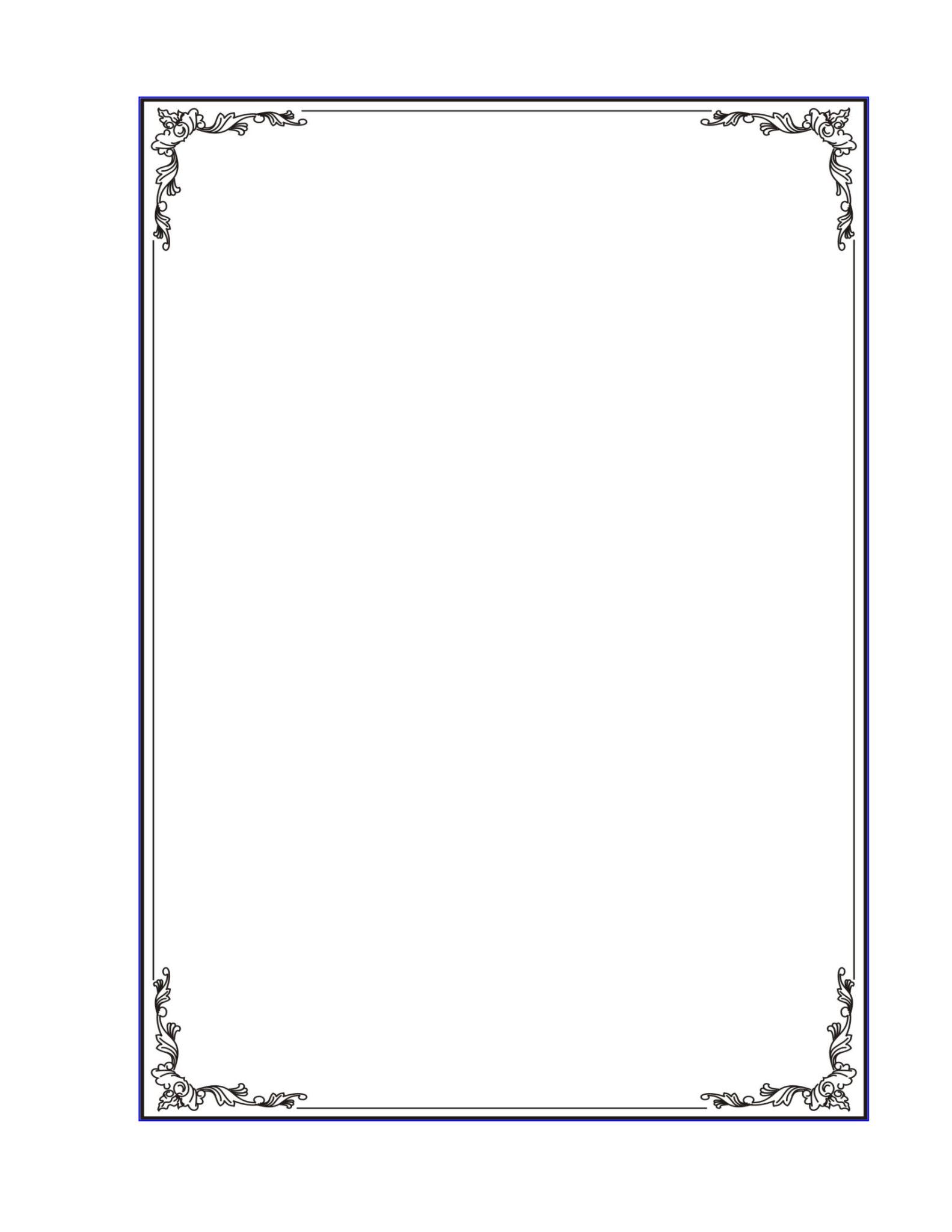
****NGÂN HÀNG NHÀ NƯỚC VIỆT NAM

**HỌC VIỆN NGÂN HÀNG**

**ĐỒ ÁN TỐT NGHIỆP**

**TÊN ĐỀ TÀI:**

***Xây dựng hệ thống quản lý và đánh giá bài tập lập trình***

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | : |
| **Đơn vị công tác của giảng viên** | : |
| **Chương trình đào tạo** | : |
| **Lớp** | : |
| **Sinh viên thực hiện** | **Mã sinh viên** |
|  |  |
|  |  |
|  |  |

**Hà Nội – 2025**

NGÂN HÀNG NHÀ NƯỚC VIỆT NAM

**HỌC VIỆN NGÂN HÀNG**

**ĐỒ ÁN TỐT NGHIỆP**

**TÊN ĐỀ TÀI:**

***Xây dựng hệ thống quản lý và đánh giá bài tập lập trình***

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn** | : |
| **Đơn vị công tác của giảng viên** | : |
| **Chương trình đào tạo** | : |
| **Lớp** | : |
| **Sinh viên thực hiện** | **Mã sinh viên** |
|  |  |
|  |  |
|  |  |

**Hà Nội – 2025**

# NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

Nhận xét:

**Xác nhận của giảng viên hướng dẫn**

# LỜI CẢM ƠN

# LỜI CAM ĐOAN

Nhóm sinh viên thực hiện

# MỤC LỤC

[LỜI CẢM ƠN 1](#_Toc195300013)

[LỜI CAM ĐOAN 2](#_Toc195300014)

[MỤC LỤC 3](#_Toc195300015)

[DANH MỤC BẢNG, HÌNH VẼ 6](#_Toc195300016)

[LỜI MỞ ĐẦU 9](#_Toc195300017)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG QUẢN LÝ VÀ ĐÁNH GIÁ BÀI TẬP LẬP TRÌNH 11](#_Toc195300018)

[1.1. Giới thiệu 11](#_Toc195300019)

[1.1.1. Khái niệm và mục đích 11](#_Toc195300020)

[1.1.2. Quy trình hoạt động 12](#_Toc195300021)

[1.1.3. Các mô hình chấm bài phổ biến trong hệ thống đánh giá bài lập trình 13](#_Toc195300022)

[1.1.4. Các hệ thống đánh giá bài tập lập trình trực tuyến 15](#_Toc195300023)

[1.2. Thực trạng hệ thống đánh giá bài tập lập trình trực tuyến tại các trường đại học trên thế giới và trong nước 20](#_Toc195300024)

[1.2.1. Các trường đại học trên thế giới 20](#_Toc195300025)

[1.2.2. Các trường đại học trong nước 21](#_Toc195300026)

[1.3. Các thành phần chính của hệ thống 23](#_Toc195300027)

[1.3.1. Thành phần quản lý người dùng 23](#_Toc195300028)

[1.3.2. Thành phần quản lý bài tập 24](#_Toc195300029)

[1.3.3. Thành phần thực thi và đánh giá mã nguồn 24](#_Toc195300030)

[1.3.4. Thành phần giao tiếp và cộng đồng 24](#_Toc195300031)

[1.3.5. Thành phần theo dõi và báo cáo 25](#_Toc195300032)

[1.3.6. Giao diện và khả năng truy cập 25](#_Toc195300033)

[1.3.7. Thành phần bảo mật và lưu trữ 25](#_Toc195300034)

[1.4. Bài toán quản lý và bài tập lập trình cho sinh viên - giảng viên Khoa Công nghệ thông tin và Kinh tế số - Học viện Ngân hàng 26](#_Toc195300035)

[1.4.1. Phát biểu bài toán 26](#_Toc195300036)

[1.4.2. Mục đích, mục tiêu 29](#_Toc195300037)

[1.4.3. Phạm vi 29](#_Toc195300038)

[1.4.4. Giải pháp đề xuất và công nghệ sử dụng 30](#_Toc195300039)

[1.5. Kế hoạch thực hiện dự án 33](#_Toc195300040)

[CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 34](#_Toc195300041)

[2.1. Mô hình nghiệp vụ của hệ thống 34](#_Toc195300042)

[2.2. Phân tích yêu cầu hệ thống 35](#_Toc195300043)

[2.2.1. Xác định các ca sử dụng 35](#_Toc195300044)

[2.2.2. Xây dựng biểu đồ ca sử dụng 37](#_Toc195300045)

[2.2.3. Đặc tả ca sử dụng 40](#_Toc195300046)

[2.3. Mô hình hóa hoạt động của hệ thống 56](#_Toc195300047)

[2.3.1. Mô hình hoạt động xác thực và điều hướng 56](#_Toc195300048)

[2.3.2. Biểu dồ tuần tự hoạt động tham gia lớp học 57](#_Toc195300049)

[2.3.3. Biểu đồ tuần tự của hoạt động làm bài và nộp bài tập lập trình 57](#_Toc195300050)

[2.3.4. Biểu đồ tuần tự của hoạt động tạo bài tập lập trình trong lớp học phần 58](#_Toc195300051)

[2.3.5. Biểu đồ tuần tự hoạt động xem thông tin nộp bài của sinh viên 58](#_Toc195300052)

[2.4. Biểu đồ lớp của hệ thống 59](#_Toc195300053)

[2.5. Thiết kế cơ sở dữ liệu 60](#_Toc195300054)

[2.5.1.Thiết kế mức khái niệm 60](#_Toc195300055)

[2.5.2. Thiết kế mức vật lý 62](#_Toc195300056)

[2.6. Thiết kế giao diện người dùng (UI/UX) 68](#_Toc195300057)

[2.6.1. Thiết kế wireframe giao diện cho sinh viên 68](#_Toc195300058)

[2.6.2 Thiết kế wireframe giao diện cho giảng viên 70](#_Toc195300059)

[2.6.3 Thiết kế wireframe giao diện cho quản trị viên 70](#_Toc195300060)

[CHƯƠNG 3: CÀI ĐẶT VÀ KIỂM THỬ HỆ THỐNG 71](#_Toc195300061)

[3.1. Bảo mật hệ thống 71](#_Toc195300062)

[3.1.1. Bảo vệ chống tấn công XSS bằng XSRF token 71](#_Toc195300063)

[3.1.2. Mã hóa mật khẩu 71](#_Toc195300064)

[3.2. Xây dựng giao hiện hệ thống 72](#_Toc195300065)

[3.2.1. Giao diện cho sinh viên 72](#_Toc195300066)

[3.2.2. Giao diện cho giảng viên 75](#_Toc195300067)

[3.2.3. Giao diện cho admin 77](#_Toc195300068)

[3.3. Triển khai Judge0 79](#_Toc195300069)

[3.3.1. Quy trình hoạt động của Judge0 trong hệ thống 79](#_Toc195300070)

[3.3.2. Tích hợp với các thành phần khác của hệ thống 80](#_Toc195300071)

[3.3.3. Tùy chỉnh Judge0 trong hệ thống 80](#_Toc195300072)

[3.4. Xây dựng mô đun 81](#_Toc195300073)

[3.4.1. Mô đun: Sinh viên nhập mã để vào lớp 81](#_Toc195300074)

[3.4.2. Mô đun: Sinh viên nộp bài tập lập trình 82](#_Toc195300075)

[3.4.3. Mô đun: Giảng viên tạo bài tập mới 83](#_Toc195300076)

[3.4.4. Mô đun: Admin thêm học phần và lớp học phần 85](#_Toc195300077)

[3.5. Thử nghiệm 86](#_Toc195300078)

[3.5.1. Thử nghiệm mô đun: Sinh viên nhập mã để vào lớp 86](#_Toc195300079)

[3.5.2. Thử nghiệm mô đun: Sinh viên nộp bài tập lập trình 88](#_Toc195300080)

[3.5.3. Thử nghiệm mô đun: Giảng viên tạo bài tập mới 90](#_Toc195300081)

[3.5.4. Thử nghiệm mô đun: Admin thêm học phần và lớp học phần 92](#_Toc195300082)

[KẾT LUẬN 95](#_Toc195300083)

[TÀI LIỆU THAM KHẢO 97](#_Toc195300084)

# DANH MỤC BẢNG, HÌNH VẼ

[Bảng 1.1. Bảng so sánh các hệ thống chấm bài tự động (Nguồn: tác giả) 18](#_Toc195290039)

[Bảng 1.2. Bảng so sánh các hệ thống chấm bài tự động của các trường tại Việt Nam (Nguồn: tác giả) 23](#_Toc195290040)

[Bảng 1.3. Bảng kế hoạch thực hiện dự án 33](#_Toc195290041)

[Bảng 2.1. Bảng các ca sử dụng cho tác nhân Admin 36](#_Toc195290042)

[Bảng 2.2. Bảng các ca sử dụng cho tác nhân Giảng viên 36](#_Toc195290043)

[Bảng 2.3. Bảng các ca sử dụng cho tác nhân Sinh viên 37](#_Toc195290044)

[Bảng 2.4. Bảng đặc tả ca sử dụng đăng nhập của Admin 41](#_Toc195290045)

[Bảng 2.5. Bảng đặc tả ca sử dụng quản lý người dùng (Tạo, Chỉnh sửa, Xoá) của Admin 42](#_Toc195290046)

[Bảng 2.6. Bảng đặc tả ca sử dụng quản lý lớp học (Tạo, Chỉnh sửa, Xoá) của Admin 44](#_Toc195290047)

[Bảng 2.7. Bảng đặc tả ca sử dụng quản lý sinh viên lớp học (Tạo, Chỉnh sửa, Xoá) của Admin 45](#_Toc195290048)

[Bảng 2.8. Bảng đặc tả ca sử dụng thêm giảng viên vào lớp học phần của Admin 46](#_Toc195290049)

[Bảng 2.9. Bảng đặc tả ca sử dụng tạo bài tập lập trình cố định của Admin 46](#_Toc195290050)

[Bảng 2.10. Bảng đặc tả ca sử dụng đăng nhập của Giảng viên 47](#_Toc195290051)

[Bảng 2.11. Bảng đặc tả ca sử dụng quản lý bài tập lập trình cho lớp học (Tạo, Chỉnh sửa, Ẩn) của Giảng viên 49](#_Toc195290052)

[Bảng 2.12. Bảng đặc tả ca sử dụng quản lý sinh viên trong lớp học phần của Giảng viên 49](#_Toc195290053)

[Bảng 2.13. Bảng đặc tả ca sử dụng theo dõi kết quả làm bài tập của sinh viên cho Giảng viên 50](#_Toc195290054)

[Bảng 2.14. Bảng đặc tả ca sử dụng đăng ký của Sinh viên 51](#_Toc195290055)

[Bảng 2.15. Bảng đặc tả ca sử dụng đăng nhập của Sinh viên 51](#_Toc195290056)

[Bảng 2.16. Bảng đặc tả ca sử dụng tham gia lớp học của Sinh viên 52](#_Toc195290057)

[Bảng 2.17. Bảng đặc tả ca sử dụng làm bài tập lập trình được giao 53](#_Toc195290058)

[Bảng 2.18. Bảng đặc tả ca sử dụng xem lại mã nguồn của Sinh viên 54](#_Toc195290059)

[Bảng 2.19. Bảng đặc tả ca sử dụng tham gia cuộc trò chuyện chung của Sinh viên 55](#_Toc195290060)

[Bảng 2.20. Bảng đặc tả ca sử dụng tham gia diễn đàn cộng đồng của Sinh viên 55](#_Toc195290061)

[Bảng 2.21. Xác định các thực thể 60](#_Toc195290062)

[Bảng 2.22. Thực thể và thuộc tính của thực thể 60](#_Toc195290063)

[Bảng 2.23. Quan hệ giữa các thực thể 61](#_Toc195290064)

[Hình 1.1. Quy trình chấm bài lập trình tự động (Nguồn: tác giả) 12](#_Toc195107405)

[Hình 1.2. Chương trình khung ngành CNTT (Nguồn: Khung CTĐT của HVNH áp dụng từ năm 2021) 27](#_Toc195107406)

[Hình 1.3. Chương trình khung ngành HTTTQL (Nguồn: Khung CTĐT của HVNH áp dụng từ năm 2021) 27](#_Toc195107407)

[Hình 1.4. Đánh giá học phần Cơ sở lập trình (Nguồn: Khung CTĐT của ngành CNTT) 28](#_Toc195107408)

[Hình 2.1. Quy trình tổ chức lớp học 34](#_Toc195296280)

[Hình 2.2. Quy trình tham gia lớp học 34](#_Toc195296281)

[Hình 2.3. Quy trình giao bài tập của giảng viên 35](#_Toc195296282)

[Hình 2.4. Quy trình làm bài tập lập trình 35](#_Toc195296283)

[Hình 2.5. Biểu đồ ca sử dụng tổng quát của hệ thống 37](#_Toc195296284)

[Hình 2.6. Biểu đồ phân rã ca sử dụng của tác nhân Admin 38](#_Toc195296285)

[Hình 2.7. Biểu đồ phân rã ca sử dụng của tác nhân Giảng viên 39](#_Toc195296286)

[Hình 2.8. Biểu đồ phân rã ca sử dụng của tác nhân Sinh viên 40](#_Toc195296287)

[Hình 2.9. Biểu đồ tuần tự cơ chế xác thực và điều hướng của hệ thống 56](#_Toc195296288)

[Hình 2.10. Biểu đồ tuần tự hoạt động tham gia lớp học 57](#_Toc195296289)

[Hình 2.11. Biểu đồ tuần tự sinh viên làm bài và nộp bài 57](#_Toc195296290)

[Hình 2.12. Biểu đồ tuần tự Giảng viên tạo bài tập lập trình trong lớp học phần 58](#_Toc195296291)

[Hình 2.13. Biểu đồ tuần tự hoạt động Xem thông tin nộp bài của sinh viên 58](#_Toc195296292)

[Hình 2.14. Biểu đồ lớp của hệ thống 59](#_Toc195296293)

[Hình 2.15. Mô hình thực thể liên kết 62](#_Toc195296294)

[Hình 2.16. Mô hình dữ liệu quan hệ 68](#_Toc195296295)

[Hình 2.17. Wireframe giao diện trang chủ. Wireframe giao diện làm bài tập 69](#_Toc195296296)

[Hình 2.18. Wireframe giao diện làm bài tập 69](#_Toc195296297)

[Hình 2.19. Wireframe giao diện chung cho giảng viên 70](#_Toc195296298)

[Hình 2.20. Wireframe giao diện chung cho admin 70](#_Toc195296299)

[Hình 3.1. Giao diện trang chủ dành cho sinh viên 72](#_Toc195296300)

[Hình 3.2. Giao diện làm bài tập 72](#_Toc195296301)

[Hình 3.3. Giao diện làm bài tập và submit 73](#_Toc195296302)

[Hình 3.4. Diễn đàn cộng đồng sinh viên 73](#_Toc195296303)

[Hình 3.5. Cuộc trò chuyện chung 74](#_Toc195296304)

[Hình 3.6. Bảng xếp hạng 74](#_Toc195296305)

[Hình 3.7. Bài tập cần làm 75](#_Toc195296306)

[Hình 3.8. Quản lý bài tập 75](#_Toc195296307)

[Hình 3. 9. Quản lý nộp bài của sinh viên 76](#_Toc195296308)

[Hình 3.10. Quản lý nội dung nộp bài của sinh viên 76](#_Toc195296309)

[Hình 3.11. Quản lý sinh viên trong lớp 77](#_Toc195296310)

[Hình 3.12. Quản lý các lớp học phần 77](#_Toc195296311)

[Hình 3.13. Quản lý giảng viên 78](#_Toc195296312)

[Hình 3.14. Quản lý người dùng 78](#_Toc195296313)

[Hình 3.15. Thử nghiệm đăng nhập dành cho sinh viên 87](#_Toc195296314)

[Hình 3.16. Thử nghiệm nhập mã để vào lớp danh cho sinh viên 88](#_Toc195296315)

[Hình 3.17. Thử nghiệm nộp bài tập lập trình cho sinh viên 90](#_Toc195296316)

[Hình 3.18. Thử nghiệm thêm bài tập mới dành cho giảng viên 92](#_Toc195296317)

[Hình 3.19. Thử nghiệm thêm học phần dành cho Admin 94](#_Toc195296318)

[Hình 3.20. Thử nghiệm thêm lớp học phần dành cho Admin 94](#_Toc195296319)

# LỜI MỞ ĐẦU

Trong bối cảnh giáo dục hiện đại, đặc biệt là trong lĩnh vực công nghệ thông tin, việc đánh giá chất lượng học tập không chỉ dừng lại ở kiểm tra kiến thức lý thuyết, mà còn đòi hỏi tính chính xác, công bằng và hiệu quả trong việc đo lường năng lực thực hành. Đối với các môn học lập trình – một trong những trụ cột của chương trình đào tạo công nghệ thông tin, quá trình chấm bài theo phương pháp truyền thống thường bộc lộ nhiều hạn chế. Những vấn đề như tiêu tốn nhiều thời gian, dễ xảy ra sai sót do yếu tố chủ quan, và thiếu phản hồi kịp thời cho sinh viên đang là rào cản lớn trong việc nâng cao chất lượng giảng dạy.

Xuất phát từ thực tiễn đó, nhóm chúng em lựa chọn và triển khai đề tài ***“Xây dựng hệ thống quản lý và đánh giá bài tập lập”***. Hệ thống được phát triển nhằm hỗ trợ giảng viên trong công tác tổ chức, quản lý lớp học cũng như kiểm tra kết quả lập trình của sinh viên một cách khoa học, khách quan và tiết kiệm thời gian.

Thông qua hệ thống, sinh viên có thể nộp bài trực tuyến, hệ thống sẽ tự động biên dịch và thực thi mã nguồn, đối chiếu với bộ test case để kiểm tra tính chính xác của kết quả đầu ra – gọi ngắn gọn là “chấm bài tự động”. Công nghệ cốt lõi trong phần này là công cụ mã nguồn mở **Judge0**, cho phép xử lý đa ngôn ngữ và cung cấp phản hồi nhanh chóng. Đây không chỉ là một giải pháp kỹ thuật giúp giảng viên tiết kiệm thời gian kiểm tra bài làm, mà còn giúp sinh viên nhận được kết quả ngay lập tức, từ đó chủ động cải thiện kỹ năng lập trình của mình. Điều này còn đặc biệt hữu ích với các sinh viên mới tiếp xúc với lập trình, và còn chưa quen với việc thiết lập môi trường lập trình/công cụ biên dịch hoặc chưa có những công cụ đó trên máy tính cá nhân để làm việc.

Mục tiêu trọng tâm của đồ án là xây dựng một hệ thống học tập hiện đại, tổ chức các lớp luyện thực hành, giao bài tập, và đánh giá kết quả nộp bài một cách tự động. Hệ thống hướng đến khả năng thực thi và kiểm tra mã nguồn lập trình dựa trên đầu ra theo bộ test case được thiết kế sẵn, đồng thời mở rộng các tính năng giúp giảng viên theo dõi tiến độ, theo dõi kết quả làm bài của sinh viên, từ đó có những đánh giá và hỗ trợ sinh viên. Đối tượng sử dụng chính là sinh viên từ năm 2 đến năm 4. Phạm vi áp dụng tập trung vào các môn học lập trình cơ bản như C, C++, và Java, Python….

Khác với các hệ thống chấm bài trực tuyến hiện có trên thị trường, hệ thống được thiết kế để phù hợp với đặc thù giảng dạy tại Học viện Ngân hàng, đồng thời cho phép tùy chỉnh linh hoạt theo từng lớp học, từng bài tập. Hệ thống không chỉ đóng vai trò là một công cụ kiểm thử, mà còn hướng đến việc nâng cao trải nghiệm học lập trình cho sinh viên, thông qua cơ chế phản hồi tự động và các công cụ theo dõi tiến độ cá nhân.

Báo cáo của đồ án bao gồm các phần như sau

**Lời mở đầu**

**Chương 1**: Tổng quan về hệ thống quản lý và đánh giá bài tập lập trình – Trình bày tổng quan đề tài, phân tích thực trạng hiện tại, các công nghệ được sử dụng và kế hoạch triển khai.

**Chương 2**: Phân tích và thiết kế hệ thống – Mô tả yêu cầu hệ thống, các chức năng chính, và quy trình thiết kế kiến trúc phần mềm.

**Chương 3**: Triển khai và phát triển hệ thống – Trình bày chi tiết các bước xây dựng hệ thống, từ frontend đến backend, cùng với quá trình kiểm thử thực tế.

# CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG QUẢN LÝ VÀ ĐÁNH GIÁ BÀI TẬP LẬP TRÌNH

## 1.1. Giới thiệu

### 1.1.1. Khái niệm và mục đích

Hệ thống quản lý và đánh giá bài tập lập trình trực tuyến là một hệ thống công nghệ được thiết kế nhằm tự động hóa quá trình tổ chức, quản lý lớp học và đánh giá bài tập lập trình của sinh viên. Khác với các hệ thống chấm điểm trắc nghiệm đơn giản chỉ so sánh đáp án đúng sai, hệ thống này tập trung vào việc kiểm tra và đánh giá sâu hơn các khía cạnh đặc thù của lập trình, bao gồm cú pháp mã nguồn, logic thuật toán, khả năng thực thi và hiệu suất chương trình. Hiện nay, trên thế giới đã xuất hiện nhiều hệ thống tương tự, từ các hệ thống miễn phí như Codecademy hay HackerRank, đến các giải pháp trả phí như Gradescope hay Codio, phục vụ nhu cầu đa dạng từ kiểm tra cơ bản đến đánh giá chuyên sâu.

Tuy nhiên, trong môi trường giáo dục truyền thống, việc quản lý bài tập lập trình và chấm điểm vẫn chủ yếu được thực hiện thủ công. Giảng viên phải dành hàng giờ để đọc từng đoạn mã, phát hiện lỗi cú pháp, kiểm tra tính đúng đắn của logic, và thậm chí chạy thử chương trình trên các trình biên dịch để xác nhận kết quả. Quy trình này không chỉ tiêu tốn thời gian mà còn tiềm ẩn nguy cơ sai sót do yếu tố chủ quan, đặc biệt khi số lượng sinh viên lớn hoặc bài tập có độ phức tạp cao. Hơn nữa, việc cung cấp phản hồi chi tiết và kịp thời cho sinh viên – yếu tố then chốt để cải thiện kỹ năng lập trình – thường bị hạn chế bởi khối lượng công việc quá tải của giảng viên.

Nhằm khắc phục những bất cập trên, hệ thống quản lý và đánh giá bài tập lập trình trực tuyến được phát triển với các mục đích cốt lõi sau:

* **Tự động hóa quy trình đánh giá**: Giảm thiểu công sức và thời gian chấm bài thủ công cho giảng viên, đồng thời đảm bảo tiến độ giảng dạy được duy trì ổn định trong suốt học kỳ.
* **Cung cấp phản hồi tức thời**: Hỗ trợ **sinh** viên nhận diện lỗi sai ngay sau khi nộp bài, từ đó rút kinh nghiệm và nâng cao kỹ năng lập trình một cách hiệu quả.
* **Đảm bảo tính công bằng và khách quan**: Loại bỏ sai sót chủ quan trong quá trình chấm điểm, tạo ra một tiêu chuẩn đánh giá thống **nhất** và minh bạch cho tất cả sinh viên.
* **Hỗ trợ quản lý lớp học toàn diện**: Cho phép giảng viên tổ chức danh sách lớp học, giao bài tập, theo dõi tiến độ học tập và lưu trữ bài nộp của sinh viên trên một hệ thống tập trung.
* **Thúc đẩy giáo dục trực tuyến**: Phù hợp với xu hướng chuyển đổi số, hỗ trợ giảng dạy và học tập từ xa, đặc biệt trong bối cảnh các mô hình học tập linh hoạt ngày càng phổ biến.

Với những đặc điểm trên, hệ thống đóng vai trò như một công cụ hỗ trợ, là tiền đề để tối ưu hóa quy trình giảng dạy, đáp ứng nhu cầu thực tiễn và mở ra cơ hội phát triển các ứng dụng công nghệ sâu rộng hơn trong tương lai.

### 1.1.2. Quy trình hoạt động

Các hệ thống quản lý và đánh giá bài tập lập trình như Codeforces, LeetCode, HackerRank, CodeSignal hay hệ thống chấm bài tự động (Online Judge - OJ) tại các trường đại học hoạt động dựa trên những nguyên lý sau:

A yellow and green rectangular sign

AI-generated content may be incorrect.

Hình 1.1. Quy trình chấm bài lập trình tự động (Nguồn: tác giả)

**1. Người dùng gửi bài**

Quy trình bắt đầu khi người dùng gửi bài làm của mình lên hệ thống. Họ tải mã nguồn lên hệ thống và chọn ngôn ngữ lập trình phù hợp theo yêu cầu của bài toán. Một số hệ thống hỗ trợ nhiều ngôn ngữ như Python, C++, Java, JavaScript, v.v.

**2. Kiểm tra cú pháp và biên dịch**

Sau khi nhận được bài làm, hệ thống kiểm tra cú pháp và tiến hành biên dịch (nếu là ngôn ngữ biên dịch). Nếu mã nguồn có lỗi cú pháp, hệ thống sẽ ngay lập tức phản hồi với thông báo lỗi để người dùng chỉnh sửa. Nếu mã nguồn hợp lệ, chương trình sẽ được biên dịch thành tệp thực thi và chuyển sang bước tiếp theo.

**3. Thực thi với TestCase**

Chương trình sau đó được thực thi trong một môi trường cô lập (sandbox) để đảm bảo an toàn và tránh ảnh hưởng đến hệ thống chính. Nó sẽ chạy với một bộ dữ liệu kiểm thử (test cases) do hệ thống cung cấp. Mỗi bộ test sẽ chứa một tập hợp đầu vào cụ thể, và chương trình phải tạo ra đầu ra chính xác theo yêu cầu.

**4. Chấm điểm**

Tiếp theo, hệ thống tiến hành chấm điểm bài làm dựa trên nhiều tiêu chí khác nhau. Trước tiên, nó kiểm tra tính đúng đắn bằng cách so sánh kết quả đầu ra của chương trình với kết quả mong đợi. Nếu đầu ra sai, hệ thống trả về thông báo lỗi như **Wrong Answer (WA)**. Ngoài ra, hệ thống cũng đánh giá hiệu suất bằng cách đo thời gian chạy và lượng bộ nhớ tiêu thụ. Nếu chương trình chạy quá thời gian giới hạn, nó sẽ bị đánh dấu **Time Limit Exceeded (TLE)**, và nếu sử dụng quá nhiều bộ nhớ, nó sẽ bị lỗi **Memory Limit Exceeded (MLE)**. Trường hợp xảy ra lỗi trong quá trình thực thi như chia cho 0 hoặc truy cập bộ nhớ sai, hệ thống sẽ báo **Runtime Error (RE)**.

**5. Ghi nhận kết quả và thống kê**

Cuối cùng, kết quả của bài làm được ghi nhận vào cơ sở dữ liệu và hiển thị cho người dùng. Hệ thống có thể cung cấp phản hồi chi tiết về lỗi, thời gian thực thi và bộ nhớ sử dụng. Bên cạnh đó, các hệ thống còn cập nhật bảng xếp hạng, thống kê số bài làm đúng, điểm số của người dùng và gợi ý cải thiện. Một số hệ thống nâng cao còn sử dụng AI để đánh giá hiệu quả thuật toán hoặc phân tích lỗi để giúp người dùng học tập tốt hơn.

### 1.1.3. Các mô hình chấm bài phổ biến trong hệ thống đánh giá bài lập trình

Trong các hệ thống quản lý và đánh giá bài tập lập trình, nhiều mô hình chấm bài khác nhau được áp dụng tùy vào mục tiêu sử dụng. Dưới đây là một số mô hình phổ biến, cùng với nguồn tham khảo từ các hệ thống uy tín.

***Chấm bài tự động (Automatic Grading)***

Chấm bài tự động là mô hình phổ biến nhất, sử dụng một tập hợp các test case đầu vào và đầu ra mong đợi để kiểm tra tính đúng đắn của bài làm. Nếu kết quả đầu ra của chương trình khớp với kết quả mong đợi trong thời gian và bộ nhớ cho phép, bài làm sẽ được chấm điểm thành công. Theo nghiên cứu của Douce et al., (2005), phương pháp chấm bài tự động giúp cải thiện tốc độ đánh giá bài tập lập trình và giảm công sức của giảng viên.

**Ưu điểm**

* Đánh giá nhanh chóng và khách quan.
* Phù hợp với các bài tập có đầu ra xác định rõ ràng.

**Nhược điểm**

* Không đánh giá được phong cách lập trình hoặc hiệu quả thuật toán ngoài thời gian chạy.
* Không phù hợp với các bài tập có đầu ra mở (open-ended).

**Một số hệ thống sử dụng mô hình này như:** Hệ thống Codeforces, AtCoder, UVa Online Judge (UVa OJ) sử dụng mô hình này.

***Chấm bài theo mức điểm (Partial Grading)***

Trong một số trường hợp, nếu bài làm chỉ đúng một phần (đúng một số test case nhưng sai các test case khác), hệ thống sẽ tính điểm theo tỷ lệ phần trăm số test case đúng. Wilcox & Settle (2018) cho thấy rằng mô hình chấm điểm theo từng phần giúp cải thiện động lực học tập của sinh viên trong các khóa học lập trình.

**Ưu điểm**

* Công bằng hơn trong đánh giá, không bắt buộc bài làm phải đúng 100% mới có điểm.
* Giúp học viên có cơ hội sửa lỗi dần dần thay vì chỉ nhận kết quả "đúng" hoặc "sai".

**Nhược điểm**

* Có thể gây hiểu lầm nếu hệ thống không cung cấp đủ thông tin về lý do bài làm chưa đạt điểm tối đa.
* Không đo lường được hiệu quả thuật toán, chỉ dựa vào kết quả đầu ra.

**Một số hệ thống sử dụng mô hình này như:** HackerRank, CodeSignal, và hệ thống quản lý học tập Moodle thường sử dụng mô hình này.

***Chấm bài theo thuật toán nâng cao (AI-assisted Grading)***

Một số hệ thống sử dụng trí tuệ nhân tạo (AI) để đánh giá bài làm không chỉ dựa trên đầu ra mà còn dựa trên cấu trúc mã nguồn, độ phức tạp thuật toán, và phong cách lập trình. AI có thể phát hiện lỗi logic trong chương trình ngay cả khi đầu ra có vẻ đúng. Nguyen và cộng sự (2021) chỉ ra rằng AI có thể hỗ trợ giáo viên trong việc phát hiện lỗi lập trình phổ biến của sinh viên.

**Ưu điểm**

* Giúp đánh giá bài làm một cách toàn diện, không chỉ dựa vào đầu ra.
* Có thể phát hiện code trùng lặp hoặc sao chép (plagiarism detection).

**Nhược điểm**

* Đòi hỏi tài nguyên tính toán lớn và phức tạp trong triển khai.
* Có thể xảy ra lỗi đánh giá sai nếu thuật toán AI không đủ chính xác.

**Một số hệ thống sử dụng mô hình này như:** CodeSignal sử dụng AI để đánh giá khả năng lập trình của ứng viên tuyển dụng; EdX và Coursera đã tích hợp AI để phân tích bài làm của sinh viên. Miller và cộng sự (2019) cho thấy rằng chấm bài bằng hệ thống kiểm tra động giúp sinh viên tư duy tổng quát hơn khi lập trình.

**1.1.4. Tầm quan trọng của hệ thống quản lý đánh giá bài lập trình trực tuyến**

Trong bối cảnh giáo dục hiện đại, nơi công nghệ thông tin ngày càng đóng vai trò trung tâm, các hệ thống quản lý và đánh giá bài tập lập trình trực tuyến không chỉ là một công cụ hỗ trợ mà còn là một giải pháp chiến lược, mang lại giá trị to lớn cho cả giảng viên và sinh viên, đặc biệt trong các chương trình đào tạo chuyên. Sự ra đời và phát triển của các hệ thống này phản ánh xu hướng chuyển đổi số trong giáo dục, đáp ứng nhu cầu nâng cao hiệu quả giảng dạy và học tập trong kỷ nguyên số hóa.

Trước hết, hệ thống quản lý và đánh giá bài tập lập trình trực tuyến giúp **tự động hóa quy trình đánh giá**, một bước tiến vượt bậc so với phương pháp chấm bài thủ công truyền thống. Với các môn học lập trình, việc kiểm tra mã nguồn đòi hỏi phân tích cú pháp, logic và khả năng thực thi – những nhiệm vụ phức tạp và tốn thời gian nếu thực hiện bằng tay. Hệ thống tự động không chỉ giảm tải khối lượng công việc cho giảng viên mà còn đảm bảo tính chính xác và khách quan trong đánh giá, loại bỏ sai sót chủ quan và tạo ra một tiêu chuẩn thống nhất cho toàn bộ sinh viên. Điều này đặc biệt quan trọng trong môi trường đào tạo quy mô lớn như Học viện Ngân hàng, nơi số lượng sinh viên đông đảo đòi hỏi sự quản lý hiệu quả và công bằng.

Thứ hai, hệ thống mang lại **phản hồi tức thời**, một yếu tố then chốt trong việc nâng cao kỹ năng lập trình của sinh viên. Khác với phương pháp truyền thống, nơi sinh viên có thể phải chờ hàng ngày hoặc hàng tuần để nhận kết quả, hệ thống trực tuyến cung cấp thông tin chi tiết về lỗi sai, hiệu suất mã nguồn và gợi ý cải thiện ngay sau khi nộp bài. Điều này không chỉ giúp sinh viên nhanh chóng rút kinh nghiệm mà còn thúc đẩy quá trình học tập chủ động, khuyến khích họ thử nghiệm và hoàn thiện kỹ năng một cách hiệu quả hơn. Trong bối cảnh đào tạo lập trình – một lĩnh vực đòi hỏi thực hành liên tục – khả năng phản hồi nhanh chóng là yếu tố tạo nên sự khác biệt giữa học lý thuyết và phát triển năng lực thực tiễn.

Thứ ba, hệ thống hỗ trợ **quản lý lớp học toàn diện**, từ việc tổ chức danh sách sinh viên, phân phối bài tập, đến theo dõi tiến độ học tập. Với giảng viên, đây là công cụ giúp giảm thiểu các tác vụ hành chính tẻ nhạt, cho phép họ tập trung vào việc giảng dạy và hỗ trợ sinh viên.

Cuối cùng, tầm quan trọng của hệ thống còn nằm ở khả năng **thúc đẩy giáo dục trực tuyến và học tập từ xa**, một xu hướng không thể đảo ngược trong thời đại toàn cầu hóa. Hệ thống cho phép sinh viên truy cập bài tập, nộp bài và nhận kết quả từ bất kỳ đâu, phá vỡ rào cản về không gian và thời gian. Trong bối cảnh các mô hình học tập linh hoạt ngày càng được ưa chuộng, đặc biệt sau những thay đổi do đại dịch COVID-19, hệ thống trực tuyến không chỉ hỗ trợ giảng dạy mà còn góp phần xây dựng một hệ sinh thái học tập số hóa bền vững, đáp ứng nhu cầu đào tạo hiện đại tại Học viện Ngân hàng.

## 1.2. Thực trạng hệ thống đánh giá bài tập lập trình trực tuyến tại các trường đại học trên thế giới và trong nước

Hầu hết các trường đại học trong và ngoài nước đã và đang triển khai các hệ thống chấm đánh giá bài tập lập trình trực tuyến nhằm nâng cao chất lượng giảng dạy và kiểm tra năng lực sinh viên một cách khách quan, chính xác. Những hệ thống này không chỉ giúp giảng viên giảm tải công việc chấm điểm mà còn cung cấp phản hồi nhanh chóng, hỗ trợ sinh viên tự học và rèn luyện kỹ năng lập trình hiệu quả. Bên cạnh đó, việc áp dụng các hệ thống đánh giá tự động còn góp phần chuẩn hóa tiêu chí đánh giá, đảm bảo tính công bằng và minh bạch trong quá trình kiểm tra. Tuy nhiên, mỗi trường có cách tiếp cận khác nhau, tùy thuộc vào điều kiện cơ sở hạ tầng, chương trình đào tạo và nhu cầu thực tế.

### 1.2.2. Các trường đại học trong nước

Tại Việt Nam, các trường đại học và cộng đồng công nghệ thông tin đã tích cực phát triển các hệ thống chấm bài lập trình trực tuyến nhằm nâng cao chất lượng giảng dạy, hỗ trợ thực hành và chuẩn bị cho sinh viên tham gia các cuộc thi lớn như Olympic Tin học Sinh viên và ACM-ICPC. Những nền tảng này không chỉ phục vụ việc kiểm tra, đánh giá trong môi trường học thuật mà còn mở rộng phạm vi, trở thành công cụ quan trọng để rèn luyện kỹ năng lập trình cho cả sinh viên và cộng đồng lập trình viên trong nước. Dưới đây là mô tả chi tiết về ba hệ thống tiêu biểu:

#### 1.2.2.1. Học viện Công nghệ Bưu chính Viễn thông (PTIT)

Học viện Công nghệ Bưu chính Viễn thông (PTIT) tận dụng nền tảng **SPOJ (Sphere Online Judge)** với một không gian riêng biệt dành cho sinh viên và giảng viên của trường để triển khai hệ thống chấm bài lập trình trực tuyến. Đây là một hệ thống phổ biến trên thế giới, được PTIT tùy chỉnh để phục vụ các mục đích giảng dạy và thi đấu nội bộ. Sinh viên có thể sử dụng các ngôn ngữ lập trình như C, C++, Java, Python, Pascal để nộp bài tập hoặc tham gia các bài kiểm tra, với kết quả được chấm tự động dựa trên các bộ test case được chuẩn bị kỹ lưỡng. Hệ thống này được tích hợp trong các học phần như lập trình cơ bản, cấu trúc dữ liệu và thuật toán, đồng thời hỗ trợ tổ chức các cuộc thi nội bộ nhằm giúp sinh viên làm quen với định dạng thi đấu chuẩn quốc tế như ACM-ICPC. Điểm nổi bật của SPOJ tại PTIT là kho bài tập phong phú, bao gồm hàng trăm bài từ cấp độ cơ bản dành cho người mới học đến các bài nâng cao thách thức khả năng tư duy thuật toán, cùng với cơ chế chấm điểm tức thì giúp sinh viên nhanh chóng nhận biết lỗi sai và cải thiện. (Sphere Online Judge)

#### 1.2.2.2. Trường Đại học FPT

Trường Đại học FPT xây dựng **CodeLearn**, một nền tảng học tập và thi đấu lập trình trực tuyến toàn diện, không chỉ phục vụ sinh viên trong trường mà còn hướng tới cộng đồng lập trình viên rộng lớn tại Việt Nam. CodeLearn vượt xa vai trò của một hệ thống chấm bài thông thường, trở thành một môi trường học tập tích hợp với các khóa học đa dạng, từ lập trình cơ bản (như C++, Python) đến các chủ đề nâng cao như thuật toán, cấu trúc dữ liệu và lập trình ứng dụng. Hệ thống hỗ trợ nhiều ngôn ngữ lập trình phổ biến, bao gồm C++, Java, Python, JavaScript và một số ngôn ngữ khác, với khả năng đánh giá không chỉ kết quả đầu ra mà còn phân tích hiệu suất mã nguồn như độ phức tạp thời gian và không gian. Sau mỗi lần nộp bài, sinh viên nhận được phản hồi chi tiết bao gồm thông báo lỗi, gợi ý cải thiện và điểm số cụ thể, giúp họ tối ưu hóa thuật toán và nâng cao kỹ năng viết mã. CodeLearn còn tổ chức các cuộc thi lập trình định kỳ, cung cấp bảng xếp hạng và chứng chỉ cho người tham gia, tạo động lực học tập và cạnh tranh. Điểm mạnh của hệ thống nằm ở sự kết hợp giữa giáo dục và thực hành thực tế, phù hợp cho cả người mới bắt đầu và lập trình viên có kinh nghiệm. (FPT Techinsight)

## 1.3. Các thành phần chính của hệ thống

Các hệ thống đánh giá bài tập lập trình tự động nói chung được thiết kế để hỗ trợ quá trình giảng dạy, học tập và quản lý trong lĩnh vực lập trình. Dù có sự khác biệt về mục đích sử dụng (giáo dục, thi đấu, hay tuyển dụng), các hệ thống này thường bao gồm một số thành phần cốt lõi nhằm đảm bảo tính hiệu quả, tiện lợi và chính xác trong việc tổ chức, giao bài, thực thi và đánh giá mã nguồn. Dưới đây là các thành phần chính thường thấy trong các hệ thống này:

### 1.3.1. Thành phần quản lý người dùng

Hệ thống cần có cơ chế quản lý thông tin và quyền truy cập của các nhóm người dùng khác nhau, thường bao gồm quản trị viên, giảng viên (hoặc người ra đề), và sinh viên (hoặc người tham gia).

* **Quản trị viên:** Chịu trách nhiệm quản lý toàn bộ hệ thống, bao gồm tạo và phân quyền tài khoản, giám sát hoạt động, và duy trì cơ sở hạ tầng.
* **Giảng viên/Người ra đề:** Có khả năng tạo bài tập, theo dõi kết quả, và quản lý danh sách người tham gia (ví dụ: sinh viên trong lớp học hoặc thí sinh trong cuộc thi).
* **Sinh viên/Người dùng:** Đăng ký, đăng nhập để truy cập bài tập, nộp bài, và nhận phản hồi từ hệ thống. Thành phần này thường tích hợp các chức năng như xác thực (authentication), phân quyền (authorization), và quản lý hồ sơ người dùng để đảm bảo tính bảo mật và cá nhân hóa trải nghiệm.

### 1.3.2. Thành phần quản lý bài tập

Đây là trung tâm của hệ thống, cho phép tạo, lưu trữ và phân phối bài tập lập trình.

* **Tạo bài tập:** Người có quyền (quản trị viên hoặc giảng viên) định nghĩa bài tập với các yếu tố như đề bài, bộ kiểm thử (test cases), thời hạn, và tiêu chí đánh giá (đúng/sai, hiệu suất).
* **Phân loại và tái sử dụng:** Bài tập có thể được phân loại theo chủ đề (thuật toán, cấu trúc dữ liệu) hoặc mức độ khó, đồng thời cho phép tái sử dụng giữa các lớp học hoặc khóa học.
* **Quản lý thời gian:** Hệ thống hỗ trợ đặt thời hạn nộp bài và theo dõi trạng thái (đã giao, đã nộp, quá hạn). Thành phần này đảm bảo bài tập được tổ chức khoa học, dễ dàng truy cập và phù hợp với mục tiêu học tập hoặc đánh giá.

### 1.3.3. Thành phần thực thi và đánh giá mã nguồn

Đây là lõi kỹ thuật của hệ thống, chịu trách nhiệm biên dịch, chạy mã nguồn và đánh giá kết quả.

* **Biên dịch và thực thi:** Hệ thống nhận mã nguồn từ người dùng, biên dịch (nếu cần) và chạy trong môi trường an toàn (sandbox) để tránh rủi ro bảo mật.
* **Đánh giá tự động:** So sánh đầu ra của mã nguồn với đầu ra mong đợi từ test cases, đồng thời đo lường hiệu suất (thời gian chạy, bộ nhớ sử dụng).
* **Phản hồi chi tiết:** Cung cấp thông tin về lỗi (cú pháp, logic, thời gian vượt quá giới hạn) và điểm số để người dùng cải thiện. Thành phần này thường sử dụng các công cụ như Judge0, Docker, hoặc các engine biên dịch nội bộ để hỗ trợ đa ngôn ngữ lập trình (C, C++, Java, Python, v.v.).

### 1.3.4. Thành phần giao tiếp và cộng đồng

Nhiều hệ thống tích hợp các kênh giao tiếp để tăng tính tương tác giữa người dùng.

* **Trò chuyện nhóm:** Cho phép người tham gia (sinh viên, giảng viên) trao đổi trực tiếp trong phạm vi lớp học hoặc nhóm.
* **Diễn đàn cộng đồng:** Cung cấp không gian thảo luận mở, nơi người dùng chia sẻ giải pháp, đặt câu hỏi hoặc học hỏi lẫn nhau.
* **Thông báo:** Hệ thống gửi thông tin về bài tập mới, thời hạn, hoặc kết quả qua email hoặc giao diện. Thành phần này thúc đẩy sự hợp tác và hỗ trợ lẫn nhau, đặc biệt trong môi trường giáo dục.

### 1.3.5. Thành phần theo dõi và báo cáo

* Hệ thống cung cấp công cụ để theo dõi tiến độ **và** phân tích kết quả.
* **Theo dõi cá nhân:** Người dùng (sinh viên) xem lịch sử nộp bài, điểm số, và phản hồi chi tiết.
* **Theo dõi nhóm:** Giảng viên hoặc quản trị viên xem thống kê lớp học, tỷ lệ hoàn thành bài tập, và hiệu suất trung bình của sinh viên.
* **Báo cáo tổng quan:** Hệ thống tạo báo cáo về hoạt động (số lượng bài nộp, lỗi phổ biến) để hỗ trợ cải thiện giảng dạy hoặc hệ thống. Thành phần này giúp cả người học và người dạy đánh giá hiệu quả học tập một cách toàn diện.

### 1.3.6. Giao diện và khả năng truy cập

Hệ thống thường được triển khai dưới dạng ứng dụng web hoặc ứng dụng đa hệ thống, với giao diện thân thiện và khả năng truy cập linh hoạt.

* **Giao diện người dùng:** Được thiết kế trực quan, hỗ trợ nhập mã nguồn, xem đề bài, và nhận kết quả trên cùng một màn hình.
* **Tương thích đa thiết bị:** Hoạt động mượt mà trên máy tính, điện thoại thông minh, và máy tính bảng, đảm bảo người dùng có thể sử dụng mọi lúc, mọi nơi với kết nối internet.
* **Hỗ trợ đa ngôn ngữ:** Một số hệ thống cung cấp giao diện và tài liệu bằng nhiều ngôn ngữ để phục vụ người dùng toàn cầu. Thành phần này đảm bảo trải nghiệm người dùng tối ưu và khả năng tiếp cận rộng rãi.

### 1.3.7. Thành phần bảo mật và lưu trữ

Để duy trì tính tin cậy và an toàn, hệ thống cần có các cơ chế bảo vệ dữ liệu và mã nguồn.

* **Bảo mật:** Sử dụng mã hóa, xác thực phiên (session), và môi trường thực thi cô lập để ngăn chặn truy cập trái phép hoặc tấn công.
* **Lưu trữ:** Cơ sở dữ liệu lưu thông tin người dùng, bài tập, bài nộp, và kết quả, thường sử dụng các hệ quản trị như MySQL, PostgreSQL hoặc MongoDB.
* **Sao lưu:** Đảm bảo dữ liệu được sao lưu định kỳ để tránh mất mát trong trường hợp sự cố. Thành phần này là hệ thống để hệ thống hoạt động ổn định và đáng tin cậy trong thời gian dài.

## 1.4. Bài toán quản lý và bài tập lập trình cho sinh viên - giảng viên

### 1.4.1. Phát biểu bài toán

### 1.4.2. Mục đích, mục tiêu

* **Tự động hóa đánh giá**: Xây dựng hệ thống tự động chấm điểm bài tập lập trình dựa trên các tiêu chí đánh giá rõ ràng (độ chính xác, hiệu suất, cú pháp), giảm thiểu công việc thủ công cho giảng viên.
* **Cung cấp phản hồi tức thời**: Đảm bảo sinh viên nhận được kết quả và gợi ý cải thiện ngay sau khi nộp bài, từ đó hỗ trợ quá trình học tập chủ động và liên tục.
* **Quản lý tập trung**: Tích hợp các chức năng quản lý lớp học, bài tập và tiến độ học tập trên một hệ thống duy nhất, giúp giảng viên dễ dàng theo dõi và điều chỉnh phương pháp giảng dạy.
* **Hỗ trợ đa dạng ngôn ngữ lập trình**: Đáp ứng nhu cầu giảng dạy các ngôn ngữ phổ biến như C, C++, Java, Python.
* **Tăng cường tính công bằng và minh bạch**: Chuẩn hóa quy trình đánh giá, đảm bảo kết quả phản ánh chính xác năng lực của sinh viên, loại bỏ yếu tố chủ quan.

### 1.4.3. Phạm vi

* **Giảng viên**: Sử dụng hệ thống để tạo bài tập, thiết lập bộ kiểm thử, chấm điểm tự động, quản lý lớp học và theo dõi tiến độ của sinh viên.
* **Sinh viên**: Truy cập hệ thống để nộp bài tập, nhận kết quả, phản hồi và cải thiện kỹ năng lập trình thông qua các bài tập thực hành.
* **Quản trị viên**: Quản lý hệ thống, đảm bảo vận hành ổn định, cập nhật tài nguyên và hỗ trợ kỹ thuật khi cần.

**Phạm vi chức năng**:

* **Quản lý bài tập**: Tạo, chỉnh sửa và phân phối bài tập lập trình cho sinh viên.
* **Đánh giá tự động**: Chấm điểm dựa trên bộ test cases được định nghĩa trước, hỗ trợ nhiều ngôn ngữ lập trình (C, C++, Java, Python).
* **Phản hồi và báo cáo**: Cung cấp kết quả chi tiết (điểm số, lỗi sai, gợi ý) và báo cáo tiến độ học tập cho cả sinh viên và giảng viên.
* **Tích hợp**: Kết nối với hệ thống quản lý học tập hiện có của Học viện (nếu khả thi) để đồng bộ dữ liệu sinh viên và lớp học.
* **Bảo mật**: Đảm bảo an toàn dữ liệu bài nộp, thông tin cá nhân và kết quả đánh giá của sinh viên.

### 1.4.4. Giải pháp đề xuất và công nghệ sử dụng

Để phát triển một hệ thống trực tuyến quản lý lớp học và đánh giá bài tập lập trình đáp ứng các yêu cầu về hiệu suất, khả năng mở rộng và bảo trì lâu dài, nhóm đã lựa chọn các công nghệ hiện đại, tiên tiến, phù hợp với xu hướng phát triển phần mềm đương đại và đặc thù của đề tài. Hệ thống được thiết kế theo kiến trúc phân tách rõ ràng giữa **Frontend** (giao diện người dùng) và **Backend** (xử lý logic và dữ liệu), giao tiếp thông qua **RESTful API** – một phương thức truyền tải dữ liệu nhẹ, hiệu quả và phổ biến trong các ứng dụng web hiện nay. Thiết kế này không chỉ tăng tính module hóa, cho phép phát triển và nâng cấp từng thành phần độc lập, mà còn đảm bảo sự linh hoạt trong việc tích hợp các tính năng mới trong tương lai, đáp ứng nhu cầu quản lý lớp học và đánh giá bài tập.

#### 1.4.4.1. Frontend: Next.js

Phần giao diện người dùng được phát triển bằng **Next.js**, một framework mạnh mẽ xây dựng trên hệ thống **React**. Next.js được lựa chọn nhờ khả năng hỗ trợ **Server-Side Rendering (SSR)** và **Static Site Generation (SSG)**, giúp tối ưu hóa tốc độ tải trang bằng cách xử lý nội dung ngay tại máy chủ trước khi gửi đến trình duyệt. Điều này mang lại trải nghiệm người dùng mượt mà, đặc biệt quan trọng đối với sinh viên và giảng viên khi truy cập hệ thống từ nhiều thiết bị khác nhau. Ngoài ra, Next.js tích hợp sẵn các công cụ tối ưu hóa công cụ tìm kiếm (**SEO**) và hỗ trợ phát triển giao diện responsive, đảm bảo khả năng tương thích trên các hệ thống như máy tính để bàn, máy tính bảng và điện thoại thông minh. Với cấu trúc component hóa của React, Next.js cho phép nhóm xây dựng giao diện trực quan, dễ sử dụng, đồng thời giảm thiểu thời gian phát triển nhờ hệ sinh thái phong phú và cộng đồng hỗ trợ rộng lớn. Đây là lựa chọn lý tưởng để đáp ứng nhu cầu truy cập đa dạng của người dùng trong môi trường học tập trực tuyến.

#### 1.4.4.2. Backend: Laravel

Ở phía xử lý logic và dữ liệu, nhóm sử dụng **Laravel** – một framework PHP hiện đại nổi bật với cú pháp rõ ràng, khả năng phát triển nhanh và hiệu quả cao trong việc xây dựng các ứng dụng web phức tạp. Laravel được lựa chọn nhờ hệ sinh thái mạnh mẽ, đặc biệt là các tính năng như **routing linh hoạt**, **middleware mạnh mẽ**, và **hỗ trợ nhanh chóng RESTful API**, giúp tăng tốc độ phát triển backend. Laravel đồng thời cung cấp giải pháp toàn diện cho các chức năng quan trọng của hệ thống, bao gồm **xác thực người dùng**, **phân quyền truy cập**, và **xử lý bài nộp** từ sinh viên. Nhờ **Eloquent ORM**, việc tương tác với cơ sở dữ liệu trở nên trực quan và dễ dàng, hỗ trợ hiệu quả cho việc quản lý thông tin phức tạp như lớp học, bài tập, tiến trình học tập và kết quả đánh giá. Về bảo mật, Laravel tích hợp sẵn nhiều cơ chế bảo vệ khỏi các lỗ hổng phổ biến như **CSRF**, **XSS**, và hỗ trợ **mã hóa dữ liệu nhạy cảm**, đảm bảo an toàn cho thông tin người dùng trong toàn hệ thống. Ngoài ra, khả năng mở rộng và tối ưu hiệu suất thông qua **caching**, **queue**, và **xử lý bất đồng bộ** giúp Laravel vận hành ổn định, ngay cả khi phải xử lý khối lượng lớn bài nộp trong các lớp học đông sinh viên – một yêu cầu thực tế tại Học viện Ngân hàng.

#### 1.4.4.4. Judge0 API

Chức năng cốt lõi của hệ thống – biên dịch và đánh giá mã nguồn lập trình – được thực hiện thông qua tích hợp **Judge0**, một API mã nguồn mở mạnh mẽ hỗ trợ hơn 40 ngôn ngữ lập trình phổ biến như C, C++, Java, Python, JavaScript và nhiều ngôn ngữ khác. Judge0 hoạt động như một engine xử lý độc lập, nhận mã nguồn từ hệ thống qua các yêu cầu HTTP, thực thi mã với các bộ **test case** được định nghĩa trước, sau đó trả về kết quả chi tiết bao gồm đầu ra, thời gian thực thi, dung lượng bộ nhớ sử dụng và thông báo lỗi (nếu có). Hệ thống gửi bài tập lập trình của sinh viên đến Judge0, nhận kết quả thực thi và so sánh với đầu ra mong đợi để đưa ra đánh giá chính xác về tính đúng đắn và hiệu suất. Việc sử dụng Judge0 thay vì tự phát triển một engine biên dịch nội bộ giúp tiết kiệm thời gian, giảm tải cho Backend và tận dụng được độ tin cậy của một dịch vụ đã được cộng đồng quốc tế kiểm chứng. Hơn nữa, tính linh hoạt của Judge0 cho phép dễ dàng mở rộng hỗ trợ thêm các ngôn ngữ lập trình trong tương lai.

**1.4.4.6. Cơ sở dữ liệu**

Hệ thống sử dụng MySQL – một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở phổ biến – để lưu trữ toàn bộ thông tin cần thiết, bao gồm dữ liệu người dùng (tài khoản sinh viên và giảng viên), bài tập (nội dung, test case, yêu cầu), và kết quả chấm điểm (điểm số, thời gian nộp bài). MySQL được chọn nhờ hiệu suất cao trong việc xử lý các truy vấn phức tạp, khả năng đảm bảo tính toàn vẹn dữ liệu thông qua các ràng buộc khóa ngoại và giao dịch, cũng như sự phổ biến trong cộng đồng lập trình, giúp nhóm dễ dàng tìm kiếm tài liệu và hỗ trợ khi cần. Ngoài ra, nhóm cũng cân nhắc PostgreSQL như một phương án thay thế, với ưu điểm về hỗ trợ các kiểu dữ liệu nâng cao và khả năng xử lý khối lượng dữ liệu lớn, nhưng hiện tại MySQL được ưu tiên vì tính đơn giản và phù hợp với quy mô ban đầu của hệ thống. Cả hai cơ sở dữ liệu đều đảm bảo tốc độ truy vấn nhanh chóng, đáp ứng tốt nhu cầu tra cứu kết quả hoặc thống kê tiến độ học tập từ phía giảng viên.

## 1.5. Kế hoạch thực hiện dự án

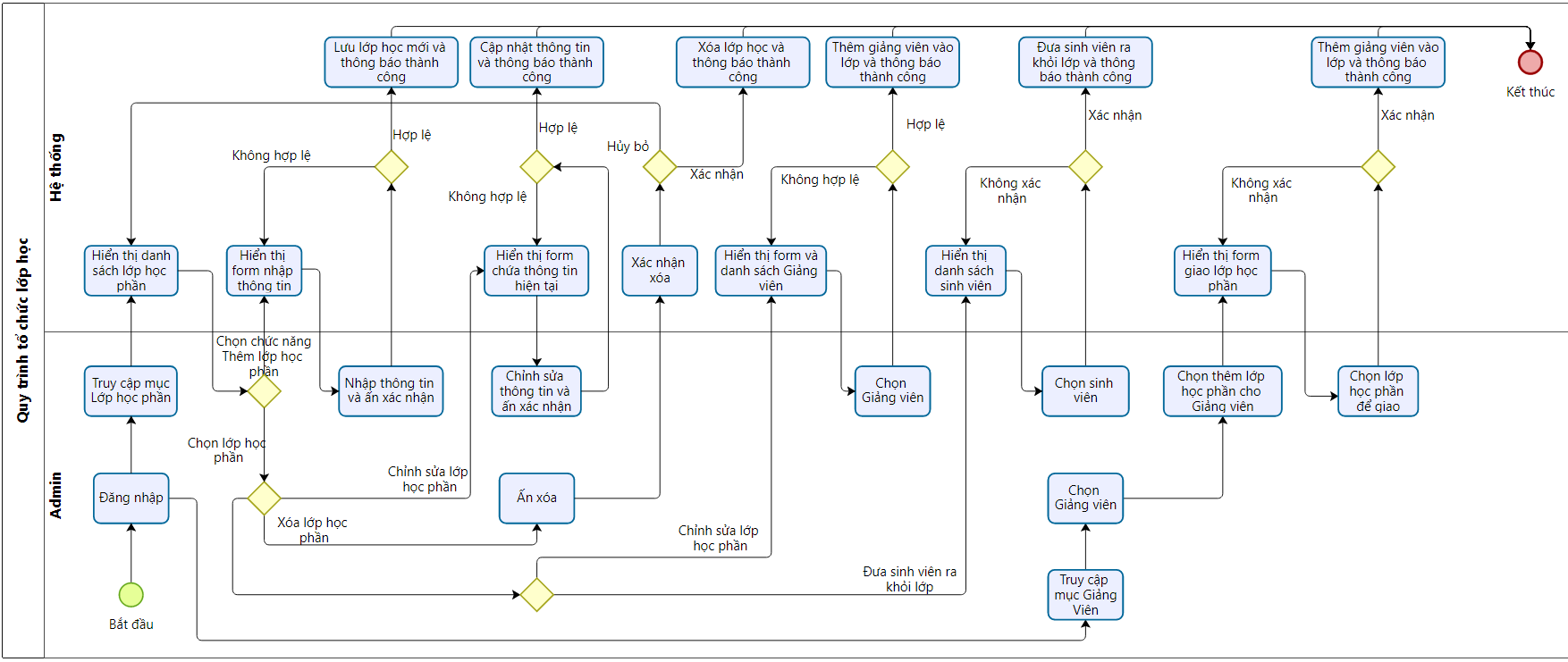
Thời gian thực hiện từ 5/1/2025- 10/03/2025

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Giai đoạn/ Công việc | Mô tả chi tiết | Thời gian bắt đầu | Thời gian kết thúc | Người phụ trách |
|  | Khởi động dự án | Xác định mục tiêu, phạm vi, lập kế hoạch tổng quan |  |  |  |
|  | Phân tích yêu cầu | Thu thập và phân tích yêu cầu của dự án |  |  |  |
|  | Thiết kế hệ thống | Thiết kế kiến trúc, cơ sở dữ liệu, giao diện |  |  |  |
|  | Phát triển | Lập trình, tích hợp các thành phần |  |  |  |
|  | Kiểm thử | Kiểm tra lỗi, tối ưu hiệu suất |  |  |  |
|  | Đánh giá và bảo trì | Tổng kết, cải thiện và bảo trì hệ thống |  |  |  |

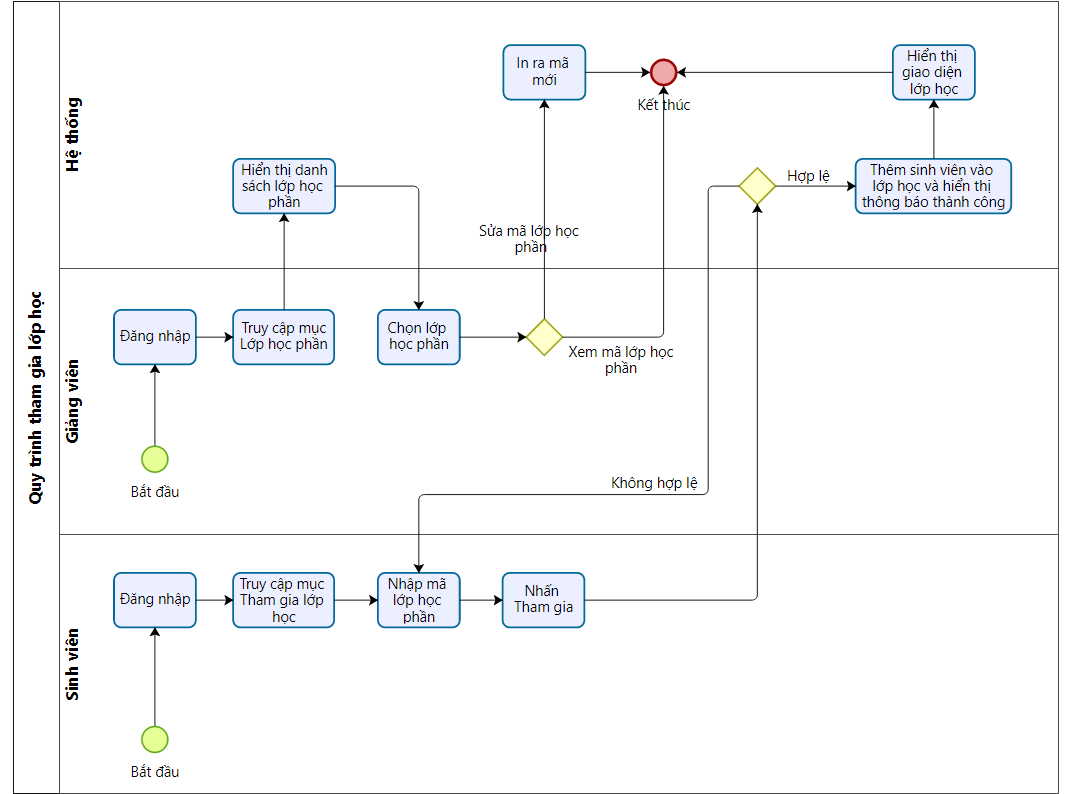
Bảng 1.3. Bảng kế hoạch thực hiện dự án

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

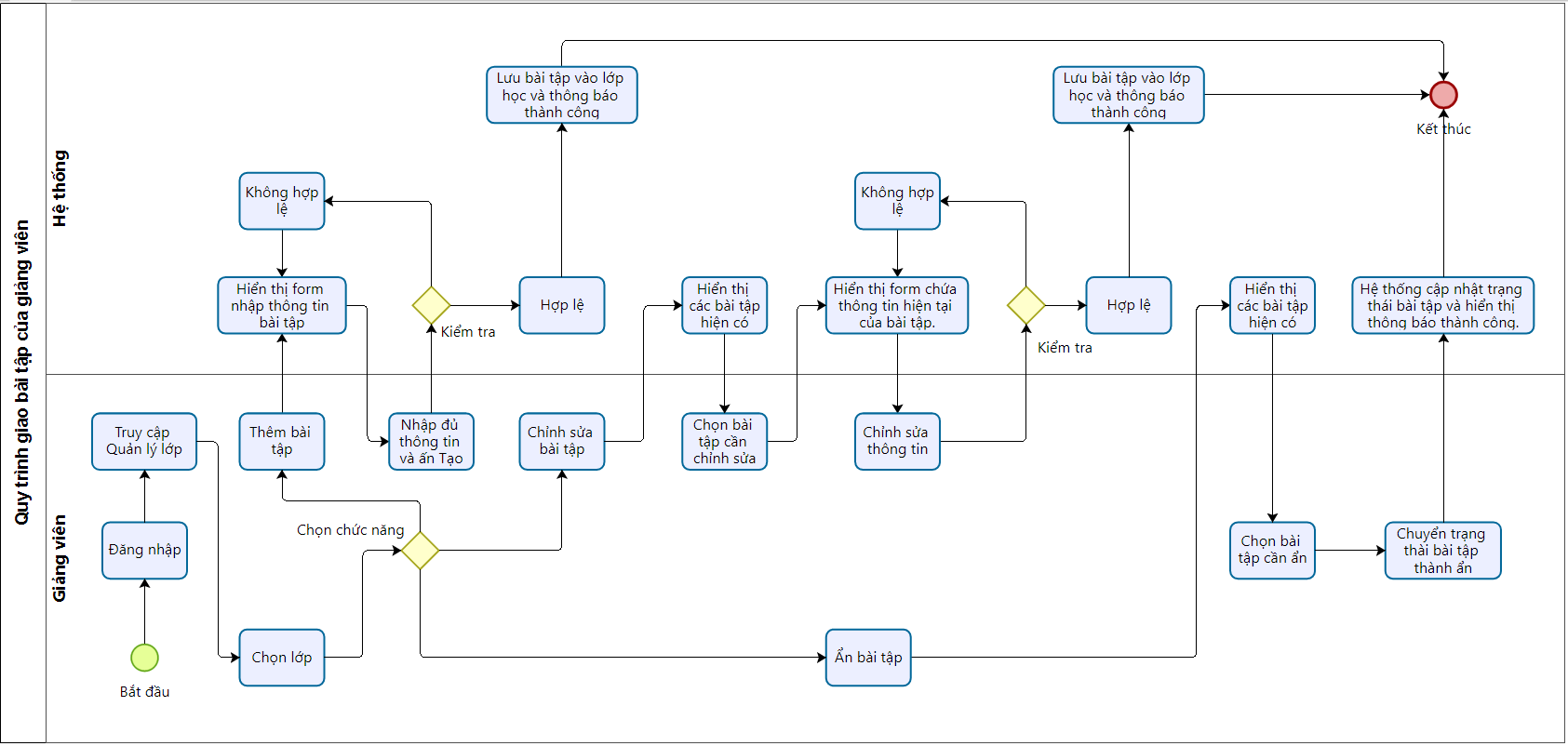
## 2.1. Mô hình nghiệp vụ của hệ thống



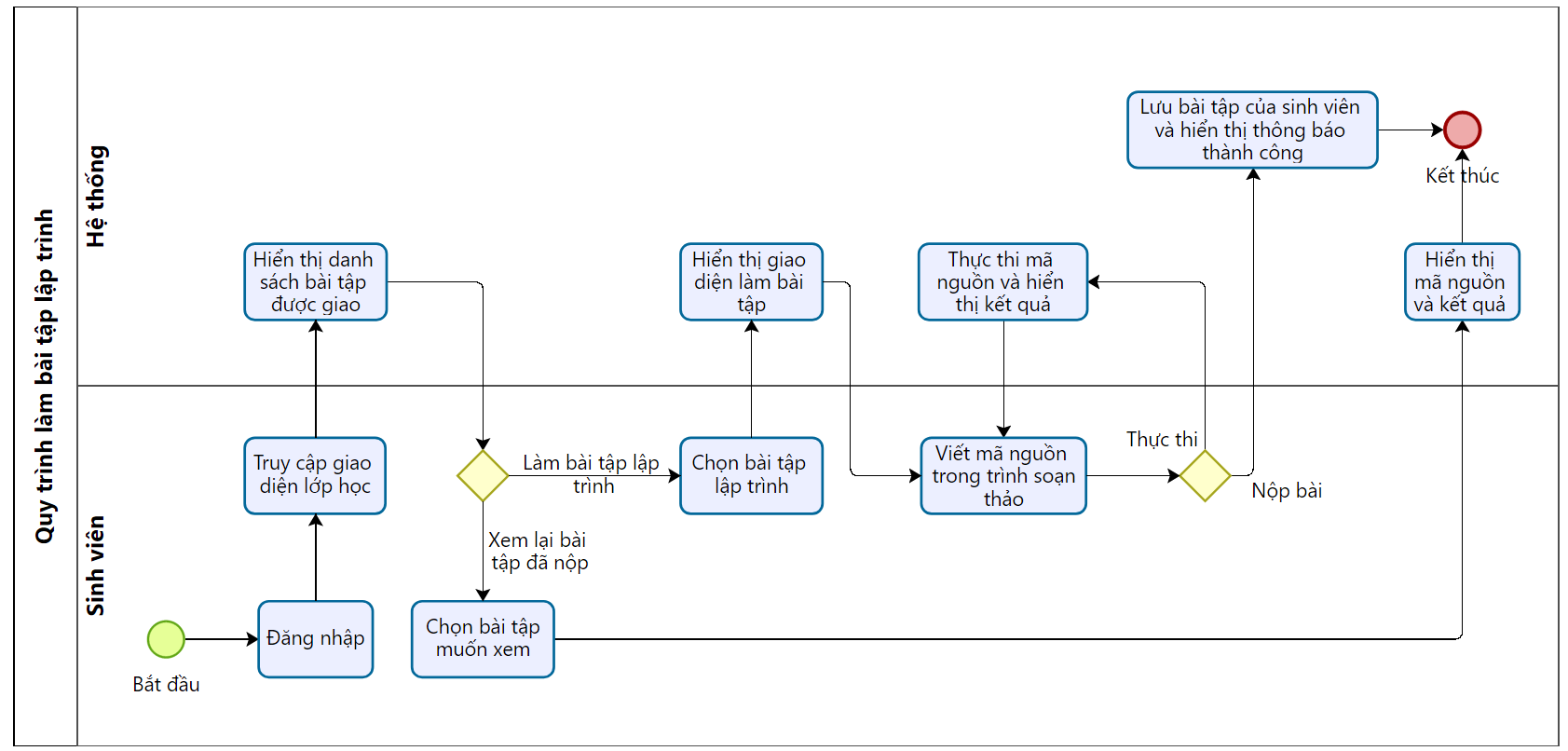
Hình 2.1. Quy trình tổ chức lớp học



Hình 2.2. Quy trình tham gia lớp học



Hình 2.3. Quy trình giao bài tập của giảng viên



Hình 2.4. Quy trình làm bài tập lập trình

## 2.2. Phân tích yêu cầu hệ thống

### 2.2.1. Xác định các ca sử dụng

##### a. Ca sử dụng cho Admin

|  |  |
| --- | --- |
| **Ca sử dụng cho Admin** | **Mô tả** |
| Đăng nhập | Amin đăng nhập vào hệ thống |
| Tạo người dùng mới | Admin tạo người dùng mới vào hệ thống và phân quyền truy cập cho người dùng |
| Chỉnh sửa thông tin người dùng | Admin chỉnh sửa thông tin, quyền truy cập của người dùng |
| Xóa người dùng | Admin xóa người dùng ra khỏi hệ thống |
| Tạo lớp học mới | Admin tạo lớp học mới vào hệ thống |
| Chỉnh sửa thông tin lớp học | Admin chỉnh sửa thông tin lớp học |
| Xóa lớp học | Admin xóa lớp học ra khỏi hệ thống |
| Thêm người dùng vào lớp học | Admin thêm người dùng vào lớp học |
| Đưa người dùng ra khỏi lớp học | Admin đưa người dùng ra khỏi lớp học |
| Tạo bài tập lập trình cố định | Admin tạo mới bài tập lập trình cố định theo học phần gồm đề bài và test case |
| Chỉnh sửa bài tập lập trình cố định | Admin chỉnh sửa bài tập lập trình cố định theo học phần gồm đề bài và test case |
| Xóa bài tập lập trình cố định | Admin xóa bài tập lập trình cố định theo học phần ra khỏi hệ thống |
| Tạo bài tập lập trình cho lớp học | Admin tạo bài tập lập trình bổ sung cho lớp học |
| Chỉnh sửa bài tập lập trình cho lớp học | Admin chỉnh sửa bài tập lập trình bổ sung cho lớp học |
| Xóa bài tập lập trình cho lớp học | Admin xóa bài tập lập trình bổ sung cho lớp học |

Bảng 2.1. Bảng các ca sử dụng cho tác nhân Admin

##### b. Ca sử dụng cho Giảng viên

|  |  |
| --- | --- |
| **Ca sử dụng cho Giảng viên** | **Mô tả** |
| Đăng nhập | Giảng viên đăng nhập vào hệ thống |
| Tạo bài tập lập trình cho lớp học | Giảng viên tạo bài tập lập trình bổ sung cho lớp học |
| Chỉnh sửa bài tập lập trình cho lớp học | Giảng viên chỉnh sửa bài tập lập trình bổ sung cho lớp học |
| Xóa bài tập lập trình cho lớp học | Giảng viên xóa bài tập lập trình bổ sung cho lớp học |
| Đưa sinh viên ra khỏi lớp học | Giảng viên đưa sinh viên ra khởi lớp học |
| Theo dõi tiến độ làm bài tập của sinh viên | Giảng viên xem tiến độ làm bài tập của sinh viên: câu nào đã làm, câu nào chưa làm,... |
| Theo dõi kết quả làm bài của sinh viên | Giảng viên xem kết quả làm bài của sinh viên |
| Xem lịch sử lập trình của sinh viên | Giảng viên xem lại đoạn mã mà sinh viên đã lập trình |

Bảng 2.2. Bảng các ca sử dụng cho tác nhân Giảng viên

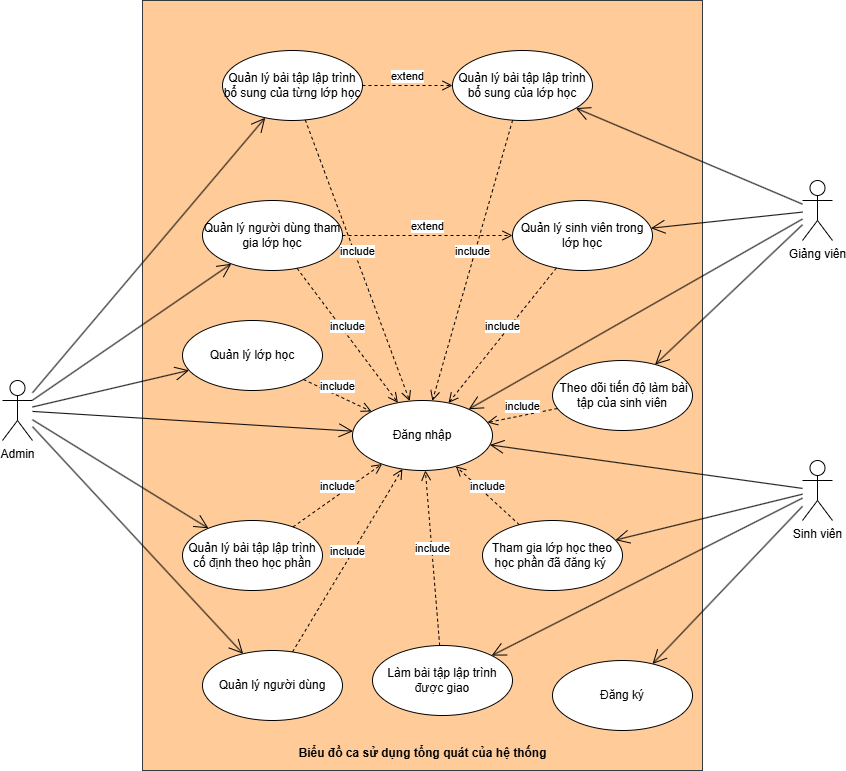
##### c. Ca sử dụng cho Sinh viên

|  |  |
| --- | --- |
| **Ca sử dụng cho Sinh viên** | **Mô tả** |
| Đăng ký | Sinh viên đăng ký tài khoản |
| Đăng nhập | Sinh viên đăng nhập vào hệ thống |
| Tham gia lớp học | Sinh viên tham gia lớp học phần đã đăng ký |
| Rời lớp học | Sinh viên rời khỏi lớp học phần |
| Làm bài tập lập trình được giao | Sinh viên làm bài tập lập trình trong lớp học đã tham gia |
| Xem kết quả thực thi mã nguồn | Sinh viên khi làm bài tập lập trình có thể thực thi mã nguồn luôn và hệ thống sẽ trả về kết quả |

Bảng 2.3. Bảng các ca sử dụng cho tác nhân Sinh viên

### 2.2.2. Xây dựng biểu đồ ca sử dụng

##### a. Biểu đồ ca sử dụng tổng quát của hệ thống

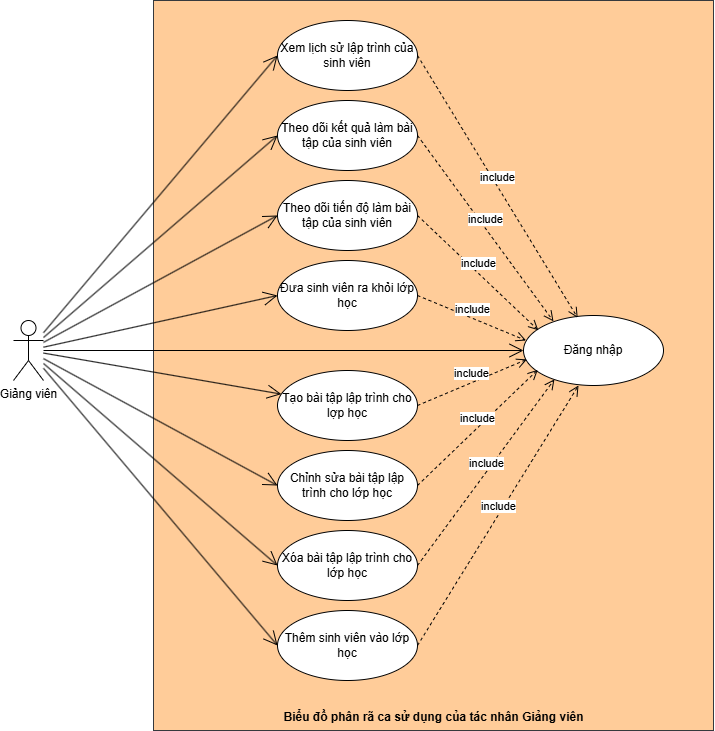


Hình 2.5. Biểu đồ ca sử dụng tổng quát của hệ thống

##### b. Biểu đồ phân rã ca sử dụng của tác nhân Admin

Hình 2.6. Biểu đồ phân rã ca sử dụng của tác nhân Admin

##### c. Biểu đồ phân rã ca sử dụng của tác nhân Giảng viên



Hình 2.7. Biểu đồ phân rã ca sử dụng của tác nhân Giảng viên

##### d. Biểu đồ phân rã ca sử dụng của tác nhân Sinh viên

Hình 2.8. Biểu đồ phân rã ca sử dụng của tác nhân Sinh viên

### 2.2.3. Đặc tả ca sử dụng

##### a. Đặc tả ca sử dụng của tác nhân Admin

###### i. Ca sử dụng đăng nhập

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Đăng nhập** |
| **Tác nhân** | Admin |
| **Mô tả** | Admin đăng nhập vào hệ thống để thực hiện các chức năng quản lý. |
| **Điều kiện tiên quyết** | - Admin đã có tài khoản trong hệ thống.  - Hệ thống đang hoạt động bình thường. |
| **Luồng sự kiện chính** | 1. Admin truy cập vào giao diện đăng nhập của hệ thống.  2. Admin nhập thông tin đăng nhập (tên người dùng/mật khẩu).  3. Admin nhấn nút "Đăng nhập".  4. Hệ thống kiểm tra thông tin đăng nhập.  5. Nếu thông tin hợp lệ, hệ thống chuyển Admin đến giao diện chính của hệ thống. |
| **Luồng sự kiện phụ** | - Nếu thông tin đăng nhập không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu Admin nhập lại. |
| **Điều kiện phụ** | - Admin đăng nhập thành công và có thể truy cập các chức năng quản lý.  - Nếu thất bại, Admin không thể truy cập hệ thống. |
| **Đặc tả chức năng** | - Hệ thống hiển thị giao diện đăng nhập với các trường nhập liệu: tên người dùng, mật khẩu.  - Hệ thống xác thực thông tin đăng nhập và chuyển hướng đến giao diện chính nếu thành công. |

Bảng 2.4. Bảng đặc tả ca sử dụng đăng nhập của Admin

###### ii. Ca sử dụng quản lý người dùng (Tạo, Chỉnh sủa, Xóa)

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Quản lý người dùng (Tạo, Chỉnh sửa, Xóa)** |
| **Tác nhân** | Admin |
| **Mô tả** | Admin thực hiện các thao tác tạo, chỉnh sửa và xóa người dùng trong hệ thống. |
| **Điều kiện tiên quyết** | - Admin đã đăng nhập vào hệ thống.  - Admin có quyền quản lý người dùng.  - Đối với chỉnh sửa/xóa: Người dùng cần chỉnh sửa/xóa đã tồn tại trong hệ thống. |
| **Luồng sự kiện chính** | **Tạo giảng viên mới:**  1. Admin truy cập vào mục "Giảng viên".  2. Admin chọn chức năng "Thêm giảng viên".  3. Hệ thống hiển thị form nhập thông tin (tên, email, mật khẩu...).  4. Admin điền thông tin và nhấn "Xác nhận".  5. Hệ thống lưu giảng viên mới và hiển thị thông báo thành công.  **Chỉnh sửa thông tin người dùng:**  1. Admin truy cập vào mục "Người dùng".  2. Admin chọn người dùng cần chỉnh sửa từ danh sách.  3. Hệ thống hiển thị form chứa thông tin hiện tại.  4. Admin chỉnh sửa thông tin (tên, email, vai trò...) và nhấn "Xác nhận".  5. Hệ thống cập nhật thông tin và hiển thị thông báo thành công.  **Xóa người dùng:**  1. Admin truy cập vào mục " Người dùng ".  2. Admin chọn người dùng cần xóa từ danh sách.  3. Admin nhấn nút "Xóa".  4. Hệ thống hiển thị thông báo xác nhận.  5. Admin xác nhận xóa.  6. Hệ thống xóa người dùng và hiển thị thông báo thành công. |
| **Luồng sự kiện phụ** | - Tạo: Nếu thông tin không hợp lệ (email trùng lặp, đã tồn tại), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Chỉnh sửa: Nếu thông tin chỉnh sửa không hợp lệ (email mới trùng lặp), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Xóa: Nếu Admin hủy hành động xóa, hệ thống trở về danh sách người dùng mà không thực hiện thay đổi. |
| **Điều kiện phụ** | - Tạo: Giảng viên mới được tạo và có thể đăng nhập.  - Chỉnh sửa: Thông tin người dùng được cập nhật.  - Xóa: Người dùng bị xóa khỏi hệ thống và không thể đăng nhập.  - Danh sách người dùng được cập nhật sau mỗi thao tác. |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện "Người dùng" với các chức năng: tạo, chỉnh sửa, xóa.  - Tạo: Form nhập thông tin với các trường bắt buộc: tên, email, mật khẩu.  - Chỉnh sửa: Form hiển thị thông tin hiện tại, cho phép chỉnh sửa các trường: tên, email, vai trò...  - Xóa: Hệ thống hiển thị nút "Xóa" và yêu cầu xác nhận trước khi thực hiện.  - Hệ thống kiểm tra tính hợp lệ của thông tin trước khi lưu/cập nhật. |

Bảng 2.5. Bảng đặc tả ca sử dụng quản lý người dùng (Tạo, Chỉnh sửa, Xoá) của Admin

###### iii. Ca sử dụng quản lý lớp học (Tạo, Chỉnh sửa, Xóa)

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Quản lý lớp học (Tạo, Chỉnh sửa, Xóa)** |
| **Tác nhân** | Admin |
| **Mô tả** | Admin thực hiện các thao tác tạo, chỉnh sửa và xóa lớp học trong hệ thống. |
| **Điều kiện tiên quyết** | - Admin đã đăng nhập vào hệ thống.  - Đối với chỉnh sửa/xóa: Lớp học hoặc học phần cần chỉnh sửa/xóa đã tồn tại trong hệ thống. |
| **Luồng sự kiện chính** | **Tạo lớp học mới:**  1. Admin truy cập vào mục "Lớp học phần".  2. Admin chọn chức năng "Thêm học phần" hoặc "Thêm lớp học phần".  3. Hệ thống hiển thị form nhập thông tin (tên lớp, mã lớp, mô tả...).  4. Admin điền thông tin và nhấn "Xác nhận".  5. Hệ thống lưu lớp học mới và hiển thị thông báo thành công.  **Chỉnh sửa thông tin lớp học:**  1. Admin truy cập vào mục " Lớp học phần ".  2. Admin chọn lớp học cần chỉnh sửa từ danh sách.  3. Hệ thống hiển thị form chứa thông tin hiện tại.  4. Admin chỉnh sửa thông tin (tên lớp, mã lớp, mô tả...) và nhấn "Xác nhận".  5. Hệ thống cập nhật thông tin và hiển thị thông báo thành công.  **Xóa lớp học:**  1. Admin truy cập vào mục " Lớp học phần ".  2. Admin chọn lớp học cần xóa từ danh sách.  3. Admin nhấn nút "Xóa".  4. Hệ thống hiển thị thông báo xác nhận.  5. Admin xác nhận xóa.  6. Hệ thống xóa lớp học và hiển thị thông báo thành công. |
| **Luồng sự kiện phụ** | - Tạo: Nếu thông tin không hợp lệ (mã lớp trùng lặp), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Chỉnh sửa: Nếu thông tin chỉnh sửa không hợp lệ (mã lớp mới trùng lặp, …), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Xóa: Nếu Admin hủy hành động xóa, hệ thống trở về danh sách lớp học mà không thực hiện thay đổi. |
| **Điều kiện phụ** | - Tạo: Lớp học mới được tạo và hiển thị trong danh sách.  - Chỉnh sửa: Thông tin lớp học được cập nhật.  - Xóa: Lớp học bị xóa khỏi hệ thống.  - Danh sách lớp học được cập nhật sau mỗi thao tác. |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện " Lớp học phần" với các chức năng: tạo, chỉnh sửa, xóa.  - Tạo: Form nhập thông tin với các trường bắt buộc: tên lớp, mã lớp.  - Chỉnh sửa: Form hiển thị thông tin hiện tại, cho phép chỉnh sửa các trường: tên lớp, mã lớp, mô tả...  - Xóa: Hệ thống hiển thị nút "Xóa" và yêu cầu xác nhận trước khi thực hiện.  - Hệ thống kiểm tra tính hợp lệ của thông tin trước khi lưu/cập nhật. |

Bảng 2.6. Bảng đặc tả ca sử dụng quản lý lớp học (Tạo, Chỉnh sửa, Xoá) của Admin

###### v. Ca sử dụng thêm giảng viên vào lớp học phần

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Thêm giảng viên vào lớp học phần** |
| **Tác nhân** | Admin |
| **Mô tả** | Admin thêm giảng viên vào lớp học phần. |
| **Điều kiện tiên quyết** | Admin đã đăng nhập vào hệ thống. |
| **Luồng sự kiện chính** | **Chọn lớp học phần:**  1. Admin truy cập vào mục "Lớp học phần".  2. Admin chọn chức năng "Chỉnh sửa lớp học phần".  3. Hệ thống hiển thị form và chọn giảng viên từ danh sách giảng viên.  4. Admin nhấn "Xác nhận".  5. Hệ thống thêm giảng viên vào lớp trong cơ sở dữ liệu.  6. Hệ thống hiển thị thông báo xác nhận thành công.  **Chọn giảng viên:**  1. Admin truy cập vào mục "Giảng viên".  2. Admin chọn giảng viên muốn thêm vào lớp học phần.  3. Admin chọn thêm lớp học phần cho giảng viên.  4. Hệ thống hiển thị form giao lớp học phần.  5. Admin chọn lớp học phần muốn thêm  6. Admin nhấn "Xác nhận".  7. Hệ thống thêm giảng viên vào lớp trong cơ sở dữ liệu.  8. Hệ thống hiển thị thông báo xác nhận thành công. |
| **Luồng sự kiện phụ** | - Nếu thông tin không hợp lệ (ví dụ: giảng viên đã được gán lớp này, lớp đã có giảng viên), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại. |
| **Điều kiện phụ** | - Giảng viên được thêm thành công và hiện trong danh sách các lớp học phần của giảng viên. |
| **Đặc tả chức năng** | - Hệ thống cung cấp 2 cách để thêm sinh viên cho lớp học phần.  - Hệ thống kiểm tra tính hợp lệ của thông tin trước khi lưu. |

Bảng 2.8. Bảng đặc tả ca sử dụng thêm giảng viên vào lớp học phần của Admin

##### b. Đặc tả ca sử dụng của tác nhân Giảng viên

###### i. Ca sử dụng đăng nhập

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Đăng nhập** |
| **Tác nhân** | Giảng viên |
| **Mô tả** | Giảng viên đăng nhập vào hệ thống để thực hiện các chức năng quản lý lớp học và bài tập. |
| **Điều kiện tiên quyết** | - Giảng viên đã có tài khoản trong hệ thống. |
| **Luồng sự kiện chính** | 1. Giảng viên truy cập vào giao diện đăng nhập của hệ thống.  2. Giảng viên nhập thông tin đăng nhập (tên người dùng/mật khẩu).  3. Giảng viên nhấn nút "Đăng nhập".  4. Hệ thống kiểm tra thông tin đăng nhập.  5. Nếu thông tin hợp lệ, hệ thống chuyển giảng viên đến giao diện chính của hệ thống. |
| **Luồng sự kiện phụ** | - Nếu thông tin đăng nhập không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu giảng viên nhập lại. |
| **Điều kiện phụ** | - Giảng viên đăng nhập thành công và có thể truy cập các chức năng quản lý.  - Nếu thất bại, giảng viên không thể truy cập hệ thống. |
| **Đặc tả chức năng** | - Hệ thống hiển thị giao diện đăng nhập với các trường nhập liệu: tên người dùng, mật khẩu.  - Hệ thống xác thực thông tin đăng nhập và chuyển hướng đến giao diện chính nếu thành công. |

Bảng 2.10. Bảng đặc tả ca sử dụng đăng nhập của Giảng viên

###### ii. Ca sử dụng quản lý bài tập lập trình cho lớp học (Tạo, Chỉnh sửa, Xoá)

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Quản lý bài tập lập trình cho lớp học (Tạo, Chỉnh sửa)** |
| **Tác nhân** | Giảng viên |
| **Mô tả** | Giảng viên thực hiện các thao tác tạo, chỉnh sửa, ẩn bài tập lập trình cho lớp học mà mình quản lý. |
| **Điều kiện tiên quyết** | - Giảng viên đã đăng nhập vào hệ thống.  - Giảng viên có quyền quản lý lớp học.  - Đối với chỉnh sửa: Bài tập cần chỉnh sửa, ẩn đã tồn tại trong lớp học. |
| **Luồng sự kiện chính** | **Tạo bài tập lập trình:**  1. Giảng viên truy cập vào mục "Quản lý lớp".  2. Giảng viên chọn lớp và chọn chức năng "Thêm bài tập".  3. Hệ thống hiển thị form nhập thông tin bài tập (tên bài, mô tả, test case, thời hạn...).  4. Giảng viên điền thông tin và nhấn "Tạo".  5. Hệ thống lưu bài tập vào lớp học và hiển thị thông báo thành công.  **Chỉnh sửa bài tập lập trình:**  1. Giảng viên truy cập vào mục "Quản lý lớp".  2. Giảng viên chọn lớp học và chọn chức năng chỉnh sửa từ danh sách.  3. Hệ thống hiển thị form chứa thông tin hiện tại của bài tập.  4. Giảng viên chỉnh sửa thông tin (tên bài, mô tả, test case, thời hạn...) và nhấn "Xác nhận".  5. Hệ thống cập nhật thông tin bài tập và hiển thị thông báo thành công.  **Ẩn bài tập lập trình:**  1. Giảng viên truy cập vào mục "Quản lý lớp".  2. Giảng viên chọn lớp học và chọn chức năng chỉnh sửa từ danh sách.  3. Hệ thống hiển thị form chứa thông tin hiện tại của bài tập.  4. Giảng viên chỉnh sửa trạng thái ẩn/hiện của bài tập và nhấn "Xác nhận".  5. Hệ thống cập nhật trạng thái bài tập và hiển thị thông báo thành công. |
| **Luồng sự kiện phụ** | - Tạo: Nếu thông tin không hợp lệ (test case không đúng định dạng), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Chỉnh sửa: Nếu thông tin chỉnh sửa không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại.  - Ẩn: Nếu giảng viên hủy hành động ẩn, hệ thống trở về danh sách bài tập mà không thực hiện thay đổi. |
| **Điều kiện phụ** | - Tạo: Bài tập mới được tạo và hiển thị trong danh sách bài tập của lớp học.  - Chỉnh sửa: Thông tin bài tập được cập nhật.  - Ẩn: Bài tập bị ẩn khỏi lớp học.  - Danh sách bài tập của lớp học được cập nhật sau mỗi thao tác. |
| Đặc tả chức năng | - Hệ thống cung cấp giao diện "Danh sách bài tập" theo lớp với các chức năng: tạo, chỉnh sửa, ẩn.  - Tạo: Form nhập thông tin với các trường bắt buộc: tên bài, mô tả, test case.  - Chỉnh sửa: Form hiển thị thông tin hiện tại, cho phép chỉnh sửa các trường: tên bài, mô tả, test case, thời hạn...  - Ẩn: Hệ thống hiển thị lựa chọn ẩn/hiện và yêu cầu xác nhận trước khi thực hiện.  - Hệ thống kiểm tra tính hợp lệ của thông tin trước khi lưu/cập nhật. |

Bảng 2.11. Bảng đặc tả ca sử dụng quản lý bài tập lập trình cho lớp học (Tạo, Chỉnh sửa, Ẩn) của Giảng viên

###### iii. Ca sử dụng quản lý sinh viên trong lớp học

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Quản lý sinh viên trong lớp học** |
| **Tác nhân** | Giảng viên |
| **Mô tả** | Giảng viên xem danh sách sinh viên lớp học mà mình quản lý. |
| **Điều kiện tiên quyết** | - Giảng viên đã đăng nhập vào hệ thống.  - Lớp học và sinh viên cần thêm/đưa ra đã tồn tại trong hệ thống.  - Giảng viên có quyền quản lý lớp học. |
| **Luồng sự kiện chính** | Thêm sinh viên vào lớp học:  1. Giảng viên truy cập vào mục "Quản lý lớp".  2. Giảng viên chọn lớp học.  3. Giảng viên chọn chức năng "Xem danh sách sinh viên".  4. Hệ thống hiển thị danh sách sinh viên trong lớp học. |
| **Luồng sự kiện phụ** | - Nếu không có sinh viên nào, hệ thống thông báo "Không có sinh viên". |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện quản lý danh sách thành viên lớp học. |

Bảng 2.12. Bảng đặc tả ca sử dụng quản lý sinh viên trong lớp học phần của Giảng viên

###### iv. Ca sử dụng theo dõi kết quả làm bài tập của sinh viên

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Theo dõi kết quả làm bài tập của sinh viên** |
| **Tác nhân** | Giảng viên |
| **Mô tả** | Giảng viên theo dõi kết quả và nội dung làm bài tập của sinh viên trong lớp học. |
| **Điều kiện tiên quyết** | - Giảng viên đã đăng nhập vào hệ thống.  - Lớp học và bài tập đã tồn tại trong hệ thống.  - Giảng viên có quyền quản lý lớp học. |
| **Luồng sự kiện chính** | Theo dõi kết quả làm bài tập:  1. Giảng viên truy cập vào mục "Quản lý lớp".  2. Giảng viên chọn lớp học cần theo dõi.  3. Giảng viên chọn bài tập, chọn chức năng "Xem nội dung nộp bài".  4. Hệ thống hiển thị danh sách sinh viên cùng kết quả và nội dung nộp bài của sinh viên đó |
| **Luồng sự kiện phụ** | - Nếu không có sinh viên hoặc bài tập nào trong lớp học, hệ thống thông báo "Không có dữ liệu để hiển thị". |
| **Điều kiện phụ** | - Giảng viên có thể xem được kết quả và nội dung làm bài tập của sinh viên trong lớp học. |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện theo dõi với chức năng theo dõi kết quả và nội dung bài làm của sinh viên.  - Theo dõi kết quả: Hiển thị chi tiết kết quả (điểm số, số test case đúng, thời gian nộp...). |

Bảng 2.13. Bảng đặc tả ca sử dụng theo dõi kết quả làm bài tập của sinh viên cho Giảng viên

##### c. Đặc tả ca sử dụng của tác nhân Sinh viên

###### i. Ca sử dụng đăng ký

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Đăng ký** |
| **Tác nhân** | Sinh viên |
| **Mô tả** | Sinh viên đăng ký tài khoản mới để sử dụng hệ thống. |
| **Điều kiện tiên quyết** | - Sinh viên chưa có tài khoản trong hệ thống. |
| **Luồng sự kiện chính** | 1. Sinh viên truy cập vào giao diện đăng ký của hệ thống.  2. Sinh viên nhập thông tin đăng ký (tên, email, mật khẩu, mã sinh viên...).  3. Sinh viên nhấn nút "Đăng ký".  4. Hệ thống kiểm tra thông tin đăng ký.  5. Nếu thông tin hợp lệ, hệ thống tạo tài khoản mới và hiển thị thông báo đăng ký thành công.  6. Sinh viên được chuyển đến giao diện đăng nhập. |
| **Luồng sự kiện phụ** | - Nếu thông tin không hợp lệ (email/mã sinh viên trùng lặp, mật khẩu không đủ mạnh...), hệ thống hiển thị thông báo lỗi và yêu cầu nhập lại. |
| **Điều kiện phụ** | - Tài khoản sinh viên được tạo thành công và sinh viên có thể đăng nhập.  - Nếu thất bại, sinh viên không thể tạo tài khoản. |
| **Đặc tả chức năng** | - Hệ thống hiển thị giao diện đăng ký với các trường nhập liệu: tên, email, mật khẩu, mã sinh viên.  - Hệ thống kiểm tra tính hợp lệ của thông tin (email/mã sinh viên không trùng lặp, mật khẩu đủ mạnh...).  - Hệ thống lưu thông tin tài khoản vào cơ sở dữ liệu sau khi xác nhận hợp lệ. |

Bảng 2.14. Bảng đặc tả ca sử dụng đăng ký của Sinh viên

###### ii. Ca sử dụng đăng nhập

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Đăng nhập** |
| **Tác nhân** | Sinh viên |
| **Mô tả** | Sinh viên đăng nhập vào hệ thống để tham gia lớp học và làm bài tập. |
| **Điều kiện tiên quyết** | - Sinh viên đã có tài khoản trong hệ thống.  - Hệ thống đang hoạt động bình thường. |
| **Luồng sự kiện chính** | 1. Sinh viên truy cập vào giao diện đăng nhập của hệ thống.  2. Sinh viên nhập thông tin đăng nhập (tên người dùng/mật khẩu).  3. Sinh viên nhấn nút "Đăng nhập".  4. Hệ thống kiểm tra thông tin đăng nhập.  5. Nếu thông tin hợp lệ, hệ thống chuyển sinh viên đến giao diện chính của hệ thống. |
| **Luồng sự kiện phụ** | - Nếu thông tin đăng nhập không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu sinh viên nhập lại. |
| **Điều kiện phụ** | - Sinh viên đăng nhập thành công và có thể truy cập các chức năng của hệ thống.  - Nếu thất bại, sinh viên không thể truy cập hệ thống. |
| **Đặc tả chức năng** | - Hệ thống hiển thị giao diện đăng nhập với các trường nhập liệu: tên người dùng, mật khẩu.  - Hệ thống xác thực thông tin đăng nhập và chuyển hướng đến giao diện chính nếu thành công. |

Bảng 2.15. Bảng đặc tả ca sử dụng đăng nhập của Sinh viên

###### iii. Ca sử dụng tham gia lớp học

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Tham gia lớp học** |
| **Tác nhân** | Sinh viên |
| **Mô tả** | Sinh viên tham gia vào một lớp học bằng cách nhập mã lớp. |
| **Điều kiện tiên quyết** | - Sinh viên đã đăng nhập vào hệ thống.  - Lớp học đã tồn tại trong hệ thống.  - Sinh viên chưa là thành viên của lớp học đó. |
| **Luồng sự kiện chính** | 1. Sinh viên truy cập vào mục "Tham gia lớp học" trên giao diện chính.  2. Sinh viên nhập mã lớp học.  3. Sinh viên nhấn nút "Tham gia".  4. Hệ thống kiểm tra mã lớp học.  5. Nếu hợp lệ, hệ thống thêm sinh viên vào lớp học và hiển thị thông báo thành công.  6. Sinh viên được chuyển đến giao diện của lớp học. |
| **Luồng sự kiện phụ** | - Nếu mã lớp học không hợp lệ hoặc sinh viên đã là thành viên của lớp, hệ thống hiển thị thông báo lỗi và yêu cầu thử lại. |
| **Điều kiện phụ** | - Sinh viên được thêm vào lớp học thành công và có thể truy cập các bài tập của lớp.  - Nếu thất bại, sinh viên không thể tham gia lớp học. |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện "Tham gia lớp học" với trường nhập mã lớp.  - Hệ thống kiểm tra tính hợp lệ của mã lớp trước khi thêm sinh viên vào lớp.  - Hệ thống cập nhật danh sách thành viên của lớp học sau khi sinh viên tham gia thành công. |

Bảng 2.16. Bảng đặc tả ca sử dụng tham gia lớp học của Sinh viên

###### iv. Ca sử dụng làm bài tập lập trình được giao

|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Làm bài tập lập trình được giao** |
| **Tác nhân** | Sinh viên |
| **Mô tả** | Sinh viên làm bài tập lập trình được giao trong lớp học mà mình tham gia. |
| **Điều kiện tiên quyết** | - Sinh viên đã đăng nhập vào hệ thống.  - Sinh viên đã tham gia lớp học.  - Bài tập lập trình đã được tạo cố định hoặc được giảng viên giao và còn trong thời hạn. |
| **Luồng sự kiện chính** | 1. Sinh viên truy cập vào giao diện lớp học.  2. Sinh viên chọn bài tập lập trình từ danh sách bài tập được giao.  3. Hệ thống hiển thị giao diện làm bài với đề bài, trình soạn thảo mã nguồn và các test case mẫu.  4. Sinh viên viết mã nguồn trong trình soạn thảo.  5. Sinh viên nhấn nút "Thực thi" hoặc "Nộp bài" để gửi mã nguồn.  6. Hệ thống thực thi mã nguồn và hiển thị kết quả (xem ca sử dụng "Xem kết quả thực thi mã nguồn").  7. Nếu sinh viên "Thực thi" thì hệ thống trả về kết quả thực thi mã nguồn, khi ấn "Nộp bài" thì Hệ thống lưu bài nộp của sinh viên và hiển thị thông báo nộp bài thành công. |
| **Luồng sự kiện phụ** | - Nếu mã nguồn có lỗi cú pháp, hệ thống thông báo lỗi và yêu cầu sinh viên sửa lại. |
| **Điều kiện phụ** | - Bài nộp của sinh viên được lưu vào hệ thống.  - Sinh viên có thể xem kết quả thực thi mã nguồn của mình.  - Trạng thái bài tập của sinh viên được cập nhật (đã nộp). |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện làm bài với: đề bài, trình soạn thảo mã nguồn, test case mẫu.  - Hệ thống hỗ trợ sinh viên viết xem kết quả và nộp mã nguồn.  - Hệ thống tự động thực thi mã nguồn sau khi sinh viên nộp bài và lưu bài nộp vào cơ sở dữ liệu. |

Bảng 2.17. Bảng đặc tả ca sử dụng làm bài tập lập trình được giao

###### vi. Ca sử dụng tham gia vào cuộc trò chuyện chung các lớp tham gia

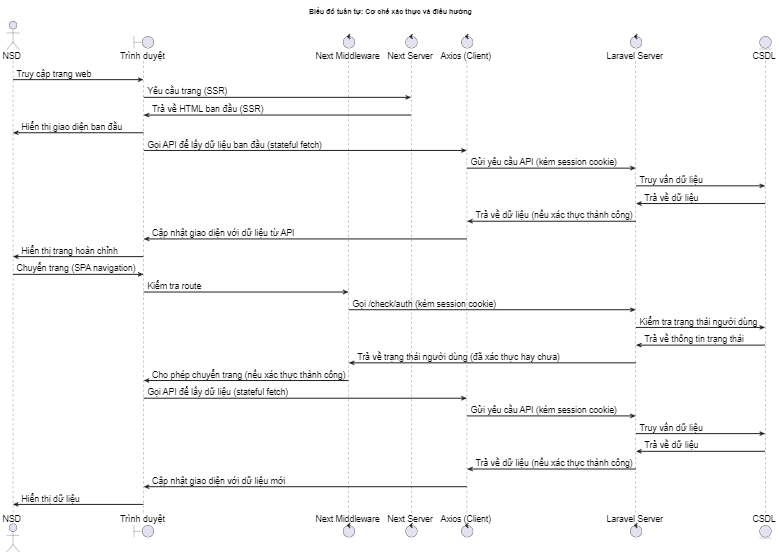
|  |  |
| --- | --- |
| **Tên ca sử dụng** | **Tham gia vào cuộc trò chuyện chung các lớp tham gia** |
| **Tác nhân** | Sinh viên |
| **Mô tả** | Sinh viên tham gia vào cuộc trò chuyện chung của các lớp học mà mình đã tham gia để trao đổi với giảng viên và các sinh viên khác. |
| **Điều kiện tiên quyết** | - Sinh viên đã đăng nhập vào hệ thống.  - Sinh viên đã tham gia ít nhất một lớp học.  - Lớp học có tính năng trò chuyện chung được kích hoạt. |
| **Luồng sự kiện chính** | 1. Sinh viên truy cập vào giao diện lớp học mà mình đã tham gia.  2. Sinh viên chọn chức năng "Chatbox" từ giao diện lớp học.  3. Hệ thống hiển thị giao diện trò chuyện chung của lớp học.  4. Sinh viên nhập nội dung tin nhắn và nhấn "Gửi".  5. Hệ thống lưu tin nhắn và hiển thị trong cuộc trò chuyện để các thành viên khác (giảng viên, sinh viên) cùng lớp có thể xem.  6. Sinh viên có thể xem các tin nhắn trước đó và trả lời nếu cần. |
| **Luồng sự kiện phụ** | - Nếu sinh viên gửi tin nhắn không hợp lệ (ví dụ: ảnh, file), hệ thống hiển thị thông báo lỗi và không gửi tin nhắn. |
| **Điều kiện phụ** | - Tin nhắn của sinh viên được gửi thành công và hiển thị trong cuộc trò chuyện chung.  - Sinh viên có thể giao tiếp với các thành viên khác trong lớp học. |
| **Đặc tả chức năng** | - Hệ thống cung cấp giao diện trò chuyện chung trong mỗi lớp học với các tính năng: gửi tin nhắn, xem lịch sử tin nhắn, trả lời tin nhắn.  - Hệ thống lưu trữ lịch sử trò chuyện để các thành viên trong lớp có thể xem lại.  - Hệ thống kiểm tra nội dung tin nhắn để đảm bảo không chứa nội dung không phù hợp. |

Bảng 2.19. Bảng đặc tả ca sử dụng tham gia cuộc trò chuyện chung của Sinh viên

## 2.3. Mô hình hóa hoạt động của hệ thống

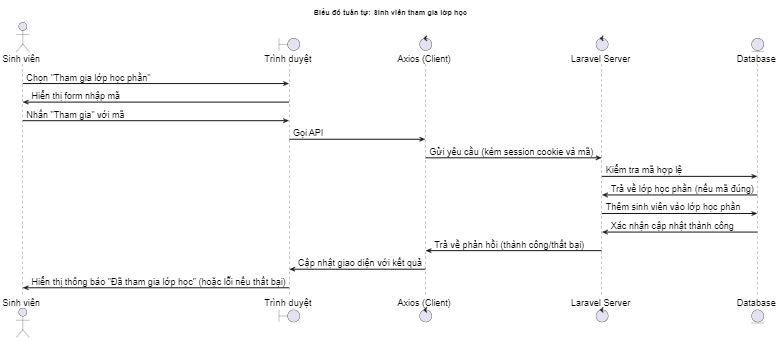
### 2.3.1. Mô hình hoạt động xác thực và điều hướng

Kiến trúc tổng quan của hệ thống được thiết kế để đảm bảo tính linh hoạt, bảo mật và hiệu suất trong việc quản lý học tập lập trình trực tuyến. Hệ thống bao gồm hai thành phần chính: Frontend được xây dựng bằng **Next.js** và Backend sử dụng **Laravel**. Frontend hoạt động như một ứng dụng **SPA (Single Page Application)**, trong đó các route được quản lý bởi **Next Middleware**. Middleware này kiểm tra trạng thái người dùng bằng cách gọi endpoint /check/auth của **Laravel** mỗi khi người dùng chuyển trang, đảm bảo tính bảo mật và đồng bộ trạng thái. Dữ liệu được lấy từ backend thông qua các yêu cầu stateful fetch sử dụng **Axios**, với session token được lưu trong cookies để xác thực người dùng. Laravel backend xử lý toàn bộ logic nghiệp vụ, quản lý dữ liệu và cung cấp các API endpoint cho frontend. Ngoài ra, hệ thống hỗ trợ **SSR (Server-Side Rendering)** thông qua **Next Server** cho một số component, giúp cải thiện hiệu suất tải trang ngay cả khi phần lớn các tương tác dữ liệu được thực hiện từ phía client. Cơ chế này đảm bảo sự cân bằng giữa hiệu suất, bảo mật và trải nghiệm người dùng.



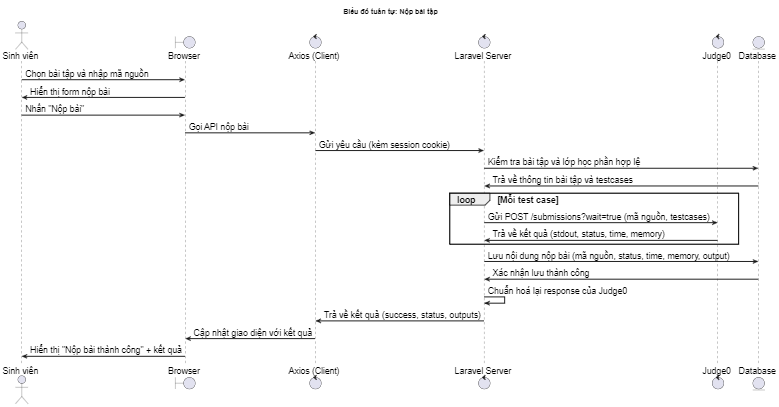
Hình 2.9. Biểu đồ tuần tự cơ chế xác thực và điều hướng của hệ thống

### 2.3.2. Biểu dồ tuần tự hoạt động tham gia lớp học

Để tham gia lớp học phần, trước tiên hết Giảng viên cần vào giao diện quản lý lớp để lấy mã tham gia học phần, sau đó gửi cho sinh viên qua các kênh giao tiếp khác hoặc cung cấp cho sinh viên khi học trên giảng đường. Sau đó sinh viên sẽ từ giao diện của mình, chọn “tham gia lớp học phần” và nhập mã tương ứng. 

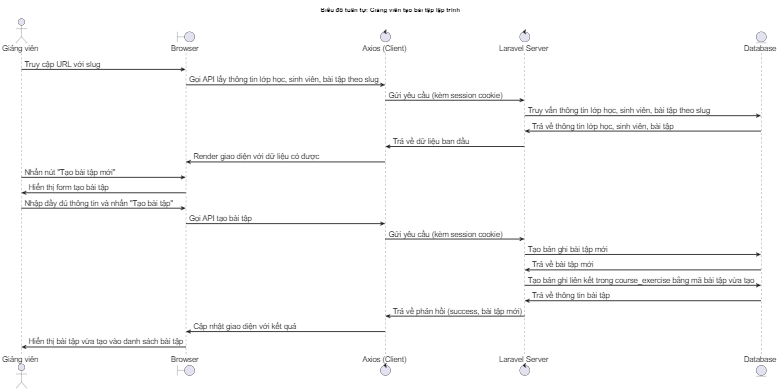
Hình 2.10. Biểu đồ tuần tự hoạt động tham gia lớp học

### 2.3.3. Biểu đồ tuần tự của hoạt động làm bài và nộp bài tập lập trình



Hình 2.11. Biểu đồ tuần tự sinh viên làm bài và nộp bài

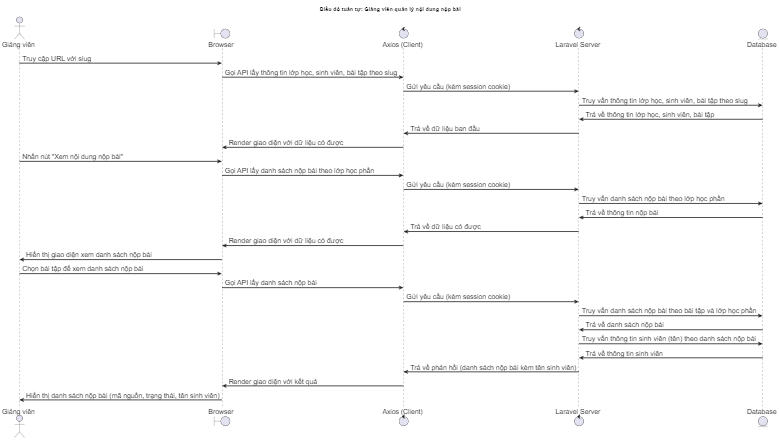
### 2.3.4. Biểu đồ tuần tự của hoạt động tạo bài tập lập trình trong lớp học phần



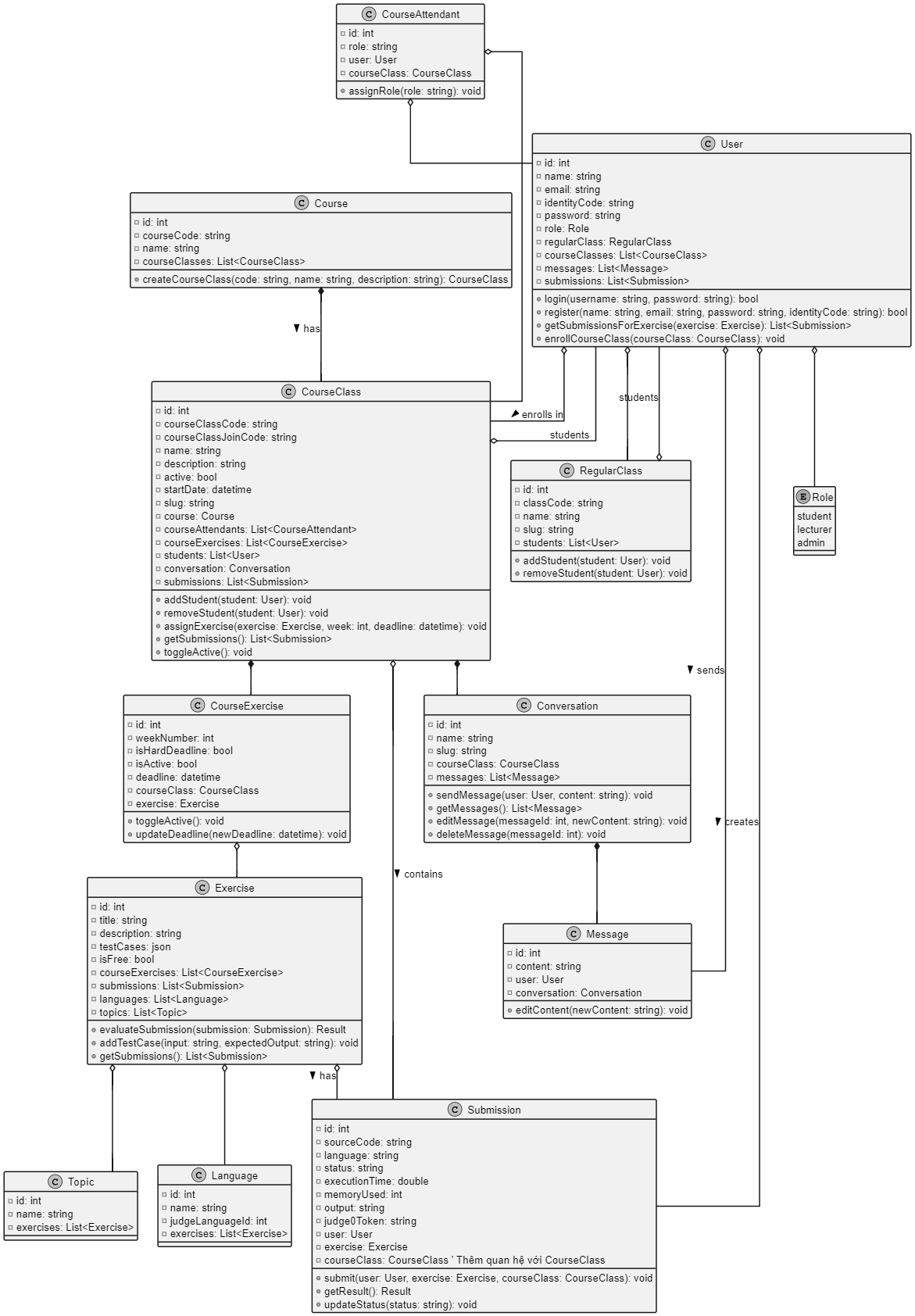
Hình 2.12. Biểu đồ tuần tự Giảng viên tạo bài tập lập trình trong lớp học phần

### 2.3.5. Biểu đồ tuần tự hoạt động xem thông tin nộp bài của sinh viên

Hình 2.13. Biểu đồ tuần tự hoạt động xem thông tin nộp bài của sinh viên



## 2.4. Biểu đồ lớp của hệ thống



Hình 2.14. Biểu đồ lớp của hệ thống

## 2.5. Thiết kế cơ sở dữ liệu

### 2.5.1.Thiết kế mức khái niệm

##### a. Các thực thể

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thực thể** | **Ý nghĩa** |
| 1 | Người dùng (User) | Đại diện người dùng hệ thống, phân quyền theo sinh viên, giảng viên và admin. |
| 2 | Lớp niên chế (Regular Class) | Đại diện các lớp niên chế. |
| 3 | Học phần (Course) | Đại diện cho học phần |
| 4 | Lớp học phần (Course Class) | Đại diện cho lớp học phần |
| 5 | Bài tập (Exercise) | Thông tin về bài tập lập trình |
| 6 | Nội dung nộp bài (Submission) | Lưu trữ thông tin nộp bài cho sinh viên. |
| 7 | Ngôn ngữ (Language) | Thông tin các ngôn ngữ lập trình được hỗ trợ. |
| 8 | Chủ đề (Topic) | Đại diện cho các chủ đề của bài tập. |

Bảng 2.21. Xác định các thực thể

##### b. Xác định thuộc tính của thực thể

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên thực thể** | **Thuộc tính** |
| 1 | Người dùng (User) | id (khoá chính), name, email, identity\_code |
| 2 | Lớp niên chế (Regular Class) | id (khoá chính), class\_code, name |
| 3 | Học phần (Course) | id (khoá chính), course\_code, name |
| 4 | Lớp học phần (Course Class) | id (khoá chính), course\_class\_code, name, course\_class\_join\_code, description, active, start\_date |
| 5 | Bài tập (Exercise) | id (khoá chính), title, description, test\_cases, is\_free |
| 6 | Nội dung nộp bài (Submission) | id (khoá chính), source\_code, language, status, execution\_time, memory\_used, output |
| 7 | Ngôn ngữ (Language) | id (khoá chính), name, judge\_language\_id |
| 8 | Chủ đề (Topic) | id (khoá chính), name |

Bảng 2.22. Thực thể và thuộc tính của thực thể

##### c. Quan hệ giữa các thực thể

|  |  |  |  |
| --- | --- | --- | --- |
| **Thực thể** | **Quan hệ** | **Thực thể** | **Bản số** |
| User | có | Course Class | n - n |
| User | có | Message | 1 - n |
| User | có | Submission | 1 - n |
| Regular Class | có | Course Class | 1 - n |
| Regular Class | có | User | 1 - n |
| Course | có | Course Class | 1 - n |
| Course | có | Exercise | n - n |
| Course Class | có | Exercise | n - n |
| Course Class | có | Submission | 1 - n |
| Course Class | có | Conversation | 1 - 1 |
| Exercise | có | Submission | 1 - n |
| Exercise | được hỗ trợ | Language | n - n |
| Exercise | có | Topic | n - n |
| Conversation | có | Message | 1 - n |

Bảng 2.23. Quan hệ giữa các thực thể

##### e. Mô hình thực thể liên kết

Hình 2.15. Mô hình thực thể liên kết

### 2.5.2. Thiết kế mức vật lý

##### a. Bảng users

|  |  |  |  |
| --- | --- | --- | --- |
| **User** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | name | VARCHAR |  |
| 3 | email | VARCHAR | UNIQUE |
| 4 | identity\_code | VARCHAR | UNIQUE |
| 5 | email\_verified\_at | TIMESTAMP |  |
| 6 | password | VARCHAR |  |
| 7 | role | ENUM | ‘student’, ‘lecturer’, ‘admin’ |
| 8 | regular\_class\_id | INT | FK (regular\_classes, id) |

Trong bảng này có identity\_code với ràng buộc UNIQUE đại diện cho mã sinh viên thực tế của sinh viên, thông tin này không bắt buộc nhưng là yếu tố để phát triển các tính năng tiện nghi sau này, ví dụ như, dựa vào danh sách sinh viên có mã sinh viên và mã học phần để import sinh viên vào lớp học nhanh chóng.

##### b. Bảng regular\_classes

|  |  |  |  |
| --- | --- | --- | --- |
| **regular\_classes** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | class\_code | VARCHAR |  |
| 3 | name | VARCHAR |  |
| 4 | slug | VARCHAR |  |

Bảng này đóng vai trò tham chiếu quan trọng, mặc dù dữ liệu của bảng này không thay đổi và thêm sửa xoá nhiều.

##### c. Bảng courses

|  |  |  |  |
| --- | --- | --- | --- |
| **courses** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | course\_code | VARCHAR | UNIQUE |
| 3 | name | VARCHAR |  |

##### d. Bảng course\_classes

|  |  |  |  |
| --- | --- | --- | --- |
| **course\_classes** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | assigned\_regular\_class\_id | INT | FK (regular\_classes, id) |
| 3 | course\_class\_code | VARCHAR | UNIQUE |
| 4 | course\_class\_join\_code | VARCHAR | UNIQUE |
| 5 | name | VARCHAR |  |
| 6 | description | TEXT |  |
| 7 | active | TINYINT |  |
| 8 | start\_date | DATETIME |  |
| 9 | course\_id | INT | FK (courses, id) |
| 10 | slug | VARCHAR |  |

Mỗi bản ghi trong bảng đại diện cho một lớp học phần, mỗi lớp học phần phải thuộc về một học phần, và có mã tham gia (course\_class\_join\_code). Ngày bắt đầu (start\_date) là tham chiếu thời gian bắt đầu và cũng dùng để tính toán tương đối thời gian kết thúc của khoá học, và cũng là tham chiếu để cập nhật trạng thái hoạt động (active) của lớp học phần.

##### e. Bảng submissions

|  |  |  |  |
| --- | --- | --- | --- |
| **submissions** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | user\_id | INT | FK (users, id) |
| 3 | exercise\_id | INT | FK (exercises, id) |
| 4 | course\_class\_id | INT | FK (course\_classes, id) |
| 5 | source\_code | TEXT |  |
| 6 | language | VARCHAR |  |
| 7 | status | VARCHAR |  |
| 8 | execution\_time | DOUBLE |  |
| 9 | memory\_used | INT |  |
| 10 | output | TEXT |  |
| 11 | judge0\_token | VARCHAR |  |

Cặp 3 khoá (user\_id, exercise\_id, course\_class\_id) sẽ quyết định một bản nộp bài duy nhất của một người, trong một lớp, dành cho một bài tập nhất định.

Các thông tin khác là thông tin bổ sung cho bài nộp đó gồm: mã nguồn của sinh viên nộp, ngôn ngữ được sử dụng bởi sinh viên, trạng thái compile của hệ thống về, các thông tin hiệu suất như thời gian thực thi, bộ nhớ sử dụng và đầu ra của mã nguồn.

##### f. Bảng course\_exercise

|  |  |  |  |
| --- | --- | --- | --- |
| **course\_exercise** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | course\_id | INT | FK (courses, id);  NULLABLE |
| 3 | course\_class\_id | INT | FK (course\_classes, id) |
| 4 | exercise\_id | INT | FK (exercises, id) |
| 5 | week\_number | INT |  |
| 6 | is\_hard\_deadline | TINYINT |  |
| 7 | is\_active | TINYINT |  |
| 8 | deadline | DATETIME |  |

Khác với bảng submissions, ở đây không sử dụng cặp 3 khoá hoàn toàn, thay vào logic sẽ phụ thuộc vào 2 cặp khoá (course\_id – exercise\_id và course\_class\_id – exercise\_id).

Bởi vì quan hệ giữa Course và Course Class đối với Exercise đều là n – n, nên thiết kế này mang tính chất quản lý tập trung.

Đối với cặp khoá course\_id – exercise\_id sẽ quyết định được các bài tập nào được định nghĩa cho toàn bộ học phần, từ đó các lớp học phần con có thể kế thừa bài tập đã được định nghĩa sẵn.

Đối với cặp khoá course\_class\_id – exercise\_id mới thực sự quyết định toàn bộ bài tập trong lớp học phần đó.

Ở đây có một ràng buộc bổ sung để đảm bảo tính toàn vẹn và chặt chẽ của bảng này là ràng buộc UNIQUE cho cặp 3 khoá (course\_id, course\_class\_id, exercise\_id).

##### g. Bảng course\_attendant

|  |  |  |  |
| --- | --- | --- | --- |
| **course\_attendant** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | course\_class\_id | INT | FK (courses, id) |
| 3 | user\_id | INT | FK (users, id) |
| 4 | role | VARCHAR |  |

Mỗi bản ghi đại diện cho một người dùng trong lớp học đó, và để phân biệt vai trò của người dùng trong lớp học đó, sử dụng cột vai trò (role) trong bảng, cột này chỉ có mục đích làm đơn giản hoá xử lý trong code, nhiều hơn là sự chặt chẽ về quan hệ bảng, bởi vì trong bảng users cũng đã chứa thông tin vai trò của người dùng.

##### h. Bảng topics

|  |  |  |  |
| --- | --- | --- | --- |
| **topics** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | name | VARCHAR |  |

##### i. Bảng languages

|  |  |  |  |
| --- | --- | --- | --- |
| **language** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | name | VARCHAR |  |
| 3 | judge\_language\_id | INT |  |

Judge\_language\_id là cố định và chỉ là thông tin tham chiếu đến đúng mã ngôn ngữ của hệ thống chấm điểm.

##### j. Bảng exercise\_topic

|  |  |  |  |
| --- | --- | --- | --- |
| **exercise\_topic** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | exercise\_id | INT | FK (exercises, id) |
| 3 | topic\_id | INT | FK (topics, id) |

##### k. Bảng exercise\_language

|  |  |  |  |
| --- | --- | --- | --- |
| **exercise\_language** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | exercise\_id | INT | FK (exercises, id) |
| 3 | language\_id | INT | FK (languages, id) |

##### l. Bảng conversations

|  |  |  |  |
| --- | --- | --- | --- |
| **conversations** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | name | VARCHAR |  |
| 3 | course\_class\_id | INT | FK (course\_classes, id) |
| 4 | slug | VARCHAR |  |

##### m. Bảng messages

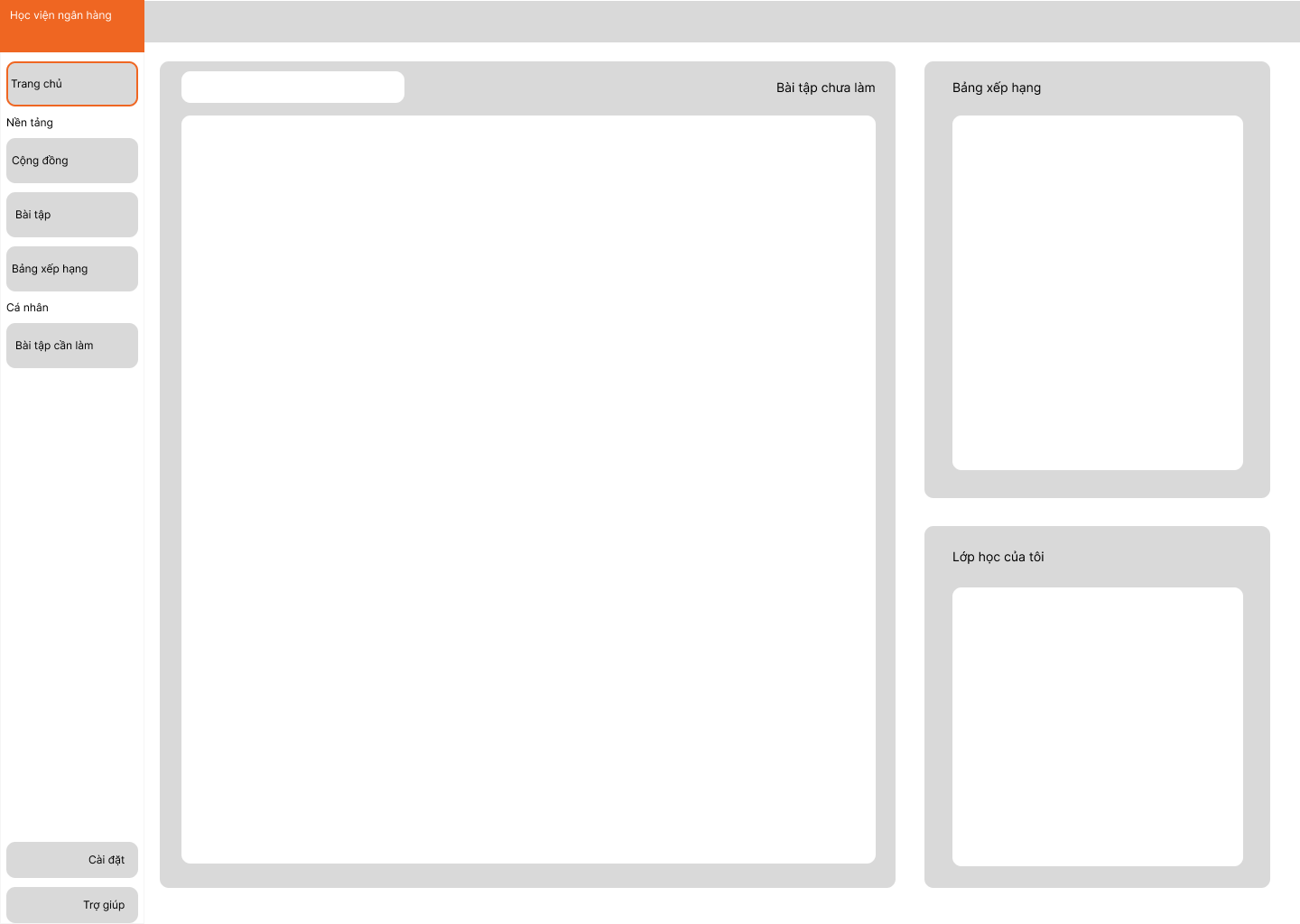
|  |  |  |  |
| --- | --- | --- | --- |
| **messages** | | | |
| **STT** | **Tên trường** | **kiểu dữ liệu** | **Chú thích** |
| 1 | id | INT | PK |
| 2 | user\_id | INT | FK (users, id) |
| 3 | course\_class\_id | INT | FK (course\_classes, id) |
| 4 | content | TEXT |  |

##### n. Mô hình dữ liệu quan hệ

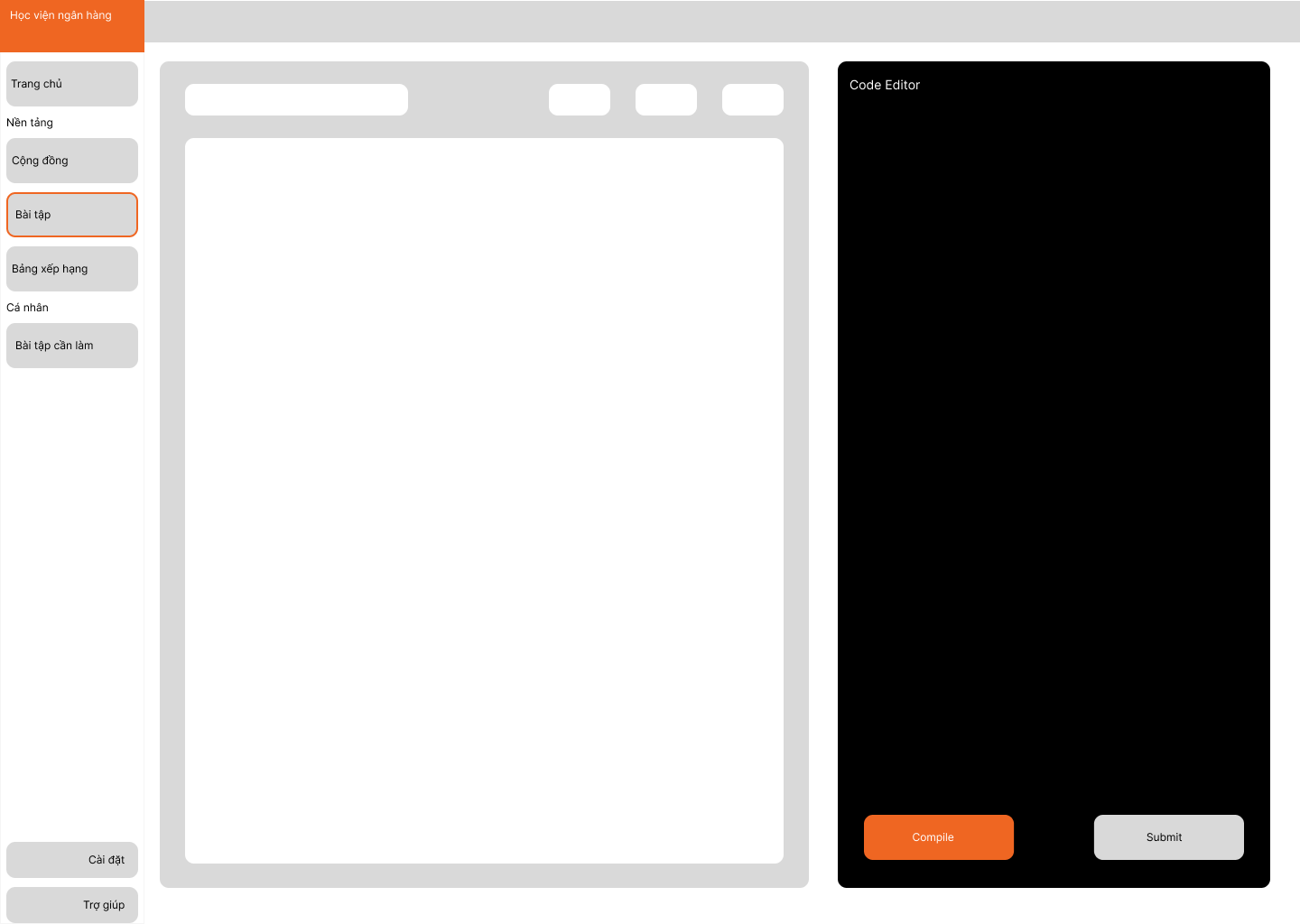
Hình 2.16. Mô hình dữ liệu quan hệ

## 2.6. Thiết kế giao diện người dùng (UI/UX)

### 2.6.1. Thiết kế wireframe giao diện cho sinh viên

****

Hình 2.17. Wireframe giao diện trang chủ. Wireframe giao diện làm bài tập

****

Hình 2.18. Wireframe giao diện làm bài tập

### 2.6.2 Thiết kế wireframe giao diện cho giảng viên



Hình 2.19. Wireframe giao diện chung cho giảng viên

### 2.6.3 Thiết kế wireframe giao diện cho quản trị viên

****

Hình 2.20. Wireframe giao diện chung cho admin

# CHƯƠNG 3: CÀI ĐẶT VÀ KIỂM THỬ HỆ THỐNG

## 3.1. Bảo mật hệ thống

Để đảm bảo an toàn cho hệ thống trước các mối đe dọa bảo mật phổ biến và bảo vệ dữ liệu người dùng, hệ thống được triển khai với các cơ chế bảo mật cần thiết. Hai biện pháp chính được áp dụng là bảo vệ chống tấn công XSS bằng XSRF token và mã hóa mật khẩu, nhằm tăng cường tính an toàn và đáng tin cậy cho hệ thống.

### 3.1.1. Bảo vệ chống tấn công XSS bằng XSRF token

Cơ chế này được triển khai nhằm ngăn chặn các cuộc tấn công Cross-Site Request Forgery (CSRF/XSS), trong đó kẻ tấn công có thể giả mạo yêu cầu từ người dùng hợp lệ để thực hiện các hành động trái phép. Trong hệ thống, phía frontend được xây dựng bằng Next.js sẽ gửi kèm XSRF token trong header của mỗi yêu cầu POST, PUT hoặc DELETE. Phía backend sử dụng Laravel với middleware VerifyCsrfToken để kiểm tra tính hợp lệ của token trong mỗi yêu cầu từ client. Token này được sinh tự động bởi Laravel và lưu trữ trong cookie phiên (session cookie), đảm bảo rằng mọi yêu cầu gửi đến đều được xác thực. Kết quả là, các yêu cầu không hợp lệ, chẳng hạn như thiếu token hoặc token không khớp, sẽ bị từ chối ngay lập tức, giúp giảm thiểu nguy cơ tấn công CSRF/XSS một cách hiệu quả.

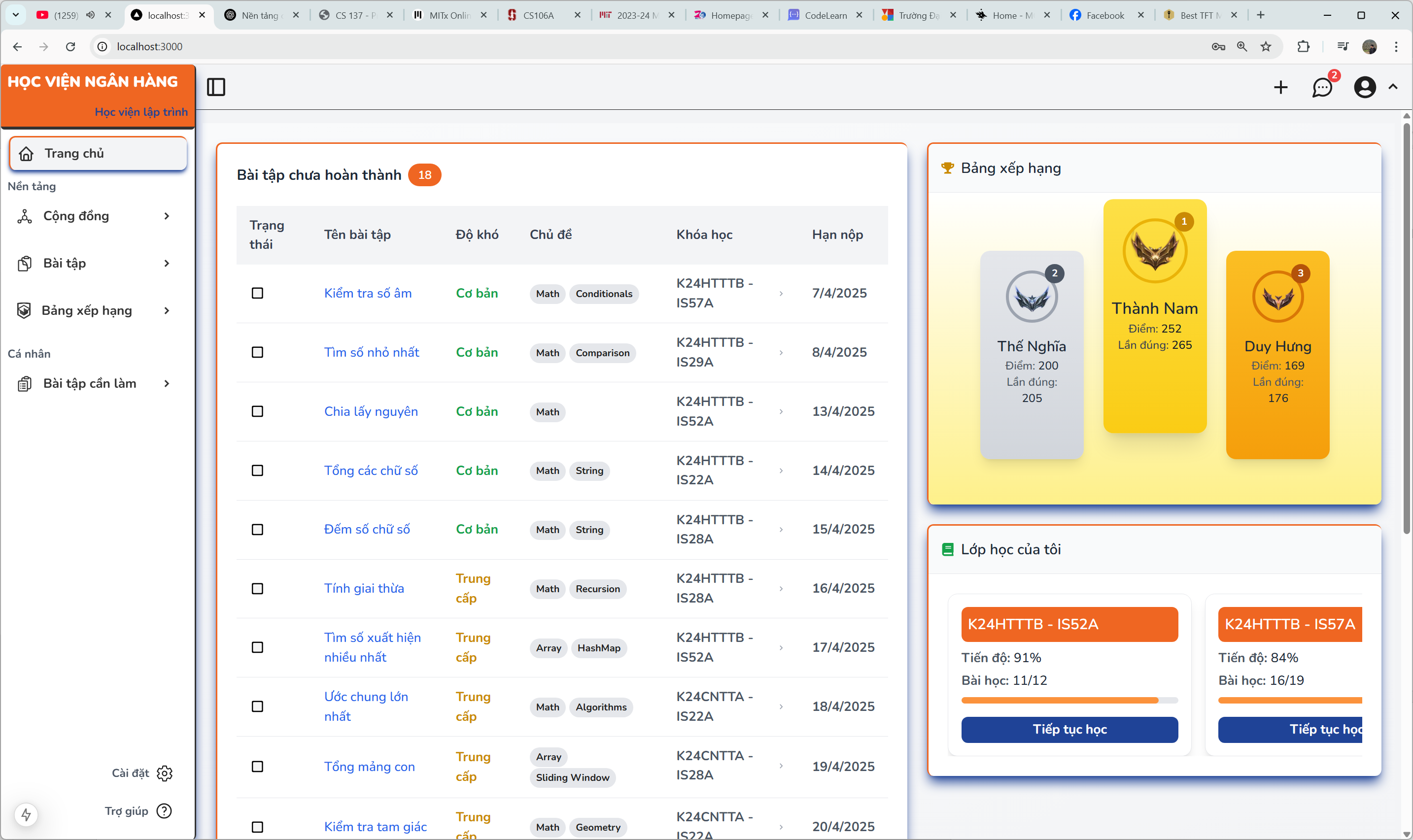
### 3.1.2. Mã hóa mật khẩu

Để bảo vệ thông tin đăng nhập của người dùng, bao gồm sinh viên, giảng viên và admin, trước nguy cơ bị lộ khi cơ sở dữ liệu bị xâm phạm, hệ thống áp dụng mã hóa mật khẩu bằng thuật toán bcrypt. Khi người dùng đăng ký hoặc cập nhật thông tin, mật khẩu sẽ được mã hóa bằng hàm Hash::make() do Laravel cung cấp trước khi lưu vào bảng users trong cơ sở dữ liệu MySQL. Trong quá trình đăng nhập, hệ thống sử dụng hàm Hash::check() để so sánh mật khẩu mà người dùng nhập với giá trị đã mã hóa trong cơ sở dữ liệu. Nhờ cơ chế này, mật khẩu được lưu dưới dạng chuỗi băm (hash) không thể giải mã ngược, đảm bảo rằng ngay cả khi dữ liệu bị truy cập trái phép, thông tin đăng nhập vẫn được bảo vệ an toàn.

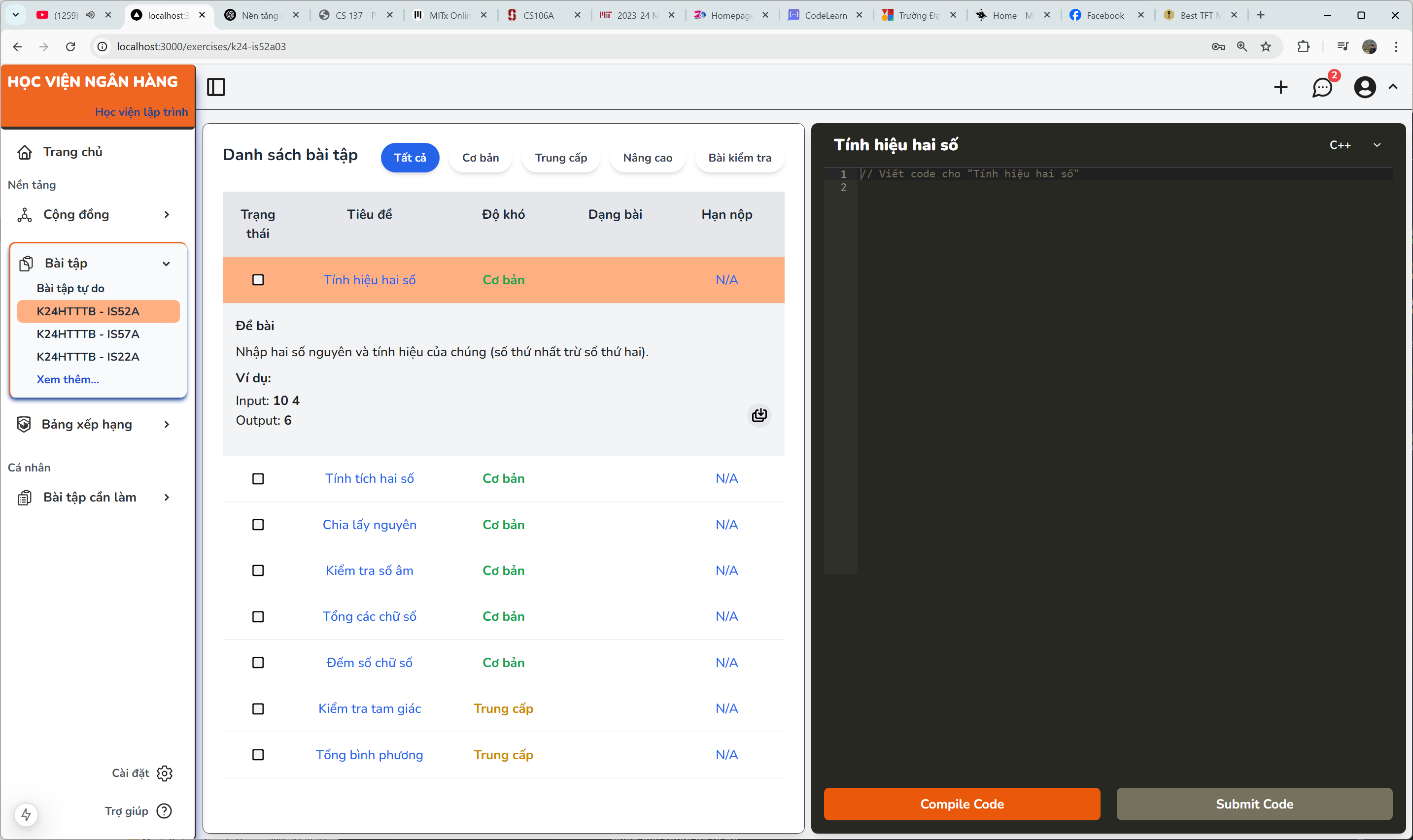
Các cơ chế bảo mật trên được triển khai dựa trên sự kết hợp của các công nghệ chính trong hệ thống. Phía frontend, Next.js chịu trách nhiệm quản lý XSRF token và gửi kèm trong các yêu cầu API. Phía backend, Laravel cung cấp middleware CSRF để xác thực token và thư viện Hash để thực hiện mã hóa mật khẩu. Cơ sở dữ liệu MySQL được sử dụng để lưu trữ mật khẩu đã mã hóa trong bảng users, đảm bảo dữ liệu được quản lý một cách an toàn và hiệu quả.

## 3.2. Xây dựng giao hiện hệ thống

### 3.2.1. Giao diện cho sinh viên



Hình 3.1. Giao diện trang chủ dành cho sinh viên

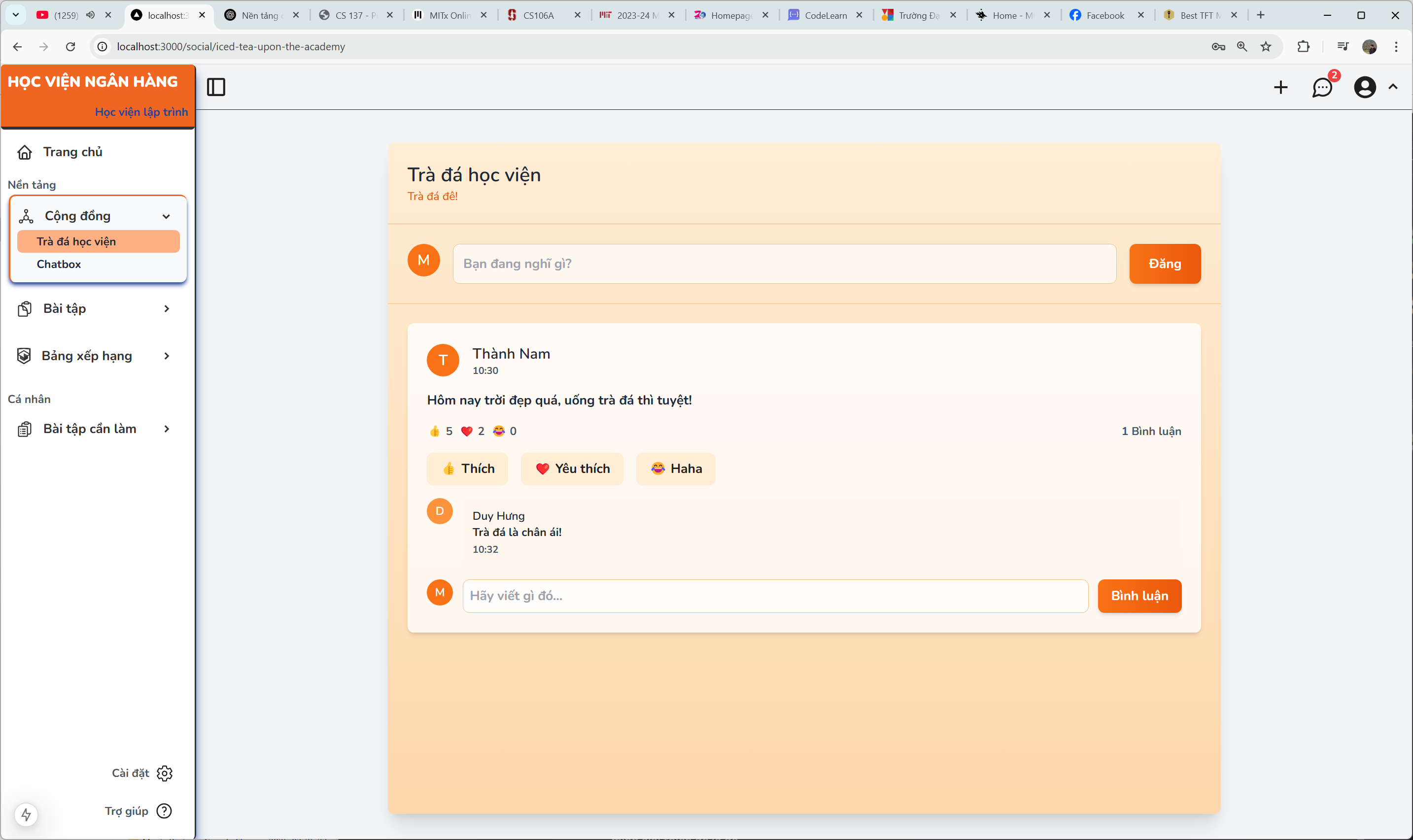


Hình 3.2. Giao diện làm bài tập

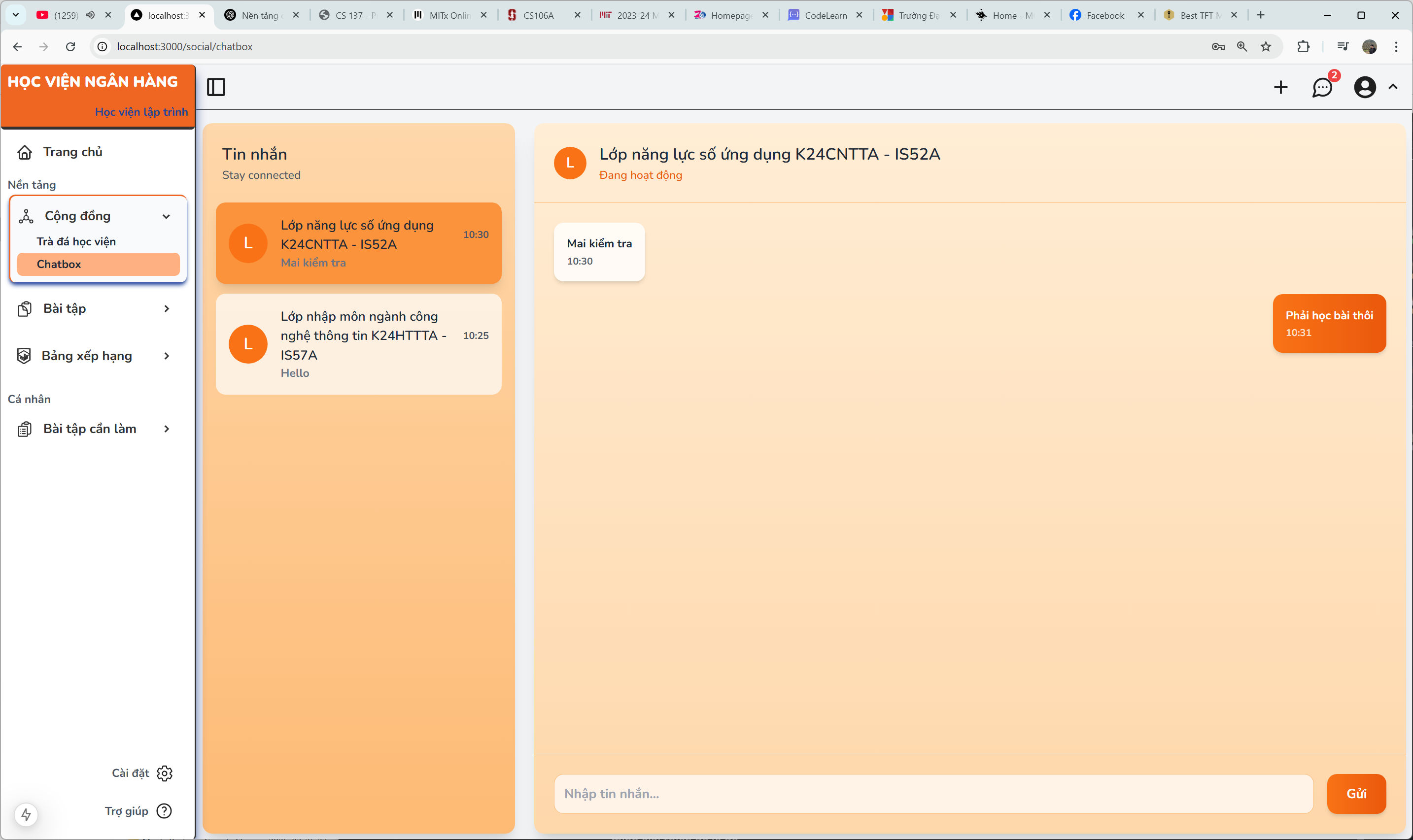
A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.3. Giao diện làm bài tập và submit



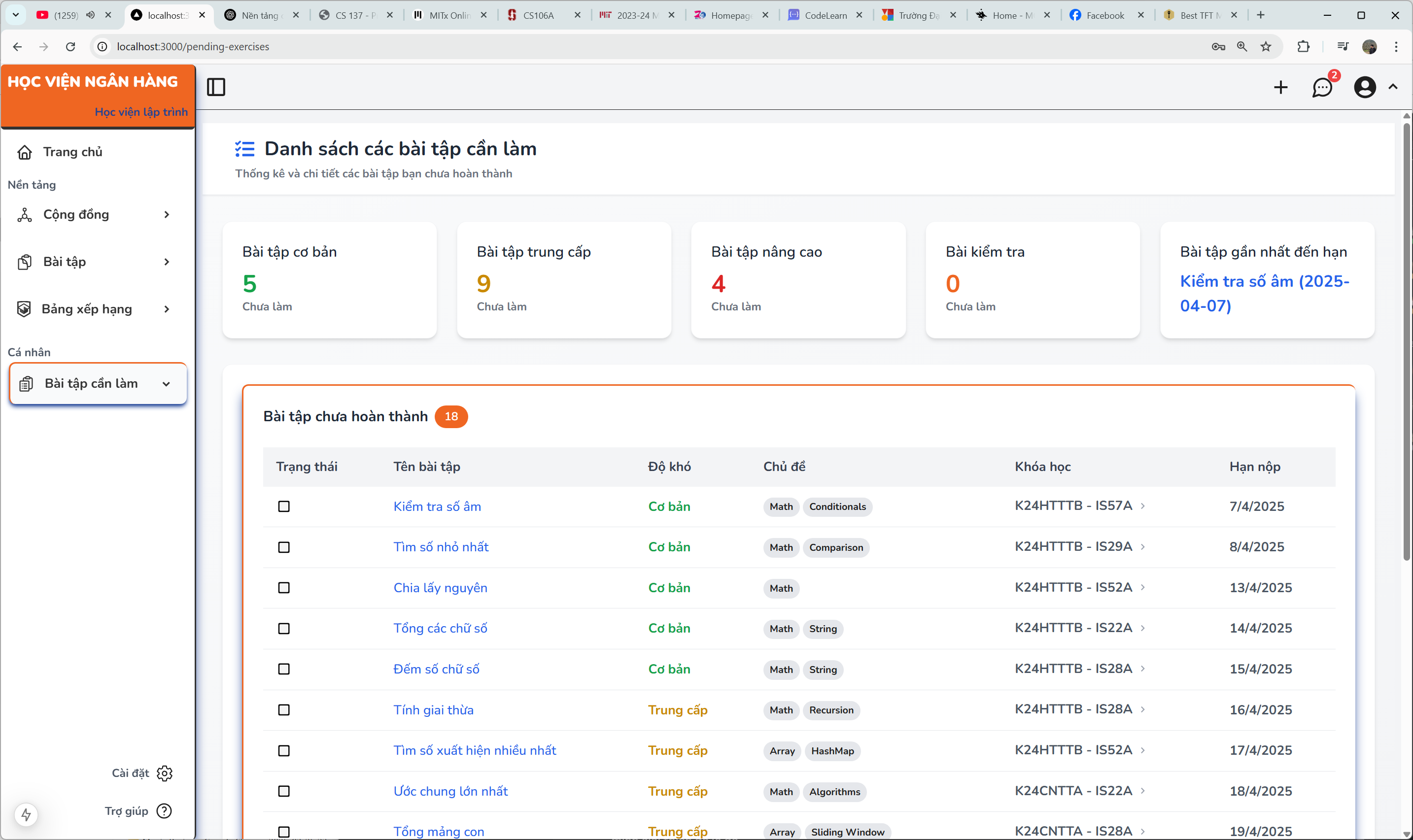
Hình 3.4. Diễn đàn cộng đồng sinh viên



Hình 3.5. Cuộc trò chuyện chung

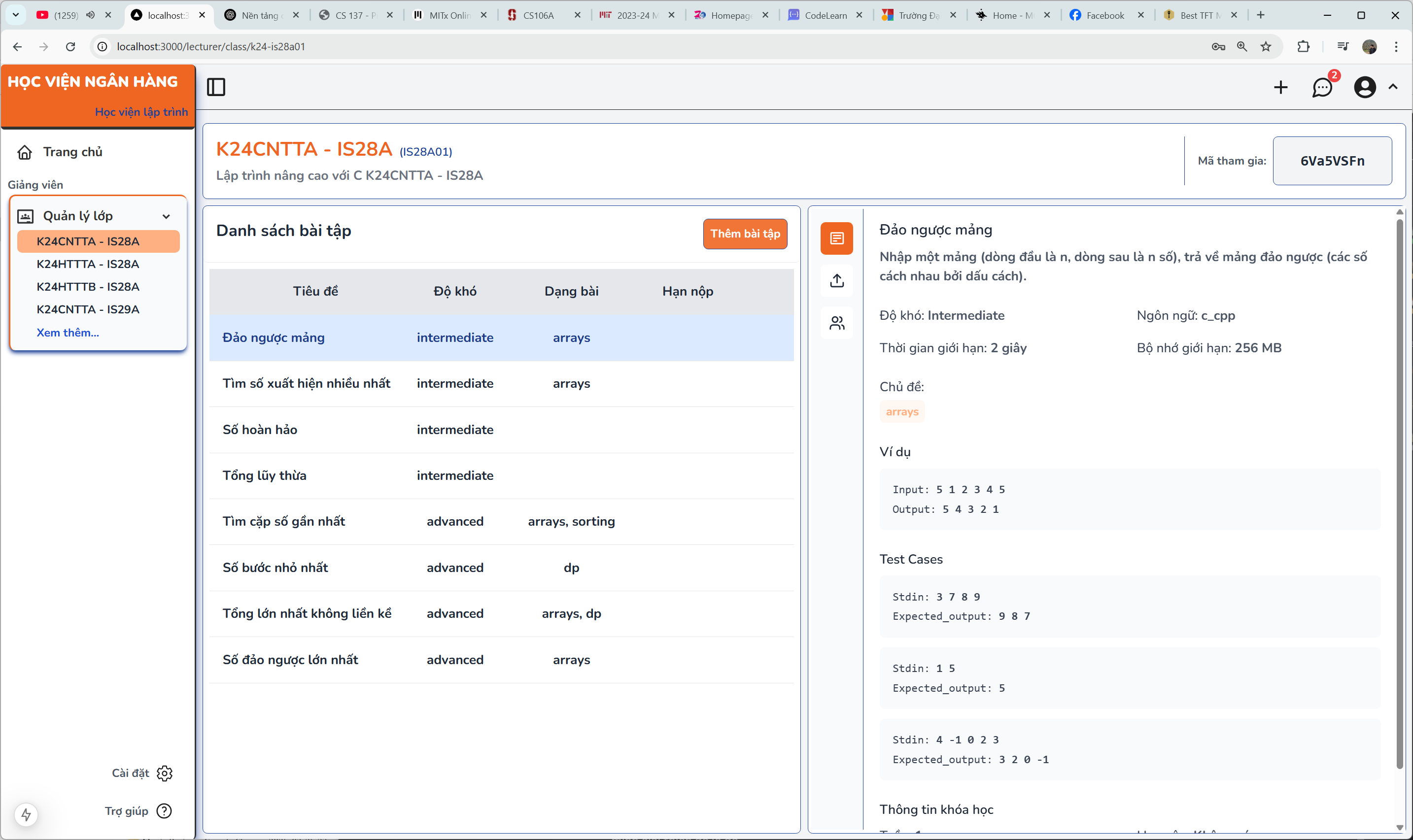


Hình 3.6. Bảng xếp hạng



Hình 3.7. Bài tập cần làm

### 3.2.2. Giao diện cho giảng viên



Hình 3.8. Quản lý bài tập

A screenshot of a computer

AI-generated content may be incorrect.

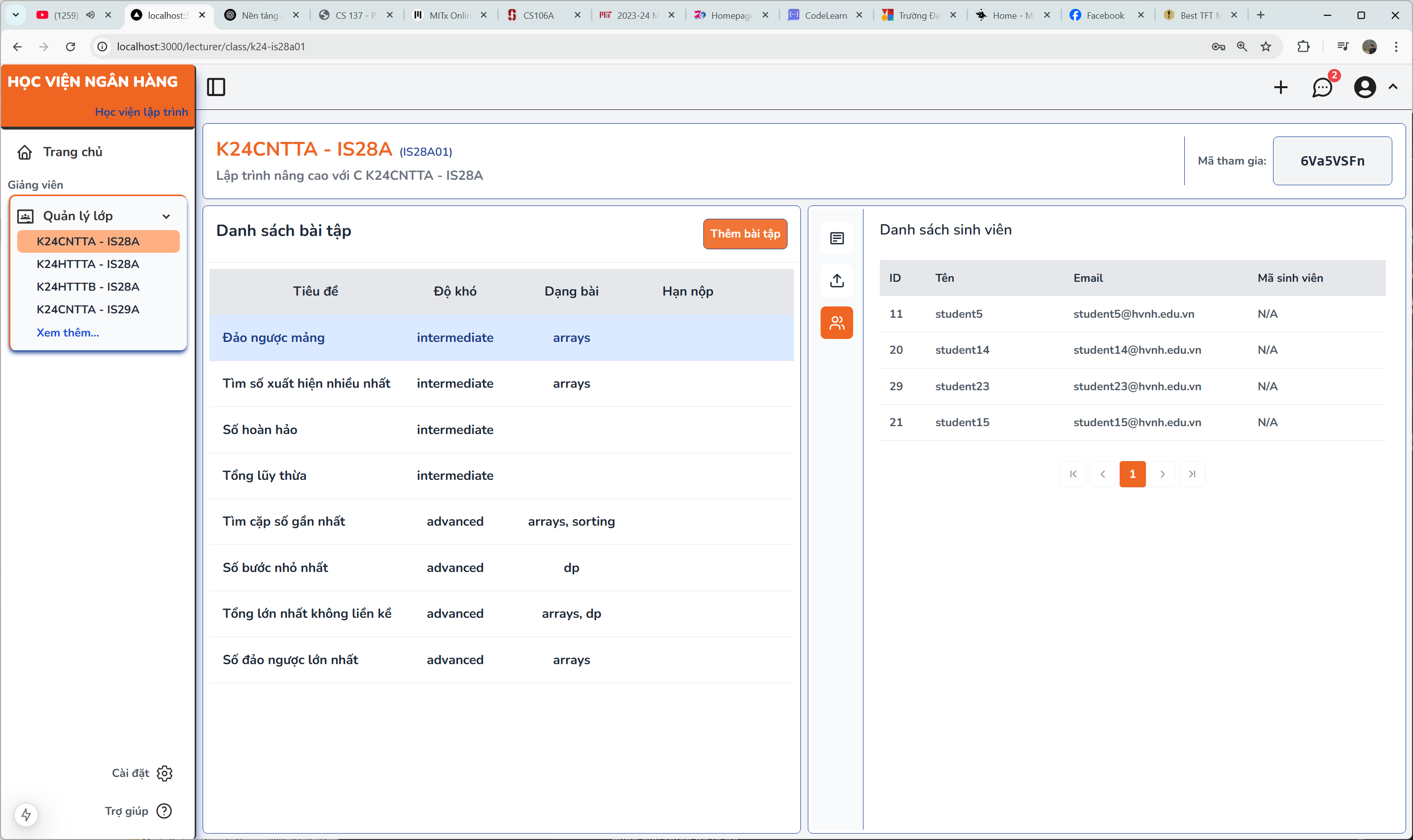
Hình 3. 9. Quản lý nộp bài của sinh viên

A screenshot of a computer

AI-generated content may be incorrect.

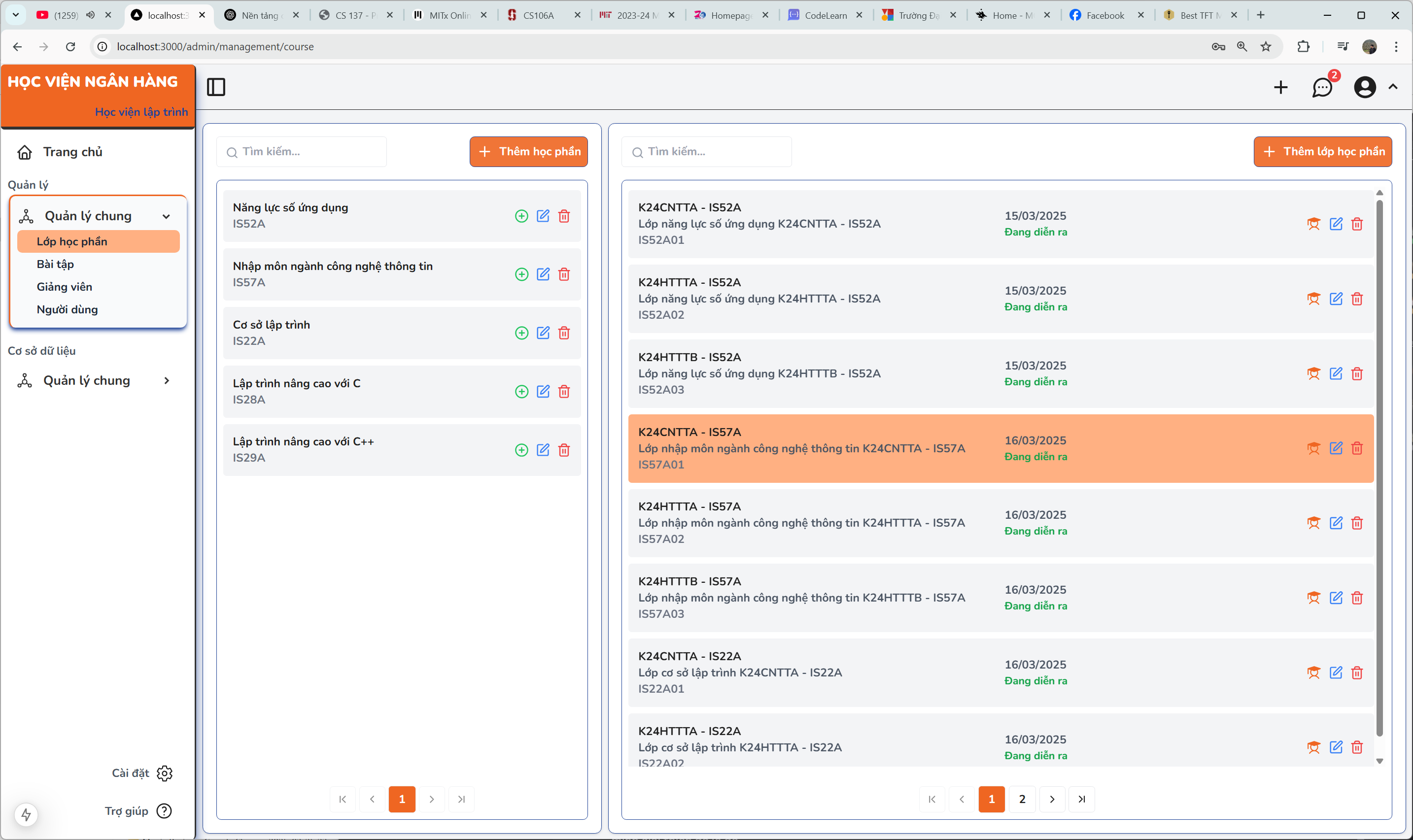
Hình 3.10. Quản lý nội dung nộp bài của sinh viên

Quản lý nội dung nộp bài của sinh viên cho từng bài

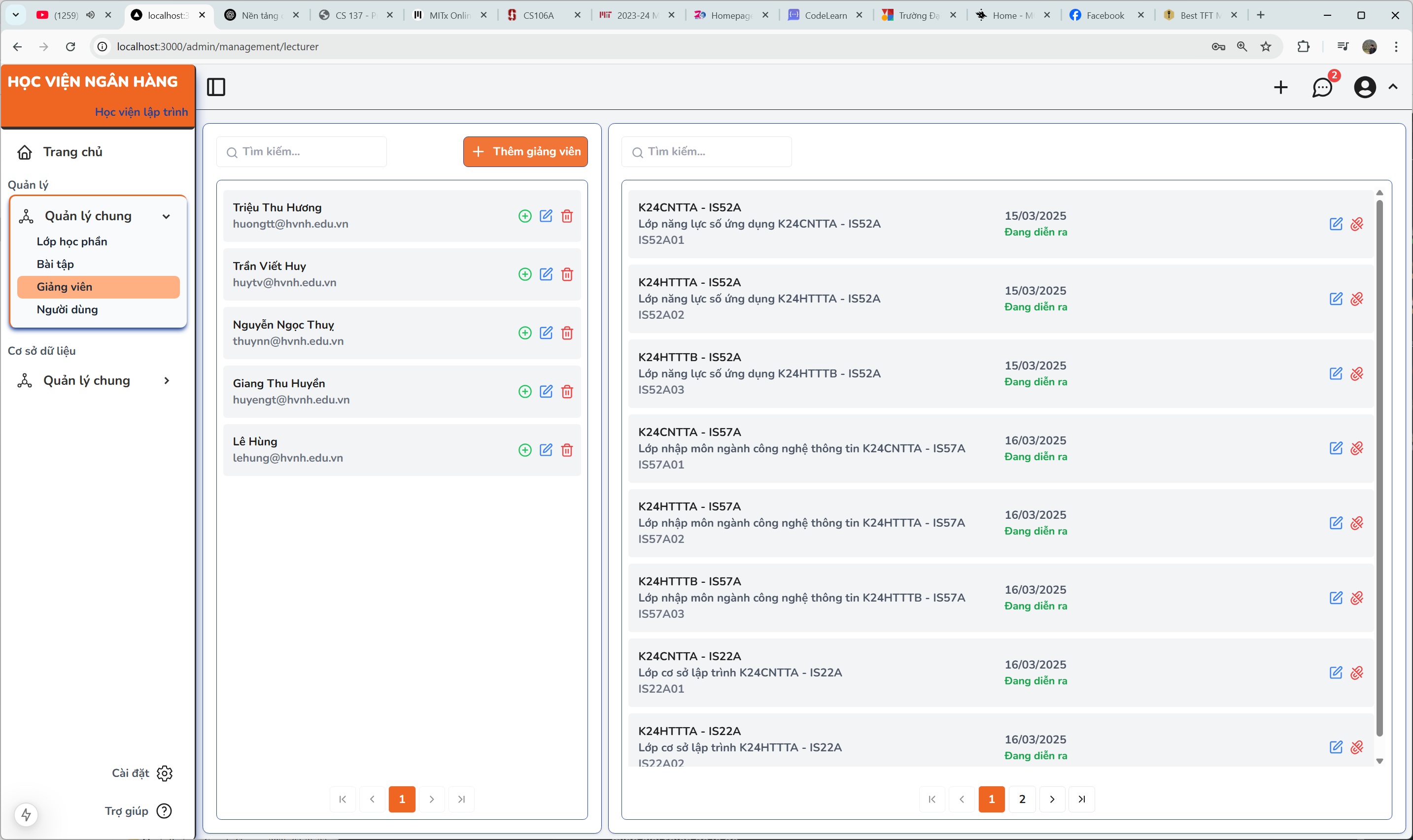


Hình 3.11. Quản lý sinh viên trong lớp

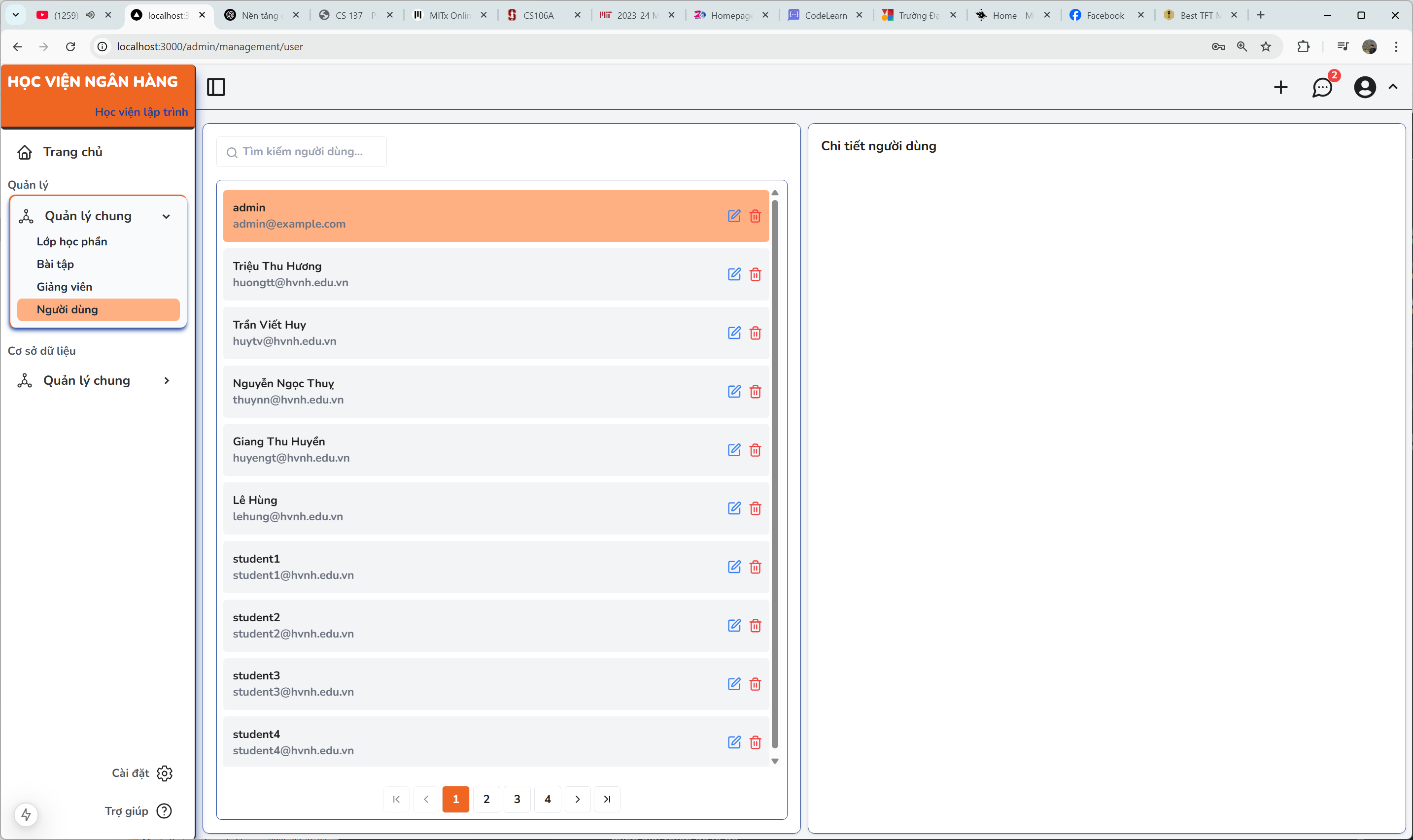
### 3.2.3. Giao diện cho admin



Hình 3.12. Quản lý các lớp học phần



Hình 3.13. Quản lý giảng viên



Hình 3.14. Quản lý người dùng

## 3.3. Triển khai Judge0

Judge0 đóng vai trò là engine thực thi và đánh giá mã nguồn, cho phép hệ thống tự động biên dịch, chạy chương trình, và so sánh đầu ra với các test case được định nghĩa trước. Nó được triển khai để hỗ trợ các ngôn ngữ lập trình chính trong chương trình đào tạo của Khoa, bao gồm C, C++, Java, và Python. Judge0 hoạt động như một dịch vụ độc lập, giao tiếp với backend qua API, đảm bảo quá trình chấm bài diễn ra nhanh chóng, chính xác, và an toàn trong môi trường cô lập.

### 3.3.1. Quy trình hoạt động của Judge0 trong hệ thống

Judge0 được tích hợp vào luồng xử lý bài nộp của sinh viên theo các bước sau:

1. **Nhận bài nộp từ sinh viên**:
   * Sinh viên viết mã nguồn trên giao diện Next.js (trình soạn thảo tích hợp) và nhấn nút "Nộp bài".
   * Frontend gửi yêu cầu POST đến endpoint /submit-exercise của backend Laravel, bao gồm:
     + source\_code: Mã nguồn do sinh viên nhập (ví dụ: chương trình C++ tính tổng hai số).
     + language: Ngôn ngữ lập trình được chọn (ví dụ: "cpp").
     + exercise\_id và course\_class\_id: Định danh bài tập và lớp học.
2. **Xử lý trên backend với Judge0**:
   * Backend lưu thông tin bài nộp vào bảng submissions trong MySQL, bao gồm user\_id (từ session), exercise\_id, course\_class\_id, source\_code, và language.
   * Backend lấy test case từ bài tập (lưu trong bảng exercises hoặc course\_exercise) và gửi yêu cầu đến Judge0 qua endpoint /submissions. Dữ liệu gửi đi bao gồm:
     + Mã nguồn (source\_code).
     + ID ngôn ngữ tương ứng trong Judge0 (ví dụ: 2 cho C++).
     + Đầu vào (stdin) từ test case (ví dụ: "2 3").
     + Đầu ra mong đợi (expected\_output, ví dụ: "5").
   * Judge0 thực thi mã nguồn trong container cô lập và trả về một judge0\_token (ví dụ: "xyz789"), được backend lưu vào cột judge0\_token trong bảng submissions.
3. **Lấy và xử lý kết quả từ Judge0**:
   * Backend gửi yêu cầu GET đến /submissions/{judge0\_token} mỗi giây (tối đa 5 lần) để lấy kết quả thực thi từ Judge0.
   * Kết quả trả về từ Judge0 bao gồm:
     + stdout: Đầu ra thực tế của chương trình (ví dụ: "5").
     + status.id: Trạng thái thực thi (3 = Accepted, 6 = Compilation Error, 11 = Time Limit Exceeded, v.v.).
     + time: Thời gian chạy (giây).
     + memory: Bộ nhớ sử dụng (KB).
   * Backend xử lý response của Judge0, và cập nhật bảng submissions với các cột status, execution\_time, memory\_used, và output.
4. **Hiển thị kết quả cho sinh viên**:
   * Backend gửi phản hồi JSON về frontend Next.js:
   * Frontend hiển thị kết quả trên giao diện, bao gồm trạng thái ("Accepted", "Wrong Answer", v.v.), điểm số, và chi tiết lỗi nếu có

### 3.3.2. Tích hợp với các thành phần khác của hệ thống

Judge0 không hoạt động độc lập mà được tích hợp chặt chẽ với các thành phần khác trong hệ thống:

* **Backend Laravel**: Judge0 giao tiếp với Laravel qua Guzzle HTTP Client để gửi và nhận dữ liệu. Laravel quản lý toàn bộ logic nghiệp vụ, từ lưu trữ bài nộp đến xử lý kết quả từ Judge0.
* **Cơ sở dữ liệu MySQL**: Bảng submissions lưu trữ thông tin bài nộp và kết quả từ Judge [**truncated** due to character limit]... được sử dụng để tham chiếu đến Judge0 và cập nhật trạng thái bài nộp.
* **Frontend Next.js**: Giao diện trình soạn thảo và hiển thị kết quả được xây dựng để tương tác trực tiếp với backend, đảm bảo sinh viên nhận phản hồi tức thời.

### 3.3.3. Tùy chỉnh Judge0 trong hệ thống

Judge0 được tùy chỉnh để phù hợp với đặc thù của dự án:

* **Giới hạn tài nguyên**: Thời gian thực thi tối đa là 2 giây và bộ nhớ tối đa 128MB, phù hợp với các **bài** tập cơ bản như lập trình C/C++, Java, Python trong chương trình đào tạo.
* **Hỗ trợ ngôn ngữ**: Chỉ kích hoạt các ngôn ngữ cần thiết (C, C++, Java, Python) để giảm tải hệ **thống** và đơn giản hóa lựa chọn cho sinh viên.
* **Quản lý đồng thời**: Sử dụng 4 worker để xử lý tối đa 4 bài nộp cùng lúc, đáp ứng quy mô lớp học 50-70 sinh viên.

## 3.4. Xây dựng mô đun

Hệ thống được xây dựng dựa trên các mô đun chính, mỗi mô đun đảm nhiệm một chức năng cốt lõi nhằm đáp ứng nhu cầu của sinh viên và giảng viên trong quá trình quản lý lớp học và đánh giá bài tập lập trình. Dưới đây là mô tả chi tiết về ba mô đun quan trọng: **Sinh viên nhập mã để vào lớp**, **Sinh viên nộp bài tập lập trình**, và **Giảng viên tạo bài tập mới**, bao gồm đầu vào, đầu ra và cách thức hoạt động.

### 3.4.1. Mô đun: Sinh viên nhập mã để vào lớp

Mô đun này cho phép sinh viên tham gia vào một lớp học phần bằng cách nhập mã lớp (course\_class\_join\_code) do giảng viên hoặc quản trị viên cung cấp. Đây là bước đầu tiên để sinh viên truy cập các bài tập và tài liệu liên quan trong lớp học.

* **Đầu vào:**
  + Mã lớp học (course\_class\_join\_code): Chuỗi ký tự duy nhất được tạo bởi hệ thống khi lớp học được thiết lập.
  + Thông tin xác thực của sinh viên: Đảm bảo sinh viên đã đăng nhập (session token hoặc cookie xác thực).
* **Quy trình xử lý:**
  + Sinh viên truy cập giao diện "Tham gia lớp học" trên trang chủ.
  + Sinh viên nhập mã lớp vào trường nhập liệu và nhấn nút "Tham gia".
  + Frontend gửi yêu cầu POST đến endpoint /join-class của backend Laravel, kèm theo mã lớp và thông tin xác thực.
  + Backend kiểm tra:
    - Mã lớp có tồn tại trong bảng course\_classes hay không.
    - Lớp học có đang ở trạng thái hoạt động (active = 1).
    - Sinh viên chưa là thành viên của lớp (kiểm tra bảng course\_attendant).
  + Nếu hợp lệ, backend thêm sinh viên vào lớp bằng cách tạo bản ghi mới trong bảng course\_attendant với vai trò "student".
  + Backend trả về thông báo thành công hoặc lỗi.
* **Đầu ra:**
  + Thành công: Thông báo "Đã tham gia lớp học thành công" và sinh viên được chuyển hướng đến giao diện lớp học.
  + Thất bại: Thông báo lỗi (ví dụ: "Mã lớp không hợp lệ" hoặc "Bạn đã tham gia lớp này").
* **Công nghệ sử dụng:**
  + Frontend: Next.js (giao diện nhập mã và xử lý phản hồi).
  + Backend: Laravel (xử lý yêu cầu, truy vấn cơ sở dữ liệu).
  + Cơ sở dữ liệu: MySQL (bảng course\_classes và course\_attendant).

### 3.4.2. Mô đun: Sinh viên nộp bài tập lập trình

Mô đun này cho phép sinh viên nộp mã nguồn lập trình để hệ thống tự động đánh giá dựa trên các test case được định nghĩa trước. Đây là chức năng cốt lõi giúp tự động hóa quá trình chấm điểm.

* **Đầu vào:**
  + Mã nguồn: Đoạn mã do sinh viên viết (text, hỗ trợ nhiều ngôn ngữ như C, C++, Java, Python).
  + Ngôn ngữ lập trình: Lựa chọn ngôn ngữ từ danh sách (ví dụ: "cpp", "java", "python3").
  + ID bài tập (exercise\_id): Định danh bài tập được giao trong lớp.
  + ID lớp học (course\_class\_id): Định danh lớp học mà sinh viên tham gia.
  + Thông tin xác thực: Session token để xác định sinh viên nộp bài.
* **Quy trình xử lý:**
  + Sinh viên truy cập giao diện bài tập trong lớp học, viết mã nguồn trên trình soạn thảo tích hợp.
  + Sinh viên nhấn nút "Nộp bài" để gửi mã nguồn.
  + Frontend gửi yêu cầu POST đến endpoint /submit-exercise của backend Laravel, kèm theo mã nguồn, ngôn ngữ, exercise\_id, course\_class\_id, và thông tin xác thực.
  + Backend thực hiện:
    - Lưu thông tin bài nộp vào bảng submissions (bao gồm user\_id, exercise\_id, course\_class\_id, source\_code, language).
    - Gửi mã nguồn và ngôn ngữ đến Judge0 API qua yêu cầu POST tới endpoint /submissions của Judge0.
    - Nhận judge0\_token từ Judge0 và lưu vào bảng submissions.
  + Backend polling Judge0 (hoặc dùng callback) để lấy kết quả thực thi:
    - Đầu ra (output), trạng thái (status), thời gian thực thi (execution\_time), bộ nhớ sử dụng (memory\_used).
  + Backend so sánh đầu ra với test case mong đợi, tính điểm và cập nhật bảng submissions.
  + Trả kết quả về frontend.
* **Đầu ra:**
  + Thành công: Kết quả chấm điểm (điểm số, trạng thái: "Accepted", "Wrong Answer", "Time Limit Exceeded", v.v.), chi tiết lỗi (nếu có), và thông báo "Nộp bài thành công".
  + Thất bại: Thông báo lỗi (ví dụ: "Lỗi cú pháp" hoặc "Hệ thống không thể xử lý").
* **Công nghệ sử dụng:**
  + Frontend: Next.js (trình soạn thảo mã, gửi yêu cầu).
  + Backend: Laravel (quản lý bài nộp, giao tiếp với Judge0).
  + Judge0 API: Thực thi và đánh giá mã nguồn.
  + Cơ sở dữ liệu: MySQL (bảng submissions).

### 3.4.3. Mô đun: Giảng viên tạo bài tập mới

Mô đun này cho phép giảng viên tạo bài tập lập trình mới, bao gồm đề bài và test case, để giao cho sinh viên trong lớp học.

* **Đầu vào:**
  + Tên bài tập: Chuỗi ký tự mô tả bài tập (ví dụ: "Tính tổng dãy số").
  + Mô tả: Nội dung chi tiết của bài tập (text hoặc HTML).
  + Test cases: Danh sách cặp đầu vào (input) và đầu ra mong đợi (expected\_output), ví dụ:
    - Test case 1: Input = "1 2 3", Expected Output = "6".
    - Test case 2: Input = "4 5", Expected Output = "9".
  + Thời hạn nộp bài (deadline): Ngày giờ cụ thể (DATETIME).
  + ID lớp học (course\_class\_id): Định danh lớp học để gắn bài tập.
  + Thông tin xác thực: Session token để xác định giảng viên.
* **Quy trình xử lý:**
  + Giảng viên truy cập giao diện "Quản lý lớp", chọn lớp học và nhấn "Thêm bài tập".
  + Giảng viên điền thông tin bài tập (tên, mô tả, test cases, thời hạn) vào form.
  + Frontend gửi yêu cầu POST đến endpoint /create-exercise của backend Laravel, kèm theo dữ liệu bài tập và thông tin xác thực.
  + Backend kiểm tra:
    - Giảng viên có quyền tạo bài tập trong lớp (role = lecturer trong course\_attendant).
    - Dữ liệu đầu vào hợp lệ (test cases có định dạng đúng, thời hạn không ở quá khứ).
  + Backend lưu bài tập vào bảng exercises (tên, mô tả) và bảng course\_exercise (liên kết với course\_class\_id, deadline).
  + Test cases được lưu vào một bảng riêng (nếu có) hoặc dưới dạng JSON trong exercises.
  + Backend trả về thông báo thành công hoặc lỗi.
* **Đầu ra:**
  + Thành công: Thông báo "Tạo bài tập thành công" và bài tập xuất hiện trong danh sách bài tập của lớp.
  + Thất bại: Thông báo lỗi (ví dụ: "Dữ liệu không hợp lệ" hoặc "Bạn không có quyền tạo bài tập").
* **Công nghệ sử dụng:**
  + Frontend: Next.js (form nhập liệu, gửi yêu cầu).
  + Backend: Laravel (xử lý logic, lưu trữ dữ liệu).
  + Cơ sở dữ liệu: MySQL (bảng exercises, course\_exercise).

### 3.4.4. Mô đun: Admin thêm học phần và lớp học phần

Cho phép admin tạo học phần (course) và lớp học phần (course class) để quản lý chương trình học.

* **Đầu vào:**
  + Thông tin học phần: Tên học phần (ví dụ: "Lập trình cơ bản"), mã học phần mô tả ngắn.
  + Thông tin lớp học phần: Mã lớp (course\_class\_join\_code), thời gian bắt đầu/kết thúc, trạng thái (active = 1/0).
  + Thông tin xác thực: Session token của admin.
* **Quy trình xử lý:**
  + Admin truy cập giao diện "Quản lý học phần" trong dashboard.
  + Điền thông tin học phần và lớp học phần vào form.
  + Frontend gửi yêu cầu POST đến endpoint /add-course-class của backend Laravel.
  + Backend:
    - Kiểm tra quyền admin (role = admin trong bảng users).
    - Xác nhận dữ liệu hợp lệ (mã học phần/lớp duy nhất, thời gian hợp lý).
    - Lưu học phần vào bảng courses (tên, mã, mô tả).
    - Lưu lớp học phần vào bảng course\_classes (mã lớp, thời gian, trạng thái, liên kết course\_id).
* **Đầu ra:**
  + Thành công: Thông báo "Thêm học phần và lớp học phần thành công", dữ liệu xuất hiện trong danh sách quản lý.
  + Thất bại: Thông báo lỗi (ví dụ: "Mã học phần đã tồn tại" hoặc "Dữ liệu không hợp lệ").
* **Công nghệ:**
  + Frontend: Next.js (form nhập liệu, hiển thị phản hồi).
  + Backend: Laravel (xử lý logic, lưu trữ).
  + Cơ sở dữ liệu: MySQL (bảng courses, course\_classes).

## 3.5. Thử nghiệm

Để đảm bảo các mô đun của hệ thống hoạt động đúng như thiết kế và đáp ứng yêu cầu chức năng, quá trình thử nghiệm được thực hiện bằng công cụ **Postman**. Postman là một công cụ phổ biến để kiểm tra API, cho phép gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) đến backend và nhận phản hồi, từ đó đánh giá tính chính xác và hiệu suất của từng mô đun. Các mô đun được thử nghiệm bao gồm: **Sinh viên nhập mã để vào lớp**, **Sinh viên nộp bài tập lập trình**, **Giảng viên tạo bài tập mới**, và **Admin thêm học phần và lớp học phần**. Dưới đây là chi tiết quá trình thử nghiệm từng mô đun.

### 3.5.1. Thử nghiệm mô đun: Sinh viên nhập mã để vào lớp

**Mục đích:** Kiểm tra khả năng sinh viên tham gia lớp học phần bằng cách nhập mã lớp thông qua API.

**Phương thức thử nghiệm:**

* **Công cụ:** Postman
* **Phương thức HTTP:** POST
* **Endpoint:** /join-class
* **Đầu vào:**
  + Header: Authorization: Bearer <session\_token> (token xác thực của sinh viên).
  + Body (JSON): {"course\_class\_join\_code": "ABC123"}
* **Đầu ra mong đợi:**
  + Status Code: 200 OK
  + Response (JSON):

{

  "status": "success",

  "message": "Đã tham gia lớp học thành công"

}

* + Trường hợp lỗi (mã lớp không hợp lệ):
    - Status Code: 400 Bad Request
    - Response:

{

  "status": "error",

  "message": "Mã lớp không hợp lệ"

}

**Quy trình thử nghiệm:**

1. Đăng nhập vào hệ thống bằng tài khoản sinh viên để lấy session token.
2. Mở Postman, thiết lập yêu cầu POST tới endpoint /join-class.
3. Thêm token vào header và mã lớp vào body.
4. Gửi yêu cầu và kiểm tra phản hồi từ server.
5. Thử lại với mã lớp không hợp lệ để kiểm tra xử lý lỗi.

**Kết quả thực tế:**

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.15. Thử nghiệm đăng nhập dành cho sinh viên

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.16. Thử nghiệm nhập mã để vào lớp danh cho sinh viên

### 3.5.2. Thử nghiệm mô đun: Sinh viên nộp bài tập lập trình

**Mục đích:** Đánh giá khả năng hệ thống nhận bài nộp từ sinh viên, gửi đến Judge0 để thực thi, và trả về kết quả chấm điểm.

**Phương thức thử nghiệm:**

* **Công cụ:** Postman
* **Phương thức HTTP:** POST
* **Endpoint:** /submit-exercise
* **Đầu vào:**
  + Header: Authorization: Bearer <session\_token> (token xác thực của sinh viên).
  + Body (JSON):

{

  "source\_code": "#include <iostream>\nint main() { int a, b; std::cin >> a >> b; std::cout << a + b; return 0; }",

  "language": "cpp",

  "exercise\_id": 1,

  "course\_class\_id": 1

}

* **Đầu ra mong đợi:**
  + Status Code: 200 OK
  + Response (JSON):

{

  "status": "success",

  "message": "Nộp bài thành công",

  "result": {

    "status": "Accepted",

    "score": 100,

    "execution\_time": 0.05,

    "memory\_used": 2048,

    "output": "5"

  }

}

* + Trường hợp lỗi (mã nguồn sai cú pháp):
    - Status Code: 200 OK
    - Response:

{

  "status": "success",

  "message": "Nộp bài thành công",

  "result": {

    "status": "Compilation Error",

    "score": 0,

    "error": "syntax error at line 2"

  }

}

**Quy trình thử nghiệm:**

1. Đăng nhập bằng tài khoản sinh viên để lấy session token.
2. Thiết lập yêu cầu POST tới /submit-exercise trong Postman.
3. Thêm token vào header và dữ liệu bài nộp (mã nguồn C++ tính tổng hai số) vào body.
4. Gửi yêu cầu và kiểm tra phản hồi từ Judge0 qua backend.
5. Thử lại với mã nguồn chứa lỗi cú pháp để kiểm tra xử lý lỗi.

**Kết quả thực tế:**

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 3.17. Thử nghiệm nộp bài tập lập trình cho sinh viên

### 3.5.3. Thử nghiệm mô đun: Giảng viên tạo bài tập mới

**Mục đích:** Xác minh khả năng giảng viên tạo bài tập lập trình mới và lưu vào hệ thống.

**Phương thức thử nghiệm:**

* **Công cụ:** Postman
* **Phương thức HTTP:** POST
* **Endpoint:** /create-exercise
* **Đầu vào:**
  + Header: Authorization: Bearer <session\_token> (token xác thực của giảng viên).
  + Body (JSON):

{

  "name": "Tính tổng dãy số",

  "description": "Viết chương trình tính tổng các số nhập từ bàn phím.",

  "test\_cases": [

    {

      "input": "1 2 3",

      "expected\_output": "6"

    },

    {

      "input": "4 5",

      "expected\_output": "9"

    }

  ],

  "deadline": "2025-04-15 23:59:00",

  "course\_class\_id": 1

}

* **Đầu ra mong đợi:**
  + Status Code: 200 OK
  + Response (JSON):

{

  "status": "success",

  "message": "Tạo bài tập thành công",

  "exercise\_id": 2

}

* + Trường hợp lỗi (dữ liệu không hợp lệ):
    - Status Code: 400 Bad Request
    - Response:

{

  "status": "error",

  "message": "Dữ liệu không hợp lệ"

}

**Quy trình thử nghiệm:**

1. Đăng nhập bằng tài khoản giảng viên để lấy session token.
2. Thiết lập yêu cầu POST tới /create-exercise trong Postman.
3. Thêm token vào header và thông tin bài tập vào body.
4. Gửi yêu cầu và kiểm tra phản hồi từ server.
5. Thử lại với dữ liệu không hợp lệ (ví dụ: thiếu test case) để kiểm tra xử lý lỗi.

**Kết quả thực tế:**

A screenshot of a computer program

AI-generated content may be incorrect.

Hình 3.18. Thử nghiệm thêm bài tập mới dành cho giảng viên

### 3.5.4. Thử nghiệm mô đun: Admin thêm học phần và lớp học phần

**Mục đích:** Kiểm tra khả năng admin tạo học phần và lớp học phần mới thông qua API.

**Phương thức thử nghiệm:**

* **Công cụ:** Postman
* **Phương thức HTTP:** POST
* **Endpoint:** /add-course-class
* **Đầu vào:**
  + Header: Authorization: Bearer <session\_token> (token xác thực của admin).
  + Body (JSON):

{

  "course": {

    "course\_code": "CS101",

    "name": "Lập trình cơ bản",

    "description": "Khóa học lập trình cơ bản với C++"

  },

  "course\_class": {

    "course\_class\_join\_code": "ABC123",

    "start\_date": "2025-04-01 00:00:00",

    "end\_date": "2025-06-30 23:59:00",

    "active": 1

  }

}

* **Đầu ra mong đợi:**
  + Status Code: 200 OK
  + Response (JSON):

{

  "status": "success",

  "message": "Thêm học phần và lớp học phần thành công",

  "course\_id": 1,

  "course\_class\_id": 1

}

* + Trường hợp lỗi (mã học phần trùng):
    - Status Code: 400 Bad Request
    - Response:

{

  "status": "error",

  "message": "Mã học phần đã tồn tại"

}

**Quy trình thử nghiệm:**

1. Đăng nhập bằng tài khoản admin để lấy session token.
2. Thiết lập yêu cầu POST tới /add-course-class trong Postman.
3. Thêm token vào header và thông tin học phần/lớp học phần vào body.
4. Gửi yêu cầu và kiểm tra phản hồi từ server.
5. Thử lại với mã học phần đã tồn tại để kiểm tra xử lý lỗi.

**Kết quả thực tế:**

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.19. Thử nghiệm thêm học phần dành cho Admin

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.20. Thử nghiệm thêm lớp học phần dành cho Admin

# KẾT LUẬN

Đồ án “***Xây dựng hệ thống quản lý và đánh giá bài tập lập trình***” đã hoàn thành mục tiêu đề ra, mang lại một giải pháp công nghệ hỗ trợ hiệu quả cho quá trình giảng dạy và học tập lập trình. Hệ thống đã được thiết kế và triển khai với các chức năng chính như quản lý lớp học, giao bài tập, nộp bài, và chấm điểm tự động, đáp ứng nhu cầu thực tiễn của Khoa Công nghệ Thông tin và Kinh tế số. Việc tích hợp công cụ Judge0 đã cho phép hệ thống thực thi và đánh giá mã nguồn một cách nhanh chóng, chính xác, đồng thời cung cấp phản hồi tức thời cho sinh viên, góp phần nâng cao hiệu quả học tập và giảm tải công việc cho giảng viên.

Quá trình thử nghiệm cho thấy các mô đun hoạt động ổn định, xử lý tốt các trường hợp đầu vào hợp lệ và không hợp lệ, đảm bảo tính tin cậy của hệ thống. Frontend được xây dựng bằng Next.js mang lại giao diện thân thiện, dễ sử dụng, trong khi backend Laravel cung cấp khả năng quản lý logic nghiệp vụ và dữ liệu mạnh mẽ. Cơ sở dữ liệu MySQL được thiết kế với các bảng quan hệ chặt chẽ, hỗ trợ lưu trữ và truy xuất thông tin hiệu quả. Kết quả này không chỉ đáp ứng yêu cầu tự động hóa quy trình đánh giá mà còn tạo tiền đề cho việc hiện đại hóa giáo dục lập trình tại Học viện Ngân hàng, phù hợp với xu hướng chuyển đổi số trong giáo dục.

Tuy nhiên, hệ thống vẫn còn tiềm năng để cải tiến và mở rộng nhằm đáp ứng tốt hơn nhu cầu trong tương lai. Một số hướng cải tiến cụ thể bao gồm:

1. **Tích hợp AI trong việc chấm bài của sinh viên:** Hiện tại, hệ thống sử dụng Judge0 để chấm bài dựa trên test case cố định, nhưng việc tích hợp trí tuệ nhân tạo (AI) có thể nâng cao khả năng đánh giá. AI có thể phân tích cấu trúc mã nguồn, phát hiện lỗi logic tinh vi ngay cả khi đầu ra đúng, và đánh giá phong cách lập trình (ví dụ: độ tối ưu của thuật toán, tính dễ đọc của mã). Điều này không chỉ giúp chấm điểm toàn diện hơn mà còn cung cấp gợi ý cải thiện cụ thể cho sinh viên, từ đó nâng cao kỹ năng lập trình một cách sâu sắc.
2. **Tích hợp hệ thống phát hiện sao chép (plagiarism detection):** Để đảm bảo tính công bằng, hệ thống có thể bổ sung tính năng sử dụng thuật toán so sánh mã nguồn (như Moss) nhằm phát hiện các trường hợp sao chép bài giữa các sinh viên. Điều này đặc biệt hữu ích trong các lớp học đông và các kỳ thi quan trọng.
3. **Hỗ trợ đa dạng ngôn ngữ lập trình và công cụ biên dịch:** Ngoài C, C++, Java, Python, hệ thống có thể mở rộng để hỗ trợ các ngôn ngữ khác như JavaScript, PHP hoặc R, đáp ứng nhu cầu của các môn học mới trong chương trình đào tạo. Đồng thời, tích hợp thêm các IDE trực tuyến để sinh viên thực hành mà không cần cài đặt môi trường trên máy cá nhân.

Những cải tiến trên không chỉ nâng cao trải nghiệm người dùng mà còn giúp hệ thống trở thành một công cụ toàn diện, phù hợp với xu hướng công nghệ hiện đại và yêu cầu đào tạo ngày càng cao của ngành công nghệ thông tin. Đồ án này là bước khởi đầu quan trọng, mở ra cơ hội nghiên cứu và phát triển sâu hơn trong tương lai, góp phần nâng cao chất lượng giáo dục tại Học viện Ngân hàng nói riêng và các cơ sở đào tạo công nghệ thông tin nói chung.

# TÀI LIỆU THAM KHẢO

AI Centre for Educational Technologies. (2024). *Coursemology* . Được truy lục từ Coursemology : https://aicet.comp.nus.edu.sg/projects/

Autograder. (không ngày tháng). Được truy lục từ https://eecs-autograder.github.io/autograder.io/#:~:text=Autograder.io%20is%20an%20open,of%20how%20to%20run%20them.

CodeSignal. (không ngày tháng). *CodeSignal Assessments*. Được truy lục từ CodeSignal Assessments: https://support.codesignal.com/hc/en-us/articles/22112130416535-Quick-Start-Guide-Assessments

Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic test-based assessment of programming: A review. *Journal on Educational Resources in Computing, 5(3)*, 4–21.

FPT Techinsight. (không ngày tháng). *CodeLearn*. Được truy lục từ CodeLearn: https://career.fpt-software.com/codelearn/

GeeksforGeeks. (2024). *GeeksforGeeks*. Được truy lục từ Stateful vs. Stateless Architecture: https://www.geeksforgeeks.org/stateful-vs-stateless-architecture/

Girardin, M. (2023, 12 11). *What Is HackerRank?* Được truy lục từ What Is HackerRank?: https://www.theforage.com/blog/skills/what-is-hackerrank

Lau, D. (2024). *Marmoset - Engineering*. Được truy lục từ Marmoset - Engineering: https://uwaterloo.atlassian.net/wiki/spaces/ECELI/pages/44119916579/Marmoset+-+Engineering

Miller, B., Ranum, D., & Pardo, A. (không ngày tháng). Dynamic test case generation for automatic assessment of programming exercises. *Journal of Computer Science Education, 29(2)*, 175-190.

Nguyen, H., Tran, M., & Le, V. (2021). Applying artificial intelligence in automatic grading systems for programming education. *IEEE Transactions on Learning Technologies.* .

PRAVEENKUMAR. (2025, 02 24). *USE OF LEET CODE*. Được truy lục từ USE OF LEET CODE: https://www.linkedin.com/pulse/use-leet-code-praveenkumar-m-tbduc

*Sphere Online Judge*. (không ngày tháng). Được truy lục từ Sphere Online Judge: https://www.spoj.com/PTIT/

*TestMyCode*. (không ngày tháng). Được truy lục từ TestMyCode: https://testmycode.github.io/

*VNOI Online Judge*. (không ngày tháng). Được truy lục từ VNOI Online Judge: https://oj.vnoi.info/problems/

Wikipedia. (2024). *CodinGame*. Được truy lục từ CodinGame: https://en.wikipedia.org/wiki/CodinGame#:~:text=CodinGame%20is%20a%20technology%20company,compete%20in%20multiplayer%20programming%20contests

Wikipedia. (2025). *Codeforces*. Được truy lục từ Codeforces: https://en.wikipedia.org/wiki/Codeforces#:~:text=Codeforces%20(Russian%3A%20%D0%9A%D0%BE%D1%83%D0%B4%D1%84%D0%BE%D1%80%D1%81%D0%B5%D1%81)%20is,University%20led%20by%20Mikhail%20Mirzayanov.

Wikipedia. (2025). *Moodle*. Được truy lục từ Moodle: https://en.wikipedia.org/wiki/Moodle

Wilcox, E., & Settle, A. (2018). Investigating the effect of automatic grading on student learning in introductory computer science courses. . *Proceedings of the 49th ACM Technical Symposium on Computer Science Education.* .