm.
- **prisma**: Quản lý migrations, mô hình dữ liệu (schema.prisma), thêm dữ liệu mẫu (seed.ts), và tránh xung đột (migration_lock.toml).
- **src**: Thư mục nguồn chính của dự án, chứa toàn bộ logic ứng dụng.

Thư mục src:

- **audit-logs**: Ghi lại và quản lý nhật ký hệ thống.
- **auth**: Xử lý xác thực người dùng, JWT, bảo mật, quyền truy cập, và đối tượng truyền dữ liệu (DTOs).
- **config**: Cấu hình toàn cục cho ứng dụng, bao gồm kết nối cơ sở dữ liệu, JWT, và các thông số khác.
- **courses**: Quản lý khóa học.
- **enrollment**: Ghi danh người dùng vào các khóa học.
- **mentor-permission**: Quản lý quyền và chức năng dành riêng cho mentor trong hệ thống.
- **prisma**: Tích hợp NestJS, hỗ trợ kết nối và thao tác cơ sở dữ liệu dễ dàng.
- **users**: Quản lý người dùng.
    + **dto**: Định nghĩa cấu trúc dữ liệu đầu vào/đầu ra.
    + **entities**: Mô hình dữ liệu ứng với cơ sở dữ liệu.
    + **controller**: Xử lý yêu cầu HTTP, điều phối luồng dữ liệu.
    + **module**: Tổ chức, gom nhóm các thành phần liên quan.
    + **service**: Chứa logic nghiệp vụ chính của ứng dụng.

Tệp cấu hình và hỗ trợ:

- **.env:** Tệp môi trường, chứa các biến nhạy cảm như khóa JWT, thông tin cơ sở dữ liệu.
- **.eslintrc.js**: Thiết lập ESLint để kiểm tra và chuẩn hóa code TypeScript.
- **.gitignore**: Liệt kê các tệp/thư mục không được theo dõi bởi Git.
- **.prettierrc**: Cấu hình Prettier, đảm bảo định dạng mã nguồn nhất quán.
- **docker-compose.yml**: Thiết lập môi trường PostgreSQL sử dụng Docker.
- **nest-cli.json**: Cấu hình CLI cho NestJS
- **package.json**: Quản lý thông tin dự án, các phụ thuộc và các script hỗ trợ.
- **package-lock.json**: Khóa phiên bản của các phụ thuộc.
- **README.md**: Hướng dẫn và mô tả dự án.

Với cấu trúc này, ứng dụng được tổ chức rõ ràng, dễ mở rộng và duy trì. Các thư mục được phân chia theo chức năng hoặc module riêng biệt, giúp quản lý logic ứng dụng một cách hiệu quả.