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

BÁO CÁO ĐỒ ÁN MÔN HỌC

MÔN HỌC: Đồ Án Tổng Hợp - Công nghệ Phần mềm

Đề tài: Grade Portal for students at HCMUT

HK241

Giảng viên hướng dẫn: Lê Đình Thuận
Sinh viên: Trần Đại Việt - 2213951
Phạm Văn Quốc Việt - 2213950
Nguyễn Nhật Khoa - 2211629
Phạm Việt Anh - 2210128
Nguyễn Gia Nguyên - 2212303
Lê Đăng Khoa - 2211599

HO CHI MINH CITY, SEPTEMBER 2024

Mục lục

1 Danh Sách Thành Viên Và Phân Công Nhiệm Vụ

MSSV	Họ và tên	Nhiệm vụ	Phần trăm công việc
2213951	Trần Đại Việt	PO	100%
2213950	Phạm Văn Quốc Việt	DEV	100%
2211629	Nguyễn Nhật Khoa	DEV	100%
2210128	Phạm Viết Anh	DEV	100%
2212303	Nguyễn Gia Nguyên	DEV	100%
2211599	Lê Đăng Khoa	DEV	100%

Bảng 1: *Danh sách sinh viên và nhiệm vụ*

2 Giới Thiệu Chung Về Dự Án

2.1 Bối cảnh chung và tính cấp thiết của đề tài

Trong bối cảnh số lượng sinh viên tại Đại học Bách Khoa - Đại học Quốc gia TP.HCM không ngừng gia tăng qua từng năm, việc quản lý thông tin học tập và điểm số của sinh viên đang trở thành một thách thức ngày càng lớn. Cùng với đó, chương trình đào tạo tại trường liên tục được cập nhật để bắt kịp với sự phát triển nhanh chóng của công nghệ và kiến thức toàn cầu, đặt ra yêu cầu cấp bách về việc xây dựng các hệ thống hỗ trợ quản lý và tra cứu thông tin học tập một cách hiệu quả. Đặc biệt, việc cung cấp một hệ thống cho phép sinh viên có thể tra cứu điểm số trong quá trình học tập là một yếu tố quan trọng, không chỉ giúp sinh viên có thể theo dõi kết quả học tập của mình mà còn tạo điều kiện thuận lợi cho giảng viên trong việc quản lý lớp học và các dữ liệu liên quan đến điểm số.

Hiện nay, tại nhiều trường đại học trên thế giới, việc áp dụng các hệ thống quản lý điểm số thông minh đã trở thành xu hướng phổ biến, nhằm giúp cả sinh viên lẫn giảng viên dễ dàng truy cập, quản lý và cập nhật thông tin về kết quả học tập. Tại Đại học Bách Khoa, nhu cầu này cũng đang trở nên cấp thiết khi số lượng sinh viên ngày càng tăng, đặc biệt là trong những môn học có đông sinh viên tham gia. Việc áp dụng một hệ thống giúp sinh viên có thể tra cứu điểm số, theo dõi quá trình học tập và cập nhật kết quả của mình sẽ giúp giảm bớt áp lực cho cả giảng viên và sinh viên trong việc xử lý các thông tin về điểm số một cách thủ công, đồng thời giúp nâng cao tính minh bạch và chính xác trong việc công bố kết quả học tập.

Vì vậy, dự án Grade Portal được đề xuất với mục tiêu xây dựng một hệ thống quản lý điểm số trực tuyến nhằm hỗ trợ sinh viên và giảng viên tại Đại học Bách Khoa - Đại học Quốc gia TP.HCM trong việc theo dõi, tra cứu và quản lý điểm số một cách thuận tiện và chính xác. Với sự phát triển không ngừng của công nghệ thông tin và nhu cầu hiện đại hóa trong giáo dục, Grade Portal hứa hẹn sẽ là một giải pháp toàn diện giúp nâng cao trải nghiệm học tập và giảng dạy trong nhà trường.

Một trong những yếu tố làm cho hệ thống này trở nên cần thiết là tính minh bạch và khả năng tự động cập nhật của nó. Khi sinh viên có thể tra cứu điểm số một cách thường xuyên , họ sẽ có cái nhìn rõ ràng hơn về tiến trình học tập của mình, từ đó có thể tự điều chỉnh chiến lược học tập một cách phù hợp. Bên cạnh đó, giảng viên sẽ có khả năng đính kèm các bảng điểm dưới dạng tập tin CSV, Excel, giúp họ dễ dàng quản lý và theo dõi các môn học mà mình phụ trách. Ngoài ra, việc hệ thống cho phép admin phân quyền cho giảng viên cũng sẽ tạo điều kiện thuận lợi cho nhà trường trong việc quản lý hệ thống điểm số một cách linh hoạt và hiệu quả.

Một điểm nổi bật khác của hệ thống này là tính năng Hall of Fame, nơi ghi nhận và tôn vinh các sinh viên có thành tích học tập xuất sắc. Điều này không chỉ tạo động lực cho sinh viên phấn đấu học tập mà còn góp phần xây dựng môi trường học tập tích cực và khuyến khích sự nỗ lực từ phía sinh viên. Việc tôn vinh những cá nhân có thành tích cao sẽ khuyến khích tinh thần học tập và cạnh tranh lành mạnh trong toàn trường, từ đó nâng cao chất lượng giảng dạy và học tập.

Ngoài ra, việc xây dựng một hệ thống quản lý điểm số thông minh còn đáp ứng nhu cầu hiện đại hóa quá trình giảng dạy và quản lý học tập tại Đại học Bách Khoa, góp phần cải thiện trải nghiệm của sinh viên và giảng viên. Thay vì phải tra cứu thông tin điểm số thông qua các phương thức truyền thống như email, giấy tờ, hoặc trực tiếp gặp gỡ giảng viên, hệ thống mới sẽ giúp sinh viên truy cập thông tin một cách dễ dàng hơn, tiết kiệm thời gian và nâng cao hiệu quả trong việc quản lý thông tin học tập.

Nhìn chung, nếu thành công, Grade Portal không chỉ là công cụ hỗ trợ đắc lực cho sinh viên và giảng viên mà còn đóng góp quan trọng vào việc hiện đại hóa và nâng cao chất lượng giáo dục tại Đại học Bách Khoa. Hệ thống sẽ trở thành một bước tiến lớn trong quá trình chuyển đổi số trong giáo dục, mang lại hiệu quả cao và đáp ứng được nhu cầu ngày càng tăng của một ngôi trường đại học hiện đại.

2.2 Các bên liên quan (Stakeholders) và nhu cầu của họ

- **Sinh viên:** Sinh viên cần một hệ thống tra cứu điểm số nhanh chóng, minh bạch và dễ dàng truy cập để theo dõi kết quả học tập của mình. Họ mong muốn có thể kiểm tra điểm số một cách tự động và liên tục cập nhật khi có sự thay đổi. Ngoài ra, sinh viên cũng mong đợi hệ thống cung cấp thông tin rõ ràng và có tính bảo mật cao để đảm bảo quyền lợi học tập của mình.
- **Giảng viên và cán bộ công tác tại trường:** Giảng viên cần một hệ thống quản lý điểm số hiệu quả, có khả năng xử lý các tệp dữ liệu lớn (danh sách điểm sinh viên) và hỗ trợ cập nhật liên tục để đảm bảo độ chính xác của điểm số. Họ cần một hệ thống linh hoạt, cho phép dễ dàng đính kèm và cập nhật các bảng điểm từ file CSV, Excel và hệ thống phải tự động tải và cập nhật khi có phiên bản mới.
- **HCMUT Administrator:** Admin có nhiệm vụ quản lý người dùng và phân quyền cho giảng viên trong hệ thống. Họ cần một hệ thống dễ sử dụng để phân quyền, quản lý tài khoản của giảng viên và sinh viên, đồng thời phải đảm bảo bảo mật thông tin trong suốt quá trình vận hành hệ thống. Admin cũng cần đảm bảo tính ổn định của hệ thống trong quá trình cập nhật và xử lý dữ liệu.

2.3 Lợi ích nhận được khi dự án hoàn thành

- **Sinh viên:** Sinh viên sẽ có thể dễ dàng tra cứu điểm số một cách chính xác và cập nhật kịp thời trong quá trình học tập. Hệ thống giúp sinh viên theo dõi kết quả học tập của mình một cách minh bạch và rõ ràng, từ đó điều chỉnh chiến lược học tập phù hợp. Việc truy cập bảng điểm qua hệ thống trực tuyến sẽ giúp sinh viên tiết kiệm thời gian và tăng cường sự chủ động trong quá trình học tập. Đặc biệt, tính năng Hall of Fame sẽ tạo động lực cho sinh viên phấn đấu đạt thành tích cao.
- **Giảng viên và cán bộ công tác tại trường:** Giảng viên sẽ được hỗ trợ tối đa trong việc quản lý và cập nhật điểm số. Họ có thể dễ dàng tải lên và quản lý các bảng điểm từ file CSV, và hệ thống sẽ tự động cập nhật khi có thay đổi. Điều này giúp giảm thiểu khối lượng công việc thủ công, đồng thời đảm bảo tính chính xác và minh bạch trong quá trình chấm điểm và công bố kết quả.
- **HCMUT Administrator:** Hệ thống sẽ cung cấp cho Admin khả năng quản lý người dùng và phân quyền một cách dễ dàng, đồng thời đảm bảo tính bảo mật trong việc quản lý thông tin của sinh viên và giảng viên. Ngoài ra, Admin cũng có thể giám sát toàn bộ hệ thống để đảm bảo quá trình vận hành trơn tru, đồng thời xử lý các vấn đề phát sinh nhanh chóng và hiệu quả.

3 Phân Tích Các Yêu Cầu Của Dự Án

3.1 Các yêu cầu chức năng (Functional Requirements)

1. **Admin:**
   - Phải trải qua bước đăng nhập để xác thực quyền truy cập vào hệ thống thông qua Google Authentication.
   - Có thể đăng xuất khỏi hệ thống sau khi thực hiện xong các hành động của mình.
   - Có khả năng thêm mới và quản lý tài khoản giảng viên.
   - Có khả năng tạo lớp học mới, gán sinh viên và giảng viên phụ trách vào lớp học.
   - Có thể phân quyền cho giảng viên quản lý lớp học tương ứng.
   - Có thể thêm hàng loạt sinh viên vào hệ thống bằng danh sách Gmail do trường cung cấp.
   - Có thể tìm kiếm lớp học theo các điều kiện con.
2. **Giảng viên:**
   - Phải trải qua bước đăng nhập để xác thực quyền truy cập vào hệ thống thông qua Gmail trường cấp.
   - Có thể đăng xuất khỏi hệ thống sau khi thực hiện xong các hoạt động của mình.
   - Có khả năng quản lý các lớp học được phân quyền bởi admin.
   - Có thể thêm sinh viên vào lớp học mình quản lý.
   - Có thể tải lên file CSV hoặc Excel chứa bảng điểm cho môn học.
   - Hệ thống sẽ tự động giám sát và đồng bộ file CSV khi có phiên bản mới và cập nhật điểm vào cơ sở dữ liệu.
3. **Sinh viên:**
   - Phải trải qua bước đăng nhập để xác thực quyền truy cập vào hệ thống thông qua Gmail trường cấp.
   - Có thể tra cứu và xem điểm của mình cho tất cả các môn học đã đăng ký.
   - Có thể tìm kiếm điểm theo môn học, học kỳ, năm học hoặc MSSV.
   - Có thể gửi phản hồi trực tiếp đến giảng viên phụ trách thông qua hệ thống (tùy chọn).
   - Có thể xem thông tin chi tiết về tình trạng lớp học đang tham gia và giảng viên phụ trách lớp đó.
4. **Chức năng Bảng Vinh Danh (Hall of Fame):**
   - Hệ thống phải có một Bảng Vinh Danh, nơi liệt kê các sinh viên có thành tích xuất sắc dựa trên điểm trung bình tích lũy (GPA) hoặc kết quả môn học cụ thể.
   - Tiêu chí để được liệt kê vào Bảng Vinh Danh có thể do Admin hoặc giảng viên thiết lập.

3.2 Các yêu cầu phi chức năng (Non-functional Requirements)

1. **Bảo mật (Security Requirements)**
   - Tất cả người dùng (Admin, giảng viên, sinh viên) đều phải xác thực bằng tài khoản Gmail trường cấp trước khi sử dụng hệ thống.
   - Thông tin về điểm số, dữ liệu bảng điểm và các thông tin cá nhân liên quan đến sinh viên và giảng viên phải được bảo mật, chỉ những người có quyền truy cập mới có thể xem hoặc chỉnh sửa.
   - Hệ thống phải đảm bảo an toàn cho dữ liệu CSV được tải lên và đồng bộ hóa, chỉ có giảng viên phụ trách lớp học và admin mới có quyền chỉnh sửa các bảng điểm này.
2. **Hiệu suất hoạt động (Performance)**
   - Hệ thống phải hỗ trợ nhiều người dùng đồng thời mà không bị giảm hiệu suất, đặc biệt là trong các thời điểm cao điểm như cuối học kỳ khi sinh viên tra cứu điểm.
   - Hệ thống phải có khả năng phản hồi nhanh chóng, đặc biệt đối với các thao tác tải lên file CSV, Excel hoặc khi tra cứu thông tin điểm số.
   - Hệ thống phải cung cấp phản hồi ngay lập tức về việc tải lên thành công hay thất bại của bảng điểm, cũng như khi sinh viên thực hiện tra cứu điểm.

3. **Khả năng sử dụng (Usability)**
    - Giao diện người dùng (UI) của hệ thống phải rõ ràng, trực quan, dễ dàng sử dụng để giảng viên và sinh viên có thể tra cứu, tải lên, hoặc chỉnh sửa thông tin một cách nhanh chóng.
    - Hệ thống phải dễ sử dụng để người dùng (giảng viên, sinh viên) có thể nhanh chóng nắm bắt cách thao tác, ngay cả khi họ sử dụng lần đầu.
    - Đảm bảo số lần thao tác tối thiểu để truy cập vào tính năng tra cứu điểm hoặc cập nhật bảng điểm.
    - Hệ thống cần cung cấp các thông báo lỗi rõ ràng khi có vấn đề xảy ra, chẳng hạn như file không hợp lệ, lỗi trong quá trình đồng bộ dữ liệu, hoặc kết nối gián đoạn.
4. **Toàn vẹn dữ liệu (Data Integrity)**
    - Mỗi yêu cầu tải lên bảng điểm và tra cứu điểm số phải ghi lại chính xác thông tin như ID sinh viên, môn học, thời gian tải lên, và điểm số tương ứng.
    - Hệ thống phải đảm bảo rằng các dữ liệu bảng điểm không thể bị chỉnh sửa hoặc xóa bỏ bởi những người không có thẩm quyền.
    - Mọi lịch sử và báo cáo liên quan đến việc tải lên và chỉnh sửa bảng điểm phải được duy trì toàn vẹn và bảo mật, đảm bảo rằng không có sự thay đổi dữ liệu trái phép.
5. **Tính sẵn sàng (Availability)**
    - Hệ thống phải luôn sẵn sàng phục vụ giảng viên và sinh viên trong suốt thời gian học tập và giảng dạy (6h-22h mỗi ngày).
    - Hệ thống phải được bảo trì định kỳ mà không làm gián đoạn hoạt động của người dùng. Các thông báo bảo trì phải được gửi trước để người dùng có thể sắp xếp công việc.
    - Hệ thống phải có khả năng xử lý tình huống khẩn cấp, đảm bảo rằng dữ liệu điểm số không bị mất hoặc hư hỏng do lỗi hệ thống.
6. **Khả năng mở rộng (Scalability)**
    - Hệ thống phải được thiết kế để có thể dễ dàng mở rộng khi số lượng sinh viên và giảng viên sử dụng hệ thống tăng lên trong tương lai.
7. **Khả năng bảo trì (Maintainability)**
    - Tất cả các công nghệ và chi tiết kỹ thuật sử dụng trong dự án phải được ghi lại rõ ràng trong tài liệu kỹ thuật, đảm bảo dễ dàng tra cứu và bảo trì.
    - Hệ thống phải có kế hoạch kiểm tra và nâng cấp định kỳ nhằm đảm bảo luôn hoạt động ổn định.
    - Các lỗi liên quan đến việc đồng bộ bảng điểm hoặc hệ thống phải được thông báo và khắc phục kịp thời.

4 Kiến trúc hệ thống (System Architecture)

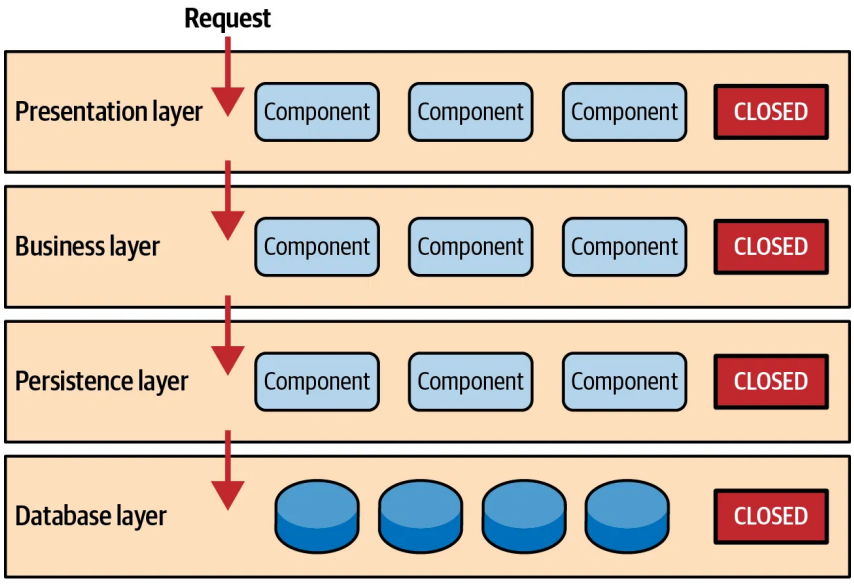
4.1 Layered Architecture

Khi phát triển một hệ thống như HCMUT Grade Portal, việc xác định kiến trúc hệ thống là yếu tố vô cùng quan trọng. Kiến trúc hệ thống đóng vai trò là "bộ khung" tổng thể, giúp định hình cách các thành phần trong hệ thống tương tác với nhau và đảm bảo hệ thống vận hành hiệu quả, an toàn và dễ bảo trì. Một kiến trúc được thiết kế tốt không chỉ hỗ trợ việc phát triển ban đầu mà còn đảm bảo hệ thống có khả năng mở rộng trong tương lai, đáp ứng được các nhu cầu mới mà không gây ảnh hưởng lớn đến các phần khác của hệ thống. Đặc biệt, với hệ thống yêu cầu độ chính xác cao khi tương tác với nhiều đối tượng thành phần như Grade Portal việc có một kiến trúc chặt chẽ và dễ bảo trì sẽ giúp việc vận hành và mở rộng hệ thống trở nên đơn giản hơn.

Mặc dù kiến trúc **Monolithic** có một số hạn chế, nhưng trong trường hợp của Grade Portal, đây vẫn là lựa chọn khả thi. Hạn chế chính của kiến trúc monolithic bao gồm việc khó mở rộng khi hệ thống lớn dần lên, khả năng bảo trì thấp khi phải thay đổi một phần nhỏ của hệ thống nhưng có thể ảnh hưởng đến toàn bộ ứng dụng, và độ phức tạp gia tăng khi tích hợp các chức năng mới. Việc kiểm tra và triển khai một thay đổi nhỏ trong hệ thống monolithic có thể dẫn đến việc phải triển khai lại toàn bộ hệ thống, gây ra thời gian ngừng hoạt động và tăng nguy cơ lỗi.

Tuy nhiên, vẫn có lý do để chọn kiến trúc **Monolithic** ở giai đoạn này. Đầu tiên, kiến trúc monolithic dễ triển khai và phát triển nhanh hơn, đặc biệt trong giai đoạn khởi đầu của dự án. Với một đội ngũ nhỏ và khi chưa có nhiều yêu cầu phức tạp về khả năng mở rộng, việc tập trung vào một ứng dụng duy nhất giúp tiết kiệm thời gian phát triển, đơn giản hóa việc quản lý code và dễ dàng kiểm thử hệ thống. Hơn nữa, với quy mô của Grade Portal ở giai đoạn ban đầu, hệ thống vẫn có thể duy trì được hiệu suất tốt khi hoạt động dưới dạng một ứng dụng monolithic. Khi hệ thống lớn dần và có nhiều tính năng mới, việc chuyển đổi sang các kiến trúc phức tạp hơn (như microservices) hoàn toàn có thể được cân nhắc sau này.

Ngoài ra để có thể nhanh chóng nắm rõ kiến trúc cần hiện thực, nhóm chúng em quyết định sẽ xây dựng hệ thống theo kiến trúc phân lớp (**Layered Architecture**), một kiến trúc thuộc loại thuộc loại kiến trúc monolithic, để đảm bảo tính tổ chức và dễ bảo trì trong quá trình phát triển. Layered Architecture sẽ chia hệ thống thành nhiều tầng (layers) với các chức năng cụ thể, đảm bảo mỗi tầng chỉ chịu trách nhiệm cho một nhiệm vụ cụ thể và độc lập với các tầng khác.

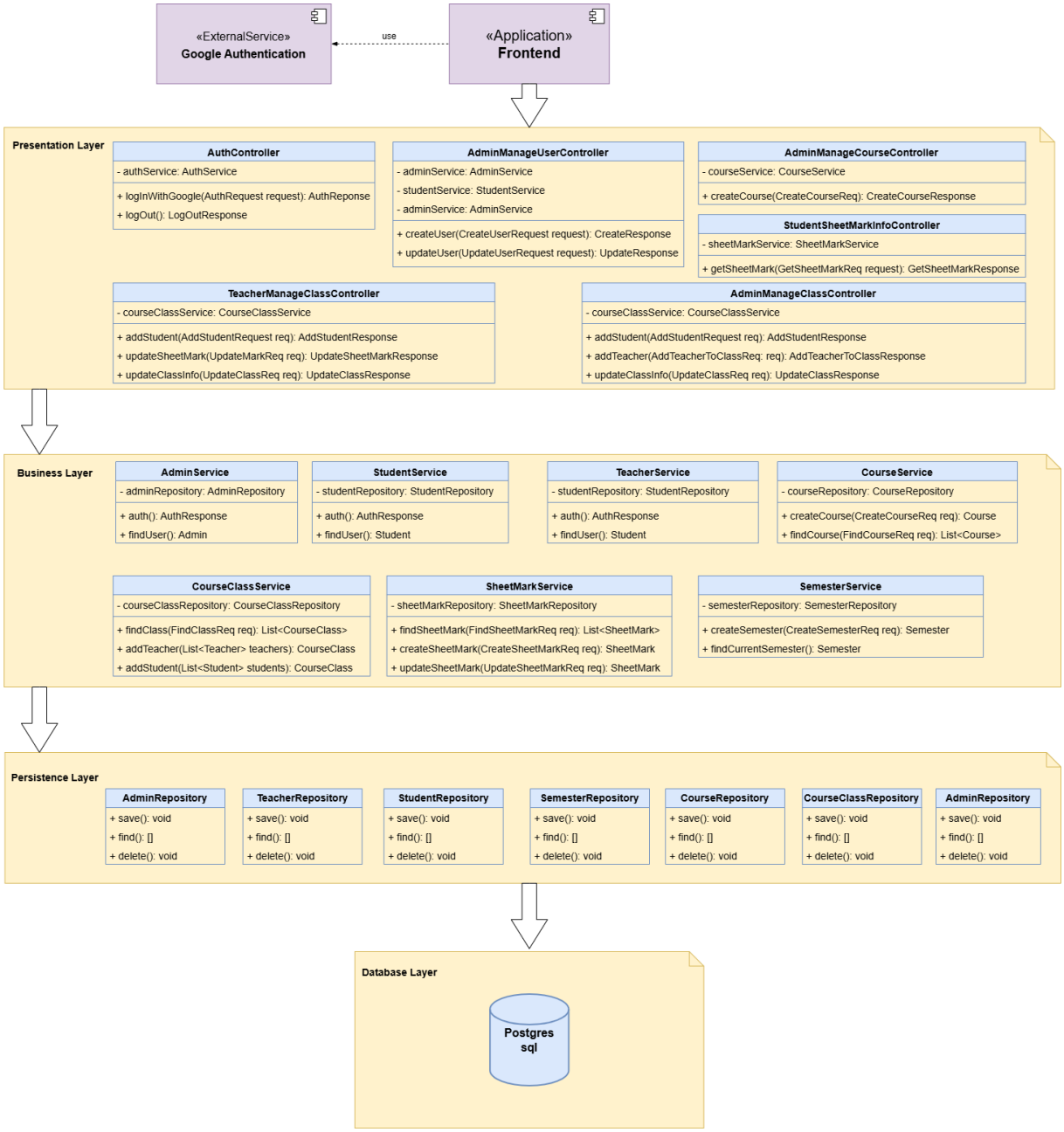


Hình 1: *Sơ đồ kiến trúc phân tầng*

Đây là một architecture pattern 3 lớp vật lý (3-tier) hay n-tier. Tất cả logic điều được move lên phía server, do đó giải quyết triệt để vấn đề về mở rộng, client bây giờ chỉ làm nhiệm vụ render mà không cần biết các thay đổi từ server ra sao.

4.2 Sơ đồ hiện thực của hệ thống

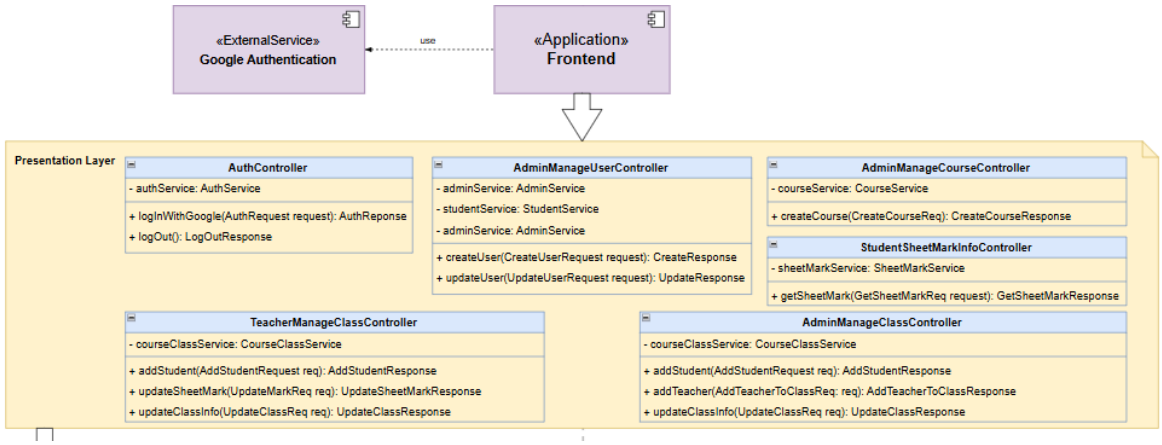
Trong phần này nhóm đã sử dụng **Draw.io** để hiện thực sơ đồ. Đường dẫn cụ thể tới sơ đồ được hiện thực nằm ở **đây**.



Hình 2: *Sơ đồ hiện thực chi tiết hệ thống*

Áp dụng kiến trúc **Layered Architecture** vào việc hiện thực Server của hệ thống. Luồng dữ liệu được thiết kế chạy theo một luồng đơn hướng, có nghĩa là Layer cao hơn sẽ có khả năng kéo được dữ liệu từ các Layer thấp hơn, nhưng ngược lại, các Layer thấp hơn sẽ không được kéo dữ liệu được từ các Layer trên mình. Điều đó tuy đảm bảo sự tách biệt rõ ràng giữa các Layer, tăng cường khả năng bảo trì và khả năng mở rộng của hệ thống nhưng nếu chúng ta không tuân theo, chúng ta có thể gặp phải vấn đề như phụ thuộc tuần hoàn (Dependency Injection).

4.2.1 Presentation Layer

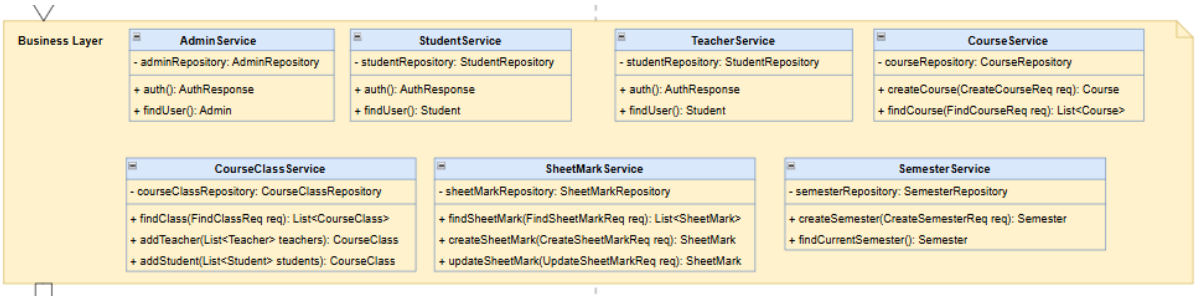


Hình 3: *Presentation Layer Of Layered Architecture*

Để có thể đem đến cho hệ thống khả năng cung cấp cho người dùng sự tương tác và cập nhập thông tin trực tiếp với giao diện, Presentation Layer là tầng đầu tiên, cũng như là tầng đảm nhiệm việc tiếp nhận các lần gọi Api từ phía Client có thể từ Mobile App hay Web App. Nhiệm vụ của tầng này là tiếp nhận và xử lý dữ liệu đầu vào được người dùng gửi từ giao diện, sau đó kiểm tra tính đầy đủ, đúng đắn để sau đó chuyển tiếp chúng xuống tầng Business Layer nơi hiện thực đầy đủ logic hệ thống để xử lý yêu cầu của người dùng. Ngoài ra đây cũng là tầng phải đảm bảo rằng dữ liệu trả về cho Client là đúng đắn, có nghĩa là phải đúng với cấu trúc được mô tả trong Document.

Tóm lại, trong kiến trúc phân tầng, tầng **Presentation** đóng một vai trò quan trọng, chịu trách nhiệm tiếp nhận lời gọi từ Client, cũng như đảm bảo dữ liệu đầu vào và đầu ra để cho hệ thống hoạt động một cách đúng đắn nhất.

4.2.2 Business Layer



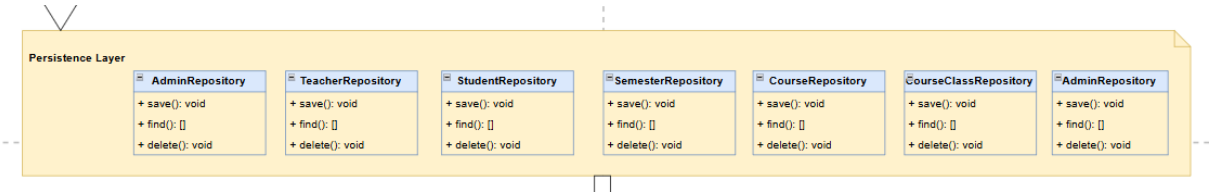
Hình 4: *Business Layer Of Layered Architecture*

Tiếp ngay sau tầng Presentation là tầng Business, đây là tầng chịu trách nhiệm cho việc đảm bảo Logic nghiệp vụ của hệ thống. Nó sẽ nhận dữ liệu đầu vào được truyền xuống từ tầng Presentation sau đó hiện thực Logic nghiệp vụ ứng với lời gọi từ Client, ngoài ra nó còn có thể gọi dữ liệu hay các function được cung cấp từ tầng Persistence để truy vấn hoặc cập nhập dữ liệu trong Database nếu cần. Ngoài ra để phần nào giải quyết việc khó bảo trì hay phát triển codebase do ảnh hưởng của kiến trúc Monolithic, các logic nghiệp vụ thường được tổ chức thành các Module có thể tái sử dụng ứng với từng nghiệp vụ khác nhau của hệ thống.

Trong trường hợp đó, việc tổ chức cũng như sửa chữa Logic nghiệp vụ của hệ thống sẽ không ảnh hưởng nhiều tới các Api gọi từ Client, bởi lẽ, các service đối với tầng Presentation như các hàm trừu tượng, đồng nghĩa với việc tầng này sẽ không quan tâm tới Logic được hiện thực như thế nào mà chỉ quan tâm kết quả trả về. Vì vậy việc sửa chữa hay bảo trì các logic nghiệp vụ sẽ không ảnh hưởng tới các Api mà hệ thống cung cấp.

Nhìn chung, việc chia tầng như thế này sẽ giúp nhóm có một cái nhìn trực quan hơn về luồng đi cũng như cấu trúc của dữ liệu đầu vào, đầu ra như thế nào để việc hiện thực báo cáo sẽ hiểu quả hơn.

4.2.3 Persistence Layer



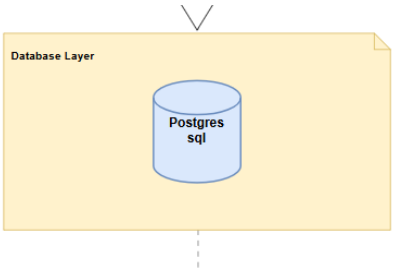
Hình 5: *Persistence Layer Of Layered Architecture*

Để xử lý các nhu cầu về việc truy vấn và cập nhập dữ liệu trong database thì Persistence Layer sẽ là nơi chứa các đối tượng trung gian thực hiện điều đó, các đối tượng này sẽ dảm nhiệm việc gửi các yêu cầu đến lớp Database để thực hiện các thao tác liên quan đến dữ liệu.

Trong hệ thống này, các đối tượng đó sẽ là các Interface Repository, chúng cung cấp các hàm trừu tường, giúp cho người phát triển tiện lợi hơn trong việc tự động tạo ra các câu truy vấn trực tiếp tới Database. Ngoài ra, khi sử dụng tầng này trong một số trường hợp, việc thay đổi nhà cung cấp dịch vụ lưu trữ dữ liệu cũng trở nên dễ dàng, ví dụ như giữa Mongo-DB và PostgresSql.

Tóm lại, đây sẽ là tầng chịu trách nhiệm cho việc tương tác trực tiếp với nơi lưu dữ liệu của hệ thống.

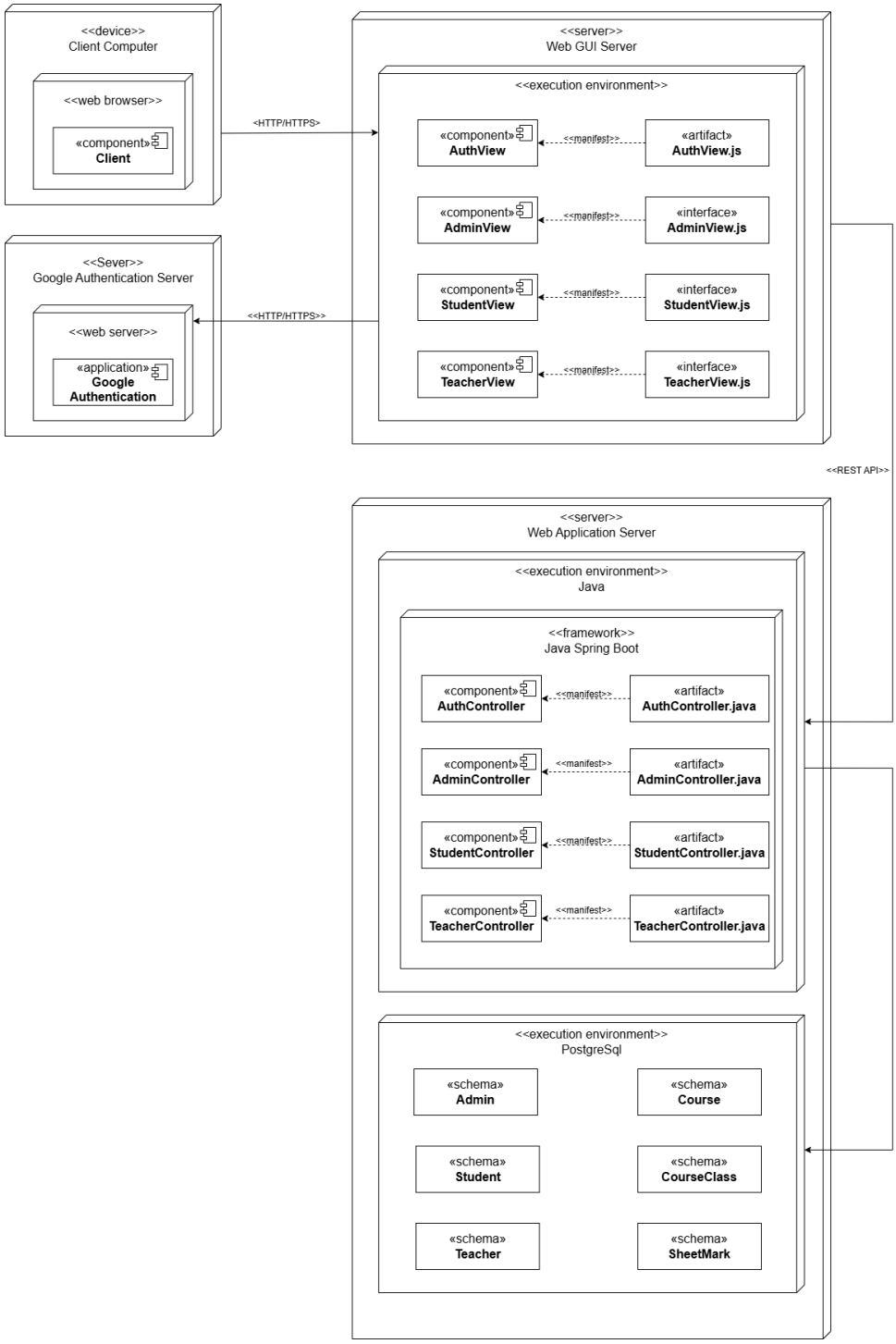
4.2.4 Data Layer



Hình 6: *Database Layer Of Layered Architecture*

Đây là tầng thấp nhất được hiện thực trong kiến trúc, cũng như đây là nơi lưu trữ dữ liệu cho toàn bộ Logic nghiệp vụ của hệ thống. Nhìn chung đây là cơ sở dữ liệu và các thành phần liên quan như hệ quản trị cơ sở dữ liệu, có nhiệm vụ quản lý cơ sở dữ liệu, thực hiện thao tác đọc và ghi dữ liệu và triển khai các truy vấn và lưu trữ dữ liệu theo cách được định nghĩa từ lớp Persistence.

4.3 Deployment Diagram



Hình 7: *Deployment Diagram*

Sau khi đọc yêu cầu của hệ thống đề ra, nhóm chúng em quyết định xây dựng bản vẽ Deployment Diagram cho hệ thống này. Tuy hệ thống cần hiện thực trong dự án này vẫn trong giai đoạn khởi đầu, nghiên cứu và tính chất kiến thức yêu cầu vẫn còn cơ bản, chúng em quyết định sẽ chia hệ thống thành 2 server riêng, một dành cho phía giao diện người dùng và phần còn lại dành cho các Logic nghiệp vụ được hiện thực trong hệ thống. Việc xây dựng mô hình này sẽ giúp cho hệ thống đảm bảo an ninh, sức chịu đựng khi có đông người sử dụng và khả năng mở rộng hệ thống.

4.4 Data Storage Approach

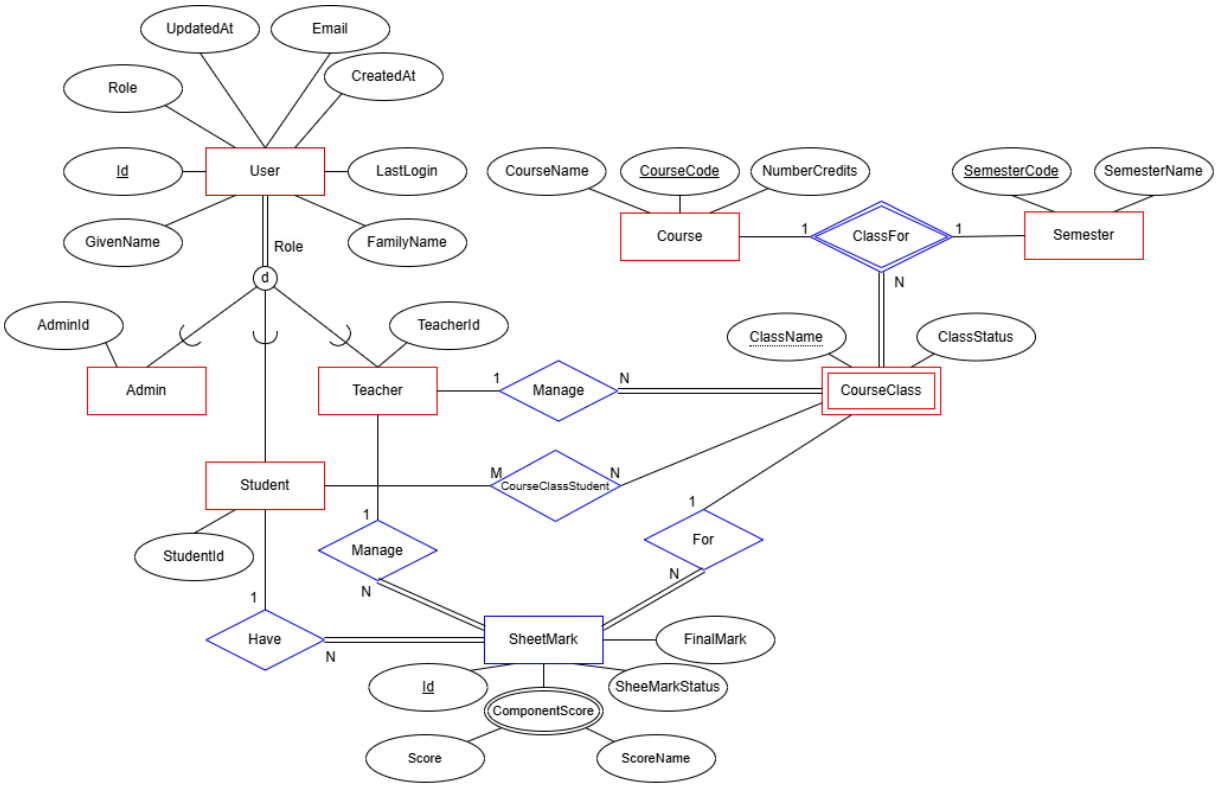
Đối với các yêu cầu của hệ thống Grade Portal, không khó để nhận ra, các hành động liên quan tới việc truy xuất, xử lý và cập nhập dữ liệu là rất quan trọng và cần sự chính xác cao. Và đó cũng chính là một trong những lí do kiến trúc phân tầng được chọn để hiện thực hệ thống, các thao tác liên quan tới việc xử lý dữ liệu hệ thống sẽ được tập trung ở tầng Persistence để tạo sự nhất quán và tránh sự phân tán cấu trúc không rõ ràng.

Ngoài ra, dữ liệu của hệ thống sẽ được lưu bằng cơ sở dữ liệu quan hệ cụ thể là sử dụng hệ quản trị cơ sở dữ liệu PostgreSql, có nghĩa là dữ liệu trong hệ thống sẽ được chia thành các thực thể (Entity) với cấu trúc các trường thuộc tính cùng các mối quan hệ được mô tả rõ ràng trong các Class được khai báo với Anotation @Table trong dự án.

Để đảm bảo hỗ trợ và cung cấp dữ liệu đầy đủ cho các Logic nghiệp vụ hệ thống, sau đây là các sơ đồ EERD, Relational Mapping hay class Diagram để chúng ta có cái nhìn chi tiết nhất về cách các thực thể được lưu trong trong Database và mối quan hệ giữa các thực thể đó.

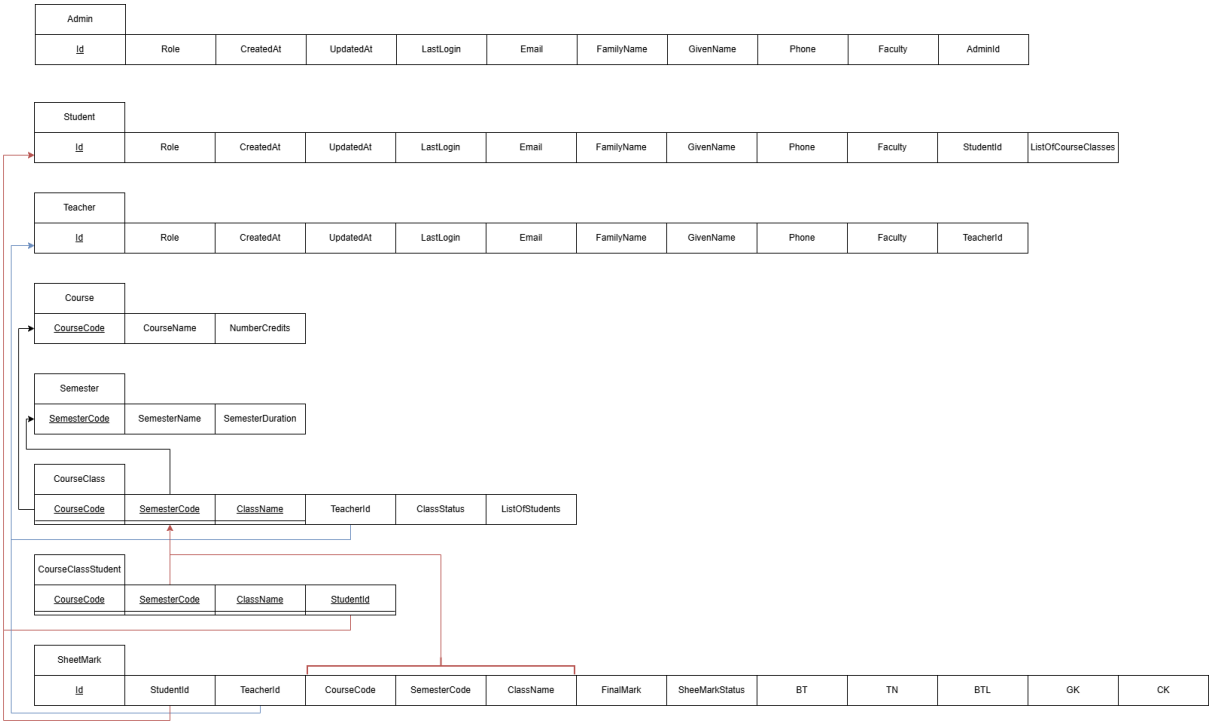
Trong phần này nhóm đã sử dụng **Draw.io** để hiện thực sơ đồ. Đường dẫn cụ thể tới sơ đồ được hiện thực nằm ở **đây**.

EERD Diagram



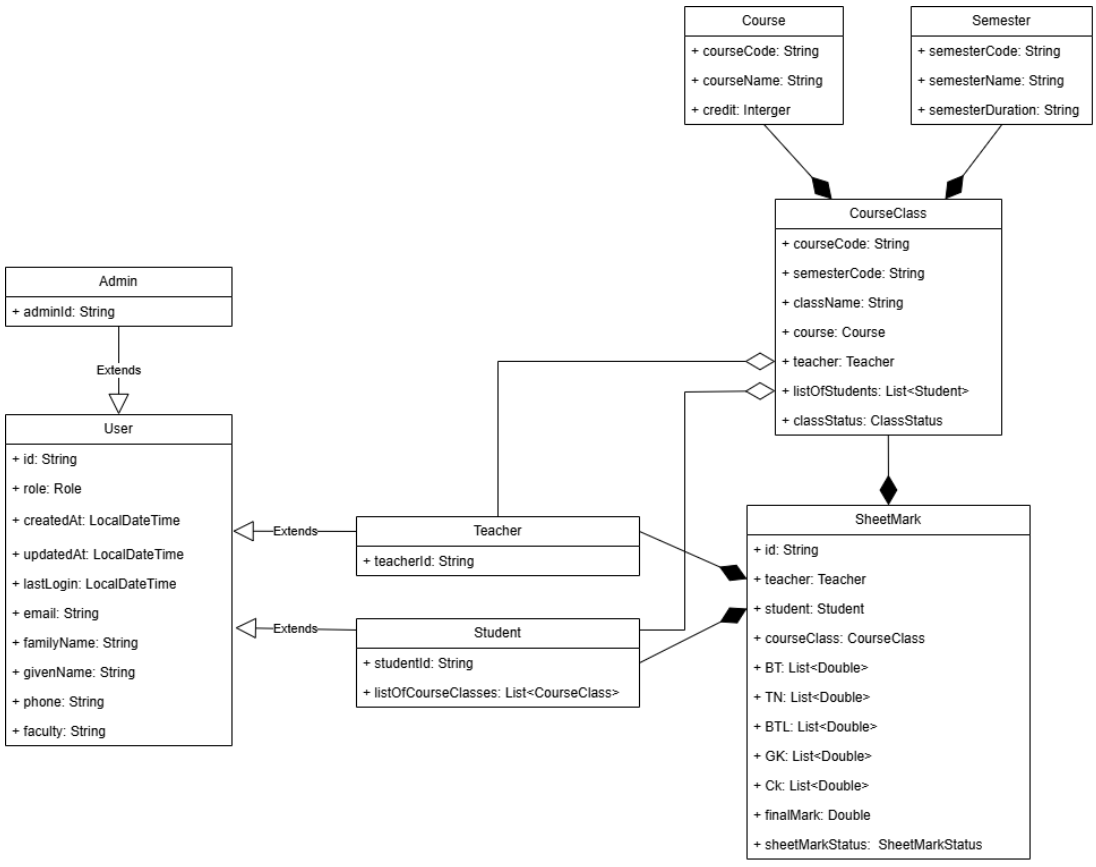
Hình 8: *EERD Diagram*

Relational Data Modal



Hình 9: *Relational Data Modal*

Class Diagram



Hình 10: *Class Diagram*