TRƯỜNG ĐẠI HỌC TIỀN GIANG

KHOA KỸ THUẬT CÔNG NGHỆ

-----🙞🙜🕮🙞🙜-----



ĐỒ ÁN KHÓA LUẬN

NGÀNH CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI

XÂY DỰNG PHẦN MỀM QUẢN LÝ VÀ HỔ TRỢ TỰ HỌC TRẮC NGHIỆM STUDY-SPACE

Người hướng dẫn: KS. Nguyễn Thái Duy

ThS. Trần Thị Diễm Trang

Sinh viên thực hiện:

Đặng Hữu Lộc 018101027

Nguyễn Ngọc Minh Yến 018101108

*Tiền Giang, ngày tháng năm 2021*

LỜI NÓI ĐẦU

Hiện nay, các bạn học sinh phải chuẩn bị rất nhiều tài liệu, sách vở khi đến trường, đặc biệt đối với các bạn thi chuyển cấp phải đem lượng lớn đề kiểm tra, đề thi thử ở lớp và cả lớp học thêm để thuận tiện cho việc ôn tập. Việc giữ nhiều tài liệu cho nhiều môn học khiến các bạn gặp nhiều khó khăn khi bảo quản, sắp xếp cũng như tìm kiếm chúng.

Bên cạnh đó, việc phát tài liệu học tập, đề kiểm tra tại các trường hiện nay vẫn hoạt động truyền thống thông qua việc in ấn. Hình ảnh một bạn đại diện lớp xách hai tay hai thùng giấy lớn tài liệu không còn là điều xa lạ. Điều này gây mất thời gian khi giáo viên tìm giao cho một bạn, cả lớp phải liên hệ bạn để nhận bài. Đồng thời gây bất tiện khi phải vận chuyển rất nhiều giấy tờ cùng lúc, lãng phí giấy mực và tốn nhiều sức lực.

Đồng thời giáo viên ở trường vừa dạy theo chương trình vừa tổ chức ôn tập nên rất ít thời gian để sửa hết các đề. Khi học thêm, một lớp ôn luyện có nhiều học viên, điều này dẫn đến giáo viên khó có thể chấm điểm từng bài, hay hướng dẫn cụ thể cho từng bạn. Vì vậy tự ôn luyện tại nhà luôn được các thầy cô khuyến khích.

Mặt khác, nhiều giáo viên có phần e dè khi sử dụng công nghệ, đặc biệt là tham gia giảng dạy trực tuyến như hiện nay vì khó sử dụng, khó thao tác. Dẫn đến việc hạn chế khả năng hỗ trợ các bạn học.

Thế nhưng việc ôn luyện tại nhà mang lại nhiều mặt tốt nhưng lại khó có thể tìm người hướng dẫn khi không tìm ra cách giải bài tập. Điều này khiến nhiều bạn gặp nản chí trong việc tự học.

Vì vậy, chúng tôi muốn xây dựng “ỨNG DỤNG ÔN LUYỆN TRẮC NGHIỆM STUDY-SPACE” giúp giải quyết những vấn đề trên và phát triển thêm nhiều tiện ích, mang lại không gian học tập tốt nhất cho người dùng.

LỜI CẢM ƠN

🙞🙞🕮🙜🙜

Chúng tôi xin gửi lời cảm ơn chân thành đến cô Trần Thị Diễm Trang và anh Nguyễn Thái Duy là người hướng dẫn đồ án cho chúng tôi. Cô và anh đã luôn theo sát quá trình thực hiện đồ án, nhiệt tình hướng dẫn, chỉ bảo, động viên để chúng tôi hoàn thành đồ án này.

Trong quá trình làm đồ án, mặc dù đã rất cố gắng nhưng trong quá trình làm đồ án không thể tránh khỏi những thiếu sót. Rất mong nhận được sự thông cảm và góp ý tận tình của quý Thầy/Cô và các bạn.

Cuối cùng chúng tôi kính chúc quý Thầy/Cô dồi dào sức khỏe và thành công trong công việc.

Xin chân thành cảm ơn!

ĐỀ CƯƠNG CHI TIẾT KHÓA LUẬN TỐT NGHIỆP

**Tên đề tài**: Xây dựng ứng dụng quản lý và tự học trắc nghiệm Study-Space.

**Người hướng dẫn**: ThS. Trần Thị Diễm Trang, Ks. Nguyễn Thái Duy

**Sinh viên thực hiện**:

Đặng Hữu Lộc MSSV: 018101027

Nguyễn Ngọc Minh Yến MSSV: 018101108

**Loại đề tài**: Xây dựng phần mềm.

**Nội dung đề tài**:

* Bối cảnh:

Ngày nay, việc làm bài, ôn bài và kiểm tra theo hình thức trắc nghiệm dần trở nên phổ biến, nhất là đối tượng học sinh, sinh viên. Hình thức kiểm tra này mang lại nhiều lợi ích như giúp kiểm tra tổng quát kiến thức đã học một cách nhanh chóng, tiết kiệm thời gian và công sức của giáo viên trong việc chấm điểm. Hầu hết các môn học hiện nay đều sử dụng hình thức kiểm tra trắc nghiệm.

Ví dụ, mỗi năm một học sinh cấp 3 phải học 9 môn chính, mỗi môn có trung bình 10 chương, mỗi chương có khoảng 2 tài liệu trắc nghiệm. Vậy trong quá trình học, một học sinh phải quản lý và bảo quản một lượng tài liệu rất lớn và rời rạc. Từ đó ta thấy, cần xây dựng một phần mềm hỗ trợ việc lưu trữ và quản lý những bài tập trắc nghiệm này một cách dễ kiểm soát.

Hiện nay, có nhiều phần mềm hỗ trợ tạo bài trắc nghiệm và thống kê kết quả kiểm tra nhưng không có những chức năng phù hợp với nhu cầu hổ trợ quản lý, lưu trữ các bài tập trắc nghiệm và hổ trợ tự học cho đối tượng học sinh, sinh viên.

Vì vậy, nhằm giải quyết những vấn đề trên việc xây dựng “Phần mềm quản lý và hỗ trợ tự học trắc nghiệm Study-Space” là vấn đề cần thiết.

* Phân tích, thiết kế, cài đặt, thử nghiệm:
  + Thiết kế Usecase cho từng nhóm người dùng.
  + Xây dựng kịch bản cho các chức năng.
  + Thiết kế cơ sở dữ liệu cho hệ thống, chia thành 3 nhóm riêng “quản lý nhóm”, “quản lý đề tài”, “quản lý làm bài”.
  + Xây dựng hệ thống theo kiến trúc Microservices.
  + Thiết kế giao diện trực quan với các chức năng cần thiết.
  + Cài đặt ứng dụng trên Docker Container để chạy thử và kiểm lỗi thường xuyên.
  + Báo cáo và chỉnh sửa theo yêu cầu.
* Yêu cầu sinh viên:
  + Có kiến thức phân tích thiết kế hệ thống thông tin.
  + Có kiến thức về thiết kế hướng lĩnh vực.
  + Hiểu biết cơ bản về kiến trúc Microservices.
  + Có kinh nghiệm về lập trình Back-end với Spring Framework và lập trình Front-end với React.
  + Biết sử dụng công cụ test Cypress và Postman.
  + Có kiến thức về các ngôn ngữ được sử dụng trong dự án: Java, Typescript, HTML, CSS.
* Nghiên cứu tài liệu chuyên môn:
  + Nghiên cứu thực tế: nghiên cứu mô hình Microservices thiết kế hệ thống, đồng thời kết hợp sử dụng React, Spring Boot, Spring Data R2DBC, hỗ trợ triển khai hệ thống bằng Kubernetes và hệ quản trị cơ sở dữ liệu MySQL.
  + Xác định phạm vi đề tài: Đề tài xây dựng nhằm mục đích phục vụ cho việc quản lý và hỗ trợ tạo bài trắc nghiệm và tự ôn tập trắc nghiệm cho các học sinh, sinh viên.
  + Nghiên cứu tài liệu chuyên môn: Nghiên cứu tài liệu tổng quan về thiết kế hướng lĩnh vực của Eric Evans, Microservices từ thiết kế đến triển khai, tài liệu hướng dẫn của Spring Framework.
* Kết quả dự kiến của đề tài:
  + Công dụng phần mềm: Quản lý và hỗ trợ việc tự học trắc nghiệm cho học sinh, sinh viên.
  + Chức năng:
    - Đăng nhập hệ thống.
    - Quản lý tài khoản.
    - Phân quyền cho tài khoản người dùng.
    - Quản lý kho câu hỏi trắc nghiệm.
    - Quản lý đề bài.
    - Quản lý bài làm, tính điểm, thống kê kết quả.
    - Quản lý nhóm học theo các môn, hạn chế quyền truy cập tài nguyên theo nhóm.
    - Quản lý nhóm và thành viên nhóm.
* Đặc điểm chính:
  + Quản lý đề bài, tạo đề ngẫu nhiên từ kho câu hỏi.
  + Xuất đề bài thành file word hoặc excel.
  + Ứng dụng hoạt động được trên nền tảng Web.
  + Hệ thống xây dựng theo mô hình Microservices, kết nối với nhau qua Kafka.
* Yêu cầu cần đạt:
  + Hệ thống xây dựng thành công bằng mô hình Microservices, đảm bảo tính phân quyền và bảo mật, hoạt động ổn định.
  + Phần mềm đầy đủ chức năng theo yêu cầu.
  + Giao diện hệ thống đơn giản, dễ sử dụng cho mọi đối tượng sử dụng, bố cục hợp lí, các menu chức năng rõ ràng.
  + Ứng dụng chạy tốt trên nền tảng Web.
* Yêu cầu công nghệ:
  + Cần cài đặt Java OpenJDK phiên bản 11, NodeJS và NPM phiên bản LTS 14.
  + Intellij IDEA Community, Visual Studio Code. Docker và docker-compose (cài đặt Docker Desktop khi dùng Win10), Kafka Desktop client Conduktor.
* Kế hoạch thực hiện:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Các nội dung, công việc cần thực hiện** | **Từ ngày** | **Đến ngày** |
| **1** | Phân tích đề tài | 01/09/2021 | 28/09/2021 |
| **2** | Thiết kế CSDL | 01/09/2021 | 28/09/2021 |
| **3** | Lập trình Back end | 28/09/2021 | 17/10/2021 |
| **4** | Lập trình Front end | 17/10/2021 | 27/11/2021 |
| **5** | Thiết kế các báo cáo | 28/11/2021 | 01/12/2021 |
| **6** | Chạy thử và sửa lỗi | 01/12/2021 | 12/12/2021 |
| **7** | Viết báo cáo và sửa lỗi đề tài | 12/12/2021 | 09/01/2022 |

**Xác nhận của người hướng dẫn** **Tiền Giang, ngày 12 tháng 01 năm 2022**

Nhóm sinh viên thực hiện

Đặng Hữu Lộc

Nguyễn Thái Duy Nguyễn Ngọc Minh Yến

NHẬN XÉT CỦA CÁN BỘ HƯỚNG DẪN

NHẬN XÉT CỦA HỘI ĐỒNG ĐÁNH GIÁ

MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI 14](#_Toc92746084)

[1.1. Lý do chọn đề tài 14](#_Toc92746085)

[1.2. Mục tiêu đề tài 14](#_Toc92746086)

[1.3. Nhiệm vụ chính của đề tài 14](#_Toc92746087)

[1.4. Đối tượng và phạm vi nghiên cứu 15](#_Toc92746088)

[1.5. Yêu cầu cần đạt 15](#_Toc92746089)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ THỰC TIỄN ĐỀ TÀI 17](#_Toc92746090)

[2.1. Cơ sở lý thuyết 17](#_Toc92746091)

[**2.1.1.** Kiến trúc Microservices 17](#_Toc92746092)

[**2.1.2.** ReactJS. 20](#_Toc92746093)

[**2.1.3.** SpringBoot. 21](#_Toc92746094)

[**2.1.4.** Spring WebFlux. 22](#_Toc92746095)

[**2.1.5.** Spring Data Reactive Relational Database Connectivity (R2DBC). 22](#_Toc92746096)

[**2.1.6.** Spring Data Java Persistence API. 22](#_Toc92746097)

[**2.1.7.** OAuth2 và Keycloak. 23](#_Toc92746098)

[**2.1.8.** Kubernates. 24](#_Toc92746099)

[2.2. Cơ sở thực tiễn: 27](#_Toc92746100)

[CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ CÀI ĐẶT GIẢI PHÁP 29](#_Toc92746101)

[3.1. Mô tả công việc nghiên cứu 29](#_Toc92746102)

[**3.1.1.** Kiến trúc hệ thống 29](#_Toc92746103)

[**3.1.2.** Sơ đồ chức năng 30](#_Toc92746104)

[**3.1.3** Sơ đồ UseCase 32](#_Toc92746105)

[**3.1.4** Mô hình thực thể kết hợp. 39](#_Toc92746106)

[**3.1.5** Sơ đồ lớp của hệ thống. 42](#_Toc92746107)

[**3.1.6** Các sơ đồ trình tự. 47](#_Toc92746108)

[3.2 Quy trình thực hiện. 52](#_Toc92746109)

[3.3 Một số hình ảnh giao diện hệ thống 53](#_Toc92746110)

[CHƯƠNG 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN 59](#_Toc92746111)

[4.1 Kết quả đạt được 59](#_Toc92746112)

[4.2 Khả năng ứng dụng của hệ thống 59](#_Toc92746113)

[4.3 Hạn chế 60](#_Toc92746114)

[4.4 Hướng phát triển 60](#_Toc92746115)

[TÀI LIỆU THAM KHẢO 61](#_Toc92746116)

DANH MỤC HÌNH

[Hình 1 Mô hình kiến trúc Microservices 17](#_Toc92746215)

[Hình 2 Chi tiết mô hình Microservices 18](#_Toc92746216)

[Hình 3 Logo ReactJS 20](#_Toc92746217)

[Hình 4 Mô hình hoạt động OAuth2 23](#_Toc92746218)

[Hình 5 Mô hình kiến trúc Kubernetes. 25](#_Toc92746219)

[Hình 6 Ví dụ hệ thống chưa sử dụng Kafka 26](#_Toc92746220)

[Hình 7 Mô hình ví dụ nhiều server có sử dụng Kafka 27](#_Toc92746221)

[Hình 8 Sơ đồ mô tả kiến trúc của hệ thống Study-Space 29](#_Toc92746222)

[Hình 9 Sơ đồ chức năng StudySpace 30](#_Toc92746223)

[Hình 10 Sơ đồ Usecase quản trị viên nhóm 32](#_Toc92746224)

[Hình 11 Sơ đồ Usecase người dùng 33](#_Toc92746225)

[Hình 12 Mô hình thực thể kết hợp của Quản lý đề 39](#_Toc92746226)

[Hình 13 Mô hình thực thể kết hợp của Quản lý bài làm 40](#_Toc92746227)

[Hình 14 Mô hình thực thể kết hợp của quản lý nhóm 41](#_Toc92746228)

[Hình 15 Sơ đồ lớp của Microservice ExamStore 42](#_Toc92746229)

[Hình 16 Sơ đồ lớp của Microservice AnswerStore 44](#_Toc92746230)

[Hình 17 Sơ đồ lớp của Microservice GroupStore 46](#_Toc92746231)

[Hình 18 Sơ đồ trình tự đăng nhập 47](#_Toc92746232)

[Hình 19 Sơ đồ trình tự tạo kho câu hỏi 48](#_Toc92746233)

[Hình 20 Sơ đồ trình tự tạo đề kiểm tra 49](#_Toc92746234)

[Hình 21 Trình tự lấy câu hỏi trong đề kiểm tra 50](#_Toc92746235)

[Hình 22 Sơ đồ trình tự làm bài 51](#_Toc92746236)

[Hình 23 Tạo kho lưu trữ github để quản lý source code 52](#_Toc92746237)

[Hình 24 Tạo project trên github để quản lý danh sách công việc cần làm. 52](#_Toc92746238)

[Hình 25 Sử dụng JDL studio 53](#_Toc92746239)

[Hình 26 Giao diện đăng nhập 53](#_Toc92746240)

[Hình 27 Giao diện quản lý kho câu hỏi 54](#_Toc92746241)

[Hình 28 Giao diện tạo mới kho câu hỏi 54](#_Toc92746242)

[Hình 29 Giao diện tạo câu hỏi cho kho 55](#_Toc92746243)

[Hình 30 Giao diện quản lý đề trắc nghiệm 56](#_Toc92746244)

[Hình 31 Giao diện tạo đề trắc nghiệm 56](#_Toc92746245)

[Hình 32 Giao diện tạo chi tiết đề trắc nghiệm 57](#_Toc92746246)

[Hình 33 Giao diện xem chi tiết đề trắc nghiệm 57](#_Toc92746247)

[Hình 34 Giao diện làm bài trắc nghiệm 58](#_Toc92746248)

DANH MỤC BẢNG

[Bảng 1 Kịch bản đăng ký tài khoản 33](#_Toc92746249)

[Bảng 2 Kịch bản đăng nhập 34](#_Toc92746250)

[Bảng 3 Kịch bản tạo kho câu hỏi 34](#_Toc92746251)

[Bảng 4 Kịch bản tạo đề trắc nghiệm 35](#_Toc92746252)

[Bảng 5 Kịch bản làm bài 36](#_Toc92746253)

[Bảng 6 Kịch bản thống kê kết quả làm bài 36](#_Toc92746254)

[Bảng 7 Kịch bản tạo nhóm 37](#_Toc92746255)

[Bảng 8 Kịch bản duyệt xin vào nhóm 37](#_Toc92746256)

[Bảng 9 Kịch bản quản lý thành viên nhóm 38](#_Toc92746257)

[Bảng 10 Mô tả các thực thể trong mô hình ExamStore 39](#_Toc92746258)

[Bảng 11 Mô tả các thực thể trong mô hình AnswerStore. 40](#_Toc92746259)

[Bảng 12 Mô tả các thực thể trong mô hình GroupStore 41](#_Toc92746260)

[Bảng 13 Thông tin về các trường của bảng DeTai 42](#_Toc92746261)

[Bảng 14 Thông tin về các trường của bảng DanhMucCauHoi 42](#_Toc92746262)

[Bảng 15 Thông tin về các trường của CauHoi 43](#_Toc92746263)

[Bảng 16 Thông tin về các trường của PhuongAn 43](#_Toc92746264)

[Bảng 17 Thông tin về các trường của DeBai 43](#_Toc92746265)

[Bảng 18 Thông tin về các trường của T\_DeBai\_DanhMuc 44](#_Toc92746266)

[Bảng 19 Thông tin về các trường của BaiLam 45](#_Toc92746267)

[Bảng 20 Thông tin các trường của ChiTietBaiLam 45](#_Toc92746268)

[Bảng 21 Thông tin về các trường của LichLamBai 45](#_Toc92746269)

[Bảng 22 Thông tin về các trường của ThoiGianBieu 46](#_Toc92746270)

[Bảng 23 Thông tin về các trường của Nhom 46](#_Toc92746271)

[Bảng 24 Thông tin về các trường của T\_User\_Nhom 47](#_Toc92746272)

DANH MỤC THUẬT NGỮ TIẾNG ANH

|  |  |  |
| --- | --- | --- |
| **STT** | **Thuật ngữ** | **Mô tả** |
| 1 |  | ĐÁng yeu |
| 2 |  |  |
| 3 |  |  |

# CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

## Lý do chọn đề tài

Việc tự học luôn mang lại hiệu quả cao cho quá trình trau dồi, rèn luyện và ôn tập kiến thức. Bên cạnh đó người dùng Internet ngày càng nhiều, việc sử dụng các ứng dụng hỗ trợ trong học tập cũng được mọi người quan tâm hơn, giúp người dùng dễ dàng tiếp cận, tổng hợp kiến thức và tra cứu thông tin một cách nhanh chóng.

Hiện nay, xu hướng học tập online trên các ứng dụng ngày càng phổ biến đặc biệt là trong tình hình dịch bệnh Covid 19, việc giảng dạy, học tập tại lớp trở nên khó khăn hơn. Chính vì thế, đây là thời điểm thích hợp để tập trung vào việc phát triển năng lực tự học của học sinh, sinh viên.

Việc xây dựng hệ thống quản lý và tự học trắc nghiệm mang lại nhiều lợi ích như: dễ dàng kiểm soát và bảo quản các tài liệu trắc nghiệm, giúp người dùng ôn tập bằng cách tạo đề kiểm tra ngẫu nhiên từ kho câu hỏi của người dùng, chia sẻ tài liệu trắc nghiệm với nhau, tiết kiệm tài nguyên thiên nhiên.

Và đây là một đề tài mới, và rất cần thiết cho học sinh, sinh viên.

## Mục tiêu đề tài

Đề tài được thực hiện nhằm xây dựng thành công ứng dụng hoạt động tốt, giao diện thân thiện, tích hợp đầy đủ tính năng phù hợp nhu cầu người dùng, mọi đối tượng sử dụng.

## Nhiệm vụ chính của đề tài

Nghiên cứu thực tế: Nghiên cứu mô hình Microservices, thiết kế hệ thống thông tin, kết hợp sử dụng React, Spring Boot, Spring Data, hỗ trợ triển khai hệ thống bằng Kubernetes và hệ quản trị cơ sở dữ liệu MySQL.

Xác định phạm vi đề tài: xây dựng ứng dụng đẩy đủ các tính năng hỗ trợ quản lý và hỗ trợ tự học trắc nghiệm cho người học trong việc quản lý tài liệu và ôn luyện trắc nghiệm.

* Nghiên cứu tài liệu chuyên môn:
  + Tổng quan về Microservices.
  + Cách thức hoạt động Kubernates.
  + Kiến thức cơ bản về React, Spring Boot.
  + Giới thiệu về Spring Data Reactive Relational Database Connectivity.

## Đối tượng và phạm vi nghiên cứu

* Đối tượng nghiên cứu:
  + Về nghiệp vụ: Nghiên cứu tài liệu, cấu trúc các đề kiểm tra, dạng kiểm tra của các cấp học.
  + Về công nghệ:
    - Tìm hiểu kiến trúc Microservices và cách xây dựng ứng dụng Microservices với Jhipster.
    - Tìm hiểu React và các thư viện JavaScript khác có liên quan để xây dựng Front-end cho dự án.
    - Tìm hiểu Spring Boot và các kỹ thuật có liên quan như ORM và lập trình bất đồng bộ trong Spring Application để tạo Back-end cho dự án.
  + Về kỹ thuật:
    - Phân tích yêu cầu, phân tích hệ thống.
    - Xây dựng một hệ thống hoàn chỉnh từ cả 2 khía cạnh client và server.
    - Sử dụng các công cụ hỗ trợ lập trình.
* Phạm vi nghiên cứu:

Nghiên cứu các ứng dụng học trực tuyến, các ứng dụng ôn luyện, tham khảo các thức xây dựng đề kiểm tra. Từ đó, đúc kết lại những thiếu sót để xây dựng hệ thống hoàn chỉnh, đáp ứng đầy đủ các tính năng, giao diện dễ sử dụng phù hợp mọi đối tượng sử dụng.

## Yêu cầu cần đạt

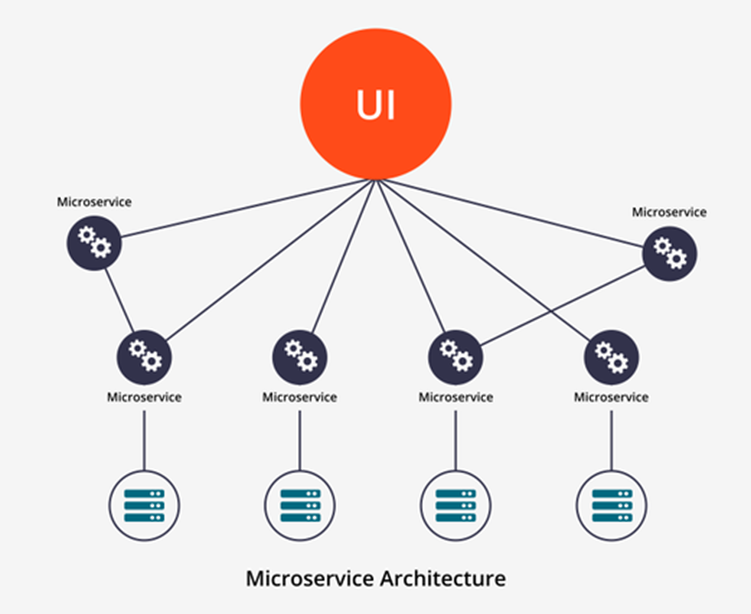
* Hệ thống xây dựng thành công bằng mô hình Microservices, đảm bảo tính phân quyền và bảo mật, hoạt động ổn định.
* Giao diện hệ thống đơn giản, dễ sử dụng cho mọi đối tượng sử dụng, bố cục hợp lí, các menu chức năng rõ ràng.
* Ứng dụng chạy tốt trên mọi trình duyệt Web (Google Chrome, Microsoft Edge, Firefox, …).

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ THỰC TIỄN ĐỀ TÀI

## Cơ sở lý thuyết

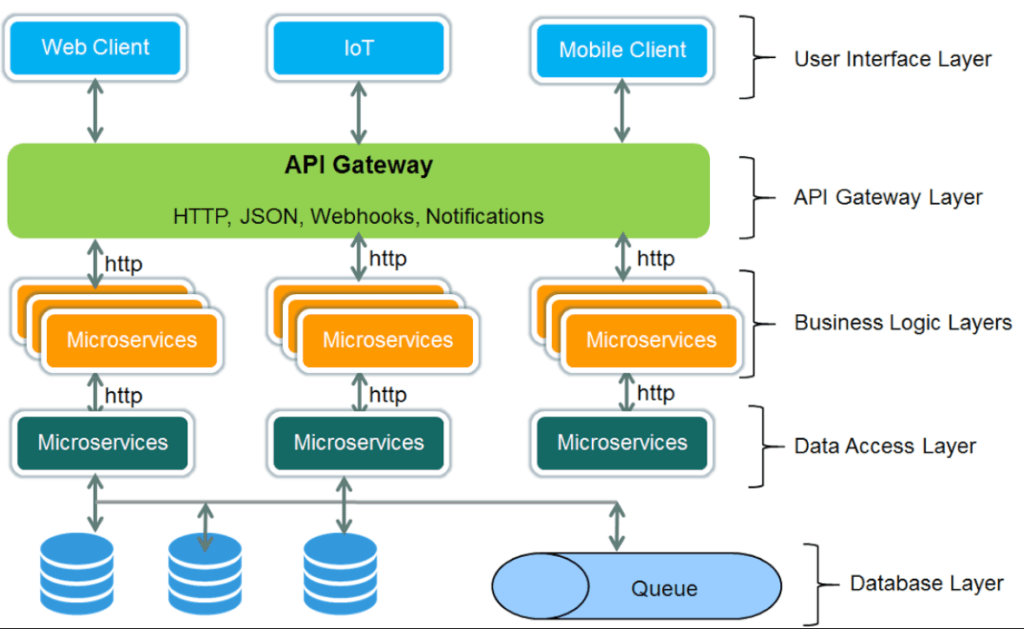
### Kiến trúc Microservices

Khái niệm: Microservices là một kiểu kiến trúc phần mềm mà các nhóm chức năng có liên quan với nhau sẽ được gộp lại với nhau vào giao cho một ứng dụng chuyên trách xử lý gọi là Microservice. Mỗi Microservice sẽ được đặt trên một server riêng để dễ dàng bảo trì và nâng cấp.



Hình Mô hình kiến trúc Microservices

Kiến trúc: Thay vì lập trình tất cả các chức năng trong cùng một ứng dụng, ta gộp các chức năng có liên quan với nhau thành nhóm các chức năng và giao cho từng Microservices tương ứng. Mỗi service có thể được đặt trên một server riêng và thực hiện giao tiếp với nhau thông qua tin nhắn.



Hình Chi tiết mô hình Microservices

Đặc điểm của Microservices:

* Decoupling : Các service trong một hệ thống phần lớn được tách rời. Vì vậy, toàn bộ ứng dụng có thể dễ dàng được xây dựng, thay đổi và thu nhỏ.
* Componentization : Microservices được coi là các thành phần độc lập có thể dễ dàng thay thế và nâng cấp.
* Business Capabilities : mỗi một thành phần trong kiến trúc Microservices rất đơn giản và tập trung vào một nhiệm vụ duy nhất.
* Autonomy : các lập trình viên hay các nhóm có thể làm việc độc lập với nhau trong quá trình phát triển.
* Continous Delivery : Cho phép phát hành phần mềm thường xuyên, liên tục.
* Single Responsibility Priciple (SRP) : Mỗi service cần có phạm vi và bị giới hạn về nghiệp vụ cụ thể, không phụ thuộc lẫn nhau. Điều đó giúp cho chúng ta đáp ứng sự phát triển linh hoạt và nhanh chóng trong cung cấp service.
* Decentralized Governance : không có mẫu chuẩn hóa hoặc bất kỳ mẫu công nghệ nào. Được tự do lựa chọn các công cụ hữu ích tốt nhất để có thể giải quyết vấn đề.

Tính chất:

* Tốc độ phát triển nhanh.
* Tính phân tán cao.
* Không cố định quy mô dịch vụ, hệ thống.
* Phù hợp với dạng hệ thống thường xuyên thay đổi, cập nhật.
* Phối hợp nhiều công nghệ trong cùng một ứng dụng.

Kiến trúc Microservices được sinh ra để khắc phục những hạn chế của kiến trúc nguyên khối. Dựa trên việc chia nhỏ hệ thống thành nhiều service riêng biệt, kiến trúc Microservice có ưu điểm như sau:

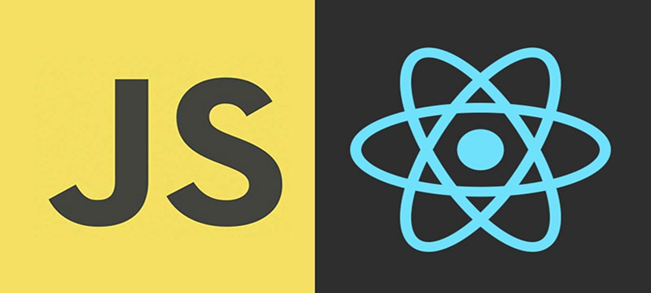
* Các service có thể được bảo trì độc lập
* Dễ dàng nắm bắt các business logic.
* Áp dụng được nhiều công nghệ.
* Dễ dàng mở rộng khi hệ thống trở nên phức tạp
* Giảm thiểu được rủi ro.
* Dễ dàng thay đổi để sử dụng các công nghệ mới.

Hạn chế:

* Microservices khuyến khích làm nhỏ gọn các dịch vụ, nhưng khi chia nhỏ sẽ dẫn đến những thứ vụn vặt, khó kiểm soát. Hơn nữa chính từ đặc tính phân tán khiến cho các lập trình viên phải lựa chọn cách thức giao tiếp phù hợp khi xử lí yêu cầu giữa các dịch vụ.
* Phải đảm bảo giao dịch phân tán (distributed transaction) .
* Testing một dịch vụ trong kiến trúc Microservices đôi khi yêu cầu phải chạy cả các dịch vụ nhỏ khác mà nó phụ thuộc
* Việc triển khai Microservices phức tạp hơn rất nhiều nếu làm thủ công theo cách đã làm với ứng dụng monolithic.

### ReactJS.

Khái niệm: React là một thư viện JavaScript được phát triển bởi Facebook được dùng để phát triển giao diện người dùng. React giúp phân chia các UI phức tạp thành các thành phần nhỏ có thể tái sử dụng.



Hình Logo ReactJS

React mang lại nhiều tính năng đặc biệt:

* Declarative: React giúp cho việc phát triển giao diện (có thể tương tác được) một cách dễ dàng. Chỉ cần thiết kế các khung hình, React sẽ cập nhật và chỉ hiển thị các thành phần phù hợp khi dữ liệu thay đổi. Declarative view giúp code dễ đoán và dễ gỡ lỗi hơn.
* Component-Based: Xây dựng các thành phần được đóng gói với trạng thái riêng của chúng. Sau đó, biên soạn chúng thành những giao diện người dùng phức tạp. Giúp tái sử dụng các Component đã viết trước đó và sử dụng cho những giao diện khác hoặc dự án khác.
* Học một lần, viết mọi nơi: React có thể kết xuất giao diện ngay trên server với Node và giúp lập trình viên mobile app dễ tiếp cận với React Native hơn.

Tuy nhiên, React cũng có một số nhược điểm:

* React chỉ là một thư viện giúp build giao diện, không có phần Model hay Controller như những MVC framework nên cần phải kết hợp với nhiều thư viện khác làm cho việc nâng cấp khó khăn hơn.
* Với những ứng dụng lớn cần phải cầu hình với TypeScript để đảm bảo type-safe và dễ hiểu.
* Khó tiếp cận cho người mới học web.

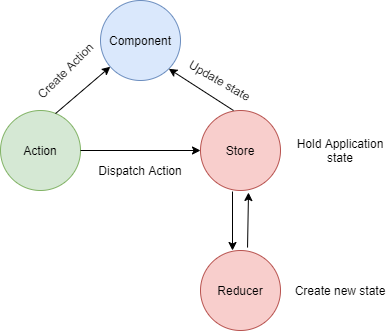
### Redux.

Giới thiệu về Redux:

* Redux là một công cụ quản lý cho các ứng dụng Javascript, có thể gọi Redux là một kho chứa.
* Mặc dù chủ yếu được sử dụng với React, nhưng nó có thể được sử dụng với bất kỳ thư viện hoặc Framework JavaScript nào khác.
* Với Redux, States của ứng dụng được giữ trong một nơi gọi là Store và mỗi Component đều có thể truy cập bất kỳ States nào mà chúng muốn từ Store này.

Cách hoạt động của Redux:

* Redux có một Store lưu trữ toàn bộ State của các Component. Mỗi Component có thể truy cập trực tiếp đến State được lưu trữ thay vì phải gửi Props từ Component này đến Component khác. Có 3 thành phần của Redux: Actions, Store, Reducers.
* Action: Actions đơn giản là các sự kiện. Chúng là cách mà chúng ta gửi dữ liệu từ ứng dụng đến Store. Những dữ liệu này có thể là từ sự tương tác của người dùng với ứng dụng, gọi từ API hoặc cũng có thể là từ gửi dữ liệu của bảng.
* Reducers: Reducers là các phương thức nguyên thủy (Pure Funtioncs) chúng lấy State hiện tại của ứng dụng, thực hiện một Action và trả về một state mới. Những State này được lưu như những đối tượng và chúng định rõ cách State của một ứng dụng thay đổi trong việc phản hồi một Action được gửi đến Store.
* Store: Store lưu trạng thái ứng dụng và nó là duy nhất trong bất kỳ một ứng dụng Redux nào. Người lập trình có thể truy các State được lưu, cập nhật State.



Hình Mô hình hoạt động Redux

### SpringBoot.

Spring Boot là một module của Spring Framework, cung cấp tính năng RAD (Rapid Application Development) – Phát triển ứng dụng nhanh, được dùng để tạo các ứng dụng độc lập dựa trên Spring không yêu cầu cấu hình XML và là một chuẩn cho cấu hình thiết kế phần mềm, tăng cao năng suất cho developer.

Ưu điểm:

* Tạo ứng dụng độc lập, có thể chạy bằng java -jar (cho cả java web)
* Nhúng trực tiếp các ứng dụng server (Tomcat, Jetty…) do đó không cần phải triển khai file WAR
* Cấu hình ít, tự động cậu hình bất kì khi nào có thể (Giảm thời gian viết code, tăng năng suất)
* Không yêu cầu XML config…
* Cung cấp nhiều plugin
* Chuẩn cho Microservices (Cloud support; giảm việc setup, config; các thư viện hỗ trợ…)
* Giảm thiểu thời gian phát triển code, tăng hiệu suất phát triển chung của cả dự án.
* Dễ dàng tích hợp các mô-đun liên quan như Sping-MVC, Spring Data, Spring Sercurity, Spring Cloud, …

Tuy nhiên, Spring Boot cũng có một số hạn chế như:

* Thiếu kiểm soát. Do style cố định, Spring Boot tạo ra nhiều phụ thuộc không được sử dụng dẫn đến kích thước tệp triển khai lớn.
* Quá trình chuyển đổi dự án Spring cũ hoặc hiện có thành các ứng dụng Spring Boot nhiều khó khăn và tốn thời gian.

### Spring WebFlux.

Spring Webflux cũng là một Framework cho việc phát triển các ứng dụng Web cung cấp Reactive Programming nhằm hổ trợ cho việc xây dựng ứng dụng web.

Là một phiên bản song song với Spring MVC và hỗ trợ non-blocking reactive streams. Nó hỗ trợ khái niệm back pressure và sử dụng Server Netty để run hệ thống reactive.

Reactive và non-blocking nhìn chung thì không làm cho ứng dụng chạy nhanh hơn. Lợi ích mà nó được kỳ vọng là mở rộng ứng dụng với số luồng nhỏ và yêu cầu ít bộ nhớ hơn. Nó làm cho các ứng dụng trở nên linh hoạt hơn khi tải.

### Spring Data Reactive Relational Database Connectivity (R2DBC).

Spring Data R2DBC là một phần trong Spring Data, là một spec định nghĩa cách thức sẽ làm việc với các relational database như MySQL, PostgreSQL, … theo cơ chế bất đồng bộ. Nó cung cấp bộ thư viện Service Provider Interface (SPI) giúp có thể implement driver cho từng loại database system tương ứng.

R2DBC là một giải pháp thay thế cho JDBC. Nó Reactive, tức là Non-Blocking. Client gửi một tuyên bố đến server, tiếp tục làm việc và cuối cùng nhận được kết quả dưới dạng một sự kiện. Nó thường được sử dụng để có khả năng mở rộng quy mô tốt hơn. Nhưng nó khó sử dụng hơn vì nó bất đồng bộ và về cơ bản ứng dụng phải xử lý 1 lượng request đủ lớn để thấy được những lợi ích của nó.

### Spring Data Java Persistence API.

Spring Data JPA là một đặc tả Java cho việc ánh xạ giữa các đối tượng Java với cơ sở dữ liệu quan hệ sử dụng công nghệ phổ biến là ORM (Object Relational Mapping).

JPA độc lập về Database giúp để thay đổi loại DB phù hợp, lập trình viên không cần phải làm việc với SQL, viết code ít hơn, hiệu năng tốt hơn và có thể tập trung vào logic nghiệp vụ.

Việc sử dụng Spring Data JPA mang lại nhiều lợi ích:

* Đơn giản hóa công nghệ cho tầng persistence (tầng dữ liệu).
* Không phụ thuộc vào các framework ORM.
* Có nhiều nhà cung cấp hỗ trợ cài đặt JPA.
* Dữ liệu có thể được lưu trữ thông qua việc ORM.

### OAuth2 và Keycloak.

OAuth2 là một giao thức chứng thực cho phép các ứng dụng chia sẻ một phần tài nguyên với nhau mà không cần xác thực qua username và password.

OAuth tượng trưng cho:

* Authentication: xác thực người dùng.
* Authorization: cấp quyền truy cập đến tài nguyên mà người dùng hiện đang nắm giữ.



Hình Mô hình hoạt động OAuth2

Ưu điểm:

* Hạn chế được những phiền toái khi phải nhập username, password ở quá nhiều nơi hoặc đăng ký quá nhiều tài khoản mật khẩu.
* Lưu token cho việc truy cập của người dùng một cách cực kỳ nhanh chóng và dễ dàng.
* Khả năng chia sẻ dữ liệu của người dùng được nâng cao lên.

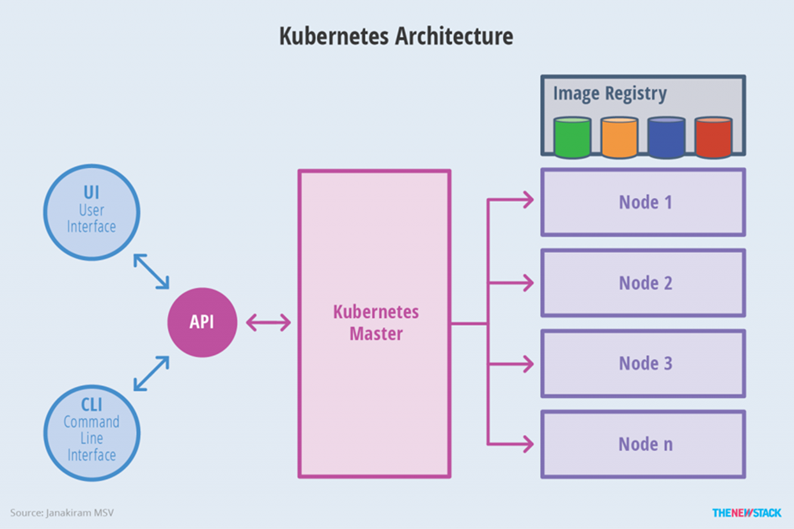
Keycloak là một phần mềm nguồn mở cho phép đăng nhập một lần (IdP) với Quản lý danh tính và Quản lý truy cập. Phần mềm hỗ trợ các giao thức liên kết danh tính theo mặc định. Tạo điều kiện thuận lợi cho việc bảo vệ các ứng dụng và dịch vụ với ít hoặc không có mã hóa.

Ưu điểm:

* Hỗ trợ nhiều giao thức.
* Cho phép xác thực đăng nhập một lần (SSO).
* Có thể thống nhất các phương pháp xác thực và làm cho chúng phát triển mà không cần sửa đổi các ứng dụng.
* Ứng dụng bảo mật tài khoản và dịch vụ đơn giản.

### Kubernates.

Kubernetes là một nền tảng nguồn mở, khả chuyển, có thể mở rộng để quản lý các ứng dụng được đóng gói và các service, giúp thuận lợi trong việc cấu hình và tự động hoá việc triển khai ứng dụng. Kubernetes là một hệ sinh thái lớn và phát triển nhanh chóng. Các dịch vụ, sự hỗ trợ và công cụ có sẵn rộng rãi.



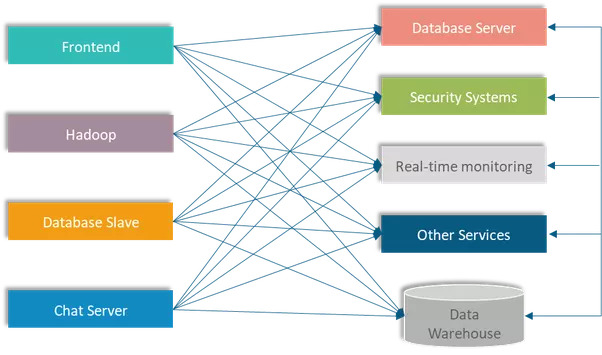
Hình Mô hình kiến trúc Kubernetes.

Kubernetes cung cấp:

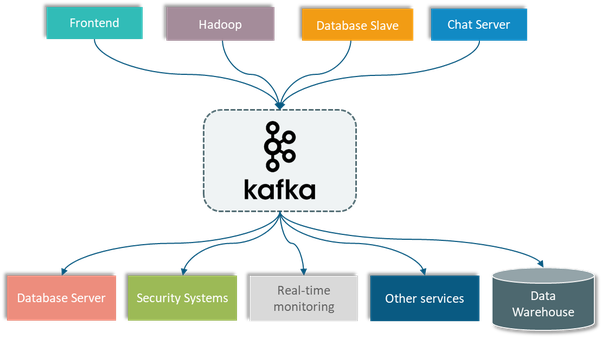
* Service discovery và cân bằng tải: Nếu lượng traffic truy cập đến một container cao, Kubernetes có thể cân bằng tải và phân phối lưu lượng mạng (network traffic) để việc triển khai được ổn định.
* Điều phối bộ nhớ: Kubernetes cho phép tự động mount một hệ thống lưu trữ mà bạn chọn, như local storages, public cloud providers, v.v.
* Tự động rollouts và rollbacks: Có thể tự động hoá Kubernetes để tạo mới các container cho việc triển khai, xoá các container hiện có và áp dụng tất cả các resource của chúng vào container mới.
* Đóng gói tự động: Kubernetes có thể điều phối các container đến các node để tận dụng tốt nhất các resource.
* Tự phục hồi: Kubernetes khởi động lại các containers bị lỗi, thay thế các container, xoá các container không phản hồi lại cấu hình health check do người dùng xác định và không cho các client biết đến chúng cho đến khi chúng sẵn sàng hoạt động.
* Quản lý cấu hình và bảo mật: Kubernetes cho phép lưu trữ và quản lý các thông tin nhạy cảm như: password, OAuth token và SSH key.
  + 1. Kafka.

Là hệ thống message publisher/subcriber phân tán (distributed messaging system).

Với một hệ thống lớn, data pipeline sẽ rất phức tạp. Nếu sử dụng Kafka tách rời các data pipeline giữa các hệ thống để làm cho việc giao tiếp giữa các hệ thống trở nên đơn giản hơn và dễ quản lý hơn.



Hình Ví dụ hệ thống chưa sử dụng Kafka



Hình Mô hình ví dụ nhiều server có sử dụng Kafka

Kafka có khả năng chịu lỗi cao, hiệu năng rất tốt và dễ dàng mở rộng mà không cần dừng hệ thống.

Kafka có khả năng truyền một lượng lớn message theo thời gian thực, trong trường hợp bên nhận chưa nhận message vẫn được lưu trữ sao lưu trên một hàng đợi và cả trên ổ đĩa bảo đảm an toàn. Đồng thời nó cũng được replicate trong cluster giúp phòng tránh mất dữ liệu.

## Cơ sở thực tiễn:

Mục tiêu, yêu cầu của hệ thống: Hiện nay với tình hình dịch bệnh diễn biến phức tạp cùng với xu hướng tự học bằng các ứng dụng ngày càng phổ biến. Thế nên, đề tài “Xây dựng ứng dựng ôn luyện trắc nghiệm Study-Space” được xây dựng và triển khai nhằm tạo ra ứng dụng tốt, phù hợp nhu cầu người dùng.

Về mặt tổng quan, hệ thống hướng tới mục tiêu:

* Giao diện trực quan, dễ sử dụng, phù hợp mọi đối tượng.
* Đáp ứng đầy đủ chức năng.
* Đảm bảo tốt khả năng bảo mật và phân quyền.
* Dễ dàng bảo trì, nâng cấp, mở rộng hệ thống.
* Đáp ứng nhu cầu hiệu năng, hoạt động hiệu quả, nhanh chóng.

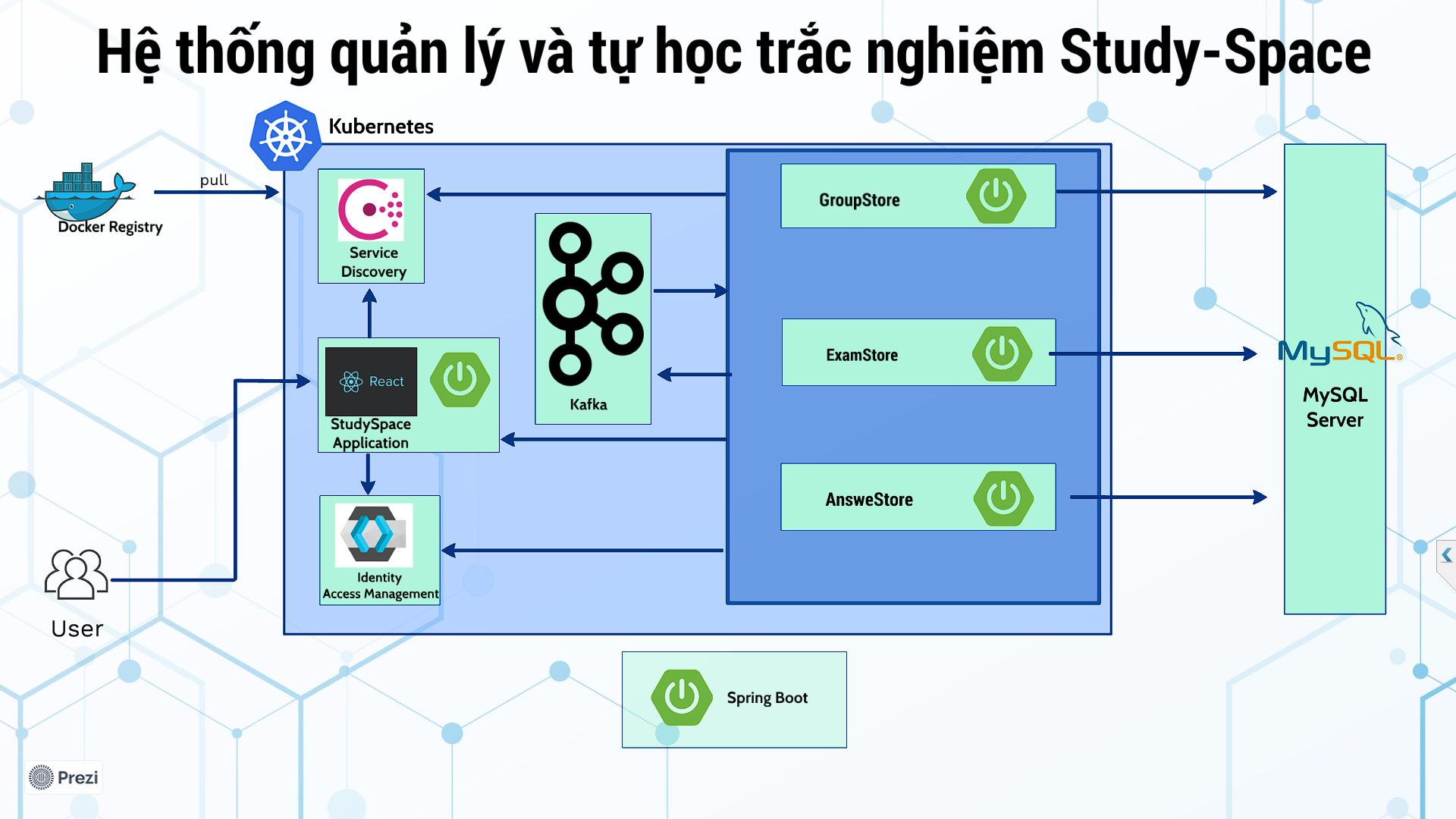
Về phạm vi chức năng, hệ thống xây dựng các chức năng như sau:

* Đăng nhập hệ thống.
* Quản lý tài khoản.
* Phân quyền cho tài khoản người dùng.
* Quản lý kho câu hỏi trắc nghiệm.
* Quản lý đề bài.
* Quản lý bài làm, tính điểm, thống kê kết quả.
* Quản lý nhóm học theo các môn, hạn chế quyền truy cập tài nguyên theo nhóm.
* Quản lý nhóm và thành viên nhóm.

# CHƯƠNG 3: PHÂN TÍCH, THIẾT KẾ CÀI ĐẶT GIẢI PHÁP

## 3.1. Mô tả công việc nghiên cứu

### Kiến trúc hệ thống

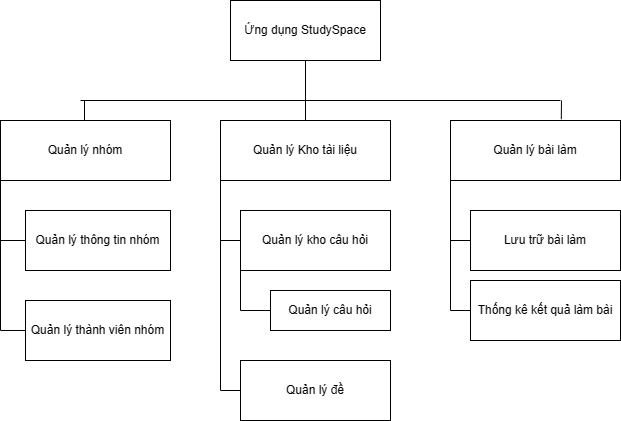


Hình Sơ đồ mô tả kiến trúc của hệ thống Study-Space

Các thành phần của hệ thống gồm:

1. StudySpace Application - Gateway: Chịu trách nhiệm giao tiếp với người dùng, đảm bảo người dùng có thể sử dụng các chức năng hệ thống cung cấp mà không cần biết các xử lý bên dưới của hệ thống.
2. ExamStore: Chịu trách nhiệm xử lý các chức năng: quản lý kho câu hỏi, quản lý cấu hình của đề kiểm tra, hạn chế quyền truy cập của người dùng với tài nguyên.
3. AnswerStore: Chịu trách nhiệm xử lý các chức năng: quản lý bài làm của người dùng, thống kê kết quả làm bài.
4. GroupStore: Chịu trách nhiệm xử lý các chức năng: quản lý thông tin nhóm, quản lý thành viên nhóm, hạn chế quyền truy cập các tài nguyên giữa các nhóm.
5. Consul – service discovery: Liên kết các ứng dụng trong hệ thống.
6. Keycloak – Identity access manager: Quản lý định danh người dùng.
7. Kafka – message broker: Hỗ trợ trao đổi dữ liệu giữa các ứng dụng trong hệ thống.

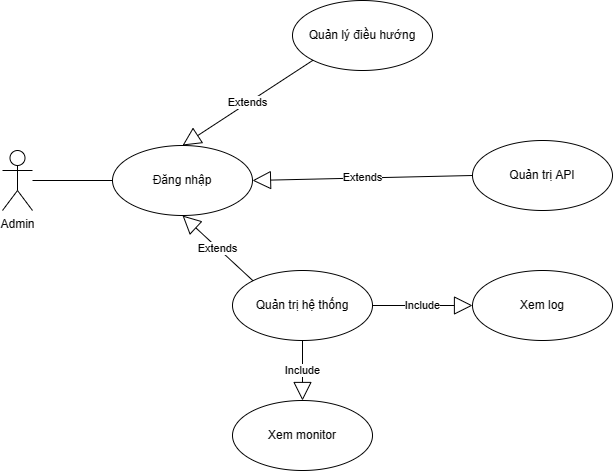
### Sơ đồ chức năng



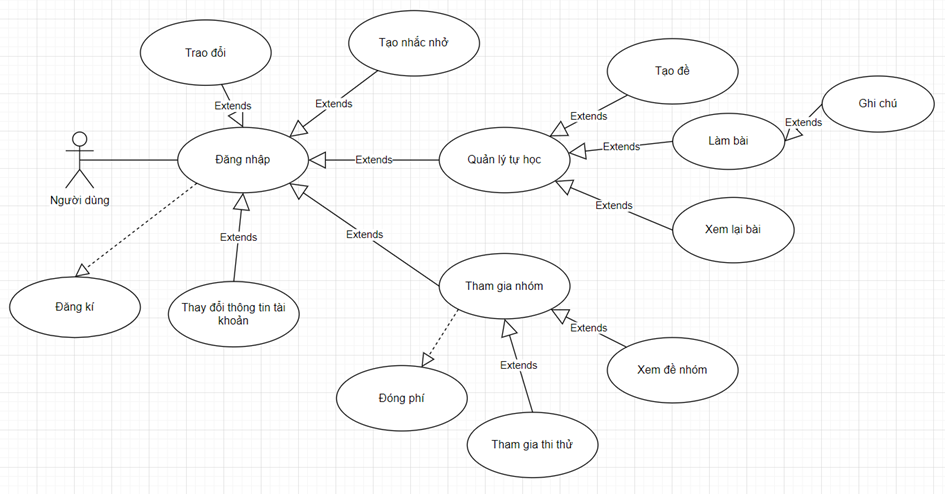
Hình Sơ đồ chức năng StudySpace

* Quản lý nhóm đảm nhận chức năng như sau:
* Tạo nhóm.
* Phân quyền admin cho trưởng nhóm.
* Thêm, xóa thành viên nhóm.
* Cập nhật thông tin nhóm.
* Lấy danh sách thành viên của nhóm.
* Tìm kiếm thành viên có trong nhóm.
* Xóa nhóm.
* Quản lý đề có chức năng như:
* Tạo danh mục câu hỏi.
* Tạo câu hỏi.
* Tạo đề tài mới.
* Tạo đề kiểm tra.
* Thêm, sửa, xóa thông tin câu hỏi.
* Thêm, xóa, sửa đáp án câu hỏi.
* Thêm, sửa, xóa bài kiểm tra.
* Kiểm tra kết quả làm bài.
* Quản lý bài làm gồm các chức năng như:
* Thêm, xóa, sửa lịch làm bài cho nhóm.
* Thêm, xóa, sửa nhắc nhở cho user.
* Tạo bài làm.
* Tạo 1 lựa chọn trong bài làm.
* Xem chi tiết bài làm.
* Xem tất cả bài làm trong lịch làm nhóm.

### Sơ đồ UseCase



Hình Sơ đồ Usecase quản trị viên nhóm



Hình Sơ đồ Usecase người dùng

* Mô tả Usecase

Bảng Kịch bản đăng ký tài khoản

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống |
| Điều kiện | Không có |
| Các bước thực hiện | 1. Chọn nút “Đăng kí” ở trang chủ. 2. Hệ thống hiển thị giao diện đăng kí. 3. Người dùng nhập thông tin. 4. Chọn “Đăng kí”. 5. Hệ thống kiểm tra thông tin 6. Hệ thống thông báo tạo tài khoản thành công. |
| Ngoại lệ | Hệ thống thông báo lỗi. |

Bảng Kịch bản đăng nhập

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống |
| Điều kiện | Đã tạo tài khoản thành công |
| Các bước thực hiện | 1. Vào giao diện đăng nhập của hệ thống. 2. Nhập tên tài khoản và mật khẩu. 3. Hệ thống kiểm tra tên giao diện và mật khẩu 4. Hệ thống thông báo đăng nhập thành công. 5. Hiển thị giao diện trang chủ. |
| Ngoại lệ | Hệ thống thông báo đăng nhập không thành công. |

Bảng Kịch bản tạo kho câu hỏi

|  |  |
| --- | --- |
| Actor | Tất cả các actor trong hệ thống |
| Điều kiện | Đã đăng nhập |
| Các bước thực hiện | 1. Chọn “Quản lý kho câu hỏi”. 2. Hệ thống hiển thị giao diện danh sách kho câu hỏi. 3. Người dùng chọn nút “Thêm kho câu hỏi mới”. 4. Nhập tên và các thông tin của kho câu hỏi. 5. Nhấn nút lưu. 6. Hệ thống thông báo tạo kho câu hỏi thành công. 7. Hiển thị giao diện thêm câu hỏi vào kho. 8. Người dùng nhấn nút “thêm câu hỏi mới”, nhập nội dung câu hỏi, câu trả lời vào form nhập. 9. Người dùng nhấn nút lưu. 10. Hệ thống thông báo lưu câu hỏi thành công. 11. Người dùng nhấn quay lại. |
| Ngoại lệ | Tạo kho câu hỏi không thành công.  Tạo câu hỏi không thành công. |

Bảng Kịch bản tạo đề trắc nghiệm

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống. |
| Điều kiện | Người dùng đã đăng nhập. |
| Các bước thực hiện | 1. Chọn “Quản lý học tập”. 2. Hiển thị giao diện danh sách đề kiểm tra. 3. Chọn “Tạo đề kiểm tra trắc nghiệm”. 4. Hiển thị form nhập. 5. Nhập các thông tin của đề kiểm tra (Tên, thời gian, phương thức trộn đề) và Form nhập. 6. Nhấn lưu. 7. Hệ thống thông báo tạo đề thành công. 8. Hệ thống hiện thị giao diện chọn kho câu hỏi. 9. Người dụng chọn kho câu hỏi muốn sử dụng trong đề kiểm tra. 10. Người dùng chọn số lượng câu hỏi trong kho để sử dụng cho đề kiểm tra. 11. Người dùng nhấn lưu. 12. Hệ thống thông báo thêm thành công 13. Người dùng nhấn quay lại. |
| Ngoại lệ | Số lượng câu hỏi được chọn lớn hơn số lượng câu hỏi trong kho. |

Bảng Kịch bản làm bài

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống. |
| Điều kiện | Người dùng đã đăng nhập. |
| Các bước thực hiện | 1. Người dùng chọn “Quản lý học tập”. 2. Hiện thị giao diện danh sách đề kiểm tra. 3. Người dùng nhấn nút “xem” ở đề kiểm tra muốn làm. 4. Hiện thì giao diện thông tin đề kiểm tra. 5. Người dùng nhấn “Làm bài”. 6. Hệ thống hiển thị giao diện làm bài. 7. Người dùng trả lời các câu hỏi trong đề kiểm tra. 8. Người dùng nhấn nút “hoàn tất”. |
| Ngoại lệ |  |

Bảng Kịch bản thống kê kết quả làm bài

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống. |
| Điều kiện | Người dùng đã đăng nhập. |
| Các bước thực hiện | 1. Người dùng chọn “Quản lý học tập”. 2. Hiện thị giao diện danh sách đề kiểm tra. 3. Người dùng nhấn nút “xem” ở đề kiểm tra muốn làm. 4. Hiện thì giao diện thông tin đề kiểm tra. 5. Người dùng nhấn “Thống kê kết quản”. 6. Hiền thị giao diện thống kê kết quả (những câu làm sai, số lần làm sai). 7. Người dùng nhấn thoát. |
| Ngoại lệ | Hết thời gian làm bài.  Hệ thống không kịp lưu kết quả làm bài. |

Bảng Kịch bản tạo nhóm

|  |  |
| --- | --- |
| Actor | Tất cả actor trong hệ thống |
| Điều kiện | Đã đăng nhập. |
| Các bước thực hiện | 1. Chọn “nhóm”. 2. Hệ thống hiện thị danh sách nhóm đã tham gia. 3. Chọn “Tạo nhóm mới”. 4. Hệ thống hiển thị form nhập thông tin nhóm. 5. Nhập thông tin nhóm và ấn lưu. 6. Hệ thống thông báo nhóm đã được tạo. 7. Hệ thống đặt user vừa tạo nhóm là Admin của nhóm. |
| Ngoại lệ |  |

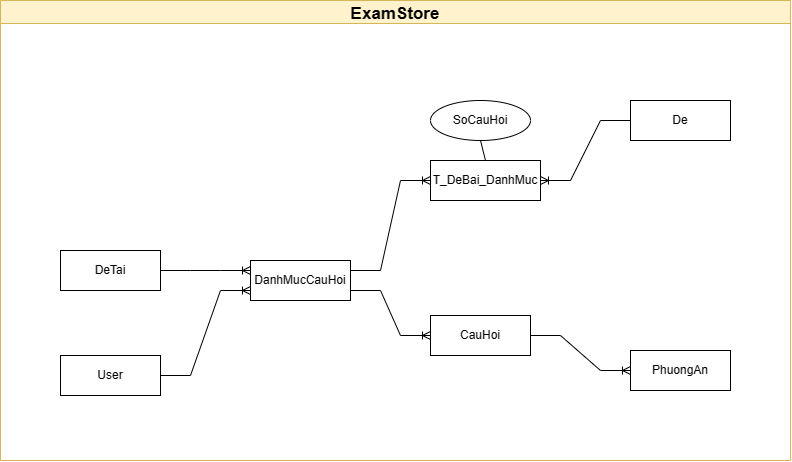
Bảng Kịch bản duyệt xin vào nhóm

|  |  |
| --- | --- |
| Actor | Quản trị viên nhóm |
| Điều kiện | Đã truy cập vào nhóm |
| Các bước thực hiện | 1. Chọn “Quản lý nhóm”. 2. Hệ thống hiển thị giao diện 2 tab danh sách. 3. Chọn tab “Chưa duyệt” để xem danh sách thành viên yêu cầu vào nhóm. 4. Hệ thống hiển thị danh sách thành viên chưa duyệt.   Chọn nút “Chấp nhận” để thêm thành viên.  Chọn nút “Xóa thành viên” để xóa khỏi nhóm. |
| Ngoại lệ | User không còn xin vào nhóm nữa. |

Bảng Kịch bản quản lý thành viên nhóm

|  |  |
| --- | --- |
| Actor | Quản trị viên nhóm |
| Điều kiện | Đã truy cập vào nhóm |
| Các bước thực hiện | 1. Chọn “Quản lý nhóm”. 2. Hệ thống hiển thị giao diện 2 tab danh sách. 3. Chọn tab “Danh sách nhóm” để xem danh sách thành viên. 4. Hệ thống hiển thị giao diện danh sách thành viên trong nhóm. 5. Chọn nút “Xem” để xem thông tin thành viên .   Chọn nút “Xóa thành viên” để xóa khỏi nhóm. |
| Ngoại lệ | User bị xóa là admin nhóm. |

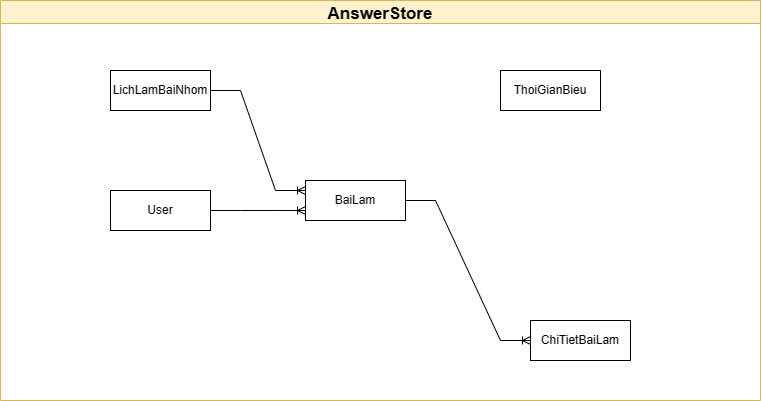
### Mô hình thực thể kết hợp.



Hình Mô hình thực thể kết hợp của Quản lý đề

Bảng Mô tả các thực thể trong mô hình ExamStore

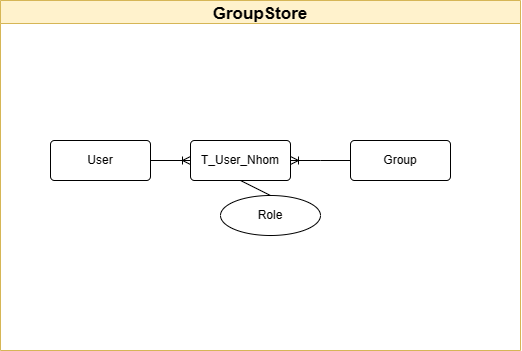
|  |  |  |
| --- | --- | --- |
| STT | Tên thực thể | Mô tả |
| 1 | DanhMucCauHoi | Thực thể giúp phân chia các câu hỏi theo danh mục, chứa (id, MaDM, TenDM, Id\_Nhom, Id\_user). |
| 2 | CauHoi | Thực thể câu hỏi chứa (Id, NoiDung, HuongDan). |
| 3 | PhuongAn | Thực thể phương án lựa chọn cho những câu hỏi nhất định, chứa (Id, NoiDung, Id\_CauHoi). |
| 4 | DeBai | Thực thể lưu trữ cấu hình 1 đề bài, chứa (Id, Ten, ThoiGian, CachTron, Id\_Nhom, Id\_User). |
| 5 | T\_DeBai\_DanhMuc | Thực thể tạo ra từ mối quan hệ N - N của 2 thực thể DanhMucCauHoi và thực thể DeBai cho biết số câu hỏi của danh mục sẽ được dùng trong đề bài, chứa (Id, Id\_DanhMuc, Id\_De). |
| 6 | DeTai | Thực thể giúp phân chia danh mục theo đề tài, chứa (Id, Ten). |



Hình Mô hình thực thể kết hợp của Quản lý bài làm

Bảng Mô tả các thực thể trong mô hình AnswerStore.

|  |  |  |
| --- | --- | --- |
| STT | Tên thực thể | Mô tả |
| 1 | BaiLam | Thực thể Bài làm, chứa  (Id, CreateAt, EndAt, Id\_De, Id\_Lich). |
| 2 | ChiTietBaiLam | Thực thể Chi tiết bài làm, chứa  (Id, Id\_CauHoi, Id\_PhuongAn, Id\_BaiLam). |
| 3 | LichLamBai | Thực thể Lịch làm bài cho nhóm, chứa  (Id, Id\_De, StartAt, EndAt, Id\_Nhom) |
| 4 | ThoiGianBieu | Thời gian biểu cho User, chứa  (Id, GhiChu, ThoiGian, Id\_User). |

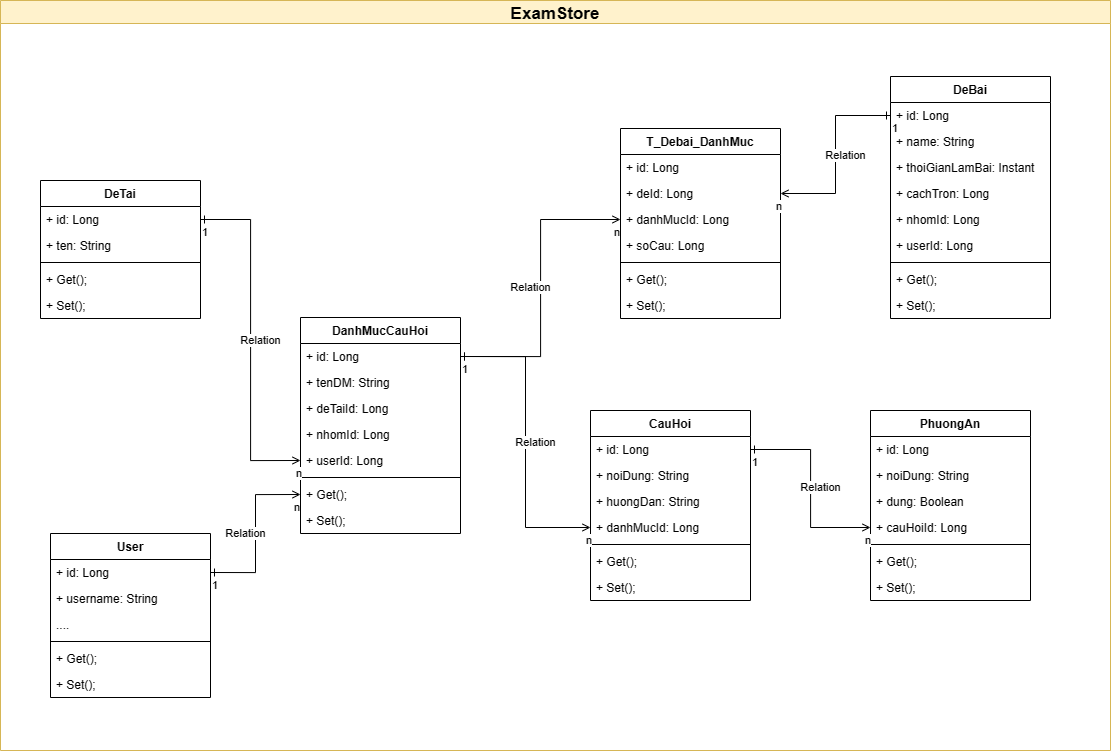


Hình Mô hình thực thể kết hợp của quản lý nhóm

Bảng Mô tả các thực thể trong mô hình GroupStore

|  |  |  |
| --- | --- | --- |
| STT | Tên thực thể | Mô tả |
| 1 | User | Thực thể Thông tin người dùng, chứa  (Id, Username, … ). |
| 2 | Nhom | Thực thể Thông tin nhóm, chứa  (Id, Ten). |
| 3 | T\_User\_Nhom | Thực thể tạo ra từ mối quan hệ N – N của 2 thực thể User và Nhom cho biết quyền của user trong nhóm (Admin, Member, WaitingMember), chứa  (Id, Id\_User, Id\_Nhom, role). |

### Sơ đồ lớp của hệ thống.



Hình Sơ đồ lớp của Microservice ExamStore

**DeTai**: Dùng để lưu thông tin về đề tài. Mỗi đề tài sẽ có id và tên để tránh bị trùng lặp

Bảng Thông tin về các trường của bảng DeTai

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long | Not |
| 2 | Ten |  |  | String |  |

**DanhMucCauHoi**: Dùng để lưu trữ thông tin về danh mục câu hỏi. Mỗi danh mục có thể thuộc về 1 nhóm hoặc cá nhân. Mỗi danh mục câu hỏi có thể thuộc 1 đề tài nhất định.

Bảng Thông tin về các trường của bảng DanhMucCauHoi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | TenDM |  |  | String |  |
| 3 | DeTaiId |  | F | Long |  |
| 4 | NhomId |  | F | Long | N |
| 5 | UserId |  | F | Long | N |

**CauHoi:** Dùng để lưu trữ thông tin câu hỏi. Mỗi câu hỏi thuộc 1 danh mục câu hỏi.

Bảng Thông tin về các trường của CauHoi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | NoiDung |  |  | String |  |
| 3 | HuongDan |  |  | String |  |
| 4 | DanhMucId |  | F | Long |  |

**PhuongAn**: Dùng đề lưu trữ thông tin câu trả lời của câu hỏi. Mỗi phương án thuộc về 1 câu hỏi

Bảng Thông tin về các trường của PhuongAn

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | NoiDung |  |  | String |  |
| 3 | Dung |  |  | Boolean |  |
| 4 | CauHoiId |  | F | Long |  |

**DeBai**: Dùng đề lưu trữ thông tin cấu hình đề bài. Mỗi đề bài có thể thuộc về một user hoặc một nhóm.

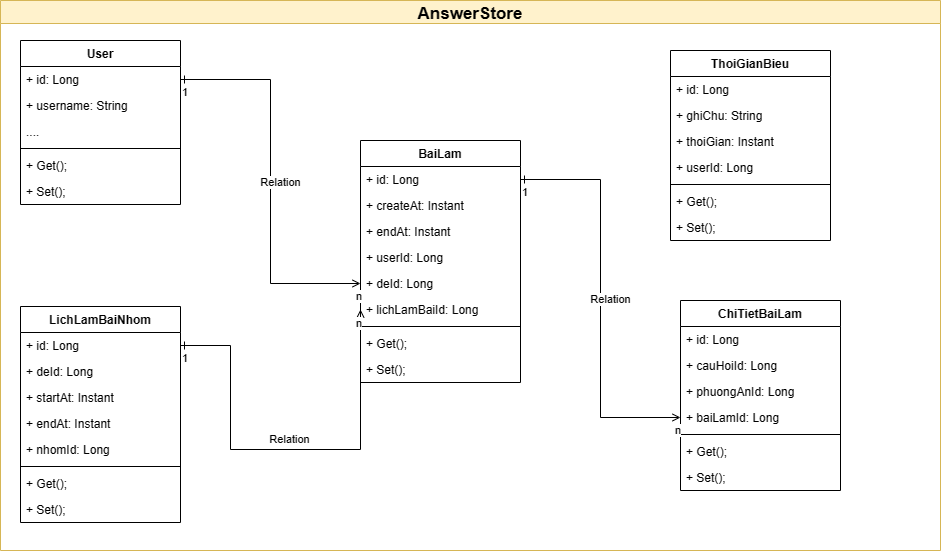
Bảng Thông tin về các trường của DeBai

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | Ten |  |  | String |  |
| 3 | ThoiGianLamBai |  |  | Instant |  |
| 4 | CachTrong |  |  | Long |  |
| 5 | NhomId |  | F | Long | N |
| 6 | UserId |  | F | Long | N |

**T\_DeBai\_DanhMuc**: Dùng đề lưu trữ thông tin chi tiết đề bài. Mỗi T\_DeBai\_DanhMuc thuộc về một DeBai và một DanhMucCauHoi.

Bảng Thông tin về các trường của T\_DeBai\_DanhMuc

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | NoiDung |  |  | String |  |
| 3 | Dung |  |  | Boolean |  |
| 4 | CauHoiId |  | F | Long |  |



Hình Sơ đồ lớp của Microservice AnswerStore

**BaiLam**: Dùng để lưu trữ thông tin bài làm của mỗi user. Mỗi bài làm thuộc về một user và thuộc về một đề bài nhất định. Mỗi bài làm có thể thuộc một lịch làm bài.

Bảng Thông tin về các trường của BaiLam

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | CreateAt |  |  | Instant |  |
| 3 | EndAt |  |  | Instant | N |
| 4 | UserId |  | F | Long |  |
| 5 | DeId |  | F | Long |  |
| 6 | LichLamBaiId |  | F | Long | N |

**ChiTietBaiLam**: Lưu trữ thông tin chi tiết của bài làm. Mỗi chi tiết thuộc về một bài làm. Mỗi chi tiết lưu trữ phương án mà user chọn cho một câu hỏi.

Bảng Thông tin các trường của ChiTietBaiLam

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | CauHoiId |  | F | Long |  |
| 3 | PhuongAnId |  | F | Long |  |
| 4 | BaiLamId |  | F | Long |  |

**LichLamBai**: Lưu trữ thông tin lịch làm bài cho nhóm. Dùng đề quản lý việc làm bài nhóm. Mỗi lịch làm bài thuộc về một nhóm cho một đề bài nhất định.

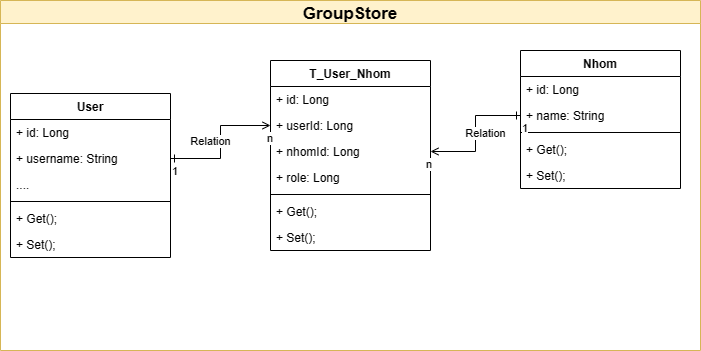
Bảng Thông tin về các trường của LichLamBai

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | DeId |  | F | Long |  |
| 3 | StartAt |  |  | Instant |  |
| 4 | EndAt |  |  | Instant |  |
| 5 | NhomId |  | F | Long |  |

**ThoiGianBieu**: Dùng để thời gian biểu cho user. Mỗi thời gian biểu thuộc về một user.

Bảng Thông tin về các trường của ThoiGianBieu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | GhiChu |  |  | String |  |
| 3 | ThoiGian |  |  | Instant |  |
| 4 | UserId |  | F | Long |  |



Hình Sơ đồ lớp của Microservice GroupStore

**Nhóm**: Lưu trữ thông tin nhóm. Một nhóm có thể có nhiều thành viên với quyền hạn khác nhau.

Bảng Thông tin về các trường của Nhom

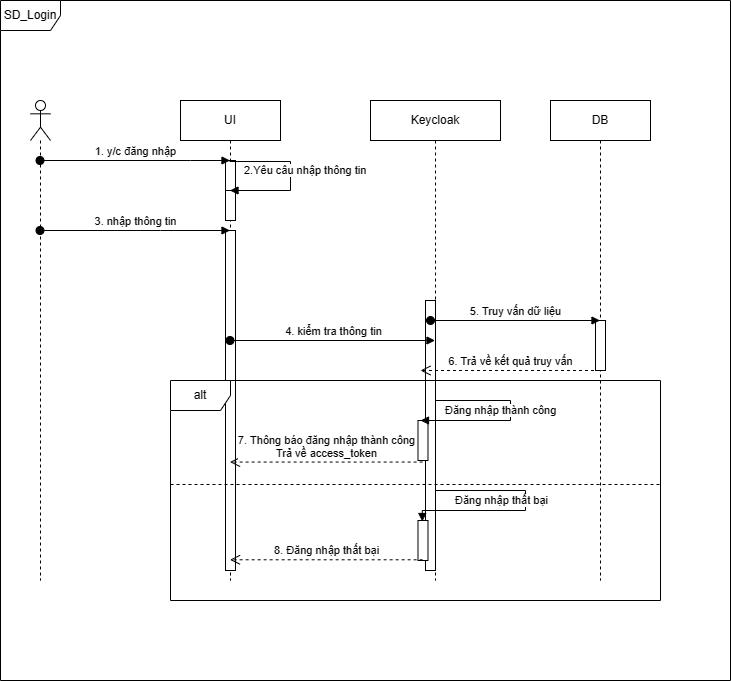
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long | Not |
| 2 | Ten |  |  | String |  |

**T\_User\_Nhom**: Lưu trữ thông tin quyền hạn của các thành viên nhóm.

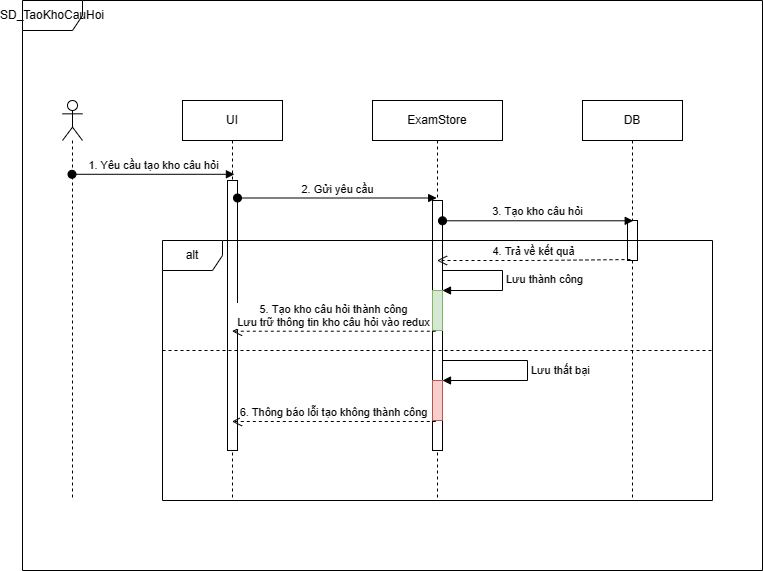
Bảng Thông tin về các trường của T\_User\_Nhom

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Tên cột | PK | FK | Kiểu | Null |
| 1 | Id | P |  | Long |  |
| 2 | UserId |  | F | Long |  |
| 3 | NhomId |  | F | Long |  |
| 4 | Role |  |  | Long |  |

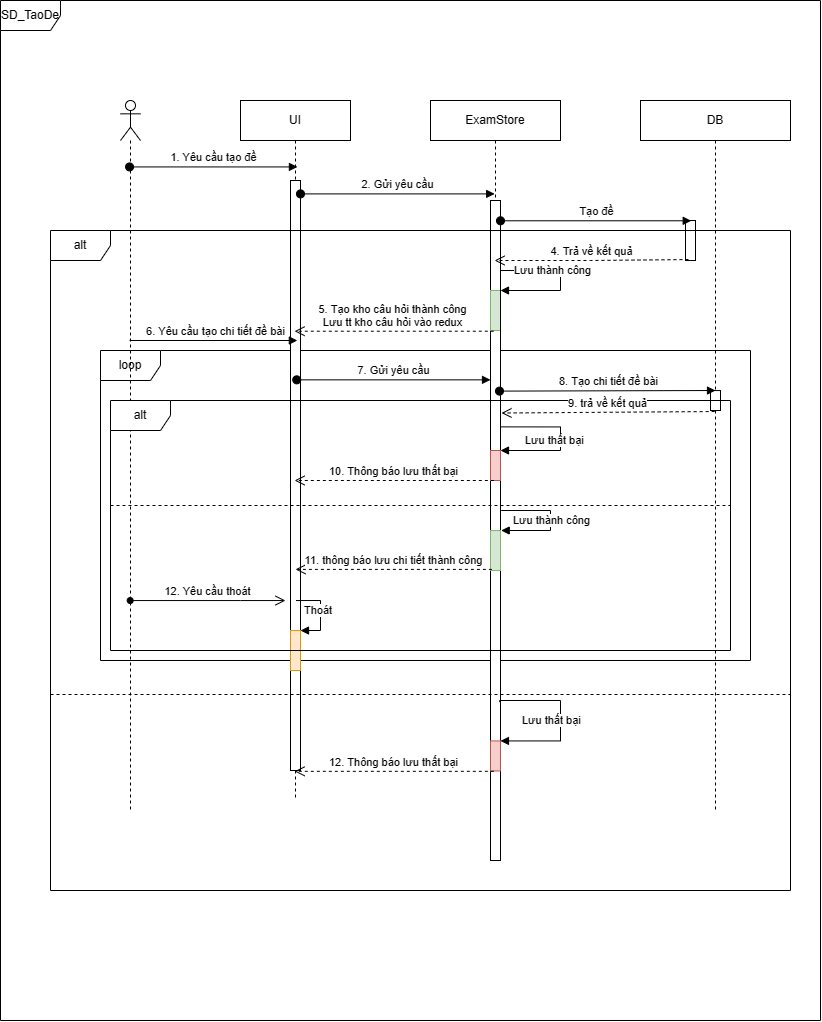
### Các sơ đồ trình tự.



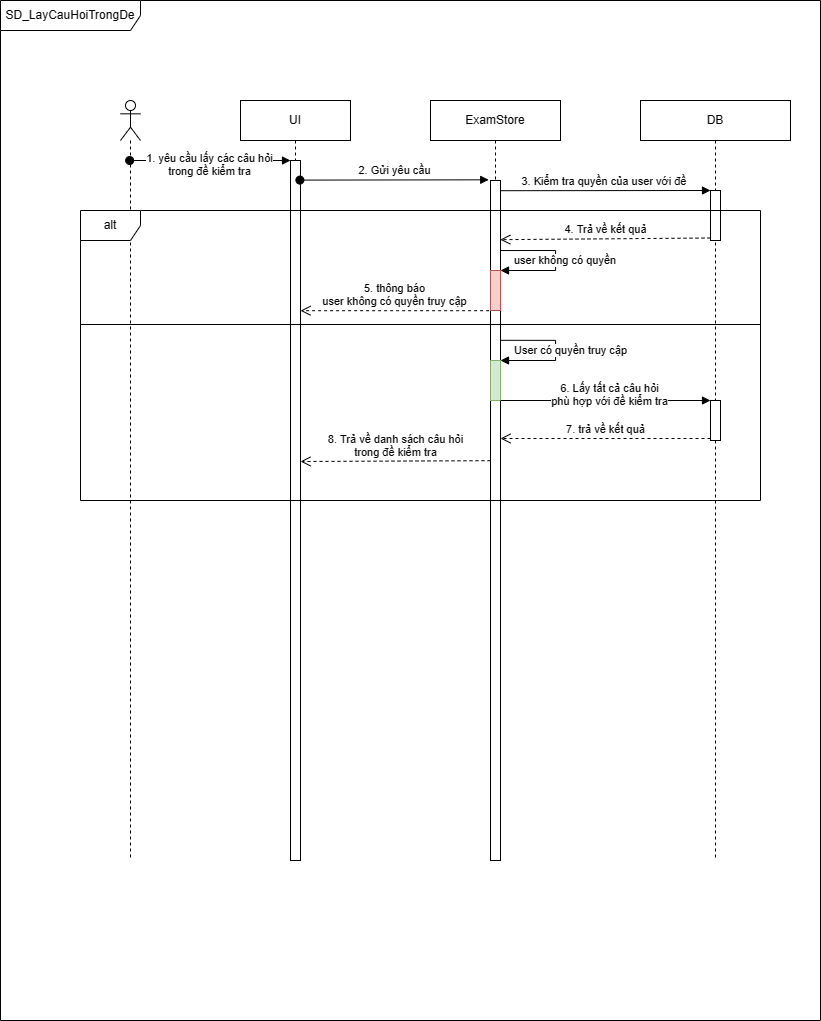
Hình Sơ đồ trình tự đăng nhập



Hình Sơ đồ trình tự tạo kho câu hỏi



Hình Sơ đồ trình tự tạo đề kiểm tra



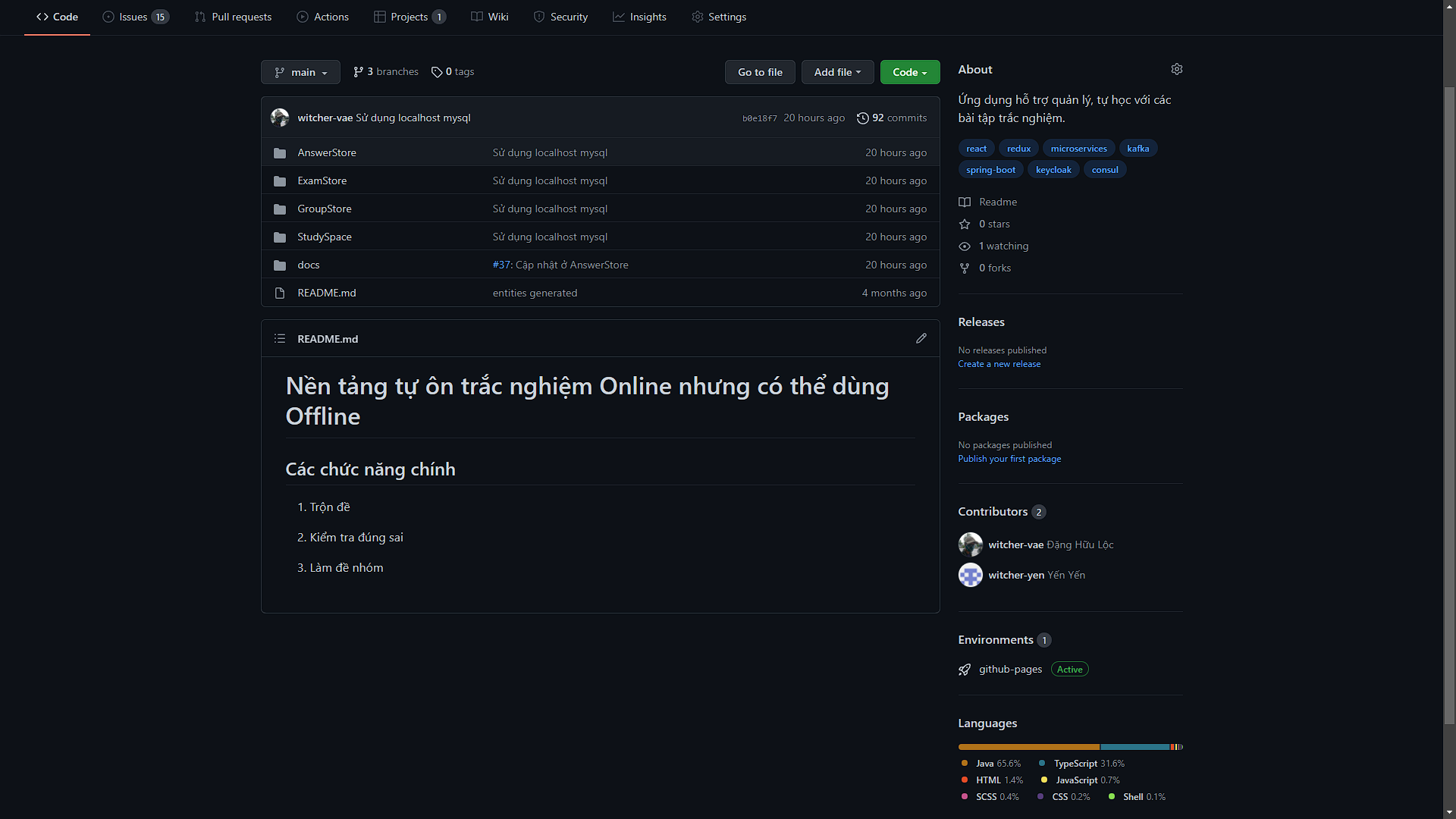
Hình Trình tự lấy câu hỏi trong đề kiểm tra



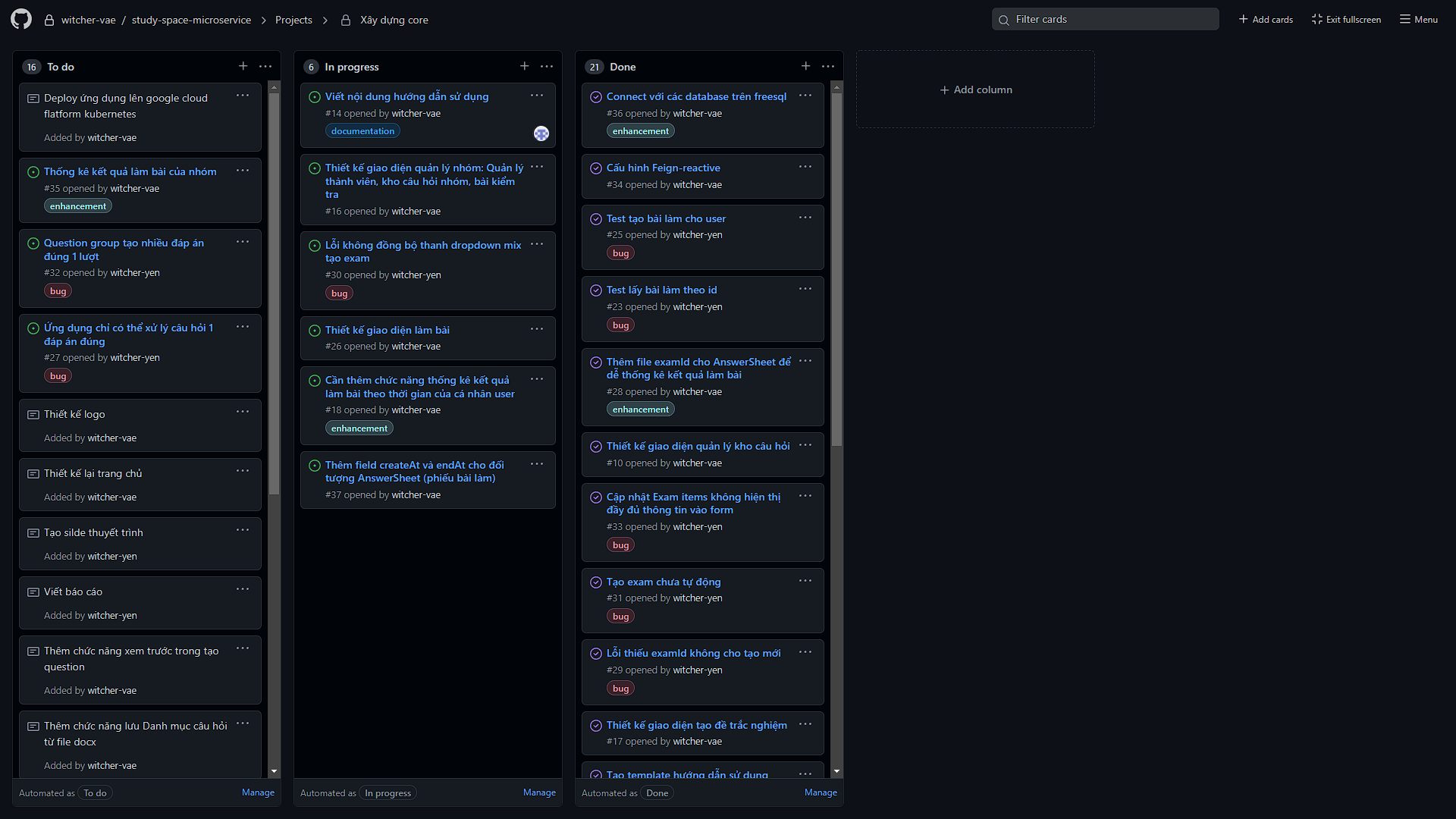
Hình Sơ đồ trình tự làm bài

## Quy trình thực hiện.

Quy trình xây dựng hệ thống:

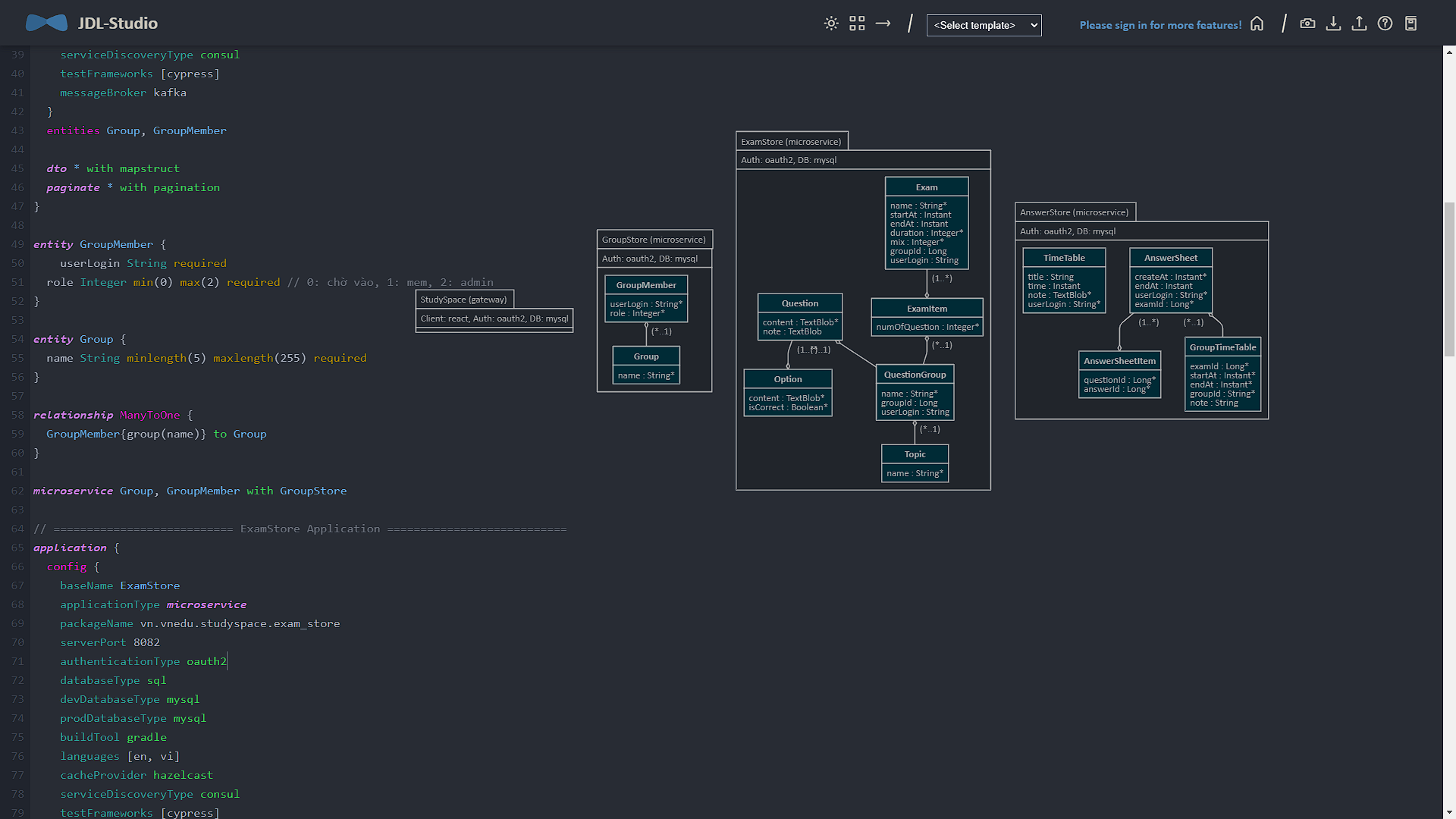


Hình Tạo kho lưu trữ github để quản lý source code



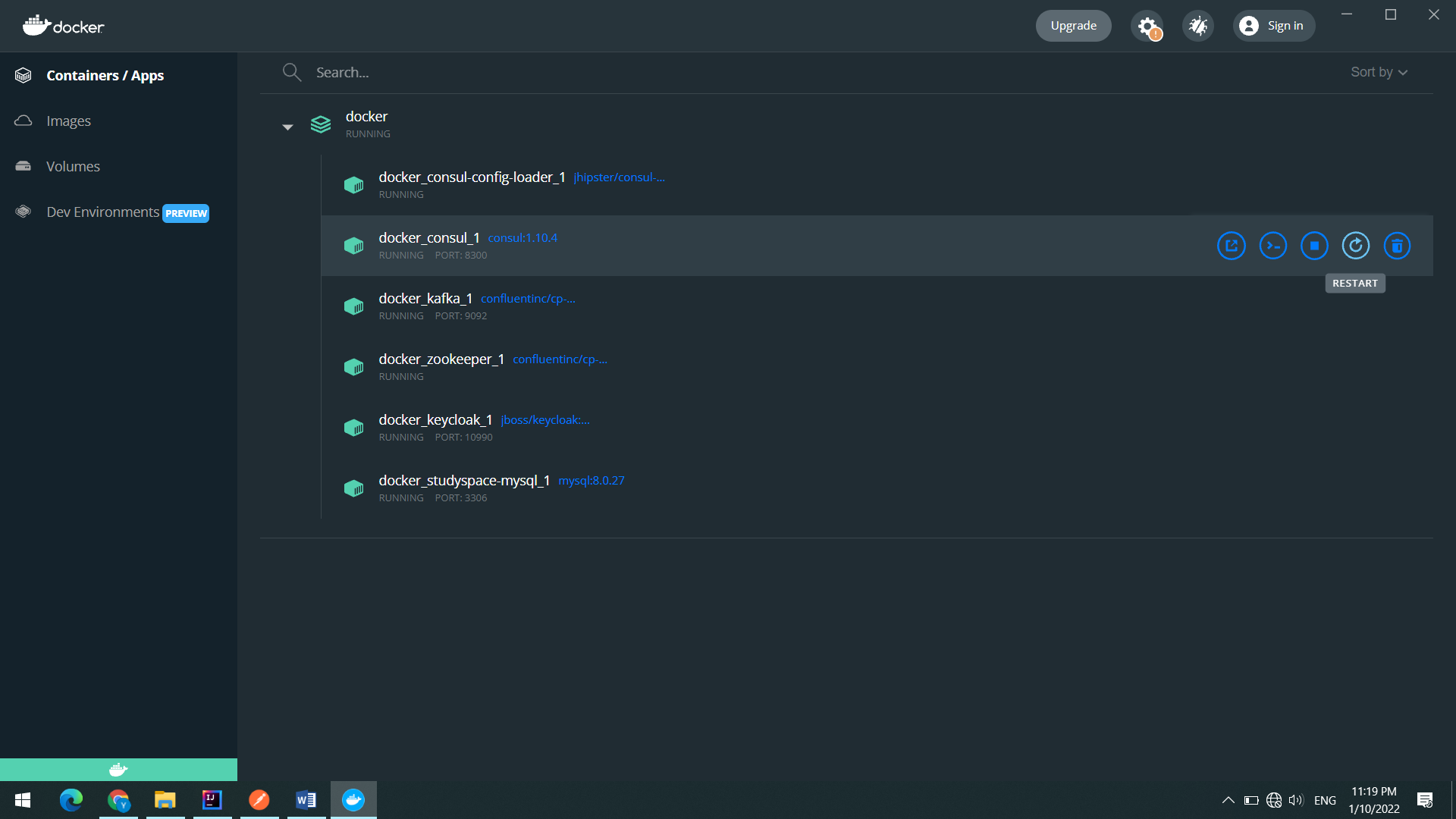
Hình Tạo project trên github để quản lý danh sách công việc cần làm.

Thiết kế cơ sở dữ liệu bằng JDL studio của Jhipster.



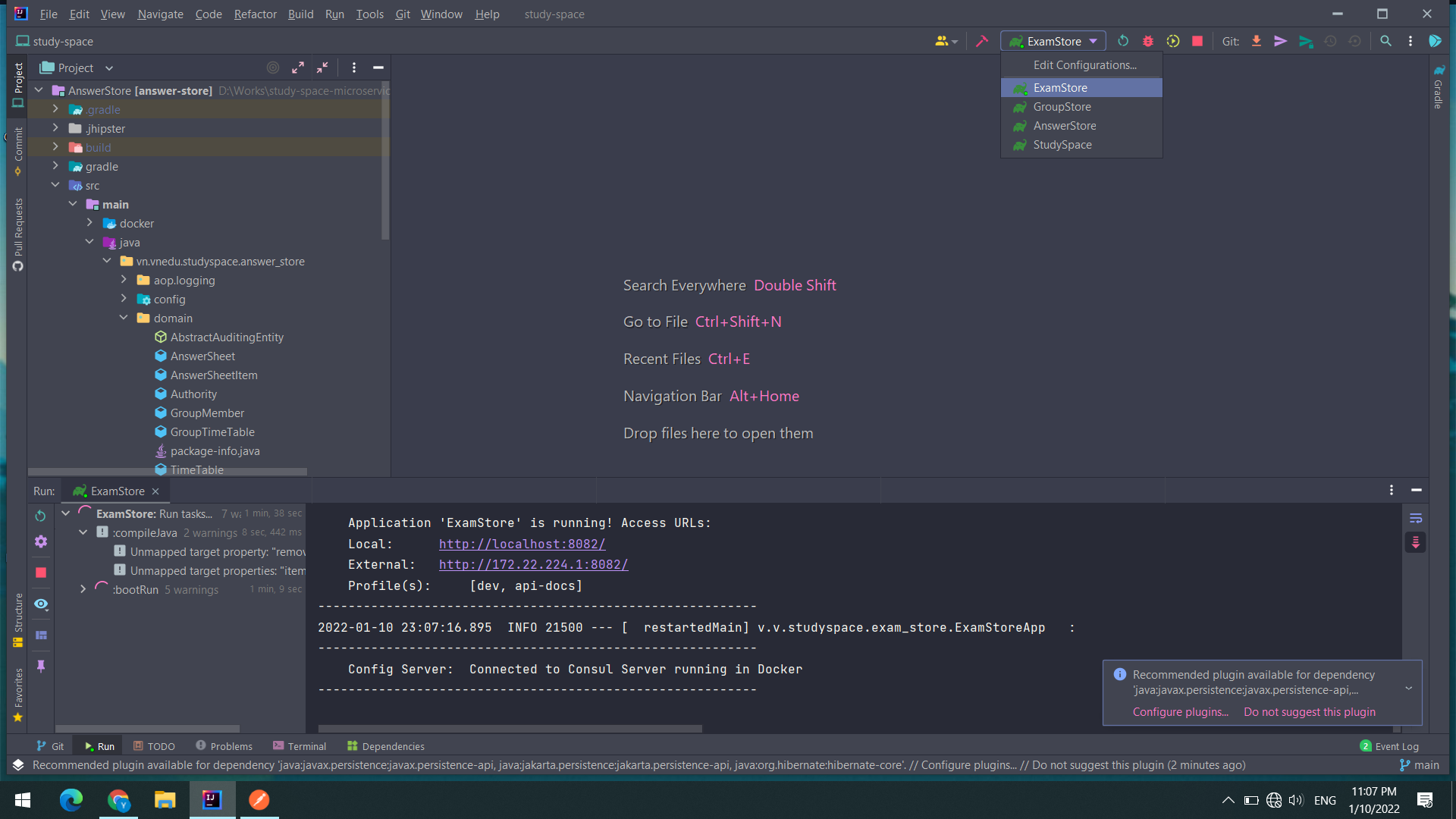
Hình Sử dụng JDL studio

Quy trình kiểm thử hệ thống:



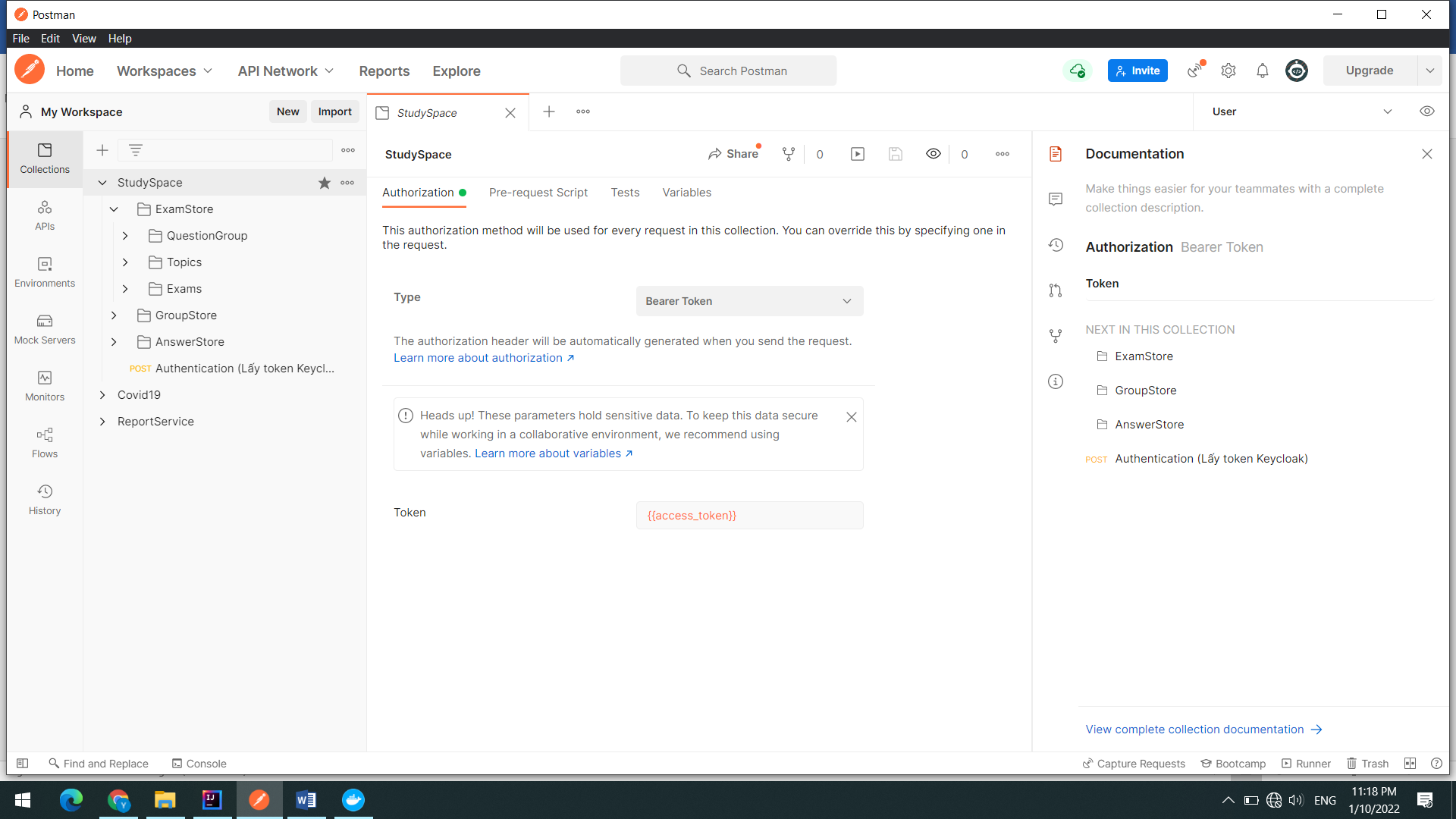
Hình Mở Docker

Mở chương trình IntelliJ IDEA Community Edition và chạy các cấu hình cần kiểm tra như ExamStore, GroupStore, AnswerStore.



Hình Chạy chương trình cấu hình trong IntelliJ IDEA

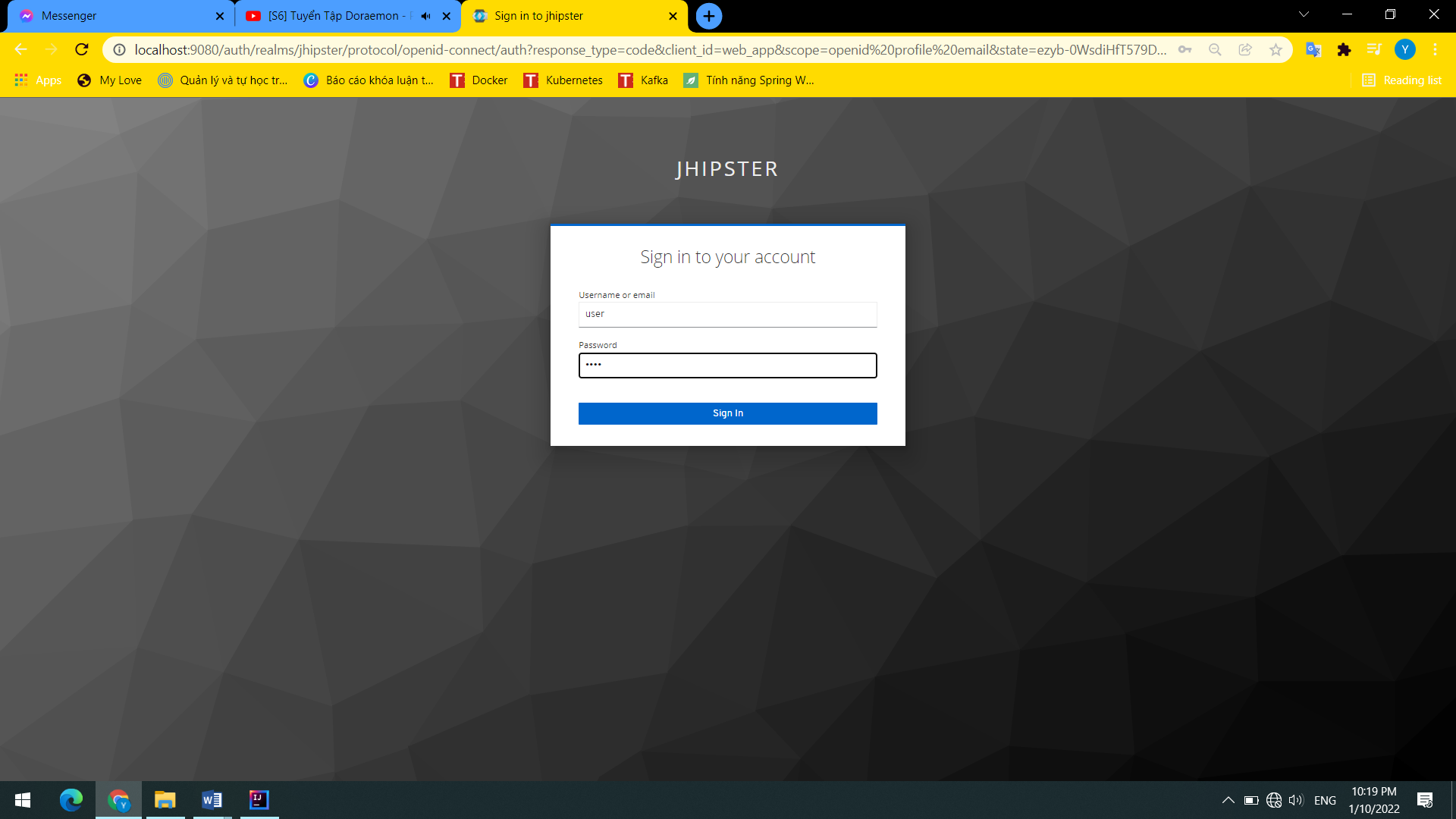
Chạy chương trình Postman để kiểm tra API cho hệ thống. Tạo thành các nhóm folder để việc kiểm tra không bị thiếu sót.



Hình Chương trình Postman

## Một số hình ảnh giao diện hệ thống

* Giao diện đăng nhập hệ thống



Hình Giao diện đăng nhập

* Giao diện quản lý kho câu hỏi

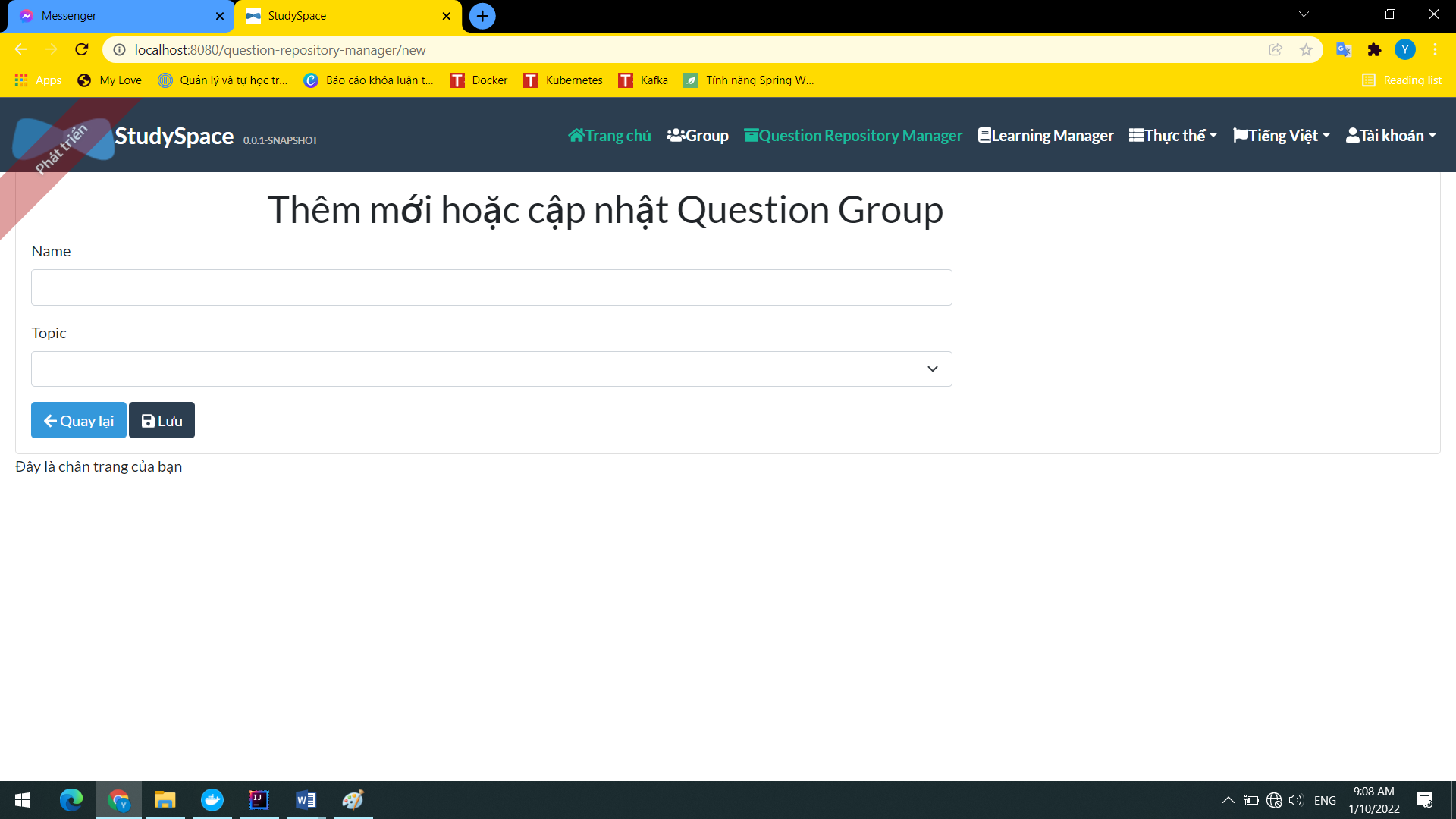
Người dùng chọn trên thanh menu “Quản lý kho câu hỏi”, nếu muốn thêm kho mới chọn “Thêm mới một kho câu hỏi”. Chọn “Sửa” để tạo câu hỏi cho kho hoặc cập nhật thông tin kho câu hỏi.



Hình Giao diện quản lý kho câu hỏi

* Giao diện tạo kho câu hỏi

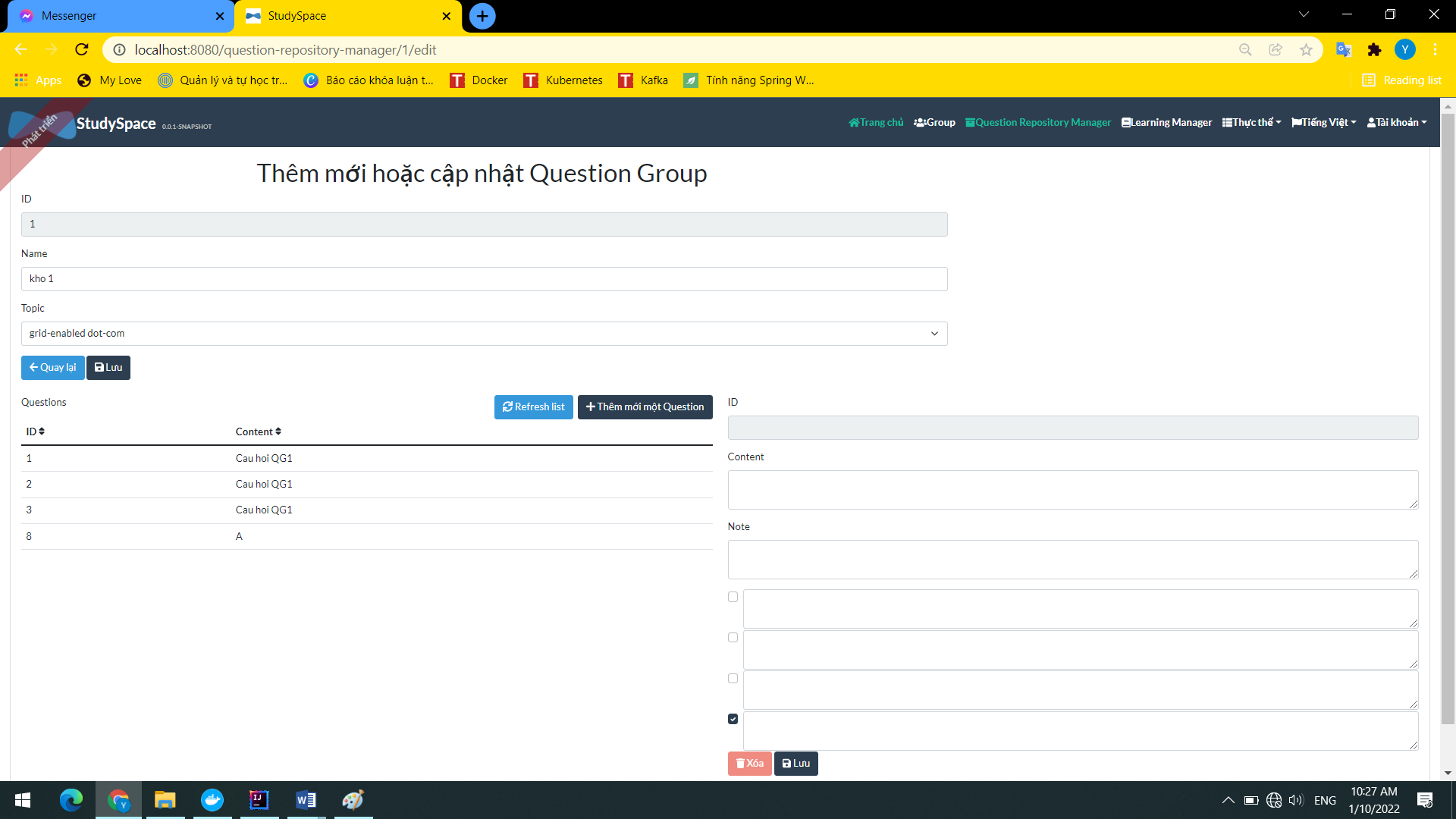
Người dùng nhập tên kho và chọn chủ đề. Sau đó bấm “Lưu” để hoàn tất việc tạo kho mới.



Hình Giao diện tạo mới kho câu hỏi

* Giao diện tạo câu hỏi cho kho

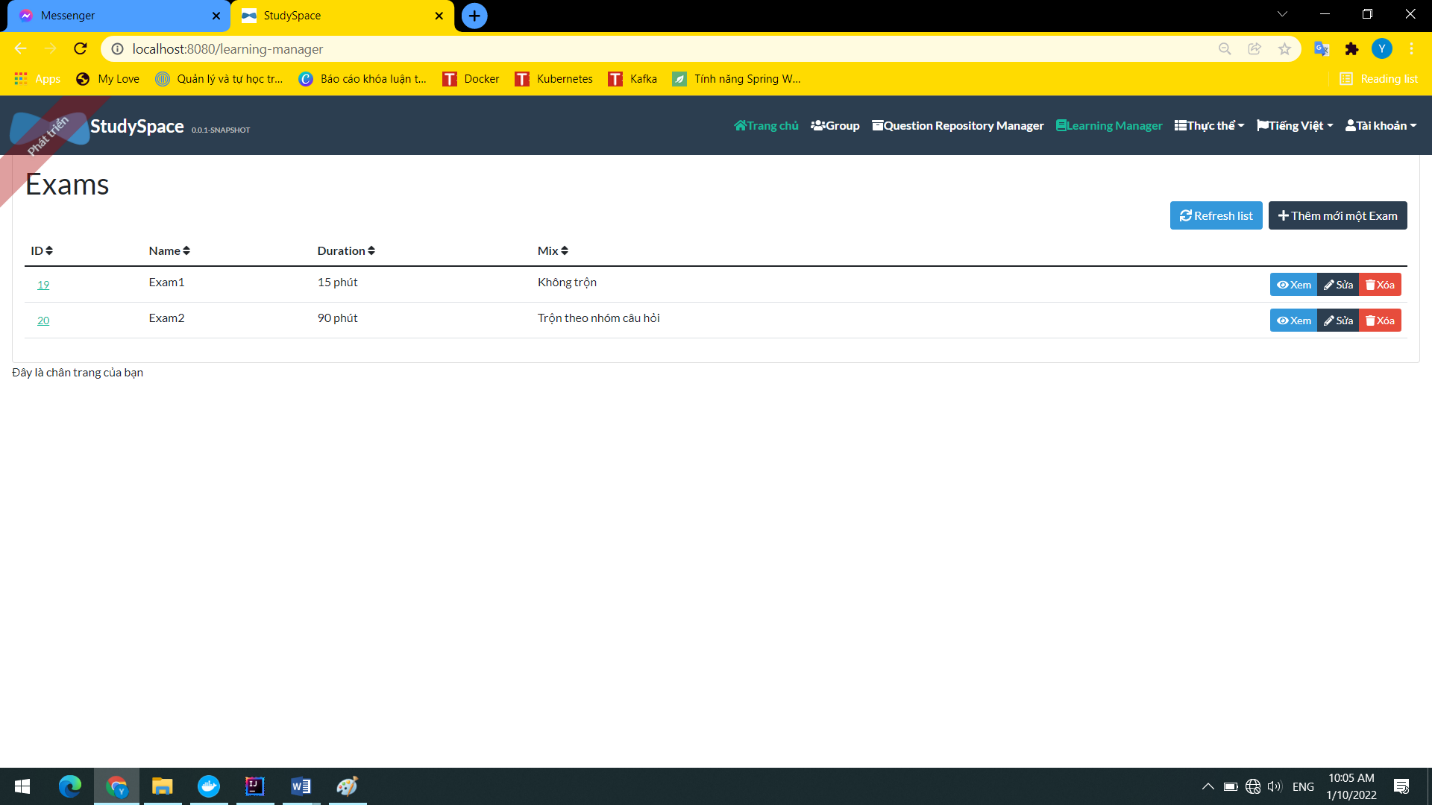
Chọn “Thêm mới một câu hỏi” để thêm câu hỏi cho kho. Nhập thông tin câu hỏi vào form bên trái màn hình. Chọn “Lưu” để hoàn tất thêm câu hỏi mới.



Hình Giao diện tạo câu hỏi cho kho

* Giao diện quản lý đề trắc nghiệm

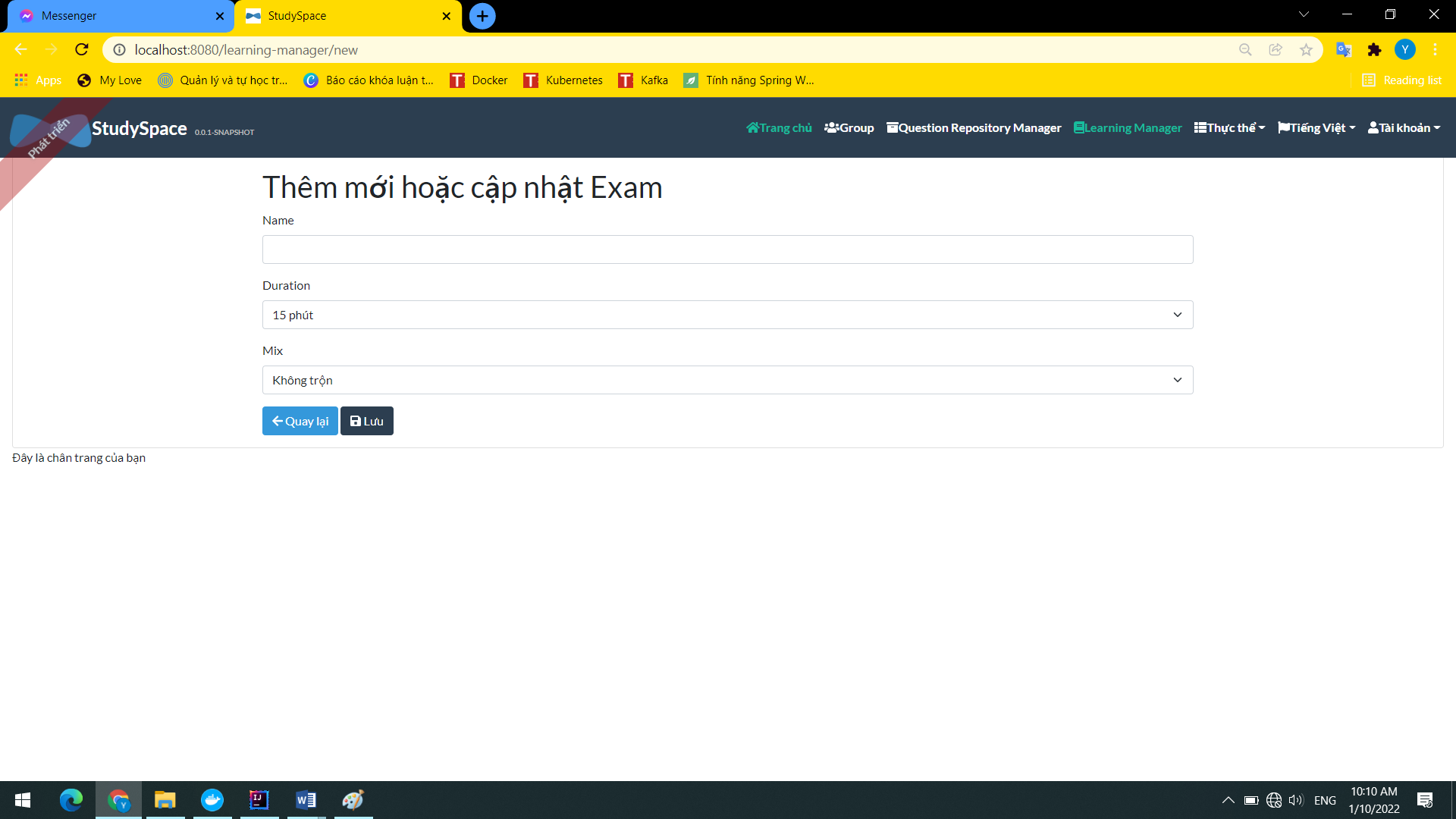
Người dùng chọn trên thanh menu “Quản lý đề”. Chọn “Thêm mới một đề” để tạo đề mới. Để tạo chi tiết đề hoặc cập nhật thông tin đề trắc nghiệm, người dùng chọn nút “Sửa”. Chọn “Xem” để xem chi tiết đề và làm đề trắc nghiệm.



Hình Giao diện quản lý đề trắc nghiệm

* Giao diện tạo đề trắc nghiệm

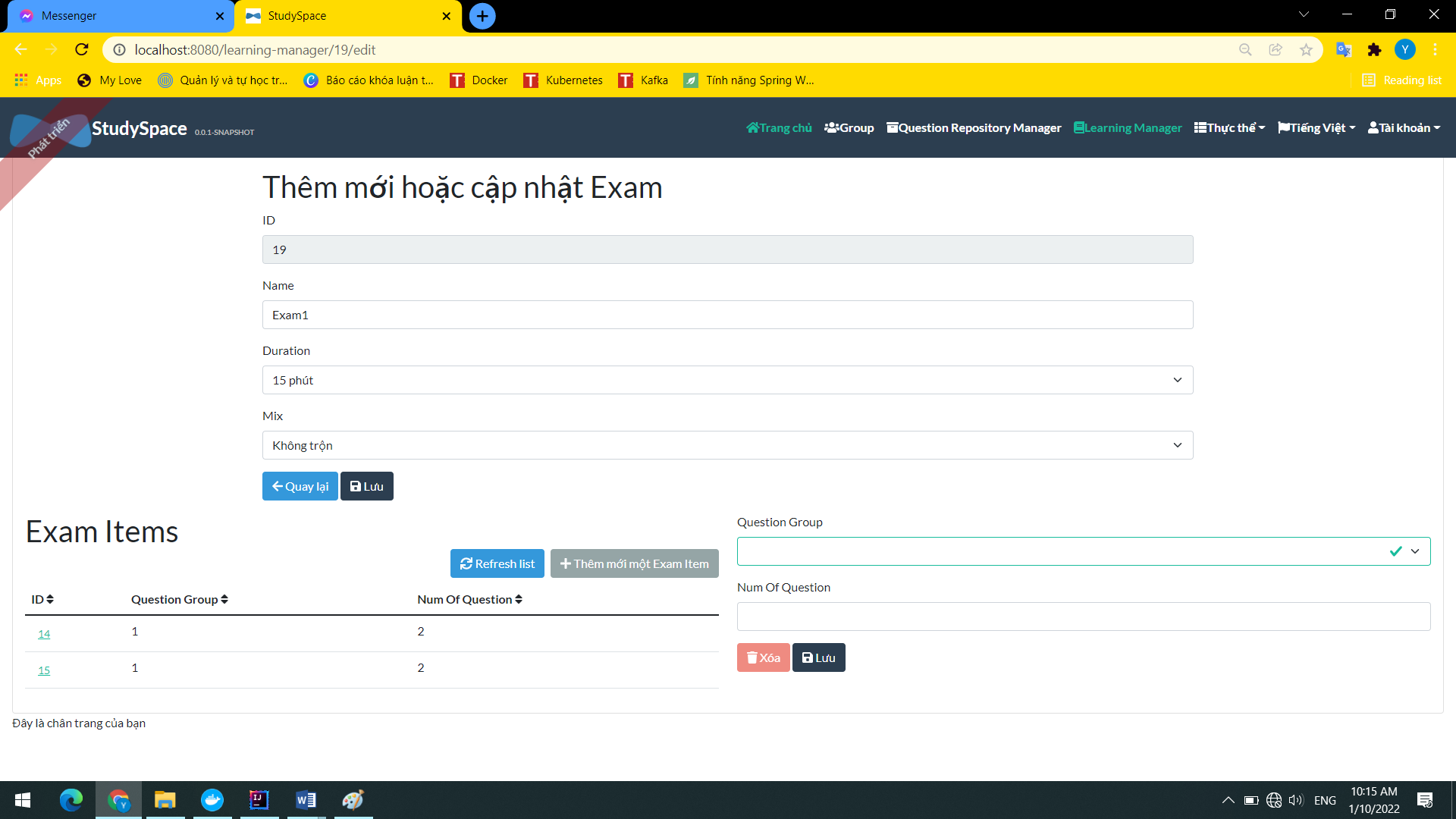
Người dùng nhập tên đề, chọn thời gian làm bài, chọn cách thức trộn đề, sau đó chọn “Lưu” để hoàn tất tạo đề trắc nghiệm.



Hình Giao diện tạo đề trắc nghiệm

* Giao diện tạo chi tiết đề trắc nghiệm

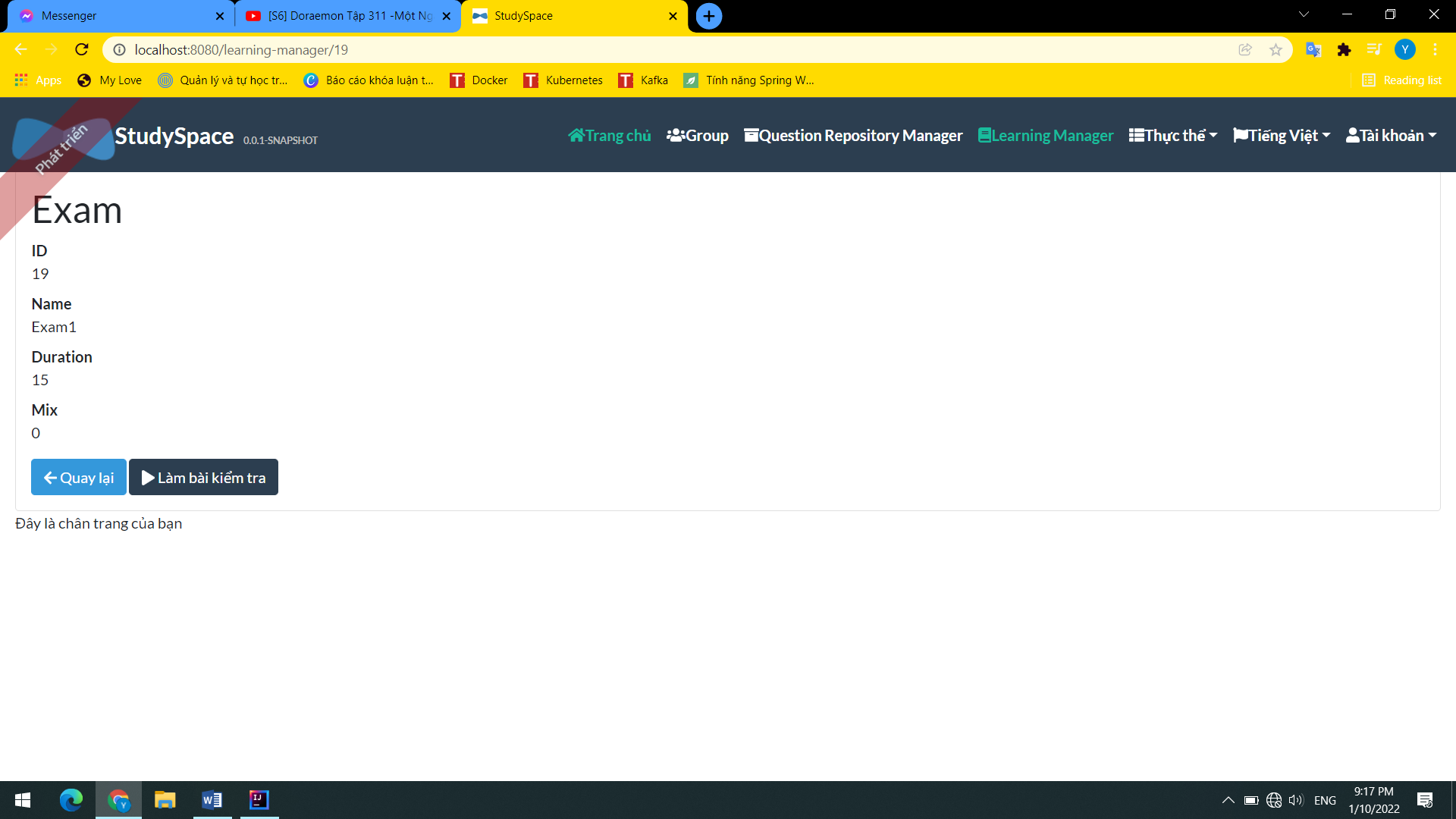
Người dùng chọn “Thêm mới một chi tiết đề”, điền thông tin đầy đủ như chọn kho câu hỏi, số câu hỏi cần lấy trong kho. Chọn “Lưu” để hoàn tất tạo chi tiết cho đề trắc nghiệm.



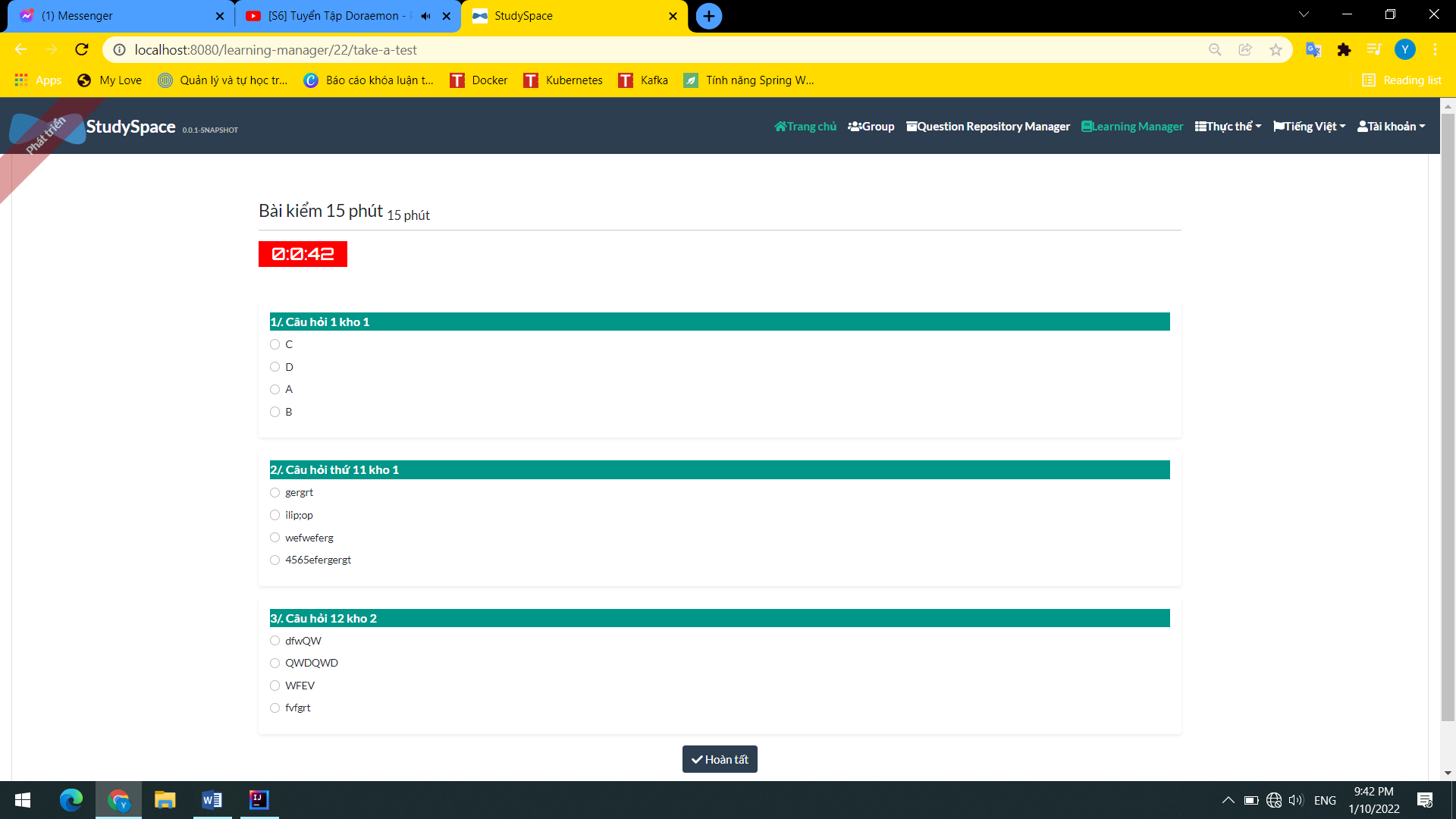
Hình Giao diện tạo chi tiết đề trắc nghiệm

* Giao diện làm đề trắc nghiệm

Chọn “Làm bài kiểm tra” để bắt đầu làm bài.



Hình Giao diện xem chi tiết đề trắc nghiệm

Khi thời gian kết thúc, hệ thống tự động nộp bài làm hoặc người dùng chọn “Hoàn tất” để nộp bài khi hoàn thành xong trong thời gian quy định.

Hình Giao diện làm bài trắc nghiệm

# CHƯƠNG 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

## Kết quả đạt được

Kết quả tìm hiểu về lý thuyết:

Sau thời gian nghiên cứu công nghệ, kiến thức chuyên môn để thực hiện đồ án, em đã biết thêm về nhiều kiến thức mới cũng như nâng cao kĩ năng của bản thân về:

* Phân tích và thiết kế hệ thống thông tin.
* Lập kịch bản cho từng chức năng.
* Thiết kế sơ đồ Usecase, sơ đồ chức năng, sơ đồ nghiệp vụ.
* Hiểu được quy trình phát triển và vòng đời phát triển phần mềm.
* Nắm được kiến trúc Microservices.
* Sử dụng được các Framework, thư viện tốt: Jhipster, Spring Boot, Hibernate,
* ReactJS, Ant Design, ..
* Sử dụng thành thạo các công cụ hỗ trợ tốt cho việc lập trình: IntelliJ IDEA,
* MySQL workbench 8.0, Power Designer, Star UML,…

Kết quả ứng dụng thực nghiệm:

Xây dựng thành công hệ thống với các mục tiêu như:

* Hệ thống đầy đủ các chức năng cơ bản.
* Đáp ứng tương đối đầy đủ cho quy trình nghiệp vụ.
* Bảo mật và phân quyền đầy đủ.
* Thiết kế giao diện trực quan, dễ sử dụng.
* Hệ thống hoạt động tốt đáp ứng các yêu cầu cần thiết của một website ứng dụng công nghệ vào công việc thực tế.

## Khả năng ứng dụng của hệ thống

* Hệ thống khả năng ứng dụng cao, hỗ trợ học sinh, sinh viên phát huy khả năng tự học và ôn luyện trắc nghiệm.
* Xây dựng được các chức năng như: quản lý đề trắc nghiệm, quản lý kho câu hỏi, làm bài trắc nghiệm, tính điểm tự động.
* Ứng dụng được xây dựng dựa trên công nghệ Microservices nên có thể mở rộng, nâng cấp tính năng dễ dàng.

## Hạn chế

Tiếp cận nhiều công nghệ mới cùng lúc nên có thể còn những hạn chế, chưa thể phát huy tối đa hiệu quả mà các công nghệ đó mang lại.

Vì thời gian còn hạn chế và gặp nhiều khó khăn trong tình hình dịch bệnh Covid-19 nên hệ thống vẫn chưa hoàn thành chức năng nhóm cho hệ thống.

## Hướng phát triển

Để tiếp tục phát triển đề tài này, nhóm em nhận thấy cần phải tiếp tục thực hiện một số việc sau:

* Xử lý các lỗi chặt chẽ hơn trước khi đưa vào sử dụng.
* Mở rộng bài toán cho nhiều trường riêng biệt.
* Thêm tính năng nhóm cho hệ thống.
* Bổ sung nhiều hơn các báo cáo, thống kê.
* Thêm tính năng ghi log thông báo chi tiết khi truy cập chương trình và thực hiện thao tác thay đổi trong hệ thống để có thể giám sát hệ thống.
* Phát triển các tính năng hỗ trợ thanh toán (ATM, PayNet, …) khi tổ chức thành các lớp online.

# TÀI LIỆU THAM KHẢO