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

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC

HỆ THỐNG HỖ TRỢ THỰC HÀNH LẬP TRÌNH VÀ GỢI Ý LỘ TRÌNH THỰC HÀNH

NGÀNH: KHOA HỌC MÁY TÍNH

Hội đồng	:	Khoa học Máy tính 04	
GVHD	:	ThS. Trần Ngọc Bảo Duy	
		PGS. TS. Huỳnh Tường Nguyên	
GVPB	:	TS. Nguyễn Hứa Phùng	
SV thực hiện	:	Vũ Văn Tiến	1713497
		Trần Huy	1711552
		Nguyễn Việt Hoàng	1710098

TP. Hồ Chí Minh, tháng 08 năm 2021

ĐẠI HỌC QUỐC GIA TP.HCM

TRƯỜNG ĐẠI HỌC BÁCH KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

KHOA: KH&KT MÁY TÍNH
BỘ MÔN: KHOA HỌC MÁY TÍNH

NHIỆM VỤ LUẬN ÁN TỐT NGHIỆP

Chú ý: Sinh viên phải dán tờ này vào trang nhất của bản thuyết trình

HỌ VÀ TÊN: Nguyễn Việt Hoàng — MSSV: 1710098
NGÀNH: Khoa học Máy tính — LỚP: MTKH17

HỌ VÀ TÊN: Trần Huy — MSSV: 1711552
NGÀNH: Khoa học Máy tính — LỚP: MTKH17

HỌ VÀ TÊN: Vũ Văn Tiến — MSSV: 1713497
NGÀNH: Khoa học Máy tính — LỚP: MTKH17

1. Đầu đề luận án: Hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành

2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):

- Tìm hiểu và đánh giá các nền tảng chấm bài tập lập trình đã hiện thực.
- Triển khai tích hợp server chấm vào trong hệ thống chấm lập trình.
- Tìm hiểu các phương pháp đánh giá năng lực người học lập trình, phương pháp gợi ý lộ trình học thực hành và tích hợp vào hệ thống chấm bài lập trình
- Kiểm thử và đánh giá hệ thống.

3. Ngày giao nhiệm vụ luận án: 15 / 01 / 2021

4. Ngày hoàn thành nhiệm vụ: 31 / 07 / 2021

5. Họ tên giảng viên hướng dẫn: ThS. Trần Ngọc Bảo Duy, PGS. TS. Huỳnh Tường Nguyên

Nội dung và yêu cầu LVTN đã được thông qua Bộ môn.

Ngày 11 tháng 08 năm 2021

CHỦ NHIỆM BỘ MÔN
(Ký và ghi rõ họ tên)

PGS. TS. Huỳnh Tường Nguyên

GIẢNG VIÊN HƯỚNG DẪN CHÍNH
(Ký và ghi rõ họ tên)

ThS. TRẦN NGỌC BẢO DUY

PHẦN DÀNH CHO KHOA, BỘ MÔN:

Người duyệt (chấm sơ bộ):

Đơn vị:

Ngày bảo vệ:

Điểm tổng kết:

Nơi lưu trữ luận án:

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 10 tháng 08 năm 2021

PHIẾU CHẤM BẢO VỆ LVTN

(Dành cho người hướng dẫn)

11. Họ và tên SV: Vũ Văn Tiến
MSSV: 1713497 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Trần Huy
MSSV: 1711552 Ngành (chuyên ngành): Khoa học Máy tính
Họ và tên SV: Nguyễn Việt Hoàng
MSSV: 1710098 Ngành (chuyên ngành): Khoa học Máy tính

2. Đề tài: Hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành

3. Họ tên người hướng dẫn: ThS. Trần Ngọc Bảo Duy, PGS. TS. Huỳnh Tường Nguyên

4. Tổng quát về bản thuyết minh:
Số trang: 136 Số chương: 6
Số bảng số liệu: 47 Số hình vẽ: 65
Số tài liệu tham khảo: Phần mềm tính toán:
Hiện vật (sản phẩm)

5. Tổng quát về các bản vẽ:
- Số bản vẽ: Bản A1: Bản A2: Khổ khác:
- Số bản vẽ vẽ tay Số bản vẽ trên máy tính:

6. Những ưu điểm chính của LVTN:
- Nhóm sinh viên đã trình bày được các ưu, khuyết điểm của các giải pháp hiện tại để rút trích ra các yếu tố cần phải cải thiện trong các nền tảng chấm bài lập trình.
- Nhóm đã tích hợp được Jobe server lên một nền tảng do nhóm đề xuất và hiện thực phần front-end.
- Nhóm sinh viên đã nghiên cứu các giải pháp đánh giá năng lực lập trình của người học dưới góc độ giáo dục và hiện thực các giải pháp đánh giá về mặt công nghệ, từ đó xây dựng quy trình gợi ý lộ trình cho người học.
- Nhóm đã kiểm thử và đánh giá hiệu suất của hệ thống trên một tập sinh viên thực.
- Nhóm đã công bố được một công trình khoa học tại hội nghị SCSE 2021.

7. Những thiếu sót chính của LVTN:
- Phương pháp đánh giá năng lực người học còn đơn giản.

8. Đề nghị: Được bảo vệ ☒ Bổ sung thêm để bảo vệ □ Không được bảo vệ □

9. 0 câu hỏi SV phải trả lời trước Hội đồng:

10. Đánh giá chung (bằng chữ: giỏi, khá, TB): Giỏi Điểm : 10 /10

Ký tên (ghi rõ họ tên)

ThS. Trần Ngọc Bảo Duy

TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KH & KT MÁY TÍNH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Ngày 10 tháng 8 năm 2021

PHIẾU CHẤM BẢO VỆ LVTN

(Dành cho người phản biện)

1. Họ và tên SV: Vũ Văn Tiến
   MSSV: 1713497 Ngành (chuyên ngành): Khoa học máy tính
   Họ và tên SV: Trần Huy
   MSSV: 1711552 Ngành (chuyên ngành): Khoa học máy tính
   Họ và tên SV: Nguyễn Việt Hoàng
   MSSV: 1710098 Ngành (chuyên ngành): Khoa học máy tính
2. Đề tài: Hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành
3. Họ tên người phản biện: TS. Nguyễn Hứa Phùng
4. Tổng quát về bản thuyết minh:
   Số trang: 136 Số chương: 6
   Số bảng số liệu: 47 Số hình vẽ: 65
   Số tài liệu tham khảo: Phần mềm tính toán:
   Hiện vật (sản phẩm)
5. Tổng quát về các bản vẽ:
   - Số bản vẽ: Bản A1: Bản A2: Khổ khác:
   - Số bản vẽ vẽ tay Số bản vẽ trên máy tính:
6. Những ưu điểm chính của LVTN:

Nhóm sinh viên thực hiện đề tài đã cải tiến hệ thống hỗ trợ thực hành hiện có để cải thiện tính hiệu quả của chức năng chấm bài (thông qua sử dụng Jobe thay vì Docker) và thực hiện chức năng xây dựng lộ trình giao bài thực hành tự động cho người học với độ khó tăng dần. Sau khi được góp ý qua buổi phản biện, nhóm sinh viên đã cải thiện luận văn tốt nghiệp, bổ sung một số nội dung liên quan đến qui trình phát triển phần mềm (phân tích yêu cầu đề tài)

Sinh viên cho biết một phần nội dung của đề tài được báo cáo tại hội nghị SCSE 2021 nhưng không đính kèm bài báo với luận văn.

7. Những thiếu sót chính của LVTN:

Luận văn tốt nghiệp không thể hiện nhóm sinh viên có thực hiện kiểm thử tính đúng đắn của các chức năng do nhóm hiện thực.

8. Đề nghị: Được bảo vệ ☒ Bổ sung thêm để bảo vệ □ Không được bảo vệ □
9. 3 câu hỏi SV phải trả lời trước Hội đồng:

10. Đánh giá chung (bằng chữ: giỏi, khá, TB):Giỏi Điểm : 9.8 /10

Ký tên (ghi rõ họ tên)

TS. Nguyễn Hứa Phùng

Lời cam đoan

Chúng tôi xin cam đoan tất cả nội dung được trình bày trong Luận văn Tốt nghiệp này, ngoại trừ những phần đã được chú thích, trích nguồn rõ ràng trong phần tài liệu tham khảo, đều là do chính bản thân nhóm thực hiện.

Chúng tôi xin chịu trách nhiệm hoàn toàn nếu có bất kì sự gian lận nào về nội dung Luận văn Tốt nghiệp của nhóm.

Lời cảm ơn

Con đường đi nào rồi thì cũng đến một ngã rẽ nào đó. Chặng đường Đại học cũng như vậy, cũng sẽ đến lúc chúng ta dừng lại và chuyển hướng sang với chặng đường mới hơn. Để kết thúc chặng đường này thì cần chứng minh được chúng ta đã vượt qua nó như thế nào, những gì còn lưu giữ lại ở phía cuối con đường này và Luận văn Tốt nghiệp chính là minh chứng đó.

Vì lẽ đó mà chúng em xin phép được cảm ơn Ths. Trần Ngọc Bảo Duy và PGS. TS. Huỳnh Tường Nguyên, hai người thầy đã hỗ trợ và giúp đỡ chúng em trong quá trình làm Luận văn Tốt nghiệp này. Không chỉ trao những lời khuyên rất có giá trị mà còn khơi gợi cho chúng em những hướng đi rất mở, giúp chúng em có thể tự do trong việc chọn lựa những cách giải quyết vấn đề và cuối cùng là cả những lời nhận xét, góp ý để chúng em hoàn thiện hơn trong việc đạt được những mục tiêu của vấn đề đã đề ra.

Chúng em cũng xin phép được cảm ơn các thầy, các cô giảng dạy trong bộ môn Khoa học Máy tính nói riêng cũng như Khoa Khoa học và Kỹ thuật Máy tính nói chung, đã đồng hành cùng chúng em trong suốt chặng đường Đại học gần bốn năm qua, đã truyền đạt cho chúng em rất nhiều kiến thức hữu ích, giúp chúng em bước vào chặng đường kế tiếp với một tâm thế vững vàng nhất.

Cuối cùng là xin cảm ơn trường Đại học Bách Khoa - Đại học Quốc gia Thành phố Hồ Chí Minh đã tạo ra một môi trường giáo dục chất lượng, giúp chúng em có cơ hội được gặp những người thầy, người cô đầy tâm huyết trong việc giảng day, được gặp những người bạn rất biết quan tâm và hỗ trợ lẫn nhau trong quá trình học suốt những năm vừa qua.

Một lần nữa, xin cảm ơn vì tất cả mọi thứ.

Tóm tắt

Học tập là một công cuộc suốt đời, nhằm mang lại kiến thức cho người học thông qua sự truyền đạt kiến thực từ phía người dạy. Tuy nhiên, trong thời gian gần đây, với sự phát triển vượt bậc của công nghệ thông tin, quá trình truyền đạt trên cũng đã có nhiều đổi mới để đáp ứng nhu cầu học tập ngày càng lớn từ phía người học. Thông qua việc ứng dụng những hệ thống hỗ trợ học tập trong các trường học, người dạy hoàn toàn có khả năng dạy học cho một số lượng lớn người học và người học cũng ngày càng nâng cao được tinh thần tự học của mình.

Nhưng nhóm nhận thấy những hệ thống hỗ trợ học tập này vẫn chưa đáp ứng đủ tốt cho những vấn đề ngách như hỗ trợ cho ngành học công nghệ thông tin vẫn còn hạn chế - ngành học hiện đang đóng vai trò quan trọng trong cuộc cách mạng công nghệ lần thứ tư. Trong đó, kỹ năng cần thiết nhất cho ngành học này là kỹ năng lập trình. Do vậy, đề tài của Luận văn Tốt nghiệp này sẽ hướng đến việc xây dựng một hệ thống đáp ứng cho kỹ năng trên.

Qua các phần của Luận văn Tốt nghiệp này, nhóm sẽ trình bày những kiến thức nghiên cứu, những phân tích về chủ đề hệ thống hỗ trợ thực hành lập trình và các khía cạnh liên quan như chấm bài tự động, ngân hàng câu hỏi, phân cụm độ khó câu hỏi cũng như gợi ý câu hỏi. Từ đó hiện thực được một hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành. Cuối cùng là đánh giá hệ thống đã hiện thực để xem xét tính khả thi khi ứng dụng vào thực tế.

Mục lục

Danh sách bảng

Danh sách hình vẽ

Danh mục viết tắt

AGS Auto Grading System. 13–15, 45, 54, 55, 57, 81, 88, 105

API Application Programming Interface. 20, 23, 48–50

BKEL Bách Khoa e-Learning. 11

CNTT Công nghệ thông tin. 17, 24

CSDL Cơ sở dữ liệu. 21, 24, 32, 55, 72, 76, 77, 88, 89, 101

CSS Cascading Style Sheets. 18, 19

CTDL&GT Cấu trúc dữ liệu và Giải thuật. 13, 14, 57, 59, 81–85

HTML HyperText Markup Language. 18, 19

KTLT Kỹ thuật lập trình. 13, 57, 59, 81–85, 112

LVTN Luận văn Tốt nghiệp. 6, 117

REST Representational state transfer. 23, 50

UI User Interface. 18, 19

ĐCLV Đề cương Luận văn. 117

Chương 1

Tổng quan

Trong chương này, nhóm chúng tôi sẽ trình bày khái quát về đề tài, bao gồm những lý do để chọn làm đề tài này, cũng như mục tiêu của đề tài này và phạm vi ứng dụng trong thực tế.

1.1 Lý do chọn đề tài

Hiện nay, giáo dục không còn đơn thuần là quá trình giảng dạy và học tập ở trường lớp. Nhờ vào sự phát triển mạnh mẽ của Internet, việc dạy và học có thể được thực hiện ở khắp mọi nơi. Người dạy sử dụng Internet để chia sẻ tài liệu học tập cho người học. Người học có thể tìm kiếm các tài liệu học tập bổ sung trên Internet và thông qua đó cũng dễ dàng tương tác, trao đổi với người dạy hơn trước. Không dừng lại ở đấy, các nền tảng học trực tuyến lớn như edX[1], Coursera[2], Udacity[3],... được ra đời và ngày càng phát triển, trở thành một nền tảng cung cấp các khóa học trực tuyến. Những khóa trực tuyến trên các nền tảng này bao gồm các video bài giảng, tài liệu học tập, các bài kiểm tra trắc nghiệm, bài tập lớn,... nhằm kiểm tra mức độ thông hiểu, khả năng vận dụng kiến thức,... của người học. Từ đó, một kiểu môi trường học tập mới được hình thành, người học dễ dàng tham gia các khóa học trực tuyến, chủ động trong việc liên hệ, tương tác với người dạy để trao đổi và bàn luận về những chủ đề, những kiến thức trong môn học nhằm hoàn thành tốt nó.

Các nền tảng học tập trực tuyến giúp các trường Đại học thích ứng được với môi trường mới như trên, với những tùy chỉnh riêng để phù hợp với đặc thù của từng trường, cung cấp sự linh hoạt cần thiết đến người dạy lẫn người học mà không thay thế hoàn toàn hình thức học tập truyền thống. Hơn nữa, vì tính trực tuyến của những nền tảng này, chúng khuyến khích người học phải luôn tự giác và nỗ lực trong việc học. Học viên có thể tự tìm hiểu xem trước nội dung bài giảng tại nhà, sắp xếp thời gian làm các bài kiểm tra và tận dụng thời gian trên lớp để hỏi thầy cô về các phần chưa rõ. Từ đó, khuyến khích người học khai phá được giới hạn bản thân và

[1]edX - https://www.edx.org/
[2]Coursera - https://www.coursera.org/
[3]Udacity - https://www.udacity.com/

rèn luyện tính tự giác học tập, cũng là khuyến khích cho tinh thần học tập suốt đời, kể cả sau khi rời khỏi ngưỡng cửa trường lớp.

Bên cạnh các hoạt động học tập lý thuyết, các môn học của ngành Công nghệ thông tin (CNTT) còn có một đặc thù riêng là người học cần được thực hành và rèn luyện kỹ năng lập trình. Người học thường được yêu cầu hiện thực toàn bộ hoặc một phần của chương trình máy tính nhằm vận dụng kiến thức lý thuyết đã học. Tuy nhiên, sau khi hoàn thành chương trình, người học vẫn chưa có khả năng xác định tính đúng đắn của chương trình, hoặc bài làm của người học có đáp ứng được các yêu cầu đặt ra hay không? Thông thường trong giờ thực hành, người dạy sẽ hướng dẫn làm bài, xem xét tính đúng đắn của các bài làm và hỗ trợ sửa lỗi giúp học viên. Tuy nhiên, với số lượng lớn các học viên tham gia vào lớp thì việc kiểm tra tính đúng đắn và sửa lỗi tốn rất nhiều thời gian và công sức của người dạy. Bên cạnh đó, người học sau khi nộp bài phải chờ một khoảng thời gian để nhận lại phản hồi từ người dạy. Điều này dẫn đến người học có thể không còn hứng thú với bài làm đó nữa, hoặc chí ít, người học cũng cần đọc lại bài làm trước đó, sau đó chỉnh sửa chương trình dựa trên những phản hồi từ người dạy. Sự chờ đợi phản hồi này có thể khiến người học cảm thấy chán nản hoặc không còn muốn thay đổi theo phản hồi. Khả năng lập trình cũng vì thế mà không được cải thiện.

Một thách thức khác ở thời điểm hiện tại là thế giới đang phải trải qua đại dịch COVID-19 đầy nguy hiểm với sức lây lan mạnh mẽ của chủng loại virus này. Để giảm thiểu nguy cơ dịch bệnh lây lan, yêu cầu về giãn cách xã hội, không tập trung nơi đông người đã được đề ra, vô hình trung làm ảnh hưởng đến quá trình học tập tập trung tại các trường học. Thời gian trống khi không phải đến trường đòi hỏi sinh viên cần phải tự học nhiều hơn. Đối với kỹ năng lập trình, người học cần có một nơi để luyện tập với một lượng bài tập đủ tốt và có tính khích lệ người học. Tuy nhiên, điểm hạn chế của các hệ thống hỗ trợ thực hành là thiếu cơ chế khuyến khích học tập phù hợp. Người học thông thường cần được hướng dẫn làm các bài tập từ dễ đến khó mà độ khó của bài tập này cần được nhìn nhận từ khả năng của người học. Bởi vì khả năng của mỗi người học là khác nhau, tri thức tích lũy của mỗi người cũng khác nhau trong thời đại công nghiệp 4.0, thời đại tiếp cận thông tin dễ dàng. Sự khác nhau về tri thức và khả năng có thể dẫn đến một câu hỏi có thể là dễ đối với người này nhưng lại là khó đối với người khác. Việc hoàn thành một lượng lớn bài tập không phù hợp với khả năng này đôi khi gây chán nản cho người học.

Trong việc dạy và học, các câu hỏi là một thành phần không thể thiếu giúp người học có thể hiểu được lý thuyết. Người dạy phải liên tục nghĩ ra những câu hỏi tốt và mới lạ để người học hiểu sâu hơn và vận dụng được vào thực tế. Hơn thế, người dạy còn phải tạo ra rất nhiều câu hỏi để tạo nên một bài kiểm tra phù hợp để đánh giá năng lực của người học. Về lâu dài, số lượng câu hỏi mà người dạy sở hữu sẽ trở nên rất lớn. Điều này dẫn đến nhu cầu không những lưu trữ mà còn là tổ chức, sắp xếp một lượng lớn câu hỏi một cách phù hợp từ phía người dạy. Từ đó, người dạy có thể tìm lại nhanh các câu hỏi mình đã từng tạo ra và sử dụng.

Từ những vấn đề nêu trên, nhóm tác giả nhận thấy nhiều công việc cần làm trong việc nghiên

cứu, đề xuất giải pháp cho một hệ thống hỗ trợ thực hành có thể giải quyết các vấn đề trên. Đó là lý do nhóm lựa chọn *Hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành* làm đề tài cho Luận văn tốt nghiệp.

1.2 Phân tích yêu cầu người dùng

Trước khi xây dựng một hệ thống hỗ trợ thực hành lập trình, nhóm tiến hành khảo sát các hệ thống hỗ trợ lập trình khác thông qua một số tiêu chí. Kết quả khảo sát sẽ giúp nhóm xác định các nhu cầu từ phía người dùng đối với một hệ thống hỗ trợ lập trình. Ở đây, nhóm chọn 4 hệ thống để khảo sát bao gồm:

- HackerRank: Nền tảng hỗ trợ lập trình trực tuyến lâu năm với đa dạng lĩnh vực trong ngành Khoa học Máy tính.
- CodeLearn: Sản phẩm của FPT Software hỗ trợ các lập trình viên Việt Nam tiếp cận với lập trình thông qua các bài tập và cuộc thi.
- CodeRunner: Một plugin dạng câu hỏi lập trình được sử dụng trong nền tảng hệ thống quản lý học tập nổi tiếng Moodle.
- AGS: Hệ thống hỗ trợ thực hành cho các môn học lập trình đã từng được triển khai tại Khoa KH&KT Máy tính trường ĐH Bách Khoa.

Bảng 1.1: *Bảng so sánh 4 hệ thống*

	HackerRank	**CodeLearn**	**Moodle sử dụng CodeRunner**	**AGS**
Phạm vi sử dụng	Quốc tế (>7,000,000 người dùng)	Trong nước (>60,000 người dùng)	Quốc tế (>2000 sites), BKeL	Khoa Máy tính Trường ĐHBK (>2000 người dùng)
Khả năng tiếp cận mã nguồn để phát triển hệ thống	Không	Không	Không có mã nguồn CodeRunner, có mã nguồn con chấm Jobe, con chấm có khả năng chạy liên tục	Có, sử dụng con chấm Docker cần bật-tắt mỗi lần chấm bài

Đối với học viên lập trình	- Giao diện làm bài thân thiện, đẹp mắt. - Luồng chấm bài bất đồng bộ	- Giao diện làm bài thân thiện, đẹp mắt. - Người học chờ đến khi câu hỏi được chấm xong	- Giao diện chưa được thân thiện với người dùng - Người học chờ đến khi câu hỏi được chấm xong	- Giao diện làm bài thân thiện, đẹp mắt. - Luồng chấm bài bất đồng bộ
Đối với người tạo câu hỏi	- Hỗ trợ nhiều ngôn ngữ hiện thực cho một câu hỏi. - Hỗ trợ nhiều kiểu câu hỏi khác (trắc nghiệm, điền từ,...) - Không có các chức năng quản lý lớp học - Có ngân hàng câu hỏi - Chọn điểm và độ khó cho từng testcase	- Hỗ trợ nhiều ngôn ngữ hiện thực cho một câu hỏi. - Hỗ trợ nhiều kiểu câu hỏi khác (trắc nghiệm, điền từ,...) - Không có các chức năng quản lý lớp học - Không khảo sát được - Không khảo sát được	- Hỗ trợ nhiều ngôn ngữ lúc cài đặt câu hỏi, nhưng một câu hỏi chỉ được trả lời bằng một ngôn ngữ - Hỗ trợ nhiều kiểu câu hỏi khác (trắc nghiệm, điền từ,...) - Có các chức năng quản lý lớp học - Có ngân hàng câu hỏi - Chọn điểm cho từng testcase	- Chỉ có thể chọn một ngôn ngữ là C++ - Chỉ hỗ trợ câu hỏi điền vào chương trình có sẵn - Có các chức năng quản lý lớp học - Không có ngân hàng câu hỏi - Các testcase có điểm và độ khó như nhau

Sau khi tiến hành khảo sát các hệ thống trên, nhóm nhận thấy cần phải xây dựng một hệ thống hỗ trợ thực hành lập trình đạt được các tiêu chí sau:

- Đối với học viên lập trình:
  - Được cung cấp giao diện làm bài thân thiện, đẹp mắt.
  - Được chấm bài bất đồng bộ, không phải chờ cho câu hỏi được chấm xong.
- Đối với người người tạo câu hỏi:
  - Được hỗ trợ nhiều ngôn ngữ để cài đặt cho câu hỏi.
  - Được cung cấp nhiều kiểu câu hỏi khác nhau.

- Có ngân hàng câu hỏi để lưu trữ câu hỏi.
- Có thể cài đặt độ khó và điểm số cho từng testcase.

1.3 Xây dựng đề tài

Trong đề tài này, nhóm xây dựng một hệ thống hỗ trợ lập trình cho công tác giảng dạy thực hành. Do vậy, hai đối tượng chính tham gia vào hệ thống là người học và người dạy. Các yêu cầu của học viên lập trình cũng giống như các yêu cầu đối với người học. Đồng thời, các yêu cầu của người tạo câu hỏi cũng sẽ tương ứng với người dạy, tuy nhiên, còn cần có thêm nhu cầu về việc quản lý lớp học.

Bên cạnh đó, nhóm nhận thấy rằng, các câu hỏi trong 2 hệ thống là HackerRank và CodeLearn có gắn thẻ xác định độ khó của câu hỏi. Nhờ đó, học viên lập trình có thể lọc các câu hỏi theo độ khó và tự lựa chọn câu hỏi để làm. Nhóm cũng nhận thấy, trong việc giảng dạy, người dạy cũng cần quan tâm đến độ khó câu hỏi để gán bài tập được phù hợp cho khả năng chung của lớp học. Do vậy, nhóm đặt thêm các câu hỏi như sau:

- Độ khó này đến từ đâu và liệu có phù hợp với người học hay chưa?
- Có cách nào để người học được gợi ý các câu hỏi để làm một cách tự động và phù hợp với cá nhân người học không?

Từ những ý trên, nhóm đề xuất xây dựng một hệ thống hỗ trợ lập trình với các yêu cầu được nhóm sắp xếp theo tính cấp bách (giảm dần) gồm:

1. Hệ thống kiểm tra tính đúng đắn của bài nộp, chấm điểm bài nộp một cách tự động và đưa ra phản hồi đến người học.
2. Hệ thống cung cấp giao diện làm bài thân thiện, đẹp mắt.
3. Hệ thống chấm bài bất đồng bộ, người học không phải chờ cho câu hỏi được chấm xong.
4. Hệ thống cho phép quản lý người học theo nhóm và gán bài tập cho họ giống như các lớp học thực hành.
5. Hệ thống có thể đáp ứng được nhiều người dùng khi số lượng người dùng tăng lên. Cụ thể, khi số lượng người dùng tăng lên, hoặc bài chấm có chi phí tính toán lớn, hệ thống vẫn có thể đáp ứng được.
6. Hệ thống có thể mở rộng hiệu năng chấm bài đơn giản bằng cách tăng số lượng con chấm để tăng khả năng đáp ứng đến người dùng.

7. Hệ thống cho phép lưu trữ câu hỏi tách biệt thành một kho lưu trữ. Hệ thống cần phải đáp ứng được nhu cầu lưu trữ số lượng lớn câu hỏi. Đồng thời, hệ thống phải cung cấp một cách quản lý để có thể tìm lại được các câu hỏi một cách tiện lợi.

8. Hệ thống giúp người dạy xem xét đánh giá lại việc soạn và gán câu hỏi dựa theo độ khó. Cụ thể, dựa trên kết quả làm bài của người học, hệ thống có thể tính toán và phân loại câu hỏi theo độ khó.

9. Hệ thống cung cấp một môi trường tự luyện tập với các câu hỏi được gợi ý phù hợp với từng người học.

10. Hệ thống cho phép tạo người dạy câu hỏi với các cấu hình: gán độ khó cho câu hỏi, cài đặt độ khó, điểm cho từng testcases, v.v..

11. Hệ thống hỗ trợ nhiều ngôn ngữ cho việc trả lời một câu hỏi lập trình.

12. Ngoài việc kiểm tra tính đúng đắn của chương trình, hệ thống cung cấp các dạng câu hỏi khác để kiểm tra lý thuyết người học như trắc nghiệm, điền từ, câu hỏi trả lời tự luận,...

Trong thời gian và khuôn khổ của LVTN này, nhóm tác giả lựa chọn và hiện thực các yêu cầu từ 1 đến 9.

1.4 Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một hệ thống hỗ trợ thực hành cho phép đánh giá người học một cách tự động. Trong hệ thống này, người dạy sẽ tạo các bài thực hành, các câu hỏi và testcases thể hiện yêu cầu cần đạt cho câu hỏi đó. Người học được cung cấp một giao diện trực quan để nhận các yêu cầu đặc tả của câu hỏi và hoàn thành chương trình. Hệ thống cung cấp các khả năng như kiểm tra tính đúng đắn của bài nộp, chấm điểm bài nộp một cách tự động và đưa ra phản hồi cho sinh viên. Sinh viên nhận kết quả phản hồi và tự kiểm tra bài làm đã đạt được các yêu cầu đề ra hay không. Dựa và kết quả phản hồi này, sinh viên tự tìm lỗi trong chương trình của mình và sửa lỗi, sau đó tiếp tục nộp bài. Việc nhận phản hồi nhanh chóng cũng giúp sinh viên dễ nắm bắt và tập trung vào bài tập đang làm, dễ tìm lỗi và sửa lỗi.

Thêm vào đó, hệ thống cung cấp một môi trường để người học vào luyện tập lập trình. Trong môi trường thực hành này, hệ thống sẽ hỗ trợ gợi ý các câu hỏi cho người học với độ khó tăng dần. Sau khi người học nộp bài, hệ thống sẽ dựa vào kết quả làm bài này cùng các kết quả trước để lựa chọn một câu hỏi mới với độ khó phù hợp cho người học làm. Cách làm này không những giảm đi cảm giác chán nản khi làm bài tập lập trình mà còn khích lệ người học luyện tập kỹ năng lập trình thêm. Người học sẽ dần dần cải thiện được kỹ năng lập trình theo hướng thích nghi tăng dần thông qua luyện tập các câu hỏi ở độ khó phù hợp hiện tại đến khi thuần thục và tiếp tục với một độ khó cao hơn.

Bên cạnh việc hỗ trợ cho người học luyện tập, hệ thống cũng sẽ hỗ trợ cho người dạy trong việc soạn câu hỏi, giao bài tập. Trong đó, để có các bài luyện tập, thành phần không thể thiếu chính là câu hỏi. Về mặt lâu dài, lượng câu hỏi người dạy tạo ra sẽ ngày càng lớn và dần khó theo dõi. Vì vậy, hệ thống sẽ cung cấp cho người dạy một nơi để lưu trữ các câu hỏi cũng như các cơ chế quản lý chúng. Nơi lưu trữ các bài tập này sẽ là ngân hàng câu hỏi được thiết kế với định hướng giúp người dạy sắp xếp các câu hỏi một cách có hệ thống, cung cấp khả năng chia sẻ câu hỏi giữa các người dạy.

1.5 Phạm vi đề tài

Hệ thống sẽ cung cấp một môi trường hỗ trợ giảng dạy thực hành cho giảng viên và sinh viên trong Khoa Khoa học và Kỹ thuật Máy tính. Bao gồm 3 hoạt động chính là:

- Giảng dạy: người dạy soạn câu hỏi, lưu trữ và quản lý câu hỏi.
- Thực hành: người học làm bài, được chấm bài và trả kết quả làm bài tự động.
- Luyện tập: người học được hệ thống gợi ý các câu hỏi từ dễ đến khó để luyện tập.

Bên cạnh đó, dạng câu hỏi mà hệ thống hỗ trợ là những câu hỏi dạng điền vào mã nguồn được thiết lập sẵn, sau đó biên dịch và so sánh với kết quả mong đợi từ người ra đề. Các môn học mà hệ thống hướng đến hỗ trợ là các môn học lập trình cơ bản như: Nhập môn điện toán, Nhập môn lập trình, Kỹ thuật lập trình, Cấu trúc dữ liệu và giải thuật.

1.6 Cấu trúc đề tài

Luận văn Tốt nghiệp này sẽ bao gồm sáu (6) chương, bao gồm cả chương này, cụ thể như sau:

1. **Chương 1: Tổng quan**
   Chương này mô tả chung về hiện trạng của quá trình học tập lập trình và từ đó nhận ra lý do vì sao phải xây dựng hệ thống hỗ trợ thực hành lập trình và gợi ý lộ trình thực hành. Sau đó, nhóm chúng tôi cũng thiết lập những mục tiêu cần hướng đến khi xây dựng hệ thống và trình bày phạm vi áp dụng của hệ thống trong thực tế.

2. **Chương 2: Kiến thức nền tảng**
   Chương này trình bày các tìm hiểu của nhóm để hoàn thành Luận văn. Các tìm hiểu này bao gồm các giải pháp công nghệ thông tin hiện có, các bài báo, nghiên cứu khoa học và các công nghệ nhóm tìm hiểu để sử dụng. Sau khi xem xét và tìm hiểu, nhóm dùng các kiến thức này để đề xuất và hiện thực các giải pháp cho hệ thống hỗ trợ thực hành lập trình.

3. **Chương 3: Thiết kế hệ thống**
   Chương này cung cấp cho người đọc một cái nhìn bao quát về hệ thống thông qua việc phân tích yêu cầu hệ thống, thiết kế cơ sở dữ liệu và sự giao tiếp giữa các thành phần trong hệ thống. Chương này sẽ là nền tảng để nhóm trình bày các giải pháp một cách cụ thể hơn.

4. **Chương 4: Thiết kế và hiện thực các giải pháp**
   Chương này sẽ mô tả chi tiết quá trình thay đổi thành phần chấm bài của hệ thống, cũng như quá trình hiện thực hai tính năng mới là ngân hàng câu hỏi cho người dạy và gợi ý lộ trình thực hành cho người học.

5. **Chương 5: Thực nghiệm và đánh giá hệ thống**
   Chương này trình bày các phần về triển khai hệ thống trên các máy chủ và một số đánh giá về hệ thống. Việc đánh giá này bao gồm đánh giá phần chấm bài của hệ thống và đánh giá từ phía người dùng sau khi trải nghiệm trên hệ thống được triển khai.

6. **Chương 6: Tổng kết**
   Chương cuối cùng sẽ đưa ra những kết quả mà hệ thống đã đạt được sau khi hiện thực. Đồng thời, những tồn đọng cũng được đưa ra để xem xét và định hướng các công việc trong tương lai.

Chương 2

Kiến thức nền tảng

Trước khi thực hiện các công việc được đề ra ở phần mục tiêu của Luận văn, nhóm cần phải đi tìm hiểu một số kiến thức nền tảng để đưa ra định hướng cho Luận văn. Trong chương này, những kiến thức đầu tiên được tìm hiểu là các giải pháp công nghệ thông tin về thực hành lập trình. Các giải pháp này nhằm định hướng cho việc hiện thực hệ thống lập trình nền tảng phù hợp với nhu cầu giảng dạy trong phạm vi đề tài. Phần tiếp theo sẽ trình bày về các nghiên cứu để làm cơ sở khoa học cho việc phân loại độ khó câu hỏi. Và cuối cùng, để hiện thực được hệ thống, nhóm trình bày các công nghệ cần tìm hiểu để phát triển hệ thống.

2.1 Các giải pháp công nghệ thông tin liên quan

2.1.1 MOOCs

MOOCs, được viết tắt từ Massive Open Online Courses, là những khóa học trực tuyến đại chúng với số lượng người tham gia không giới hạn. Khái niệm về MOOC được đặt ra bởi Dave Cormier dựa trên khóa học của giáo sư người Canada Stephen Downes, với khóa học trực tuyến *Connectivism and Connective Knowledge* gồm 25 học viên trả phí cho việc học tập tại trường đại học Manitoba, cùng với 2200 sinh viên toàn thế giới không phải trả một đồng phí nào (Chris Parr, 2013).

Vài năm sau đó, cùng với sự phát triển của Internet, thuật ngữ MOOC bắt đầu trở nên phổ biến. Các tổ chức thương mại bắt đầu xây dựng các nền tảng MOOCs nổi tiếng như edX, Coursera, Udacity,.. Và từ đó dẫn đến sự xuất hiện hai dạng hình thái phổ biến của MOOCs:

- cMOOCs (Connectivist MOOCs): Như tên gọi của nó, những tư liệu học tập của khóa học này không chỉ nằm trên 1 trang web. Thay vào đó, cMOOCs là nơi tập hợp lại các tư liệu học tập, giúp người dùng có thể tiếp cận với tất cả tư liệu học tập có sẵn trên Internet.
- xMOOCs (Extended MOOCs): Nắm bắt nhu cầu của việc học trực tuyến và học từ xa, các tổ chức xây dựng các nền tảng học trực tuyến của riêng mình. Các tư liệu học tập của

xMOOCs chỉ nằm trên một nền tảng, một trang web duy nhất và không thể chia sẻ ra bên ngoài nhưng vẫn đảm bảo tính miễn phí, tinh mở của MOOCs.

Dù là hình thái nào, MOOCs chỉ đa số đáp ứng các tư liệu học tập dưới dạng slide, video bài giảng, quiz,... Các khóa học MOOCs về lập trình lại cần có sự luyện tập nhiều hơn bên cạnh lý thuyết. Do đó, các khóa học này thường cung cấp hai mã nguồn: mã nguồn khởi tạo của chương trình và mã nguồn đáp án khi hoàn thành chương trình. Người học sẽ vừa theo dõi video lý thuyết, vừa thực hành với mã nguồn khởi tạo trên thiết bị cá nhân. Sau khi làm bài xong, người dùng tự so sánh bài làm với mã nguồn đáp án được cung cấp. Cách luyện tập này không trực quan và hiệu quả bằng việc có một hệ thống hỗ trợ người học nộp bài tập, chấm điểm tự động và phản hồi kết quả làm bài.

2.1.2 HackerRank

Hình 2.1: *HackerRank*

HackerRank là một nền tảng giúp các lập trình viên luyện tập lập trình trực tuyến, giải quyết các bài toán về lĩnh vực Khoa học Máy tính như Giải thuật, Học máy, hoặc Trí tuệ nhân tạo, cùng các kỹ thuật lập trình khác như lập trình hàm,... (Hackerrank, 2021). Bên cạnh đó, HackerRank còn có một số hoạt động khác như:

- Tổ chức các cuộc thi lập trình giúp các lập trình viên có thể cạnh tranh với nhau hoặc doanh nghiệp tìm ra giải pháp cho các vấn đề của mình.
- Cung cấp các chứng chỉ cho người dùng sau khi hoàn thành một khóa học.
- Là cầu nối giữa các doanh nghiệp và lập trình viên trong vấn đề tìm việc và tuyển dụng. Nhà tuyển dụng sử dụng các chứng chỉ và thành tích của các cuộc thi trên HackerRank như một thước đo đánh giá khả năng của lập trình viên.

HackerRank có tuổi đời khá lâu cùng với cộng đồng lớn mạnh nên cung cấp được rất nhiều câu hỏi, bài tập lập trình cùng những khóa học giúp người dùng có thể học lập trình.

2.1.3 CodeLearn

CodeLearn là một sản phẩm của FPT Software, được phát triển tiếp tục từ nền tảng thi lập trình của cuộc thi CodeWar Fsoft và FJP năm 2018, với tầm nhìn trở thành một kênh giúp các lập trình viên tại Việt Nam chia sẻ, học tập các kiến thức về Khoa học Máy tính. CodeLearn được mô tả là một hệ thống và nền tảng trực tuyến giúp người dùng học, luyện tập và được đánh giá khả năng lập trình thông qua các bài tập, cuộc thi lập trình bằng chức năng tự chấm điểm, giúp người dùng nâng cao kỹ năng và tăng năng suất lập trình (CodeLearn, 2021).

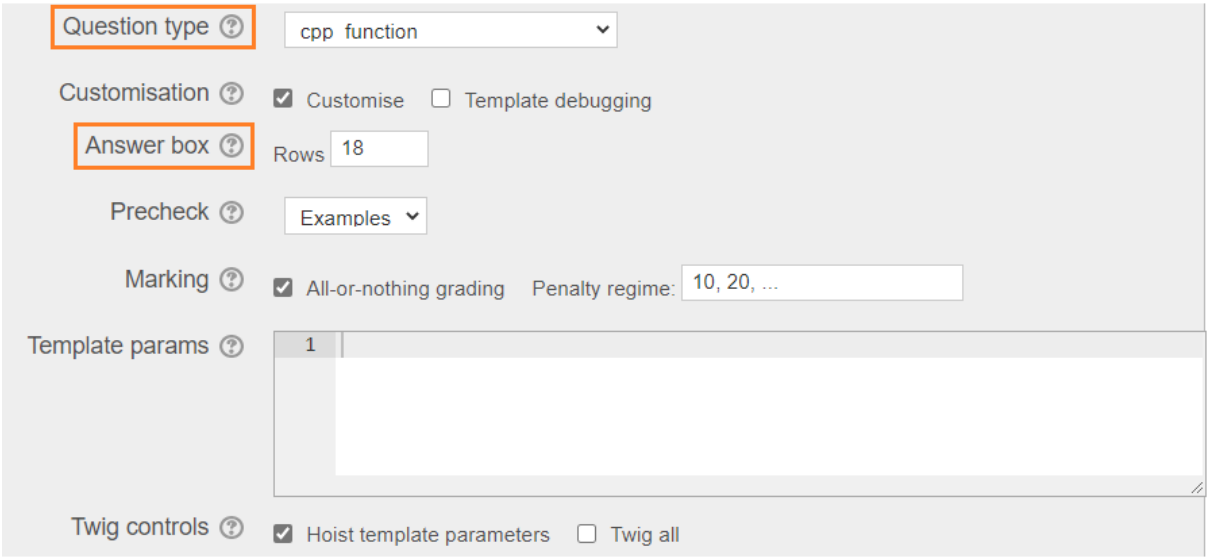
Hình 2.2: *CodeLearn*

Là người đi sau so với HackerRank, CodeLearn hiện có lượng bài tập và khóa học lập trình khá hạn chế so với HackerRank. Tuy nhiên, thế mạnh của CodeLearn chính là các bài tập và khóa học đều có Tiếng Việt giúp dễ tiếp cận hơn với người học lập trình Việt Nam.

2.1.4 CodeRunner

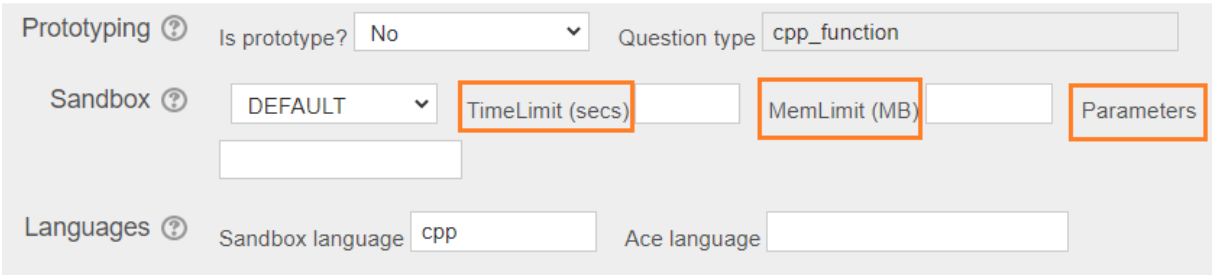
Moodle (Green, 2021) là một hệ thống quản lý học tập phổ biến trên thế giới. CodeRunner (Lobb and Hunt, 2021) là một tiện ích có trong Moodle cho phép tạo các bài tập thực hành lập trình trên nhiều ngôn ngữ lập trình khác nhau. Người học sử dụng CodeRunner có thể nộp bài ở dạng mã nguồn, sau đó hệ thống chạy mã nguồn, chấm điểm và hiển thị kết quả cho người học. Trong đề tài này, nhóm chỉ khảo sát CodeRunner được cài đặt trên hệ thống Bách Khoa e-Learning (BKEL), các câu hỏi lập trình được khảo sát chỉ thuộc ngôn ngữ lập trình C/C++. Sau khi tìm hiểu, nhóm lựa chọn các tính năng mà nhóm cảm thấy là ưu điểm của CodeRunner để trình bày trong phần này:

- *Precheck*: cho phép bài chấm chạy trên một số testcases được đánh dấu, các testcases này được dùng để kiểm tra cú pháp và các trường hợp đơn giản, giúp sinh viên tránh được lỗi sai khi nộp bài chính thức (bằng nút *Kiểm tra*).
- *Question type*: cho phép chọn một kiểu câu hỏi đã được định nghĩa sẵn. Trong đó có kiểu câu hỏi *c_function*, *cpp_function* cho phép gom các testcode (là các testcases ở dạng mã nguồn được thay thế vào chương trình), giúp trong một lần chạy chương trình có được kết quả của tất cả testcode, tăng tốc thời gian phản hồi kết quả cho người học, đồng thời giảm tải cho máy chủ chấm bài.



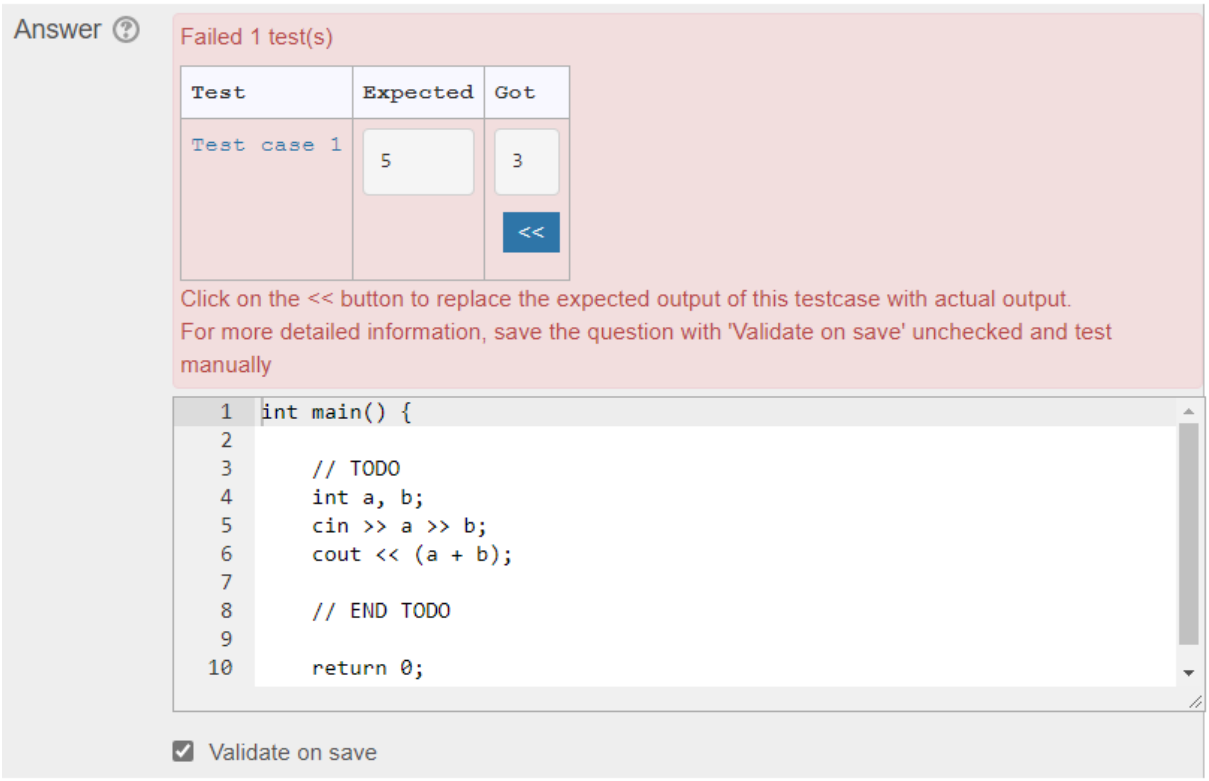
Hình 2.3: *CodeRunner - Precheck và Question type*

- *Sandbox*: cho phép cài đặt giới hạn thời gian chạy và kích thước bộ nhớ khi chạy của chương trình. Người dùng có thể cài đặt thêm trong phần *Parameter* các thông tin như địa chỉ IP để trỏ CodeRunner gửi bài tập đến địa chỉ của con chấm khác con chấm mặc định, các tùy chọn (option) cho trình biên dịch,... Khuyết điểm của *Parameter* là yêu cầu thông tin nhập vào ở dạng json, người dùng cần biết các key được định nghĩa trước mới có thể nhập thông tin hợp lệ.



Hình 2.4: *CodeRunner - Sandbox*

- *Validate on save*: nếu người tạo câu hỏi cung cấp cả mã nguồn đáp án và đầu ra mong muốn, người dùng có thể sử dụng tính năng này để hệ thống chạy kiểm tra kết quả đầu ra của mã nguồn và đầu ra cung cấp có trùng khớp hay không. Nếu không trùng nhau, CodeRunner cung cấp tính năng sử dụng kết quả chạy trên mã nguồn đáp án là kết quả mong muốn.



Hình 2.5: *CodeRunner - Validate on save*

- *Testcases*: mỗi testcase cho phép dùng testcode hoặc luồng nhập chuẩn (stdin), cho phép cài đặt ẩn/hiện testcase sau khi chấm bài.

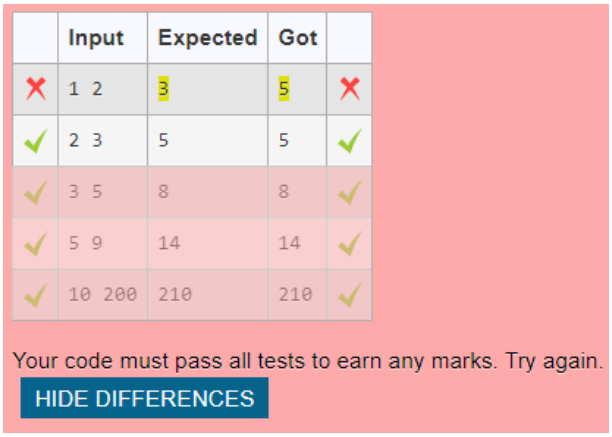
- *Hiển thị kết quả chấm bài*: kết quả chấm bài trên CodeRunner được hiển thị trực quan, giúp người học thấy được kết quả bài nộp chạy và kết quả mong đợi của bài tập.

2.1.5 Hệ thống Auto Grading System - AGS

AGS là một hệ thống hỗ trợ thực hành lập trình, được xây dựng và đưa vào sử dụng trong quá trình giảng dạy và học tập của Khoa Khoa học và Kỹ thuật Máy tính trong thời gian qua.

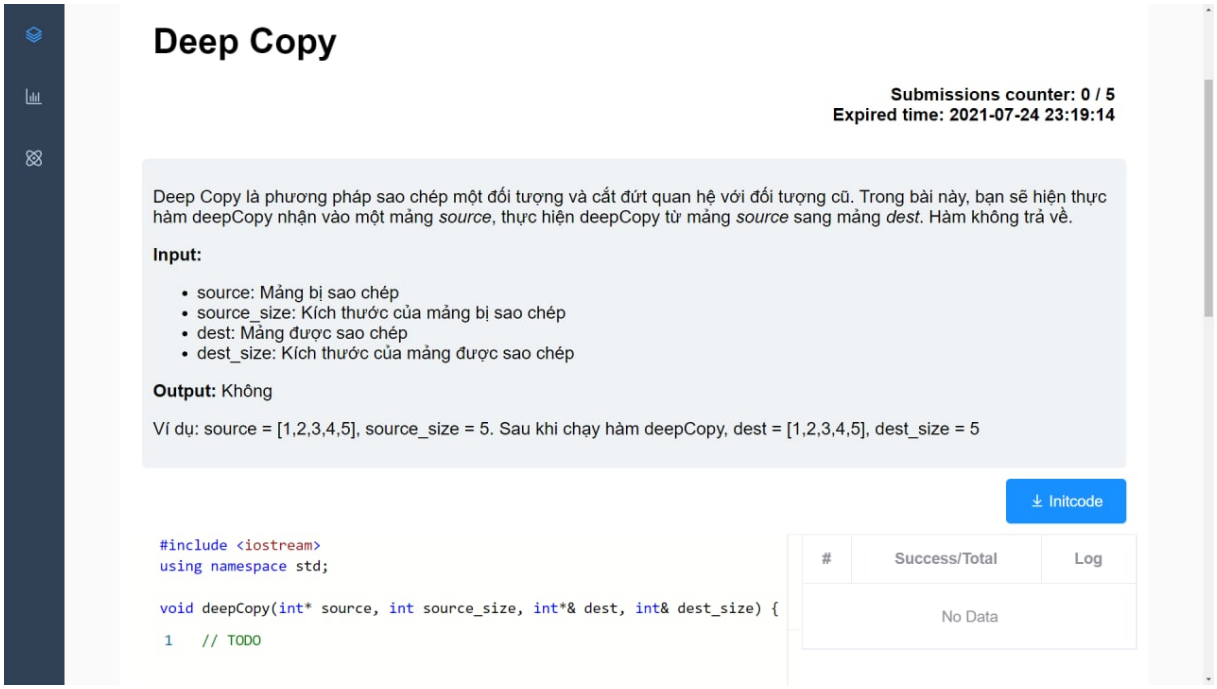
AGS đang thực hiện hai chức năng chính của một hệ thống hỗ trợ thực hành, đó là:

- Quản lý danh sách các bài tập của từng môn học và cho phép gán bài tập vào những nhóm người dùng nào đó.



Hình 2.6: *CodeRunner - Hiển thị kết quả chấm bài*

- Chấm các bài nộp sau khi người dùng làm bài theo nội dung và yêu cầu của câu hỏi.



Hình 2.7: *AGS - Màn hình làm bài và nộp bài với nội dung và yêu cầu của câu hỏi kèm theo*

Để thực hiện những chức năng đó, AGS đã được thiết kế với năm thành phần và mỗi thành phần thực hiện một chức năng khác nhau:

- *AGS Back-end*: Nơi lưu trữ các câu hỏi và xử lý các request từ các thành phần khác gọi qua.
- *AGS Front-end*: Nơi tương tác giữa người học với hệ thống.
- *AGS Front-end Admin*: Nơi tương tác giữa những người dùng được cấp quyền đặc biệt như người quản trị hệ thống, giảng viên giảng dạy, trợ giảng, v.v... với hệ thống.

- *AGS Grader*: Nơi chấm bài cho các bài nộp được gửi lên bởi người dùng.
- *AGS Watcher*: Đóng vai trò là một cân bằng tải cho bước chấm bài của hệ thống.

AGS được nhóm đánh giá là một hệ thống dễ sử dụng và đáng tin cậy. Tuy nhiên, vì chỉ mới được phát triển gần đây nên vẫn còn thiếu vắng những tính năng cần thiết khác để AGS có thể được đem ra sử dụng và phát triển lâu dài.

2.2 Các nghiên cứu khoa học liên quan

Trong khuôn khổ luận văn, tính năng quan trọng mà có tính khoa học cần khảo sát, nghiên cứu các công trình là tính năng phân loại độ khó câu hỏi dưới góc nhìn của người học. Tính năng này cần khảo sát các yếu tố liên quan đến độ khó của câu hỏi lập trình cũng như các phương pháp phân loại độ khó câu hỏi.

2.2.1 Các nghiên cứu liên quan đến độ khó câu hỏi lập trình

Khi tiến hành xem xét độ khó các câu hỏi lập trình, có rất nhiều yếu tố ảnh hưởng đến độ khó khác nhau do các tác giả đi trước đề xuất. Nhóm xin trình bày một số các yếu tố ảnh hưởng đến độ khó câu hỏi lập trình mà các tác giả trước đã sử dụng.

Điểm trung bình là yếu tố thường được sử dụng để đánh giá độ khó của các câu hỏi. Simon et al. (2012) chỉ sử dụng điểm trung bình của người làm bài để đo đạc độ khó của các câu hỏi kiểm tra lập trình. Một ví dụ khác, Mahatme and Bhoyar (2016) đã lấy tổng tất cả điểm của sinh viên chia cho tổng số sinh viên làm trọng số để phân loại các câu hỏi trong môi trường e-learning.

Ngoài yếu tố điểm trung bình, đối với các câu hỏi dạng lập trình, các yếu tố khác cũng được xem xét để mô tả độ khó của chúng.

Để dự đoán kết quả làm bài của sinh viên bằng dữ liệu của một hệ thống chấm tự động, nhóm tác giả Chen and Ward (2021) sử dụng các phương pháp phân loại và hồi quy với 4 đặc trưng bao gồm:

- Tỉ lệ số testcases làm đúng so với tổng số testcases
- Kết quả in ra của các testcases
- Khoảng thời gian giữa các lần nộp bài
- Số lần nộp bài

Trong nghiên cứu của Vamsi et al. (2020), độ khó của câu hỏi được đề xuất là tỉ lệ thuận với tổng số lượng bài nộp đối với một câu hỏi.

Awat and Ballera (2018) đã tiến hành khảo sát các câu hỏi trong bài kiểm tra (item analysis) bằng các kết quả làm bài của sinh viên. Một trong các bước khảo sát câu hỏi chính là xác định độ khó của một câu. Công thức được trình bày là số sinh viên làm bài đúng chia cho tổng số lượng sinh viên làm bài.

Với mục tiêu đo lường độ khó của các câu hỏi lập trình, trong những thông tin được trích từ tập dữ liệu, Chowdhury and Watanobe (2018) thực hiện nghiên cứu dữ liệu bằng phương pháp gom cụm, chọn số sinh viên đạt, số bài nộp đạt, và các thông tin khác làm đặc trưng (features).

Mặc dù độ khó câu hỏi có thể sử dụng dưới dạng công thức hoặc là một đặc trưng cho các phương pháp Học máy, nhóm nhận thấy có một vài yếu tố ảnh hưởng được sử dụng để xác định độ khó. Bảng 2.1 trình bày tóm tắt các yếu tố được sử dụng trong các nghiên cứu trên để mô tả độ khó.

Bảng 2.1: *Tóm tắt các yếu tố liên quan đến độ khó trong các nghiên cứu*

Yếu tố	Bài báo
Điểm trung bình	Simon et al. (2012), Mahatme and Bhoyar (2016)
Số sinh viên làm đúng	Awat and Ballera (2018), Chowdhury and Watanobe (2018)
Số bài nộp đạt	Chowdhury and Watanobe (2018), Mahatme and Bhoyar (2016)
Điểm bài nộp cao nhất	Chen and Ward (2021)
Số lần nộp bài	Chen and Ward (2021), Vamsi et al. (2020)

2.2.2 Các phương pháp phân loại độ khó câu hỏi

a) Thuật toán di truyền mờ (Fuzzy genetic algorithm)

Với công việc phân loại độ khó câu hỏi, Pérez et al. (2012) và Verdú et al. (2010) xây dựng một hệ thống chuyên gia (expert system) sử dụng giải thuật di truyền và logic mờ (fuzzy logic) để để phân loại câu hỏi. Việc ước lượng độ khó của câu hỏi được thực hiện qua 2 giai đoạn. Trong giai đoạn 1, mô hình mờ sẽ học từ các mẫu phản hồi câu hỏi từ sinh viên, sau đó tự tạo các quy tắc phân loại và các tập mờ của biến đầu vào cho dữ liệu cụ thể. Trong giai đoạn 2, hệ thống sẽ suy ra độ khó của các câu hỏi.

Yếu tố chính cung cấp thông tin cho quá trình học là mẫu phản hồi câu hỏi từ sinh viên. Mẫu này cung cấp 3 tham số mà hệ thống quan tâm là thời gian theo phút từ lần đọc đề cuối cùng của câu hỏi đến khi sinh viên nộp bài, số điểm đạt được cho bài nộp đó và số lần sinh viên đọc câu hỏi trước khi nộp bài.

Ban đầu, tất cả các câu hỏi đều có một độ khó được gán bởi người dạy. Với mỗi độ khó, các mẫu phản hồi của các câu hỏi có cùng độ khó này sẽ được gom lại và trích xuất ra 3

tham số trên, dữ liệu này được sử dụng như một tập rõ nét (crisp sets). Từ tập rõ nét, tập mờ và các quy tắc được sinh ra, từ đó có thể suy luận độ khó của từng mẫu phản hồi. Độ khó của một câu hỏi sẽ là trung vị (median) của các mẫu phản hồi cho câu hỏi đó. Như vậy độ khó của câu hỏi là sự kết hợp từ góc nhìn của người dạy và kết quả làm bài từ người học.

b) Thuật toán *k*-means

Một thuật toán khác thường được sử dụng để phân loại câu hỏi là thuật toán phân cụm *k*-means.

Vamsi et al. (2020) sử dụng *k*-means trong phân loại độ khó của câu hỏi lập trình trên *HackerRank*. Các đầu vào cho thuật toán là: thời gian để giải quyết câu hỏi, điểm đạt được cho bài nộp và độ khó được phân loại từ người ra đề. Số cụm của thuật toán là 3, tương ứng với dễ, trung bình, khó.

Awat and Ballera (2018) sử dụng *k*-means trên kết quả làm bài trong khóa học *Computer Programming 1*. Tác giả thực hiện phân loại các câu hỏi thành 5 độ khó là: rất dễ, dễ, trung bình, khó và rất khó. Đầu vào của thuật toán là tỉ số giữa số bài nộp đúng trên tổng số bài nộp của sinh viên cho câu hỏi đó.

Thuật toán *k*-means có thể giúp phân loại độ khó câu hỏi hoàn toàn dựa vào kết quả làm bài của người học mà không cần đến sự phân loại ban đầu từ người dạy.

c) Thảo luận

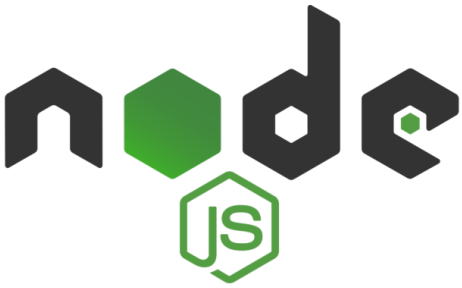
Kết quả độ khó trong phương pháp phân loại sử dụng Thuật toán di truyền mờ là sự kết hợp của người dạy và người học. Trong khi đó, độ khó sử dụng phương pháp *k*-means chỉ dựa trên kết quả làm bài của người học nên không có sự ước lượng của người dạy. Trong Luận văn này, nhóm tác giả mong muốn độ khó có được là hoàn toàn từ góc nhìn của người học. Do đó, phương pháp *k*-means được nhóm lựa chọn sử dụng.

Tuy nhiên, các nghiên cứu sử dụng thuật toán *k*-means chưa đề cập đến cách để xác định một cụm tương ứng với độ khó nào. Trong Luận văn này, nhóm sẽ đề xuất một phương pháp gán độ khó phù hợp với cụm, từ đó hoàn thiện giải pháp phân loại độ khó.

2.3 Một số công nghệ được sử dụng

2.3.1 Node.js

Node.js (OpenJS, 2021) là một môi trường mã nguồn mở đa nền tảng, dùng để khởi chạy mã nguồn JavaScript. Đây cũng là một công cụ phổ biến cho hầu hết các loại dự án CNTT.



Hình 2.8: *Node.js*

Node.js có những đặc tính sau, giúp một ứng dụng dựa trên Node.js có được hiệu năng cao, phù hợp cho một hệ thống đáp ứng được số lượng lớn người dùng và có khả năng phát triển thêm trong tương lai:

- Ứng dụng dựa trên Node.js chỉ chạy trên một tiến trình (process) duy nhất, giúp tránh được một lượng lớn các lỗi xảy ra do đụng độ.
- Node.js sử dụng cơ chế bất đồng bộ trong việc truy xuất các tác vụ đầu vào/đầu ra (input/output, gọi tắt là I/O) như truy xuất mạng, truy xuất cơ sở dữ liệu, truy xuất tập tin (file), v.v... nhằm mang lại hiệu năng tốt hơn vì không phải chờ kết quả trả về như cơ chế đồng bộ.
- Node.js cũng có một kho lưu trữ các thư viện rất lớn là npm [1], với hơn 1 triệu thư viện mã nguồn mở miễn phí. Các thư viện này cung cấp các thao tác cần thiết cho một số công việc nhất định, giúp lập trình viên tiết kiệm được thời gian phát triển hệ thống. Nhờ vậy, các lập trình viên có thể xây dựng được nhiều loại hệ thống khác nhau với đa dạng các tính năng dựa trên hệ sinh thái Node.js này.

2.3.2 Vue.js

Vue.js (gọi tắt là Vue) là một framework linh động (progressive) dùng để xây dựng giao diện người dùng (user interface, gọi tắt là UI) (Vue.js, 2021).

Nhờ vào sự phát triển của JavaScript, các trang web hiện nay ngày càng có giao diện người dùng linh động hơn. Tuy nhiên, nếu chỉ dùng mỗi Javascript thì các mã nguồn về giao diện người dùng sẽ có hàng nghìn dòng code JavaScript được kết nối với một vài tập tin HyperText Markup Language (HTML) và Cascading Style Sheets (CSS) mà không có một sự tổ chức cụ thể nào cả. Với mục tiêu như các framework JavaScript khác, Vue.js được tạo ra với mục tiêu

[1] npm - https://www.npmjs.com/

Hình 2.9: *Vue.js*

xây dựng một khung lập trình giúp lập trình viên có thể duy trì được mã nguồn tốt hơn và dựng lên giao diện người dùng một cách nhanh chóng.

Vue được giới thiệu như một framework với 3 tính chất:

- **Dễ tiếp cận** (Approachable): Là một framework UI được ra mắt trễ hơn (so với một số framework nổi tiếng như Angular, React, v.v...), Vue được thiết kế dễ học, dễ sử dụng cho những lập trình viên mới bắt đầu học về lập trình UI lẫn người đã có kinh nghiệm lâu năm về một framework khác muốn chuyển sang.
- **Linh hoạt** (Versatile): Như lời giới thiệu của Vue - một framework linh động, Vue có thể được dùng như một thành phần gắn thêm của ứng dụng (chỉ một vài trang, hoặc component) để đem lại sự trực quan cho giao diện. Tuy nhiên, Vue vẫn có thể đáp ứng được nếu người dùng muốn sử dụng Vue trong toàn bộ ứng dụng của mình với Vuex, Vue-Router, v.v....
- **Hiệu suất tốt** (Performant): Vue cam kết có thể đạt hiệu suất ngang ngửa với Angular (một framework UI nổi tiếng về hiệu năng, tốc độ phản hồi), đem lại trải nghiệm trực quan, phản hồi nhanh cho người dùng ứng dụng.

Ngoài những tính chất được nêu trên, Vue còn có đem lại những tính năng, lợi ích khác như:

- Vue có thể đáp ứng được khả năng mở rộng (scalability) của ứng dụng bằng cách cung cấp cho lập trình viên khả năng chia mã nguồn UI thành nhiều thành phần có thể tái sử dụng (Reusable Components).
- Vue còn cung cấp một giao diện dòng lệnh (command line interface, gọi tắt là CLI) giúp lập trình viên có thể khởi động, khởi tạo project hoặc tạo mới các thành phần một cách nhanh chóng.
- Lập trình viên thậm chí không cần phải tạo sự tương tác giữa các tập tin HTML, CSS và JavaScipt mà có thể sử dụng tính năng Single File Vue Components. Tính năng này giúp lập trình viên có thể gói gọn cả 3 thành phần trên thành 1 tập tin duy nhất.

2.3.3 Flask

Flask là một Python framework, cung cấp các công cụ hỗ trợ xây dựng một ứng dụng web. Flask được xây dựng hướng đến sự gọn nhẹ (lightweight) và nhỏ gọn (micro).

Hình 2.10: *Flask*

Các lợi ích của việc sử dụng Flask (Flask documentation, 2021; Introduction to Flask, 2021; 7 lý do chọn Flask, 2021):

- Flask là một micro web framework. *micro* có nghĩa là Flask cung cấp một lõi chức năng đơn giản nhất có thể cho ứng dụng web nhưng dễ dàng mở rộng. Ban đầu, Flask không bao gồm lớp trừu tượng cơ sở dữ liệu (database abstraction layer), xác thực biểu mẫu (form validation) hoặc bất cứ gì khác mà các thư viện khác nhau đã tồn tại để xử lý. Thay vào đó, Flask hỗ trợ các tiện ích mở rộng để thêm chức năng đó vào ứng dụng khi người dùng cần.
- Flask dễ dàng cài đặt và triển khai. Để sử dụng Flask, người dùng chỉ cần cài đặt 1 package là Flask.
- Flask phù hợp cho việc xây dựng các ứng dụng web có quy mô vừa và nhỏ, các Application Programming Interface (API) và các web service: việc xây dựng web application rất giống với việc viết các module Python chuẩn, cấu trúc gọn gàng và rõ ràng; người dùng tự chọn cài các thành phần phù hợp với tính năng hướng đến.
- Flask giúp người dùng tập trung xây dựng ý tưởng. Flask có kiến trúc nhỏ gọn, tối giản, dễ tìm hiểu và sử dụng; giúp người dùng mới có thể tiếp cận và hiểu cách hoạt động của Flask một cách nhanh chóng. Sau đó, người dùng tập trung vào công việc xây dựng các tính năng mà bản thân mong muốn.
- Nguồn tài liệu tham khảo phong phú và có một cộng đồng người dùng lớn.
- Flask được sử dụng phổ biến bởi các tổ chức uy tín như Pinterest, LinkedIn, Reddit, Netflix, v.v....

Tính năng phân loại câu hỏi và gợi ý lộ trình thực hành được nhóm xác định cần thực hiện nhiều bước xử lý dữ liệu. Hơn nữa, tính năng phân loại câu hỏi được định hướng sử dụng thuật toán *k*-means để phân cụm. Các công việc xử lý dữ liệu, cũng như công việc liên quan đến Học máy (như *k*-means) hiện nay thường được thực hiện trên Python. Python mang đến khả năng hiện thực mã nguồn nhanh chóng và có nhiều thư viện Học máy mạnh mẽ đã được hiện thực sẵn. Do đó, nhóm chọn ngôn ngữ Python cho các công việc phân loại độ khó và gợi ý câu hỏi. Cùng với các lý do nêu trên, Flask được lựa chọn làm web framework cho ngôn ngữ Python, giúp hiện thực các giao tiếp từ mô-đun này đến các mô-đun khác.

2.3.4 Redis

Redis (Redis Labs, 2021) là một nền tảng mã nguồn mở, sử dụng bộ nhớ chính (RAM) để lưu trữ dữ liệu có cấu trúc (structured data) và thường được dùng như một Cơ sở dữ liệu (CSDL) dạng key-value hoặc làm bộ nhớ đệm (cache) với tốc độ truy cập nhanh chóng, hiệu quả, giúp nâng cao hiệu năng hệ thống.

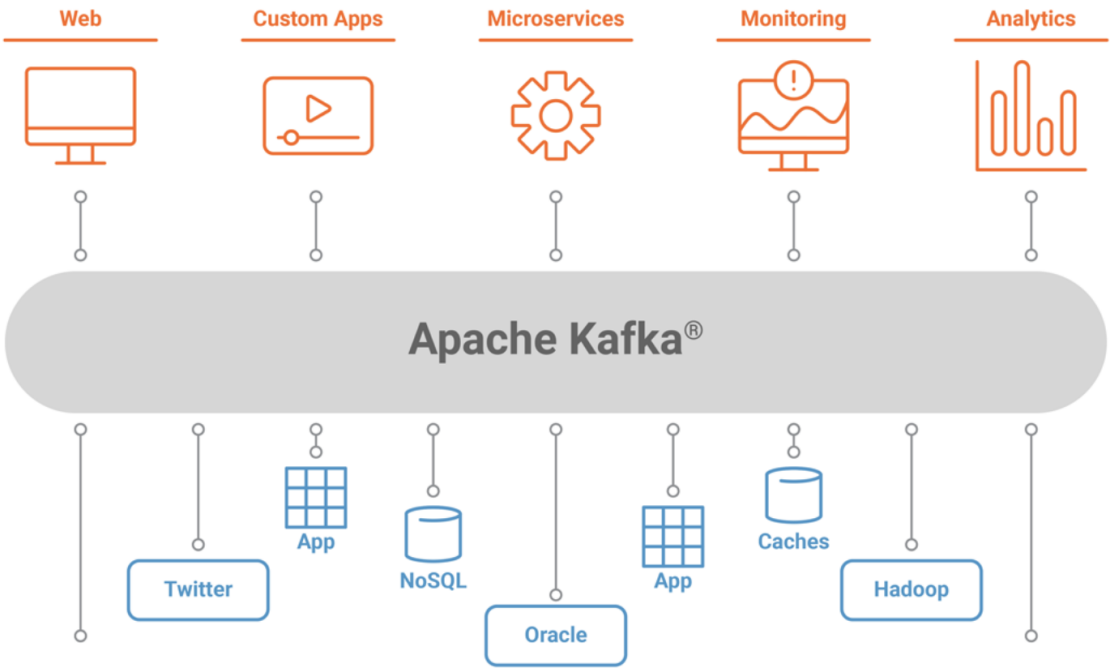
Hình 2.11: *Redis*

Với một hệ thống hỗ trợ thực hành lập trình, câu hỏi sau khi được soạn bởi người dạy sẽ được lưu trong hệ thống có xu hướng đọc nhiều (read-intensive). Do vậy, nguồn dữ liệu này nên được lưu vào một bộ nhớ đệm như Redis để đọc dữ liệu nhanh hơn.

2.3.5 Kafka

Kafka (Confluent, 2021), hay tên đầy đủ là Apache Kafka, là một nền tảng truyền tải sự kiện phân tán (distributed event streaming), và thường được sử dụng làm một hàng đợi thông điệp (message queue) với khả năng đáp ứng một số lượng lớn sự kiện trong ngày.

Sau khi khảo sát về các hệ thống hỗ trợ thực hành lập trình, nhóm nhận thấy thành phần chấm bài là nơi chịu áp lực tải tương đối lớn do phải liên tục chấm bài từ phía người học. Với khả năng được giới thiệu như trên, nhóm nhận định rằng Kafka là một sự lựa chọn hoàn toàn phù hợp cho quá trình chấm bài của hệ thống. Kafka mang lại hiệu năng cần thiết để đáp ứng cho nhu cầu của số lượng lớn sinh viên nộp bài cùng lúc, cũng như đảm bảo được thứ tự trước sau cho các bài nộp này trong một hàng đợi.



Hình 2.12: *Kafka*

2.3.6 PostgreSQL

PostgreSQL (The PostgreSQL Global Development Group, 2021) là một hệ thống cơ sở dữ liệu quan hệ đối tượng (object-relational database system), mã nguồn mở, sử dụng ngôn ngữ SQL kết hợp với nhiều tính năng giúp lưu trữ và chia tỷ lệ các khối dữ liệu phức tạp một cách an toàn.

Là một hệ cơ sở dữ liệu phổ biến với tuổi đời hơn 30 năm và vẫn còn được tiếp tục phát triển, PostgreSQL nổi tiếng về độ tin cậy, cung cấp các tính năng mạnh mẽ và hiệu suất cao. Bên cạnh đó, các tính năng của PostgreSQL đáp ứng được hầu hết nhu cầu truy xuất cơ bản của một hệ thống.

2.3.7 Jobe

Jobe (viết gọn cho Job Engine (Richard Lobb, 2021)) là một dịch vụ hỗ trợ biên dịch và thực thi những đoạn chương trình nhỏ, hỗ trợ nhiều ngôn ngữ phổ biến hiện tại như C, C++, Python

Hình 2.13: *PostgreSQL*

3, Java, Pascal,...

Jobe cung cấp một giao diện (interface) để tương tác thông qua Representational state transfer (REST) API, trong đó đặc tả các thành phần như ngôn ngữ sử dụng, mã nguồn, đầu vào chuẩn cho mã nguồn (nếu có) và tùy chọn các tập tin gắn kèm dùng trong quá trình khởi chạy. Jobe sẽ biên dịch và khởi chạy mã nguồn với ngôn ngữ đã được xác định, kèm theo dữ liệu đầu vào đã cho (đầu vào chuẩn hoặc tập tin gắn kèm) và cuối cùng sẽ trả về những thông tin như đầu ra chuẩn hoặc thông tin lỗi chuẩn (nếu có).

Phiên bản hiện tại của Jobe đã hiện thực một số API trong tài liệu hiện đang được cung cấp, nhằm đáp ứng cho nhu cầu chấm bài cơ bản của CodeRunner (đã được giới thiệu ở mục 2.1.4). CodeRunner là một tiện ích của Moodle chuyên hỗ trợ tạo, chấm điểm các bài tập lập trình, trong đó sử dụng Jobe như một môi trường sandbox để chấm bài. Và với mức độ phủ sóng rộng rãi ở các trường Đại học, đặc biệt là các trường có giảng dạy về những ngành Công nghệ thông tin, với hơn 600 trang web sử dụng Jobe trên nền tảng CodeRunner trên toàn cầu.

Các câu hỏi trong hệ thống hiện tại thường yêu cầu người học phải hiện thực các chương trình theo yêu cầu của người ra đề, và các chương trình này không quá lớn nên nhóm đánh giá việc sử dụng Jobe ở bước chấm bài cho hệ thống là hợp lý, phù hợp với mục tiêu mà Jobe hướng đến. Khả năng tích hợp dễ dàng thông qua REST API và tính linh hoạt trong quá trình sử dụng, cộng với việc hỗ trợ nhiều ngôn ngữ khác nhau và nhiều kiểu dữ liệu đầu vào cho quá trình chấm bài cũng là những điểm cộng lớn khi xem xét sử dụng Jobe trong hệ thống.

Một số API mà Jobe hỗ trợ như:

- Biên dịch, thực thi mã nguồn và trả về kết quả (bao gồm cả đầu ra chuẩn, thông tin lỗi chuẩn hoặc các lỗi khi biên dịch, thực thi, v.v...)
- Tải tập tin đính kèm lên Jobe
- Kiểm tra xem tập tin đính kèm đã có trên Jobe hay chưa

Jobe cung cấp thông tin chi tiết về các API ở trên cũng như các API khác tại Richard Lobb (2021).

Do vậy, nhóm đánh giá Jobe là một con chấm tiềm năng để tích hợp vào hệ thống mới. Tuy nhiên, cũng cần phải có những bước đánh giá tính tương tích và hiệu năng chấm bài sau khi tích hợp thành công để có được kết luận cuối cùng cho lựa chọn này.

2.4 Kết chương

Chương này trình bày các Kiến thức nền tảng mà nhóm đã tìm hiểu trong Luận văn này. Các kiến thức này sẽ là cơ sở để nhóm đề xuất và hiện thực các giải pháp trong các chương kế tiếp.

Trước khi sang các chương tiếp theo, nhóm xin điểm lại các thông tin chính trong chương này:

a) Về các giải pháp CNTT liên quan:

- MOOCs là một nền tảng hỗ trợ học tập trực tuyến đại chúng nhưng chưa có sự hỗ trợ mạnh mẽ cho bài tập lập trình.
- Một số hệ thống khác chuyên hỗ trợ thực hành lập trình được tìm hiểu là HackerRank, CodeLearn, CodeRunner, AGS.
- AGS cung cấp nền tảng hỗ trợ thực hành khá đầy đủ và có thể tùy biến mã nguồn được nên được nhóm lựa chọn là hệ thống phát triển các giải pháp mới.

b) Các nghiên cứu khoa học liên quan:

- Một số yếu tố liên quan đến độ khó câu hỏi lập trình được nhóm khảo sát và trình bày.
- Nhóm tìm hiểu một số phương pháp phân loại, sau đó nhóm quyết định sử dụng thuật toán phân cụm k-means cùng với đề xuất một phương pháp gán độ khó tương ứng với cụm.

c) Các công nghệ sử dụng:

- Node.js được sử dụng để hiện thực các chức năng chính của hệ thống, như chấm bài, quản lý câu hỏi, quản lý người dùng, v.v...
- Vue.js được sử dụng làm nền tảng phát triển giao diện người dùng.
- Flask làm web framework cho mô-đun phân loại độ khó và gợi ý câu hỏi.
- Redis làm bộ nhớ đệm cho hệ thống.
- Kafka làm hàng đợi nhận bài nộp để chấm bài.
- PostgreSQL làm CSDL của hệ thống.
- Jobe làm thành phần thực thi mã nguồn và trả về kết quả làm bài của người dùng.

Chương 3

Thiết kế hệ thống

Chương này sẽ giới thiệu về hệ thống nền mà nhóm dùng để phát triển tiếp - hệ thống AGS thông qua ba phần: Kiến trúc hệ thống, các tính năng và cơ sở dữ liệu. Từ đó, nhóm bắt đầu thiết kế hệ thống mới và chỉ ra những thay đổi so với hệ thống cũ để làm nền tảng cho việc hiện thực các giải pháp mới.

3.1 Giới thiệu hệ thống AGS

3.1.1 Lý do lựa chọn hệ thống AGS

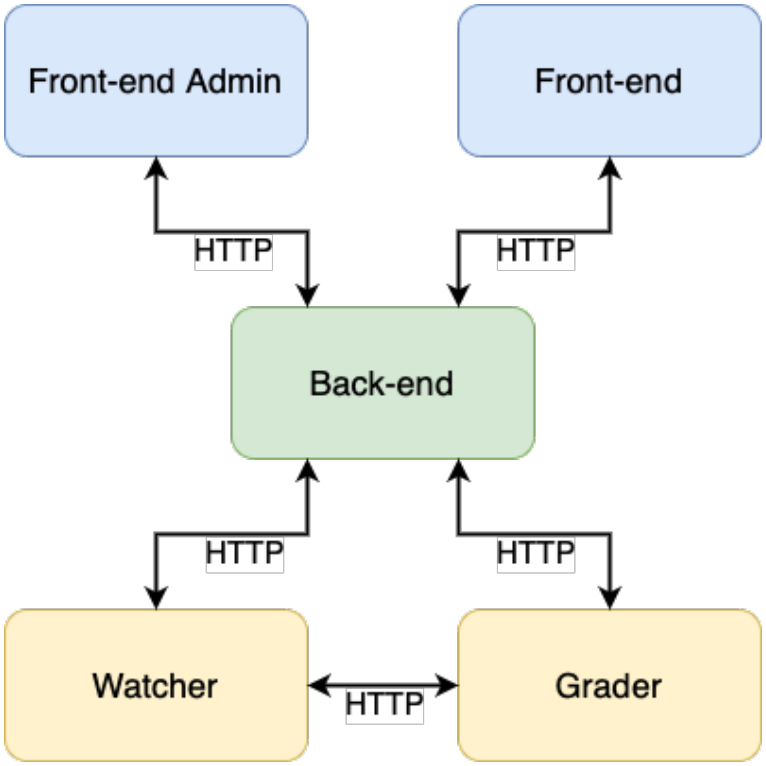
Như đã trình bày trong mục 1.2, bảng 1.1 trình bày các yếu tố nổi bật của hệ thống AGS như sau:

- Hệ thống chấm bài nộp của người học theo phương pháp bất đồng bộ. Từ đó, người học không phải chờ hệ thống phản hồi mỗi khi nộp bài.
- Hệ thống có khả năng tiếp cận mã nguồn để phát triển.

Bên cạnh đó, hệ thống đã được hiện thực một số chức năng làm bài lập trình cơ bản, quản lý lớp học, quản lý bài thực hành. Tuy nhiên, hệ thống vẫn còn những khiếm khuyết cần cải thiện như con chấm Docker liên tục bật-tắt làm giảm hiệu năng chấm bài, ngôn ngữ hỗ trợ chỉ có C++,... Nhờ việc tiếp cận mã nguồn, nhóm có thể tái sử dụng một phần chương trình, giúp giảm thời gian và công sức phát triển phần mềm. Vì những lý do trên, nhóm quyết định chọn AGS để cải thiện và phát triển tiếp thay vì phải xây dựng một hệ thống mới từ đầu.

3.1.2 Kiến trúc hệ thống AGS

Trong hệ thống AGS, các thành phần được thiết kế theo hướng microservices - tức mỗi thành phần là một dịch vụ nhỏ. Mỗi dịch vụ thực hiện một hoặc một số chức năng nhất định trong hệ



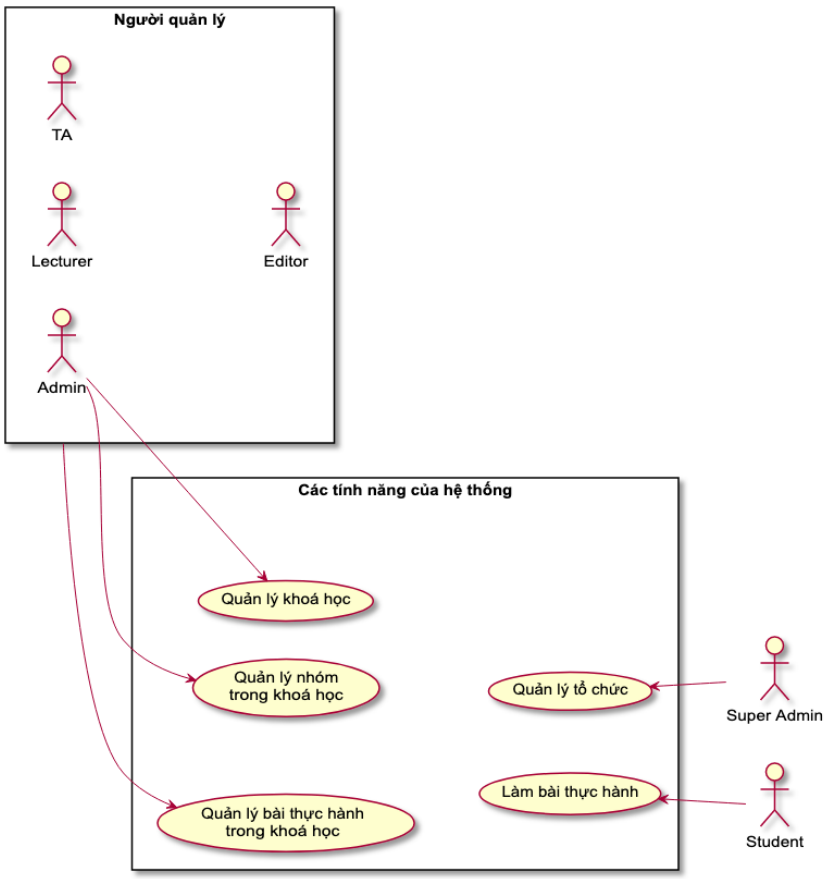
Hình 3.1: *Các thành phần trong hệ thống AGS*

thống. Qua đó, khi các thành phần phối hợp với nhau, các nghiệp vụ đề ra sẽ được hệ thống thực hiện. Cũng nhờ vào kiến trúc microservices, hệ thống có thể đáp ứng được nhiều chức năng cho nhiều người dùng cùng lúc, phù hợp với mục tiêu mà nhóm đã đặt ra cho một hệ thống hỗ trợ thực hành lập trình.

Các thành phần trong hệ thống AGS bao gồm:

- Front-end Admin: Nơi tương tác giữa những người dùng được cấp quyền đặc biệt như quản trị viên, giảng viên, trợ giảng, v.v... với hệ thống.
- Front-end: Nơi tương tác giữa người học với hệ thống.
- Back-end: Nơi kết nối với các thành phần khác trong hệ thống để xử lý nghiệp vụ sinh ra từ lời gọi (request) của người dùng.
- Watcher: Cân bằng tải cho quá trình chấm bài của hệ thống.
- Grader: Nơi chấm bài cho các bài nộp được gửi lên bởi người dùng.

Các mối quan hệ cũng như cách tương tác giữa các thành phần trong hệ thống AGS được mô tả lại như hình vẽ dưới đây:



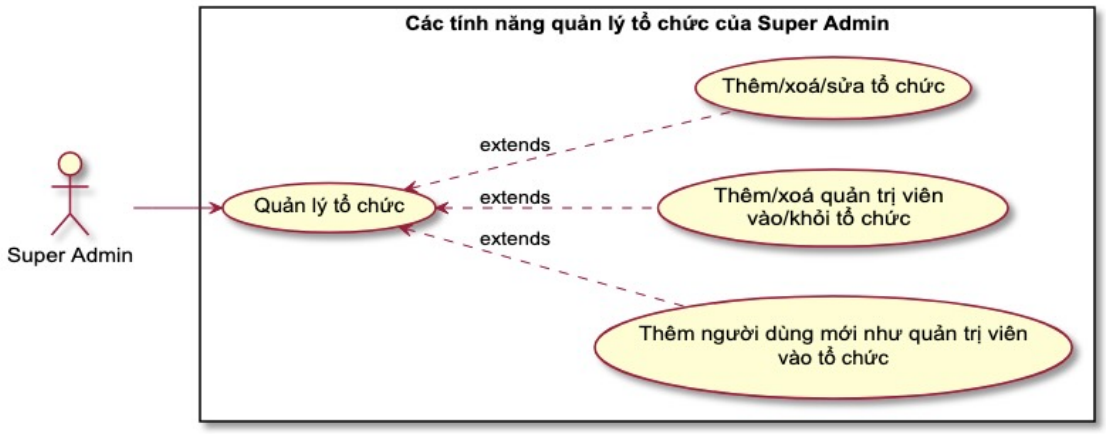
Hình 3.2: *Use-case diagram của hệ thống AGS*

3.1.3 Các tính năng trong hệ thống AGS

Hệ thống AGS cũng cung cấp khá đầy đủ các tính năng cho người dùng, phục vụ cho nhu cầu dạy và học thực hành lập trình. Các tính năng được biểu diễn như sơ đồ chức năng (use-case diagram) như sau:

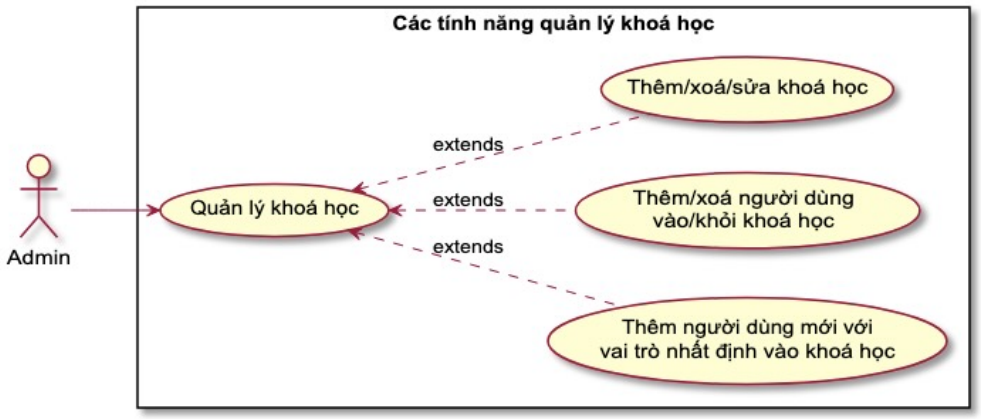
Chi tiết cho từng cụm tính năng được trình bày theo các hình dưới đây:

- Cụm tính năng về quản lý tổ chức:



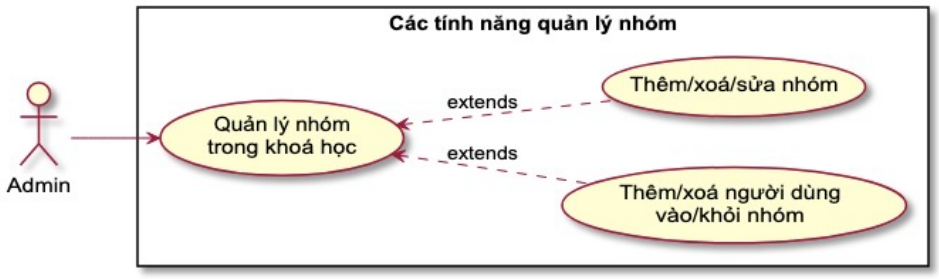
Hình 3.3: *Cụm tính năng về quản lý tổ chức của Super Admin*

- Cụm tính năng về quản lý khoá học:



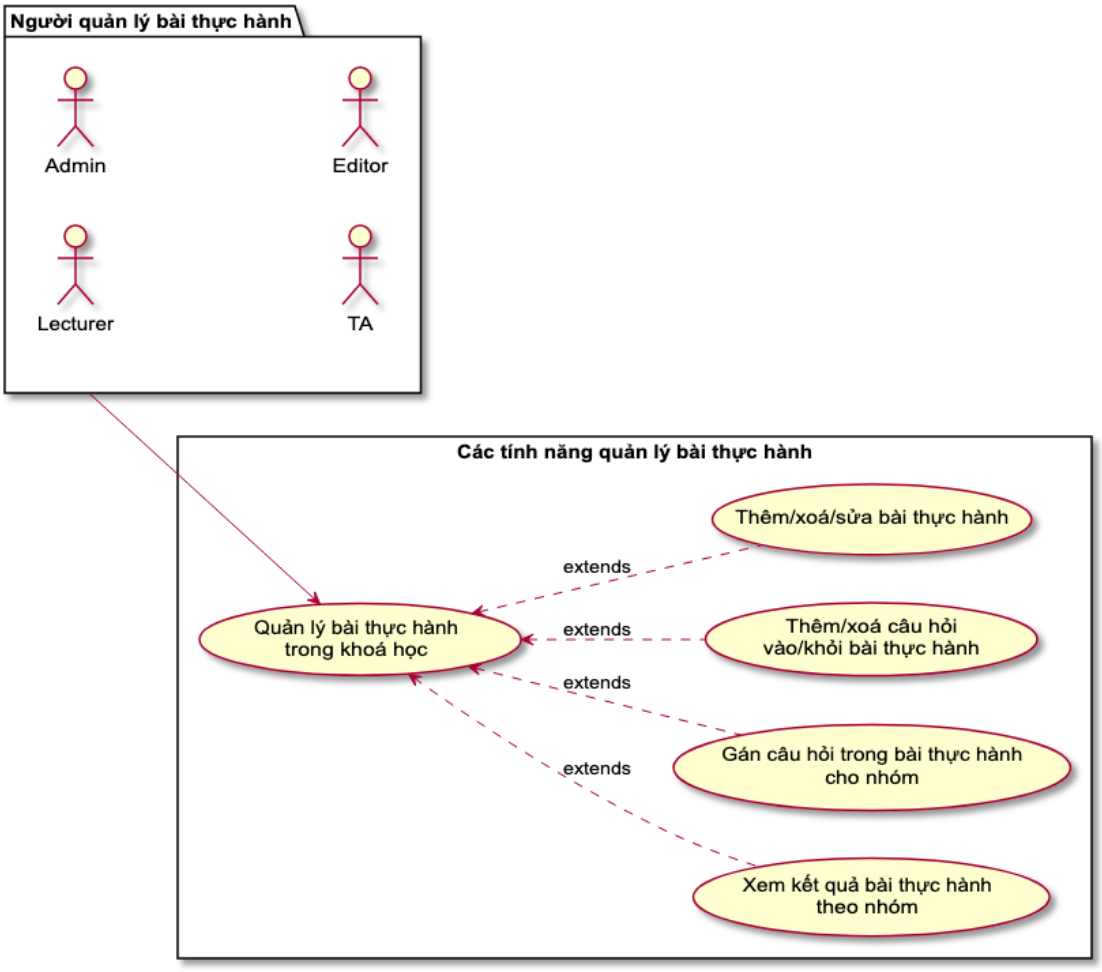
Hình 3.4: *Cụm tính năng về quản lý khoá học*

- Cụm tính năng về quản lý nhóm trong khoá học:



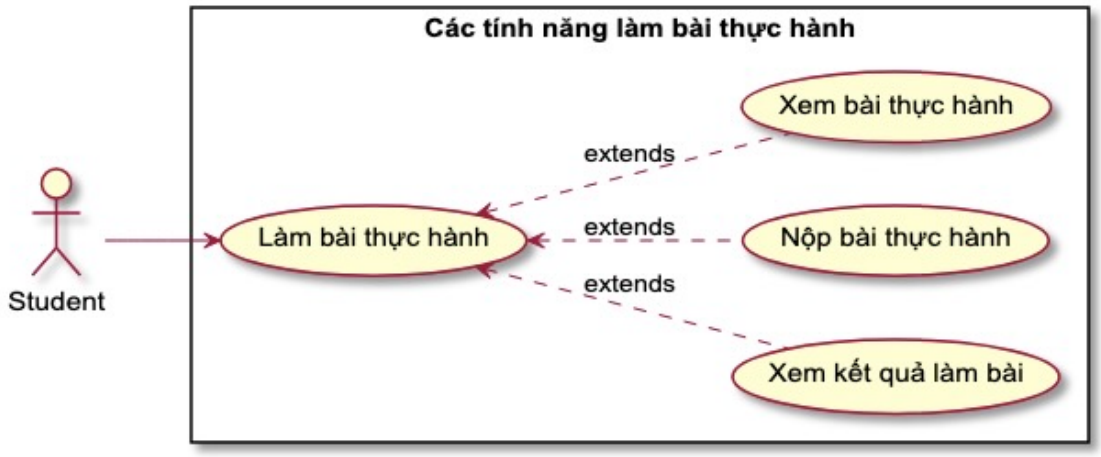
Hình 3.5: *Cụm tính năng về quản lý nhóm trong khoá học*

- Cụm tính năng về quản lý bài thực hành trong khoá học:



Hình 3.6: *Cụm tính năng về quản lý bài thực hành trong khoá học*

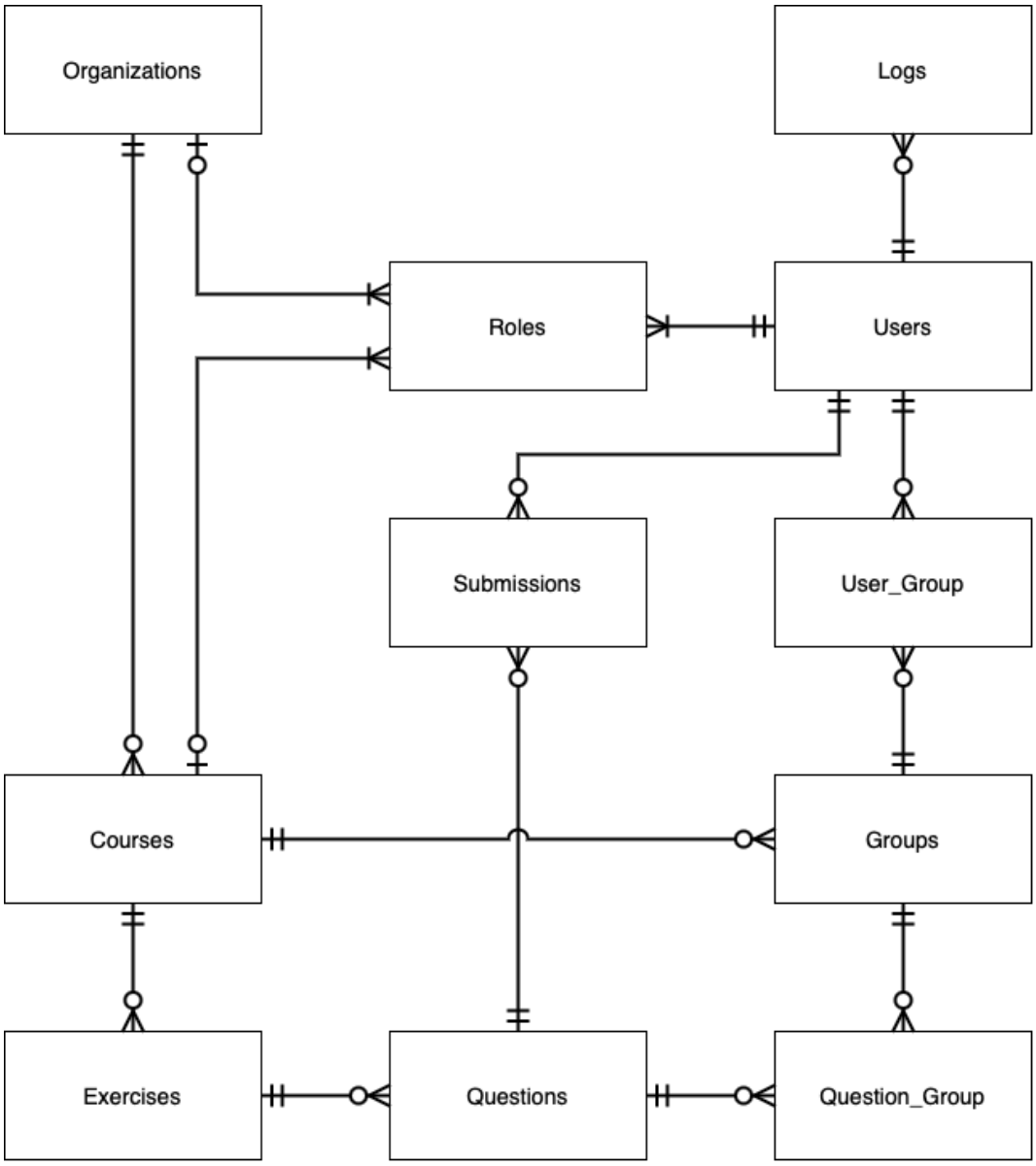
- Cụm tính năng về làm bài thực hành:



Hình 3.7: *Cụm tính năng về làm bài thực hành*

3.1.4 Cơ sở dữ liệu hệ thống AGS

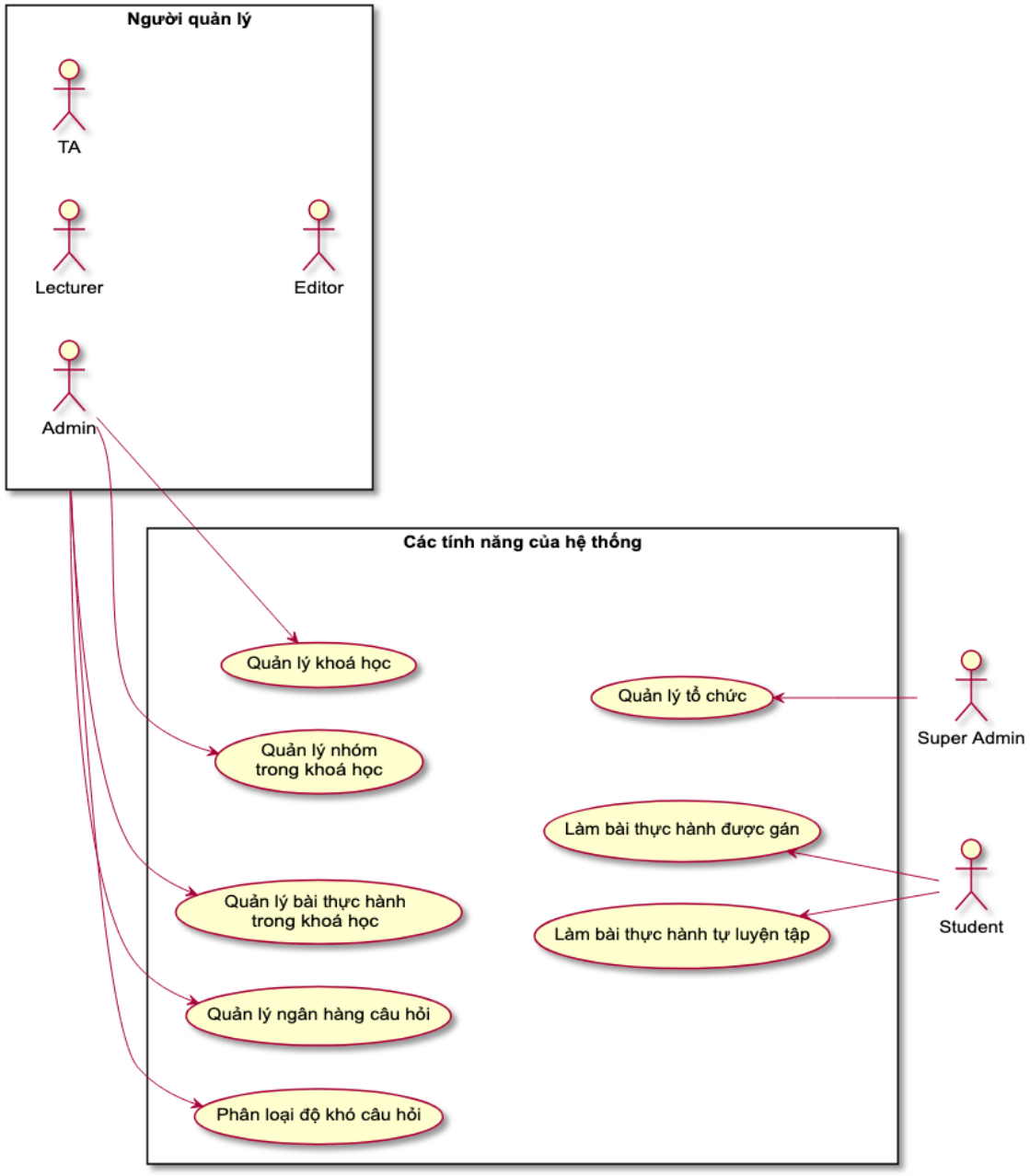
Hình 3.8 trình bày về các thực thể và các quan hệ được sử dụng trong hệ thống AGS. Sơ đồ quan hệ thực thể được sử dụng ký hiệu vết chân chim (được trình bày cụ thể trong chương Phụ lục C).



Hình 3.8: *Sơ đồ quan hệ thực thể trong cơ sở dữ liệu của hệ thống AGS*

3.2 Các tính năng trong hệ thống mới

Với mục tiêu đề ra cho hệ thống mới, nhóm dự kiến hệ thống sẽ bao gồm những cụm chức năng như sau:



Hình 3.9: *Use-case diagram của hệ thống xây dựng*

Trong các tính năng trên thì các tính năng đã có trong AGS và được sử dụng lại là: quản lý tổ chức, quản lý khoá học, quản lý nhóm, quản lý bài thực hành trong khoá học và làm bài thực hành. Như vậy, các tính năng mới nhóm sẽ xây dựng là: làm bài thực hành tự luyện tập, quản lý ngân hàng câu hỏi, phân loại độ khó câu hỏi.

Đối với mỗi tính năng, nhóm xác định các yêu cầu sau:

- Tính năng làm bài thực hành nói chung (bài tập được gán và tự luyện tập): hệ thống kiểm tra tính đúng đắn của bài nộp và đưa kết quả đến người học. Việc chấm bài cần được thực

hiện bất đồng bộ. Hệ thống chấm bài một cách ổn định khi số lượng người dùng tăng lên, và có cơ chế mở rộng đơn giản bằng cách tăng con chấm.

- Tính năng làm bài thực hành tự luyện tập: tự động gợi ý các câu hỏi từ dễ đến khó và phù hợp với khả năng của từng người học.
- Tính năng quản lý ngân hàng câu hỏi: giúp người dạy tổ chức lưu trữ các câu hỏi lập trình một cách tiện lợi, là nguồn cung cấp câu hỏi đến người học.
- Tính năng phân loại câu hỏi: phân loại độ khó câu hỏi trên kết quả làm bài của người học, giúp người dạy xem xét đánh giá lại độ khó của câu hỏi gán cho người học.

3.3 Kiến trúc hệ thống mới

Cùng với những tính năng mới và một số yêu cầu được đặt ra, nhóm đề xuất một số thay đổi trên kiến trúc hệ thống đang có, cụ thể:

- Hệ thống sẽ có thêm những thành phần mới như Classifier dùng cho tính năng phân loại độ khó câu hỏi, Suggestor dùng cho tính năng gợi ý câu hỏi.
- Hệ thống phải thay đổi thành phần chấm bài, vì hiệu năng của con chấm được sử dụng là không tốt. Nguyên nhân cơ bản là do cơ chế bật tắt liên tục giữa mỗi lần chấm bài, do vậy cần tìm một giải pháp luôn bật để cải thiện về hiệu năng cho quá trình chấm bài. Ngoài ra, cần xem xét về luồng bất đồng bộ đang được sử dụng để tối ưu hơn nếu có thể.

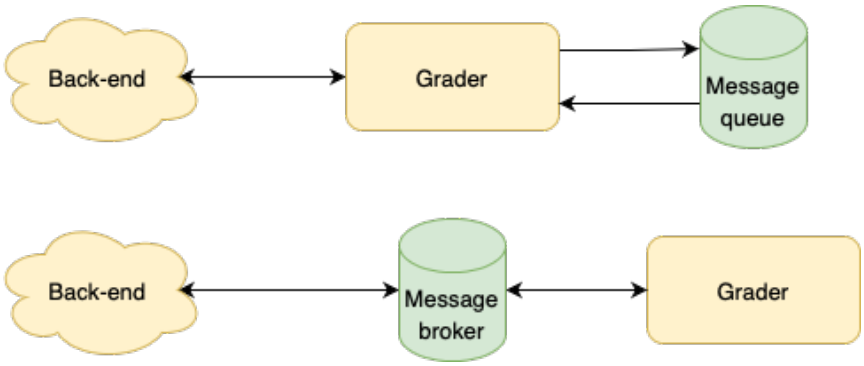
Với sự thay đổi về các thành phần mới như Classifier, Suggestor:

- Nhờ vào kiến trúc microservices hiện có của AGS, nhóm thêm vào hai thành phần mới vào hệ thống để thực hiện các chức năng mới mà không gây ảnh hưởng gì đến các nghiệp vụ hiện có.
- Các thông tin mà hai thành phần này cần như câu hỏi, bài nộp đều có sẵn trên CSDL của hệ thống. Do vậy, nhóm sẽ thiết kế hai thành phần này chỉ nhận lời gọi (request) từ phía Back-end và còn lại sẽ kết nối trực tiếp với CSDL của hệ thống để truy xuất thông tin.
- Nhóm cũng cân nhắc về luồng thực thi cho cả hai quá trình là phân loại câu hỏi cũng như gợi ý câu hỏi. Vì cả hai đều cần lượng thông tin lớn, xử lý dữ liệu nhiều, do vậy thời gian phản hồi tương đối lâu. Vì vậy, nhóm sẽ học tập luồng thực thi chấm bài để thiết kế các luồng thực thi cho hai thành phần này theo hướng bất đồng bộ để tối ưu hiệu năng của hệ thống.

Với sự thay đổi về thành phần con chấm, nhóm nhận thấy, con chấm Jobe của tiện ích CodeRunner cho nền tảng Moodle là một sự thay thế phù hợp. Không những có cơ chế hoạt động như nhóm mong đợi, mà nó còn đang được sử dụng khá rộng rãi và ổn định trên các trang lớn như Bách Khoa e-Learning với số lượng người dùng tương đối lớn. Các thông tin chi tiết hơn và các đánh giá về Jobe sẽ được trình bày sau, tại phần 4.1.2.

Về luồng chấm bài bất đồng bộ hiện có của AGS, nhóm đánh giá là vẫn có cơ hội để cải tiến hơn nữa, thông qua việc ứng dụng một công nghệ mới là trung gian truyền tải thông điệp (message broker).

Message broker có khả năng chuyển hoàn toàn luồng thực thi về bất đồng bộ, thay vì phải sử dụng kết hợp tổ hợp lời gọi chấm bài và một hàng đợi thông điệp (message queue) như hiện tại.



Hình 3.10: *So sánh hai giải pháp về luồng thực thi bất đồng bộ*

Hơn nữa, message queue đang sử dụng được dựa trên nền tảng của Redis. Vì đặc tính lưu trữ dữ liệu trên bộ nhớ chính và có khả năng bị mất mát, do vậy bài nộp có khả năng bị mất đi trước khi được thực sự chấm bài. Nên nhóm sẽ tìm kiếm một giải pháp message broker có khả năng lưu trữ bền vững để hạn chế tình trạng này xảy ra.

Hai giải pháp phổ biến được nhóm xem xét cho message broker và thoả mãn được tiêu chỉ lưu trữ bền vững cũng như cho hiệu năng tốt, gồm RabbitMQ và Kafka. Trong đó, điểm khác biệt nhất giữa hai giải pháp này là cách tiếp cận đẩy (push) và kéo (pull) message.

- Với cách tiếp cận push-based của RabbitMQ, bài nộp sau khi được ghi nhận thì sẽ được "đẩy" sang thành phần chấm bài. Nên nếu như có số lượng lớn bài nộp được gửi cùng lúc sẽ dẫn đến thành phần chấm bài có thể bị quá tải.

- Ngược lại, với cách tiếp cận pull-based của Kafka, việc chấm bài sẽ không bị ép như cơ chế trên. Mà thay vào đó, khi nào thành phần chấm bài chấm xong một bài nộp sẽ tiến hành "kéo" bài nộp mới về để chấm. Điều này hoàn toàn phù hợp với cách thức hoạt động của thành phần chấm bài mà nhóm mong muốn.

- Dù RabbitMQ có khả năng giới hạn lại số bài nộp bị "đẩy" sang, tức việc chấm bài cũng

không bị quá ép buộc. Tuy nhiên, với cách tiếp cận tốt như Kafka, nhóm nhận thấy đây là một điểm cộng để lựa chọn Kafka làm message broker cho hệ thống.

Hơn nữa, Kafka cũng được đánh giá là cho thông năng (throughput) cao hơn RabbitMQ[1], giúp đáp ứng được nhu cầu có rất nhiều người nộp bài cùng lúc, phù hợp cho việc phát triển trong tương lai. Một số tổ chức lớn với rất nhiều người dùng trên toàn thế giới cũng đã ứng dụng Kafka trong hệ thống của họ như Netflix, Twitter, Spotify,...

Tổng kết lại, nhóm sẽ bổ sung Kafka vào hệ thống để hoạt động như là một message broker cho các luồng xử lý bất đồng bộ - bao gồm quá trình chấm bài lẫn hai quá trình mới là phân loại độ khó câu hỏi và gợi ý câu hỏi.

Ngoài ra, nhóm cũng muốn tối ưu hơn về cách truy xuất dữ liệu của hệ thống bằng cách ứng dụng bộ đệm (cache). Các dữ liệu như bài thực hành, câu hỏi, sau khi được soạn bởi người dạy trong hệ thống thường có xu hướng đọc nhiều (read-intensive). Do vậy, nếu sử dụng cache dựa trên bộ nhớ chính (RAM) để đọc các dữ liệu này sẽ mang lại hiệu năng tốt hơn, giảm thời gian phản hồi cho người dùng.

Một số giải pháp cache được nhóm tìm thấy và xem xét, bao gồm:

- Redis
- Memcached
- Tự hiện thực

Với giải pháp tự hiện thực, nhóm nhận thấy việc bỏ ra chi phí để tự hiện thực - cả về logic bên trong cũng như cách kết nối ra bên ngoài là quá lớn nên nhóm không thể sử dụng giải pháp này. Hơn nữa, các giải pháp còn lại vốn đã được sử dụng rộng rãi ngoài công nghiệp nên tính đáng tin cậy đã được kiểm chứng. Do đó, nhóm tiếp tục xem xét hai giải pháp là Redis và Memcached.

Cả Redis lẫn Memcached đều lưu trữ dữ liệu theo dạng key-value, tức với một khoá (key) sẽ có một giá trị (value) tương ứng. Điều này giúp đơn giản hoá quá trình tích hợp và logic sử dụng cũng dễ hiểu hơn cho lập trình viên. Chúng đều cung cấp mức độ phản hồi ở khoảng millisecond, giúp việc truy xuất nhanh hơn nhiều so với database truyền thống vốn dựa trên bộ nhớ thứ cấp (disk)[2].

Tuy nhiên, giữa hai giải pháp vẫn sẽ có đôi chút khác biệt:

- Memcached hướng đến nhu cầu sử dụng đơn giản và lưu trữ thuần là key-value, với dung lượng tối đa cho key là 250 bytes và cho value là 1 megabyte.

[1]So sánh hiệu năng giữa các message broker - https://www.confluent.io/blog/kafka-fastest-messaging-system/

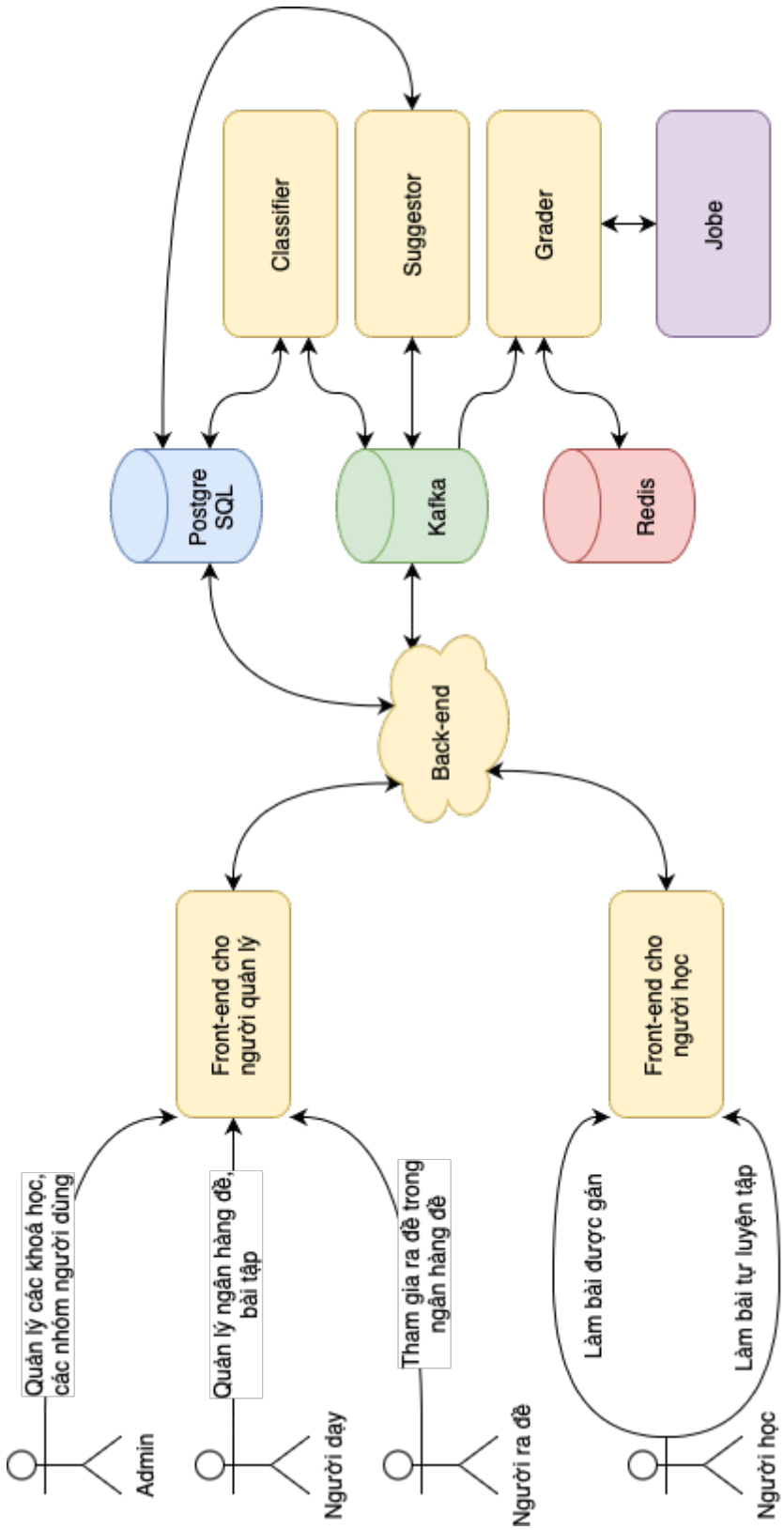
[2]So sánh Redis và Memcached - https://aws.amazon.com/elasticache/redis-vs-memcached/

- Trong khi đó, Redis hướng đến việc cung cấp một nền tảng đa chức năng, bên cạnh việc lưu trữ key-value. Chẳng hạn như dùng Redis như một hàng đợi thông điệp gửi nhận (pub/sub message queue), hay ứng dụng làm bảng xếp hạng với kiểu cấu trúc được hỗ trợ là sorted sets,... Redis cũng cung cấp mức dung lượng tối đa cao hơn, với 512 megabytes cho key và 512 megabytes cho value[3].

Với nhu cầu phát triển nhiều tính năng hơn trong tương lai, nhóm mong muốn tìm kiếm một giải pháp đa chức năng và hỗ trợ tối đa cho quá trình hiện thực. Do vậy, nhóm quyết định sử dụng Redis là giải pháp cache cho hệ thống.

Như vậy, hệ thống sau khi được thêm vào những thành phần mới và áp dụng những giải công nghệ như trên, kiến trúc mới sẽ như sau:

[3]So sánh Redis và Memcached - https://www.imaginarycloud.com/blog/redis-vs-memcached/



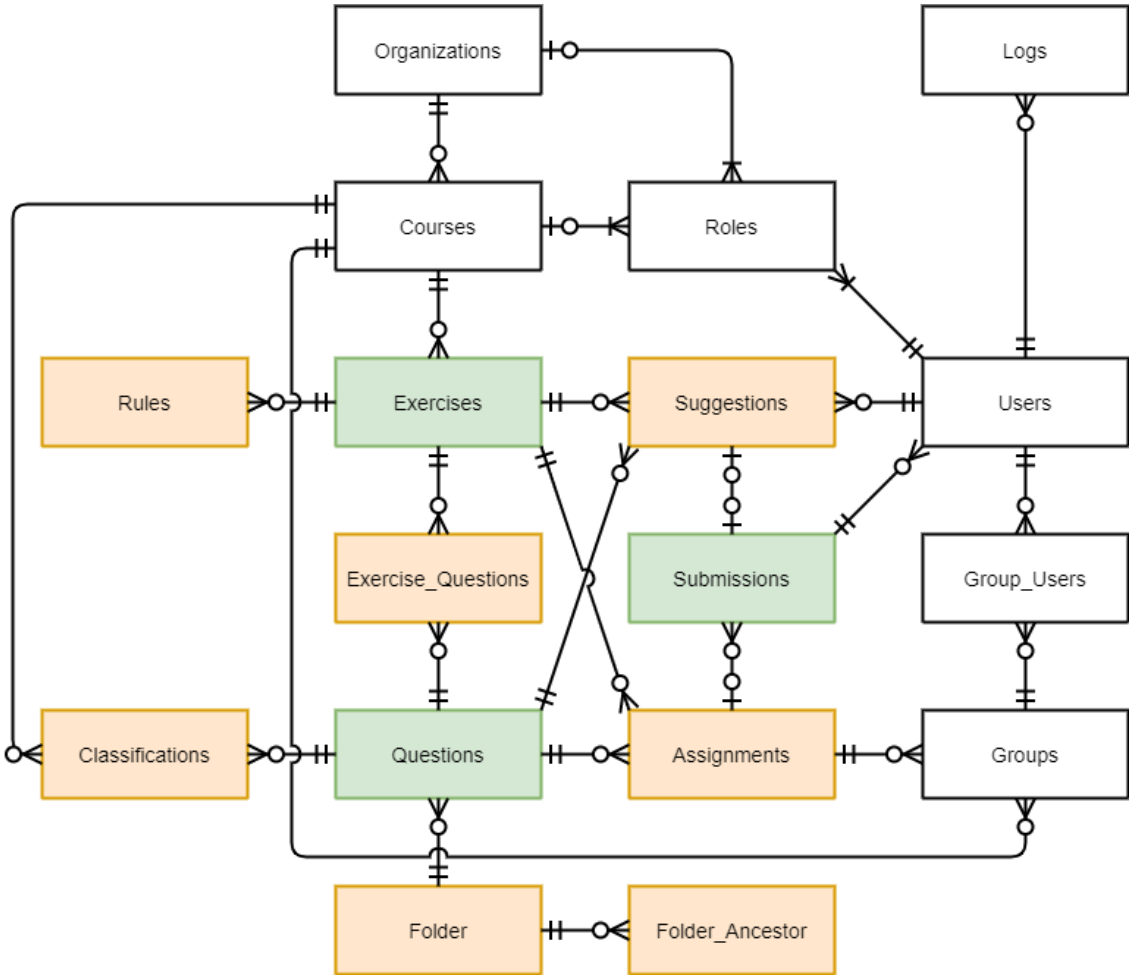
Hình 3.11: *Tổng quan kiến trúc hệ thống*

3.4 Thiết kế cơ sở dữ liệu

Hệ cơ sở dữ liệu của hệ thống cũng được thiết kế lại cho phù hợp với các tính năng mới. Một số vẫn được nhóm giữ cấu trúc như cũ, song, cần thiết kế các cá thể mới và các quan hệ mới.

Chi tiết về sơ đồ quan hệ thực thể được trình bày qua hình 3.12. Trong đó:

- Các bảng mới được đánh dấu bằng màu cam.
- Các bảng cũ nhưng được cập nhật lại thì được đánh dấu bằng màu xanh lá.
- Các bảng còn lại được giữ nguyên từ hệ thống cũ.



Hình 3.12: *Sơ đồ quan hệ thực thể của hệ thống đề xuất*

So sánh với sơ đồ quan hệ thực thể của hệ thống AGS cũ, sơ đồ quan hệ thực thể mới sẽ có các bảng và quan hệ mới như sau:

- Bảng Assignments được tạo mới, thay thế cho bảng Question_Group trong hệ thống cũ. Với mỗi phép gán, một câu hỏi trong một bài thực hành sẽ được gán cho một nhóm sinh

viên nào đó. Sau khi có được phép gán, những bài nộp của các bạn sinh viên sẽ gắn kèm với phép gán này.

- Quan hệ giữa Exercises và Questions đã biến mất vì nhóm đã đưa câu hỏi ra khỏi mối quan hệ với bài thực hành, giúp câu hỏi không bị ràng buộc và có thể được tái sử dụng cho nhiều bài thực hành khác nhau.

- Ngoài ra, một số bảng mới như Exercise_Questions, Folder, Folder_Ancestor, Classifications, Rules, Suggestions và các quan hệ có liên quan sẽ được trình bày cụ thể trong phần hiện thực ở mục 4.2.8 và 4.3.13.

Chi tiết về các bảng cùng với các trường thông tin trong bảng sẽ được trình bày lần lượt dưới đây để có nguồn tham khảo nếu cần.

Bảng Organizations chứa thông tin về các tổ chức trong hệ thống. Một tổ chức được xem như là một trường học khi xét đến ngữ cảnh sử dụng của hệ thống.

Bảng 3.1: *Bảng Organizations*

Tên trường	**Kiểu dữ liệu**	**Mô tả trường**
id	UUID, PK	
code	STRING(32)	Mã hiệu tổ chức
name	STRING(256)	Tên tổ chức
description	TEXT	Mô tả tổ chức
created_by	UUID	Người tạo ra tổ chức
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Users chứa thông tin của người dùng trong hệ thống.

Bảng 3.2: *Bảng Users*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
email	TEXT	Email của người dùng
username	TEXT	Username của người dùng
password	TEXT	Mật khẩu đã được mã hóa của người dùng
first_name	TEXT	Tên của người dùng
last_name	TEXT	Họ và tên lót của người dùng
is_active	BOOLEAN	Trạng thái của tài khoản người dùng
number_rand	INTEGER	Số ngẫu nhiên để tái lập mật khẩu
created_by	UUID	Người tạo ra tài khoản
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Courses chứa thông tin về các khóa học trong hệ thống.

Bảng 3.3: *Bảng Courses*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
code	STRING(32)	Mã hiệu môn học
name	STRING(256)	Tên môn học
description	TEXT	Mô tả môn học
start	TIMESTAMP	Thời điểm bắt đầu môn học
end	TIMESTAMP	Thời điểm kết thúc môn học
created_by	UUID	Người tạo ra môn học
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Roles chứa thông tin về quyền của người dùng trong hệ thống. Mỗi người dùng đều có một quyền nhất định trong hệ thống. Với người dùng là Super Admin thì quyền của họ là hợp lệ trên toàn bộ hệ thống. Còn đối với những người dùng khác thì quyền của họ sẽ ràng buộc trên từng khóa học, tức một người dùng có thể là người học trong khóa học này nhưng cũng có thể

là người dạy hoặc trợ giảng trong khóa học khác.

Bảng 3.4: *Bảng Roles*

Tên trường	**Kiểu dữ liệu**	**Mô tả trường**
id	UUID, PK	
organization_id	UUID, FK	ID của tổ chức
course_id	UUID, FK	ID của môn học
user_id	UUID, FK	ID của người dùng
role	INTEGER	Quyền của người dùng, các giá trị có thể có: • 1: Super admin • 2: Admin • 3: Lecturer • 4: Editor • 5: TA • 6: Student
created_by	UUID	Người thêm quyền cho người dùng
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Groups chứa thông tin về các nhóm người dùng theo từng khóa học.

Bảng 3.5: *Bảng Groups*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
course_id	UUID, FK	ID của môn học
code	STRING(32)	Mã nhóm
name	STRING(256)	Tên nhóm
description	TEXT	Mô tả nhóm
created_by	UUID	Người tạo nhóm
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Group_Users chứa thông tin về việc gán người dùng vào một nhóm nào đó.

Bảng 3.6: *Bảng Group_Users*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
group_id	UUID, FK	ID của nhóm
user_id	UUID, FK	ID của người dùng
created_by	UUID	Người thêm vào
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Exercises chứa thông tin về các bài thực hành theo từng khóa học.

Bảng 3.7: *Bảng Exercises*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
course_id	UUID, PK	ID của khóa học
code	STRING(32)	Mã hiệu bài thực hành
name	STRING(256)	Tên bài thực hành
description	TEXT	Mô tả bài thực hành
type	INTEGER	Kiểu của bài thực hành. Các giá trị có thể có: • 1: Bài tập thực hành • 2: Bài tự luyện tập
created_by	UUID	Người tạo bài thực hành
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Questions chứa thông tin về câu hỏi trong hệ thống.

Bảng 3.8: *Bảng Questions*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
folder_id	UUID, FK	ID của thư mục chứa câu hỏi
name	TEXT	Tên câu hỏi
level	INTEGER	Độ khó câu hỏi
keywords	TEXT	Từ khóa câu hỏi
description	TEXT	Mô tả câu hỏi
limit_mem	INTEGER	Giới hạn bộ nhớ khi chấm (đơn vị MB)
limit_time	INTEGER	Giới hạn thời gian chấm (đơn vị giây)
compile_args	TEXT	Các tham số khi biên dịch
hidden_testcases	TEXT	Các testcase bị ẩn đi

language	INTEGER	Ngôn ngữ của câu hỏi. Các giá trị có thể có: • 1: C++
initcode_name	TEXT	Tên của tập tin mã nguồn đề
initcode_uri	TEXT	Đường dẫn của tập tin mã nguồn đề
solution_name	TEXT	Tên của tập tin mã nguồn lời giải
solution_uri	TEXT	Đường dẫn của tập tin mã nguồn lời giải
expect_name	TEXT	Tên của tập tin kết quả mong đợi
expect_uri	TEXT	Đường dẫn của tập tin kết quả mong đợi
std_input_name	TEXT	Tên của tập tin đầu vào chuẩn
std_input_uri	TEXT	Đường dẫn của tập tin đầu vào chuẩn
file_input_name	TEXT	Tên của tập tin đầu vào bằng tập tin
file_input_uri	TEXT	Đường dẫn của tập tin đầu vào bằng tập tin
code_input_name	TEXT	Tên của tập tin đầu vào bằng mã
code_input_uri	TEXT	Đường dẫn của tập tin đầu vào bằng mã
created_by	UUID	ID của người tạo câu hỏi
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Assignments chứa thông tin về việc gán một câu hỏi trong bài thực hành vào nhóm người học.

Bảng 3.9: *Bảng Assignments*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
exercise_id	UUID, FK	ID của bài tập
question_id	UUID, FK	ID của câu hỏi
group_id	UUID, FK	ID của nhóm
order	INTEGER	Thứ tự của câu hỏi
max_submissions	INTEGER	Số lần nộp tối đa cho câu hỏi
start	TIMESTAMP	Thời điểm bắt đầu làm bài
end	TIMESTAMP	Thời điểm kết thúc làm bài
created_by	UUID	Người tạo thư mục
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Bảng Submissions chứa thông tin về các bài nộp trong hệ thống.

Bảng 3.10: *Bảng Submissions*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
user_id	UUID, FK	ID của người dùng nộp bài
assignment_id	UUID, FK	ID của phép gán
status	INTEGER	Trạng thái chấm bài
is_compile_error	INTEGER	Bài nộp có bị lỗi compile không
compile_error_message	TEXT	Lỗi khi compile bài nộp
num_testcases	INTEGER	Số testcases
num_passed_testcases	INTEGER	Số testcases mà bài nộp đạt được
uri	TEXT	Đường dẫn của file bài nộp
log_uri	TEXT	Đường dẫn của file log cho bài nộp
created_by	UUID, FK	Người nộp bài
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

Chương 4

Thiết kế và hiện thực các giải pháp

Sau khi có cái nhìn chung về hệ thống mà mình phát triển, nhóm bắt đầu tiến hành phân tích các giải pháp khác nhau nhằm hiện thực các mục tiêu đề ra. Công việc đầu tiên được trình bày là xem xét cách chấm bài cũ và đề ra giải pháp cải thiện khả năng chấm bài của hệ thống. Tiếp theo, nhóm tiến hành phân tích giải pháp ngân hàng câu hỏi để hỗ trợ cho việc lưu trữ câu hỏi với số lượng lớn. Công việc cuối cùng là phân tích giải pháp Xây dựng lộ trình thực hành giúp người học được hướng dẫn các câu hỏi phù hợp với khả năng bản thân. Ba công việc trên sẽ được trình bày chi tiết trong chương này.

4.1 Thay đổi thành phần chấm bài của hệ thống sang Jobe

Một trong những tồn đọng của hệ thống AGS chính là hiệu năng chấm bài còn chưa tốt và nhóm phải cải thiện nó thông qua việc đánh giá và sử dụng một con chấm mới với hiệu suất cao hơn và tối ưu hơn cho quá trình chấm bài. Nhóm sẽ lần lượt trình bày những phân tích về thành phần chấm bài cũ để chỉ ra những nguyên nhân vì sao cần được thay thế, sau đó chọn ra một con chấm mới để sử dụng và cuối cùng là thích nghi lại luồng thiết lập câu hỏi cũng như luồng chấm bài của hệ thống với con chấm mới này. Nhóm cũng thay đổi một số chi tiết khác như bộ nhớ đệm và giao thức liên lạc để tối ưu hơn hiệu năng chấm bài.

4.1.1 Phân tích về thành phần chấm bài cũ

Khi cân nhắc về kiến trúc của hệ thống, nhóm nhận được thông tin từ những người làm nên hệ thống AGS về thành phần chấm bài đang sử dụng nhiều tài nguyên về CPU hơn những thành phần khác. Vì đây là một thành phần quan trọng trong hệ thống, phục vụ cho số lượng lớn người học để thực hiện quá trình chấm bài tự động nên thành phần này đã phải được triển khai riêng trên một máy chủ khác để tránh ảnh hưởng lên các thành phần còn lại.

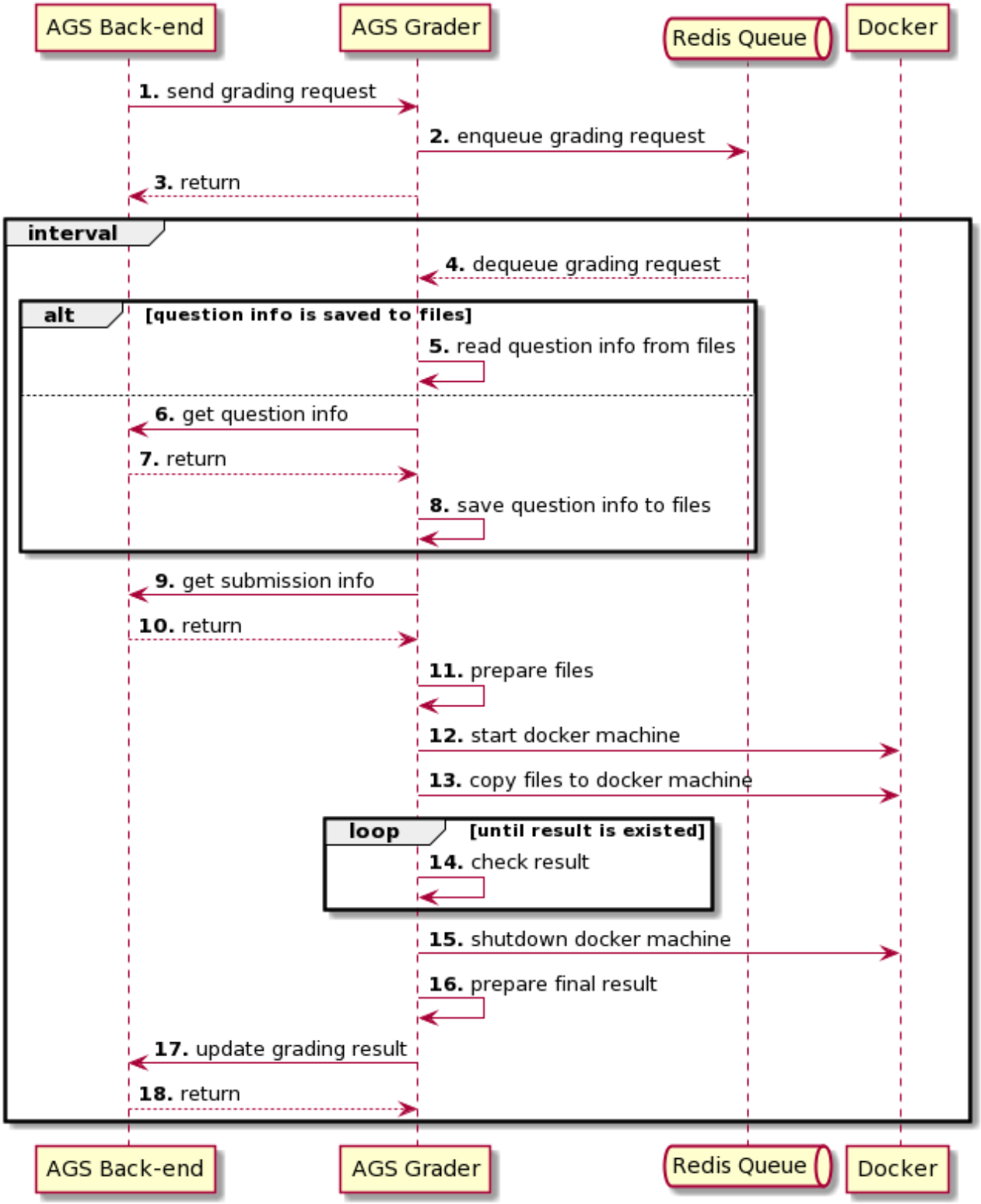
Song, nhóm đã tìm hiểu sâu hơn và biết được nguồn gốc của vấn đề là ở giải pháp được chọn lựa cho bước chấm bài, nên cho dù được triển khai riêng hay không thì giải pháp này vẫn cần

được cải tiến hoặc thay thế, để một bước quan trọng như bước chấm bài được hoạt động hiệu quả hơn.

Với bước chấm bài, tính độc lập giữa các bài nộp luôn phải được đảm bảo. Do vậy, giải pháp gốc được chọn lựa là sử dụng Docker để tạo một môi trường sandbox cho bước này. Chi tiết cho các bước trong quá trình chấm bài này như sau:

- Đầu tiên, người học nộp bài lên hệ thống và Back-end sẽ ghi nhận lại bài nộp với hai thông tin chính cần được quan tâm là question_id và submission_id.
- Sau đó, ở **bước 1** trong khâu chấm bài, Grader sẽ nhận được lời gọi chấm bài từ Back-end với 2 thông tin là question_id và submission_id như trên.
- **Bước 2 - 3**, chúng được đưa vào một hàng đợi (dựa trên nền tảng của Redis) và trả phản hồi về Back-end. Cơ chế này thể hiện rõ cách thức chấm bài bất đồng bộ của hệ thống, tức lời gọi chấm bài sẽ không chờ cho đến khi quá trình chấm bài được hoàn tất rồi mới trả phản hồi về.
- Hàng đợi được nhắc đến ở trên sẽ được Grader lấy ra từng phần tử sau mỗi 2 giây, và với một phần tử, Grader sẽ có thông tin của question_id và submission_id (**bước 4**).
- **Bước 5 - 8**, với question_id, Grader sẽ kiểm tra xem các file cần thiết cho câu hỏi này có xuất hiện trong hệ thống không, nếu có sẽ đọc các file này lên để có được những thông tin cần thiết. Nếu không, Grader sẽ gửi lời gọi sang Back-end để lấy những thông tin đó, và lưu lại vào các file trong hệ thống để sử dụng ở những lần sau. Các file được lưu như thế, đóng vai trò là cache cho hệ thống, nhằm lấy được những thông tin của câu hỏi được nhanh hơn.
- **Bước 9 - 10**, với submission_id, Grader sẽ luôn gửi lời gọi sang Back-end để lấy thông tin của bài nộp vì bài nộp sẽ luôn là những thông tin mới, không sử dụng cache như câu hỏi được.
- Sau khi đã có thông tin về câu hỏi và bài nộp, Grader sẽ chuẩn bị các file cần thiết dùng cho Docker trong quá trình chấm. Sau đó Grader sẽ khởi tạo Docker Machine và sao chép những file đã được chuẩn bị vào để bắt đầu chấm bài (**bước 11 - 13**).
- Grader sẽ liên tục kiểm tra cho đến khi file kết quả từ Docker Machine được ghi ra hệ thống. Docker Machine cũng sẽ được tắt đi sau bước này (**bước 14 - 15**).
- Cuối cùng (**bước 16 - 18**), Grader sẽ đọc file kết quả và chuẩn bị những thông tin về kết quả của bài nộp để gửi request về Back-end nhằm cập nhật kết quả trong cơ sở dữ liệu.

Quá trình trên được mô tả lại như hình 4.1.



Hình 4.1: *Luồng chấm bài cũ*

Luồng thực hiện như thế này tồn tại một vấn đề là chi phí sử dụng Docker Machine là rất lớn tại một bước quan trọng và phải chịu tải lớn như bước chấm bài. Với mỗi bài nộp lên, quá trình

khởi chạy và tắt Docker Machine cứ được lặp lại, ảnh hưởng lớn đến hiệu năng chấm bài của hệ thống.

4.1.2 Thay thế bằng thành phần chấm bài mới - Jobe

Không như Docker - với cơ chế bật tắt liên tục, Jobe - một con chấm khác đã được giới thiệu ở phần 2.3.7, hoạt động như một dịch vụ luôn chạy để chấm bài. Vì vậy cơ chế khác biệt như vậy nên về cảm quan, nhóm nhận thấy Jobe có khả năng mang lại hiệu năng tốt hơn giải pháp được sử dụng trước đó là Docker. Nhóm cũng đã thực hiện một đánh giá sơ bộ về hiệu năng của Jobe để chứng minh cho tính ứng dụng của Jobe trong thực tế ở phần B.

Cụ thể hơn, một số API mà Jobe cung cấp (Richard Lobb, 2021) để Grader có thể ứng dụng được như :

- API chấm bài:
  - Method: POST
  - Target resource: /jobe/index.php/restapi/runs/
  - Body parameters: run_spec

  API này được dùng để biên dịch và thực thi mã nguồn với những thông tin kèm theo trong tham số run_spec.

- API kiểm tra file có tồn tại trên Jobe:
  - Method: HEAD
  - Target resource: /jobe/index.php/restapi/files/
  - Request parameters: uniqueid

  API này được dùng để kiểm tra xem file có tồn tại trong Jobe hay không, và thường được sử dụng trong quá trình chấm bài cần kèm theo các file khác để làm testcase trong quá trình chấm bài. Tham số uniqueid biểu diễn định danh duy nhất cho một file. Nhóm quyết định sử dụng hàm băm MD5 lên nội dung file để lấy giá trị cho tham số uniqueid này.

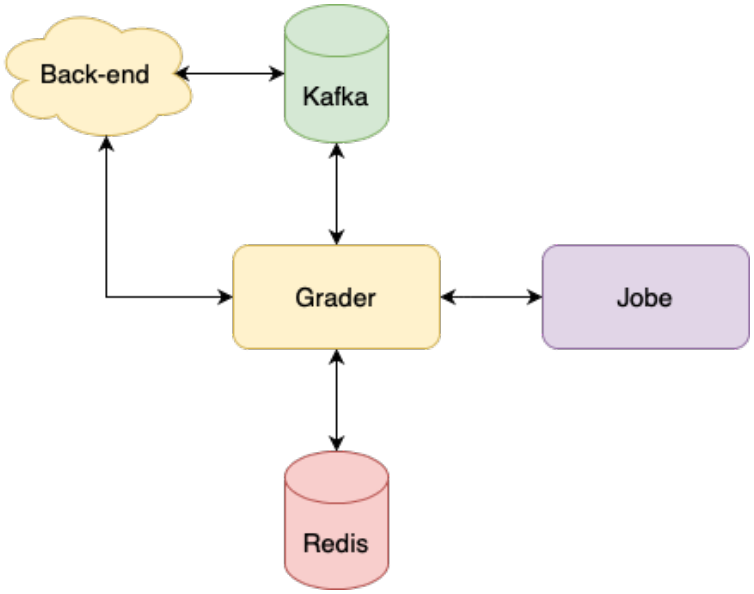
- API upload file lên Jobe
  - Method: PUT
  - Target resource: /jobe/index.php/restapi/files/
  - Request parameters: uniqueid

API này được dùng để tải file lên Jobe và ngữ cảnh sử dụng là tương tự với API trên. Luồng thực thi chấm bài thường gọi API kiểm tra file trước, nếu kết quả là không có thì Grader sẽ dùng API này để tải file lên Jobe. Tham số uniqueid được dùng tương tự như trên.

Một điểm đáng chú ý là Jobe chỉ có chức năng biên dịch và thực thi mã nguồn, sau đó trả về kết quả đầu ra chuẩn hoặc lỗi chuẩn, chứ không bao gồm chức năng so sánh với kết quả mong đợi. Điều này dẫn đến Jobe không thể cho ra điểm số cuối cùng cho bài nộp. Do vậy, hệ thống cần một thành phần để chuẩn bị dữ liệu gửi sang Jobe, cũng như xử lý kết quả mà Jobe trả về để hoàn tất khâu chấm bài.

Do quá trình chầm bài với con chấm Jobe có nhiều điểm khác biệt so với con chấm cũ, nhóm quyết định thay thế thành phần Grader cũ bằng một thành phần Grader hoàn toàn mới. Trong đó, Grader mới vẫn sẽ kết nối với Back-end để nhận các thông tin về câu hỏi và bài nộp, đồng thời thực hiện được hai chức năng mới là chuẩn bị dữ liệu gửi sang Jobe, cũng như xử lý kết quả mà Jobe trả về.

Kiến trúc mới cho khối Back-end, Grader và Jobe để thực hiện quá trình chấm bài như sau:



Hình 4.2: *Các thành phần trong luồng chấm bài mới*

4.1.3 Sử dụng phương thức giao tiếp bất đồng bộ mới - Kafka

Nhóm đánh giá cao giải pháp sử dụng mô hình giao tiếp bất đồng bộ giữa thành phần Back-end và Grader trong hệ thống cũ (bước 1-3 trong hình 4.1). Do chấm bài là quá trình phải chịu tải từ số lượng lớn người dùng một cách liên tục nên không thể thiết kế theo hướng gửi lời gọi và trả kết quả về ngay (mô hình giao tiếp đồng bộ), mà nên là gửi lời gọi và trả kết quả về sau (mô hình giao tiếp bất đồng bộ) để không gây áp lực lên con chấm. Vì lẽ đó, thành phần Grader

mới sẽ tiếp tục được thiết kế theo hướng giao tiếp bất đồng bộ với thành phần Back-end nhằm mang lại hiệu năng chấm bài tốt nhất.

Nhóm cũng thực hiện phần thay đổi cho giao thức kết nối giữa Back-end và Grader để tối ưu hiệu năng hơn. Thông qua việc sử dụng Kafka làm giao thức kết nối, thay cho REST API như trước. Lý do là Kafka có thể đóng vai trò như một hàng đợi thông điệp (message queue) hiệu năng cao (được giới thiệu tại mục 2.3.5), đáp ứng được nhu cầu tiếp nhận rất nhiều bài nộp của sinh viên cùng lúc mà vẫn giữ được thứ tự của chúng. Nhờ vậy, bài nộp nào được nộp trước sẽ được chấm trước, đảm bảo tính công bằng khi chấm bài cho sinh viên.

Nhóm nhận thấy trong quá trình chấm bài cũ, nội dung bài nộp không được gửi kèm cùng với lời gọi chấm bài mà Grader phải gọi API sang Back-end một lần nữa để lấy được nội dung (bước 9 trong hình 4.1). Do vậy nhóm muốn tối ưu thêm bằng cách gửi kèm nội dung bài nộp trong lời gọi chấm bài để hạn chế số lượng lời gọi API ra ngoài Internet.

Với cách tối ưu trên, cấu trúc của một thông điệp được gửi từ phía Back-end sang Grader để chấm bài như sau:

- question_id: Định danh cho câu hỏi, dùng cho bước lấy nội dung về câu hỏi khi cần.
- submission_id: Định danh cho bài nộp, dùng cho bước cập nhật điểm của bài nộp.
- submission_name: Tên của bài nộp, dùng trong lời gọi sang Jobe để chấm bài.
- submission: Nội dung của bài nộp.

Nội dung của câu hỏi không nằm trong cấu trúc trên vì đây là dữ liệu có thể tái sử dụng qua nhiều lần chấm bài. Do vậy, chỉ ở lần đầu tiên chấm bài cho một câu hỏi, Grader sẽ gọi API sang Back-end để lấy nội dung của câu hỏi và lưu vào bộ nhớ đệm để sử dụng cho lần sau mà không cần phải truyền lại trong mỗi thông điệp chấm bài.

4.1.4 Thay đổi nền tảng bộ nhớ đệm sang Redis

Xuyên suốt quá trình chấm bài, hai thông tin ít khi bị thay đổi là nội dung của câu hỏi và kết quả mong đợi của câu hỏi đó theo từng testcase. Do vậy, hai thông tin này nên được lưu trong bộ nhớ đệm để tiết kiệm thời gian chấm bài. Với nội dung của câu hỏi, việc caching giúp hạn chế số lần lấy nội dung câu hỏi từ phần Back-end. Còn về kết quả mong đợi thì việc caching sẽ giúp quá trình so sánh kết quả chấm bài với kết quả mong đợi được nhanh hơn.

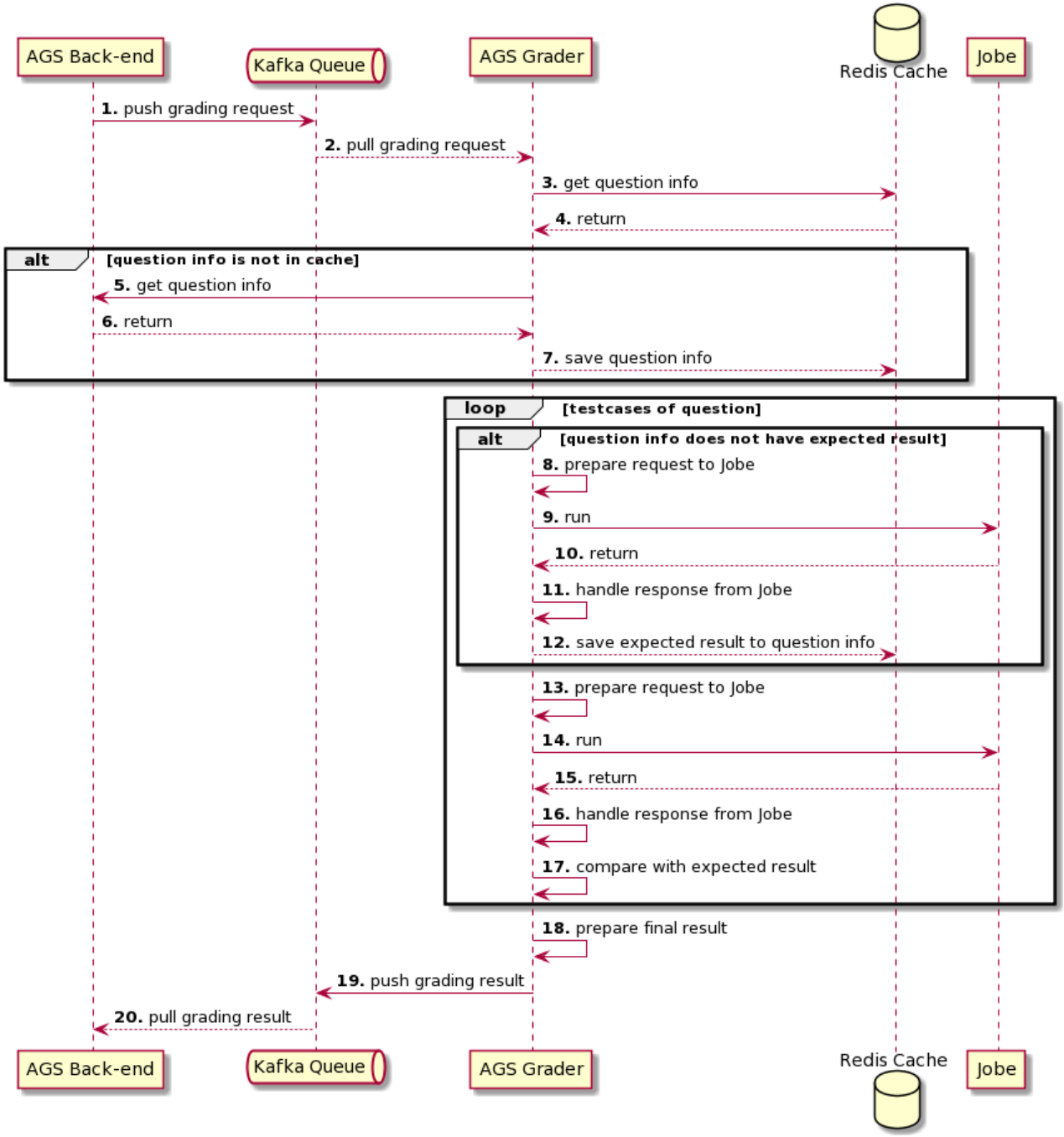
Nhóm sẽ sử dụng Redis như là nền tảng caching của hệ thống, vì đây là một trong những giải pháp caching có hiệu năng tốt và được sử dụng phổ biến trong thực tế (đã được giới thiệu ở 2.3.4).

4.1.5 Kết quả sau khi cải thiện

Luồng thực hiện quá trình chấm bài sử dụng con chấm mới là Jobe, và ứng dụng thêm các giải pháp mới như Kafka và Redis được mô tả lại như 4.3.

Chi tiết như sau luồng chấm bài mới như sau:

- **Bước 1 - 2:** Grader nhận lời gọi chấm bài từ Back-end thông qua Kafka.
- **Bước 3 - 7:** Grader kiểm tra nội dung của câu hỏi đã có trong Redis hay không. Nếu không thì gọi sang Back-end để lấy nội dung rồi cache lại trong Redis để sử dụng ở lần chấm bài sau.
- **Bước 8 - 12:** Grader kiểm tra trên Redis về kết quả kỳ vọng của testcase. Nếu không xuất hiện, Grader sẽ tiến hành chấm bài dựa theo lời giải của người dạy. Sau đó, Grader lưu lại kết quả này để so sánh với lần chấm bài kế tiếp mà không phải chấm lại lời giải.
- **Bước 13 - 17:** Grader tiến hành chấm bài theo bài nộp của sinh viên. Sau khi có kết quả chấm bài từ Jobe, Grader sẽ so sánh với kết quả mong đợi ở bước 8 - 12 để cho ra kết quả của bài nộp theo testcase.
- Các bước 8 - 12 và 13 - 17 sẽ được lặp lại theo số lượng testcase của câu hỏi.
- **Bước 18:** Grader sẽ tổng hợp điểm theo các testcase nhằm cho ra kết quả cuối cùng của bài nộp.
- **Bước 19 - 20:** Grader gửi trả kết quả của bài nộp về Back-end thông qua Kafka.



Hình 4.3: *Luồng chấm bài mới*

So sánh với luồng chấm bài cũ, quá trình chấm bài đã được hiện thực lại mới toàn bộ với sự trợ giúp của những công nghệ được bổ sung vào như Kafka, Redis và Jobe.

Đánh giá sơ bộ về kiến trúc ở bước chấm bài này, nhóm đã thực hiện ba thay đổi để cải thiện hiệu năng chấm bài của hệ thống:

- Thay đổi được thành phần chấm bài lõi từ Docker sang Jobe.
- Sử dụng Kafka làm giao thức kết nối giữa Back-end và Grader.

- Thay đổi cách lưu câu hỏi vào bộ nhớ đệm, sử dụng Redis thay vì tập tin.

4.1.6 Thêm mới những trường thông tin khi thiết lập câu hỏi

Jobe cung cấp khá nhiều tùy chọn khi chấm bài giúp đa dạng hóa các yêu cầu mà người dạy có thể đặt ra với câu hỏi. Bao gồm:

- Giới hạn bộ nhớ sử dụng khi chấm bài.
- Giới hạn thời gian chấm bài.
- Cho phép sử dụng tập tin đính kèm trong quá trình chấm bài.
- Thêm tham số khi biên dịch, thực thi mã nguồn.

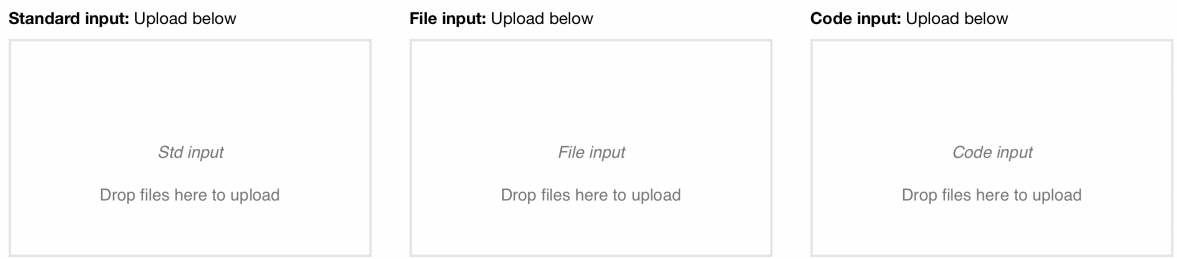
So sánh với các tùy chọn trong quá trình thiết lập câu hỏi trước đây, việc sử dụng Jobe đã cung cấp thêm hai tùy chọn mới để tích hợp vào khung thiết lập câu hỏi là giới hạn bộ nhớ sử dụng và tham số khi biên dịch, thực thi. Qua các tùy chọn này, người dạy có thể tăng cường khả năng kiểm soát quá trình chấm bài, giúp sinh viên nâng cao được chất lượng mã nguồn để đáp ứng kỳ vọng của người dạy.

Bên cạnh đó, nhóm cũng xem xét bổ sung thêm một số cách thức chấm bài khác bên cạnh cách thức sử dụng đầu vào chuẩn (standard input) mà khung chấm bài trước đây đã có. Đó là:

- Đầu vào bằng tập tin: Giúp người dạy kiểm tra được khả năng tương tác với tập tin (đọc, ghi, v.v...) của người học. Khả năng chấm bài bằng tập tin cũng được hỗ trợ bởi con chấm mới là Jobe nên nhóm quyết định sẽ tích hợp thêm cách thức chấm bài này vào khung thiết lập câu hỏi.
- Đầu vào bằng mã (testcode): Một kiểu testcase ngay trong mã nguồn của câu hỏi, giúp người dạy kiểm tra được bài nộp bằng cách thêm vào một đoạn mã nguồn khác chứ không chỉ đơn thuần là sử dụng đầu vào chuẩn hoặc tập tin.

  Nhóm nhận thấy việc tích hợp thêm cách thức chấm bài này cũng đơn giản nên đây sẽ là cách thức chấm bài thứ 3 mà khung thiết lập câu hỏi mới sẽ hỗ trợ.

Kết quả sau khi thêm các cách thức chấm bài mới vào khung thiết lập câu hỏi:



Hình 4.4: *Các cách thức chấm bài được hỗ trợ*

4.2 Thiết kế tính năng quản lý ngân hàng câu hỏi

4.2.1 Giới thiệu ngân hàng câu hỏi

Ngân hàng câu hỏi là nhu cầu cần thiết cho việc lưu trữ khi số lượng câu hỏi ngày càng lớn. Ngân hàng câu hỏi sẽ giúp người dùng có thể lưu trữ, quản lý và truy xuất các câu hỏi một cách có tổ chức. Một ngân hàng câu hỏi được tổ chức đủ tốt sẽ là nền tảng tốt để phát triển hệ thống hỗ trợ lập trình lớn. Một số lợi ích có thể kể đến của ngân hàng câu hỏi:

- Ngân hàng câu hỏi giúp người dạy có thể lưu trữ tất cả câu hỏi mà họ đã từng soạn. Lấy ví dụ cho trường hợp không có ngân hàng câu hỏi, nếu có nhu cầu sử dụng lại một câu hỏi cũ, người dạy sẽ phải bỏ công sức soạn đi soạn lại một câu hỏi vì câu hỏi này không được lưu lại. Mặt khác, nếu có ngân hàng câu hỏi, người dạy chỉ cần đi tìm kiếm lại trong kho lưu trữ là đã có thể sử dụng được ngay vì câu hỏi đã được lưu trong ngân hàng.
- Các câu hỏi sẽ có tính tái sử dụng cao. Một câu hỏi có thể được dùng cho nhiều bài thực hành khác nhau.
- Với một số lượng câu hỏi được tích lũy ngày càng lớn, người dùng sẽ gặp khó khăn khi tìm lại chính xác một câu hỏi nào đó. Một cơ chế tổ chức tốt sẽ giúp người soạn bài thực hành tìm ra các câu hỏi phù hợp cho bài thực hành.
- Nếu số lượng câu hỏi đủ lớn, ta có thể tạo tự động một bài thực hành được cấu thành từ rất nhiều câu trong ngân hàng câu hỏi. Ví dụ, hệ thống có thể tạo một bài thực hành gồm 30 câu ngẫu nhiêu (10 dễ, 10 trung bình, 10 khó) từ 1000 câu hỏi trong ngân hàng câu hỏi. Tuy nhiên, cần phải đảm bảo rằng độ khó của các bài thực hành không quá chênh lệch nhau. Nếu có một cách đảm bảo được vấn đề trên, phương pháp tạo bài thực hành tự động sẽ hạn chế được sự gian lận, sao chép bài giữa các người học.

Nhận thấy sự cấp thiết và tầm ảnh hưởng của ngân hàng câu hỏi, nhóm quyết định thiết kế thêm tính năng quản lý ngân hàng câu hỏi cho hệ thống.

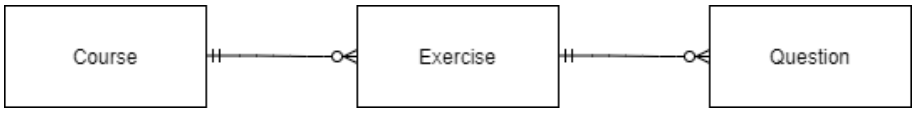
4.2.2 Phân tích cách tổ chức, quản lý ngân hàng câu hỏi của AGS và Moodle

Trước khi thiết kế hệ thống quản lý ngân hàng câu hỏi, nhóm tiến hành tham khảo, quan sát một số cách thức quản lý ngân hàng đã có và đưa ra nhận xét. Nhóm quyết định chọn phân tích 2 hệ thống có tính năng quản lý câu hỏi là AGS và Moodle. Phân tích của nhóm sẽ dựa trên 2 mặt là cách thức lưu trữ và các chức năng quản lý.

a) Cách quản lý câu hỏi trong AGS

AGS hiện tại chưa có một hệ thống quản lý ngân hàng đề một cách chính thống. Thay vào đó, AGS quản lý câu hỏi thông qua 3 lớp chính: khóa học, bài thực hành và câu hỏi. Từ đó, AGS cung cấp các chức năng giúp người dùng quản lý và lưu trữ các câu hỏi trong một bài thực hành.

Về mặt lưu trữ, mỗi câu hỏi sẽ thuộc về một bài thực hành. Mỗi bài thực hành thì chỉ nằm trong một khóa học. Thiết kế CSDL của AGS hiện tại cho tính năng quản lý câu hỏi được mô tả ở hình 4.5.



Hình 4.5: *Một phần của mô hình quan hệ thực thể của AGS về cách lưu trữ câu hỏi*

Về mặt chức năng, AGS cung cấp đủ các chức năng giúp người dạy có thể quản lý câu hỏi, cụ thể như là:

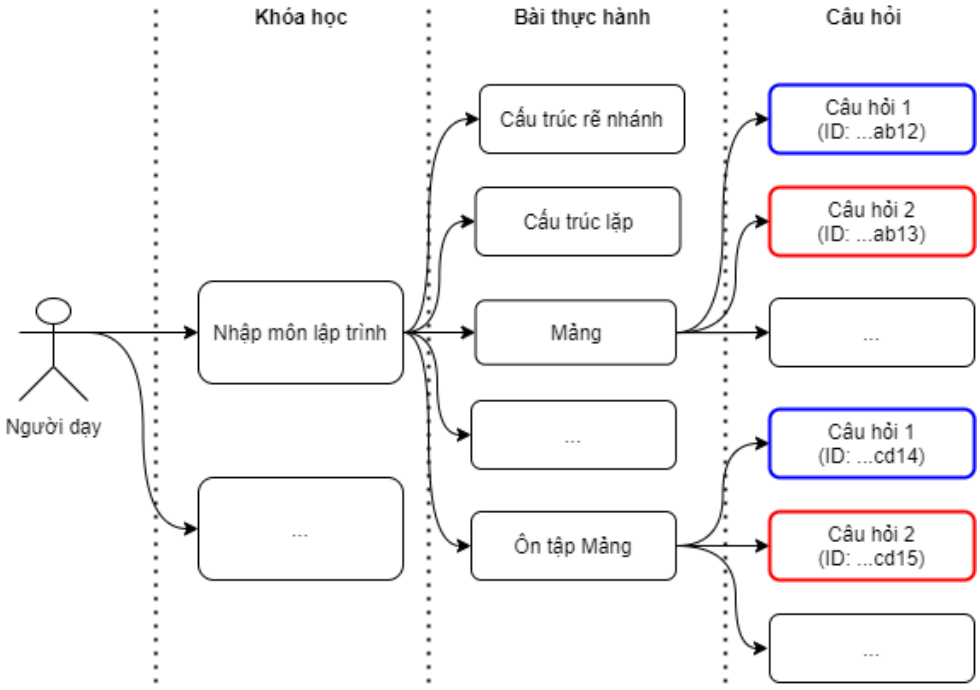
- Thêm một câu hỏi trong bài thực hành
- Xóa một câu hỏi trong bài thực hành
- Chỉnh sửa câu hỏi trong bài thực hành
- Xem trước một câu hỏi trong bài thực hành

Với cách hiện thực trên, vì không có ngân hàng câu hỏi nên một câu hỏi chỉ thuộc về một bài thực hành, một bài thực hành chỉ thuộc về một khóa học duy nhất. Điều này sẽ ảnh hưởng xấu đến tính mở rộng của hệ thống. Khi tạo một bài thực hành mới, ta phải tạo thêm những câu hỏi mới, không thể dùng lại những câu hỏi của một bài thực hành cũ, dẫn đến dư thừa dữ liệu nếu muốn tái sử dụng.

Lấy ví dụ, sự dư thừa này có thể được minh họa thông qua hình 4.6. Với mục đích tạo một bài thực hành *Ôn tập Mảng*, người dạy muốn sử dụng lại *Câu hỏi 1* đã được soạn trong bài thực hành *Mảng*. Để có thể làm được điều này, người dạy phải tạo một *Câu hỏi 1* mới trong bài thực hành *Ôn tập Mảng* nhưng nội dung thì không khác so với câu hỏi trong bài thực hành *Mảng*. Điều này dẫn đến cơ sở dữ liệu phải lưu trữ 2 bản ghi khác nhau cho 1 câu hỏi giống hệt nhau về nội dung.

b) Tính năng Ngân hàng câu hỏi trong một khóa học của Moodle

Ngân hàng câu hỏi là tính năng mà Moodle cung cấp cho người dạy giúp cho họ có thể tạo, xem trước, xóa, sửa các câu hỏi của mình trong một cơ sở dữ liệu câu hỏi. Ngân hàng câu hỏi này sẽ thuộc về một khóa học nhất định. Sau khi đã thêm vào ngân hàng, câu



Hình 4.6: *Cách quản lý câu hỏi của AGS*

hỏi có thể được thêm vào một bài thực hành bất kỳ mà người dạy mong muốn (Moodle Question Bank, 2021).

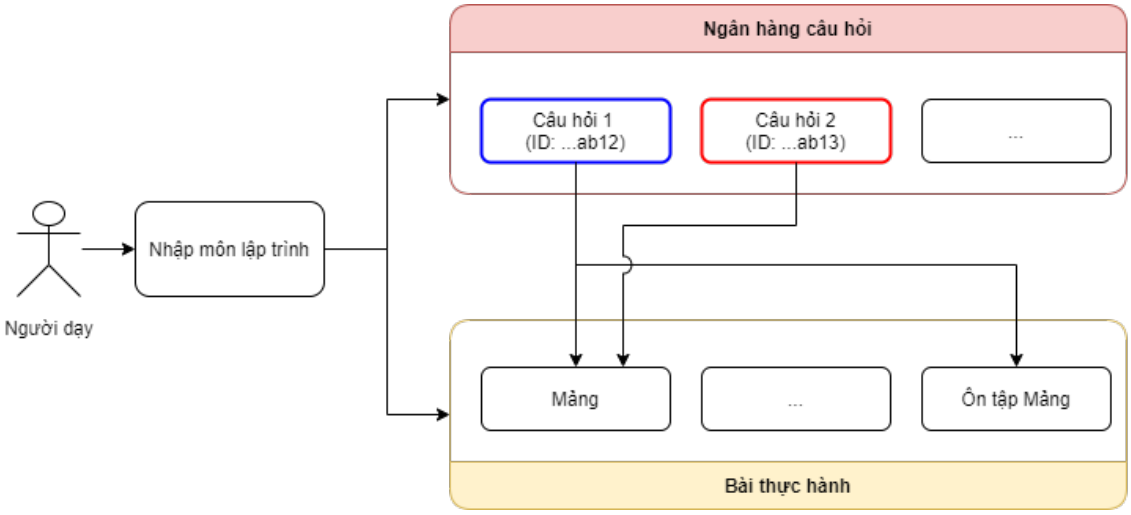
Về mặt lưu trữ, nhờ vào ngân hàng câu hỏi, giờ đây mỗi câu hỏi sẽ thuộc về nhiều bài thực hành. Bên cạnh đó, Moodle cũng cung cấp một tính năng là Thể loại (Category) cho ngân hàng câu hỏi. Mỗi Thể loại có thể chứa nhiều câu hỏi con cũng như thể loại con. Tính năng này sẽ giúp người dùng tự sắp xếp các câu hỏi của mình vào các Thể loại khác nhau để dễ tìm kiếm lại.

Về mặt chức năng, Moodle cung cấp nhiều chức năng hơn AGS vì có thêm các thao tác cho ngân hàng câu hỏi, cụ thể là:

- Thêm/xóa/sửa/xem trước một câu hỏi trong ngân hàng câu hỏi
- Thêm/sửa/chọn một thể loại câu hỏi
- Chia sẻ ngân hàng câu hỏi
- Thêm một câu hỏi vào bài thực hành từ ngân hàng

Tính năng ngân hàng câu của Moodle đã giúp câu hỏi đã có tính tái sử dụng. Một câu hỏi khi được tạo ra sẽ không thuộc về một bài thực hành nào mà sẽ có thể nằm trong nhiều bài thực hành.

Lấy ví dụ cũ đã được đề cập ở mục 4.2.2**a)**, người dạy muốn tạo một bài thực hành để ôn tập kiến thức Mảng. Đồng thời trước đó, người dạy đã có một bài thực hành chủ đề Mảng và các câu hỏi trong bài này đã được lưu trữ trong ngân hàng câu hỏi. Giờ đây, nếu muốn



Hình 4.7: *Tính tái sử dụng của ngân hàng câu hỏi Moodle*

cho sinh viên làm lại những câu hỏi cũ, người dạy có thể truy cập vô ngân hàng câu hỏi và sau đó chọn lại các câu hỏi cũ để thêm vào bài thực hành Ôn tập Mảng. Hình 4.7 cho thấy rằng, khác với AGS, Câu hỏi 1 giờ đây có tính tái sử dụng và không cần phải tạo câu hỏi mới. Cơ sở dữ liệu lúc bấy giờ chỉ cần một bản ghi cho *Câu hỏi 1* nhưng câu hỏi này có thể sử dụng đến 2 lần.

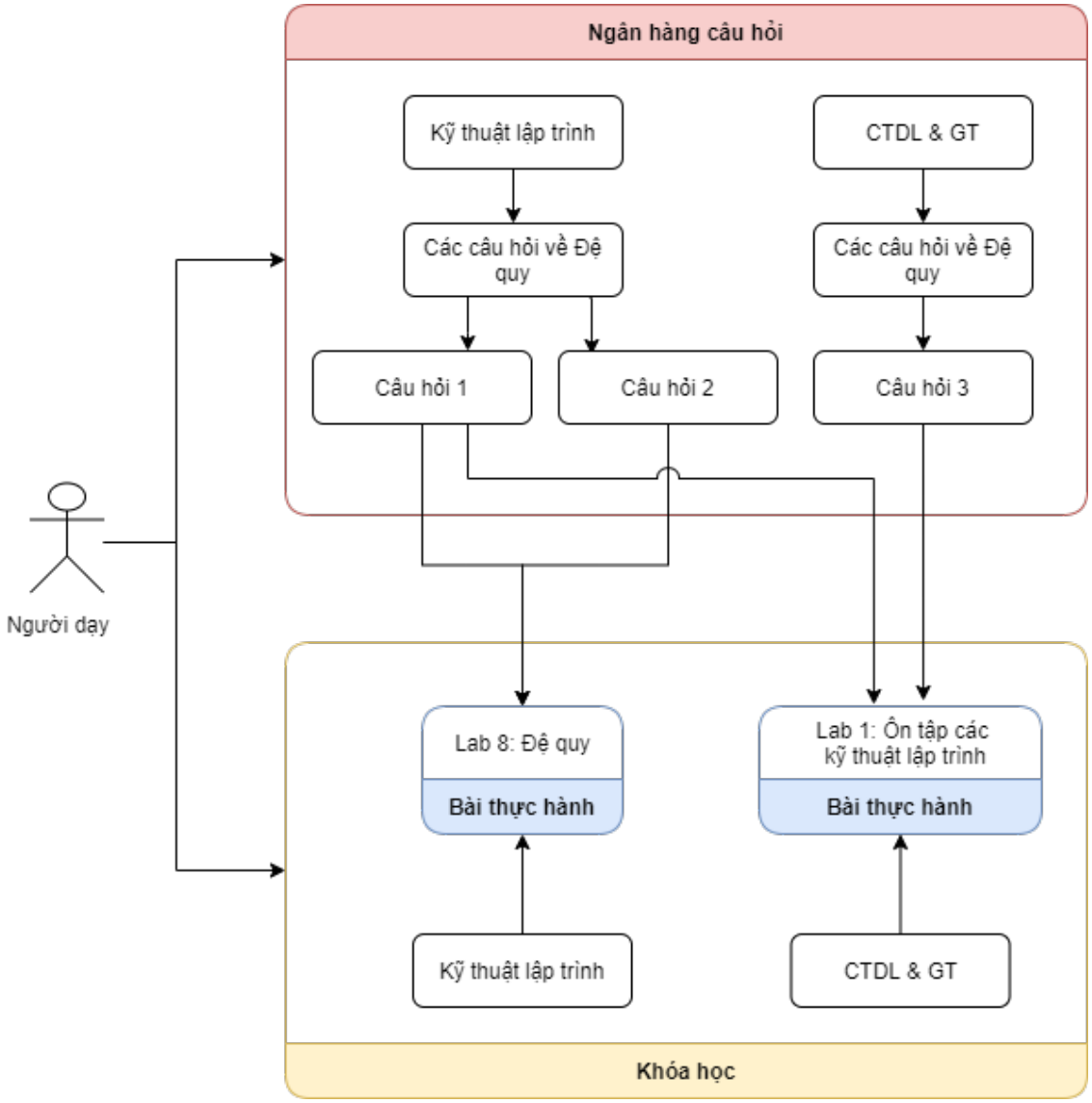
4.2.3 Tổng quan về tính năng quản lý ngân hàng câu hỏi của hệ thống

Sau khi đã khảo sát về tình trạng hiện tại của AGS cũng như học hỏi các tính năng hiện có của Moodle, nhóm quyết định phát triển thêm cho hệ thống hiện tại tính năng quản lý ngân hàng câu hỏi.

Trong phiên bản mới được phát triển từ AGS, học hỏi Moodle, nhóm sẽ phát triển một ngân hàng câu hỏi riêng biệt để lưu trữ câu hỏi. Các bài thực hành giờ không còn là nơi lưu trữ câu hỏi mà sẽ được cấu thành từ các câu hỏi trong ngân hàng.

Tuy nhiên, khác với Moodle, ngân hàng câu hỏi của nhóm sẽ không nằm trong hay thuộc về một khóa học. Thay vào đó, ngân hàng câu hỏi này sẽ thuộc về một tổ chức (ví dụ như khoa Máy tính), tức là các câu hỏi sẽ được chia sẻ trong phạm vi một tổ chức. Lý do nhóm hiện thực điều này là vì các khóa học trong phạm vi do nhóm xác định (ở phần 1.5) là những môn học lập trình cơ bản. Các câu hỏi của những môn học này thường có thể sử dụng xen lẫn nhau. Lấy ví dụ, chủ đề Đệ quy của môn Kỹ thuật lập trình (KTLT) có thể được dùng để ôn tập các kiến thức lập trình cho môn Cấu trúc dữ liệu và Giải thuật (CTDL&GT)).

Hình 4.8 mô tả cụ thể hơn cho ví dụ trên cũng như trình bày tổng quan về ngân hàng câu hỏi của hệ thống. Từ đó, ngân hàng câu hỏi mà nhóm đề xuất về lâu dài sẽ tạo thành một kho câu hỏi lập trình ngày càng lớn.



Hình 4.8: *Tổng quan ngân hàng câu hỏi của hệ thống*

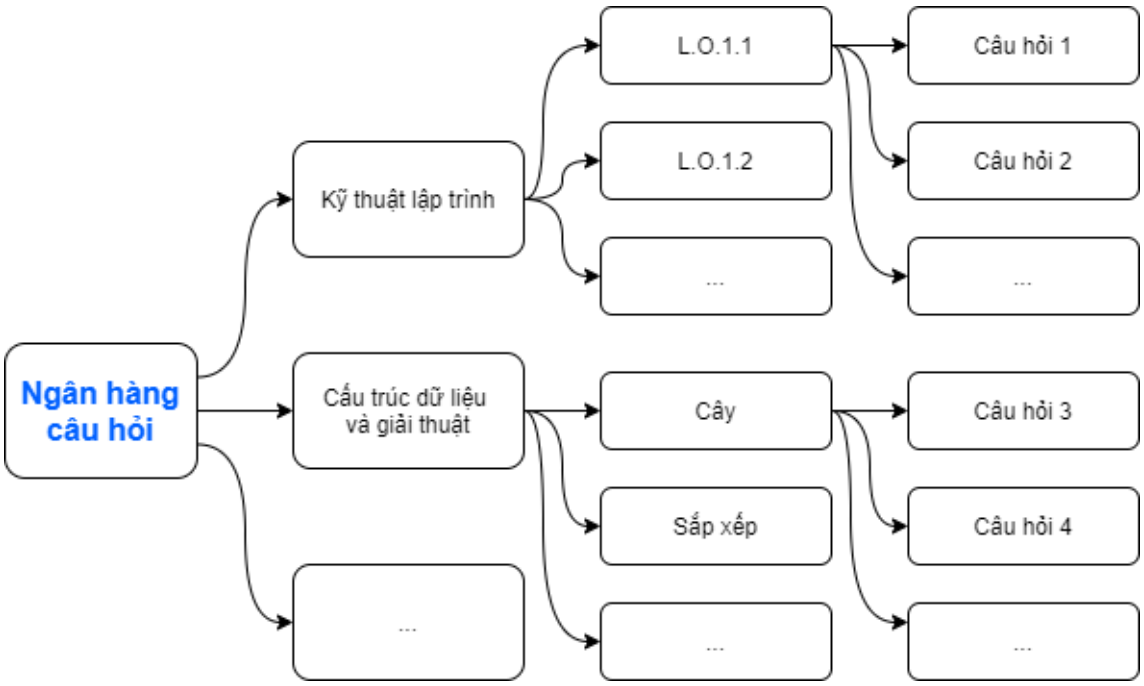
Các phần sau sẽ lần lượt giới thiệu về các tính năng mà nhóm cung cấp đối với ngân hàng câu hỏi này. Đầu tiên, mục 4.2.4 sẽ trình bày về thiết kế cây phân cấp thư mục giúp quản lý ngân hàng câu hỏi một cách có tổ chức. Kế đến, nhóm đề xuất một phương pháp giúp chia sẻ giữa các người dạy thông qua ngân hàng chung và ngân hàng riêng ở mục 4.2.5. Cuối cùng, sơ đồ use-case và bảng chi tiết các chức năng quản lý ngân hàng câu hỏi sẽ được trình bày ở mục 4.2.6.

4.2.4 Cây phân cấp thư mục quản lý ngân hàng câu hỏi

Sau khi đã tách các câu hỏi ra một nơi riêng biệt, các câu hỏi này cần một sự quản lý, sắp xếp về mặt lưu trữ sao cho phù hợp với đặc thù giảng dạy của người dạy như: các khóa học, các học kỳ, các bài lab, v.v... Tham khảo ngân hàng câu hỏi của Moodle về tính năng Category,

nhóm đề xuất việc tạo một cây phân cấp thư mục để chứa các câu hỏi. Cây thư mục do nhóm đề xuất sẽ có các tính chất sau:

- Mỗi nút của cây có thể là một câu hỏi hoặc một thư mục.
- Các nút thư mục sẽ có bậc không cố định (tức là nút này có thể có vô số nút thư mục và vô số câu hỏi là con của nó).
- Các nút câu hỏi sẽ là nút lá.
- Nút gốc của cây là ngân hàng câu hỏi.



Hình 4.9: *Mô hình cây phân cấp thư mục câu hỏi*

Với cách xây dựng cây như trên, ngân hàng câu hỏi có thể đáp ứng hầu hết các nhu cầu lưu trữ của người dạy. Hình 4.9 trình bày một cấu trúc lưu trữ ngân hàng câu hỏi mà nhóm lấy làm ví dụ. Trong ngân hàng câu hỏi này, tầng đầu tiên của cây sẽ là môn học, kế đến là các chủ đề của môn hoặc bài lab và cuối cùng là các câu hỏi. Cụ thể, môn CTDL&GT sẽ có các chủ đề như Cây, Giải thuật sắp xếp, v.v... Trong chủ đề Cây sẽ là các câu hỏi như Câu hỏi 3, Câu hỏi 4. Tuy nhiên, nếu người dạy muốn tổ chức một lớp lưu trữ khác, lấy ví dụ là các chuẩn đầu ra (L.O.1.1, L.O.2.2, ...). Người dạy có thể hoàn toàn tạo thêm một số thư mục như vậy như việc tổ chức của môn học KTLT trong hình 4.9.

4.2.5 Ngân hàng câu hỏi chung và riêng

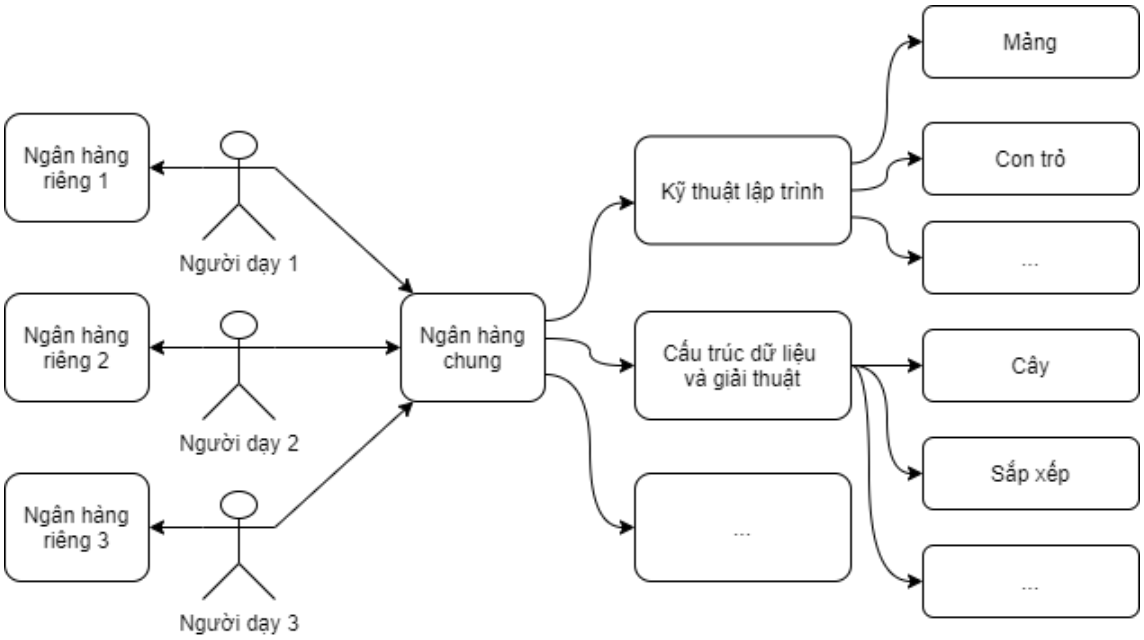
Khi thiết kế ngân hàng câu hỏi, ngoài tạo ra một nơi để lưu trữ câu hỏi, nhóm còn hướng đến giải pháp giúp những người dạy có thể chia sẻ câu hỏi với nhau. Thông thường, trong công

việc giảng dạy thực hành ở các môn lập trình cơ bản của khoa Khoa học và Kỹ thuật Máy tính, Đại học Bách Khoa, các trợ giảng sẽ có một số bài lab được soạn sẵn bởi một nhóm ra đề. Điều này sinh ra nhu cầu cần có một nơi lưu trữ câu hỏi mà mọi người dạy đều được chia sẻ và sử dụng. Để giải quyết nhu cầu này, nhóm đề xuất một ngân hàng chung có thể được sử dụng bởi tất cả người dạy trong cùng một tổ chức.

Tuy nhiên, nếu một người dạy nào đó muốn tạo một câu hỏi mà không muốn chia sẻ với người dạy khác (giả dụ như câu hỏi nháp chưa đảm bảo về chất lượng) thì chỉ duy nhất một ngân hàng được sử dụng chung sẽ không đáp ứng được nhu cầu này. Điều này dẫn đến nhu cầu một nơi lưu trữ câu hỏi cá nhân riêng biệt, tránh ảnh hưởng đến ngân hàng câu hỏi chung.

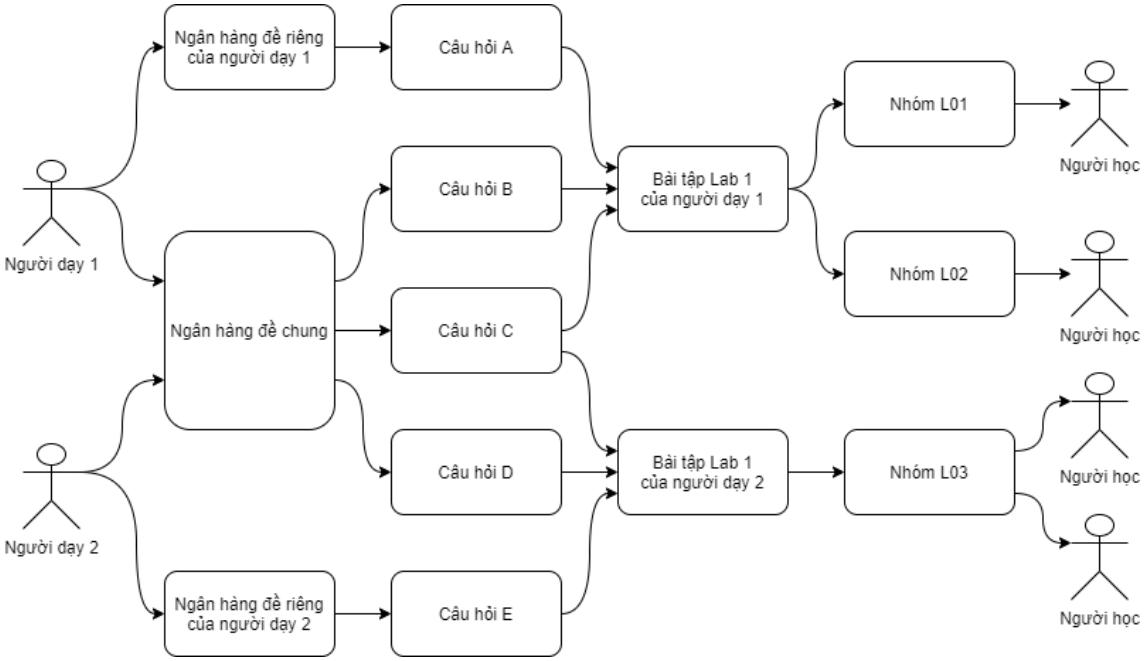
Vì vậy, nhóm đề xuất giải pháp ngân hàng chung và riêng cho ngân hàng câu hỏi của hệ thống, trong đó:

- **Ngân hàng chung**: Nơi các người dạy trong cùng một tổ chức có thể lưu trữ, chia sẻ các câu hỏi được sử dụng chung cho việc giảng dạy.
- **Ngân hàng riêng**: Nơi lưu trữ các câu hỏi riêng của mỗi người dạy, chỉ có chính họ mới được truy cập.



Hình 4.10: *Minh họa về ngân hàng câu hỏi chung và riêng*

Giải pháp này có thể minh họa thông qua hình 4.10. Trong đó, tất cả người dạy 1, 2 và 3 đều có thể sử dụng ngân hàng câu hỏi chung. Song song, mỗi người dạy đều có một ngân hàng câu hỏi riêng để có thể lưu trữ câu hỏi của cá nhân.



Hình 4.11: *Ứng dụng của ngân hàng câu hỏi chung và riêng*

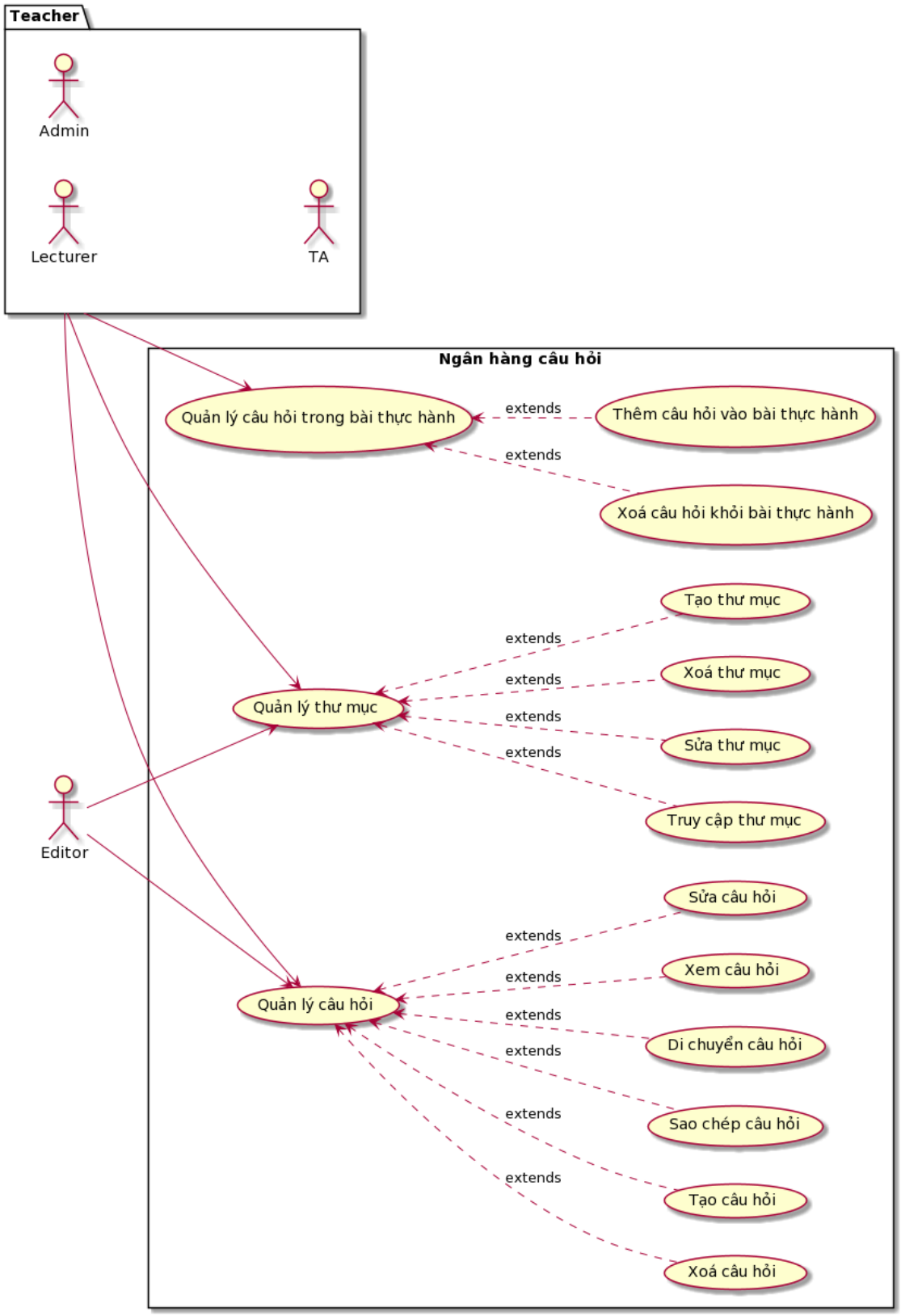
Để thấy được tính ứng dụng của ngân hàng chung và riêng, ta lấy một ví dụ minh họa cụ thể bằng hình 4.11. Trong hình này, nhóm xin phép bỏ qua phần cây phân cấp thư mục để diễn đạt những ý chính. Người dạy 1 và người dạy 2 đều có thể truy cập vào ngân hàng câu hỏi chung, đã được tạo sẵn các câu hỏi của nhóm ra đề. Tuy nhiên, để điều chỉnh độ khó hoặc nội dung của bài tập phù hợp với lớp giảng dạy, người dạy 1 và 2 đã lần lượt tạo các câu hỏi A và E trong ngân hàng riêng của họ. Sau đó, câu hỏi A, B và C được chọn để tạo thành một bài thực hành của riêng người dạy 1 và gán cho các nhóm lớp L01 và L02. Điều này cũng tương tự cho người dạy 2 với các câu hỏi C, D và E.

4.2.6 Thiết kế các chức năng quản lý ngân hàng câu hỏi

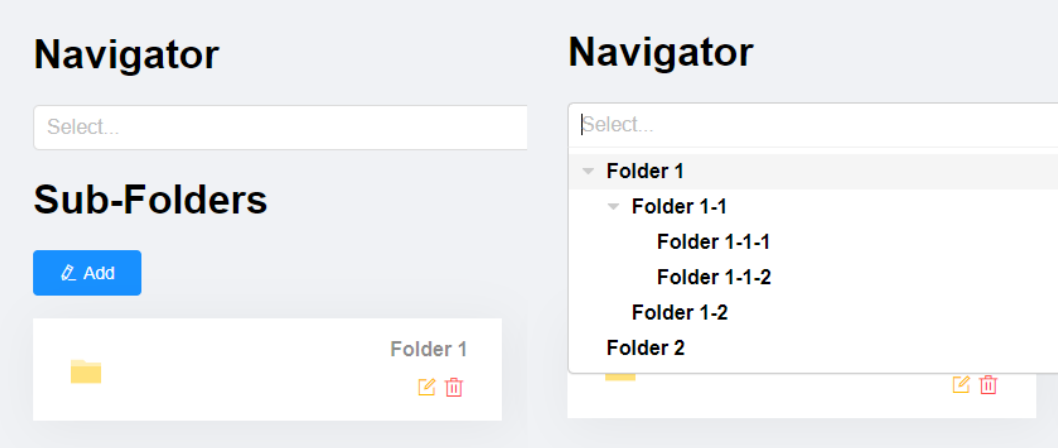
Trong mục này, nhóm xin phép trình bày chi tiết các chức năng mà người dùng có thể thao tác với ngân hàng câu hỏi của hệ thống. Trước tiên, để có cái nhìn chung về tính năng quản lý ngân hàng câu hỏi, nhóm sử dụng sơ đồ use-case ở hình 4.12 để giới thiệu các chức năng.

Đối với các chức năng quản lý thư mục, nhóm xin phép được trình bày chi tiết các chức năng thông qua các bảng use-case dưới đây:

- **Truy cập thư mục:** Giao diện và cách truy cập được nhóm thiết kế ở hình dưới. Nhóm cung cấp người dùng 2 cách truy cập, hoặc thông qua cây phân cấp, hoặc thông qua truy cập từng thư mục.



Hình 4.12: *Sơ đồ use-case của tính năng quản lý ngân hàng câu hỏi*



Hình 4.13: *Hai cách truy cập thư mục trong ngân hàng câu hỏi*

Bảng 4.1: *Bảng mô tả use-case cho chức năng Truy cập thư mục*

Tên use-case	Truy cập thư mục
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng truy cập vào một thư mục, hiển thị các thư mục con và câu hỏi bên trong.
Tiền điều kiện	Người dùng đang ở màn hình của một thư mục trong Ngân hàng câu hỏi
Luồng đi thông thường	1. Người dùng nhấn vào một thư mục con, hoặc nhấn vào một thư mục trong thanh điều hướng. 2. Các thư mục con và câu hỏi của thư mục được chọn được hiển thị cho người dùng.
Ngoại lệ	Không
Luồng đi thay thế	Không

- **Tạo thư mục:** Khi tạo một thư mục, người dùng nhập tên, vị trí mong muốn và một thư mục trống sẽ được tạo ra.

Bảng 4.2: *Bảng mô tả use-case cho chức năng Tạo thư mục*

Tên use-case	Tạo thư mục
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng tạo một thư mục để chứa câu hỏi.
Tiền điều kiện	Người dùng đang ở màn hình của một thư mục trong Ngân hàng câu hỏi.
Luồng đi thông thường	1. Người dùng nhấn nút *Add*. 2. Hộp thoại hiện ra yêu cầu người dùng nhập tên thư mục. 3. Người dùng nhập tên thư mục. 4. Người dùng nhấn nút *Confirm*. 5. Một thư mục con thêm vào bên trong thư mục hiện tại.
Ngoại lệ	*Ngoại lệ 1: bước 4* 4a. Nếu người dùng tạo một thư mục con với tên đã tồn tại, hệ thống hiện thông báo đến người dùng.
Luồng đi thay thế	*Thay thế 1: bước 4* 4a. Người dùng nhập một tên khác. Sau đó tiếp tục bước 4 trong **Luồng đi thông thường**.

- **Sửa thư mục:** Khi sửa một thư mục, người dùng được chỉnh sửa tên, nội dung, v.v... của thư mục, ngoài ra còn được phép thay đổi vị trí của thư mục bằng cách thay đổi thư mục cha của nó. Hệ thống cũng sẽ ngăn không cho việc thư mục cha di chuyển vô trong thư mục con, một hành vi không hợp lí.

Bảng 4.3: *Bảng mô tả use-case cho chức năng Sửa thư mục*

Tên use-case	Sửa thư mục
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng chỉnh sửa tên thư mục hoặc di chuyển thư mục đến một vị trí khác.
Tiền điều kiện	Người dùng đang ở màn hình của một thư mục trong Ngân hàng câu hỏi.

Luồng đi thông thường	1. Người dùng nhấn vào icon *Edit*. 2. Hộp thoại hiện ra các thông tin hiện tại của thư mục gồm: tên, thư mục cha. 3. Người dùng thực hiện chỉnh sửa. 4. Người dùng nhấn nút *Confirm*. 5. Thư mục được cập nhật.
Ngoại lệ	*Ngoại lệ 1: bước 4* 4a. Nếu người dùng thay đổi tên đã tồn tại trong thư mục cha, hệ thống hiện thông báo đến người dùng. *Ngoại lệ 2: bước 4* 4b. Nếu vị trí được di chuyển tới là một thư mục con của thư mục hiện tại, hệ thống hiện thông báo đến người dùng.
Luồng đi thay thế	*Thay thế 1: bước 4* 4a. Người dùng nhập một tên khác. Sau đó tiếp tục bước 4 trong **Luồng đi thông thường**. *Thay thế 2: bước 4* 4b. Người dùng nhập chọn một điểm đến khác. Sau đó tiếp tục bước 4 trong **Luồng đi thông thường**.

- **Xóa thư mục:** Ta cung cấp 2 hành vi cho người dùng khi xóa thư mục. Đầu tiên là xóa tất cả nội dung bên trong, cả thư mục con lẫn câu hỏi. Thứ hai là di chuyển tất cả thư mục và câu hỏi trong thư mục chuẩn bị xóa sang một thư mục khác.

Bảng 4.4: *Bảng mô tả use-case cho chức năng Xóa thư mục*

Tên use-case	Xóa thư mục
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng xóa một thư mục, người dùng có thể chọn giữa việc xóa tất cả nội dung bên trong (bao gồm thư mục con và các câu hỏi) hoặc di chuyển tất cả nội dung này sang một thư mục khác.

Tiền điều kiện	Người dùng đang ở màn hình của một thư mục trong Ngân hàng câu hỏi.
Luồng đi thông thường	1. Người dùng nhấn vào icon *Delete*. 2. Hộp thoại hiện ra để người dùng chọn xóa tất cả hoặc xóa và di chuyển nội dung. 3. Nếu người chọn xóa tất cả. 3.1. Thư mục và tất cả nội dung bên trong bị xóa. 4. Nếu người chọn một thư mục muốn chuyển tới và nhấn xóa. 4.1. Thư mục bị xóa, tất cả nội dung được chuyển sang thư mục chỉ định.
Ngoại lệ	*Ngoại lệ 1: bước 4* 4a. Nếu vị trí được di chuyển tới là một thư mục con của thư mục hiện tại, hệ thống hiện thông báo đến người dùng.
Luồng đi thay thế	*Thay thế 1: bước 4* 4a. Người dùng nhập chọn một điểm đến khác. Sau đó tiếp tục bước 4 trong **Luồng đi thông thường**.

Đối với các chức năng quản lý câu hỏi, để ngân hàng câu hỏi chung và riêng hoạt động tốt và hỗ trợ lẫn nhau, nhóm có tạo thêm chức năng Sao chép/Di chuyển câu hỏi, trong đó:

- Sao chép câu hỏi: Chức năng này sẽ giúp người dùng có thể đưa câu hỏi từ ngân hàng chung về ngân hàng riêng hoặc ngược lại dưới dạng một bản sao. Hành vi sao chép từ ngân hàng chung sang ngân hàng riêng sẽ cần thiết khi người dùng muốn chỉnh sửa một câu hỏi mà tránh làm ảnh hưởng ngân hàng chung. Ngược lại, hành vi sao chép từ ngân hàng riêng sang ngân hàng chung sẽ thực hiện mục đích chia sẻ câu hỏi của mình với cộng đồng.
- Di chuyển câu hỏi: Chức năng này sẽ giúp người dùng có thể di chuyển một câu hỏi sang một thư mục khác.

Chi tiết của chức năng Sao chép/Di chuyển câu hỏi được trình bày qua bảng 4.5. Các chức năng quản lý câu hỏi còn lại (trong sơ đồ use-case 4.12) đã có trong hệ thống cũ nên nhóm xin phép không trình bày.

Bảng 4.5: *Bảng mô tả use-case cho chức năng Sao chép/Di chuyển câu hỏi*

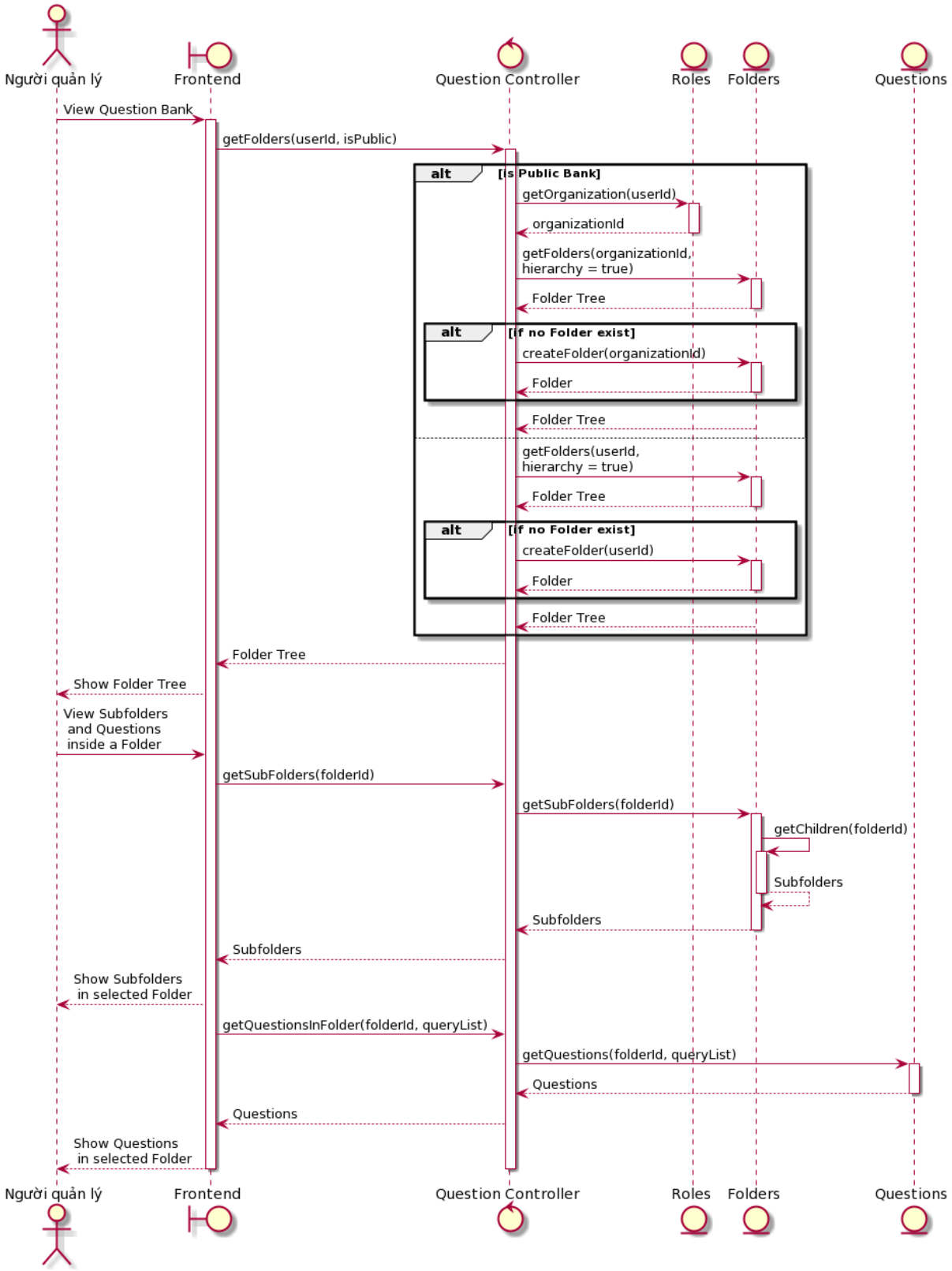
Tên use-case	Sao chép/Di chuyển câu hỏi
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng tạo một bản sao của câu hỏi hoặc di chuyển câu hỏi đến một thư mục khác.
Tiền điều kiện	Người dùng đang ở màn hình của một thư mục trong Ngân hàng câu hỏi.
Luồng đi thông thường	1. Người dùng nhấn vào icon *Sao chép/Di chuyển*. 2. Hộp thoại hiện ra để người dùng chọn điểm đến của bản sao/câu hỏi bị di chuyển. 3. Người dùng nhấn nút *Confirm*. 4. Nếu người dùng sao chép. 4.1. Một bản sao của câu hỏi được tạo trong thư mục được chỉ định. 5. Nếu người dùng di chuyển. 5.1. Câu hỏi được di chuyển đến thư mục được chỉ định.
Ngoại lệ	*Ngoại lệ 1: bước 4* 4a. Nếu thư mục được chọn đã tồn tại một câu hỏi có tên giống với bản sao, đi đến **Thay thế 1**. *Ngoại lệ 2: bước 5* 5a. Nếu vị trí được di chuyển tới đã tồn tại một câu hỏi có tên giống với câu hỏi sắp di chuyển tới, hệ thống hiện thông báo đến người dùng.
Luồng đi thay thế	*Thay thế 1: bước 4* 4a. Tên của câu hỏi được gắn thêm chuỗi " - Copy", việc tạo bản sao diễn ra như bình thường.

Sau khi ngân hàng câu hỏi được phát triển, các bài thực hành sẽ được tạo thành bằng cách lựa chọn các câu hỏi trong ngân hàng. Chức năng Thêm/Xóa một câu hỏi trong bài thực hành sẽ được cập nhật lại. Nhóm xin trình bày chi tiết use-case cho chức năng này thông qua bảng 4.6.

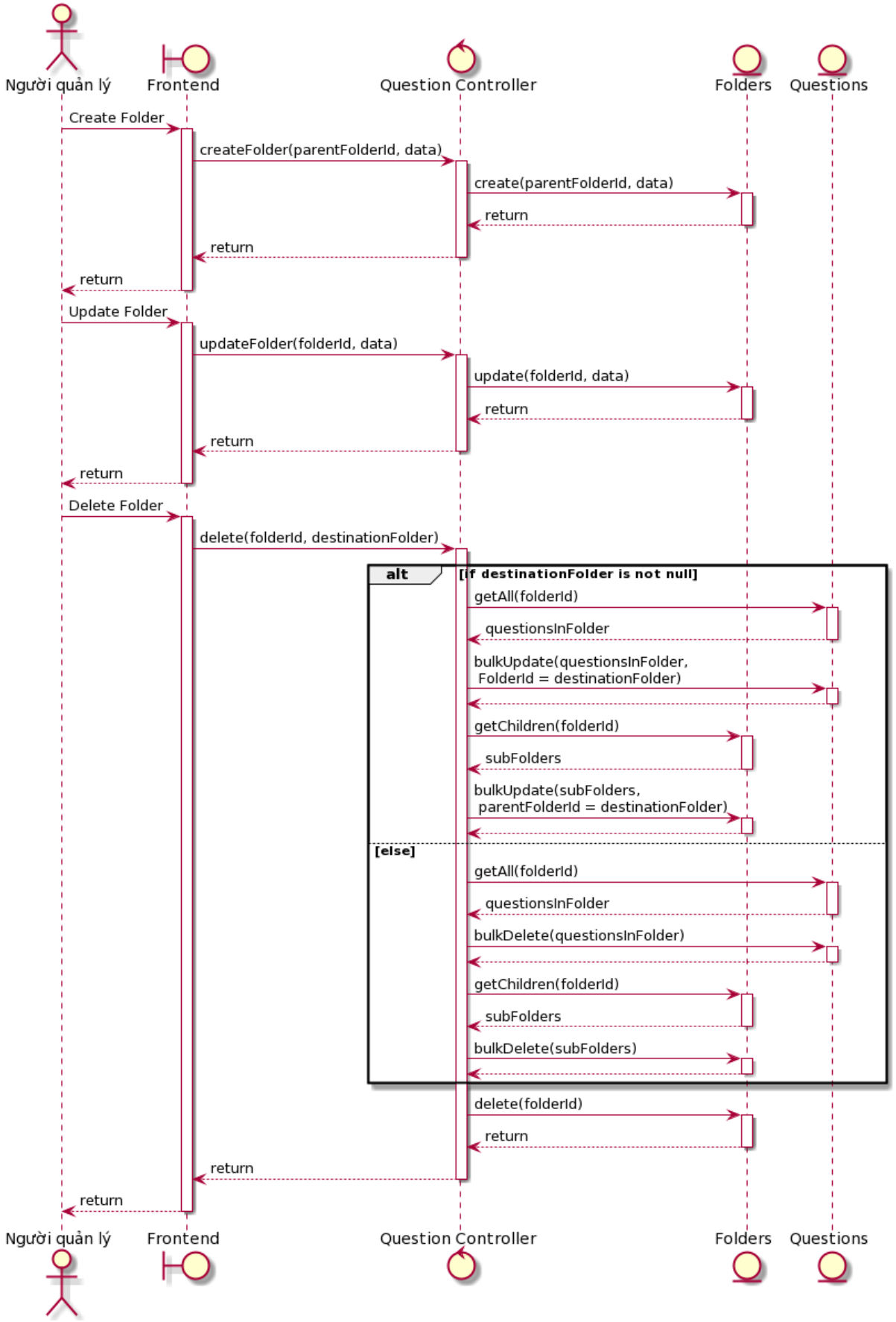
Bảng 4.6: *Bảng mô tả use-case cho chức năng Thêm/Xóa một câu hỏi trong bài thực hành*

Tên use-case	Thêm/Xóa một câu hỏi trong bài thực hành
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng thêm/xóa một câu hỏi trong ngân hàng câu hỏi vào trong/ra khỏi bài thực hành.
Tiền điều kiện	Người dùng đang ở màn hình của một bài thực hành.
Luồng đi thông thường	1. Người dùng nhấn vào nút *Add Question*/ icon Delete Question. 2. Nếu người dùng nhấn vào nút *Add Question* trong bài thực hành. 2.1. Hộp thoại hiện ra là một phiên bản thu gọn của ngân hàng câu hỏi, sử dụng chức năng truy cập của ngân hàng câu hỏi để tìm câu hỏi mong muốn. 2.2. Người dùng nhấn icon *Add*. 2.3. Một câu hỏi được thêm vào bài thực hành. 3. Nếu người dùng nhấn vào nút icon Delete Question của một câu hỏi. 3.1. Câu hỏi được xóa khỏi bài thực hành.
Ngoại lệ	*Ngoại lệ 1: bước 2* 2a. Nếu câu hỏi đã tồn tại trong bài thực hành, hệ thống hiện thông báo đến người dùng.
Luồng đi thay thế	Không

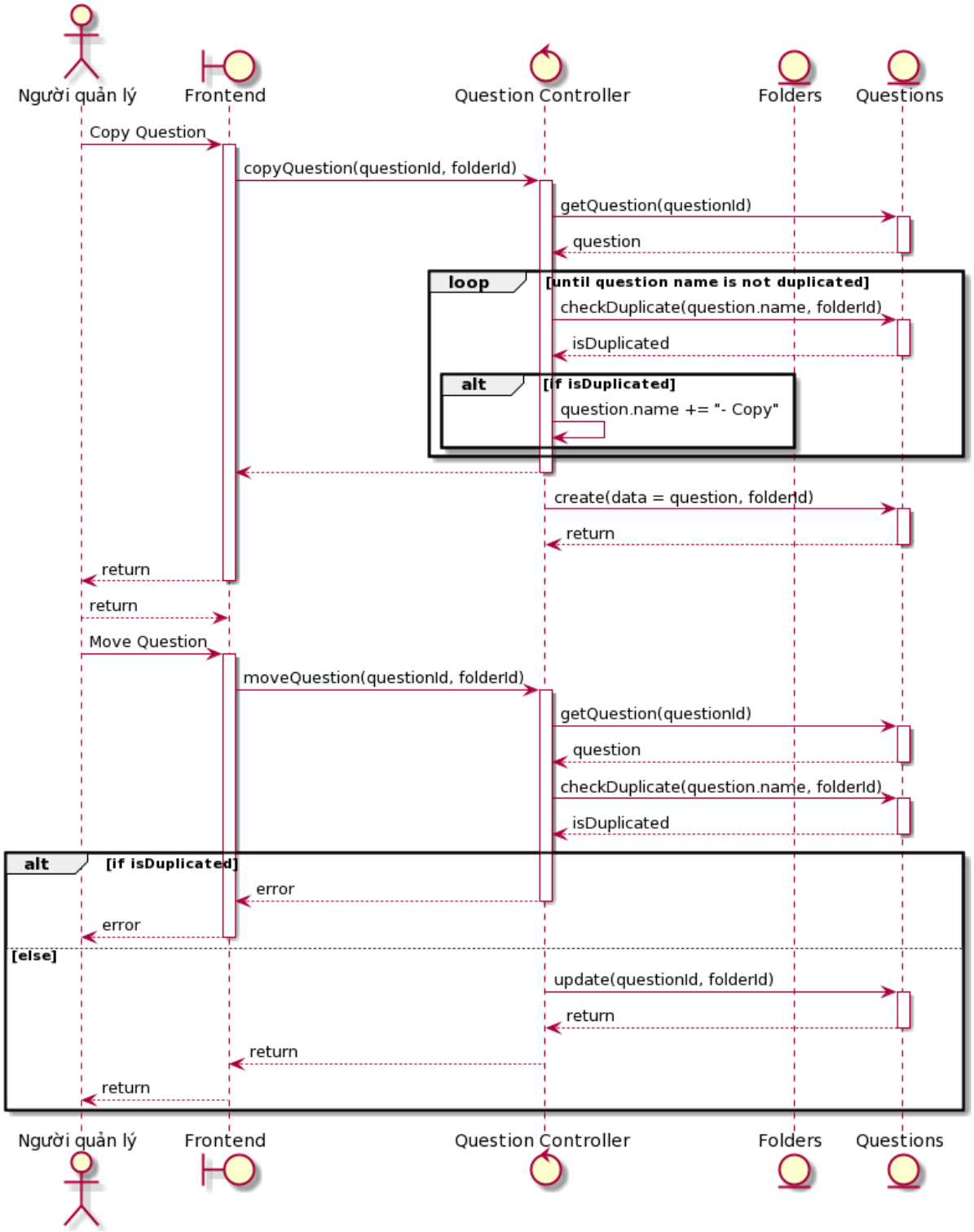
4.2.7 Hiện thực các chức năng quản lý ngân hàng câu hỏi



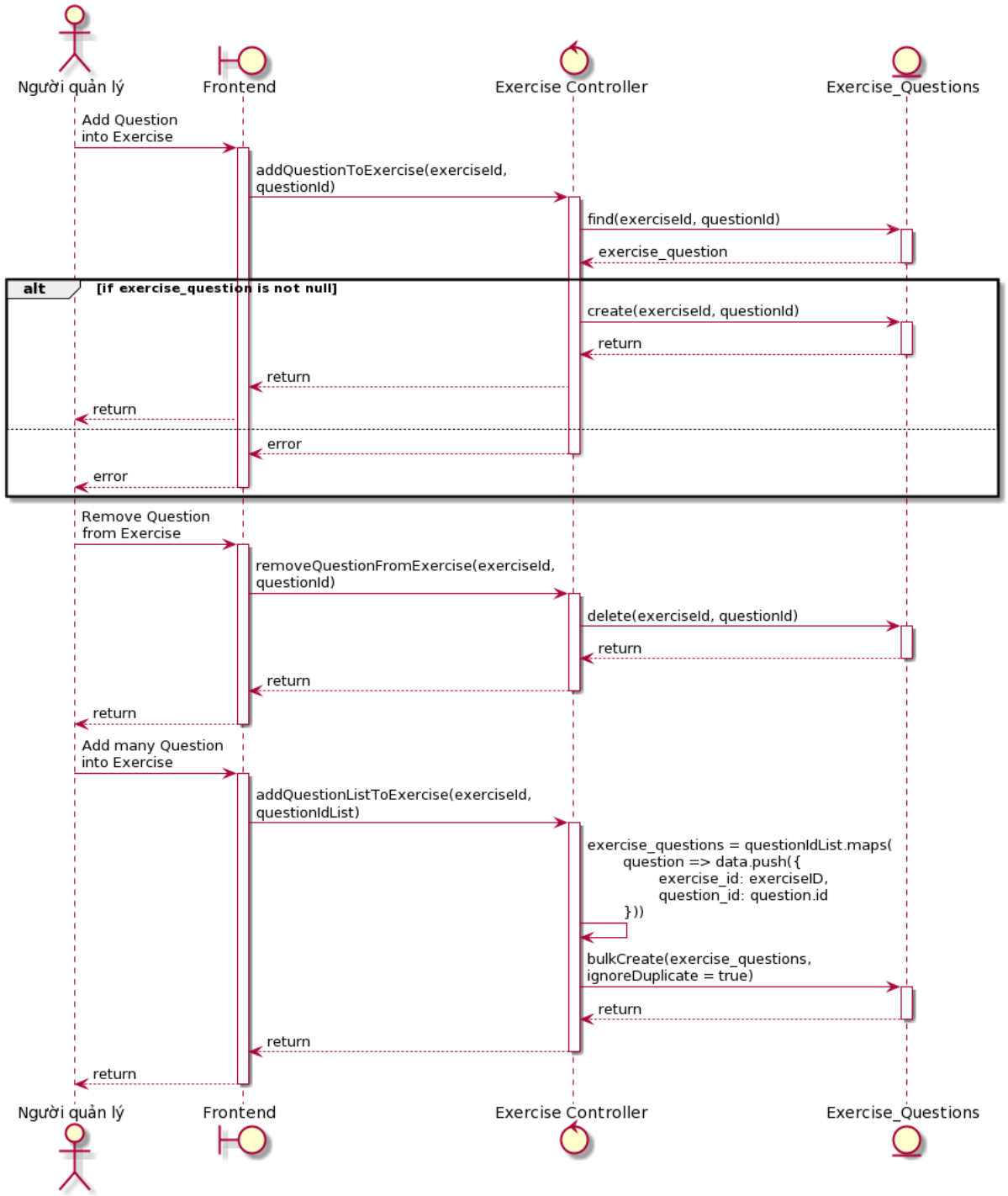
Hình 4.14: *Sequence Diagram cho chức năng xem ngân hàng câu hỏi và truy cập thư mục*



Hình 4.15: *Sequence Diagram cho các chức năng quản lý thư mục*



Hình 4.16: *Sequence Diagram cho chức năng sao chép hoặc di chuyển câu hỏi*



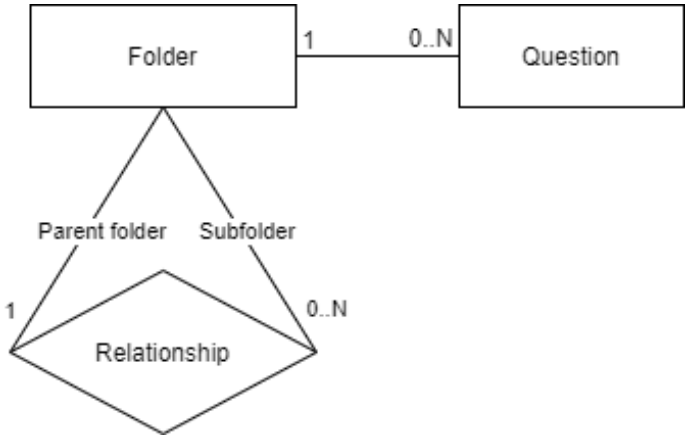
Hình 4.17: *Sequence Diagram cho chức năng Thêm/Xóa câu hỏi trong bài thực hành*

4.2.8 Thay đổi cơ sở dữ liệu cho tính năng quản lý ngân hàng câu hỏi

Để có thể hiện thực được các giải pháp đề xuất cho ngân hàng câu hỏi, nhóm cần phải có một số điều chỉnh cho việc lưu trữ dưới CSDL. Ban đầu, mối quan hệ-thực thể dưới cơ sở dữ liệu của AGS được mô tả như hình 4.18. Việc đầu tiên chúng ta cần làm là tách sự phụ thuộc của của Exercise và Course đối với Question.

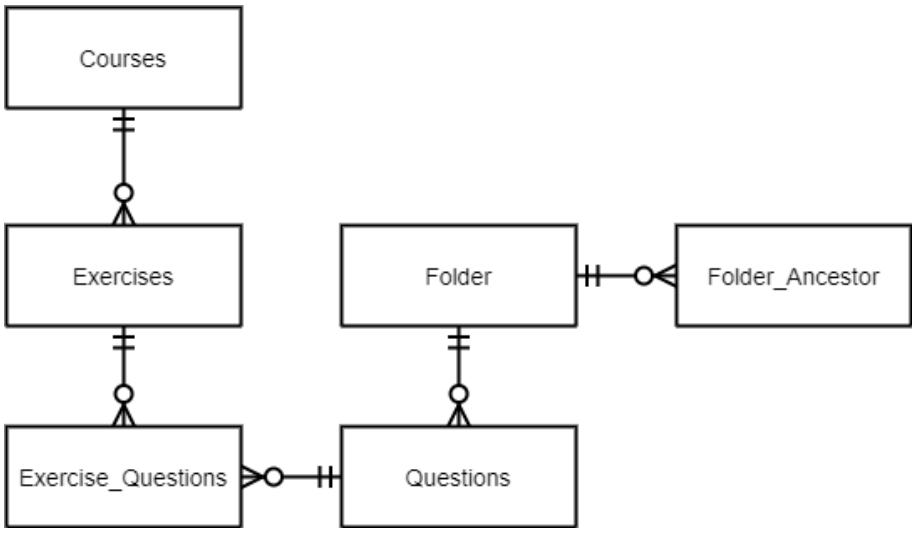
Hình 4.18: *Mô hình cơ sở dữ liệu lưu trữ câu hỏi của hệ thống AGS*

Thay vào đó, các câu hỏi giờ đây sẽ thuộc về một ngân hàng câu hỏi và cụ thể chính là các thư mục. Trong đó, thư mục gốc sẽ là thư mục của một tổ chức (ngân hàng chung) hoặc thư mục gốc của một cá nhân (ngân hàng riêng). Sơ đồ ý niệm của cách lưu trữ cây phân cấp thư mục được trình bày như hình 4.19.



Hình 4.19: *Sơ đồ ý niệm của cách lưu trữ câu hỏi trong ngân hàng câu hỏi*

Bên cạnh đó, nhóm cũng có một số điều chỉnh, ví dụ như quan hệ giữa câu hỏi với bài thực hành giờ đây sẽ là quan hệ many-to-many. Tổng quan lại, sơ đồ quan hệ-thực thể dùng để lưu trữ câu hỏi được nhóm thay đổi thành như hình 4.20.



Hình 4.20: *Mô hình cơ sở dữ liệu của tính năng quản lý ngân hàng câu hỏi*

Chi tiết cụ thể các thay đổi được nhóm hiện thực như sau:

- Thêm bảng **Folder**: Đây là bảng lưu trữ các thư mục chứa các câu hỏi.

Bảng 4.7: *Bảng Folders*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID, PK	
name	TEXT	Tên thư mục
organization_id	UUID	Khóa ngoại trỏ đến tổ chức (organization) của thư mục (bị null nếu là thư mục của ngân hàng riêng)
user_id	UUID	Khóa ngoại trỏ đến người dùng của thư mục
parent_id	UUID, FK	Khóa ngoại trỏ đến thư mục cha
hierarchy_level	INTEGER	Chiều cao của nút so với nút gốc
created_by	UUID	Người tạo thư mục
created_at	TIMESTAMP	Được tạo ra vào lúc
updated_at	TIMESTAMP	Được cập nhật vào lúc
deleted_at	TIMESTAMP	Được xóa bỏ vào lúc

- Thêm bảng **Folder Ancestors**: Đây là bảng lưu trữ mối quan hệ giữa các thư mục, được tạo với mục đích giúp việc truy xuất dữ liệu được thuận tiện hơn.

Bảng 4.8: *Bảng Folder Ancestors*

Tên trường	Kiểu dữ liệu	Mô tả trường
folder_id	UUID, FK	Khóa ngoại trỏ đến một thư mục
ancestor_id	UUID, FK	Khóa ngoại trỏ đến một thư mục là đời trước của thư mục trên(là nút cha, nút ông, ...)

- Cập nhật bảng **Question**: Thêm cột folder_id, một câu hỏi giờ phải thuộc về một thư mục.

Bảng 4.9: *Cập nhật bảng Questions*

Tên trường	Kiểu dữ liệu	Mô tả trường
folder_id	UUID, FK	Khóa ngoại trỏ đến thư mục chứa câu hỏi.

- Thêm bảng **Exercise_Questions**: Đây là bảng lưu trữ mối quan hệ giữa một bài thực hành với một câu hỏi (Vì một câu hỏi có thể nằm trong nhiều bài thực hành, và ngược lại).

Bảng 4.10: *Bảng Exercise_Questions*

Tên trường	Kiểu dữ liệu	Mô tả trường
exercise_id	UUID, FK	Khóa ngoại trỏ đến bài thực hành
question_id	UUID, FK	Khóa ngoại trỏ đến câu hỏi

4.3 Xây dựng lộ trình thực hành cho người học

4.3.1 Giới thiệu động cơ của tính năng

Trong các phần trước, nhóm tác giả đã giới thiệu và hoàn thiện một số tính năng làm bài lập trình và chấm bài tự động cho người học. Tính năng làm bài cung cấp một môi trường trực quan giúp người học dễ dàng điền các đoạn mã nguồn yêu cầu. Tính năng chấm bài tự động giúp người học nhận được phản hồi cho bài nộp một cách nhanh chóng, từ đó tiếp tục chỉnh sửa bài nộp và tăng tốc độ luyện tập lập trình. Nhìn chung, các tính năng này mang lại động lực luyện tập lập trình lớn cho sinh viên.

Mặt khác, nhóm xem xét sâu hơn về vấn đề động lực học tập của người học trên các hệ thống hỗ trợ lập trình. Nhóm nhận thấy, các hệ thống hỗ trợ lập trình này có điểm hạn chế chung là thiếu cơ chế khuyến khích việc học tập phù hợp đến từng người học. Việc khuyến khích học tập ở đây được định nghĩa là người học được đưa ra một câu hỏi vừa sức, người học sẽ được hướng dẫn làm các bài tập từ dễ đến khó. Từ đó, người học có thể cảm thấy hứng thú, và mong muốn tiếp tục luyện tập. Nhóm xin trình bày một số nguyên nhân dẫn đến việc thiếu động lực học tập như sau:

1. **Giảng viên chưa ước lượng độ khó phù hợp với sinh viên.** Thông thường, giảng viên phụ trách một lớp gồm nhiều sinh viên. Giảng viên sẽ thực hiện công việc ra đề, ước lượng độ khó cùng số lượng câu hỏi để gán sao cho phù hợp với sinh viên. Tuy nhiên, khảo sát trong nghiên cứu của van de Watering and van der Rijt (2006) chỉ ra rằng, người dạy chỉ ước lượng chính xác một phần nhỏ độ khó của các câu hỏi đối với góc nhìn của người học.

2. **Tri thức từ trước (prior knowledge) và khả năng của mỗi người học là khác nhau.** Trong thời đại Công nghiệp 4.0, mọi người đều có thể tiếp cận thông tin một cách dễ dàng. Do vậy, mỗi người đều có thể học hỏi rất nhiều từ Internet. Dẫn đến tri thức từ trước của mỗi người có thể có sự khác nhau lớn. Khả năng học hỏi của mỗi người cũng khác nhau, có người có thể tiếp thu nhanh một số nội dung và với người khác thì điều đó khó khăn hơn. Do đó, một số câu hỏi được gán chung cho tập thể người học sẽ là quá dễ hoặc quá khó đối với mỗi người. Việc làm nhiều các câu hỏi quá dễ hoặc quá khó đều dẫn đến chán nản cho người học, hơn nữa với các câu hỏi quá khó, có thể làm người học từ bỏ học tập.

3. **Người học gặp áp lực vì điểm số.** Với các câu hỏi quá khó, người học không thể làm bài được và tìm các nguồn để tham khảo. Thông thường, người học tìm kiếm thông tin về câu hỏi trên Internet. Tuy nhiên, lượng thông tin trên Internet là quá lớn. Người học gặp vấn đề này thường là những người mới bắt đầu học lập trình. Do vậy, họ không thể lựa chọn, hoặc mất nhiều thời gian lựa chọn các nguồn tham khảo cần thiết để trả lời câu hỏi. Một vấn đề khác gặp phải là đã tìm được một mã nguồn đáp án cho câu hỏi, nhưng người học không thể hiểu được mã nguồn này.

Do đó, nhóm hướng đến hiện thực tính năng *Xây dựng lộ trình thực hành cho người học* trong hệ thống hiện tại với các mục tiêu sau:

1. **Tạo một môi trường cho người học luyện tập.** Đây là một môi trường làm bài riêng bên cạnh môi trường làm các bài tập do giảng viên gán. Môi trường này cho phép người học được vào luyện tập tự do với nhiều câu hỏi và không bị tính điểm.

2. **Tự động hướng dẫn người học làm các câu hỏi từ dễ đến khó.** Môi trường này sẽ gồm nhiều câu hỏi ở cả 3 mức độ dễ, trung bình, khó. Người học sẽ được hướng dẫn làm các câu hỏi với độ khó là dễ trước. Sau khi luyện tập đủ nhiều và quen với độ khó hiện tại, người học sẽ được hướng dẫn làm các câu hỏi ở độ khó cao hơn.

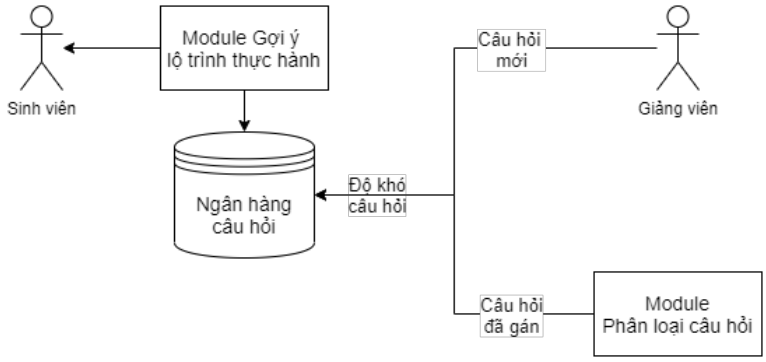
3. **Độ khó được xác định trên kết quả làm bài của người học.** Độ khó của các câu hỏi không nên bị đánh giá quá cao hoặc quá thấp bởi người dạy, dẫn đến không phù hợp với người học. Môi trường này sẽ sử dụng kết quả làm bài của người học để ước lượng độ khó cho câu hỏi.

4.3.2 Tổng quan tính năng

Tính năng *Xây dựng lộ trình thực hành cho người học* được nhóm xác định gồm 2 tính năng chính. Tính năng đầu tiên là phân loại độ khó dưới góc nhìn của người học. Sau khi đã có độ khó của các câu hỏi, tính năng thứ hai thực hiện gợi ý câu hỏi đến người học sao cho độ khó là phù hợp.

Tính năng phân loại độ khó là nền tảng để có được độ khó của một câu hỏi dựa trên kết quả làm bài của sinh viên. Nhóm tác giả tìm hiểu các công thức liên quan đến độ khó của một câu hỏi ở mục 4.3.3. Sau đó, mục 4.3.4 giới thiệu giải thuật k-means để gom cụm các câu hỏi theo độ khó. Mục 4.3.5 sẽ trình bày việc thực nghiệm trên dữ liệu làm bài của AGS, trích xuất các giá trị tương ứng với các công thức liên quan ở trên và tiến hành phân cụm dữ liệu. Để hoàn thành việc phân loại, mục này trình bày một phương pháp gán độ khó cho cụm được nhóm để xuất. Sau đó đánh giá kết quả phân loại độ khó đạt được. Sau khi đã trình bày các cơ sở và đánh giá phương pháp, nhóm xây dựng một tính năng mới trên hệ thống dựa vào phương pháp phân loại này ở mục 4.3.6 và thực hiện các thay đổi cần thiết trên CSDL ở mục 4.3.8.

Tính năng gợi ý câu hỏi cần biết độ khó của một câu hỏi để thực hiện gợi ý. Ban đầu, độ khó này đến từ người dạy. Lúc tạo một câu hỏi trong hệ thống, người dạy sẽ cài đặt độ khó mà bản thân ước lượng. Sau đó, độ khó này có thể được cải thiện. Sau khi câu hỏi được người học làm, người dạy có thể sử dụng tính năng phân loại câu hỏi ở trên để xem xét độ khó theo kết quả làm bài. Từ đó, người dạy có thể lựa chọn để thay đổi độ khó của câu hỏi theo độ khó được phân loại.



Hình 4.21: *Tổng quan tính năng Xây dựng lộ trình thực hành*

Hình 4.21 mô tả tổng quan về tính năng Xây dựng lộ trình thực hành. Tính năng gợi ý lộ trình thực hành sẽ thực hiện 2 công việc: xác định độ khó phù hợp với khả năng của người học (mục 4.3.9) và gợi ý câu hỏi tiếp theo cho người học làm (mục 4.3.10. Sau khi trình bày cơ sở cho 2 công việc trên, nhóm trình bày phần hiện thực tính năng ở mục 4.3.11 và các thay đổi cần thiết trên CSDL ở mục 4.3.13.

4.3.3 Các công thức mô tả độ khó

Để xác định độ khó của câu hỏi, nhóm đã đi tìm hiểu các công thức liên quan đến độ khó của câu hỏi. Với việc tìm hiểu các nghiên cứu liên quan (ở Mục 2.2.1), nhóm tác giả quyết định sử dụng 4 công thức trong việc mô tả độ khó là: số người làm bài đạt, số bài nộp là đạt, điểm trung bình cao nhất theo người làm bài và số lần nộp bài trung bình theo người làm bài. Bên cạnh đó, để có thể đánh giá kết quả phân loại, nhóm cũng xem xét công thức về điểm trung bình, đây là công thức thường được sử dụng để đánh giá kết quả học tập.

Các công thức này được xây dựng dựa trên kết quả của người học, từ đó hướng đến mục tiêu là mô tả độ khó của câu hỏi dựa theo góc nhìn của người học. Trong các phần sau, nhóm xin trình bày các công thức một cách chi tiết hơn để chỉ ra sự liên hệ giữa công thức và độ khó của câu hỏi.

a) Điều kiện đạt

Trước khi đi vào các công thức, nhóm xin trình bày rõ hơn ý định nghĩa của từ *ngưỡng đạt* và *đạt* được sử dụng trong các phần sau.

Trước hết, nhóm xin định nghĩa về *ngưỡng đạt*. *Ngưỡng đạt* của một câu hỏi là tỉ số thấp nhất giữa điểm đạt được và điểm tối đa của câu hỏi. *Ngưỡng đạt* là một số thực nằm trong đoạn từ [0 - 1].

Một bài nộp được gọi là *đạt* nếu tỉ lệ giữa điểm của bài nộp đó trên điểm tối đa của câu hỏi không nhỏ hơn *ngưỡng đạt*.

Một người làm bài được tính là *đạt* đối với một câu hỏi nếu tỉ lệ giữa điểm cao nhất mà người đó làm được trên điểm tối đa của câu hỏi không nhỏ hơn *ngưỡng đạt*.

Trong luận văn này, nhóm tác giả xác định *ngưỡng đạt* là 1.0.

b) Điểm trung bình

Điểm trung bình là một yếu tố thường được sử dụng để đánh giá kết quả học tập của một nhóm lớp. Do một câu hỏi lập trình có thể có nhiều lần nộp bài, nhóm chọn điểm của một sinh viên cho một câu hỏi bằng trung bình điểm của các lần nộp bài. Điểm trung bình của một câu hỏi là trung bình điểm của các sinh viên đạt được. Điểm này được chuẩn hóa bằng cách chia cho điểm tối đa của câu hỏi. Công thức đề xuất là:

$$F_0 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{C_i}\sum_{j=1}^{C_i}\frac{c_{ij}}{C_{max}}\right) \tag{4.1}$$

Trong đó, N là số sinh viên tham gia trả lời câu hỏi, C_i là số lần nộp bài của sinh viên thứ i, c_{ij} là điểm của lần nộp bài thứ j của sinh viên thứ i, và C_{max} là điểm cao tối đa của câu hỏi.

c) Số người làm bài đạt

Công thức này hướng đến số người làm bài đạt. Mỗi câu hỏi có một khoảng thời gian nhất định cho phép sinh viên làm bài. Trong khoảng thời gian này, một câu hỏi dễ sẽ có nhiều sinh viên làm bài đạt. Ngược lại, một câu hỏi khó sẽ có ít sinh viên làm bài đạt. Công thức đề xuất không tính những sinh viên không có bài nộp và hướng đến chuẩn hóa về khoảng [0 - 1]. Công thức đề xuất là:

$$F_1 = \frac{S}{S_{tot}} \tag{4.2}$$

Trong đó, S là số sinh viên làm bài đạt của câu hỏi, và S_{tot} là số sinh viên có ít nhất 1 bài nộp cho câu hỏi.

d) Số bài nộp đạt

Một câu hỏi khó sẽ có nhiều bài làm không đạt trước khi có một bài nộp đạt. Đối với câu hỏi lập trình, sinh viên thường không nộp thêm bài làm khi đã đạt một câu hỏi. Do đó, tỉ

lệ giữa số bài làm đạt trên tổng số bài nộp sẽ thấp. Ngược lại, với một câu hỏi dễ thì số lượng bài làm thử không đạt là thấp, tỉ lệ giữa số bài nộp đạt trên tổng số bài làm sẽ cao. Công thức đề xuất là:

$$F_2 = \frac{U}{U_{tot}} \tag{4.3}$$

Trong đó, U là số bài nộp là đạt cho câu hỏi, và U_{tot} là tổng số bài nộp của câu hỏi đó.

e) Trung bình điểm cao nhất

Thông thường, một câu hỏi sẽ được gán trong một bài thực hành, sinh viên thường có một tuần để trả lời câu hỏi và được phép nộp bài nhiều lần. Trong khoảng thời gian này, sinh viên có thể đạt điểm thấp ở các lần nộp bài đầu tiên, nhưng sau một thời gian cải thiện bài làm, điểm số đạt được sẽ thay đổi. Công thức này chỉ xem xét lần nộp bài có điểm cao nhất của sinh viên. Công thức đề xuất là:

$$F_3 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\max\{Ci*\}}{C_{max}}\right) \tag{4.4}$$

Trong đó, N là số sinh viên tham gia trả lời câu hỏi, C_{max} là điểm tối đa của câu hỏi, và C_{i*} là tập hợp điểm của các lần nộp bài của sinh viên thứ i.

f) Trung bình số lần nộp bài

Đối với một câu hỏi, sinh viên thường dừng lại không nộp bài khi đã đúng tất cả các testcases hoặc đã sử dụng hết số lần nộp cho phép của câu hỏi. Nếu sinh viên vẫn chưa đúng tất cả, sinh viên thường tiếp tục tìm lỗi, chỉnh sửa mã nguồn và tiếp tục nộp bài. Đối với một câu hỏi khó, sinh viên sẽ gặp lỗi và số lần nộp sẽ nhiều trước khi làm đúng tất cả testcases. Công thức đề xuất là:

$$F_4 = \frac{1}{N}\sum_{i=1}^{N}\frac{U_i}{U_{max}} \tag{4.5}$$

Trong đó N là số sinh viên tham gia trả lời câu hỏi, U_i là số bài nộp của sinh viên thứ i, và U_{max} là số lần nộp bài tối đa của câu hỏi.

g) Tóm tắt các công thức liên quan đến độ khó Bảng 4.11 tóm tắt các công thức mô tả độ khó ở trên cùng với khoảng giá trị và tính chất của chúng.

Bảng 4.11: *Tóm tắt 5 công thức*

Tên công thức	Công thức biểu diễn	Khoảng giá trị	Tính chất
Điểm trung bình	$F_0 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{C_i}\sum_{j=1}^{C_i}\frac{c_{ij}}{C_{max}}\right)$	[0 - 1]	Càng lớn càng dễ
Số người làm bài đạt	$F_1 = \frac{S}{S_{tot}}$	[0 - 1]	Càng lớn càng dễ
Số bài nộp đạt	$F_2 = \frac{U}{U_{tot}}$	[0 - 1]	Càng lớn càng dễ
Trung bình điểm cao nhất	$F_3 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\max\{Ci*\}}{C_{max}}\right)$	[0 - 1]	Càng lớn càng dễ
Trung bình số lần nộp bài	$F_4 = \frac{1}{N}\sum_{i=1}^{N}\frac{U_i}{U_{max}}$	[0 - 1]	Càng lớn càng khó

4.3.4 Phương pháp phân cụm *k*-means

Giải thuật phân cụm *k*-means là một giải thuật thuộc lớp Unsupervised Learning, dùng để giải các bài toàn về phân cụm. Nhóm tác giả hướng tới phân loại câu hỏi dựa trên góc nhìn của người học mà không có đánh giá (gắn nhãn độ khó) từ người dạy, do đó kỹ thuật *k*-means được lựa chọn để phân loại độ khó.

Các bước thực hiện của thuật toán *k*-means:

1. Lựa chọn ngẫu nhiên k center.
2. Với mỗi điểm dữ liêu, gán điểm dữ liệu đó bằng nhãn của center gần nó nhất.
3. Cập nhật lại các center. Center mới sẽ là giá trị trung bình của các điểm dữ liệu thuộc cùng một nhãn.

4. Quay lại bước 2 cho đến khi các center không còn thay đổi sau khi cập nhật.

Sau khi đã phân cụm, có các cách khác nhau để đánh giá chất lượng phân cụm của thuật toán. Nhóm chúng tôi chọn sử dụng điểm Silhouette (Silhouette Score) để kiểm tra lại kết quả phân cụm. Điểm Silhouette được tính như sau:

- Khoảng cách trung bình giữa điểm đang xét với tất cả các điểm trong cùng một cụm (a^i).

$$a^i = \frac{1}{|C(i)|-1} \sum_{j \in C(i), i \neq j} d(i,j) \tag{4.6}$$

- Khoảng cách trung bình nhỏ nhất giữa điểm đang xét với tất cả các điểm thuộc cụm khác (b^i).

$$b^i = \min_{k \neq i} \frac{1}{|C(k)|} \sum_{j \in C(k)} d(i,j) \tag{4.7}$$

- Giá trị Silhouette được tính là:

$$s^i = \frac{(b^i - a^i)}{max(a^i, b^i)} \tag{4.8}$$

  Nếu số điểm trong cụm là 1 thì giá trị Silhouette s^i được coi là bằng 0.

Ta thấy, điểm Silhouette có giá trị nằm trong đoạn $[-1,1]$. Nếu s^i có giá trị gần với 1 thì mẫu nằm gần nhau trong một cụm và nằm xa các cụm khác. Nếu s^i có giá trị gần với 0 thì mẫu nằm trên hoặc rất gần ranh giới quyết định giữa 2 cụm lân cận. Nếu s^i có giá trị âm thì mẫu có thể đã được gán cho cụm sai.

4.3.5 Thực nghiệm và đánh giá cho phân loại độ khó

a) Chuẩn bị thí nghiệm

Trước khi thiết kế tính năng, nhóm tác giả thực hiện phân loại câu hỏi trên tập dữ liệu đã có để đánh giá sơ bộ kết quả phân loại.

Về độ khó của câu hỏi, nhóm chỉ xem xét 3 mức độ về độ khó là: dễ, trung bình, khó.

Về tập dữ liệu thí nghiệm, nhóm sử dụng kết quả nộp bài của sinh viên trên hệ thống AGS ở 2 khóa học là KTLT và CTDL&GT. Thông tin cụ thể của tập dữ liệu được ghi trong bảng 4.12.

Bảng 4.12: *Thông tin của tập dữ liệu*

	Học kỳ	Số câu hỏi	Số sinh viên	Số bài nộp
KTLT	Học kỳ 2, năm học 2019-2020	32	673	38543
CTDL&GT	Học kỳ 2, năm học 2019-2020	32	58	2604

b) Mô hình phân cụm độ khó câu hỏi

Nhóm đề xuất 2 mô hình sau:

(a) Mô hình 1: sử dụng *k*-means với số cụm là 3, đầu vào của một điểm dữ liệu là 1 giá trị từ công thức F0.

(b) Mô hình 2: sử dụng *k*-means với số cụm là 3, đầu vào của một điểm dữ liệu là một vector gồm 4 giá trị <F1, F2, F3, F4>.

Mô hình 1 được sử dụng để xem xét và đánh giá sơ bộ mô hình 2. Cả 2 mô hình sau đó sẽ được sử dụng để hiện thực tính năng phân loại câu hỏi.

c) Xử lý dữ liệu

Các bước chuẩn bị dữ liệu để phân loại câu hỏi:

(a) Lấy tất cả các bài nộp thuộc 1 khóa học.

(b) Loại bỏ đi các bài nộp không phải do sinh viên làm.

(c) Loại bỏ đi các bài nộp cho các bài tập lớn.

(d) Loại bỏ các bài nộp bị trùng của một sinh viên cho một câu hỏi. Bài nộp trùng có thể xảy ra do khi sinh viên nhấn nộp bài nhưng giao diện chưa có phản hồi kịp, sinh viên tiếp tục nhấn nộp bài dẫn đến có bài nộp trùng, hoặc sinh viên cố tính nộp tiếp bài nộp cũ.

(e) Từ các bài nộp còn lại, lấy tất cả các câu hỏi có bài nộp.

d) Gán độ khó phù hợp cho cụm

Giải thuật *k*-means chỉ giúp thực hiện gom cụm và các điểm dữ liệu (câu hỏi) nằm trong một cụm sẽ được đánh dấu cùng một nhãn. Trong mục này, nhóm tác giả trình bày cách gán độ khó phù hợp cho từng cụm.

Đối với mô hình 1, việc phân cụm dựa trên đầu vào là 1 giá trị. Sau khi giải thuật *k*-means kết thúc, nhóm lấy 3 center của 3 cụm và sắp xếp theo giá trị tăng dần. Cụm có giá trị của

center cao nhất được gán mức độ là khó, center có giá trị thấp nhất được gán là dễ, center còn lại được gán giá trị là trung bình.

Đối với mô hình 2, việc phân cụm dựa trên vector 4 chiều. Bước đầu, nhóm cũng lấy 3 vector của 3 center sau khi phân cụm. Tuy nhiên không có cách định nghĩa mối quan hệ "lớn hơn" hoặc "nhỏ hơn" về độ khó của 2 vector nên không thể sắp xếp 3 vector. Nhóm đề xuất sử dụng thông tin trong bảng 4.11 cột *Tính chất* để xem xét một hoán vị của 3 vector vừa lấy. Nhóm tác giả định nghĩa *score* là số điểm mà một hoán vị đạt được. Việc gán độ khó phù hợp được thực hiện qua các bước:

(a) Tạo ra 6 hoán vị của 3 vector 4 chiều.

(b) Với mỗi 2 vector liền kề, xem xét tất cả các cặp giá trị của cùng một công thức, nếu 2 giá trị này thoả mãn tính chất trong bảng của công thức đó thì tăng *score* thêm 1 (giả sử đang sắp xếp các vector theo độ khó tăng dần).

(c) Chọn hoán vị vectors có *score* cao nhất, lần lượt gán tâm của hoán vị này các mức độ dễ, trung bình, khó.

Ví dụ về cách tính score của một hoán vị:

Giả sử một hoán vị 3 vector như sau:
$v1 =< 0.3, 0.4, 0.5, 0.6 >$
$v2 =< 0.2, 0.3, 0.4, 0.3 >$
$v3 =< 0.1, 0.2, 0.3, 0.4 >$

Xét các giá trị của F_1: 0.3, 0.2, 0.1, vì $0.3 > 0.2$ và $0.2 > 0.1$ nên score của F_1 là 2 (sắp xếp theo độ khó tăng dần). Tương tự, score của F_2 là 2, score của F_3 là 2. Xét F_3: có $0.4 > 0.3$ nhưng $0.3 < 0.6$ (không thoả tính chất), nên score của F_3 là 1. Do đó, score cho hoán vị này là 7.

e) Kết quả phân loại

Điểm Silhouette của 2 mô hình được ghi lại trong bảng 4.13.

Bảng 4.13: *Điểm Silhouette của 2 mô hình*

	Mô hình 1	**Mô hình 2**
KTLT	0.69	0.66
CTDL&GT	0.78	0.61

Kết quả phân loại câu hỏi của hai khóa học KTLT và CTDL&GT được trình bày lần lượt trong bảng 4.14 và bảng 4.15. Mỗi mức độ dễ, trung bình, khó được ghi tương ứng trong bảng là E, M, D.

Bảng 4.14: *Kết quả phân loại độ khó của môn KTLT*

Question	1	2	3	4	5	**6**	**7**	8	9	10	11	12	13	**14**	15	16
Model 1	D	M	M	M	E	M	D	M	M	M	E	E	M	E	M	M
Model 2	D	M	M	M	E	E	D	M	M	M	E	E	M	M	M	M

Question	**17**	18	19	20	21	22	**23**	**24**	25	**26**	27	**28**	29	30	31	32
Model 1	M	M	E	E	E	E	M	M	E	M	E	M	M	M	M	E
Model 2	E	M	E	E	E	E	E	E	E	D	E	E	M	M	M	E

Bảng 4.15: *Kết quả phân loại độ khó của môn CTDL&GT*

Question	1	2	3	4	5	**6**	**7**	**8**	9	**10**	11	12	13	14	15	16
Model 1	M	E	M	D	D	M	M	D	M	M	M	M	D	M	M	E
Model 2	M	E	M	D	D	E	E	M	M	E	M	M	D	M	M	E

Question	17	18	19	20	21	22	23	24	25	26	27	28	**29**	30	31	**32**
Model 1	M	M	M	M	E	M	E	E	M	M	M	M	M	E	E	M
Model 2	M	M	M	M	E	M	E	E	M	M	M	M	E	E	E	E

f) Đánh giá kết quả phân loại

Trong môn KTLT, ta thấy có 7 câu hỏi có kết quả phân loại khác nhau giữa 2 mô hình: 6, 14, 17, 23, 24, 26, 28. Trong đó, các câu hỏi 6, 14 được mô hình 2 phân loại khó hơn so với mô hình 1. Các câu hỏi còn lại được mô hình 2 phân loại dễ hơn mô hình 1.

Trong môn CTDL&GT, ta thấy có 6 câu hỏi có kết quả phân loại khác nhau giữa 2 mô hình: 6, 7, 8, 10, 29, 32. Trong đó, tất cả các câu hỏi đều được mô hình 2 phân loại dễ hơn so với mô hình 1.

Không có câu hỏi nào có kết quả phân loại độ khó từ 2 mô hình là trái ngược nhau. Ví dụ về một phân loại trái ngược nhau là mô hình 1 cho kết quả dễ trong khi mô hình 2 cho kết quả khó.

Nhìn chung, mô hình 2 có xu hướng phân loại độ khó dễ hơn so với kết quả của mô hình 1. Ta thấy điều này được thể hiện rõ trong môn CTDL&GT, có thể vì đây là môn học khó hơn so với môn KTLT. Do vậy ở các lần nộp đầu sinh viên có thể đạt điểm thấp, hoặc trong số các lần nộp bài, các bài nộp có điểm thấp chiếm phần nhiều. Nhưng qua quá trình suy nghĩ và cải thiện mã nguồn, đến cuối cùng sinh viên có thể đạt được điểm tối đa. Khi đó, điểm trung bình của sinh viên có thể không cao, tuy nhiên các yếu tố khác như điểm

trung bình cao nhất, số bài nộp, số sinh viên làm bài đạt sẽ mang lại nhiều thông tin để xem xét hơn cho câu hỏi.

Kết quả tính toán của các công thức trên 2 khóa học được nêu trong Phụ lục A. Nhóm tác giả xin phép lấy ra 2 câu hỏi 6 và 7 trong môn CTDL&GT để xem xét các yếu tố trên một cách cụ thể hơn. Bảng 4.16 ghi lại kết quả tính toán của các công thức cho câu hỏi 6 và 7.

Bảng 4.16: *Kết quả tính theo công thức của câu hỏi 6 và 7 môn CTDL&GT*

Câu hỏi	F0	F1	F2	F3	F4
6	0.70	0.94	0.51	0.963	0.36
7	0.71	0.96	0.50	0.996	0.35

Từ bảng, ta thấy điểm trung bình của 2 câu hỏi là khoảng 7 điểm (trên thang điểm 10) và mô hình 1 phân loại đây là 2 câu hỏi trung bình. Tuy nhiên, tỉ lệ sinh viên đạt 2 câu hỏi này là khoảng 95%. Điểm trung bình cao nhất cho câu hỏi 6 là 9.6/10, cho câu hỏi 7 là gần 10/10. Tỉ lệ về số lần nộp bài cũng khá thấp (0.35), nếu sinh viên có tối đa 5 lần nộp bài thì chỉ cần 2 lần nộp bài là có thể đúng tất cả các testcases (và dừng lại không nộp bài). Đồng thời, số bài nộp cũng có giá trị khá hợp lí, trong 2 lần nộp bài thì lần đầu tiên là chưa đạt và lần thứ hai là đạt, tỉ lệ số bài nộp đạt là 0.5. Mô hình 2 đã phân loại độ khó của 2 câu hỏi này là dễ.

Nhóm chúng tôi cũng xem xét kết quả trên 2 độ đo sau:

(a) **Độ giống nhau:** là tỉ lệ phần trăm về số câu hỏi được cả 2 mô hình phân loại giống nhau.

(b) **Độ tương tự:** là tỉ lệ phần trăm về số câu hỏi mà kết quả phân loại từ 2 mô hình là tương tự nhau. Hai kết quả được coi là tương tự nhau nếu 2 độ khó là giống nhau, hoặc khác nhau nhưng không trái ngược nhau. Ví dụ các độ khó sau là tương tự: M-M, E-M, D-M.

Kết quả 2 độ đo trên đối với 2 khóa học KTLT và CTDL&GT được mô tả trong bảng 4.17

Bảng 4.17: *Độ giống nhau và độ tương tự của kết quả phân loại*

	Độ giống nhau	Độ tương tự
KTLT	78.13%	100%
CTDL	81.25%	100%

Từ các đánh giá trên, kết quả phân loại từ mô hình 2 (sử dụng 4 công thức đề xuất) có độ trùng khớp cao so với phân loại của mô hình 1 (sử dụng công thức tính trung bình phổ

biến). Nhóm nhận thấy kết quả phân loại từ mô hình 2 là hợp lý và có thể sử dụng mô hình 2 để hiện thực tính năng phân loại câu hỏi.

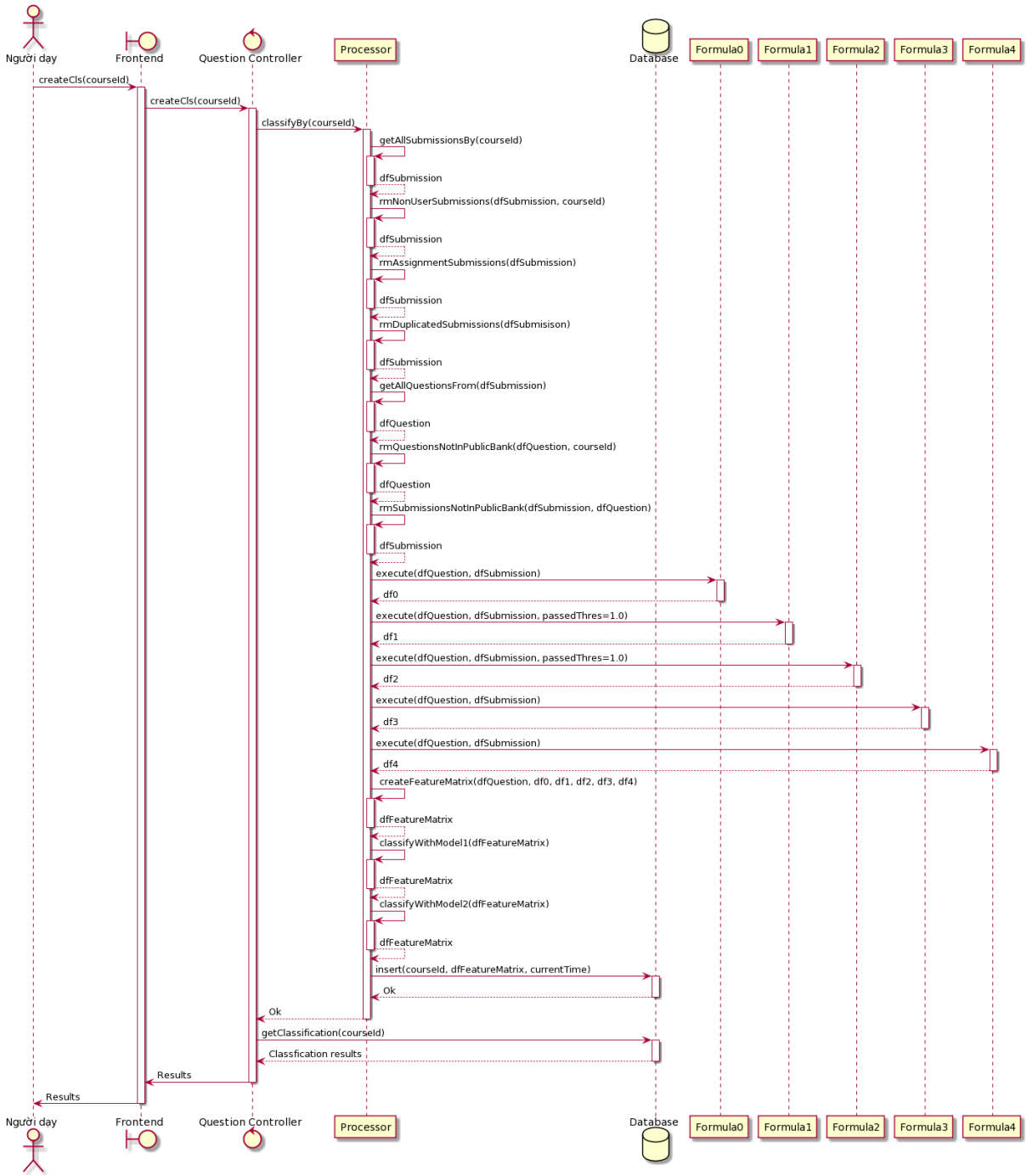
4.3.6 Thiết kế tính năng phân loại độ khó câu hỏi

Sau khi đã trình bày cơ sở khoa học cho phương pháp phân loại độ khó và đánh giá kết quả, nhóm tích hợp phương pháp này vào hệ thống thông qua tính năng phân loại độ khó câu hỏi. Chi tiết tính năng phân loại độ khó câu hỏi được trình bày trong bảng 4.18 bên dưới.

Bảng 4.18: *Bảng mô tả use-case cho tính năng phân loại câu hỏi*

Tên use-case	Phân loại độ khó câu hỏi
Người tác động	Lecturer, Admin
Mô tả	Tính năng hỗ trợ phân loại câu hỏi của một khóa học sau 1 học kỳ. Người dùng nhấn nút *Phân loại câu hỏi*, giao diện hiển thị các câu hỏi, 5 kết quả tính toán của 5 công thức F_0-F_4, độ khó của câu hỏi trước khi phân loại, 2 độ khó của câu hỏi là kết quả phân loại từ 2 mô hình.
Tiền điều kiện	Người dùng đang ở màn hình của tính năng phân loại câu hỏi.
Luồng đi thông thường	1. Người dùng nhấn nút *Phân loại câu hỏi*. 2. Hệ thống gửi yêu cầu phân loại câu hỏi của khóa học mà người dùng đang ở trong. 3. Hệ thống thực hiện xử lý dữ liệu (mục **c)**). 4. Hệ thống thực hiện phân cụm bằng *k*-means. 5. Hệ thống gán độ khó phù hợp cho cụm (mục **d)**). 6. Hệ thống hiển thị các câu hỏi cùng kết quả tính toán và kết quả phân loại cho người dùng.
Ngoại lệ	Không
Luồng đi thay thế	Không

4.3.7 Hiện thực tính năng phân loại câu hỏi



Hình 4.22: *Sequence Diagram cho chức năng Phân loại câu hỏi*

Tuy hình ảnh tổng quan nhìn khá nhỏ, tác giả nhóm đã cố gắng giảm kích thước và đảm bảo chất lượng ảnh khi phóng to. Người đọc có thể phóng to để nhìn rõ các hàm hơn.

4.3.8 Thay đổi cơ sở dữ liệu cho tính năng phân loại câu hỏi

Khi hiện thực tính năng, vì CSDL đã được thay đổi so với trong hệ thống AGS nên việc lấy dữ liệu cũng gặp khó khăn. Đôi khi, việc xử lý dữ liệu hoặc tính toán các công thức đòi hỏi phải thực hiện phép toán "JOIN" 3-4 bảng trong CSDL, dẫn đến nhiều chi phi truy vấn và thời gian tính toán. Trong mục này, nhóm thực hiện các thay đổi trong CSDL nhằm các mục đích sau:

- Tạo một bảng mới để lưu trữ kết quả phân loại.
- Thêm các cột mới vào các bảng đã có để truy vấn dữ liệu nhanh và thuận tiện.

Các điều chỉnh trên CSDL:

- Bảng **Questions**

Bảng 4.19: *Thêm mô tả độ khó vào bảng Questions*

Tên trường	Kiểu dữ liệu	Mô tả trường
level	INTEGER	Độ khó của câu hỏi với 1: dễ, 2: trung bình, 3: khó.

- Bảng **Exercise_Questions**: thêm *level* vào bảng còn nhằm mục đích cài đặt độ khó của câu hỏi trong một bài thực hành.

Bảng 4.20: *Thêm mô tả độ khó vào bảng Exercise_Questions*

Tên trường	Kiểu dữ liệu	Mô tả trường
level	INTEGER	Độ khó của câu hỏi trong bài thực hành với 1: dễ, 2: trung bình, 3: khó. Hiện tại độ khó này bằng với độ khó gốc của câu hỏi.

- Bảng **Submissions**:

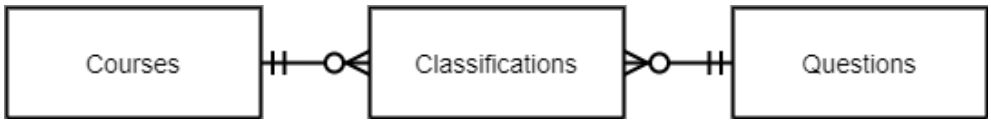
Bảng 4.21: *Cập nhật bảng Submissions cho tính năng phân loại độ khó*

Tên trường	Kiểu dữ liệu	Mô tả trường
course_id	UUID	ID của khóa học mà bài nộp này thuộc về
question_id	UUID	ID của câu hỏi mà bài nộp này thuộc về
hash_content	TEXT	Giá trị băm của nội dung bài nộp

Trong các bước xử lý dữ liệu (mục **c)**), bước 4 thực hiện loại bỏ các bài nộp bị trùng. Nhóm chỉ xem xét đơn giản các bài làm bị trùng là các bài nộp của một sinh viên có nội dung giống nhau. Hệ thống sẽ thực hiện băm nội dung của bài nộp và lưu giá trị băm vào CSDL để hỗ trợ bước xử lý này.

- Bảng **Classifications**: đây là bảng mới chứa kết quả phân loại.

  Kết nối của bảng với các bảng đã có như sau:



Hình 4.23: *ERD của Classifications và các bảng liên quan*

  Về mối quan hệ với Classification:

  – Classifications-Courses: 1 Khoá học có thể có 0 hoặc nhiều lần Phân loại, 1 Phân loại bắt buộc phải thuộc về 1 Khoá học.

  – Classifications-Questions: Một Câu hỏi có thể có 0 hoặc nhiều Phân loại, 1 Phân loại bắt buộc phải liên quan đến (là kết quả phân loại cho) 1 Câu hỏi.

  – Cụ thể, nếu một lần phân loại gồm có 15 câu hỏi cần phân loại, sẽ có 15 hàng (row) mới được thêm vào CSDL. Chúng có cùng: course_id, order, classifying_at, status; nhưng các thông tin còn lại sẽ khác nhau và tương ứng với một câu hỏi (trong 15 câu hỏi).

  Thông tin chi tiết của Bảng Classifications như sau:

Bảng 4.22: *Bảng Classifications*

Tên trường	**Kiểu dữ liệu**	**Mô tả trường**
id	UUID	ID của phân loại
course_id	UUID	ID của khóa học mà phân loại này thuộc về
question_id	UUID	ID của câu hỏi mà phân loại này thực hiện
level	INTEGER	Độ khó phân loại được của câu hỏi (1: dễ, 2: trung bình, 3: khó).
order	INTEGER	Biểu diễn lần phân loại thứ mấy của khóa học này (có course_id tương ứng), giá trị bắt đầu từ 1
classifying_at	TIMESTAMP	Thời điểm thực hiện phân loại

status	INTEGER	Trạng thái của việc phân loại: – 0: đang phân loại – 1: phân loại xong – 2: phân loại có lỗi (có >= 2 hoán vị có score cao nhất)
passed_students	DOUBLE PRECISION	Kết quả tính của F_1
passed_submissions	DOUBLE PRECISION	Kết quả tính của F_2
avg_max_grades	DOUBLE PRECISION	Kết quả tính của F_3
avg_submissions	DOUBLE PRECISION	Kết quả tính của F_4
pass_thres	DOUBLE PRECISION	Ngưỡng đạt sử dụng cho phân loại

Sau khi thực hiện các thay đổi trên, tính năng phân loại câu hỏi được xem như hoàn tất. Các mục sau sẽ trình bày về tính năng Gợi ý lộ trình thực hành.

4.3.9 Xác định độ khó phù hợp với người học

Với các công việc cần làm cho tính năng Gợi ý lộ trình thực hành, công việc đầu tiên là xác định độ khó câu hỏi phù hợp với khả năng của người học. Để làm điều đó, người học ban đầu sẽ được xác định tương ứng với mức độ là dễ. Nhóm chúng tôi đề xuất các điều kiện để kiểm tra xem khả năng của người học đã ổn định với độ khó hiện tại hay chưa. Nếu người học thỏa mãn các điều kiện này, người học sẽ được đưa lên một độ khó cao hơn.

Nhóm tác giả xin trình bày các điều kiện cần xem xét như sau:

1. **Số câu hỏi tối thiểu mà sinh viên cần làm.**
   Trước hết người học cần được tiếp xúc với một số lượng câu hỏi nhất định ở độ khó hiện tại. Khi gặp một câu hỏi, người học sẽ đọc hiểu đề, suy nghĩ cách làm. Người học có thể đưa ra được một lời giải không hoàn chỉnh hoặc hoàn chỉnh. Khi làm nhiều, người học sẽ quen với các dạng đề, các hướng giải quyết thông thường và có thể nắm được cách lập

trình để trả lời cho các câu hỏi ở độ khó hiện tại. Số câu hỏi tối thiểu là điều kiện cần cho việc kiểm tra khả năng của người học đã ổn định hay chưa.

2. **Điểm tối thiểu cần đạt được của 3 bài làm cuối (thuộc 3 câu hỏi khác nhau).**
   Khi đã ổn định với độ khó hiện tại, nhóm mong muốn rằng người học sẽ đạt được mức điểm tương đối tốt ở 3 lần làm bài liên tiếp. Điều này yêu cầu người học đã trải qua bước đọc hiểu đề, đã có hướng giải quyết cho câu hỏi và đã hiện thực một đoạn chương trình liên quan để trả lời câu hỏi. So với điều kiện trước, điều kiện này kiểm tra người học đến bước cuối cùng của việc trả lời câu hỏi. Thêm vào đó, người học cần phải làm được việc này với 3 câu hỏi liên tiếp để chứng tỏ người học không phải may mắn đạt được điểm cao ở 1 hoặc 2 câu hỏi.

3. **Số câu hỏi tối thiểu cần đạt.**
   Một câu hỏi được xem là đạt nếu thỏa mãn đồng thời 2 điều kiện sau:

   (a) **Số bài nộp tối đa**: thể hiện cận trên của số bài nộp, một bài làm càng tốt nếu có số lần nộp thử càng ít.

   (b) **Số điểm tối thiểu**: thể hiện cận dưới của điểm bài làm, một bài làm càng tốt sẽ có điểm càng cao.

   Một câu hỏi đạt trong điều kiện này là một câu hỏi với số bài nộp ít và số điểm cao. Qua đó thể hiện người học thực sự làm tốt câu hỏi này. Và số câu mà người học làm tốt cần phải đủ nhiều thì mới chứng tỏ người học đã thành thạo với độ khó này.

Ban đầu, người học sẽ được gán các câu hỏi dễ. Như vậy, sẽ có 2 bộ điều kiện cần xác định cho người học:

1. Bộ điều kiện xác định người học từ mức độ dễ lên mức độ trung bình.
2. Bộ điều kiện xác định người học từ mức độ trung bình lên mức độ khó.

Các thông tin trong điều kiện trên sẽ do người dạy nhập vào.

4.3.10 Gợi ý câu hỏi cho người học

Sau khi xác định được độ khó phù hợp với khả năng người học, hệ thống sẽ gợi ý một câu hỏi trong độ khó này đến người học. Việc lựa chọn một câu hỏi trong ngân hàng câu hỏi có thể hiện thực đơn giản bằng việc lựa chọn ngẫu nhiên. Tuy nhiên, nhóm tác giả có một số nhận xét sau:

1. Sinh viên ít cảm thấy chán nếu được làm một câu hỏi mới chưa từng gặp trước đó.
2. Nếu luôn gợi ý câu hỏi chưa làm thì một ngân hàng đề nhỏ sẽ nhanh chóng hết câu hỏi.

3. Khi ngân hàng đề hết câu hỏi mới thì cần lấy các câu hỏi sinh viên đã làm để gợi ý.
4. Sinh viên cảm thấy chán nếu được gợi ý làm lại câu hỏi vừa làm.
5. Sinh viên sẽ ít cảm thấy chán nếu được gợi ý một câu hỏi đã làm cách đây khá lâu.
6. Các câu hỏi đã làm ở thời điểm càng xa thời điểm gợi ý sẽ càng ít gây chán cho sinh viên.
7. Các câu hỏi đã làm ở thời điểm càng gần thời điểm gợi ý sẽ càng gây chán nhiều cho sinh viên.

Từ đó, nhóm chúng tôi đề xuất, bên cạnh việc lựa chọn ngẫu nhiên một câu hỏi mới để gợi ý, xác suất xuất hiện của các câu hỏi là khác nhau. Cụ thể:

1. Xác suất xuất hiện của những câu hỏi chưa làm là cao nhất và bằng nhau.
2. Những câu hỏi đã làm ở thời điểm càng xa thời điểm gợi ý thì có xác suất xuất hiện càng cao, nhưng phải nhỏ hơn xác suất của những câu hỏi chưa làm.
3. Những câu hỏi đã làm ở thời điểm càng gần thời điểm gợi ý thì có xác suất xuất hiện càng thấp.

Như vậy, xác suất xuất hiện của các câu hỏi đã làm phụ thuộc vào thời điểm làm. Nhóm tác giả xin trình bày một ví dụ để xem xét xác suất xuất hiện của các câu hỏi theo thời điểm. Giả sử hiện tại có thông tin của các câu hỏi như trong bảng 4.23. Thêm vào đó, giả sử thời điểm thực hiện gợi ý là $t = 10$.

Bảng 4.23: *Ví dụ về thông tin của câu hỏi*

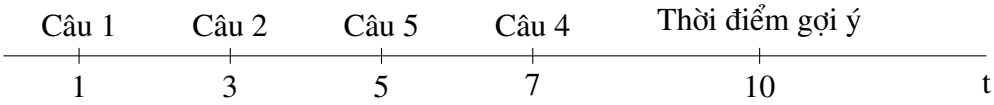
Câu hỏi	1	2	3	4	5	6
Điểm cao nhất	7.5	10.0	-1	6.0	9.0	-1
Thời điểm	1	3	-1	7	5	-1
Chênh lệch thời gian	9	7		3	5	

Trong bảng 4.23:

- Dòng **Câu hỏi** ghi lại các câu hỏi trong ngân hàng đề.
- Dòng **Điểm cao nhất** ghi lại điểm cao nhất mà sinh viên đạt được cho câu hỏi này. Điểm tối đa của các câu hỏi là 10.0. Nếu câu hỏi chưa được làm thì giá trị ghi tại cột đó tương ứng là -1.
- Dòng **Thời điểm** ghi lại thời điểm làm bài đạt điểm cao nhất ở dòng trên. Nếu có nhiều

bài nộp đạt điểm cao nhất thì lấy thời điểm làm bài cuối cùng. Nếu câu hỏi chưa được làm thì giá trị ghi tại cột câu hỏi tương ứng là -1.

- Dòng cuối ghi lại chênh lệch thời gian so với thời điểm gợi ý.



Ta thấy câu 1 cách xa thời điểm gợi ý nhất và câu 4 gần thời điểm gợi ý nhất. Khoảng thời gian chênh lệch biểu diễn mức độ đóng góp vào khả năng xuất hiện của câu hỏi. Mục tiêu của nhóm là biến đổi khoảng thời gian chênh lệch này thành xác suất xuất hiện của một câu hỏi. Tuy nhiên, nhóm cần xử lý mức độ đóng góp này cho câu hỏi 3 và 6. Theo đề xuất trên, các câu hỏi 3, 6 là các câu hỏi chưa được làm, mức độ đóng góp của 2 câu này là cao nhất. Nhóm chọn mức độ đóng góp cho 2 câu 3, 6 là 9 * 2 = 18, với 9 là chênh lệch thời gian lớn nhất.

Bảng 4.24: *Ví dụ về mức độ đóng góp vào khả năng xuất hiện của các câu hỏi*

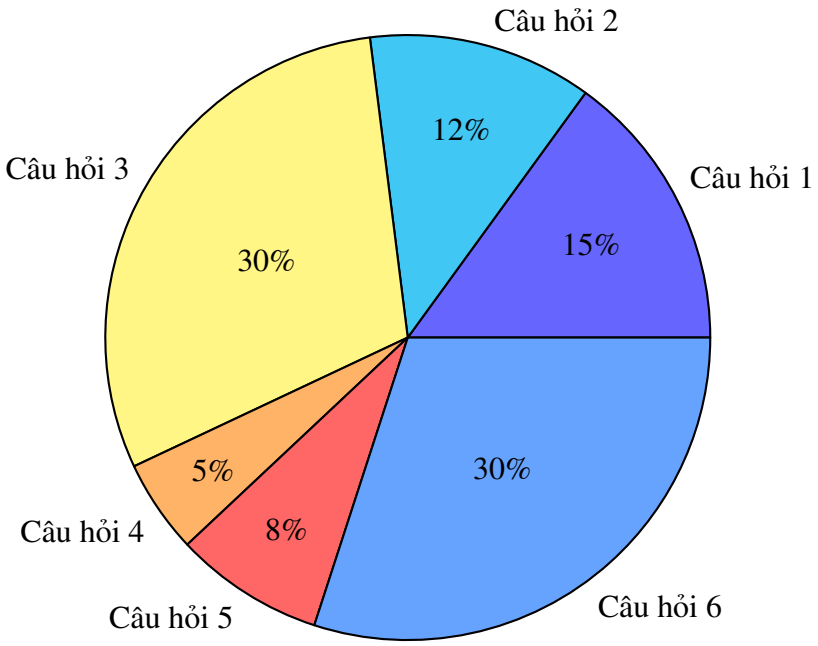
Câu hỏi	1	2	3	4	5	6
Mức độ đóng góp vào khả năng xuất hiện	9.0	7.0	18.0	3.0	5.0	18.0

Xác suất xuất hiện của mỗi câu hỏi được tính là

$$p_i = \frac{s_i}{sum\_s}$$

Trong đó, p_i là xác suất xuất hiện của câu hỏi i, s_i là mức độ đóng góp của câu hỏi i, sum_s là tổng tất cả mức độ đóng góp của các câu hỏi. Xác suất xuất hiện của các câu hỏi được biểu diễn trong hình 4.24.

Hình 4.24: *Ví dụ về xác suất xuất hiện của câu hỏi*



Sau khi tìm ra xác suất xuất hiện của từng câu hỏi, công việc tiếp theo cần làm là lựa chọn ngẫu nhiên một câu hỏi. Nhóm thực hiện biến đổi xác suất của mỗi câu thành tổng tích lũy của các xác suất như sau:

Bảng 4.25: *Ví dụ tổng xác suất tích lũy của các câu hỏi*

Câu hỏi	1	2	3	4	5	6
Xác suất xuất hiện	0.15	0.12	0.30	0.05	0.08	0.30
Tổng tích luỹ xác suất	0.15	0.27	0.57	0.62	0.70	1.00

Để lựa chọn câu hỏi, ta sẽ sinh ngẫu nhiên một số thực x trong nửa khoảng $[0-1)$, chọn câu hỏi có tổng tích lũy *lớn hơn x và gần x nhất*. Ví dụ:

1. x = 0.05 => chọn câu hỏi 1.
2. x = 0.63 => chọn câu hỏi 5.

Giải thuật gợi ý câu hỏi gồm giải thuật tính xác suất xuất hiện của mỗi câu hỏi (Thuật toán 1) và giải thuật chọn câu hỏi gợi ý (Thuật toán 2).

Algorithm 1: Tính xác suất xuất hiện của các câu hỏi.

Input:

ques_times[]: Mảng các thời điểm nộp bài của lần nộp bài cuối cùng của các câu hỏi. Nếu câu hỏi vẫn chưa được làm thì thời điểm có giá trị là -1.0.

Output:

Mảng các xác suất xuất hiện của các câu hỏi.

begin

$N \longleftarrow ques\_scores.length$;
$currentTime \longleftarrow GET\_CURRENT\_TIME()$;
$s[N]$; `/* create new array s of size N */`
for $i = 0$ **to** $N-1$ **do**
  if $ques\_times[i] == -1$ **then** $s[i] \longleftarrow -1$;
  else $s[i] \longleftarrow currentTime - ques\_times[i]$;
end
$maxS \longleftarrow MAX(s)$;
for $i = 0$ **to** $N-1$ **do**
  if $s[i] == -1$ **then** $s[i] = maxS * 2$; `/* Unassigned question */`
end
`/* Calculate the probabilities */`;
$sumS \longleftarrow SUM(s)$;
$p[N]$;
for $i = 0$ **to** $N-1$ **do** $p[i] \longleftarrow s[i]/sumS$;
return p;

end

Algorithm 2: Chọn câu hỏi gợi ý.

Input:

p[]: Mảng các xác suất xuất hiện của các câu hỏi.

Output:

Chỉ số của câu hỏi được đề xuất.

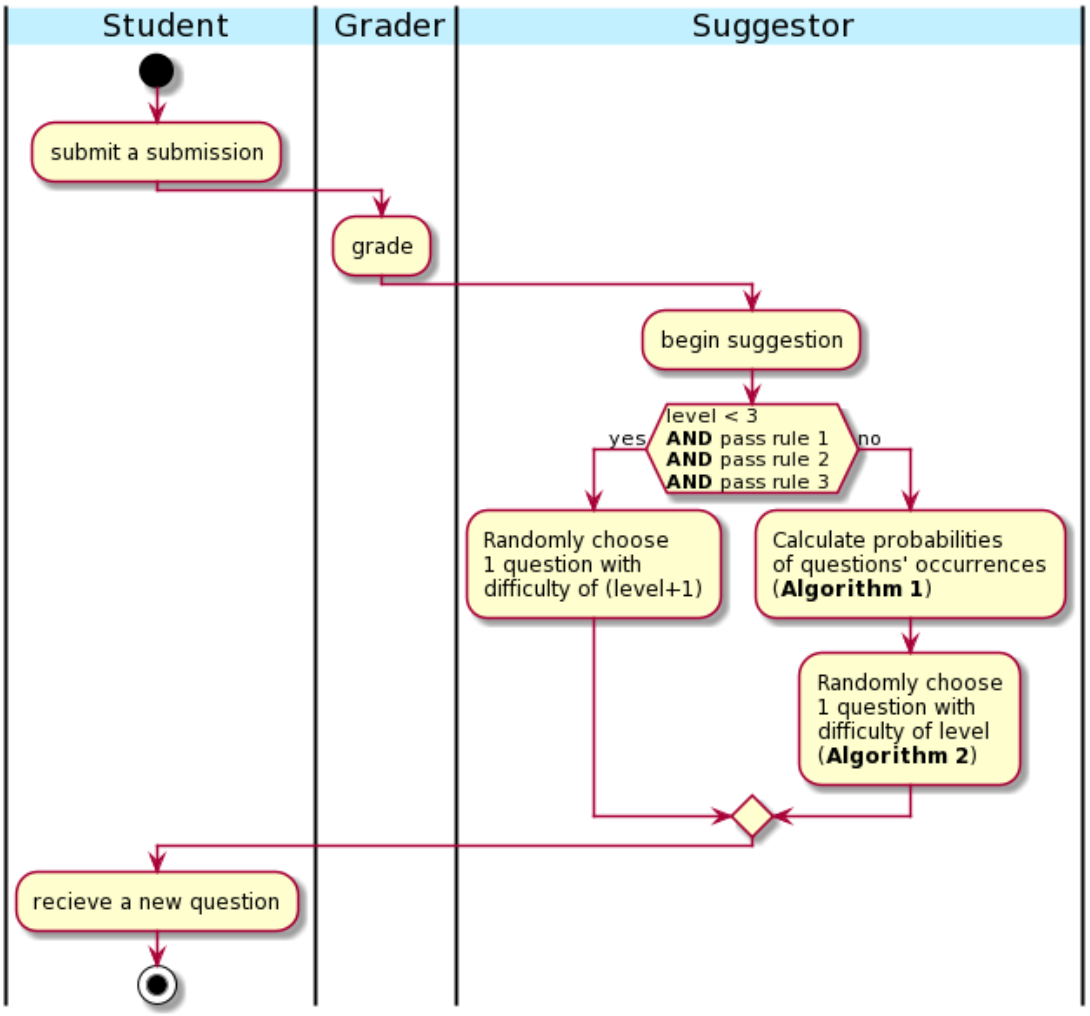
begin

$cumSumP[N]$;
$cumSumP[0] \longleftarrow p[i]$;
for $i = 1$ **to** $N-1$ **do** $cumSumP[i] \longleftarrow cumSumP[i-1] + p[i]$;
$idx \longleftarrow 0$;
`/* Randomly generates a floating-point in` $[0-1)$ `*/`
$x = RANDOM\_FLOAT(0,1)$;
while $x >= cumSumP[idx]$ **do** $idx++$;
return idx

end

4.3.11 Thiết kế tính năng gợi ý câu hỏi

Trong mục này, nhóm tác giả sẽ trình bày tính năng gợi ý câu hỏi dựa trên những mô tả trong 2 mục trước. Sơ đồ hoạt động (activity diagram) của tính năng được mô tả trong hình 4.25.



Hình 4.25: *Sơ đồ hoạt động của tính năng gợi ý câu hỏi*

Trong sơ đồ:

- *level* là độ khó tương ứng với khả năng hiện tại của sinh viên.
- *rule 1*, *rule 2*, *rule 3* lần lượt tương ứng với các điều kiện 1, 2, 3 được nêu tại 4.3.9.
- **Algorithm 1** là thuật toán tìm xác suất xuất hiện của các câu hỏi được nêu tại Thuật toán 1.
- **Algorithm 2** là thuật toán chọn câu hỏi gợi ý được nêu tại Thuật toán 2.

Sinh viên sử dụng tính năng này bằng cách vào một bài Thực hành có kiểu là *Bài luyện tập*. Ở lần đầu tiên vào, mô-đun gợi ý sẽ đánh dấu độ khó tương ứng cho sinh viên là dễ và chọn ngẫu nhiên một câu hỏi dễ bất kỳ cho sinh viên. Sau khi sinh viên làm xong và nộp bài, quá trình gợi ý được mô tả trên sơ đồ, sinh viên sẽ có một câu hỏi mới để tiếp tục luyện tập.

Khi cài đặt cho *Bài luyện tập*, người dạy sẽ cài đặt các thông số cần thiết cho 2 bộ điều kiện

để kiểm tra khả năng của người học (từ dễ lên trung bình và từ trung bình lên khó). Bảng 4.26 mô tả chi tiết tính năng Cấu hình *Bài luyện tập*.

Bảng 4.26: *Bảng mô tả use-case cho tính năng Cấu hình Bài luyện tập*

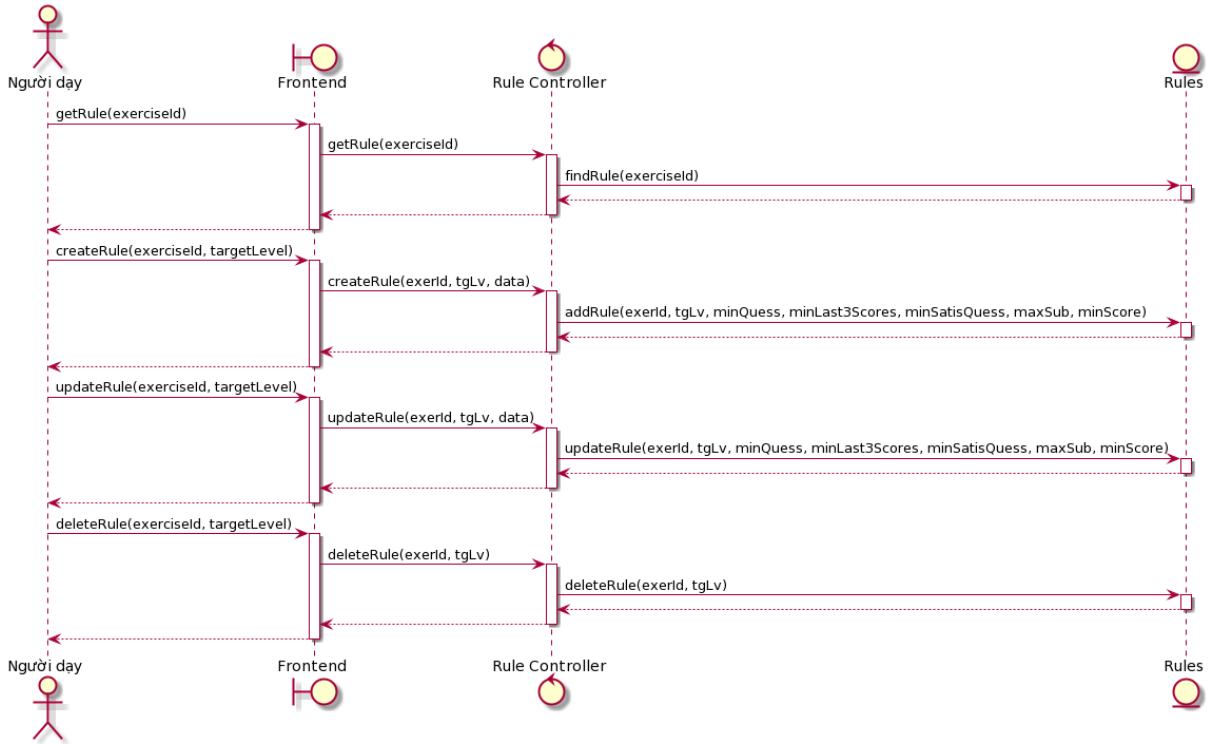
Tên use-case	Tạo bài luyện tập
Người tác động	Lecturer, Admin
Mô tả	Tính năng giúp người dùng cấu hình các thông số cần thiết cho 2 bộ điều kiện.
Tiền điều kiện	Người dùng đang ở màn hình của Bài luyện tập.
Luồng đi thông thường	1. Người dùng nhấn nút *Config rules*. 2. Hộp thoại hiện ra các thông số cần cấu hình cho điều kiện từ dễ lên trung bình. 3. Người dùng nhập các thông tin cần cấu hình. 4. Người dùng nhấn chọn tab để chuyển sang cấu hình cho điều kiện từ trung bình lên khó. 5. Người dùng nhập các thông tin cần cấu hình. 6. Người dùng nhấn nút *Confirm*. 7. Người dùng chọn câu hỏi để thêm vào.
Ngoại lệ	*Ngoại lệ 1: bước 3* 3a. Nếu người dùng không nhập một trong các thông tin cấu hình, hệ thống hiện thông báo đến người dùng. *Ngoại lệ 2: bước 4* 4a. Nếu người dùng nhấn *Confirm* mà không chọn tab để cấu hình từ trung bình lên khó, hệ thống hiện thông báo đến người dùng. *Ngoại lệ 3: bước 5* 5a. Nếu người dùng không nhập một trong các thông tin cấu hình, hệ thống hiện thông báo đến người dùng.

Luồng đi thay thế	*Thay thế 1: bước 3* 3a. Người dùng tiếp tục điền các thông tin còn thiếu. Sau đó tiếp tục bước 4 trong **Luồng đi thông thường**. *Thay thế 2: bước 4* 4a. Người dùng tiếp tục bước 4 trong **Luồng đi thông thường**. *Thay thế 3: bước 5* 5a. Người dùng tiếp tục điền các thông tin còn thiếu. Sau đó tiếp tục bước 6 trong **Luồng đi thông thường**.

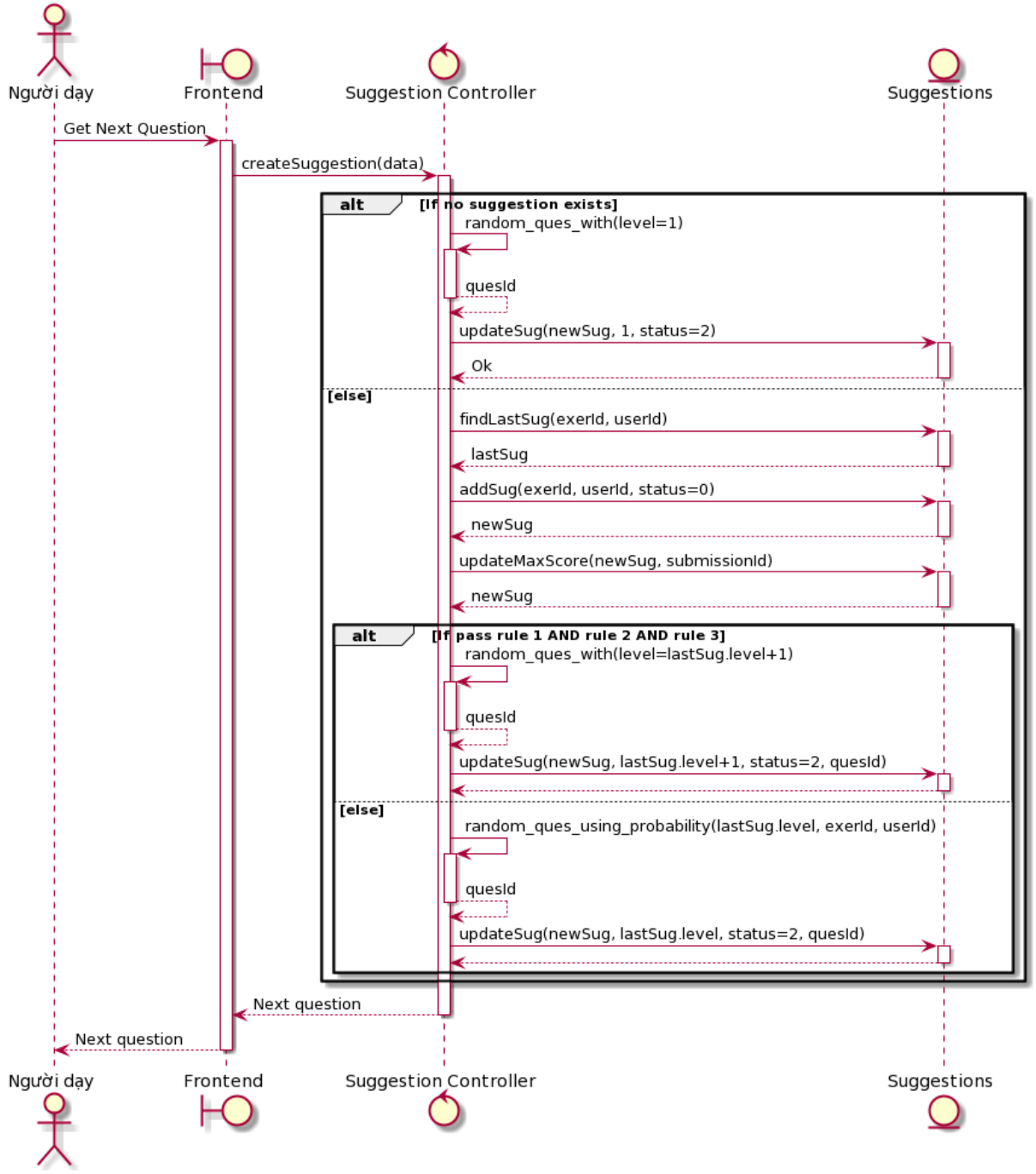
Ở **Luồng đi thông thường**, bước 3 và 4, người dùng sẽ nhập các thông tin cần cấu hình cho 2 bộ điều kiện: từ dễ lên trung bình và từ trung bình lên khó. Đối với mỗi bộ điều kiện, có 5 thông tin cần cấu hình gồm:

1. Số câu tối thiểu cần làm
2. Điểm tối thiểu cần đạt của 3 bài làm cuối cùng
3. Số câu tối thiểu cần đạt đồng thời 2 điều kiện về số lần nộp bài tối đa và số điểm tối thiểu
4. Số lần nộp bài tối đa (là một điều kiện cần thỏa của điều kiện 3)
5. Số điểm tối thiểu cần đạt (là một điều kiện cần thỏa của điều kiện 3).

4.3.12 Hiện thực tính năng gợi ý câu hỏi



Hình 4.26: *Sequence Diagram cho chức năng Cấu hình bài luyện tập*



Hình 4.27: *Sequence Diagram cho chức năng Gợi ý câu hỏi luyện tập*

4.3.13 Thay đổi cơ sở dữ liệu cho tính năng gợi ý câu hỏi

Mục này mô tả các điều chỉnh trên CSDL gồm tạo các bảng mới và thêm các cột mới vào bảng đã có để phục vụ tính năng gợi ý câu hỏi. Các điều chỉnh trên CSDL:

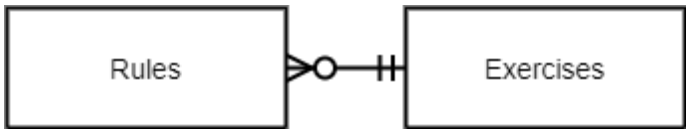
- Bảng **Exercises**: chỉnh sửa giá trị có thể có cho kiểu bài tập.

Bảng 4.27: *Thêm kiểu bài tập mới cho bảng Exercises*

Tên trường	Kiểu dữ liệu	Mô tả trường
type	INTEGER	Kiểu bài tập. Thêm kiểu bài tập 4: Bài luyện tập.

- Bảng **Rules**: bảng mới chứa các điều kiện để kiểm tra nếu khả năng của người học đã tăng lên đến độ khó cao hơn.

  Kết nối của Rules với các bảng đã có như sau:



Hình 4.28: *ERD của Rules và các bảng liên quan*

  Về mối quan hệ với Rules:

  - Một Exercises có 0 hoặc nhiều Rules: Exercise kiểu bài tập được gán sẽ không cần các bộ điều kiện nên có 0 Rules; Exercise cho người học tự liệu tập sẽ cần có 2 bộ điều kiện (dễ lên trung bình, trung bình lên khó), vì cách ký hiệu vết chân chim chỉ có 1 hoặc nhiều nên sẽ dùng ký hiệu nhiều Rules.
  - Một Rule nếu có phải bắt buộc thuộc về 1 Exercise.

  Thông tin chi tiết của Bảng Rules như sau:

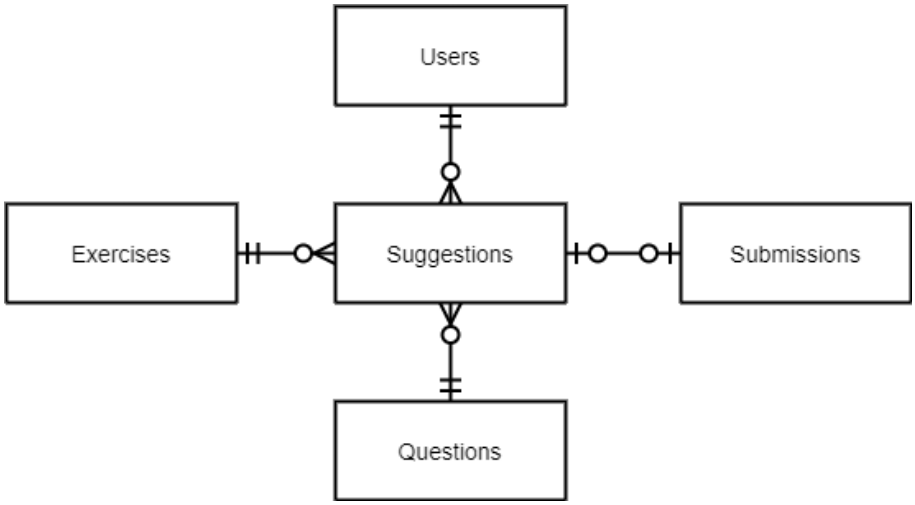
Bảng 4.28: *Bảng Rules*

Tên trường	Kiểu dữ liệu	Mô tả trường
id	UUID	ID của bộ điều kiện
exercise_id	UUID	ID Bài luyện tập mà bộ điều kiện này cấu hình
target_level	INTEGER	Số nguyên biểu diễn độ khó mà nếu sinh viên thoả mãn bộ điều kiện này thì sinh viên sẽ được tăng lên độ khó đó. Các giá trị có thể có: 2-trung bình, 3-khó.
min_questions	INTEGER	Số câu tối thiểu mà sinh viên cần làm (thông tin cấu hình 1).
min_last_3_scores	DOUBLE PRECISION	Điểm tối thiểu của 3 bài làm cuối cùng thuộc 3 câu hỏi khác nhau (thông tin cấu hình 2).

min_satisfied_ questions	INTEGER	Số câu tối thiểu cần đạt đồng thời 2 điều kiện về số lần nộp bài tối đa và số điểm tối thiểu cần đạt (thông tin cấu hình 3).
max_submissions_ each_question	INTEGER	Số lần nộp bài tối đa của một câu hỏi (thông tin cấu hình 4).
min_score_ each_question	DOUBLE PRE-CISION	Số điểm tối thiểu cần đạt (thông tin cấu hình 5).

- Bảng **Suggestions**: bảng mới lưu trữ thông tin của một lần gợi ý.

  Kết nối của Suggestions với các bảng đã có như sau:



Hình 4.29: *ERD của Suggestions và các bảng liên quan*

Về mối quan hệ với Suggestions:

  - 1 User có thể có 0 Suggestion khi chưa tự luyện tập, hoặc có thể có nhiều Suggestion. 1 Suggestion bắt buộc phải liên quan đến chỉ 1 User.
  - 1 Submission có thể thuộc về 1 Suggestion hoặc 0 Suggestion (nếu Submission đó thuộc về Assignment). 1 Suggestion có thể không có Submissions khi người học chưa nộp bài trả lời cho câu hỏi trong Suggestion, hoặc có nhiều nhất 1 Submission nếu có bài nộp trả lời cho câu hỏi.
  - 1 Exercise có 0 hoặc nhiều Suggestion (chỉ có Exercise kiểu tự luyện tập mới có Suggestion), 1 Suggestion phải bắt buộc thuộc về 1 Exercise.
  - 1 Question có 0 hoặc nhiều Suggestion do có thể câu hỏi đó chưa được gợi ý, hoặc được gợi ý nhiều lần. 1 Suggestion phải bắt buộc liên quan đến 1 Question.

Thông tin chi tiết của Bảng Suggestions như sau:

Bảng 4.29: *Bảng Suggestions*

Tên trường	**Kiểu dữ liệu**	**Mô tả trường**
id	UUID	ID gợi ý.
exercise_id	UUID	ID Bài luyện tập mà gợi ý này liên quan đến.
question_id	UUID	Câu hỏi mà gợi ý này đề xuất cho sinh viên.
user_id	UUID	Sinh viên mà gợi ý này đang đề xuất.
level	INTEGER	Độ khó tương ứng với khả năng của sinh viên tại lúc đề xuất.
max_score	DOUBLE PRECISION	Điểm cao nhất đạt được cho câu hỏi được gợi ý (tương ứng với question_id).
submitting_time	TIMESTAMP	Thời điểm nộp bài của bài đạt được max_score. Nếu có nhiều lần nộp bài có cùng điểm cao nhất, lấy thời điểm của lần làm bài cuối cùng.
status	INTEGER	Trạng thái của việc gợi ý. Các giá trị có thể có: – 0: đang thực hiện tính toán, chưa tìm ra câu hỏi gợi ý – 1: đã tính toán xong, có câu hỏi mới ở độ khó hiện tại – 2: đã tính toán xong, có câu hỏi mới ở độ khó cao hơn

- Bảng **Submissions**: thêm cột suggestion_id.

Bảng 4.30: *Cập nhật bảng Submissions cho tính năng gợi ý câu hỏi*

Tên trường	**Kiểu dữ liệu**	**Mô tả trường**
suggestion_id	UUID	Id của gợi ý mà bài nộp này làm cho câu hỏi được đề xuất trong gợi ý đó.

4.4 Kết chương

Trong chương này, nhóm đã đề xuất các giải pháp để hoàn thiện các tính năng cũ cũng như bổ sung thêm các tính năng mới cho hệ thống. Song song, nhóm cũng biện luận và trình bày cách hiện thực các giải pháp này. Công việc của nhóm trong chương này được tóm tắt như sau:

- Thay đổi thành phần chấm bài của hệ thống sang Jobe:
  - Nhóm thực hiện phân tích thành phần chấm bài cũ, và so sánh nó với Jobe - một giải pháp chấm bài mới.
  - Thay đổi con chấm của hệ thống bằng Jobe.
  - Sử dụng thêm hàng đợi Kafka và bộ nhớ đệm Redis, nhóm giới thiệu luồng chấm bài mới hiệu quả hơn so với luồng chấm bài cũ.
  - Mở rộng thêm các cách thức mới cho việc đọc đầu vào testcases bao gồm: bằng tập tin hoặc bằng testcode.
- Thiết kế tính năng quản lý ngân hàng câu hỏi:
  - Nhận thấy nhu cầu của một ngân hàng câu hỏi, nhóm tiến hành khảo sát cách quản lý câu hỏi của hệ thống AGS và hệ thống Moodle.
  - Sau khi phân tích, nhóm quyết định xây dựng ngân hàng câu hỏi cho hệ thống hiện tại với 2 giải pháp:
    * Ngân hàng câu hỏi với cây phân cấp thư mục giúp lưu trữ có tổ chức.
    * Ngân hàng chung và riêng giúp chia sẻ giữa các người dạy.
  - Cuối cùng, nhóm trình bày chi tiết các hoạt động mà người dùng có thể tương tác với ngân hàng câu hỏi.
- Xây dựng lộ trình thực hành cho người học: nhóm xác định việc xây dựng lộ trình thực hành sẽ gồm hai bước: phân loại độ khó câu hỏi và gợi ý câu hỏi có độ khó thích hợp hợp cho người học.
  - Phân loại độ khó câu hỏi:
    * Nhóm đề xuất 5 công thức biểu diễn độ khó của một câu hỏi.
    * Sử dụng phương pháp phân cụm k-means cùng với một phương pháp gán độ khó cho cụm do nhóm đề xuất, nhóm xây dựng được hai mô hình giúp phân loại độ khó câu hỏi.
    * Sau khi tiến hành thực nghiệm trên dữ liệu làm bài trước đây của AGS và thực hiện đánh giá mô hình, nhóm phát triển tính năng phân loại độ khó câu hỏi dựa trên mô hình đã xây dựng.

- Gợi ý câu hỏi cho người học:
  - Nhóm đề xuất các điều kiện để đảm bảo người học đã thuần thục một độ khó của một chủ đề luyện tập. Qua đó, hệ thống sẽ xác định độ khó câu hỏi tiếp theo để gán cho người học.
  - Sau khi đã có độ khó, nhóm cũng đề xuất 2 thuật toán áp dụng lên các câu hỏi có cùng độ khó này: thuật toán tính xác suất xuất hiện của mỗi câu hỏi trong lần gợi ý tiếp theo đến người học, và thuật toán lựa chọn ngẫu nhiên câu hỏi để đề xuất.
  - Kết hợp cả hai phần trên, nhóm phát triển tính năng gợi ý câu hỏi cho người học.

Chương 5

Triển khai và đánh giá hệ thống

Để nhìn nhận lại các giải pháp đã đề xuất, nhóm tiến hành đánh giá chúng trong chương này. Đầu tiên, nhóm trình bày phương pháp triển khai hệ thống mới để làm nền tảng cho việc đánh giá. Sau đó, thành phần chấm bài mới của hệ thống được đem ra đánh giá trên để xem xét khả năng đáp ứng và tính mở rộng. Đồng thời, các công việc triển khai thí điểm hệ thống và thu thập góp ý từ người dùng cũng được trình bày trong chương này để xem xét khả năng đáp ứng của hệ thống đối với người dùng.

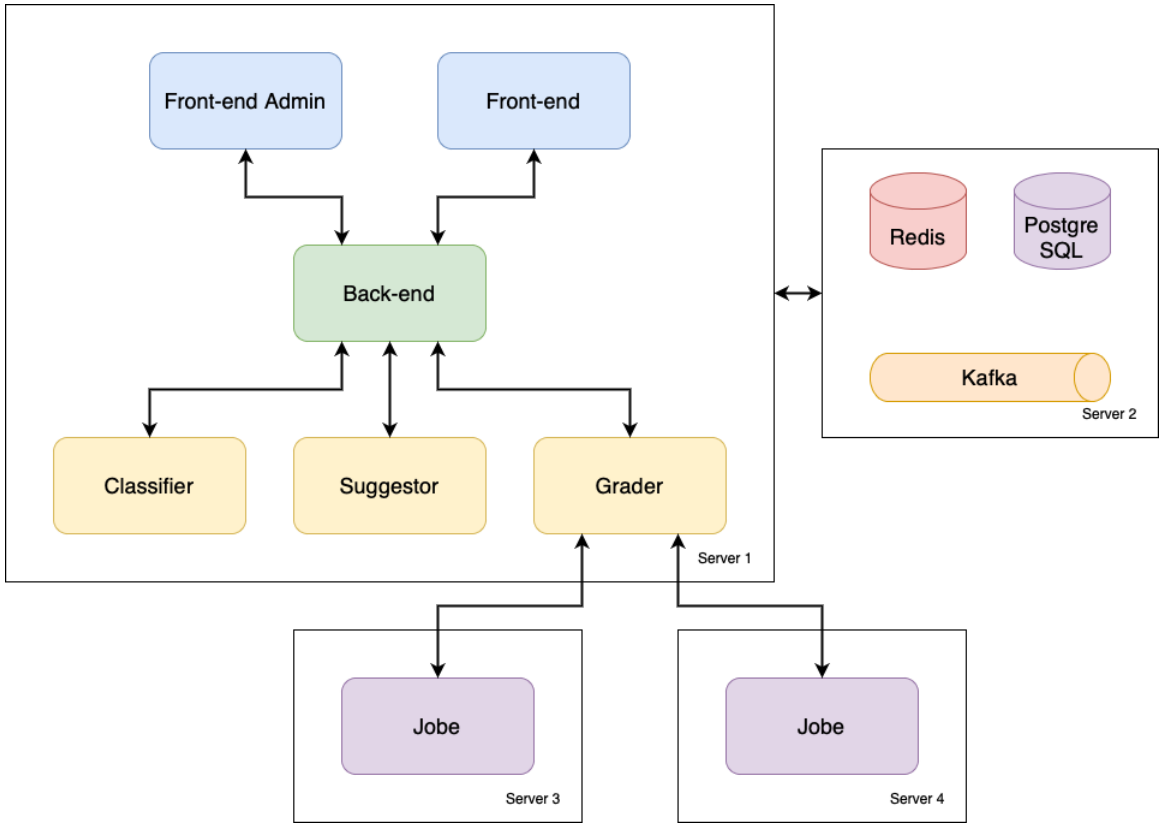
5.1 Triển khai hệ thống lên máy chủ

5.1.1 Kế hoạch triển khai

Nhóm chia các thành phần cần triển khai trong hệ thống thành 3 loại, gồm:

- Các thành phần lớp dịch vụ: Front-end Admin, Front-end, Back-end, Classifier, Suggestor và Grader. Những thành phần này sẽ được triển khai trên cùng một máy chủ.
- Các thành phần lớp hạ tầng: Redis, PostgreSQL, Kafka. Những thành phần này cũng sẽ được triển khai trên cùng một máy chủ.
- Thành phần con chấm Jobe: Nhóm sẽ triển khai hai con chấm trên hai máy chủ riêng biệt để tăng cường khả năng chấm bài.

Việc chia như trên được mô tả lại trong hình 5.1. Với cách gom nhóm như thế này, nhóm đã đưa các thành phần có chung tính chất lên cùng một các máy chủ, qua đó tạo tiền đề để dễ dàng theo dõi sau này.



Hình 5.1: *Kế hoạch triển khai cho các thành phần trong hệ thống*

5.1.2 Lựa chọn cấu hình

Như kế hoạch triển khai đã được trình bày ở phần trên, nhóm tiến hành lựa chọn cấu hình cho từng máy chủ để đảm bảo được nhu cầu mà hệ thống cần.

Với các máy chủ 1 và 2 trong hình 5.1, nhiều thành phần của hệ thống đang được triển khai chung nên nhóm quyết định chọn mức cấu hình như sau cho mỗi máy chủ để đảm bảo các thành phần ấy hoạt động ổn định nhất:

- CPU: 2 nhân
- RAM: 4 GB
- SSD: 80 GB

Với các máy chủ 3 và 4 trong hình 5.1, mỗi máy chủ chỉ cần triển khai con chấm Jobe nên mức cấu hình được lựa chọn cho mỗi máy chủ sẽ thấp hơn để tối ưu chi phí sử dụng:

- CPU: 1 nhân
- RAM: 2 GB
- SSD: 50 GB

5.2 Đánh giá tính năng chấm bài

Nhóm nhận thấy tính năng chấm bài là một tính năng quan trọng và ảnh hưởng lớn đến trải nghiệm của người học khi sử dụng hệ thống. Do vậy, sau khi triển khai thành công hệ thống lên các máy chủ, nhóm thực hiện việc đánh giá hiệu năng của tính năng chấm bài này để có được cái nhìn tổng quan về khả năng đáp ứng của hệ thống khi có nhiều người học sử dụng cùng lúc.

Việc đánh giá hiệu năng của tính năng chấm bài sẽ bao gồm 2 phần:

- Phần 1: Nhóm so sánh giữa hai giải pháp chấm bài cũ (sử dụng Docker) và mới (sử dụng Jobe) để củng cố hơn về sự lựa chọn Jobe là con chấm mới của hệ thống.
- Phần 2: Nhóm sẽ đánh giá một cách chi tiết hơn với nhiều ngữ cảnh để có được những con số cụ thể về khả năng đáp ứng của tính năng chấm bài khi được sử dụng trong thực tế.

Để phục vụ riêng cho quá trình đánh giá này, nhóm đã triển khai thêm hai máy chủ khác để tránh tình trạng chồng lấp dữ liệu lên các máy chủ hiện có, vốn được triển khai để được sử dụng ngoài thực tế.

Hai máy chủ mới được sử dụng thay thế cho hai máy chủ 1 và 2 như hình 5.1, gọi là máy chủ mới 1 và máy chủ mới 2 với cấu hình tương đương. Những máy chủ này tùy vào ngữ cảnh của phần đánh giá sẽ được triển khai các thành phần cần thiết.

5.2.1 So sánh hai giải pháp chấm bài cũ và mới

Hai giải pháp chấm bài cũ và mới khác biệt nhau chủ yếu ở con chấm được sử dụng. Trong khi giải pháp cũ sử dụng thành phần Grader với Docker làm nền tảng cho con chấm và chạy bên trong thành phần Grader, thì giải pháp mới sử dụng con chấm là Jobe đứng độc lập với thành phần Grader.

Nhóm đã thực hiện kế hoạch triển khai như sau để so sánh hai giải pháp chấm bài cũ và mới thông qua hai giai đoạn:

- Giai đoạn 1 là giai đoạn đánh giá giải pháp chấm bài cũ với kế hoạch triển khai như sau:
  - Thành phần Grader với nền tảng Docker bên trong được triển khai trên máy chủ mới 1.
  - Các thành phần phụ trợ khác như Back-end, Watcher, Redis, PostgreSQL được triển khai trên máy chủ mới 2.
- Giai đoạn 2 là giai đoạn đánh giá giải pháp chấm bài mới với kế hoạch triển khai như sau:
  - Thành phần con chấm Jobe được triển khai trên máy chủ mới 1.

– Các thành phần phụ trợ khác như Back-end, Grader, Kafka, Redis, PostgreSQL được triển khai trên máy chủ mới 2.

Với từng giai đoạn, khi đánh giá đến giải pháp nào thì kế hoạch triển khai tương ứng sẽ được thực hiện.

Sau khi triển khai lên các máy chủ mới, nhóm chọn một câu hỏi đơn giản, sử dụng ngôn ngữ C++, với bộ 20 testcases để đánh giá (mỗi testcase với số lần lấy input - cin, ngẫu nhiên trong khoảng từ 20 đến 50).

Kết quả khi thực hiện đánh giá hiệu năng với 10 vòng lặp cách nhau 5 giây, mỗi vòng lặp sẽ có 5 bài nộp cùng lúc (tức tổng cộng có 50 bài nộp tỏng vòng 50 giây) như sau:

- Với giải pháp dùng Docker: Trung bình, một bài nộp tốn khoảng 1 phút để hoàn thành.
- Với giải pháp dùng Jobe: Trung bình, một bài nộp tốn khoảng 15 giây để hoàn thành.

Như vậy, với phép so sánh như trên, có thể nhận định Jobe là con chấm tiềm năng hơn và hoàn toàn đủ khả năng để đáp ứng cho hệ thống về sau. Không những mang lại hiệu năng tốt, Jobe còn cung cấp nhiều tùy chỉnh để khai thác như đã trình bày ở phần 2.3.7, phù hợp cho nhu cầu mở rộng trong tương lai.

5.2.2 Đánh giá chi tiết về tính năng chấm bài của hệ thống

Sử dụng các thành phần đã được triển khai như ở phần đánh giá trước, nhóm tiếp tục đánh giá chi tiết hơn về quá trình chấm bài sử dụng con chấm mới là Jobe để có được cái nhìn đầy đủ hơn khi được sử dụng ngoài thực tế.

Nhóm xây dựng một số kịch bản để đánh giá như sau:

Bảng 5.1: *Các kịch bản để đánh giá chi tiết hiệu năng chấm bài*

Thứ tự kịch bản	J	U	T	N
1	1	5	5	5
2	2	5	5	5
3	2	5	3	5
4	2	10	3	5

Trong đó,

- J là số lượng máy chủ chạy Jobe
- U là số lượng người dùng nộp bài cùng lúc
- T là khoảng thời gian giữa các lần nộp bài

- N là số lần nộp bài

Các kịch bản được lập như trên có thể đáp ứng được nhu cầu đánh giá khả năng mở rộng của hệ thống thông qua sự khác biệt giữa kịch bản 1 và 2 (số lượng máy chủ chạy Jobe tăng lên).

Và các kịch bản 2, 3, 4 lại được dùng để đánh giá giới hạn đáp ứng của hệ thống khi cường độ nộp bài (số lượng bài nộp cần chấm trong một khoảng thời gian) thay đổi, biểu diễn thông qua sự thay đổi về số lượng người dùng nộp bài cùng lúc hoặc sự thay đổi về khoảng thời gian giữa các lần nộp bài.

Tất cả các kịch bản đều có cùng số lần lặp lại cho việc nộp bài đồng thời là 5 để đảm bảo áp lực tải (stress) lên hệ thống.

Kết quả của các kịch bản khi được thực thi như sau:

Bảng 5.2: *Kết quả đánh giá hiệu năng chấm bài theo kịch bản*

Thứ tự kịch bản	**Độ trễ trung bình**[1] **(s)**	**Tổng thời gian hoàn thành kịch bản** (s)
1	125,7	244
2	64,3	133
3	69,8	136
4	135,9	267

Nhận xét về kết quả thực thi:

- Kịch bản 1 và 2: Với cùng một số lượng bài cần phải chấm trong cùng một khoảng thời gian, khi tăng thêm một (1) máy chủ chạy Jobe để chấm bài, tức tăng gấp đôi, thì tổng thời gian hoàn thành giảm đi gần một nửa (46%) và độ trễ trung bình cũng giảm ở mức tương tự (48%). Như vậy, có thể thấy khả năng mở rộng của thiết kế hệ thống này gần như tuyến tính, một điểm rất tốt.

- Kịch bản 2 và 3: Khi giảm thời gian giữa các lần nộp từ năm (5) giây xuống còn ba (3) giây để tăng cường độ chấm bài thì cả hai tiêu chí độ trễ trung bình lẫn tổng thời gian hoàn thành đều không chênh lệch nhau quá nhiều.

- Kịch bản 3 và 4: Hai kịch bản này chỉ thay đổi một tham số duy nhất là số lượng người dùng nộp bài đồng thời (tăng gấp đôi), tức tổng số bài nộp cần phải chấm cũng tăng lên tương ứng. Hệ thống cho thấy khả năng đáp ứng tốt cho cả hai kịch bản này, thông qua kết quả tổng thời gian hoàn thành tăng tuyến tính với số lượng bài nộp, nhưng một điểm hạn chế là độ trễ trung bình lại tăng lên đáng kể.

[1]Độ trễ trung bình là trung bình khoảng thời gian từ lúc nộp bài đến khi bài nộp có kết quả

Như vậy, về cơ bản, hiệu năng của Jobe là đủ tốt để được sử dụng làm con chấm mới cho hệ thống, đáp ứng được cường độ sử dụng cao. Và nếu nhu cầu chấm bài có gia tăng thì khả năng mở rộng cũng là rất tốt dựa theo những đánh giá đã được thực hiện.

5.3 Thí điểm hệ thống

Sau khi đã đo đạc và đánh giá về mặt kỹ thuật, nhóm hướng đến lấy các đánh giá từ trải nghiệm thật của người dùng để xem xét khả năng đáp ứng yêu cầu của hệ thống. Các công việc gồm có: chuẩn bị, triển khai thí điểm và lấy phản hồi.

Công việc đầu tiên là cần chuẩn bị một lượng câu hỏi đủ lớn để người dùng vào luyện tập. Đối tượng người dùng trải nghiệm hệ thống là các sinh viên đang ôn tập thi môn KTLT học kỳ 2 năm học 2020-2021. Do đó, các câu hỏi cần nằm trong các chủ đề lập trình khác nhau và có độ khó đa dạng. Nhóm chuẩn bị 2 bài ôn tập với các thông tin như sau:

- Số câu hỏi: 19. Bao gồm: 8 câu dễ, 10 câu trung bình và 1 câu khó.
- Các chủ đề: mảng, chuỗi, con trỏ, hàm, đệ quy.

Bên cạnh đó, nhóm cũng tạo một bài tập được gợi ý câu hỏi gồm 15 câu hỏi về chủ đề Mảng. Sinh viên có thể vào bài tập này để tự luyện tập với các câu hỏi được gợi ý tăng dần độ khó. Thông tin của bài tập này như sau:

- Số câu hỏi dễ, trung bình, khó lần lượt là 5, 5, 5. Tổng câu hỏi là 15.
- Bộ điều kiện từ dễ lên trung bình:
  - Số câu hỏi tối thiểu sinh viên cần làm: 3
  - Điểm tối thiểu cần đạt của 3 lần làm bài cuối: 7.5
  - Cần tối thiểu 3 câu hỏi thỏa đồng thời: điểm tối thiểu là 7.5 và số lần nộp bài tối đa là 3
- Bộ điều kiện từ trung bình lên khó:
  - Số câu hỏi tối thiểu sinh viên cần làm: 3
  - Điểm tối thiểu cần đạt của 3 lần làm bài cuối: 9
  - Cần tối thiểu 3 câu hỏi thỏa đồng thời: điểm tối thiểu là 8.5 và số lần nộp bài tối đa là 3

Nhóm cho sinh viên đăng ký và có 60 người tham gia làm bài tập trên hệ thống. Sau khi sinh viên trải nghiệm hệ thống, nhóm nhờ các bạn đánh giá quá trình sử dụng theo mẫu tại đường dẫn sau:

https://forms.gle/kh14QmJqGptKtWn5A

5.4 Kết quả khảo sát từ người dùng

Nhóm tiến hành thu thập mẫu đơn phản hồi từ các bạn sinh viên và đã nhận được khoảng 40 phản hồi. Mẫu đơn lấy phản hồi từ người dùng gồm 3 vấn đề chính như sau:

- Bạn đánh giá như thế nào về trải nghiệm làm bài?
- Bạn đánh giá như thế nào về tính năng gợi ý câu hỏi?
- Các góp ý khác về hệ thống

Hai câu hỏi đầu tiên gồm các nhận định, được đánh giá trị từ 1 đến 5 tương ứng với rất tệ đến rất tốt. Kết quả đánh giá được tổng hợp bằng cách lấy trung bình điểm số do người dùng phản hồi. Chi tiết về các nhận định và điểm số được trình bày thông qua các bảng 5.3, và 5.4.

Bảng 5.3: *Đánh giá về trải nghiệm làm bài*

Bạn đánh giá như thế nào về trải nghiệm làm bài?	
Tiêu chí đánh giá	**Điểm**
Giao diện mô tả của câu hỏi đầy đủ	4.23
Trải nghiệm với vùng điền code là thoải mái, dễ hiểu	4.12
Thời gian chấm bài và phản hồi chấp nhận được	4.04
Phần hiển thị kết quả của bài nộp gần nhất có đầy đủ những thông tin mà bạn cần	4.08

Bảng 5.4: *Đánh giá về tính năng gợi ý câu hỏi*

Bạn đánh giá như thế nào về tính năng gợi ý câu hỏi?	
Tiêu chí đánh giá	**Điểm**
Độ khó câu hỏi được gợi ý cho bạn là phù hợp	4.15
Độ khó câu hỏi có tăng dần khi bạn làm đúng nhiều	4.31
Giao diện hiển thị kết quả đầy đủ	4.04

Đối với câu hỏi số 3, nhóm nhận được nhiều góp ý và tổng hợp lại một vài ý chính như sau:

- Thanh cuốn vùng điền mã nguồn cần thao tác dễ dàng hơn, tránh bị nhầm lẫn với thanh cuốn của trình duyệt.
- Kết quả testcases hiển thị bằng cửa sổ nổi lên (popup) không trực quan với các testcases

dài.

- Tính năng gợi ý câu "Bài luyện tập về Array": nên cho sửa bài khi làm sai, chỉ khi làm đúng 100% testcases thì mới chuyển sang câu hỏi tiếp theo.

Nhìn chung, các tính năng làm bài và gợi ý câu hỏi đều nhận được các đánh giá tích cực từ phía người dùng. Đây là một tín hiệu tốt để nhóm có thể phát triển thêm các tính năng và sử dụng hệ thống trong việc học tập các môn học lập trình trong Khoa. Mặc dù vậy, trước khi đưa ra sử dụng rộng rãi, hệ thống vẫn cần phải cải thiện những chỗ được phản hồi từ người dùng.

5.5 Kết chương

Chương này trình bày về cách nhóm triển khai hệ thống và các đánh giá đối với hệ thống. Các đánh giá này sẽ chỉ ra các kết quả mà nhóm đạt được cũng như các tồn đọng của hệ thống (được trình bày chi tiết ở chương sau). Trước khi sang chương Tổng kết, nhóm xin điểm lại một số thông tin chính trong chương:

- Nhóm trình bày cách triển khai hệ thống lên các máy chủ với cấu hình cụ thể.
- Sau khi đã triển khai, nhóm tiến hành đánh giá tính năng chấm bài để củng cố cho việc thay đổi con chấm sang Jobe.
- Đồng thời, nhóm cũng tiến hành triển khai thí điểm hệ thống đến với các bạn sinh viên đang ôn tập thi cuối kỳ môn Kỹ thuật lập trình và thu thập phản hồi. Các phản hồi cho thấy phản ứng của người học đối với hệ thống là tương đối tích cực. Tuy nhiên, hệ thống vẫn còn một số điểm cần cải thiện để có thể hoạt động tốt hơn.

Chương 6

Tổng kết

Chương này sẽ đưa ra những đánh giá của nhóm về những ưu điểm của hệ thống và những điểm còn tồn đọng sau qua trình hiện thực để dựa vào đó đề xuất những cải tiến trong tương lai. Bên cạnh đó, chương này còn làm rõ những đóng góp của nhóm với hệ thống trong quá trình cải thiện những tính năng cũ, nâng cao hiệu năng và bổ sung những tính năng mới vào hệ thống.

6.1 Đánh giá ưu nhược điểm của hệ thống

Qua luận văn này, nhóm cơ bản đã hiện thực một hệ thống hỗ trợ thực hành lập trình. Tổng kết lại, nhóm rút ra được những ưu nhược điểm của hệ thống như sau.

6.1.1 Ưu điểm

- Đối với người học:
  - Hệ thống cung cấp một môi trường thực hành lập trình với giao diện trực quan, gọn gàng, có khả năng chấm bài tự động và trả kết quả về cho người học.
  - Hệ thống cho thấy cải tiến về tốc độ chấm bài và sự ổn định so với hệ thống cũ trước đó (được trình bày ở Mục 5.2.1).
  - Hệ thống cung cấp khả năng gợi ý bài tập theo mức độ từ dễ đến khó cho từng người học. Cách thức này tăng hứng thú cho người học lập trình và khuyến khích người học tự luyện tập, từ đó cải thiện khả năng lập trình.
- Đối với người dạy:
  - Hệ thống cung cấp nhiều cách cấu hình câu hỏi, đáp ứng các yêu cầu khác nhau từ người dạy trong việc kiểm tra khả năng của sinh viên thông qua các câu hỏi lập trình.

  - Hệ thống cung cấp một cách tổ chức ngân hàng câu hỏi có khả năng tùy biến cao bằng cách quản lý các câu hỏi theo thư mục. Thêm vào đó, hệ thống cũng có một ngân hàng chung chia sẻ giữa các người dạy với nhau để tạo một ngân hàng đề đa dạng.
  - Tính năng phân loại câu hỏi của hệ thống được dùng làm nguồn tham khảo để đánh giá chất lượng của các câu hỏi. Từ đó, người dạy có thể đưa ra điều chỉnh bổ sung cho bài thực hành hoặc ngân hàng đề.

- Đối với việc phát triển và duy trì hệ thống:
  - Con chấm mới hỗ trợ chấm nhiều ngôn ngữ lập trình, cung cấp khả năng mở rộng cho hệ thống với ít thay đổi.
  - Hệ thống cũng hỗ trợ tăng khả năng chấm bài bằng cách cài đặt thêm các con chấm trên các máy chủ mới, thông qua chứng minh về khả năng mở rộng ở Mục 5.2.2.

6.1.2 Nhược điểm

- Ngân hàng câu hỏi của hệ thống chỉ mới hỗ trợ tìm kiếm câu hỏi theo tên của nó. Việc tìm kiếm này nhằm hỗ trợ người dạy tìm nhanh chóng một câu hỏi để xem nội dung, tái sử dụng, điều chỉnh, v.v... Tuy nhiên, tên câu hỏi thường ngắn, không biểu đạt được hết nội dung của câu hỏi và người dùng đôi khi không nhớ được chính xác từ ngữ có trong tên. Điều này cho thấy việc tìm kiếm theo tên câu hỏi là chưa đủ mạnh, đặc biệt khi số lượng câu hỏi là rất lớn.

- Hiện tại hệ thống chỉ xem xét khả năng của người học sẽ ổn định và tăng sau một thời gian làm bài. Nếu người học đang ở một độ khó nào đó thì không thể quay về độ khó thấp hơn. Tuy nhiên, hệ thống cần xem xét đến trường hợp người học bị giảm khả năng vì một số lý do khách quan, ví dụ với người học sau một thời gian dài không tiếp cận với lập trình. Khi đó, độ khó này sẽ gây mệt mỏi cho người học vì cứ tiếp tục được gợi ý các câu khó hơn năng lực của mình.

- Người dạy vẫn cần phải cấu hình nhiều thông số cho các điều kiện. Dẫn đến tốn nhiều thời gian, công sức để người dạy tìm ra các thông số hợp lý.

- Hệ thống chỉ mới hỗ trợ chấm bài tập thực hành lập trình. Một bài kiểm tra cần cung cấp cả việc kiểm tra lý thuyết bằng các câu hỏi trắc nghiệm, điền từ ngắn hoặc viết luận.

6.2 Đóng góp của nhóm

Sau khoảng thời gian dài gồm cả hai giai đoạn ĐCLV và LVTN, nhóm đúc kết lại các công việc cũng như kết quả mà nhóm đã thực hiện được như sau:

1. Tích hợp con chấm Jobe vào một hệ thống hỗ trợ thực hành riêng. Con chấm đã thực hiện được chức năng kiểm tra tính đúng đắn của bài nộp lập trình với tốc độ nhanh, chịu tải và phân phối tải tốt. Hơn nữa, với kiến trúc đã đề xuất của nhóm, hệ thống hoàn toàn có thể mở rộng được khả năng chấm bài thông qua việc triển khai thêm các con chấm Jobe mới và kết nối với hệ thống thông qua bước chỉnh cấu hình đơn giản.

2. Xây dựng một ngân hàng câu hỏi có thể đáp ứng được nhu cầu thao tác thuận tiện với số lượng lớn câu hỏi. Ngân hàng câu hỏi này được xây dựng dưới kiến trúc cây thư mục phân cấp giúp đem lại sự linh hoạt trong việc sắp xếp, tổ chức câu hỏi. Thêm vào đó, cung cấp một cách thức giúp chia sẻ bài tập giữa các người dạy trong cùng một tổ chức. Từ đó, ngân hàng sẽ là một nền tảng tốt để xây dựng các tính năng liên quan đến câu hỏi, ví dụ như tạo một đề kiểm tra ngẫu nhiên từ ngân hàng câu hỏi.

3. Đề xuất 5 công thức mô tả độ khó. Đồng thời xây dựng hai mô hình phân loại độ khó của câu hỏi dựa trên các công thức này. Mô hình này sử dụng thuật toán phân cụm k-means để phân cụm cho các câu hỏi. Nhóm cũng đề xuất một giải thuật gán độ khó tương ứng cho các cụm khi thực hiện phân cụm trên đầu vào là vector nhiều chiều. Kết quả của các công thức và kết quả phân loại là nguồn tham khảo cho người dạy đánh giá chất lượng câu hỏi.

4. Đề xuất các điều kiện để kiểm tra khả năng của người học đã ổn định với độ khó hiện tại và có thể tăng lên mức độ khó hơn hay không. Thêm vào đó, đề xuất cách lựa chọn ngẫu nhiên một câu hỏi mới gán cho người học để người học ít cảm thấy chán. Từ đó, xây dựng một mô-đun gợi ý câu hỏi lập trình cho người học mang tính khích lệ người học rèn luyện kỹ năng lập trình.

5. Tham gia công bố khoa học về phương pháp phân loại độ khó của câu hỏi lập trình. Phương pháp của nhóm đã được duyệt tại **Symposium on Computer Science & Engineering 2021** với chủ đề: *An efficient approach to measure the difficulty degree of practical programming exercises based on student performances*.

6.3 Hướng phát triển

Bên cạnh các kết quả mà nhóm đạt được, nhóm xem xét các vấn đề tồn đọng của hệ thống và đề xuất các hướng pháp triển cụ thể như sau:

1. Xây dựng mô-đun gắn thẻ từ khóa cho câu hỏi. Từ đó, người dạy có thể tìm kiếm lại các câu hỏi hoặc thư mục này nhanh chóng bằng cách tìm theo từ khóa.

2. Phát triển tính năng gợi ý cho người học theo hướng thích nghi. Nếu người học chưa trả lời được các câu hỏi được gợi ý thì độ khó có thể được hạ xuống. Độ khó sẽ được điều chỉnh tăng hay giảm phù hợp với người học.

3. Tìm cách kiểm tra khả năng của người học ít phụ thuộc vào người dạy hơn. Có thể cố gắng tìm cách giảm số lượng thông số cần cấu hình cho người dạy. Xa hơn là công việc nghiên cứu cách xác định khả năng của người học một cách tự động.

4. Mở rộng hệ thống để hỗ trợ thực hành trên nhiều ngôn ngữ lập trình khác nhau.

5. Phát triển hệ thống cho các dạng câu hỏi khác như trắc nghiệm, điền từ, viết luận,... Xem xét mở rộng hệ thống hỗ trợ các dạng bài tập đặc thù của các môn học khác thuộc ngành Máy tính như automata trong môn Mô hình hóa toán học.

Tài liệu tham khảo

Chris Parr, Mooc creators criticise courses' lack of creativity. https://www.timeshighereducation.com/news/mooc-creators-criticise-courses-lack-of-creativity/2008180.article, 2013; truy cập lần cuối: 00:00 16/05/2021.

Hackerrank, Frequently Asked Questions. https://www.hackerrank.com/faq, 2021; truy cập lần cuối: 00:00 16/05/2021.

CodeLearn, About us. https://codelearn.io/aboutus, 2021; truy cập lần cuối: 00:00 16/05/2021.

Green, M. Moodle. https://moodle.org/, 2021; truy cập lần cuối: 23h00 15/05/2021.

Lobb, R.; Hunt, T. Moodle CodeRunner. https://moodle.org/plugins/qtype_coderunner, 2021; truy cập lần cuối: 23h00 15/05/2021.

Simon, B.; D'Souza, D.; Sheard, J.; Harland, J.; Carbone, A.; Laakso, M.-J. Can computing academics assess the difficulty of programming examination questions? **2012**,

Mahatme, V. P.; Bhoyar, K. K. Questions Categorization in E-Learning Environment Using Data Mining Technique. *International Journal of Information, Control and Computer Sciences* **2016**, *9.0*.

Chen, H.; Ward, P. A. Predicting student performance using data from an Auto-grading system. *arXiv preprint arXiv:2102.01270* **2021**,

Vamsi, S.; Balamurali, V.; Teja, K. S.; Mallela, P. Classifying Difficulty Levels of Programming Questions on HackerRank. **2020**, 301–308.

Awat, K. A. S.; Ballera, M. A. Applying K-Means Clustering on Questionnaires Item Bank to Improve Students' Academic Performance. 2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM). 2018; pp 1–6.

Chowdhury, I.; Watanobe, Y. Cluster Analysis to Estimate the Difficulty of Programming Problems. 2018.

Pérez, E. V.; Santos, L. M. R.; Pérez, M. J. V.; de Castro Fernández, J. P.; Martín, R. G. Automatic classification of question difficulty level: Teachers' estimation vs. students' perception. **2012**, 1–5.

Verdú, E.; Regueras, L.; Verdú, M.; De Castro, J. P. Estimating the Difficulty Level of the Challenges Proposed in a Competitive e-Learning Environment. **2010**, *6096*, 1–5.

OpenJS, F. Introduction to Node.js. https://nodejs.dev/learn, 2021; truy cập lần cuối: 22h00 15/05/2021.

Vue.js, Introduction. https://vuejs.org/v2/guide/, 2021; truy cập lần cuối: 00:00 16/05/2021.

Flask documentation, Flask documentation. 2021; https://flask.palletsprojects.com/en/1.1.x/foreword/#what-does-micro-mean, truy cập lần cuối: 23h00 15/05/2021.

Introduction to Flask, Introduction to Flask. 2021; https://pymbook.readthedocs.io/en/latest/flask.html#introduction-to-flask, truy cập lần cuối: 23h00 15/05/2021.

7 lý do chọn Flask, 7 lý do bạn nên chọn Flask Framework. 2021; https://csc.edu.vn/lap-trinh-va-csdl/tin-tuc/kien-thuc-lap-trinh/7-ly-do-ban-nen-chon-Flask-Framework-4130, truy cập lần cuối: 23h00 15/05/2021.

Redis Labs, Introduction to Node.js. https://redis.io/topics/introduction, 2021; truy cập lần cuối: 22h00 15/05/2021.

Confluent, What is Apache Kafka? https://www.confluent.io/what-is-apache-kafka/, 2021; truy cập lần cuối: 22h00 15/05/2021.

The PostgreSQL Global Development Group, PostgreSQL: The World's Most Advanced Open Source Relational Database. https://www.postgresql.org, 2021; truy cập lần cuối: 22h00 15/05/2021.

Richard Lobb, JOBE. https://github.com/trampgeek/jobe, 2021; truy cập lần cuối: 22h00 15/05/2021.

Richard Lobb, JOBE APIs. https://github.com/trampgeek/jobe/blob/master/restapi.pdf, 2021; truy cập lần cuối: 22h00 15/05/2021.

Moodle Question Bank, Question Bank. https://docs.moodle.org/310/en/Question_bank, 2021; truy cập lần cuối: 03:00 17/05/2021.

van de Watering, G.; van der Rijt, J. Teachers' and students' perceptions of assessments: A review and a study into the ability and accuracy of estimating the difficulty levels of assessment items. *Educational Research Review* **2006**, *1*, 133–147.

Grafana Labs, k6: The best developer experience for load testing. https://k6.io, 2021; truy cập lần cuối: 22h00 15/05/2021.

Phụ lục A

Kết quả phân loại câu hỏi môn KTLT và CTDL&GT

Bảng A.1: *Kết quả phân loại của môn học KTLT*

Tên câu hỏi	F1	F2	F3	F4	Mô hình 2	Mô hình 1	F0
KTLT.01.001	0.78	0.19	0.87	0.60	D	D	0.43
KTLT.01.002	0.90	0.38	0.97	0.37	M	M	0.75
KTLT.01.003	0.90	0.40	0.98	0.35	M	M	0.76
KTLT.02.001	0.94	0.45	0.97	0.34	M	M	0.72
KTLT.02.002	0.96	0.53	0.98	0.29	E	E	0.79
KTLT.02.003	0.97	0.56	0.98	0.31	E	M	0.77
KTLT.02.004	0.89	0.31	0.91	0.48	D	D	0.51
KTLT.03.001	0.96	0.52	0.98	0.33	M	M	0.77
KTLT.03.002	0.96	0.50	0.97	0.34	M	M	0.76
KTLT.03.003	0.93	0.42	0.97	0.36	M	M	0.71
KTLT.04.001	0.99	0.73	0.99	0.26	E	E	0.88
KTLT.04.002	1.00	0.68	1.00	0.30	E	E	0.84
KTLT.04.003	0.89	0.44	0.96	0.43	M	M	0.74
KTLT.04.004	0.93	0.50	0.97	0.39	M	E	0.79
KTLT.04.005	0.91	0.47	0.94	0.43	M	M	0.68
KTLT.05.001	0.87	0.39	0.94	0.41	M	M	0.69

KTLT.05.002	0.98	0.60	0.98	0.31	E	M	0.76
KTLT.05.003	0.93	0.46	0.94	0.34	M	M	0.67
KTLT.06.001	0.97	0.61	0.99	0.30	E	E	0.84
KTLT.06.002	0.98	0.66	0.98	0.29	E	E	0.82
KTLT.06.003	0.99	0.65	0.99	0.29	E	E	0.82
KTLT.06.004	0.99	0.68	1.00	0.27	E	E	0.86
KTLT.06.005	0.98	0.61	0.98	0.31	E	M	0.77
KTLT.07.001	0.96	0.55	0.96	0.33	E	M	0.74
KTLT.07.002	0.98	0.63	0.98	0.28	E	E	0.79
KTLT.07.003	0.83	0.34	0.96	0.42	D	M	0.76
KTLT.07.004	0.93	0.56	0.98	0.32	E	E	0.85
KTLT.08.001	0.98	0.62	0.98	0.27	E	M	0.77
KTLT.08.002	0.95	0.50	0.95	0.31	M	M	0.69
KTLT.08.003	0.97	0.49	0.97	0.31	M	M	0.70
KTLT.08.004	0.89	0.40	0.92	0.36	M	M	0.64
KTLT.08.005	0.98	0.68	0.98	0.23	E	E	0.84

Bảng A.2: *Kết quả phân loại của môn học CTDL&GT*

Tên câu hỏi	**F1**	**F2**	**F3**	**F4**	**Mô hình 2**	**Mô hình 1**	**F0**
CTDL.01.001	0.84	0.32	0.85	0.53	M	M	0.53
CTDL.01.002	0.96	0.69	0.98	0.28	E	E	0.87
CTDL.01.003	0.90	0.40	0.94	0.47	M	M	0.61
CTDL.01.004	0.00	0.00	0.43	0.47	D	D	0.35
CTDL.01.005	0.00	0.00	0.00	0.36	D	D	0.00
CTDL.02.001	0.94	0.51	0.96	0.36	E	M	0.70
CTDL.02.002	0.96	0.50	1.00	0.35	E	M	0.71
CTDL.02.003	0.58	0.22	0.67	0.49	M	D	0.38

CTDL.02.004	0.84	0.38	0.90	0.42	M	M	0.66
CTDL.03.001	0.96	0.42	0.96	0.37	E	M	0.62
CTDL.03.002	0.87	0.33	0.89	0.38	M	M	0.56
CTDL.03.003	0.58	0.16	0.88	0.42	M	M	0.59
CTDL.03.004	0.39	0.11	0.56	0.35	D	D	0.25
CTDL.03.005	0.75	0.27	0.88	0.42	M	M	0.65
CTDL.03.006	0.31	0.08	0.86	0.52	M	M	0.56
CTDL.03.007	1.00	0.87	1.00	0.17	E	E	0.93
CTDL.04.001	0.76	0.33	0.84	0.39	M	M	0.61
CTDL.04.002	0.82	0.29	0.89	0.39	M	M	0.63
CTDL.04.003	0.88	0.41	0.94	0.36	M	M	0.67
CTDL.04.004	0.50	0.17	0.73	0.50	M	M	0.48
CTDL.04.005	0.97	0.74	0.97	0.26	E	E	0.84
CTDL.05.001	0.80	0.35	0.94	0.46	M	M	0.73
CTDL.05.002	1.00	0.65	1.00	0.20	E	E	0.85
CTDL.05.003	1.00	0.86	1.00	0.23	E	E	0.93
CTDL.05.004	0.48	0.18	0.89	0.33	M	M	0.69
CTDL.05.005	0.46	0.18	0.68	0.50	M	M	0.52
CTDL.06.001	0.71	0.32	0.85	0.34	M	M	0.65
CTDL.06.002	0.62	0.30	0.86	0.32	M	M	0.71
CTDL.07.001	0.95	0.48	0.99	0.40	E	M	0.65
CTDL.07.002	1.00	0.95	1.00	0.22	E	E	0.98
CTDL.07.003	0.95	0.56	0.95	0.36	E	E	0.78
CTDL.07.004	0.95	0.59	0.95	0.32	E	M	0.72

Phụ lục B

Kết quả đánh giá hiệu năng sơ bộ của con chấm Jobe

Jobe trả về nhiều loại kết quả tuỳ vào lời gọi cũng như trạng thái của Jobe tại thời điểm gọi. Tuy nhiên, nhóm chỉ chọn quan tâm đến hai thông tin sau khi đánh giá sơ bộ về hiệu năng của Jobe:

- Status code 200: Bài nộp đã được biên dịch và thực thi xong.
- Outcome 15: Bài nộp được hoàn thành mà không có bất cứ lỗi nào khác như lỗi biên dịch hay lỗi thực thi v.v.

Đây là kết quả khi một bài nộp được chấm đúng cách và cũng là điều mà nhóm mong đợi khi Jobe thực hiện chức năng biên dịch và thực thi mã nguồn. Các status code khác hay outcome khác được trình bày chi tiết tại tài liệu của Jobe ở đường dẫn sau (Richard Lobb, 2021).

Nhóm sử dụng K6 (Grafana Labs, 2021) để làm công cụ đánh giá hiệu năng sơ bộ cho con chấm với các thông số cần quan tâm như sau:

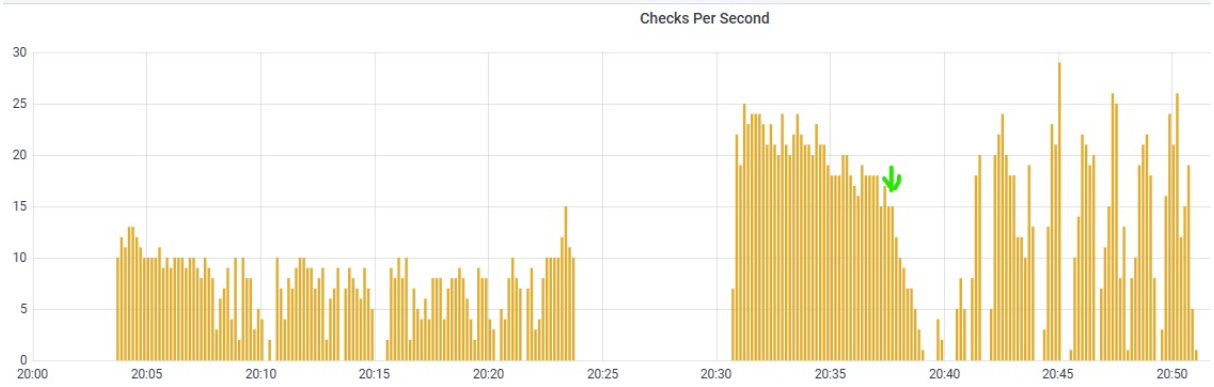
- VUs (virtual users): Số người dùng chạy đồng thời trong quá trình đánh giá
- Duration: Thời gian chạy đánh giá

Kịch bản được thiết lập để đánh giá như sau:

Bảng B.1: *Các kịch bản để đánh giá Jobe*

Thứ tự kịch bản	VUs	Duration (phút)
1	100	2
2	100	1
3	200	2
4	200	1
5	300	2
6	300	1
7	400	2
8	400	1
9	0	5

Kết quả thu thập từ k6 được biểu diễn như hình B.1. Hình này thể hiện hai giai đoạn chấm bài với cùng kịch bản như trên nhưng nửa đầu là với một máy chủ chạy Jobe còn nửa sau là hai máy chủ.



Hình B.1: *Kết quả đánh giá Jobe bằng k6*

Một vài nhận xét mà nhóm đưa ra sau khi đánh giá bằng k6 như sau:

- Jobe có khả năng đáp ứng khá ổn định khi số lượng người dùng sử dụng đồng thời khoảng 100 (VUs = 100), thể hiện qua khoảng 1/4 đầu tiên của mỗi giai đoạn. Trong khoảng 1/4 đầu tiên này, số lượng lời gọi thoả mãn cả hai tiêu chí đề ra là khá ổn định, tương ứng ở mức 10 và 20 lời gọi trên giây cho hai giai đoạn đánh giá.
- Hai con số 10 và 20 cũng cho thấy khả năng mở rộng của Jobe là khả thi, tức khi tăng gấp đôi số lượng máy chủ chạy Jobe thì khả năng đáp ứng cũng tăng lên tương ứng.
- Đối với các mức VUs cao hơn thì Jobe đang cho thấy khả năng đáp ứng không ổn định,

đặt ra một yêu cầu khi sử dụng Jobe là phải có một giới hạn về số bài nộp được chấm đồng thời.

Phụ lục C

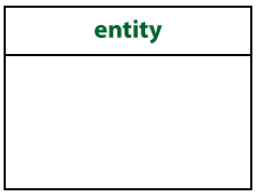
Khái niệm ký hiệu vết chân chim trong sơ đồ quan hệ thực thể

Khi giới thiệu về thiết kế cơ sở dữ liệu của nhóm, nhóm trình bày sơ đồ quan hệ thực thể dựa trên chuẩn ký hiệu vết chân chim (Crow's Foot). Để người đọc dễ theo dõi, chương này nhóm xin trình bày các khái niệm về ký hiệu vết chân chim.

C.1 Thực thể (Entities)

Một thực thể là một đại diện của một lớp đối tượng. Nó có thể là người, địa điểm, sự vật, v.v ... Các thực thể thường có các thuộc tính (attributes) mô tả chúng.

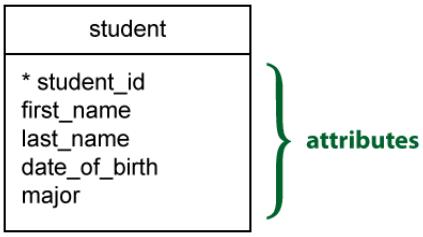
Trong ký hiệu vết chân chim, một thực thể được biểu thị bằng một hình chữ nhật, với tên của nó ở trên cùng.



Hình C.1: *Ký hiệu vết chân chim của Thực thể*

C.2 Thuộc tính (Attributes)

Thuộc tính là một thuộc tính mô tả một thực thể cụ thể.



Hình C.2: *Ký hiệu vết chân chim của Thuộc tính*

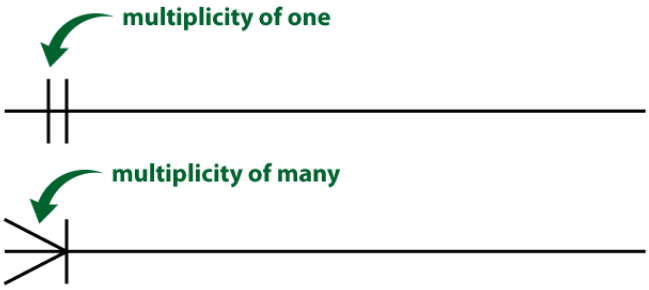
C.3 Quan hệ (Relationships)

Quan hệ minh họa sự liên kết giữa hai thực thể và được trình bày dưới dạng một đường thẳng. Thông thường, các quan hệ trong ký hiệu vết chân chim là quan hệ nhị phân (tức đường thẳng chỉ có 2 đầu mút). Trong mô hình ER, việc biểu diễn mối quan hệ bậc ba hoặc bậc cao hơn được cho là phức tạp nên sẽ không được đề cập.

C.4 Lực lượng (Cardinality)

Mỗi đầu mút của quan hệ đều có 2 ký hiệu nhỏ. Bao gồm:

- **Tính đa dạng** (Multiplicity): số lần tối đa mà một instance của một thực thể có thể được liên kết với các instance trong thực thể liên quan. Nó có thể là một hoặc nhiều.



Hình C.3: *Ký hiệu vết chân chim của tính đa dạng*

- **Tính bắt buộc** (Mandatory/Optional): số lần tối thiểu một instance có thể liên kết đến những instance trong thực thể liên quan. Nó có thể bằng không hoặc một, và dưới 2 hình thức bắt buộc hoặc không bắt buộc

Hình C.4: *Ký hiệu vết chân chim của tính bắt buộc*

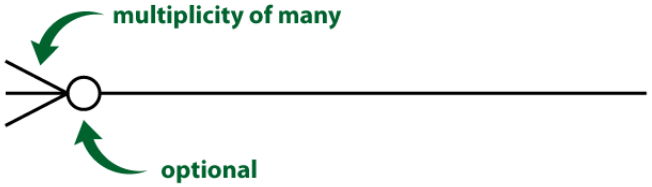
Cả 2 ký hiệu này phải luôn xuất hiện với thứ tự tính đa dạng nằm ngoài cùng và kế tiếp là tính bắt buộc.

Trong ký hiệu vết chân chim:

- Ký hiệu số lượng một trong tính đa dạng hoặc sự bắt buộc trong tình bắt buộc được biểu diễn bằng dấu gạch xuống.
- Ký hiệu dấu chân chim thể hiện số lượng nhiều trong tính đa dạng.
- Ký hiệu tròn thể hiện sự không bắt buộc của tính bắt buộc

Bằng cách phối hợp các ký hiệu này, ta có thể biểu diễn các đầu mút quan hệ sau:

- Không hoặc nhiều (0..N)

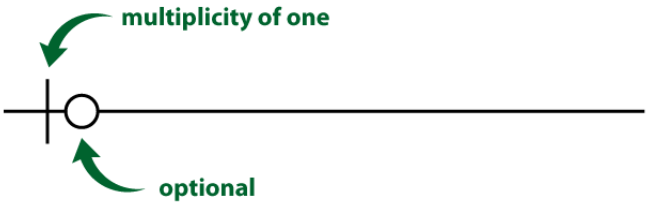


Hình C.5: *Ký hiệu vết chân chim của đầu mút quan hệ Không hoặc nhiều*

- Một hoặc nhiều (1..N)

Hình C.6: *Ký hiệu vết chân chim của đầu mút quan hệ Một hoặc nhiều*

- Không hoặc một (0..1)



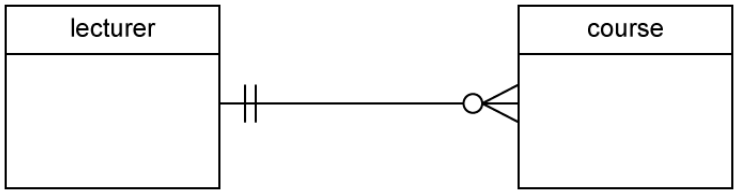
Hình C.7: *Ký hiệu vết chân chim của đầu mút quan hệ Không hoặc một*

- Một và chỉ một (1)

Hình C.8: *Ký hiệu vết chân chim của đầu mút quan hệ một và chỉ một*

Lấy ví dụ, hình C.9 mô tả mối quan hệ một-nhiều giữa hai thực thể Lecturer và Course, cụ thể:

- Một Lecturer có thể có không hoặc nhiều Course
- Một Course phải thuộc về duy nhất một và chỉ một Lecturer



Hình C.9: *Sơ đồ quan hệ thực thể giữa Lecturer và Course*

Phụ lục D

Bài báo nghiên cứu khoa học về chủ đề phân loại độ khó cho câu hỏi lập trình

An efficient approach to measure the difficulty degree of practical programming exercises based on student performances

Huy Tran, Tien Vu Van, Hoang Nguyen Viet,
Duy Tran Ngoc Bao, Thinh Tien Nguyen, and Thanh Van Le
Faculty of Computer Science and Engineering
Ho Chi Minh City University of Technology, VNU-HCM, Vietnam
Email: {huy.tran14; tien.vuvan3499; hoang.nguyen.k2017; duytnb; ntthinh; ltvan}@hcmut.edu.vn

***Abstract*—This study examines the generality of easy to hard practice questions in programming subjects. One of the most important contributions is to propose four new formulas for determining the difficulty degree of questions. These formulas aim to describe different aspects of difficulty degree from the learner's perspective instead of the instructor's subjective opinions. Then, we used clustering technique to group the questions into three easy, medium and difficult degrees. The results will be the baseline to consider the generality of the exercise sets according to each topic. The proposed solution is then tested on the data set that includes the results of the two subjects: Programming Fundamentals, Data Structures and Algorithms from Ho Chi Minh City University of Technology. The most important result is to suggest the instructors complete various degrees according to each topic for better evaluating student's performance.**

***Keywords*—e-learning, difficulty degree, automatic question classification, student's perception, effective coverage exercise**

1. INTRODUCTION

With the help of technology support, teaching and learning processes can be deployed entirely on the Internet [1]. People who join an online course can access all necessary resources with no restrictions and do the assessment tests. The instructors can observe and evaluate the learning process of course participants through well-designed tests.

The platform for online teaching and learning as above is normally called e-learning education system, becoming more and more popular in schools, particularly in universities [2]. The higher level of education, the higher requirement of self-study in the learning process of learners. Moreover, thanks to the online environment, learners can preview educational materials at home and take tests at any time. Overall, the process will become more effective when applying e-learning to teaching and learning. Realizing the above learning foundation's effectiveness, the Faculty of Computer Science and Engineering (CSE) of Ho Chi Minh City University of Technology (HCMUT) has developed and deployed an Auto Grading System (AGS) (see Section 3.1) starting since Semester 1 of 2019.

The world is currently experiencing the COVID-19 pandemic because of its widely substantial spread. To reduce the risk of disease spreading, the government requires keeping social distance and encourages the online education process. In HCMUT, the AGS can completely satisfy the above demand for the programming practice lessons. If the system has sufficiently good exercises, can encourage learners, and is supported by instructional materials, teaching practice programming can be done essentially online.

Besides the above benefits, the programming support system's common drawback is the lack of mechanisms to encourage the appropriate learning and provide a suitable learning path for each learner. Learners usually need to be instructed how to do exercises from easy to complex and the number of questions at each degree should be appropriate. In the case where the question bank has too many easy questions, learners will be bored with them. In the other case where there are many difficult questions, they will be discouraged and gives up. Therefore, an exercise with a sufficient number of questions and appropriate difficulty degree will provide an incentive environment for learners.

The difficulty degree of each question mentioned here should be based on the learners' view, although the actual result of learners depends on the evaluation process including solution designed and grading scale estimated solely defined by instructors. Typically, the average grade point of students is a factor that instructors frequently observe and refer to represent difficulty degree. However, if there is only one factor to consider, there will be a lack of perspective on other aspects, resulting in a one-sided evaluation. Therefore, in this paper, some other factors will be proposed and considered since difficulty degree of learner's point of view may also be revealed through problem solving progress indicators such as the number of submission trials, the solving time duration, etc.

The process of creating questions and assessment their difficulty degree is only from teacher's subjective opinion. The authors in [3] point out that teachers only accurately estimate a fraction of question difficulty compared to learners. That statement raises the question of whether a programming exercise, which is a group of questions, has covered enough different degrees for learners to practice. From our proposed methodology, taking one step further,

we investigate this coverage problem. The results provide an opportunity for teachers to look back to exercises by difficulty level and give teachers some direction to improve the question bank.

To the best of our knowledge, this is the first time the difficulty-related factors are explored in depth and tackled. Our contribution is fourfold:

1) The investigation of studying important factors that affect difficulty degree of programming practice questions.
2) The development of clustering questions based on the learner's perspective.
3) A comparison between difficulty-related factors and between the learner's and teacher's perspective in the view point of difficulty degree of question bank.
4) A real-life application of the problem to detect the lack of ease-to-difficulty degree of each subject in the question bank on the learner's perspective.

The rest of the paper is structured as follows. Section 2 presents related researches about factors that affect question difficulty. Section 3 focuses on our proposed approach to difficulty-related formulas and clustering approach; Auto Grading System will also be briefly introduced in this section. Section 4 shows the experimental results and our evaluation for question classification task. Section 5 will consider coverage of programming topics by making statistics on classification results. Finally, Section 6 concludes this study with results and future researches.

2. RELATED WORK

2.1. Difficulty-related factors

Average score is a factor commonly used to evaluate the difficulty of questions. For instance, the authors Simon et al. only used students' average marks to measure difficulty of programming examination questions [4]. In addition, the ratio of students' marks to the number of students as a weight is proposed by Mahatme's group for categorizing questions in e-learning environment research [5].

Other factors are also considered to describe the difficulty degree of question. When predicting student performance using data from an Auto-grading system, the authors in [6] select four features: the individual passing rate of the best submission, individual testcase outcomes of the best submission, the time interval between the time of submissions and the task deadline, and the number of submissions for classification and regression tasks. In [7], the difficulty degree is also considered to be proportional to the total number of attempts for a problem. In 2018, Awat et al. did an item analysis using the examination results of students [8]. One of the processes in performing item analysis is determining the difficulty level of an item. The item (question) difficulty is stated as the number of correct students divided by the total number of students.

With the objective to estimate the difficulty of programming problems, among information extracted from the dataset, Chowdhury et al. examined clustering by choosing the number of passed students, the number of passed submissions, and many others as clustering features [9].

Table 1 covers five difficulty-related factors that will be examined in this study.

TABLE 1. EXAMINED DIFFICULTY-RELATED FACTORS

Factor	References
Average score	[5], [4]
Number of passed students	[8], [9]
Number of passed submissions	[9], [5]
Max score	[6]
Number of submissions	[6], [7]

2.2. Data mining techniques

The authors in [10], [11] have proposed a fuzzy genetic algorithm to estimate the real difficulty of the questions. K-means [12] algorithm is used to cluster difficulty degree in an e-learning environment [5] and *HackerRank* [7]. After examining many clustering techniques, the authors in [9] focuses on Fuzzy C-Mean Clustering algorithm to estimate the difficulty of programming problems which got high accuracy score on the testing set.

2.3. Programming question coverage

The authors Petersen et al. [13] have evaluated CS1 examinations from a range of schools across North America. They considered the distribution of question types and the average number of concepts besides question contents. The question contents include writing code, reading code, programming concepts and non-programming. The question types are multiple-choice, short answer, writing code, drawing diagrams. There are 28 question concepts in this research; some fundamental concepts are trivial syntax, variables, function structure and expressions.

3. PROPOSED APPROACH

3.1. Auto Grading System

Auto Grading System (AGS in short) is a system that supports practicing programming, built and used in the Faculty of CSE of HCMUT. In this research, we will focus on two key features of AGS, which are:

- Manage the questions bank (add/modify), and assign a set of questions to appropriate groups of students.
- Grade the submissions of students through an automatic mechanism.

The two sections below describe these two key features in detail.

3.1.1. Manage and assign the questions. The instructor, who manages the laboratory class, is required to setup an exercise for every topic of knowledge in a course. An exercise contains some relative questions to evaluate the students' understanding about the topic. One question in the AGS system must be configured with the main component, i.e., a suite of input as testcases for submissions. This suite is used in the grading process for submission.

After finishing setting up the exercise, the instructor needs to assign it to groups of students. Some interesting fields of data to configured in assigning question are:

- The maximum number of submissions.
- The start and end time for submitting an answer.

When the assigning step is done, students can start doing this exercise.

3.1.2. Automate grading the submissions of students. Whenever AGS system records a submission for a question from a student, it compiles the submission source code then runs with the configured testcases to produce a set of output. The submission score is determined by the number of correct testcases that represented in the output set.

Following this phase, we can collect all the score from submissions of students which forms the data source for this paper, including some useful information inferred from the data source, such as:

- The average score of submissions for a question
- The number of students that passed a threshold
- The number of submissions that passed a threshold
- The best submission of a student for a question
- The number of submissions of a student for a question

3.2. Difficulty-related formulas

For readability, in this research, all following words including *Average score*, *Passed submissions*, *Passed submissions*, *Best submission*, *Number of submissions* will be used as a factor or a formula name interchangeably.

Our research proposes 4 formulas that related to the difficulty of programming questions and constructed based on student's submission results. By observing all 4 formulas, we aim to describe the difficulty of programming questions based on student's performances. To increase the reliability of our research, we simultaneously compare those formulas with the average score formula, which is a commonly used formula to determine the difficulty degree of questions.

3.2.1. Average score. Average score is a factor that is widely used to describe difficulty as the mean of student scores. Because programming questions often have many submissions (as for trying and correcting), a student's score is the mean of all his/her submission scores, which is then normalized to the range [0-1] based on max score, then calculate the mean score according to students. The formula is stated as

$$F_0 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{C_i}\sum_{j=1}^{C_i}\frac{c_{ij}}{C_{max}}\right) \quad (1)$$

where N is the number of students answering the question, C_i is the number of submissions of the ith student, c_{ij} is the score of the jth submision of the ith student, and C_{max} is the maximum score of the question.

3.2.2. Passed students. Passed students refers to information about the number of students who passed the test. Since a difficult question will have few students finding a solution within the time limit, students who do not have the submission will not be included. The formula is then normalized to range $[0-1]$. The proposed formula is

$$F_1 = \frac{S}{S_{tot}} \quad (2)$$

where S is the number of passed students, and S_{tot} is the number of students who had at least one submission for the question.

3.2.3. Passed submissions. Passed submissions refers to the number of passed submissions for a question. For programming questions, students typically stop submit when they have passed them. For a difficult question, students will have a few failed requests before reaching the passed one. Therefore, the ratio of the number of passed submissions to the number of total submissions will be low. Conversely, for an easy question, the number of failed submissions is low and that ratio will be high. Additionally, if a person is recognized as failed on *Passed students*, Passed submissions gives extra information about the number of failed requests. The proposed formula is:

$$F_2 = \frac{U}{U_{tot}} \quad (3)$$

where U is the number of passed submissions, and U_{tot} is the number of total submissions.

3.2.4. Best submission. Best submission addresses the student's submission score. The easier the question, the higher the student's score on the question. During the time the question is open, it is possible that the student did not get a good score at the beginning, but after a while, the submission improved, the score increased. Hence, *Best submission* suggests taking the highest score in a student's submissions for the question.

$$F_3 = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{\max\{Ci*\}}{C_{max}}\right) \quad (4)$$

where N is the number of students, C_{max} is the maximum score, and C_{i*} is the score set of the ith student's submissions for the question.

3.2.5. Number of submissions. Number of submissions refers to the number of times a student needs to work to achieve a question. The harder the question, the more times it has to be done. However, because AGS provides an editor to fill the code directly, students who do not pass a test will often change a little code right on the system and submit their code without checking carefully on the IDE. This approach increases the number of submissions but does not help improve student skills. The AGS system can provide and fulfill the following conditions:

- The number of tests is limited for learners to try and carefully work on each submission. Careful work helps the data reflect the student's work effort.
- The number of questions is large enough that learners switch to another question when one question is finished.
- The time to open the question is not too much for learners to spend time doing different questions, not too much time left to do many passes for one question.

The proposed formula is

$$F_4 = \frac{1}{N}\sum_{i=1}^{N}\frac{U_i}{U_{max}} \tag{5}$$

where N is the number of students, U_i is the number of submissions of ith student, U_{max} is the maximum number of submissions for the question.

3.2.6. Comments about difficulty-related formulas. Table 1 summarizes the five formulas introduced above with their range of values and properties.

TABLE 2. SUMMARY OF FORMULAS

Notation	Name	Range of values	Properties
F_0	Average score	[0-1]	The bigger the easier
F_1	Passed students	[0-1]	The bigger the easier
F_2	Passed submissions	[0-1]	The bigger the easier
F_3	Best submission	[0-1]	The bigger the easier
F_4	Number of submissions	[0-1]	The bigger the harder

To this study's concern, each formula F_1, F_2, F_3, and F_4 provides various aspects regarding the difficulty. Of all these perspectives, it can be seen that 3 main factors that affect difficulty are students, number of submissions, and grades. Furthermore, each formulation may have more than one contributing factor with varying degrees. Table 3 describes the above 3 factors with 3 contributing degrees 0, 1, 2.

- If a factor is not shown in the formula, its contributing degree is 0.
- If the formula is relevant to the factor, the factor contribution is at degree 1.
- If a factor is an inseparable part of the formula, its contributing degree is 2.

Table 3 also shows that each formula has a different combination of factors' contributing degree. This study aims to recognize difficulty on many different aspects, so our research model uses a feature vector $< F_1, F_2, F_3, F_4 >$ with four corresponding values of formula F_1 to F_4 to describe the difficulty for that question.

TABLE 3. FACTORS THAT AFFECT DIFFICULTY OF A PRACTICAL PROGRAMMING QUESTION

Formula	Student	Submission	Score
F_1	2	0	1
F_2	0	2	1
F_3	1	0	2
F_4	1	2	0

3.3. Clustering approach

Clustering is a technique of grouping similar data without being affected by a specific purpose other than data points themselves. K-mean is the well-known clustering technique published as a journal article in [12]. The algorithm performs the following steps:

- Select K cluster centers at random.
- For each data point, calculate the distance to each center and assign that point to cluster with the nearest center.
- Recalculate the new center for each cluster by calculating the new average point in each group.
- If the stopping condition is satisfied, then stop. Otherwise, repeat step (2).

The stopping condition may be the maximum number of iterations reached, or the displacement of the centers between two adjacent iterations is lower than a defined threshold.

This study does not focus on comparing and selecting the better clustering methods. K-means is appropriate to metric, easy to capture the structure of data and guarantee the convergence. Moreover, due to its frequent occurrence in categorizing questions [5], [7], we choose k-means as the clustering technique in this research.

After clustering, we choose Silhouette Score to measure the goodness of the result. The Silhouette Score is calculated as

$$\frac{b-a}{\max(a,b)}$$

where a is the mean distance from a sample to the other samples in the same cluster, b is the distance between a sample and the nearest cluster that the sample is not a part of [14]. The range of the Silhouette Score is between -1 to

1. The sample is considered to have been assigned to the correct cluster if close to 1. Whereas the value -1 implies that a sample has been assigned to the wrong cluster.

3.4. Coverage approach

The authors in [13] give some directions to do coverage in terms of content and type of questions. However, in this study, the question content and the question type are both writing code; the concept (we call the topic) of a question is simply the topic it belongs to. Taking another direction, the main focus of our study is the difficulty degree of questions. Therefore, this study statistics the number of questions according to each of the difficulty degrees for each programming topic.

4. QUESTION DIFFICULTIES CLASSIFICATION

This section describes the experiments and evaluation for question difficulties classification. *Pandas* tool [15] helps manipulate tabular data, which is used for preprocessing and calculating formulas' values for each question. *Scikit-learn* [16] package is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. Our study used k-means algorithm from *Scikit-learn* to cluster question's difficulty.

4.1. Methodology

This study proposes two clustering models to categorize difficulty degree of programming questions into three degrees of easy, moderate and difficult. The clustering results of two models then will be statistics for evaluation in each programming topic.

The experiment dataset is two submission results of two laboratory courses: FP (Fundamentals of Programming) and DSA (Data Structures & Algorithms). These data are obtained from AGS exercise data in semester 2, academic year 2019-2020. Table 4 gives more information on the dataset.

TABLE 4. DATASET INFORMATION

	Number of questions	Number of students	Number of submissions	Number of topics
FP	32	673	38543	8
DSA	32	58	2604	7

4.2. Clustering models for determining question's difficulty

This study proposes two clustering models:

1) Model 1: use k-mean of scikit-learn with the number of clusters is 3, other parameters are let as default. The information for a training point is a value from formula F_0.
2) Model 2: use k-mean of scikit-learn with the number of clusters is 3, other parameters are let as default. The information for a training point is a vector of 4 values $< F_1, F_2, F_3, F_4 >$.

4.3. Clustering results

The Silhouette Score of each model is relatively good (Table 5).

TABLE 5. SILHOUETTE SCORE OF MODELS

	Model 1	Model 2
FP	0.69	0.66
DSA	0.78	0.61

The clustering results of two models of two courses are presented in Table 6 and Table 7. Each degree of easy, moderate, and difficult is marked respectively in tables as E, M, D.

4.4. Method to assign degrees to clusters

In model 1, after clustering, we find the center of each cluster represented by a scalar. According to the properties of average score in Table 2: the biggest center corresponds to the easy cluster, the smallest center corresponds to the difficulty cluster, and the last center corresponds to the moderate one.

In model 2, we also find these three centers, but with a four values vector interprets each one, it is impossible to determine the smaller and greater relationship between these two vectors. We define variable $score$ as the score of a permutation of these three vectors and proceed with the following steps:

1) Generate six permutations of three 4-dimensional vectors.
2) For every two adjacent vectors, consider all pairs of values belonging to the same formula; if these two values satisfy the properties in the Table 2 of that formula, then increase $score$ by 1, assuming you need to sort these vectors with increasing difficulty.
3) Choose the permutation with the highest $score$, assign the clusters of the centers of this permutation with easy, medium, and difficult degrees, respectively.

The assignment manner for model 2 can fail if there is more than one permutation with the same highest score, then we don't know which permutation to assign the cluster. With experiment data, there is only one permutation with the highest score and cluster assignment is achieved.

TABLE 6. CLUSTERING RESULT OF FP

Question	1	2	3	4	5	**6**	**7**	8	9	10	11	12	13	**14**	15	16	**17**	18	19	20	21	22	**23**	**24**	25	**26**	27	**28**	29	30	31	32
Model 1	D	M	M	M	E	M	D	M	M	M	E	E	M	E	M	M	M	M	E	E	E	E	M	M	E	M	E	M	M	M	M	E
Model 2	D	M	M	M	E	E	D	M	M	M	E	E	M	M	M	M	E	M	E	E	E	E	E	E	E	D	E	E	M	M	M	E

TABLE 7. CLUSTERING RESULT OF DSA

Question	1	2	3	4	5	**6**	**7**	**8**	9	**10**	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	**29**	30	31	**32**
Model 1	M	E	M	D	D	M	M	D	M	M	M	M	D	M	M	E	M	M	M	M	E	M	E	E	M	M	M	M	M	E	E	M
Model 2	M	E	M	D	D	E	E	M	M	E	M	M	D	M	M	E	M	M	M	M	E	M	E	E	M	M	M	M	E	E	E	E

4.5. Evaluation of clustering result

In course FP, we can see that there are 7 questions with different results between two models: 6, 14, 17, 23, 24, 26, 28. Questions 6, 14 are clustered by model 2 as more complex than model 1. The remaining questions are clustered by model 2 as easier than model 1.

In course DSA, we can see that there are 6 questions with different results between two models: 6, 7, 8, 10, 29, 32. All of them are clustered by model 2 as easier than model 1.

There is no question in both FP and DSA that the two models give conflict results.

Generally, model 2 tends to rank the degree as easier than model 1. We can see it more evident in DSA, maybe because DSA is a more challenging course than FP. Students may not get high marks on the first few tries and even got 0 points, affecting the mean score to get lower. Nevertheless, after thinking and trying the test, maybe students will eventually pass the test with the highest score. At that point, the student's mean score will not be high, but other aspects such as the highest score, the number of submissions, and the number of accomplished students may give more information to have a better perspective about the question.

Table 8 shows the calculated values of 5 formulas corresponding to questions 6 and 7 in DSA. According to *Average score*, we see the average score is about 7/10 points and ranked by model 1 as moderate. However, the rate of passed students is about 95%, the highest average score is 9.6/10 with question 6 and nearly 10/10 for question 7. The submission rate is also relatively low; if students have a maximum of 5 submissions, they only need 2 times to pass that question. Also, *Passed submissions* has a reasonable value: in two submissions, the first one is unsuccessful and the second one succeeds, so the rate of submission is about 50%.

Although there is no clear conclusion about difficulty, questions with different results as above should be considered further with our four proposed formulas in addition to the *Average score*. They are assisting teachers in doing reviews and deciding appropriate actions.

TABLE 8. FORMULA VALUE FOR QUESTION 7 AND 8 OF DSA

Question	F_0	F_1	F_2	F_3	F_4
6	0.70	0.94	0.51	0.963	0.36
7	0.71	0.96	0.50	0.996	0.35

We propose two measures for evaluating the clustering result:

- **The sameness**: the percentage of the number of questions that two models have the same degree.
- **The similarity**: the percentage of the number for which the degree from two models is similar. Two degrees are considered similar if they are the same or less likely close but not contradictory.

Table 9 records two above measures of courses FP and DSA. We can see that the results of model 2 have high matching with one of model 1.

TABLE 9. THE SAMENESS AND SIMILARITY OF CLASSIFICATION RESULTS

	The sameness	**The similarity**
FP	78.13%	100%
DSA	81.25%	100%

5. ANALYSIS OF PROGRAMMING TOPIC'S COVERAGE

5.1. Statistics of degree for each programming topic

To observe the programming topic's coverage, we use classification results from 2 models and does statistics on the number of questions of each difficulty degree. The statistical results are shown in Table 10 and Table 11. The columns *Es*, *Ms*, and *Ds*, respectively correspond to the number of easy, medium and difficult sentences in each topic.

TABLE 10. STATISTICS OF DEGREE IN COURSE FP

		Model 1			Model 2		
Index	**Topic**	**Es**	**Ms**	**Ds**	**Es**	**Ms**	**Ds**
1	Loop & If ... Else	**0**	2	1	**0**	2	1
2	Array	1	2	1	2	1	1
3	String	**0**	3	**0**	**0**	3	**0**
4	Function and parameter passing	4	1	**0**	5	**0**	**0**
5	Recursion	3	2	**0**	2	3	**0**
6	Pointer	**0**	3	**0**	1	2	**0**
7	Link list	2	2	**0**	3	**0**	1
8	OOP	1	4	**0**	2	3	**0**

TABLE 11. STATISTICS OF DEGREE IN COURSE DSA

		Model 1			Model 2		
Index	**Topic**	**Es**	**Ms**	**Ds**	**Es**	**Ms**	**Ds**
1	C/C++ Review, Recursion by C/C++	1	2	2	1	2	2
2	Implement AList Class with Template Programming	**0**	3	1	2	2	**0**
3	Implement singly linked list and its applications	1	5	1	2	4	1
4	Implement Stack and Queue with its applications	1	4	**0**	1	4	**0**
5	Implement Binary Tree and Binary Search Tree	2	3	**0**	2	3	**0**
6	Implement AVL and its applications	**0**	2	**0**	**0**	2	**0**
7	Implement heap, hash and their applications	2	2	**0**	4	**0**	**0**

5.2. Evaluation of statistical results

In Table 10, 6 out of 8 topics with different difficulty coverage are topics 2, 4, 5, 6, 7, 8. Consider topic 7 (Linked list): according to model 1, the classification has 2 easy questions, 2 moderate questions; according to model 2, there are 3 easy questions and 1 difficult question. The coverage of model 1 is plausible if the instructor wants the learners to do basic exercises. Teachers want students to get used to Linked lists, since Linked lists are quite advanced topics for an introduction course. The coverage of model 2 shows students' suitability if the teachers want learners to challenge some difficult questions. Also, it offers some lack the average question.

In Table 11, 3 out of 7 topics with different difficulty coverage are topics 2, 3, 7. In topic 2, model 1 shows a lack of easy questions, while model 2 shows a lack of difficult questions. In topic 7, model 1 shows a lack of difficult questions, while model 2 shows a lack of moderate and difficult questions.

Therefore, two models can give different coverage for a topic. Although there is no clear conclusion as to which model is better in the above coverage, two observations can be made as follows:

- Firstly, for topics with similar coverage in two models, which states that the coverage has a high consensus, we can consider that coverage is reliable for representing the difficulty degree of the topic.
- Secondly, if the number of questions on the same difficulty degree in the two models is the same and equal to zero, it shows the lack of questions in this difficulty.

The instructor can rely on the above two observations to:

- Observe coverage if there is a similar coverage.
- Adding more questions to the difficulty degree of a topic where the number of questions is 0 in both models.

6. CONCLUSION

In this study, we aim to fulfill the difficulty degrees coverage of practical programming questions. We have proposed formulas related to the average score, passed students, passed submissions, best submission and the number of submissions to determine the difficulty of each question. Having proposed formulas, we use clustering techniques to group questions into three groups. The results will be an opportunity to look at the coverage of the topic-by-topic practice exercise set. Moreover, the results are directed to suggest the instructor to supply the questions with the missing difficulty degree according to each topic. These results are analyzed not based on the subjective opinions of the instructor but by observing the information set that stores not only the results of the learners but also the whole process of the work to submit.

The proposed solution has been measured across two subjects: Fundamentals of Programming and Data Structures & Algorithms. Data collected from these subjects through the second semester of 2019 at Ho Chi Minh City University of Technology. In view, the results of the study are as follows:

- Classify questions difficulty based on 2 models: one uses the formula using the average score formula and the other uses 4 formulas that we proposed. With the results, although there are some questions with different degrees of difficulty, in general, the two models give similar classification results, and they are reasonable.
- Conduct difficulty degree statistics on each programming topic. The results showed that there are differences in the difficulty degree coverage between model 1 and model 2. The study proposes to produce both classification results, along with the value of 4 proposed formulas and 1 average score formula. From there, the instructor will have many different perspectives to make decisions to add the necessary questions to the topic.

In the future, we still focus on these directions for this study:

- Enrich the dataset to enhance the accuracy of clustering models.
- Widen the scope of the fulfilling question difficulty coverage problem. This study is only focused on fulfilling the degree to which its amount of questions is zero. However, with a more significant number of questions due to enrichment, the zero value mostly will not be appeared. Therefore, the fulfilling method will not depend on the zero value and need a threshold-based configuration.
- Utilize the classification results for recommending the question with suitable difficulty for students. Furthermore, the next stage is the system to suggest programming practicing path for student.
- Expand the degrees of question difficulty, determine a number of degrees for having sensible jumps between degrees.

REFERENCES

[1] C. Coman, L. Ţîru, L. Mesesan Schmitz, C. Stanciu, and M. Bularca, "Online teaching and learning in higher education during the coronavirus pandemic: Students' perspective," pp. 1–2, 2020.

[2] Ł. Tomczyk, K. Potyrała, A. Włoch, J. Wnek-Gozdek, and N. Demeshkant, "Evaluation of the functionality of a new e-learning platform vs. previous experiences in e-learning and the self-assessment of own digital literacy," p. 2, 2020.

[3] G. van de Watering and J. van der Rijt, "Teachers' and students' perceptions of assessments: A review and a study into the ability and accuracy of estimating the difficulty levels of assessment items," *Educational Research Review*, vol. 1, no. 2, pp. 133–147, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1747938X06000236

[4] B. Simon, D. D'Souza, J. Sheard, J. Harland, A. Carbone, and M.-J. Laakso, "Can computing academics assess the difficulty of programming examination questions?" 11 2012.

[5] V. P. Mahatme and K. K. Bhoyar, "Questions categorization in e-learning environment using data mining technique," *International Journal of Information, Control and Computer Sciences*, vol. 9.0, no. 1, jan 2016. [Online]. Available: https://doi.org/10.5281/zenodo.1338630

[6] H. Chen and P. A. Ward, "Predicting student performance using data from an auto-grading system," *arXiv preprint arXiv:2102.01270*, 2021.

[7] S. Vamsi, V. Balamurali, K. S. Teja, and P. Mallela, "Classifying difficulty levels of programming questions on hackerrank," pp. 301–308, 2020.

[8] K. A. S. Awat and M. A. Ballera, "Applying k-means clustering on questionnaires item bank to improve students' academic performance," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 2018, pp. 1–6.

[9] I. Chowdhury and Y. Watanobe, "Cluster analysis to estimate the difficulty of programming problems," 11 2018.

[10] E. Verdú, L. Regueras, M. Verdú, and J. P. De Castro, "Estimating the difficulty level of the challenges proposed in a competitive e-learning environment," vol. 6096, pp. 1–5, 2010.

[11] E. V. Pérez, L. M. R. Santos, M. J. V. Pérez, J. P. de Castro Fernández, and R. G. Martín, "Automatic classification of question difficulty level: Teachers' estimation vs. students' perception," pp. 1–5, 2012.

[12] S. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[13] A. Petersen, M. Craig, and D. Zingaro, "Reviewing cs1 exam question content," in *Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011, pp. 631–636.

[14] "sklearn.metrics.silhouette_score," last access: 19h30 28/05/2021. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

[15] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4524629

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.