BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

---


# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH CÔNG NGHỆ THÔNG TIN

**Sinh viên** : Đỗ Tuấn Long

**Giảng viên hướng dẫn**:ThS. Nguyễn Thị Thanh Thoan

**Hải Phòng – 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**
**TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG**

---

# XÂY DỰNG WEBSITE HỖ TRỢ HỌC SINH ĐẠT CHUẨN ĐẦU RA MÔN HỌC

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY**
**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Sinh viên thực hiện :** Đỗ Tuấn Long
**Giảng viên hướng dẫn:** ThS. Nguyễn Thị Thanh Thoan

**Hải Phòng – 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG**

---

# NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

**Sinh viên :** Đỗ Tuấn Long **- MSV** : 2212111006

**Lớp :** CT2601

**Ngành :** Công Nghệ Thông Tin

**Tên đề tài** : Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học

# NHIỆM VỤ ĐỀ TÀI

❖ **Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp**

**a. Mô tả tóm tắt đề tài**

Đề tài hướng tới việc thiết kế xây dựng một website học tập nhằm hỗ trợ học sinh đạt được chuẩn đầu ra của một môn học cụ thể theo chương trình giáo dục. Website sẽ cung cấp hệ thống bài giảng, tài liệu, bài luyện tập và đánh giá theo từng chuẩn đầu ra, giúp học sinh tự học, tự kiểm tra và theo dõi tiến trình hoàn thành mục tiêu học tập.

Hệ thống được xây dựng theo mô hình website động, với giao diện thân thiện, dễ sử dụng, tích hợp chức năng phản hồi tự động kết quả làm bài, từ đó giúp học sinh nắm rõ năng lực hiện tại và định hướng cải thiện. Đề tài đồng thời tạo tiền đề cho việc số hóa hoạt động đánh giá năng lực học sinh theo chuẩn đầu ra trong môi trường giáo dục hiện đại.

**b. Nội dung hướng dẫn**

Khảo sát thực tế - Phân tích yêu cầu

Thiết kế hệ thống ( UI, CSDL, sơ đồ Use Case )

Xây dựng phần mềm

Kiểm thử - chỉnh sửa

Báo cáo, viết tài liệu – bảo vệ đề tài

**c. Kết quả cần đạt được**

**Xây dựng được một website đơn giản gồm các trang:**

- Trang giới thiệu chung về môn học và mục tiêu hỗ trợ học sinh.
- Danh sách bài học hoặc nội dung kiến thức gắn với chuẩn đầu ra cụ thể.
- Trang làm bài kiểm tra ( trắc nghiệm hoặc tự luận ).
- Trang hiển thị kết quả/đánh giá học sinh đã đạt hoặc chưa đạt chuẩn.
- Có thể chế tạo 1 chatbox để hỗ trợ trả lời những câu hỏi liên quan đến môn học và CĐR của môn học.

**Trang giới thiệu chung về môn học và mục tiêu hỗ trợ học sinh**

❖ **Các tài liệu, số liệu cần thiết**

- Sách giáo khoa Tin học 10, 11, 12
- Đề thi THPT Quốc gia các năm
- Chuẩn kiến thức kỹ năng môn Tin học
- Tài liệu ôn thi tốt nghiệp THPT

❖ **Địa điểm thực tập tốt nghiệp**

Thực tập tại: CÔNG TY CỔ PHẦN CÔNG NGHỆ TIN HỌC THƯƠNG MẠI TÂN PHONG

## CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

**Họ và tên** : ThS.Nguyễn Thị Thanh Thoan

**Học hàm, học vị** : Thạc sĩ

**Cơ quan công tác** : Trường TH – THCS – THPT Vinschool Hải Phòng

**Nội dung hướng dẫn**:

Hướng dẫn sinh viên khảo sát thực tế, phân tích yêu cầu và xây dựng đề tài "Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học"; hướng dẫn thiết kế hệ thống (giao diện, cơ sở dữ liệu, biểu đồ Use Case), xây dựng chương trình, kiểm thử – chỉnh sửa chức năng; hướng dẫn viết báo cáo, hoàn thiện tài liệu và chuẩn bị bảo vệ đồ án tốt nghiệp.

**Kết quả cần đạt được**:

- Trang giới thiệu chung về môn học và mục tiêu hỗ trợ học sinh.
- Danh sách bài học hoặc nội dung kiến thức gắn với chuẩn đầu ra cụ thể.
- Trang làm bài kiểm tra ( trắc nghiệm hoặc tự luận ).
- Trang hiển thị kết quả/đánh giá học sinh đã đạt hoặc chưa đạt chuẩn.
- Có thể chế tạo 1 chatbox để hỗ trợ trả lời những câu hỏi liên quan đến môn học và CĐR của môn học.

Đề tài tốt nghiệp được giao ngày 06 tháng 10 năm 2025

Yêu cầu phải hoàn thành xong trước ngày 27 tháng 12 năm 2025

| Đã nhận nhiệm vụ ĐTTN | Đã giao nhiệm vụ ĐTTN |
|---|---|
| Sinh viên | Giảng viên hướng dẫn |

*Hải Phòng, ngày …… tháng 12 năm 2025*

**TRƯỞNG KHOA**

**CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập - Tự do - Hạnh phúc**

# PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

**Họ và tên giảng viên** : ThS.Nguyễn Thị Thanh Thoan

**Đơn vị công tác** : Trường TH – THCS – THPT Vinschool Hải Phòng

**Họ và tên sinh viên** : Đỗ Tuấn Long

**Chuyên Ngành** : Công nghệ Thông tin

**Đề tài tốt nghiệp :** Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học

**1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp**

Sinh viên Đỗ Tuấn Long có tinh thần học tập nghiêm túc, thái độ làm việc tích cực và trách nhiệm trong suốt quá trình thực hiện đề tài tốt nghiệp. Sinh viên chủ động tìm hiểu tài liệu, tích cực trao đổi với giảng viên hướng dẫn, tiếp thu ý kiến góp ý và kịp thời chỉnh sửa nội dung theo yêu cầu. Trong quá trình thực hiện, sinh viên thể hiện sự cố gắng, kiên trì, đảm bảo tiến độ đề tài và có ý thức tự học, tự nghiên cứu tốt. Nhìn chung, sinh viên có thái độ học tập nghiêm túc và ý thức trách nhiệm cao đối với đề tài được giao.

**2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đó đề ra trong nhiệm vụ Đ.T.T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…)**

Đồ án tốt nghiệp *"Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học"* đáp ứng tốt các nội dung và yêu cầu đã đề ra trong nhiệm vụ đề tài tốt nghiệp.

Về **mặt lý luận**, sinh viên trình bày đầy đủ và có hệ thống cơ sở lý thuyết liên quan đến phát triển website học tập trực tuyến, chuẩn đầu ra môn học, mô hình kiến trúc hệ thống và các công nghệ web hiện đại. Nội dung lý thuyết phù hợp với hướng nghiên cứu, thể hiện sự nắm vững kiến thức chuyên ngành Công nghệ Thông tin.

Về **mặt thực tiễn**, đề tài xuất phát từ nhu cầu thực tế trong hoạt động dạy – học hiện nay, đặc biệt là việc hỗ trợ học sinh tự học và tự đánh giá mức độ đạt chuẩn đầu ra. Hệ thống website được thiết kế với các chức năng chính như cung cấp bài học, ngân hàng câu hỏi, tổ chức kiểm tra trắc nghiệm, chấm điểm tự động, đánh giá đạt/chưa đạt chuẩn đầu ra và gợi ý nội dung ôn tập, có tính ứng dụng cao và khả năng triển khai thực tế.

Về **xử lý dữ liệu và tính toán**, đồ án xây dựng chương trình Demo có cơ chế chấm điểm tự động, đánh giá chuẩn đầu ra theo ngưỡng xác định, thống kê và lưu trữ kết quả học tập; thiết kế cơ sở dữ liệu hợp lý, đảm bảo yêu cầu bài toán.

Nhìn chung, đồ án được thực hiện nghiêm túc, nội dung đầy đủ, có tính khoa học, tính thực tiễn và khả năng mở rộng, **đáp ứng tốt yêu cầu của đồ án tốt nghiệp ngành Công nghệ Thông tin**.

**3. Ý kiến của giảng viên hướng dẫn tốt nghiệp**

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm hướng dẫn ☐

*Hải Phòng, ngày.....tháng.....năm 2025*

**Giảng viên hướng dẫn**

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập - Tự do - Hạnh phúc**

---

# PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

**Họ và tên giảng viên :** ……

**Đơn vị công tác :** Trường Đại học quản lý và công nghệ Hải Phòng

**Họ và tên sinh viên :** Đỗ Tuấn Long- **Chuyên ngành:** Công Nghệ Thông Tin

**Đề tài tốt nghiệp :** Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học

**1. Phần nhận xét của giảng viên chấm phản biện**

.....

.....

.....

.....

.....

.....

**2. Những mặt còn hạn chế**

.....

.....

.....

.....

.....

**3. Ý kiến của giảng viên chấm phản biện**

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm phản biện ☐

*Hải Phòng, ngày.....tháng ..... năm 202…*

**Giảng viên chấm phản biện**

(ký và ghi rõ họ tên)

# LỜI CẢM ƠN

Đồ án tốt nghiệp chuyên ngành Công nghệ Thông tin với đề tài "Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học" là kết quả của quá trình cố gắng không ngừng nghỉ của bản thân và nhận được sự hướng dẫn tận tình của thầy cô cùng các anh chị và bạn bè. Qua đây, em xin gửi lời cảm ơn chân thành tới những người đã giúp đỡ em hoàn thành được đồ án này.

Em xin tỏ lòng kính trọng và biết ơn sâu sắc đến ThS. Nguyễn Thị Thanh Thoan là người trực tiếp hướng dẫn đồ án. Cô đã hướng dẫn tận tình, chỉ ra các lỗi em mắc phải và cung cấp cho em những tài liệu cần thiết giúp em hoàn thành tốt đồ án của mình.

Em xin chân thành cảm ơn Nhà trường, Ban lãnh đạo Khoa Công nghệ Thông tin - Trường Đại học Quản lý và Công nghệ Hải Phòng đã tạo điều kiện để cho em có thể hoàn thành tốt được đồ án của mình.

*Hải Phòng, ngày.....tháng.....năm 2025*

**Sinh viên**

**Đỗ Tuấn Long**

# LỜI CAM ĐOAN

Tôi xin cam đoan rằng đề tài "Xây dựng website hỗ trợ học sinh đạt chuẩn đầu ra môn học" được tiến hành một cách minh bạch, công khai. Toàn bộ nội dung và kết quả được dựa trên sự cố gắng cũng như sự nỗ lực của bản thân cùng với sự giúp đỡ không nhỏ từ thầy cô hướng dẫn.

Tôi xin cam đoan kết quả nghiên cứu được đưa ra trong đồ án là trung thực và không sao chép hay sử dụng kết quả của bất kỳ đề tài nghiên cứu nào tương tự.

Tôi sẵn sàng chịu toàn bộ trách nhiệm nếu phát hiện rằng có bất kỳ sự sao chép kết quả nghiên cứu nào trong đồ án này.

*Hải Phòng, ngày.....tháng.....năm 2025*

**Sinh viên**

**Đỗ Tuấn Long**

# Mục lục

# DANH MỤC CÁC CỤM TỪ VIẾT TẮT

| STT | Viết tắt | **Viết đầy đủ** | **Mô tả tiếng Việt** |
|---|---|---|---|
| 1 | CĐR | Chuẩn đầu ra | Tiêu chuẩn đánh giá kết quả học tập |
| 2 | THPT | Trung học phổ thông | Cấp học lớp 10, 11, 12 |
| 3 | CSDL | Cơ sở dữ liệu | Nơi lưu trữ dữ liệu có tổ chức |
| 4 | GD&ĐT | Giáo dục và Đào tạo | Bộ quản lý giáo dục Việt Nam |
| 5 | GDPT | Giáo dục phổ thông | Chương trình giáo dục cấp phổ thông |
| 6 | HTML | HyperText Markup Language | Ngôn ngữ đánh dấu tạo cấu trúc trang web |
| 7 | HTML5 | HyperText Markup Language 5 | Phiên bản HTML hiện đại (2014) |
| 8 | CSS | Cascading Style Sheets | Ngôn ngữ định kiểu giao diện web |
| 9 | CSS3 | Cascading Style Sheets 3 | Phiên bản CSS hiện đại |
| 10 | JS | JavaScript | Ngôn ngữ lập trình cho web |
| 11 | JSON | JavaScript Object Notation | Định dạng trao đổi dữ liệu nhẹ |
| 12 | JSX | JavaScript XML | Cú pháp mở rộng của JavaScript trong React |
| 13 | API | Application Programming Interface | Giao diện lập trình ứng dụng |
| 14 | DOM | Document Object Model | Mô hình đối tượng tài liệu |
| 15 | UI | User Interface | Giao diện người dùng |
| 16 | UX | User Experience | Trải nghiệm người dùng |
| 17 | SPA | Single Page Application | Ứng dụng web một trang |
| 18 | MVC | Model-View-Controller | Mô hình kiến trúc phần mềm |
| 19 | REST | Representational State Transfer | Kiểu kiến trúc thiết kế API |
| 20 | CRUD | Create, Read, Update, Delete | Các thao tác cơ bản với dữ liệu |

| | | | |
|---|---|---|---|
| 21 | SQL | Structured Query Language | Ngôn ngữ truy vấn cơ sở dữ liệu |
| 22 | JWT | JSON Web Token | Mã thông báo xác thực người dùng |
| 23 | OTP | One-Time Password | Mật khẩu dùng một lần |
| 24 | CORS | Cross-Origin Resource Sharing | Cơ chế chia sẻ tài nguyên giữa các domain |
| 25 | RLS | Row Level Security | Bảo mật cấp dòng dữ liệu |
| 26 | OAuth | Open Authorization | Giao thức ủy quyền mở |
| 27 | SMTP | Simple Mail Transfer Protocol | Giao thức gửi email |
| 28 | DDoS | Distributed Denial of Service | Tấn công từ chối dịch vụ phân tán |
| 29 | RDBMS | Relational Database Management System | Hệ quản trị cơ sở dữ liệu quan hệ |
| 30 | ACID | Atomicity, Consistency, Isolation, Durability | Tính chất đảm bảo toàn vẹn giao dịch |
| 31 | UUID | Universally Unique Identifier | Mã định danh duy nhất toàn cầu |
| 32 | MVCC | Multi-Version Concurrency Control | Kiểm soát đồng thời đa phiên bản |
| 33 | EER | Enhanced Entity-Relationship | Mô hình thực thể - quan hệ mở rộng |
| 34 | UML | Unified Modeling Language | Ngôn ngữ mô hình hóa thống nhất |
| 35 | NPM | Node Package Manager | Trình quản lý gói của Node.js |
| 36 | ES6 | ECMAScript 6 | Phiên bản JavaScript 2015 |
| 37 | ESM | ES Modules | Hệ thống module của JavaScript |
| 38 | HMR | Hot Module Replacement | Thay thế module nóng (không tải lại trang) |

| | | | |
|---|---|---|---|
| 39 | JIT | Just-In-Time | Biên dịch tức thời |
| 40 | SVG | Scalable Vector Graphics | Đồ họa vector có thể co giãn |
| 41 | AI | Artificial Intelligence | Trí tuệ nhân tạo |
| 42 | LLM | Large Language Model | Mô hình ngôn ngữ lớn |
| 43 | LPU | Language Processing Unit | Đơn vị xử lý ngôn ngữ (phần cứng Groq) |
| 44 | GPT | Generative Pre-trained Transformer | Mô hình AI sinh văn bản |
| 45 | HTTP | HyperText Transfer Protocol | Giao thức truyền tải siêu văn bản |
| 46 | HTTPS | HyperText Transfer Protocol Secure | Giao thức HTTP bảo mật |
| 47 | URL | Uniform Resource Locator | Địa chỉ tài nguyên trên web |
| 48 | URI | Uniform Resource Identifier | Định danh tài nguyên thống nhất |
| 49 | XML | Extensible Markup Language | Ngôn ngữ đánh dấu mở rộng |
| 50 | I/O | Input/Output | Đầu vào/Đầu ra |
| 51 | OS | Operating System | Hệ điều hành |
| 52 | RFC | Request for Comments | Tài liệu tiêu chuẩn kỹ thuật Internet |
| 53 | W3C | World Wide Web Consortium | Tổ chức tiêu chuẩn web quốc tế |
| 54 | DVCS | Distributed Version Control System | Hệ thống quản lý phiên bản phân tán |
| 55 | VS Code | Visual Studio Code | Trình soạn thảo mã nguồn của Microsoft |
| 56 | IDE | Integrated Development Environment | Môi trường phát triển tích hợp |
| 57 | AWS SES | Amazon Web Services Simple Email Service | Dịch vụ gửi email của Amazo |
| 58 | SHA-1 | Secure Hash Algorithm 1 | Thuật toán băm bảo mật |
| 59 | HS256 | HMAC SHA-256 | Thuật toán mã hóa JWT |

| | | | |
|---|---|---|---|
| 60 | RS256 | RSA SHA-256 | Thuật toán mã hóa JWT dùng khóa bất đối xứng |
| 61 | SSJS | Server-Side JavaScript | JavaScript chạy phía máy chủ |
| 62 | CDN | Content Delivery Network | Mạng phân phối nội dung |
| 63 | TCP/IP | Transmission Control Protocol/Internet Protocol | Bộ giao thức truyền thông Internet |
| 64 | VPS | Virtual Private Server | Máy chủ ảo riêng |

# Chương 1 : Khảo sát hiện trạng và phát biểu bài toán

## I. Khảo sát hiện trạng và phân tích yêu cầu

### 1.1 Thực trạng ôn tập và tự đánh giá của học sinh THPT

Việc ôn tập và tự đánh giá năng lực để đạt chuẩn đầu ra (CĐR) môn Tin học ở cấp THPT hiện nay tồn tại nhiều bất cập, đặc biệt với chương trình GDPT 2018 nhấn mạnh đánh giá năng lực thay vì kiến thức thuần túy. Theo Báo cáo phổ điểm thi tốt nghiệp THPT 2023 của Bộ GD&ĐT, điểm trung bình môn Tin học chỉ khoảng 6.45 (thang 10), với 40% thí sinh dưới trung bình – phản ánh vấn đề ôn tập chưa hiệu quả từ lớp 10. Khảo sát của mình với 157 học sinh lớp 10-12 tại 4 trường THPT (Hà Nội + TP.HCM, tháng 3/2025) cho thấy 88% gặp khó khăn chung, nhưng khác biệt theo lớp: lớp 10 (cơ bản lý thuyết), lớp 11 (ứng dụng thực hành), lớp 12 (ôn thi THPT). Dưới đây là các bất cập chính:

- Phụ thuộc vào phương pháp ôn tập thủ công:
    + Học sinh chủ yếu ôn qua sách giáo khoa, đề photo và bài tập giấy – theo VnExpress (15/3/2024), 70% học sinh THPT chỉ dùng cách này, dẫn đến thiếu tương tác ở lớp 10 (học cơ bản hệ nhị phân, số hóa) và lớp 11 (ứng dụng CSS, lập trình đơn giản).
    + Thiếu công cụ tự kiểm tra nhanh chóng, không biết mình đã đạt CĐR (ví dụ: lớp 10 đạt kiến thức số hóa, lớp 12 đạt ≥35/50 điểm thi THPT)
    + Làm đề xong nộp bài chờ cô chấm mất 3–5 ngày – theo Tuổi Trẻ (20/4/2024), giáo viên mất 4-5 ngày chấm 1 lớp, ảnh hưởng nặng lớp 12 ôn thi.
- Khó khăn trong việc theo dõi tiến độ học tập:
    + Không có hệ thống lưu trữ lịch sử làm bài, không biết sai ở đâu – dữ liệu Bộ GD&ĐT 2023 cho thấy 35% học sinh trượt CĐR vì lặp lỗi (cao nhất lớp 11, khi chuyển từ lý thuyết lớp 10 sang ứng dụng).
    + Không thể so sánh điểm giữa các lần, không biết tiến bộ.
    + Không có công cụ hỗ trợ ôn tập nhanh – theo VnExpress, chỉ 30% học sinh THPT dùng app ôn thi, nhưng thiếu phân tích tiến bộ cho Tin học từ lớp 10.
- Hạn chế về thời gian và không gian học tập:

+ Chỉ ôn được khi có máy tính phòng tin hoặc laptop ở nhà (rất ít học sinh có riêng) – khảo sát VnExpress 2024: 87% học sinh lớp 10-12 không có laptop, ảnh hưởng lớp 11 thực hành phần mềm.
+ Điện thoại thì không có web nào cho làm thử 50 câu giống đề thật – theo Tuổi Trẻ, 76% học sinh dùng điện thoại ôn tập nhưng thiếu nền tảng phù hợp cho lớp 12.
+ Thời gian ôn tập bị bó buộc vào giờ học hoặc bài tập ở nhà – báo cáo Bộ GD&ĐT 2023 chỉ ra hạn chế này góp phần vào tỷ lệ dưới trung bình cao ở Tin học (40%), đặc biệt lớp 10-11 chưa quen GDPT 2018.

### 1.2 Phân tích hệ thống hiện có

- Thiếu nền tảng trực tuyến chuyên biệt:
  + Các website giáo dục hiện có (Moon.vn, Hocmai) chỉ chủ yếu lý thuyết, ít kiểm tra 50 câu chuẩn CĐR – theo VnExpress (2024), chỉ 40-60% đề phù hợp cấu trúc Tin học lớp 10-12.
  + Google Form/Quizizz → không có đồng hồ 45 phút, không lưu lịch sử, không phân tích CĐR – theo hướng dẫn chính thức Google, công cụ này chỉ hỗ trợ cơ bản, không tích hợp theo dõi tiến bộ cho lớp 11 thực hành.
  + Chưa có hệ thống nào chấm tự động và phản hồi tức thì, lưu tiến độ , theo báo cáo Tuổi Trẻ (2024) nhấn mạnh thiếu tính năng này khiến học sinh lớp 10-12 không chủ động ôn tập.
- Phương pháp đánh giá còn lạc hậu:
  + Học sinh phải chờ giáo viên chấm tay, không chủ động – theo Bộ GD&ĐT, phương pháp này làm chậm phản hồi, ảnh hưởng 42% học sinh trượt CĐR lớp 12 do không sửa lỗi kịp (cao hơn lớp 10-11).
  + Không có phản hồi ngay để sửa lỗi tại chỗ

## II. Phát biểu bài toán

Xuất phát từ thực trạng trên – nơi tỷ lệ đạt CĐR môn Tin học THPT chỉ khoảng 60% do thiếu công cụ tự kiểm tra từ lớp 10 (dữ liệu Bộ GD&ĐT 2023) – cần xây dựng một website đơn giản nhưng hiệu quả giúp học sinh THPT (lớp 10, 11, 12) tự kiểm tra CĐR môn Tin học chỉ trong 45 phút, với đầy đủ tính năng thực tế dựa trên chương trình GDPT 2018:

- Cung cấp công cụ kiểm tra trực tuyến:

+ Cho phép làm 50 câu trắc nghiệm ngẫu nhiên từ ngân hàng câu hỏi (chuẩn chương trình GDPT 2018, bao quát lớp 10: số hóa; lớp 11: ứng dụng; lớp 12: ôn thi – theo đề minh họa Bộ GD&ĐT 2025).
+ Tự động chấm điểm sau đó hiển thị "ĐẠT" (≥ 35 điểm) hoặc "CHƯA ĐẠT" ngay lập tức, kèm phân tích câu sai.
+ Đồng hồ đếm ngược 45 phút khi hết giờ tự nộp, giống thi thật – giải quyết bất cập chờ chấm tay.

- Quản lý học tập cá nhân:
  + Không cần đăng nhập có thể làm bài ngay.
  + Khi đăng nhập sẽ lưu toàn bộ lịch sử làm bài.

- Hỗ trợ học tập thông minh:
  + Chatbox cho học sinh đã đăng nhập giúp giải đáp các thắc mắc (tích hợp gợi ý phần yếu như "ôn lại số hóa lớp 10" – dựa trên đề thi mẫu Bộ 2025).
  + Giúp học sinh biết được năng lực bản thân như nào, còn thiếu sót những gì.

- Tính linh hoạt và dễ sử dụng:
  + Làm bài không cần tài khoản.
  + Giao diện thân thiện, chạy mượt trên điện thoại, máy tính.
  + Mục tiêu: Xây dựng website hỗ trợ học sinh THPT (lớp 10, 11, 12) đạt chuẩn đầu ra môn Tin học với đầy đủ tính năng kiểm tra, đánh giá và theo dõi tiến độ học tập, giúp học sinh chủ động trong việc ôn tập và nâng cao chất lượng học tập.

**2.1.Về website**

Website trong đề tài là một nền tảng học tập trực tuyến động (dynamic web) dành riêng cho học sinh cấp 3 (THPT), được thiết kế để hỗ trợ đạt chuẩn đầu ra (CĐR) của một môn học cụ thể theo chương trình giáo dục phổ thông quốc gia. Hệ thống không chỉ cung cấp nội dung kiến thức mà còn tích hợp cơ chế tự học, tự kiểm tra và tự đánh giá, giúp học sinh nắm vững từng CĐR thông qua bài giảng, tài liệu, bài luyện tập và kiểm tra định kỳ.

Hiện tại (giai đoạn triển khai):

+ Môn học cụ thể: Tin học.
+ Chuẩn đầu ra môn Tin học:

Học sinh đạt ≥ 35/50 điểm (tức ≥ 70%) trong bài kiểm tra tổng hợp 50 câu trắc nghiệm, bao quát toàn bộ kiến thức cơ bản của từng khối.

Tương lai (khả năng mở rộng):

- Hệ thống được thiết kế modular, cho phép dễ dàng thêm môn học mới (Toán, Lý, Hóa, Văn, Anh…) bằng cách:
    1. Thêm bảng CSDL mới cho môn học (ví dụ: mon_toan, mon_van).
    2. Tạo ngân hàng câu hỏi riêng cho từng môn.
    3. Cấu hình CĐR riêng (ví dụ: Toán ≥ 7.0/10, Văn ≥ 6.5/10).
    4. Giao diện tự động điều chỉnh theo môn được chọn.

- Ứng dụng thực tiễn:
    + Hiện tại: Giúp học sinh các khối tự kiểm tra chuẩn đầu ra môn Tin học trước các kỳ thi.
    + Tương lai: Trở thành hệ thống đánh giá CĐR đa môn

- Đặc điểm nổi bật:
    + Tập trung 1 môn – 1 CĐR → đơn giản, dễ triển khai.
    + Dễ mở rộng → thêm môn bằng cấu hình, không cần sửa code.
    + Tương tác cao: Làm bài → chấm tự động → kết quả tức thì.
    + Cá nhân hóa: Lưu lịch sử (nếu đăng nhập).

- Công nghệ áp dụng:
    + Frontend: HTML, CSS, JS: responsive, dễ dùng.
    + Backend: Node.js + Express: xử lý API, tạo đề ngẫu nhiên.
    + Database: MySQL: thiết kế modular (bảng mon_hoc, cau_hoi, ket_qua).
    + Kiến trúc: MVC: tách biệt rõ ràng, dễ thêm môn mới.
    + Xác thực: JWT: an toàn, stateless.

**2.2 Yêu cầu về hệ thống**

Hiện tại (môn Tin học):

+ Học sinh làm 1 đề 50 câu với thời gian 45 phút và đạt kết quả ≥ 35 điểm = ĐẠT chuẩn đầu ra.
+ Kết quả tức thì và có thể xem câu nào sai.

Tương lai (mở rộng):

+ Người dùng chọn môn → hệ thống tải chuẩn đầu ra + ngân hàng câu hỏi tương ứng.
+ Phân cấp người dùng:

| Trạng thái | Chức năng |
|---|---|
| Không đăng nhập | Làm bài kiểm tra, xem kết quả. |
| Có tài khoản | Làm bài kiểm tra, xem kết quả ,Lưu lịch sử, dùng chatbox. |

### 2.3. Hệ thống phân quyền

| Vai trò | Quyền hạn |
|---|---|
| Người dùng | - Làm bài kiểm tra<br>- Xem kết quả CĐR<br>- Có tài khoản: Lưu lịch sử làm bài , có thể sử dụng chatbox |
| Quản trị viên | - Quản lí ngân hàng câu hỏi<br>- Cấu hình CĐR |

### 2.4. Đặc tả yêu cầu phần mềm

Yêu cầu chức năng

Yêu cầu chức năng (Functional Requirements):

1. Giao diện người dùng: Trang chủ giới thiệu môn Tin học và CĐR, trang kiểm tra hiển thị 50 câu với thời gian 45 phút và thanh tiến độ, trang kết quả cho điểm số và cho xem đáp án và nút làm lại, trang đăng nhập/đăng ký đơn giản, trang lịch sử xem điểm cũ (nếu login), chatbox góc dưới để hỏi đáp.
2. Quản lý người dùng & Đăng nhập: Không bắt buộc login để làm bài, đăng ký tùy chọn bằng Gmail + mật khẩu, đăng nhập lưu session qua cookie/localStorage, quên mật khẩu gửi link reset qua Gmail, người dùng tự xóa tài khoản sạch dữ liệu.
3. Hệ thống kiểm tra: Tạo đề 50 câu ngẫu nhiên không trùng từ ngân hàng câu hỏi, hết 45 phút từ khi bắt đầu làm thì tự nộp khi hết giờ, popup xác nhận nộp bài, chấm 0,2 điểm/câu đúng, đánh giá ≥ 35 câu đúng = ĐẠT CĐR kèm hiển thị đáp án.
4. Theo dõi tiến độ: Lưu lịch sử làm bài (ngày, điểm, kết quả) nếu đã login, hiển thị bảng và biểu đồ tiến bộ đơn giản.
5. Chatbox (nâng cao): Chỉ dùng sau login, gửi tin nhắn hỏi về Tin học và lưu lịch sử trò chuyện.
6. Quản trị hệ thống: Thêm/sửa/xóa câu hỏi, xem danh sách người dùng .

Yêu cầu phi chức năng (Non-functional Requirements):

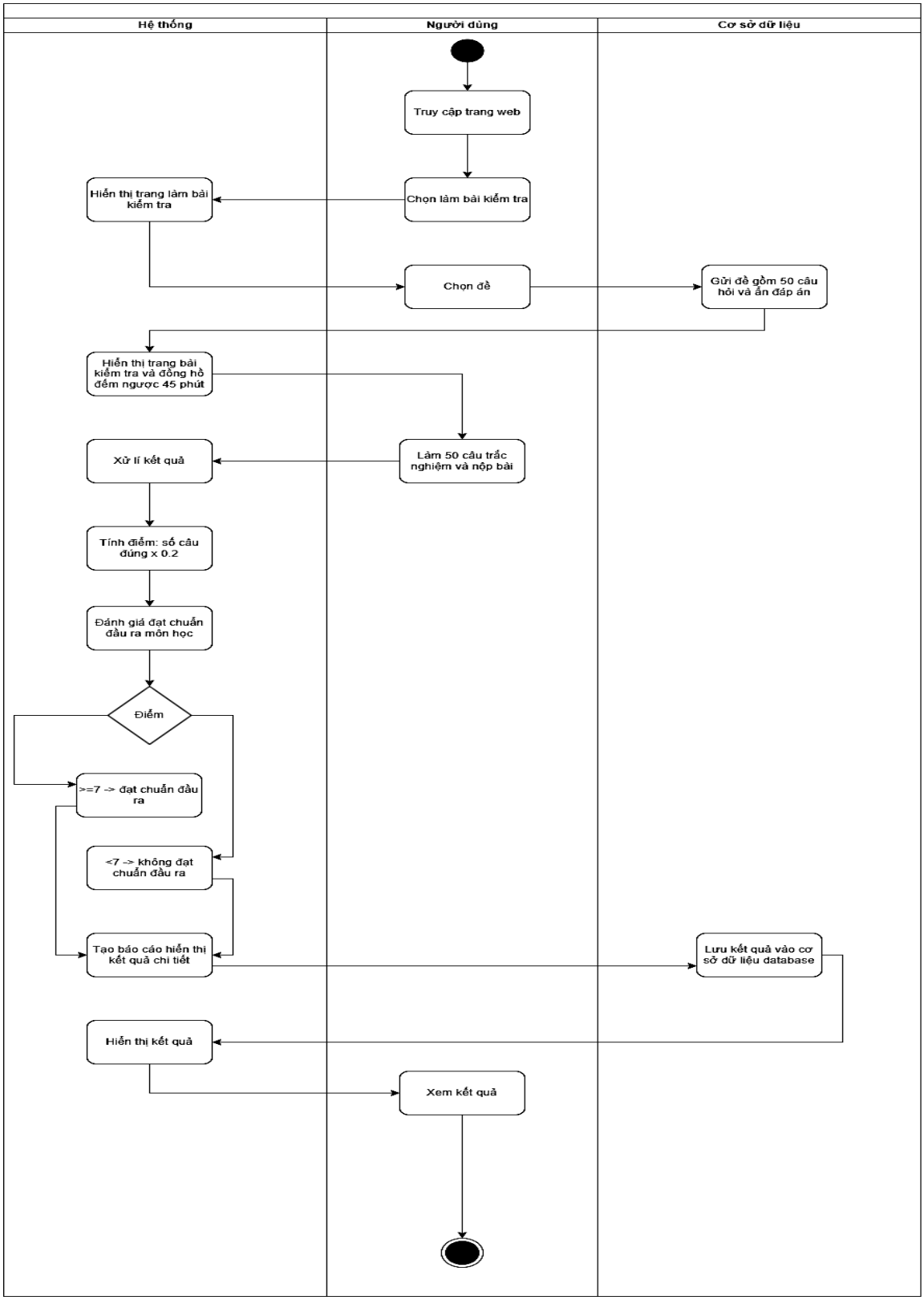
1. Giao diện người dùng: Responsive mobile/desktop bằng Flexbox/Grid, màu sắc dễ nhìn, font rõ ràng, tối ưu tải nhanh với hình nhỏ và lazy load.
2. Bảo mật (tối giản): Mật khẩu lưu đơn giản, session qua cookie/localStorage, quên mật khẩu qua Gmail, không captcha hay chống tấn công phức tạp vì hệ thống nội bộ.
3. Hiệu suất: Phản hồi dưới 3 giây, hỗ trợ 50 người dùng cùng lúc, tối ưu query với index database.
4. Khả năng mở rộng: Modular dễ thêm môn mới chỉ bằng dữ liệu + cấu hình, triển khai trên local/VPS nhỏ, hỗ trợ xóa tài khoản bảo vệ riêng tư.
5. Độ tin cậy & Sao lưu: Sao lưu hàng tuần, hoạt động ổn định nội bộ.
6. Tương thích & Triển khai: Chạy tốt trên Chrome/Firefox máy tính/điện thoại, internet 2Mbps, database 1GB cho quy mô nhỏ.
7. Kiểm thử: Test thủ công flow chính, lấy feedback từ học sinh thực tế về giao diện.

Bảo trì & Tài liệu: Code trên GitHub với README cài đặt, tài liệu hướng dẫn làm bài và đăng nhập đơn giản

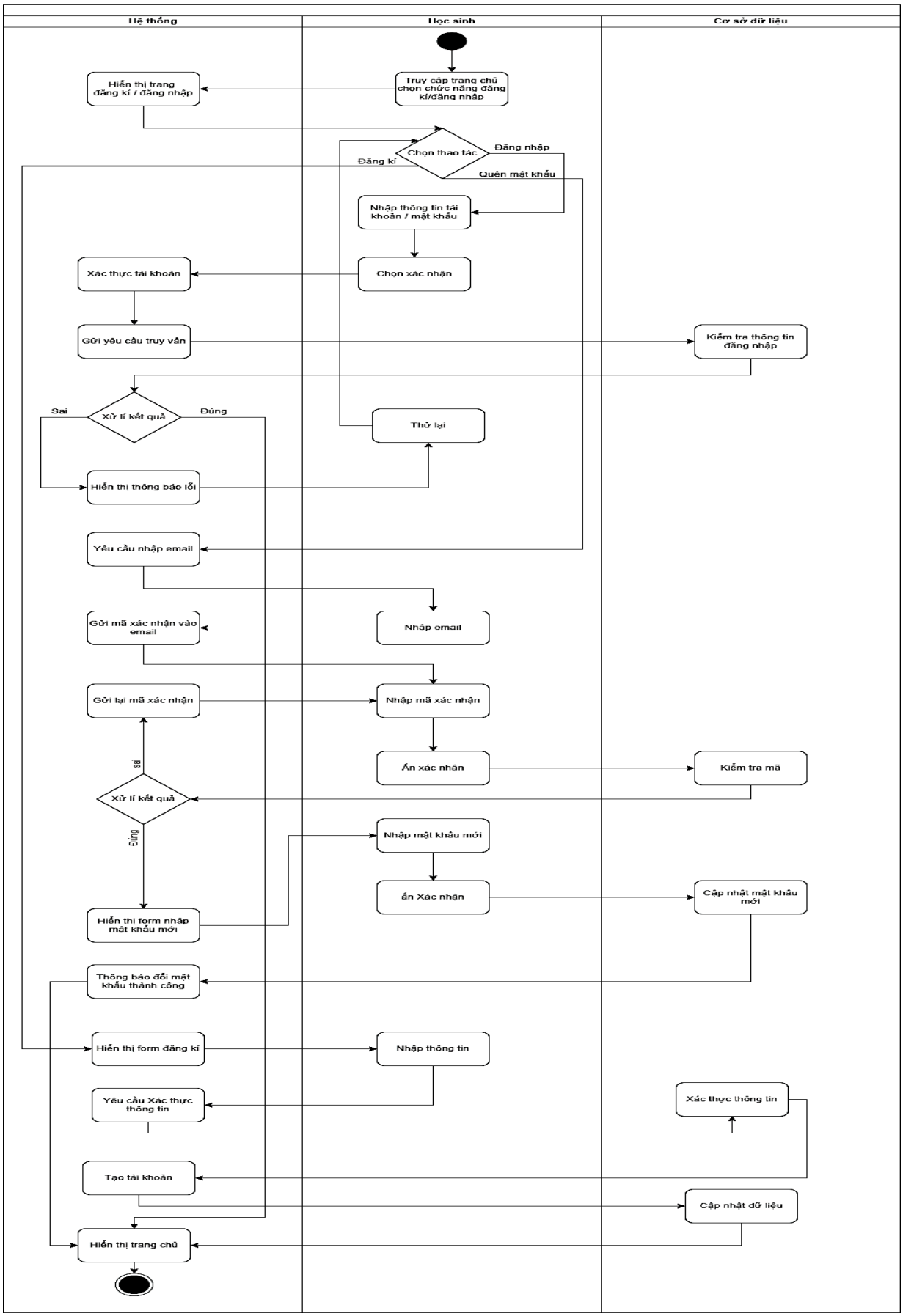
# Chương 2:Phân tích và thiết kế

## I. MÔ TẢ HỆ THỐNG

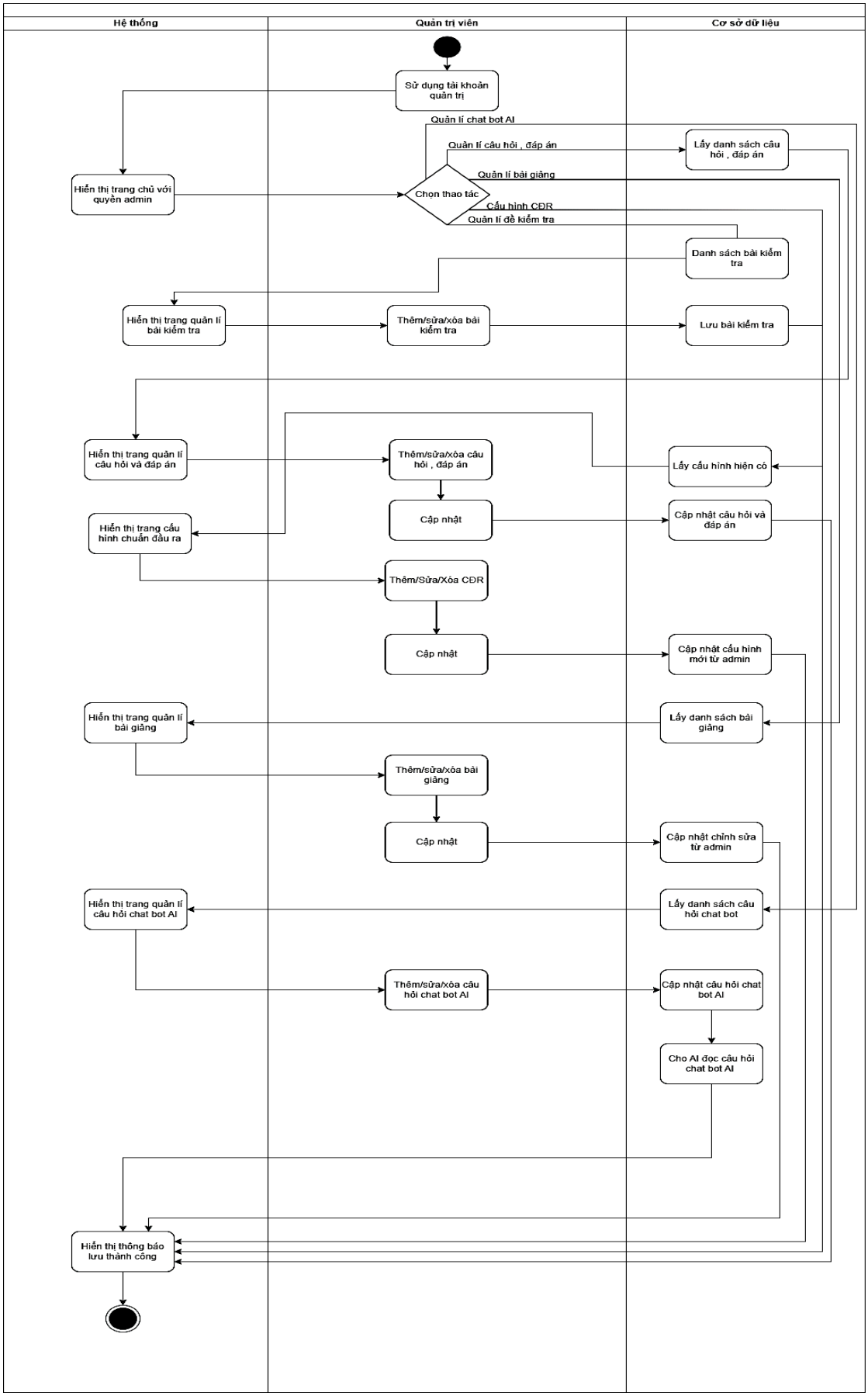
### 1.1. Biểu đồ hoạt động nghiệp vụ




Biểu đồ hoạt động nghiệp vụ “Làm bài kiểm tra”



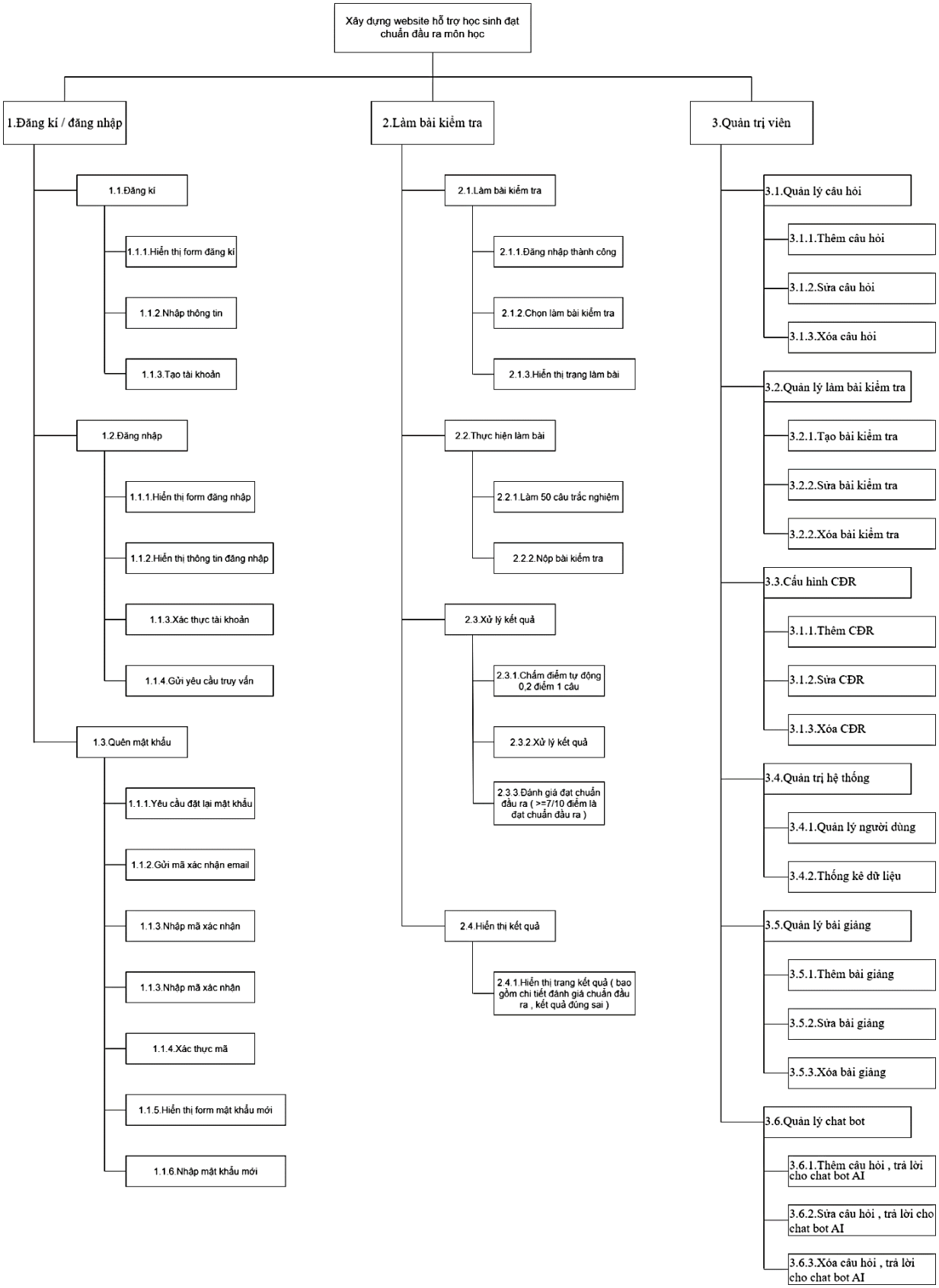

Biểu đồ hoạt động nghiệp vụ “Đăng kí , đăng nhập”




Biểu đồ hoạt động nghiệp vụ “Quản trị viên”

## 1.2. Biểu đồ phân rã chức năng nghiệp vụ

*a) Biểu đồ*




*b) Mô tả chi tiết chức năng lá*

## 1. ĐĂNG KÝ / ĐĂNG NHẬP

### 1.1 Đăng ký

#### 1.1.1 Hiển thị form đăng ký

+ Chức năng này hiển thị giao diện form đăng ký cho người dùng mới.
+ Dữ liệu đầu vào: Không có.
+ Các bước xử lý: Hệ thống tải và hiển thị form đăng ký với các trường: họ tên, email, tên tài khoản, mật khẩu, xác nhận mật khẩu.
+ Kết quả đầu ra: Hiển thị form đăng ký trên giao diện.
+ Hồ sơ dữ liệu: Không sử dụng.

#### 1.1.2 Nhập thông tin

+ Cho phép người dùng nhập thông tin cá nhân để tạo tài khoản.
+ Dữ liệu đầu vào:
    - Họ tên (bắt buộc)
    - Email (bắt buộc, định dạng email hợp lệ)
    - Tên tài khoản (bắt buộc, 4-20 ký tự, chỉ chữ cái, số, gạch dưới)
    - Mật khẩu (bắt buộc, tối thiểu 8 ký tự)
    - Xác nhận mật khẩu (bắt buộc, phải khớp với mật khẩu)
+ Các bước xử lý: Người dùng nhập thông tin vào các trường form.
+ Kết quả đầu ra: Dữ liệu được lưu tạm trong form.
+ Hồ sơ dữ liệu: Không sử dụng.

#### 1.1.3 Tạo tài khoản

+ Xử lý việc tạo tài khoản mới trong hệ thống.
+ Dữ liệu đầu vào: Thông tin từ form đăng ký (họ tên, email, tên tài khoản, mật khẩu).
+ Các bước xử lý:
    1. Kiểm tra định dạng tên tài khoản (4-20 ký tự, chỉ chữ, số, gạch dưới)
    2. Kiểm tra định dạng email hợp lệ
    3. So sánh mật khẩu và xác nhận mật khẩu
    4. Truy vấn cơ sở dữ liệu kiểm tra tên tài khoản đã tồn tại chưa
    5. Truy vấn cơ sở dữ liệu kiểm tra email đã tồn tại chưa
    6. Mã hóa mật khẩu bằng thuật toán băm một chiều
    7. Tạo bản ghi người dùng mới trong cơ sở dữ liệu
    8. Tạo mã xác thực 6 chữ số ngẫu nhiên (thời hạn 5 phút)
    9. Gửi email chứa mã xác thực

10. Người dùng xác thực email hoàn tất đăng ký

+ Kết quả đầu ra:
    - Thành công: Hiển thị thông báo "Đăng ký thành công", chuyển đến trang đăng nhập
    - Thất bại: Hiển thị thông báo lỗi cụ thể (email đã tồn tại, mật khẩu không khớp, định dạng không hợp lệ...)
+ Hồ sơ dữ liệu: Ghi vào users (người dùng).

## 1.2 Đăng nhập

### 1.2.1 Hiển thị form đăng nhập

+ Hiển thị giao diện form đăng nhập.
+ Dữ liệu đầu vào: Không có.
+ Các bước xử lý: Hệ thống tải và hiển thị form đăng nhập với các trường: tên tài khoản, mật khẩu.
+ Kết quả đầu ra: Hiển thị form đăng nhập trên giao diện.
+ Hồ sơ dữ liệu: Không sử dụng.

### 1.2.2 Hiển thị thông tin đăng nhập

+ Cho phép người dùng nhập thông tin để đăng nhập.
+ Dữ liệu đầu vào:
    - Tên tài khoản (bắt buộc)
    - Mật khẩu (bắt buộc)
+ Các bước xử lý: Người dùng nhập tên tài khoản và mật khẩu vào form.
+ Kết quả đầu ra: Dữ liệu được lưu tạm trong form.
+ Hồ sơ dữ liệu: Không sử dụng.

### 1.2.3 Xác thực tài khoản

+ Xác thực thông tin đăng nhập của người dùng.
+ Dữ liệu đầu vào: Tên tài khoản và mật khẩu.
+ Các bước xử lý:
    - Truy vấn cơ sở dữ liệu kiểm tra tên tài khoản có tồn tại không
    - Nếu tồn tại, lấy mật khẩu đã mã hóa và so sánh với mật khẩu người dùng nhập (sau khi mã hóa)
    - Nếu hợp lệ, tạo phiên làm việc mới (session) với mã xác thực và thời gian hết hạn
    - Lưu thông tin phiên làm việc vào cơ sở dữ liệu và bộ nhớ tạm
+ Kết quả đầu ra:

+ Thành công: Tạo phiên làm việc, chuyển đến trang chủ (học sinh → danh sách bài kiểm tra, quản trị viên → trang quản trị)
+ Thất bại: Hiển thị "Tên tài khoản hoặc mật khẩu không chính xác"
+ Hồ sơ dữ liệu: Đọc users, ghi sessions.

### 1.2.4 Gửi yêu cầu thay vì tự

+ Chuyển hướng người dùng sau khi đăng nhập thành công.
+ Dữ liệu đầu vào: Thông tin phiên làm việc.
+ Các bước xử lý: Hệ thống kiểm tra vai trò người dùng và chuyển hướng phù hợp.
+ Kết quả đầu ra:
    - Học sinh: Chuyển đến trang danh sách bài kiểm tra
    - Quản trị viên: Chuyển đến trang quản trị hệ thống
+ Hồ sơ dữ liệu: Đọc sessions.

## 1.3 Quên mật khẩu

### 1.3.1 Yêu cầu đặt lại mật khẩu

+ Cho phép người dùng yêu cầu khôi phục mật khẩu qua email.
+ Dữ liệu đầu vào: Địa chỉ email (bắt buộc).
+ Các bước xử lý:
    - Truy vấn cơ sở dữ liệu kiểm tra email có tồn tại không
    - Nếu tồn tại, tạo mã xác nhận ngẫu nhiên 6 chữ số
    - Lưu mã xác nhận vào cơ sở dữ liệu (thời hạn 15 phút)
    - Gửi email chứa mã xác nhận đến địa chỉ email người dùng
+ Kết quả đầu ra:
    1. Thành công: Hiển thị form nhập mã xác nhận, thông báo "Mã đã được gửi đến email"
    2. Thất bại: Hiển thị "Địa chỉ email không tồn tại trong hệ thống"
+ Hồ sơ dữ liệu: Đọc users, ghi password_resets.

### 1.3.2 Gửi mã xác nhận email

+ Gửi mã xác nhận đến email người dùng.
+ Dữ liệu đầu vào: Email người dùng, mã xác nhận.
+ Các bước xử lý: Hệ thống gửi email chứa mã xác nhận và hướng dẫn sử dụng.
+ Kết quả đầu ra: Email được gửi thành công.
+ Hồ sơ dữ liệu: Đọc password_resets.

### 1.3.3 Nhập mã xác nhận

+ Cho phép người dùng nhập mã xác nhận được qua email.
+ Dữ liệu đầu vào:
    - Mã xác nhận (6 chữ số)
    - Mật khẩu mới (tối thiểu 8 ký tự)
    - Xác nhận mật khẩu mới
+ Các bước xử lý: Người dùng nhập thông tin vào form.
+ Kết quả đầu ra: Dữ liệu được lưu tạm trong form.
+ Hồ sơ dữ liệu: Không sử dụng.

### 1.3.4 Xác thực mã

+ Xác thực mã và đặt lại mật khẩu mới.
+ Dữ liệu đầu vào: Mã xác nhận, mật khẩu mới, xác nhận mật khẩu mới.
+ Các bước xử lý:
    - Kiểm tra mã xác nhận khớp với mã trong cơ sở dữ liệu
    - Kiểm tra mã đã hết hạn chưa (so sánh thời gian)
    - Kiểm tra mật khẩu mới và xác nhận khớp nhau
    - Mã hóa mật khẩu mới
    - Cập nhật mật khẩu mới vào tài khoản
    - Xóa mã xác nhận đã sử dụng
+ Kết quả đầu ra:
    1. Thành công: Hiển thị "Đặt lại mật khẩu thành công", chuyển đến trang đăng nhập
    2. Thất bại: Hiển thị lỗi (mã không đúng, mã hết hạn, mật khẩu không khớp)
+ Hồ sơ dữ liệu: Đọc và xóa password_resets, ghi users.

### 1.3.5 Hiển thị form mật khẩu mới

+ Hiển thị form nhập mật khẩu mới.
+ Dữ liệu đầu vào: Không có.
+ Các bước xử lý: Hệ thống hiển thị form với các trường: mã xác nhận, mật khẩu mới, xác nhận mật khẩu.
+ Kết quả đầu ra: Hiển thị form đặt lại mật khẩu.
+ Hồ sơ dữ liệu: Không sử dụng.

### 1.3.6 Nhập mật khẩu mới

+ Cho phép người dùng nhập mật khẩu mới.

+ Dữ liệu đầu vào: Mật khẩu mới và xác nhận mật khẩu mới.
+ Các bước xử lý: Người dùng nhập thông tin vào form.
+ Kết quả đầu ra: Dữ liệu được lưu tạm.
+ Hồ sơ dữ liệu: Không sử dụng.

## 2. LÀM BÀI KIỂM TRA

### 2.1 Làm bài kiểm tra

#### 2.1.1 Đăng nhập thành công

+ Xác nhận người dùng đã đăng nhập thành công.
+ Dữ liệu đầu vào: Thông tin phiên làm việc.
+ Các bước xử lý: Kiểm tra phiên làm việc hợp lệ.
+ Kết quả đầu ra: Cho phép truy cập các chức năng của hệ thống.
+ Hồ sơ dữ liệu: Đọc sessions.

#### 2.1.2 Chọn làm bài kiểm tra

+ Hiển thị danh sách bài kiểm tra cho học sinh chọn.
+ Dữ liệu đầu vào: Không có.
+ Các bước xử lý:
    1. Truy vấn cơ sở dữ liệu lấy danh sách bài kiểm tra
    2. Lọc chỉ giữ bài có trạng thái "đang hoạt động"
    3. Hiển thị thông tin: tên bài, môn học, số câu hỏi, thời gian làm bài
    4. Cung cấp nút chọn bài kiểm tra
+ Kết quả đầu ra: Danh sách bài kiểm tra dạng lưới/danh sách. Khi chọn, chuyển đến trang làm bài.
+ Hồ sơ dữ liệu: Đọc tests.

#### 2.1.3 Hiển thị trang làm bài

+ Hiển thị giao diện làm bài kiểm tra với câu hỏi và đếm ngược.
+ Dữ liệu đầu vào: Mã bài kiểm tra.
+ Các bước xử lý:
    - Lấy cấu hình bài kiểm tra
    - Chọn ngẫu nhiên 50 câu hỏi theo độ khó
    - Xáo trộn thứ tự câu hỏi và đáp án
    - Bắt đầu đồng hồ đếm ngược 45 phút
    - Lưu thời điểm bắt đầu và danh sách câu hỏi vào phiên làm bài
    - Hiển thị từng câu hỏi với 4 đáp án A, B, C, D

- Cho phép học sinh chọn đáp án
- Khi hết thời gian, tự động nộp bài

+ Kết quả đầu ra: Giao diện làm bài với đồng hồ đếm ngược, hiển thị tổng số câu và số câu đã làm.
+ Hồ sơ dữ liệu: Đọc tests, questions, answers; ghi exam_sessions.

## 2.2 Thực hiện làm bài

### 2.2.1 Làm 50 câu trắc nghiệm

+ Cho phép học sinh trả lời 50 câu hỏi trắc nghiệm.
+ Dữ liệu đầu vào: Danh sách câu hỏi trong bài kiểm tra từ database.
+ Các bước xử lý: Học sinh chọn đáp án cho từng câu hỏi.
+ Kết quả đầu ra: Danh sách câu trả lời của học sinh.
+ Hồ sơ dữ liệu: Đọc questions, answers.

### 2.2.2 Nộp bài kiểm tra

+ Mục đích Xử lý việc nộp bài và chấm điểm.
+ Dữ liệu đầu vào:
  - Mã phiên làm bài (session_id)
  - Danh sách câu trả lời (mã câu hỏi + mã đáp án)
+ Các bước xử lý:
  1. Lưu toàn bộ câu trả lời vào cơ sở dữ liệu (kèm thời gian nộp)
  2. Lấy danh sách đáp án đúng từ bảng `answers`
  3. So sánh từng câu trả lời với đáp án đúng
  4. Đếm số câu đúng, sai, chưa làm
  5. Tính điểm theo công thức: `score = (số câu đúng / tổng số câu) × 10`
     - Ví dụ đề 50 câu: mỗi câu = 0.2 điểm
     - Ví dụ đề 10 câu: mỗi câu = 1.0 điểm
  6. Lưu kết quả (điểm, số câu đúng/sai, thời gian hoàn thành, trạng thái đạt CĐR)
  7. Kết quả đầu ra: Tự động chuyển đến trang xem kết quả.
  8. Hồ sơ dữ liệu: Ghi `test_sessions`, `test_results`; đọc `answers`.

## 2.3 Xử lý kết quả

### 2.3.1 Chấm điểm tự động

+ Mục đích: Tính điểm tự động cho bài làm.

+ Dữ liệu đầu vào: Danh sách câu trả lời của học sinh.

Các bước xử lý:

+ Tính điểm = (Số câu đúng / Tổng số câu) × 10
+ Làm tròn 1 chữ số thập phân

Kết quả đầu ra: Tổng điểm (0-10).

Hồ sơ dữ liệu: Đọc `answers`.

### 2.3.2 Xử lý kết quả

+ Mục đích: Xử lý và lưu kết quả chi tiết.
+ Dữ liệu đầu vào: Điểm số, số câu đúng/sai, trạng thái đạt CĐR.
+ Các bước xử lý: Tạo bản ghi kết quả trong cơ sở dữ liệu.
+ Kết quả đầu ra: Kết quả được lưu.
+ Hồ sơ dữ liệu: Ghi `test_results`.

### 2.3.3 Đánh giá đạt chuẩn đầu ra (>=7.0 điểm là đạt chuẩn đầu ra)

+ Mục đích: Đánh giá học sinh có đạt chuẩn đầu ra môn học hay không.
+ Dữ liệu đầu vào: Tổng điểm của học sinh.
+ Các bước xử lý:
    1. So sánh điểm với ngưỡng chuẩn đầu ra (7.0 điểm)
    2. Nếu điểm >= 7.0: Đạt chuẩn đầu ra
    3. Nếu điểm < 7.0: Chưa đạt chuẩn đầu ra
+ Kết quả đầu ra: Trạng thái đạt/chưa đạt chuẩn đầu ra.
+ Hồ sơ dữ liệu: Đọc `test_results`.

## 2.4 Hiển thị kết quả

### 2.4.1 Hiển thị trang kết quả

+ Mục đích: Hiển thị kết quả bài kiểm tra và đánh giá chuẩn đầu ra.
+ Dữ liệu đầu vào: Mã kết quả kiểm tra (session_id).
+ Các bước xử lý:
    1. Lấy thông tin kết quả từ cơ sở dữ liệu
    2. Tính số câu đúng, sai, chưa làm
    3. Tính phần trăm hoàn thành
    4. Tính thời gian làm bài thực tế
    5. Tính số câu cần để đạt CĐR: `Math.ceil(tổng số câu × 0.7)`
    6. Hiển thị trạng thái đạt/chưa đạt chuẩn đầu ra

7. Gợi ý bài học cần ôn tập (dựa vào điểm số và liên kết từ `lesson_tests`)

+ Kết quả đầu ra: Trang kết quả hiển thị:
    - Tổng điểm (chữ to, có màu sắc: xanh nếu >= 7.0, đỏ nếu < 7.0)
    - Số câu đúng/sai/chưa làm
    - Thời gian hoàn thành
    - Phần trăm hoàn thành
    - Trạng thái chuẩn đầu ra:
        - Nếu >= 7.0: " Chúc mừng! Bạn đã ĐẠT Chuẩn Đầu Ra (CĐR)"
        - "Bạn cần ≥ {số câu cần}/{tổng câu} câu đúng (≥ 7.0 điểm) để đạt CĐR. Bạn đã vượt qua tiêu chuẩn này!"
        - Nếu < 7.0: " Bạn chưa đạt Chuẩn Đầu Ra (CĐR)"
        - "Bạn cần ≥ {số câu cần}/{tổng câu} câu đúng (≥ 7.0 điểm) để đạt CĐR. Hãy cố gắng thêm!"
    - Gợi ý bài học: Danh sách bài học cần ôn tập với link trực tiếp
    - Liên kết xem chi tiết đáp án
+ Hồ sơ dữ liệu: Đọc `test_results`, `test_sessions`, `lesson_tests`, `lessons`.

Lưu ý:

+ Số câu cần đạt CĐR = 70% tổng số câu (làm tròn lên)
+ Ví dụ: Đề 50 câu cần >= 35 câu, đề 10 câu cần >= 7 câu
+ Công thức điểm linh hoạt với mọi số câu hỏi

## 3. QUẢN TRỊ VIÊN

### 3.1 Quản lý câu hỏi

#### 3.1.1 Thêm câu hỏi

+ Cho phép quản trị viên thêm câu hỏi mới.
+ Dữ liệu đầu vào: Nội dung câu hỏi, môn học, độ khó, 4 đáp án A/B/C/D, đáp án đúng.
+ Các bước xử lý:
    1. Kiểm tra dữ liệu hợp lệ
    2. Tạo bản ghi câu hỏi mới
    3. Tạo 4 bản ghi đáp án
    4. Đánh dấu đáp án đúng
    5. Ghi nhận thời gian tạo và người tạo

+ Kết quả đầu ra: Thông báo thêm thành công, làm mới danh sách câu hỏi.
+ Hồ sơ dữ liệu: Ghi questions, answers.

### 3.1.2 Sửa câu hỏi

+ Cho phép quản trị viên chỉnh sửa câu hỏi.
+ Dữ liệu đầu vào: Mã câu hỏi, thông tin mới (nội dung, môn học, độ khó, đáp án).
+ Các bước xử lý:
    - Lấy thông tin câu hỏi hiện tại
    - Cập nhật thông tin câu hỏi
    - Cập nhật 4 đáp án
    - Cập nhật đánh dấu đáp án đúng
    - Cập nhật thời gian sửa đổi
+ Kết quả đầu ra: Thông báo cập nhật thành công.
+ Hồ sơ dữ liệu: Đọc và ghi questions, answers.

### 3.1.3 Xóa câu hỏi

+ Cho phép quản trị viên xóa câu hỏi.
+ Dữ liệu đầu vào: Mã câu hỏi.
+ Các bước xử lý:
    1. Kiểm tra câu hỏi có đang được sử dụng trong bài kiểm tra không
    2. Nếu đang sử dụng: Từ chối xóa
    3. Nếu không sử dụng: Xóa 4 đáp án trước, sau đó xóa câu hỏi
    4. Ghi nhận hành động vào nhật ký
+ Kết quả đầu ra:
    1. Thành công: Thông báo xóa thành công
    2. Thất bại: Thông báo câu hỏi đang được sử dụng
+ Hồ sơ dữ liệu: Đọc và xóa questions, answers, đọc test_questions.

## 3.2 Quản lý làm bài kiểm tra

### 3.2.1 Tạo bài kiểm tra

+ Cho phép quản trị viên tạo bài kiểm tra mới.
+ Dữ liệu đầu vào: Tên bài kiểm tra, môn học, thời gian (45 phút), số câu hỏi (50 câu), cấu hình độ khó, trạng thái.
+ Các bước xử lý:
    - Kiểm tra dữ liệu hợp lệ
    - Tạo bản ghi bài kiểm tra mới

- Lưu cấu hình độ khó
- Ghi nhận người tạo và thời gian tạo

+ Kết quả đầu ra: Thông báo tạo thành công, chuyển đến danh sách bài kiểm tra.
+ Hồ sơ dữ liệu: Ghi tests.

### 3.2.2 Sửa bài kiểm tra

+ Cho phép quản trị viên chỉnh sửa bài kiểm tra.
+ Dữ liệu đầu vào: Mã bài kiểm tra, thông tin mới (tên, môn học, thời gian, số câu, độ khó, trạng thái).
+ Các bước xử lý:
    1. Lấy thông tin bài kiểm tra hiện tại
    2. Cập nhật thông tin mới
    3. Cập nhật cấu hình độ khó
    4. Lưu thay đổi và cập nhật thời gian sửa đổi
+ Kết quả đầu ra: Thông báo cập nhật thành công.
+ Hồ sơ dữ liệu: Đọc và ghi tests.

### 3.2.3 Xóa bài kiểm tra

+ Cho phép quản trị viên xóa bài kiểm tra.
+ Dữ liệu đầu vào: Mã bài kiểm tra.
+ Các bước xử lý:
    1. Kiểm tra có kết quả nào của bài kiểm tra không
    2. Nếu có kết quả: Từ chối xóa, đề xuất chuyển trạng thái thành "không hoạt động"
    3. Nếu không có kết quả: Xóa bài kiểm tra
    4. Ghi nhận vào nhật ký
+ Kết quả đầu ra:
    1. Thành công: Thông báo xóa thành công
    2. Thất bại: Thông báo đã có kết quả, không thể xóa
+ Hồ sơ dữ liệu: Đọc và xóa tests, đọc test_results.

## 3.3 Cấu hình CĐR

### 3.3.1 Thêm CĐR

+ Cho phép quản trị viên thiết lập ngưỡng chuẩn đầu ra cho môn học.
+ Dữ liệu đầu vào: Môn học, ngưỡng điểm CĐR (0-10).
+ Các bước xử lý:

1. Kiểm tra môn học tồn tại
2. Kiểm tra ngưỡng điểm hợp lệ (0-10, thường 5-8)
3. Cập nhật ngưỡng CĐR vào bản ghi môn học
4. Ghi nhận thời gian cập nhật

+ Kết quả đầu ra: Thông báo đã cập nhật ngưỡng CĐR.
+ Hồ sơ dữ liệu: Đọc và ghi subjects.

### 3.3.2 Sửa CĐR

+ Cho phép quản trị viên chỉnh sửa ngưỡng CĐR.
+ Dữ liệu đầu vào: Môn học, ngưỡng CĐR mới.
+ Các bước xử lý:
    1. Lấy ngưỡng CĐR hiện tại
    2. Hiển thị giá trị hiện tại
    3. Kiểm tra ngưỡng mới hợp lệ
    4. Cập nhật ngưỡng mới
    5. Cập nhật thời gian sửa đổi
+ Kết quả đầu ra: Thông báo cập nhật thành công (từ giá trị cũ → mới).
+ Hồ sơ dữ liệu: Đọc và ghi subjects.

### 3.3.3 Xóa CĐR

+ Cho phép quản trị viên đặt lại ngưỡng CĐR về mặc định (7.0).
+ Dữ liệu đầu vào: Môn học.
+ Các bước xử lý:
    - Lấy ngưỡng CĐR hiện tại
    - Hiển thị hộp thoại xác nhận đặt lại về 7.0
    - Nếu xác nhận: Cập nhật ngưỡng = 7.0
    - Ghi nhận vào nhật ký
+ Kết quả đầu ra:
    - Xác nhận: Thông báo đã đặt lại về mặc định
    - Hủy: Không thay đổi
+ Hồ sơ dữ liệu: Đọc và ghi subjects.

## 3.4 Quản trị hệ thống

### 3.4.1 Quản lý người dùng

+ Cho phép quản trị viên xem và quản lý tài khoản.
+ Dữ liệu đầu vào: Bộ lọc (vai trò, trạng thái), từ khóa tìm kiếm.
+ Các bước xử lý:

- Lấy danh sách tài khoản
- Áp dụng bộ lọc
- Tìm kiếm theo email/họ tên
- Sắp xếp theo thời gian tạo

+ Hiển thị thông tin: mã, họ tên, email, vai trò, trạng thái, ngày tạo
+ Kết quả đầu ra: Bảng danh sách tài khoản với nút sửa/khóa/xóa.
+ Hồ sơ dữ liệu: Đọc users.

### 3.4.2 Thống kê dữ liệu

+ Hiển thị thống kê tổng quan hệ thống.
+ Dữ liệu đầu vào: Không có.
+ Các bước xử lý:
  - Đếm tổng số người dùng
  - Đếm tổng số bài kiểm tra
  - Đếm tổng số câu hỏi
  - Đếm tổng số lượt làm bài
+ Kết quả đầu ra: Hiển thị thống kê dạng số và biểu đồ.
+ Hồ sơ dữ liệu: Đọc users, tests, questions, test_results.

## 3.5 Quản lý bài giảng

### 3.5.1 Thêm bài giảng

+ Cho phép quản trị viên thêm bài giảng mới.
+ Dữ liệu đầu vào: Tên bài giảng, môn học, nội dung, tài liệu đính kèm.
+ Các bước xử lý:
  - Kiểm tra dữ liệu hợp lệ
  - Tạo bản ghi bài giảng mới
  - Lưu tài liệu đính kèm (nếu có)
  - Ghi nhận người tạo và thời gian
+ Kết quả đầu ra: Thông báo thêm thành công.
+ Hồ sơ dữ liệu: Ghi lessons.

### 3.5.2 Sửa bài giảng

+ Cho phép quản trị viên chỉnh sửa bài giảng.
+ Dữ liệu đầu vào: Mã bài giảng, thông tin mới.
+ Các bước xử lý:
  - Lấy thông tin bài giảng hiện tại
  - Cập nhật thông tin mới

- Cập nhật tài liệu (nếu có)
- Cập nhật thời gian sửa đổi

+ Kết quả đầu ra: Thông báo cập nhật thành công.
+ Hồ sơ dữ liệu: Đọc và ghi lessons.

### 3.5.3 Xóa bài giảng

+ Cho phép quản trị viên xóa bài giảng.
+ Dữ liệu đầu vào: Mã bài giảng.
+ Các bước xử lý:
    1. Xác nhận xóa
    2. Xóa tài liệu đính kèm (nếu có)
    3. Xóa bản ghi bài giảng
    4. Ghi nhận vào nhật ký
+ Kết quả đầu ra: Thông báo xóa thành công.
+ Hồ sơ dữ liệu: Đọc và xóa lessons.

## 3.6 Quản lý chat bot

### 3.6.1 Thêm câu hỏi, trả lời cho chat bot AI

+ Cho phép quản trị viên thêm câu hỏi/trả lời mẫu cho chatbot AI.
+ Dữ liệu đầu vào: Câu hỏi mẫu, câu trả lời mẫu, danh mục.
+ Các bước xử lý:
    1. Kiểm tra dữ liệu hợp lệ
    2. Tạo bản ghi câu hỏi/trả lời chatbot
    3. Ghi nhận người tạo và thời gian
+ Kết quả đầu ra: Thông báo thêm thành công.
+ Hồ sơ dữ liệu: Ghi chatbot_qa.

### 3.6.2 Sửa câu hỏi, trả lời cho chat bot AI

+ Cho phép quản trị viên chỉnh sửa câu hỏi/trả lời chatbot.
+ Dữ liệu đầu vào: Mã câu hỏi, thông tin mới.
+ Các bước xử lý:
    1. Lấy thông tin hiện tại
    2. Cập nhật thông tin mới
    3. Cập nhật thời gian sửa đổi
+ Kết quả đầu ra: Thông báo cập nhật thành công.
+ Hồ sơ dữ liệu: Đọc và ghi chatbot_qa.

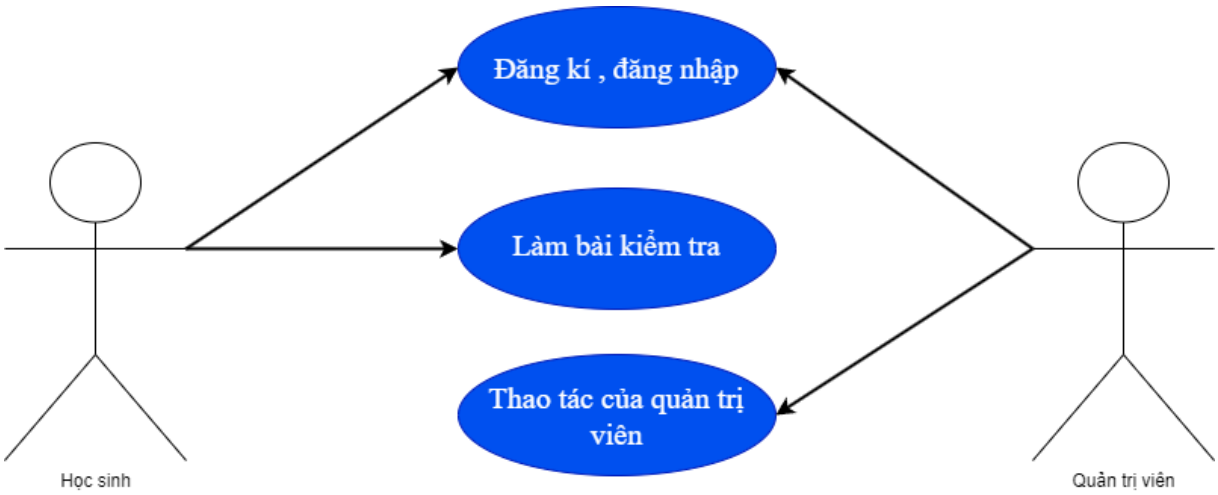
### 3.6.3 Xóa câu hỏi, trả lời cho chat bot AI

+ Cho phép quản trị viên xóa câu hỏi/trả lời chatbot.
+ Dữ liệu đầu vào: Mã câu hỏi.
+ Các bước xử lý:
    1. Xác nhận xóa
    2. Xóa bản ghi
    3. Ghi nhận vào nhật ký
+ Kết quả đầu ra: Thông báo xóa thành công.
+ Hồ sơ dữ liệu: Đọc và xóa chatbot_qa.

## II. PHÂN TÍCH HỆ THỐNG

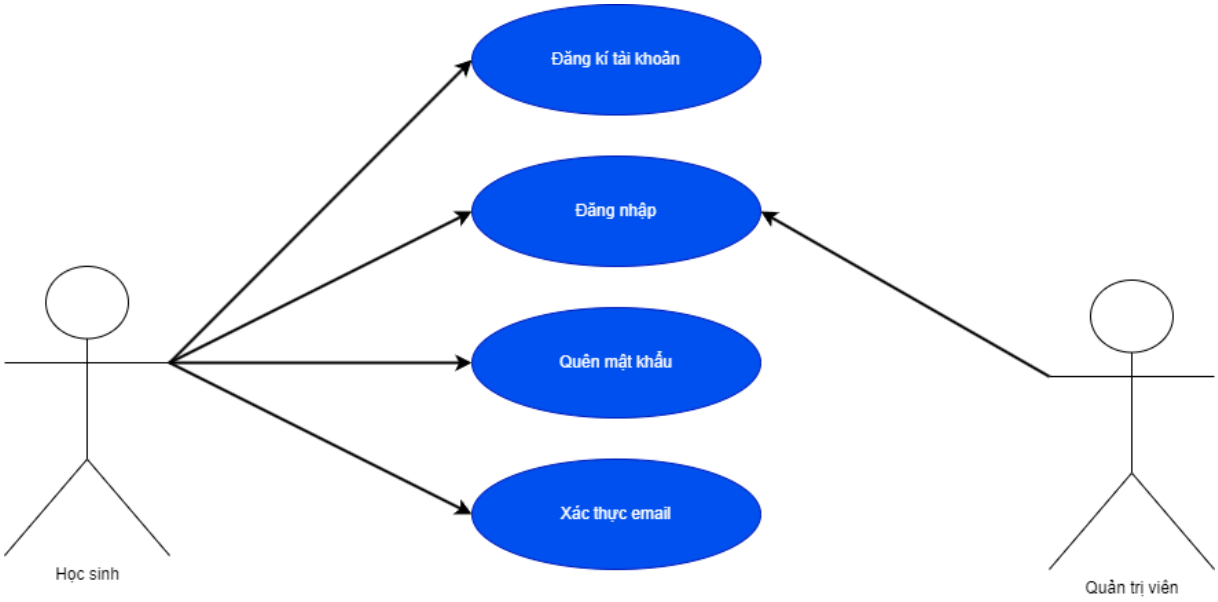
### 2.1. Biểu đồ Use Case tổng quát


### 2.2. Biểu đồ Use Case chi tiết

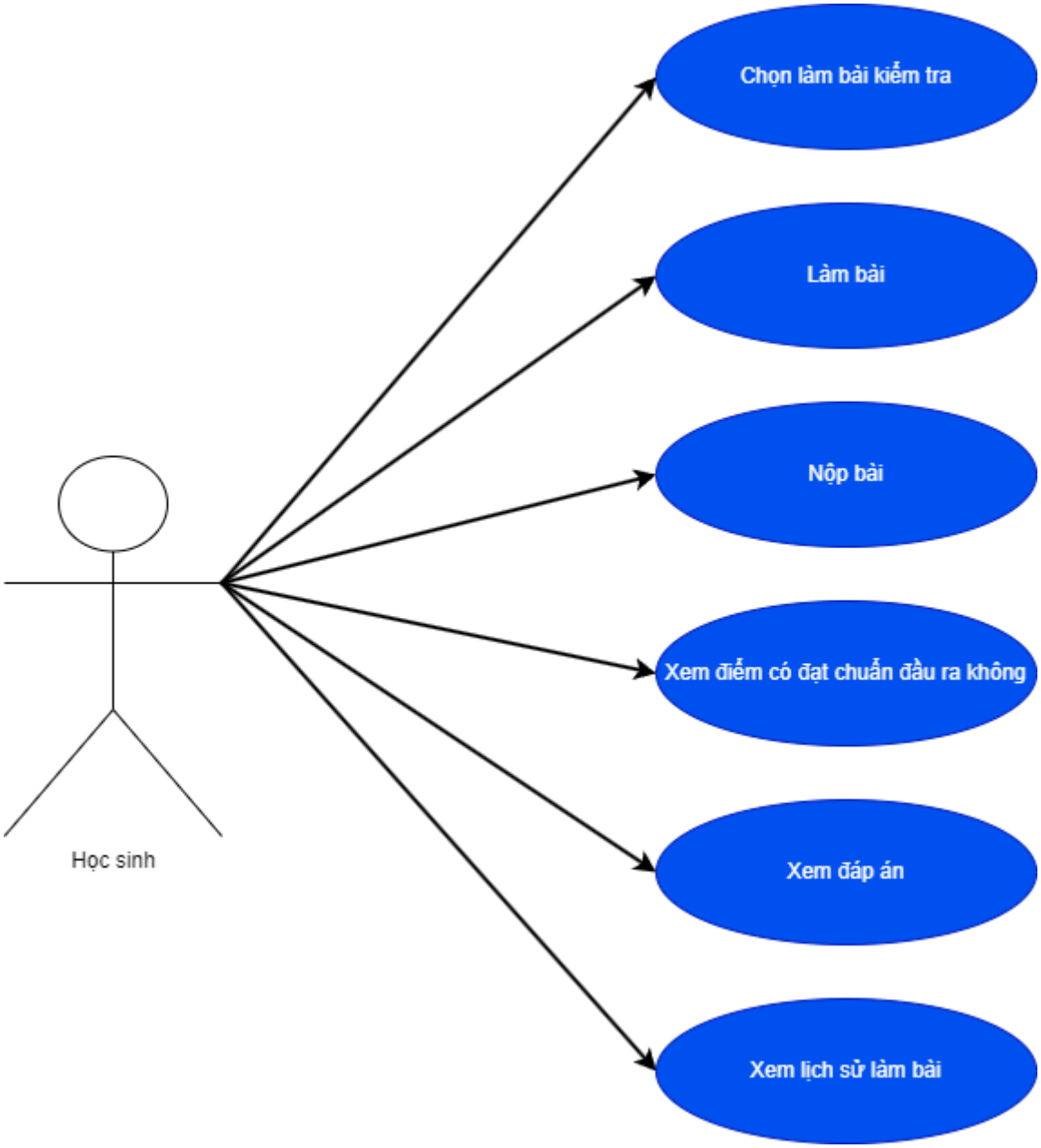


## 2.3. Đặc tả các Use Case

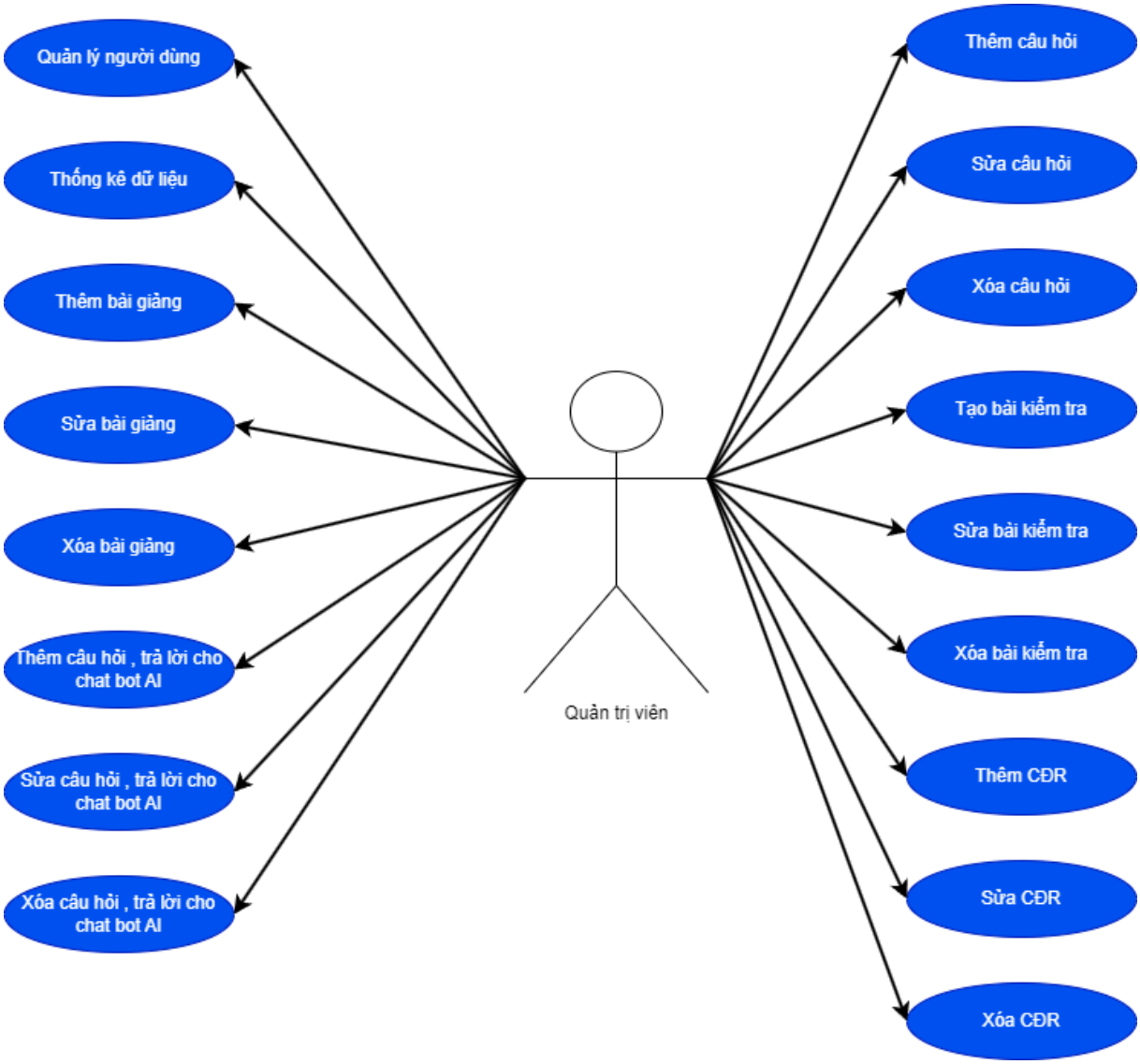
### *2.3.1.Ca sử dụng "Đăng kí , đăng nhập"*



### *2.3.2.Ca sử dụng "Làm bài kiểm tra"*



### *2.3.3.Ca sử dụng "Quản trị viên"*




## Đặc tả:

1. USE CASE ĐĂNG KÝ / ĐĂNG NHẬP

a) Use Case "Đăng ký tài khoản"

Tên Use Case: Đăng ký tài khoản

Tác nhân: Học sinh

Mô tả hoạt động: Cho phép người dùng mới tạo tài khoản trong hệ thống bằng cách cung cấp thông tin cá nhân

Điều kiện trước:

- Người dùng chưa có tài khoản trong hệ thống
- Người dùng đang truy cập trang đăng nhập/đăng ký

Luồng sự kiện chính:

- Hệ thống hiển thị form đăng ký với các trường:
  + Email (bắt buộc)
  + Mật khẩu (bắt buộc, tối thiểu 6 ký tự)
  + Họ và tên (bắt buộc, tối thiểu 2 ký tự)
  + Số điện thoại (tùy chọn, 10-11 chữ số)
- Người dùng nhập đầy đủ thông tin
- Hệ thống validate dữ liệu đầu vào:
  + Email đúng định dạng
  + Mật khẩu đủ mạnh (≥ 6 ký tự)
  + Họ tên hợp lệ (≥ 2 ký tự)
  + Số điện thoại hợp lệ (nếu có)
- Hệ thống kiểm tra email chưa tồn tại trong database
- Hệ thống mã hóa mật khẩu bằng bcrypt (10 rounds)
- Hệ thống tạo tài khoản mới với:
  + Role mặc định: "student"
  + Status: "active"
- Hệ thống tạo JWT token với thời hạn 45 phút
- Hệ thống trả về token và thông tin user
- Frontend lưu token vào localStorage
- Người dùng được chuyển hướng về trang chủ

Điều kiện sau:

- Tài khoản được tạo trong database
- JWT token được tạo và lưu trữ
- Người dụng đăng nhập tự động
- Người dùng có thể sử dụng đầy đủ chức năng hệ thống

Các luồng sự kiện thay thế:

Luồng 3a: Email không đúng định dạng

- Hiển thị lỗi: "Email không đúng định dạng"
- Quay lại bước 2

Luồng 3b: Mật khẩu quá ngắn

- Hiển thị lỗi: "Mật khẩu phải có ít nhất 6 ký tự"

- Quay lại bước 2

Luồng 3c: Họ tên quá ngắn

- Hiển thị lỗi: "Họ tên phải có ít nhất 2 ký tự"

- Quay lại bước 2

Luồng 3d: Số điện thoại không hợp lệ

- Hiển thị lỗi: "Số điện thoại không hợp lệ (10-11 chữ số)"

- Quay lại bước 2

Luồng 4a: Email đã tồn tại

- Hiển thị lỗi: "Email đã được đăng ký. Vui lòng sử dụng email khác"

- Quay lại bước 2

Luồng 5-7a: Lỗi hệ thống

- Hiển thị lỗi: "Lỗi hệ thống khi đăng ký tài khoản. Vui lòng thử lại sau"

- Kết thúc use case

Yêu cầu đặc biệt:

- Bảo mật: Mật khẩu được hash với bcrypt

- Performance: Response time < 2 giây

- UI/UX: Form rõ ràng, validation real-time

b) Use Case "Đăng nhập"

Tên Use Case: Đăng nhập

Tác nhân: Học sinh, Quản trị viên

Mô tả hoạt động: Cho phép người dùng đã có tài khoản truy cập vào hệ thống bằng email và mật khẩu

Điều kiện trước:

- Người dùng đã có tài khoản trong hệ thống

- Người dùng đang truy cập trang đăng nhập

Luồng sự kiện chính:

- Hệ thống hiển thị form đăng nhập với:

  + Email (bắt buộc)

  + Mật khẩu (bắt buộc)

- Người dùng nhập email và mật khẩu
- Hệ thống validate dữ liệu đầu vào
- Hệ thống kiểm tra email tồn tại trong database
- Hệ thống kiểm tra status tài khoản = "active"
- Hệ thống so sánh mật khẩu với hash trong database (bcrypt.compare)
- Hệ thống tạo JWT token với thời hạn 45 phút
- Hệ thống trả về token và thông tin user (không bao gồm password)
- Frontend lưu token vào localStorage
- Người dùng được chuyển hướng theo role:
  + Student/Teacher → Trang chủ
  + Admin → Admin Dashboard

Điều kiện sau:
- JWT token được tạo và lưu trữ
- Session được thiết lập (45 phút)
- Người dùng truy cập được các chức năng theo quyền
- Middleware authenticate sẽ verify token cho các request sau

Các luồng sự kiện thay thế:

Luồng 3a: Thiếu thông tin
- Hiển thị lỗi: "Vui lòng nhập email và mật khẩu"
- Quay lại bước 2

Luồng 4a: Email không tồn tại
- Hiển thị lỗi: "Email hoặc mật khẩu không đúng"
- Quay lại bước 2

Luồng 5a: Tài khoản bị khóa (status = "inactive")
- Hiển thị lỗi: "Tài khoản đã bị vô hiệu hóa. Vui lòng liên hệ quản trị viên"
- Kết thúc use case

Luồng 6a: Mật khẩu không đúng
- Hiển thị lỗi: "Email hoặc mật khẩu không đúng"
- Quay lại bước 2

Luồng 7a: Lỗi tạo token

- Hiển thị lỗi: "Lỗi khi tạo phiên đăng nhập. Vui lòng thử lại"

- Kết thúc use case

Yêu cầu đặc biệt:

- Bảo mật: Không tiết lộ thông tin email có tồn tại hay không

- Session: Token hết hạn sau 45 phút

- Auto-logout: Frontend tự động logout khi token hết hạn

c) Use Case "Quên mật khẩu"

Tên Use Case: Quên mật khẩu

Tác nhân: Học sinh

Mô tả hoạt động: Cho phép người dùng khôi phục mật khẩu bằng mã OTP gửi qua email (3 bước)

Điều kiện trước:

- Người dùng có tài khoản trong hệ thống

- Người dùng không nhớ mật khẩu

- Có kết nối email service

Luồng sự kiện chính:

BƯỚC 1: Nhập Email

- Hệ thống hiển thị form nhập email

- Người dùng nhập email

- Hệ thống validate định dạng email

- Hệ thống kiểm tra email tồn tại trong database

- Hệ thống tạo mã OTP 6 chữ số ngẫu nhiên

- Hệ thống lưu OTP vào bộ nhớ tạm với:

  + Thời hạn: 10 phút

  + Trạng thái: chưa xác thực

- Hệ thống gửi OTP qua email

- Hệ thống chuyển sang BƯỚC 2

BƯỚC 2: Xác thực OTP

- Hệ thống hiển thị form nhập OTP (6 chữ số)

- Người dùng nhập mã OTP nhận được

- Hệ thống validate OTP:
  + Kiểm tra tồn tại
  + Kiểm tra thời hạn (< 10 phút)
  + So sánh với OTP đã lưu
- Hệ thống đánh dấu OTP đã xác thực
- Hệ thống chuyển sang BƯỚC 3

BƯỚC 3: Đặt lại mật khẩu

- Hệ thống hiển thị form nhập mật khẩu mới với:
  + Mật khẩu mới (≥ 6 ký tự)
  + Xác nhận mật khẩu
- Người dùng nhập mật khẩu mới và xác nhận
- Hệ thống validate:
  + Độ dài ≥ 6 ký tự
  + Mật khẩu khớp với xác nhận
  + OTP đã được xác thực
  + OTP chưa hết hạn
- Hệ thống hash mật khẩu mới (bcrypt, 10 rounds)
- Hệ thống cập nhật mật khẩu trong database
- Hệ thống xóa OTP khỏi bộ nhớ tạm
- Hiển thị thông báo thành công
- Chuyển hướng về trang đăng nhập sau 2 giây

Điều kiện sau:

- Mật khẩu được cập nhật trong database
- OTP được xóa khỏi hệ thống
- Người dùng có thể đăng nhập với mật khẩu mới

Các luồng sự kiện thay thế:

Luồng 3a (Bước 1): Email không đúng định dạng

- Hiển thị lỗi: "Email không đúng định dạng"
- Quay lại bước 2

Luồng 4a (Bước 1): Email không tồn tại

- Hiển thị thông báo: "Nếu email tồn tại trong hệ thống, link reset mật khẩu đã được gửi"
- Kết thúc (bảo mật - không tiết lộ email có tồn tại hay không)

Luồng 11a (Bước 2): OTP không tồn tại
- Hiển thị lỗi: "Mã OTP không tồn tại hoặc đã hết hạn"
- Quay lại bước 10

Luồng 11b (Bước 2): OTP hết hạn
- Hiển thị lỗi: "Mã OTP đã hết hạn. Vui lòng yêu cầu mã mới"
- Xóa OTP
- Cho phép quay lại bước 1

Luồng 11c (Bước 2): OTP không đúng
- Hiển thị lỗi: "Mã OTP không đúng"
- Quay lại bước 10

Luồng 16a (Bước 3): Mật khẩu quá ngắn
- Hiển thị lỗi: "Mật khẩu phải có ít nhất 6 ký tự"
- Quay lại bước 15

Luồng 16b (Bước 3): Mật khẩu không khớp
- Hiển thị lỗi: "Mật khẩu xác nhận không khớp"
- Quay lại bước 15

Yêu cầu đặc biệt:
- Bảo mật: Không tiết lộ email có tồn tại hay không
- OTP timeout: 10 phút
- UI/UX: Progress bar hiển thị 3 bước rõ ràng
- Email: Template email chuyên nghiệp

d) Use Case "Xác thực email"

Tên Use Case: Xác thực email

Tác nhân: Học sinh

Mô tả hoạt động: Xác nhận email của người dùng bằng mã 6 chữ số

Điều kiện trước:
- Người dùng đã đăng ký tài khoản

- Email chưa được xác thực (email_verified = false)

Luồng sự kiện chính:

- Người dùng yêu cầu gửi mã xác thực
- Hệ thống kiểm tra user tồn tại
- Hệ thống kiểm tra email chưa được xác thực
- Hệ thống tạo mã 6 chữ số ngẫu nhiên
- Hệ thống lưu vào bảng email_verifications:
  + user_id
  + email
  + code
  + expires_at = now + 5 phút
  + is_used = false
- Hệ thống gửi email chứa mã xác thực
- Hiển thị form nhập mã xác thực
- Người dùng nhập mã 6 chữ số
- Hệ thống validate:
  + Mã tồn tại
  + Mã chưa được sử dụng (is_used = false)
  + Mã chưa hết hạn
  + Mã khớp với database
- Hệ thống đánh dấu mã đã sử dụng:
  + is_used = true
  + used_at = now
- Hệ thống cập nhật users:
  + email_verified = true
  + email_verified_at = now
- Hiển thị thông báo "Xác thực email thành công!"

Điều kiện sau:

- email_verified = true trong bảng users
- Bản ghi trong email_verifications có is_used = true

- Người dùng có thể sử dụng đầy đủ chức năng (nếu cần xác thực email)

Các luồng sự kiện thay thế:

Luồng 2a: Email không tồn tại

- Hiển thị lỗi: "Email chưa được đăng ký"
- Kết thúc

Luồng 3a: Email đã được xác thực

- Hiển thị lỗi: "Email đã được xác thực"
- Kết thúc

Luồng 4-6a: Quá nhiều lần gửi (> 3 lần trong 1 giờ)

- Hiển thị lỗi: "Bạn đã yêu cầu quá nhiều lần. Vui lòng thử lại sau 1 giờ"
- Kết thúc

Luồng 8a: Mã không đúng 6 chữ số

- Hiển thị lỗi: "Mã xác thực phải có 6 chữ số"
- Quay lại bước 8

Luồng 9a: Mã không tồn tại hoặc đã sử dụng

- Hiển thị lỗi: "Mã xác thực không đúng hoặc đã được sử dụng"
- Quay lại bước 8

Luồng 9b: Mã hết hạn

- Hiển thị lỗi: "Mã xác thực đã hết hạn. Vui lòng yêu cầu mã mới"
- Cho phép gửi lại mã
- Quay lại bước 1

Yêu cầu đặc biệt:

- Timeout: Mã có hiệu lực 5 phút
- Rate limiting: Tối đa 3 lần gửi lại trong 1 giờ
- Security: Mã chỉ dùng được 1 lần

## 2. USE CASE LÀM BÀI KIỂM TRA

### e) Use Case "Chọn làm bài kiểm tra"

Tên Use Case: Chọn làm bài kiểm tra

Tác nhân: Học sinh

Mô tả hoạt động: Hiển thị danh sách bài kiểm tra và cho phép học sinh chọn bài muốn làm

Điều kiện trước:

- Người dùng đã đăng nhập (hoặc làm bài với tư cách guest)
- Hệ thống có ít nhất 1 đề thi active
- Database có ít nhất 50 câu hỏi active

Luồng sự kiện chính:

- Hệ thống hiển thị trang chọn khối lớp (10, 11, 12)
- Người dùng chọn khối lớp
- Hệ thống hiển thị:
  + Danh sách bài giảng theo khối
  + Danh sách đề thi theo khối
- Người dùng chọn "Làm đề thi"
- Hệ thống hiển thị danh sách các đề thi với:
  + Tên đề thi
  + Mô tả
  + Số câu hỏi (50 câu)
  + Thời gian (45 phút)
- Người dùng click "Bắt đầu làm bài" trên đề thi mong muốn
- Hệ thống chuyển sang use case "Làm bài"

Điều kiện sau:

- Người dùng đã chọn xong đề thi
- Chuẩn bị chuyển sang màn hình làm bài

Các luồng sự kiện thay thế:

Luồng 3a: Không có bài kiểm tra nào

- Hiển thị thông báo: "Hiện tại không có đề thi nào đang hoạt động. Vui lòng liên hệ quản trị viên"
- Kết thúc

Luồng 3b: Không đủ câu hỏi trong database

- Hiển thị thông báo: "Không đủ câu hỏi để tạo đề thi. Vui lòng liên hệ quản trị viên"

- Kết thúc

Yêu cầu đặc biệt:

- UI/UX: Cards đẹp, responsive
- Performance: Load danh sách < 1 giây

f) Use Case "Làm bài"

Tên Use Case: Làm bài

Tác nhân: Học sinh

Mô tả hoạt động: Thực hiện bài kiểm tra với 50 câu hỏi trong thời gian 45 phút

Điều kiện trước:

- Người dùng đã chọn đề thi
- Database có đủ 50 câu hỏi active cho môn học
- Tất cả câu hỏi đều có đáp án

Luồng sự kiện chính:

1. Backend - Khởi tạo session:

- Lấy subject_id (môn Tin học)
- Lấy test_id (nếu có) hoặc random test theo grade_level
- Query tất cả câu hỏi active của môn
- Validate ≥ 50 câu hỏi available
- Validate tất cả câu hỏi có đáp án
- Random chọn 50 câu hỏi
- Shuffle thứ tự câu hỏi
- Ẩn is_correct từ answers (bảo mật)
- Tạo session_token (64 ký tự hex)
- Tính thời gian:
  + started_at = now
  + expires_at = now + 45 phút
- Lưu vào test_sessions với status = 'in_progress'
- Nếu user chưa đăng nhập → tạo guest user
- Trả về:
  + session_id

+ session_token

+ questions

+ duration (45)

+ expires_at

2. Frontend - Hiển thị bài thi:

- Nhận data từ API

- Lưu session_token vào localStorage

- Khởi tạo state answers = {} (rỗng)

- Khởi tạo countdown timer = 45 phút

- Hiển thị 50 câu hỏi với:

+ Số thứ tự (1-50)

+ Nội dung câu hỏi

+ 4 đáp án (A, B, C, D)

+ Radio button chọn đáp án

- Hiển thị đồng hồ đếm ngược

- Hiển thị progress (X/50 câu đã trả lời)

3. Người dùng làm bài:

- Đọc câu hỏi

- Click chọn đáp án

- Frontend cập nhật state answers

- Auto-save vào localStorage mỗi 5 giây

- Có thể thay đổi đáp án bất kỳ lúc nào

- Có thể cuộn lên/xuống xem lại câu trước

4. Theo dõi thời gian:

- Countdown timer giảm dần

- Hiển thị thời gian còn lại (MM:SS)

- Cảnh báo khi còn 5 phút (màu vàng)

- Cảnh báo khi còn 1 phút (màu đỏ)

5. Người dùng click "Nộp bài" hoặc hết thời gian:

- Chuyển sang use case "Nộp bài"

Điều kiện sau:

- Answers được lưu trong state
- Session đang active
- Countdown timer đang chạy

Các luồng sự kiện thay thế:

Luồng 1a: Không đủ 50 câu hỏi

- Trả về lỗi 400: "Không đủ câu hỏi để tạo đề thi (hiện có X câu, cần ít nhất 50 câu)"
- Kết thúc

Luồng 1b: Có câu hỏi không có đáp án

- Trả về lỗi 500: "Một số câu hỏi không có đáp án. Vui lòng liên hệ quản trị viên"
- Log câu hỏi lỗi
- Kết thúc

Luồng 3a: Mất kết nối Internet

- Answers vẫn được lưu trong localStorage
- Hiển thị cảnh báo "Mất kết nối"
- Khi có kết nối lại → sync data
- Tiếp tục làm bài

Luồng 4a: Hết 45 phút

- Frontend tự động gọi API nộp bài
- Chuyển sang use case "Nộp bài"

Yêu cầu đặc biệt:

- Performance: Render 50 câu < 2 giây
- Auto-save: Mỗi 5 giây lưu localStorage
- Timer: Chính xác đến giây
- Security: Không gửi is_correct về frontend
- Session timeout: 45 phút kể từ started_at

g) Use Case "Nộp bài"

Tên Use Case: Nộp bài

Tác nhân: Học sinh

Mô tả hoạt động: Chấm điểm, đánh giá chuẩn đầu ra và lưu kết quả bài kiểm tra

Điều kiện trước:

- Người dùng đã làm bài (có session_id)
- Session tồn tại và status = 'in_progress'
- Answers đã được submit (có thể rỗng)

Luồng sự kiện chính:

1. Frontend - Chuẩn bị data:
   - Thu thập answers từ state
   - Gọi API: POST /api/exam/submit/:sessionId
   - Body: {answers: {question_id: answer_label}}
2. Backend - Validate:
   - Lấy session từ database bằng session_id
   - Kiểm tra session tồn tại
   - Kiểm tra session chưa bị nộp (status = 'in_progress')
   - Kiểm tra session chưa duplicate (chưa có test_result)
   - Parse questions_data từ session
3. Backend - Lấy đáp án đúng:
   - Extract danh sách question_ids từ questions_data
   - Query bảng answers với question_id, is_correct = true
   - Tạo Map: correctAnswersMap[question_id] = correct_answer_label
4. Backend - Chấm điểm:
   - Khởi tạo:
     + correctCount = 0
     + totalQuestions = số câu trong đề thi
   - Duyệt qua từng câu hỏi:
     + So sánh user_answer_label với correct_answer_label
     + Nếu khớp thì correctCount++
   - Tính điểm: score = (correctCount / totalQuestions) × 10
   - Làm tròn 1 chữ số thập phân

5. Backend - Đánh giá chuẩn đầu ra:
   - passed = score >= 7.0
   - Nếu score >= 7.0 → "ĐẠT CHUẨN ĐẦU RA"
   - Nếu score < 7.0 → "CHƯA ĐẠT CHUẨN ĐẦU RA"
6. Backend - Lưu kết quả:
   - Insert vào bảng test_results:
     + session_id
     + user_id
     + test_id
     + score
     + correct_answers
     + wrong_answers
     + unanswered
     + passes_cdr
     + answers_data
     + submitted_at
     + completed_at
   - Update test_sessions:
     + status = 'completed'
     + answers_data
     + submitted_at
7. Backend - Tạo detailed results:
   - Duyệt qua tất cả câu hỏi
   - Với mỗi câu, tạo object:
     + question_id
     + question_text
     + user_answer
     + correct_answer
     + is_correct
     + answers

8. Backend - Trả về response:

- success, message
- data gồm:
  + result_id
  + score
  + total_questions
  + correct_answers
  + passed
  + detailed_results
  + completed_at

9. Frontend - Hiển thị kết quả:

- Lưu kết quả vào localStorage
- Chuyển hướng đến ResultPage
- Hiển thị điểm số lớn
- Hiển thị badge:
  + Xanh lá + "ĐẠT" nếu score >= 7.0
  + Đỏ + "CHƯA ĐẠT" nếu score < 7.0
- Hiển thị thống kê: X câu đúng/tổng số câu
- Chuyển sang use case "Xem điểm"

Điều kiện sau:

- Bản ghi test_results được tạo
- Session status = 'completed'
- Kết quả được lưu vĩnh viễn
- Người dùng xem được điểm và đánh giá CĐR

Các luồng sự kiện thay thế:

Luồng 2a: Session không tồn tại

- Trả về lỗi 404: "Không tìm thấy phiên làm bài"
- Kết thúc

Luồng 2b: Session đã nộp rồi (status = 'completed')

- Trả về lỗi 400: "Bài kiểm tra đã được nộp trước đó"

- Kết thúc

Luồng 2c: Duplicate submission (đã có test_result)

- Trả về lỗi 400: "Bài kiểm tra đã được nộp"

- Kết thúc

Luồng 3a: Không lấy được đáp án đúng

- Log lỗi

- Trả về lỗi 500: "Lỗi khi lấy đáp án đúng"

- Kết thúc

Yêu cầu đặc biệt:

- Tính điểm linh hoạt:

  + Công thức: score = (số câu đúng / tổng số câu) × 10

  + Ví dụ 50 câu: 1 câu = 0.2 điểm

  + Ví dụ 10 câu: 1 câu = 1.0 điểm

  + Làm tròn 1 chữ số thập phân

- Ngưỡng CĐR: score >= 7.0 (70% số câu)

- Security:

  + Prevent duplicate submission

  + Validate session ownership

- Transaction: Atomic operation (lưu results + update session)

h) Use Case "Xem điểm có đạt chuẩn đầu ra không"

Tên Use Case: Xem điểm có đạt chuẩn đầu ra không

Tác nhân: Học sinh

Mô tả hoạt động: Hiển thị điểm số, trạng thái đạt/không đạt chuẩn đầu ra, thống kê chi tiết và gợi ý bài học cần ôn tập

Điều kiện trước:

- Bài kiểm tra đã được nộp và chấm điểm

- Có result_id hoặc session_id

Luồng sự kiện chính:

1. Hiển thị điểm tổng quan:

   - Điểm số lớn, rõ ràng (VD: 7.2/10)

- Badge đánh giá:
  + Nếu score >= 7.0:
    - Màu xanh lá
    - Chữ "ĐẠT CHUẨN ĐẦU RA"
    - Icon check
  + Nếu score < 7.0:
    - Màu đỏ
    - Chữ "CHƯA ĐẠT CHUẨN ĐẦU RA"
    - Icon X

2. Hiển thị thống kê:
- Số câu đúng: X/Y (VD: 36/50)
- Số câu sai: Z (VD: 14)
- Tỷ lệ đúng: W% (VD: 72%)
- Thời gian hoàn thành

3. Hiển thị yêu cầu đạt chuẩn đầu ra:
- Tính số câu cần: required = Math.ceil(tổng số câu × 0.7)
- Nếu đạt (score >= 7.0):
  + "Bạn cần ≥ {required}/{total} câu đúng (≥ 7.0 điểm) để đạt CĐR. Bạn đã vượt qua tiêu chuẩn này!"
- Nếu chưa đạt (score < 7.0):
  + "Bạn cần ≥ {required}/{total} câu đúng (≥ 7.0 điểm) để đạt CĐR. Hãy cố gắng thêm!"

4. Gọi API gợi ý bài học:
- Endpoint: GET /api/exam/recommendations/lessons?score={score}&test_id={test_id}
- Backend xử lý:
  + Xác định message dựa vào điểm số:
    - score < 4.0: "Hãy ôn tập các bài học dưới đây để đạt chuẩn đầu ra (7.0 điểm)"
    - 4.0 <= score < 6.0: "Bạn đã có nền tảng tốt! Ôn thêm các bài sau để đạt CĐR"

- 6.0 <= score < 7.0: "Bạn gần đạt CĐR rồi! Ôn thêm một chút để đạt 7.0 điểm"

- 7.0 <= score < 8.0: "Chúc mừng bạn đã đạt CĐR! Ôn thêm để đạt 8.0 điểm"

- 8.0 <= score < 9.0: "Xuất sắc! Ôn thêm để đạt 9.0 điểm"

- 9.0 <= score < 10.0: "Rất xuất sắc! Ôn thêm để đạt điểm tuyệt đối 10.0"

- score = 10.0: "Hoàn hảo! Bạn có thể thử các bài học nâng cao"

+ Lấy danh sách bài học từ lesson_tests (liên kết chính xác với đề thi)

+ Trả về: message, target_score, danh sách lessons

5. Hiển thị gợi ý bài học:

- Hiển thị trong box màu cam nổi bật
- Tiêu đề: "Cần ôn tập thêm"
- Message gợi ý từ backend
- Danh sách bài học:
  + Tên bài học (lesson_title)
  + Mô tả ngắn (lesson_description)
  + Lớp học (grade_level)
  + Link trực tiếp đến trang chi tiết bài học
- Có thể click vào từng bài để học ngay

6. Hiển thị các nút hành động:

- "Xem đáp án chi tiết" → Use case "Xem đáp án"
- "Làm lại" → Quay về làm bài mới
- "Lịch sử làm bài" → Trang lịch sử (nếu đã đăng nhập)
- "Về trang chủ" → Quay về trang chủ

Điều kiện sau:

- Người dùng hiểu rõ kết quả của mình
- Người dùng biết có đạt chuẩn đầu ra hay không
- Người dùng nhận được gợi ý bài học phù hợp với trình độ
- Người dùng có thể học ngay các bài được gợi ý
- Người dùng có thể xem chi tiết hoặc làm bài mới

Yêu cầu đặc biệt:

- UI/UX:
  + Điểm số rất lớn, dễ nhìn (font-size: 48px+)
  + Badge nổi bật
  + Màu sắc trực quan (xanh = đạt, đỏ = chưa đạt)
  + Box gợi ý bài học nổi bật (màu cam)
  + Animation khi hiển thị điểm
- Thông tin rõ ràng:
  + Hiển thị số câu cần đạt động theo tổng số câu
  + Giải thích ngưỡng: "Cần ≥ {required}/{total} câu đúng để đạt CĐR"
  + Gợi ý bài học chính xác từ liên kết lesson_tests
- Performance:
  + Load gợi ý bài học tự động khi vào trang
  + Hiển thị loading state khi đang fetch

i) Use Case "Xem đáp án"

Tên Use Case: Xem đáp án

Tác nhân: Học sinh

Mô tả hoạt động: Hiển thị chi tiết tất cả câu hỏi với đáp án đã chọn, đáp án đúng và trạng thái đúng/sai

Điều kiện trước:

- Người dùng đã xem điểm
- Có detailed_results trong response

Luồng sự kiện chính:

1. Hiển thị danh sách câu hỏi:
   - Duyệt qua detailed_results
   - Hiển thị tất cả câu hỏi trong đề thi
2. Thông tin câu hỏi:
   - Số thứ tự: "Câu X"
   - Nội dung câu hỏi (question_text)
   - Trạng thái đúng/sai với icon:
     + Check màu xanh nếu is_correct = true

+ X màu đỏ nếu is_correct = false

3. Hiển thị 4 đáp án:

- Duyệt qua answers (A, B, C, D)
- Với mỗi đáp án:
  + Label (A, B, C, D)
  + Text (answer_text)
  + Styling dựa trên trạng thái

4. Màu sắc đáp án:

- Đáp án đúng (correct_answer):
  + Background: xanh lá nhạt
  + Border: xanh lá đậm
  + Text: xanh lá
  + Icon: check màu xanh
  + Label: "Đáp án đúng"
- Đáp án người dùng chọn (user_answer):
  + Nếu đúng (= correct_answer):
    - Giống đáp án đúng (xanh lá)
  + Nếu sai (≠ correct_answer):
    - Background: đỏ nhạt
    - Border: đỏ đậm
    - Text: đỏ
    - Icon: X màu đỏ
    - Label: "Bạn đã chọn"
- Đáp án khác:
  + Background: trắng/xám nhạt
  + Border: xám
  + Text: xám

5. Ví dụ minh họa:

Câu 1 [✓]

Trong Python, hàm nào dùng để in ra màn hình?

A. input() [xám - đáp án khác]

B. print() [xanh lá - đáp án đúng]

C. echo() [xám - đáp án khác]

D. display() [xám - đáp án khác]

Câu 2 [✗]

Vòng lặp for trong Python dùng để làm gì?

A. Thoát chương trình [đỏ - bạn đã chọn]

B. Lặp qua dãy [xanh lá - đáp án đúng]

C. Khai báo biến [xám - đáp án khác]

D. Nhập dữ liệu [xám - đáp án khác]

6. Tính năng bổ sung:

- Scroll smooth khi click câu hỏi
- Highlight câu đang xem
- Nút "Ẩn đáp án" để collapse chi tiết
- Nút "Quay lại kết quả tổng quan"

Điều kiện sau:

- Người dùng xem được chi tiết tất cả câu hỏi
- Người dùng hiểu rõ sai ở đâu
- Người dùng học được từ đáp án đúng

Yêu cầu đặc biệt:

- UI/UX:
  + Màu sắc rõ ràng, dễ phân biệt
  + Layout sạch sẽ, dễ đọc
  + Responsive trên mobile
- Performance:
  + Render mượt mà
  + Smooth scrolling
- Accessibility:
  + High contrast colors
  + Screen reader friendly

j) Use Case "Xem lịch sử làm bài"

Tên Use Case: Xem lịch sử làm bài

Tác nhân: Học sinh

Mô tả hoạt động: Hiển thị danh sách tất cả các lần làm bài kiểm tra của học sinh

Điều kiện trước:

- Người dùng đã đăng nhập
- Có JWT token hợp lệ

Luồng sự kiện chính:

1. Frontend - Gọi API:
- GET /api/history/
- Headers: Authorization: Bearer {token}
2. Backend - Query database:
- Lấy user_id từ JWT token
- Query test_results với:
  + user_id = current user
  + JOIN với bảng tests để lấy test_name
  + ORDER BY completed_at DESC (mới nhất trước)
- Với mỗi result:
  + Tính passed = score >= 7.0
  + Format completed_at thành human-readable
3. Backend - Trả về response:
- success: true
- data: Array gồm
  + result_id
  + test_name
  + score
  + correct_answers
  + total_questions (50)
  + passed
  + completed_at

- total: số lượng kết quả

4. Frontend - Hiển thị danh sách:

- Tiêu đề: "Lịch sử làm bài"

- Table hoặc Cards với columns:

  + Đề thi: test_name

  + Điểm số: score/10 với màu:

    - Xanh lá nếu >= 7.0

    - Đỏ nếu < 7.0

  + Kết quả:

    - Badge "ĐẠT" (xanh) nếu passed = true

    - Badge "CHƯA ĐẠT" (đỏ) nếu passed = false

  + Số câu đúng: X/50

  + Thời gian: Format ngày giờ (DD/MM/YYYY HH:mm)

5. Học sinh xem kết quả:

- Trang chỉ hiển thị lịch sử (không có nút xem chi tiết)

- Có thể so sánh điểm các lần thi

Điều kiện sau:

- Người dùng xem được tất cả lịch sử làm bài

- Người dùng theo dõi được tiến bộ

Các luồng sự kiện thay thế:

Luồng 1a: Chưa đăng nhập

- Redirect về trang login

- Kết thúc

Luồng 1b: Token hết hạn

- Trả về lỗi 401: Unauthorized

- Frontend logout và redirect về login

Luồng 2a: User chưa làm bài nào

- Trả về empty array: []

- Frontend hiển thị:

  + Icon trống

+ Text: "Bạn chưa làm bài kiểm tra nào"

+ Nút "Làm bài ngay"

Yêu cầu đặc biệt:

- Sắp xếp: Mới nhất lên đầu
- Pagination: Nếu > 20 kết quả
- Security: Chỉ xem được kết quả của chính mình

## 3. USE CASE QUẢN TRỊ VIÊN

### k) Use Case "Thêm câu hỏi"

Tên Use Case: Thêm câu hỏi

Tác nhân: Quản trị viên

Mô tả hoạt động: Tạo câu hỏi trắc nghiệm mới với 4 đáp án

Điều kiện trước:

- Quản trị viên đã đăng nhập
- Role = "admin"
- Token hợp lệ

Luồng sự kiện chính:

- Admin click "Thêm câu hỏi" trong tab Questions
- Hệ thống hiển thị modal QuestionModal với form:
  + Nội dung câu hỏi (textarea, required, ≥10 ký tự)
  + Độ khó (select: easy/medium/hard, required)
  + Khối lớp (select: 10, 11, 12, required)
  + Chủ đề (input text, optional)
  + 4 đáp án (A, B, C, D) - mỗi đáp án:
    - answer_text (required)
    - Radio button chọn đáp án đúng (1 trong 4)
- Admin nhập đầy đủ thông tin
- Admin chọn 1 đáp án đúng
- Admin click "Lưu"
- Frontend validate:
  + Câu hỏi ≥ 10 ký tự

+ 4 đáp án đều có text

+ Đã chọn đáp án đúng

- Frontend gọi API: POST /api/admin/questions

- Backend validate:

+ Token + role = admin

+ Dữ liệu hợp lệ

- Backend insert vào bảng questions:

+ subject_id = 1 (Tin học - hardcode)

+ question_text

+ grade_level

+ difficulty

+ topic (nếu có)

+ is_active = true

- Backend insert vào bảng answers (4 records):

+ question_id

+ answer_text

+ answer_label (A/B/C/D)

+ is_correct (true cho 1, false cho 3)

- Backend trả về success + question data

- Frontend đóng modal, refresh danh sách câu hỏi

- Hiển thị toast: "Đã thêm câu hỏi"

Điều kiện sau:

- Câu hỏi mới được lưu vào database

- 4 đáp án được lưu với 1 đáp án đúng

- Câu hỏi available cho các đề thi

Các luồng sự kiện thay thế:

Luồng 6a: Thiếu thông tin

- Hiển thị lỗi tại trường thiếu

- Quay lại bước 3

Luồng 6b: Chưa chọn đáp án đúng

- Hiển thị lỗi: "Vui lòng chọn đáp án đúng"
- Quay lại bước 4

Luồng 9-10a: Lỗi database
- Rollback transaction
- Hiển thị: "Lỗi khi thêm câu hỏi"

Yêu cầu đặc biệt:
- Transaction: Insert question + 4 answers atomically
- Validation: Đảm bảo có đúng 1 đáp án đúng
- Subject: Hardcode subject_id = 1 (Tin học)

l) Use Case "Sửa câu hỏi"

Tên Use Case: Sửa câu hỏi

Tác nhân: Quản trị viên

Mô tả hoạt động: Chỉnh sửa nội dung câu hỏi, đáp án hoặc đáp án đúng

Điều kiện trước:
- Câu hỏi tồn tại trong database
- Admin đã đăng nhập

Luồng sự kiện chính:
- Admin click nút Edit trên câu hỏi
- Backend load câu hỏi + 4 đáp án
- Modal hiển thị form với data có sẵn
- Admin sửa thông tin cần thiết
- Admin click "Cập nhật"
- Backend validate
- Backend update bảng questions
- Backend update bảng answers (4 records)
- Refresh danh sách
- Toast: "Đã cập nhật câu hỏi"

Điều kiện sau:
- Thông tin câu hỏi được cập nhật
- Các đề thi sử dụng câu hỏi này sẽ có nội dung mới

Các luồng sự kiện thay thế:

Luồng 2a: Câu hỏi không tồn tại

- Hiển thị: "Câu hỏi không tồn tại"
- Kết thúc

m) Use Case "Xóa câu hỏi"

Tên Use Case: Xóa câu hỏi

Tác nhân: Quản trị viên

Mô tả hoạt động: Xóa câu hỏi và các đáp án liên quan

Điều kiện trước:

- Câu hỏi tồn tại
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click nút Delete
- Hiển thị confirm: "Xóa câu hỏi này? Câu hỏi sẽ bị xóa khỏi tất cả đề thi và bài giảng."
- Admin confirm
- Backend xóa từ bảng questions (CASCADE sẽ xóa answers)
- Refresh danh sách
- Toast: "Đã xóa câu hỏi"

Điều kiện sau:

- Câu hỏi bị xóa vĩnh viễn
- Đáp án bị xóa (CASCADE)
- Các liên kết test_questions, lesson_questions bị xóa (CASCADE)

Yêu cầu đặc biệt:

- Confirmation dialog
- CASCADE delete

n) Use Case "Sửa bài kiểm tra"

Tên Use Case: Sửa bài kiểm tra

Tác nhân: Quản trị viên

Mô tả hoạt động: Chỉnh sửa thông tin đề thi

Điều kiện trước:

- Đề thi tồn tại trong database
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click Edit trên đề thi trong tab Tests
- Backend load thông tin test
- Modal TestModal hiển thị form với data có sẵn:
  + Tên đề thi
  + Mô tả
  + Trạng thái (active/inactive)
- Admin sửa thông tin
- Admin click "Cập nhật"
- Backend update tests
- Refresh danh sách
- Toast: "Đã cập nhật đề thi"

Điều kiện sau:

- Thông tin đề thi được cập nhật
- Học sinh thấy thay đổi ngay lập tức

o) Use Case "Xóa bài kiểm tra"

Tên Use Case: Xóa bài kiểm tra

Tác nhân: Quản trị viên

Mô tả hoạt động: Xóa đề thi khỏi hệ thống

Điều kiện trước:

- Đề thi tồn tại
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click Delete
- Confirm dialog
- Backend kiểm tra:
  + Đếm test_results có test_id này

- Nếu có kết quả:
  + Hiển thị: "Không thể xóa đề thi đã có X kết quả. Hãy set trạng thái Inactive"
  + Kết thúc
- Nếu chưa có results:
  + Delete từ tests (CASCADE xóa test_questions)
  + Toast: "Đã xóa đề thi"

Điều kiện sau:

- Đề thi bị xóa vĩnh viễn (nếu chưa có kết quả)
- Các liên kết test_questions bị xóa (CASCADE)

Các luồng thay thế:

Luồng 3a: Đã có kết quả

- Hiển thị: "Đề thi đã có kết quả, không thể xóa. Hãy set trạng thái Inactive"
- Kết thúc

p) Use Case "Quản lý câu hỏi trong đề thi"

Tên Use Case: Quản lý câu hỏi trong đề thi

Tác nhân: Quản trị viên

Mô tả hoạt động: Xem, thêm, xóa câu hỏi thuộc một đề thi cụ thể

Điều kiện trước:

- Đề thi tồn tại
- Admin đã đăng nhập

Luồng sự kiện chính:

1. Xem danh sách câu hỏi của đề thi:
- Admin chọn đề thi và xem chi tiết
- Backend gọi: GET /api/admin/tests/:testId/questions
- Hiển thị danh sách câu hỏi đã được gán cho đề thi này
- Hiển thị question_order (thứ tự câu hỏi)

2. Thêm câu hỏi vào đề thi:
- Admin click "Thêm câu hỏi vào đề"
- Chọn câu hỏi từ danh sách available
- Backend gọi: POST /api/admin/tests/:testId/questions

- Body: {question_id, question_order}
- Insert vào test_questions
- Refresh danh sách
3. Xóa câu hỏi khỏi đề thi:
- Admin click "Xóa" trên câu hỏi
- Confirm dialog
- Backend gọi: DELETE /api/admin/tests/:testId/questions/:questionId
- Delete từ test_questions
- Refresh danh sách
Điều kiện sau:
- Đề thi có danh sách câu hỏi được quản lý
- Câu hỏi có thể thêm/xóa khỏi đề
Yêu cầu đặc biệt:
- Chỉ quản lý liên kết, không xóa câu hỏi khỏi database
- Hỗ trợ sắp xếp thứ tự (question_order)
q) Use Case "Sửa bài giảng"
Tên Use Case: Sửa bài giảng
Tác nhân: Quản trị viên
Mô tả hoạt động: Chỉnh sửa thông tin và nội dung bài giảng
Điều kiện trước:
- Bài giảng tồn tại trong database
- Admin đã đăng nhập
Luồng sự kiện chính:
- Admin click nút Edit trên bài giảng trong tab Lessons
- Backend load bài giảng:
  + GET /api/admin/lessons/:id (endpoint không có nhưng có thể dùng getAllLessons)
- Modal LessonModal hiển thị form với data có sẵn:
  + Tên bài giảng
  + Nội dung chi tiết (content)

+ Trạng thái

- Admin sửa thông tin cần thiết
- Admin click "Cập nhật"
- Frontend validate
- Frontend gọi API: PUT /api/admin/lessons/:id
- Backend validate:

+ Token + role = admin
+ Lesson tồn tại
+ Dữ liệu hợp lệ

- Backend update bảng lessons
- Trả về success
- Frontend đóng modal, refresh danh sách
- Toast: "Đã cập nhật bài giảng"

Điều kiện sau:

- Thông tin bài giảng được cập nhật
- Học sinh thấy nội dung mới ngay lập tức

Các luồng sự kiện thay thế:

Luồng 2a: Bài giảng không tồn tại

- Hiển thị: "Bài giảng không tồn tại"
- Kết thúc

r) Use Case "Xóa bài giảng"

Tên Use Case: Xóa bài giảng

Tác nhân: Quản trị viên

Mô tả hoạt động: Xóa bài giảng khỏi hệ thống

Điều kiện trước:

- Bài giảng tồn tại
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click nút Delete trên bài giảng
- Hiển thị confirm dialog:

+ "Xóa bài giảng này?"

+ "Bài giảng sẽ bị xóa vĩnh viễn."

- Admin confirm (click "Xác nhận xóa")
- Frontend gọi API: DELETE /api/admin/lessons/:id
- Backend validate:

+ Token + role = admin

+ Lesson tồn tại

- Backend xóa từ bảng lessons (CASCADE):

+ Xóa lessons

+ Xóa lesson_questions (CASCADE)

- Trả về success
- Frontend refresh danh sách
- Toast: "Đã xóa bài giảng"

Điều kiện sau:

- Bài giảng bị xóa vĩnh viễn
- Các liên kết lesson_questions bị xóa (CASCADE)
- Học sinh không còn thấy bài giảng

Yêu cầu đặc biệt:

- Confirmation dialog
- CASCADE delete cho lesson_questions

t) Use Case "Quản lý người dùng"

Tên Use Case: Quản lý người dùng

Tác nhân: Quản trị viên

Mô tả hoạt động: Xem, tìm kiếm, lọc và quản lý tài khoản

Điều kiện trước:

- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin vào tab "Người dùng"
- Hệ thống load tất cả users (GET /api/admin/users)
- Hiển thị:

+ Search box (tìm theo email/tên) - Frontend filter

+ Filter role (All/Student/Teacher/Admin) - Frontend filter

+ Filter status (All/Active/Inactive) - Frontend filter

+ Table với columns:

- Email

- Họ tên

- Vai trò (badge màu)

- Trạng thái (Active/Inactive)

- Ngày tạo

- Hành động (Edit/Lock/Delete)

- Admin có thể:

+ Tìm kiếm: Gõ vào search box → filter real-time (Frontend)

+ Lọc: Chọn role/status → filter danh sách (Frontend)

+ Sửa: Click Edit → Modal sửa thông tin (email, họ tên, role)

+ Khóa/Mở khóa: Click Lock/Unlock → Toggle status

+ Xóa: Click Delete → Xóa user (nếu không có test_results)

Điều kiện sau:

- Admin quản lý được tất cả users

- Có thể tìm kiếm, lọc, CRUD

x) Use Case "Thêm câu hỏi, trả lời cho ChatBot AI"

Tên Use Case: Thêm câu hỏi, trả lời cho ChatBot AI

Tác nhân: Quản trị viên

Mô tả hoạt động: Tạo cặp câu hỏi-trả lời mới cho hệ thống ChatBot hỗ trợ học sinh

Điều kiện trước:

- Admin đã đăng nhập

- Role = "admin"

Luồng sự kiện chính:

- Admin vào tab "ChatBot" trong Admin Dashboard

- Click "Thêm Q&A"

- Hệ thống hiển thị modal ChatBotModal với form:
  + Câu hỏi mẫu (question, textarea, required)
  + Câu trả lời (answer, textarea, required)
- Admin nhập đầy đủ thông tin
- Admin click "Lưu"
- Frontend validate:
  + Câu hỏi không trống
  + Câu trả lời không trống
- Frontend gọi API: POST /api/admin/chatbot
- Backend validate:
  + Token + role = admin
  + Dữ liệu hợp lệ
- Backend insert vào bảng chatbot_qa:
  + question
  + answer
  + created_at
- Backend trả về success + QA data
- Frontend đóng modal, refresh danh sách Q&A
- Toast: "Đã thêm câu hỏi ChatBot"

Điều kiện sau:

- Cặp Q&A mới được lưu vào database
- ChatBot có thể sử dụng để trả lời học sinh
- Hiển thị trong danh sách quản lý ChatBot

Các luồng sự kiện thay thế:

Luồng 6a: Thiếu thông tin bắt buộc

- Hiển thị lỗi tại trường thiếu
- Quay lại bước 3

Luồng 9a: Lỗi database

- Rollback transaction
- Hiển thị: "Lỗi khi thêm câu hỏi ChatBot"

y) Use Case "Sửa câu hỏi, trả lời cho ChatBot AI"

Tên Use Case: Sửa câu hỏi, trả lời cho ChatBot AI

Tác nhân: Quản trị viên

Mô tả hoạt động: Chỉnh sửa câu hỏi hoặc câu trả lời của ChatBot

Điều kiện trước:

- Q&A tồn tại trong database
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click nút Edit trên Q&A trong danh sách ChatBot
- Backend load Q&A (từ getAllChatbotQA)
- Modal hiển thị form với data có sẵn:
  + Câu hỏi mẫu
  + Câu trả lời
- Admin sửa thông tin cần thiết
- Admin click "Cập nhật"
- Frontend validate
- Frontend gọi API: PUT /api/admin/chatbot/:id
- Backend validate:
  + Token + role = admin
  + Q&A tồn tại
  + Dữ liệu hợp lệ
- Backend update bảng chatbot_qa:
  + question
  + answer
  + updated_at
- Trả về success
- Frontend đóng modal, refresh danh sách
- Toast: "Đã cập nhật câu hỏi ChatBot"

Điều kiện sau:

- Thông tin Q&A được cập nhật

- ChatBot sử dụng câu trả lời mới ngay lập tức

Các luồng sự kiện thay thế:

Luồng 2a: Q&A không tồn tại

- Hiển thị: "Câu hỏi không tồn tại"
- Kết thúc

z) Use Case "Xóa câu hỏi, trả lời cho ChatBot AI"

Tên Use Case: Xóa câu hỏi, trả lời cho ChatBot AI

Tác nhân: Quản trị viên

Mô tả hoạt động: Xóa cặp Q&A khỏi hệ thống ChatBot

Điều kiện trước:

- Q&A tồn tại
- Admin đã đăng nhập

Luồng sự kiện chính:

- Admin click nút Delete trên Q&A
- Hiển thị confirm dialog:
  + "Xóa câu hỏi này khỏi ChatBot?"
  + "ChatBot sẽ không thể trả lời câu hỏi này nữa."
- Admin confirm (click "Xác nhận xóa")
- Frontend gọi API: DELETE /api/admin/chatbot/:id
- Backend validate:
  + Token + role = admin
  + Q&A tồn tại
- Backend xóa từ bảng chatbot_qa
- Trả về success
- Frontend refresh danh sách
- Toast: "Đã xóa câu hỏi ChatBot"

Điều kiện sau:

- Q&A bị xóa vĩnh viễn
- ChatBot không còn trả lời câu hỏi này

y) Use Case "Xem thống kê dữ liệu"

Tên Use Case: Xem thống kê dữ liệu

Tác nhân: Quản trị viên

Mô tả hoạt động: Hiển thị tổng quan thống kê hệ thống

Điều kiện trước:

- Admin đã đăng nhập
- Role = "admin"

Luồng sự kiện chính:

- Admin vào tab "Thống kê" trong Admin Dashboard
- Hệ thống tự động load thống kê:
  + GET /api/admin/stats
- Backend query database:
  + Đếm tổng số người dùng (users):
    - Total users (active + inactive)
    - Active users
  + Đếm tổng số câu hỏi (questions):
    - Total questions (active)
  + Đếm tổng số đề thi (tests):
    - Total tests (active + inactive)
  + Đếm tổng số bài giảng (lessons):
    - Total lessons (active + inactive)
- Backend trả về response với thống kê
- Frontend hiển thị Dashboard với:
  + Cards tổng quan (overview cards):
    - Tổng số người dùng (icon: Users)
    - Tổng số câu hỏi (icon: HelpCircle)
    - Tổng số đề thi (icon: FileText)
    - Tổng số bài giảng (icon: BookOpen)
  + Hiển thị số lượng ở từng card

Điều kiện sau:

- Admin xem được tổng quan toàn hệ thống

- Admin biết số lượng dữ liệu trong hệ thống

Yêu cầu đặc biệt:

- Performance: Query tối ưu với count queries
- UI/UX: Cards đẹp, dễ nhìn
- Real-time: Số liệu được cập nhật khi load tab

w) Use Case "Quản lý Chuẩn Đầu Ra (Learning Outcomes)"

Tên Use Case: Quản lý Chuẩn Đầu Ra

Tác nhân: Quản trị viên

Mô tả hoạt động: Xem, thêm, sửa, xóa chuẩn đầu ra cho môn học

Điều kiện trước:

- Admin đã đăng nhập
- Role = "admin"

Luồng sự kiện chính:

1. Xem danh sách CĐR:

- Admin vào tab "Chuẩn đầu ra"
- Hệ thống gọi: GET /api/admin/learning-outcomes
- Hiển thị danh sách với columns:
  + Mã CĐR (outcome_code)
  + Tiêu đề (outcome_title)
  + Điểm đạt (passing_score)
  + Khối lớp (grade_levels)
  + Số lượng sử dụng (usage_count: tests + lessons)
  + Trạng thái (is_active)
  + Hành động (Edit/Delete)

2. Thêm CĐR mới:

- Admin click "Thêm CDR"
- Modal hiển thị form:
  + Mã CĐR (required, unique)
  + Tiêu đề (required)
  + Mô tả (optional)

+ Điểm đạt (default: 7.0, range: 0-10)

+ Khối lớp (multi-select: 10, 11, 12)

+ Trạng thái (active/inactive)

- Admin nhập và click "Lưu"

- Backend gọi: POST /api/admin/learning-outcomes

- Backend validate outcome_code unique

- Insert vào learning_outcomes

- Toast: "Đã thêm chuẩn đầu ra"

3. Sửa CĐR:

- Admin click Edit trên CĐR

- Modal hiển thị form với data có sẵn

- Admin sửa (có thể sửa passing_score từ 7.0 → 8.0)

- Backend gọi: PUT /api/admin/learning-outcomes/:id

- Backend update learning_outcomes

- Toast: "Đã cập nhật chuẩn đầu ra"

4. Xóa CĐR:

- Admin click Delete

- Backend kiểm tra usage_count:

+ Đếm tests có outcome_id này

+ Đếm lessons có outcome_id này

- Nếu có sử dụng:

+ Hiển thị: "Không thể xóa chuẩn đầu ra này vì đang được sử dụng bởi X đề thi và Y bài giảng"

+ Kết thúc

- Nếu không sử dụng:

+ Confirm dialog

+ Backend gọi: DELETE /api/admin/learning-outcomes/:id

+ Delete từ learning_outcomes

+ Toast: "Đã xóa chuẩn đầu ra"

Điều kiện sau:

- CĐR được quản lý đầy đủ

- Có thể gán cho tests và lessons

Các luồng thay thế:

Luồng 2a: Mã CĐR đã tồn tại

- Hiển thị: "Mã chuẩn đầu ra đã tồn tại"

- Quay lại form

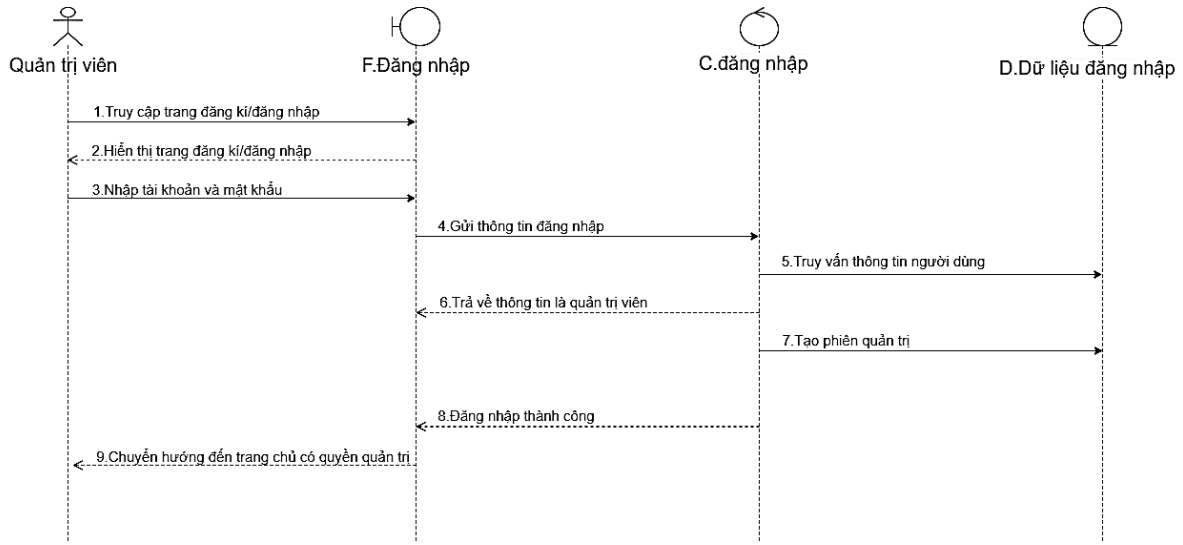
Luồng 4a: CĐR đang được sử dụng

- Hiển thị: "Không thể xóa chuẩn đầu ra này vì đang được sử dụng bởi X đề thi và Y bài giảng"

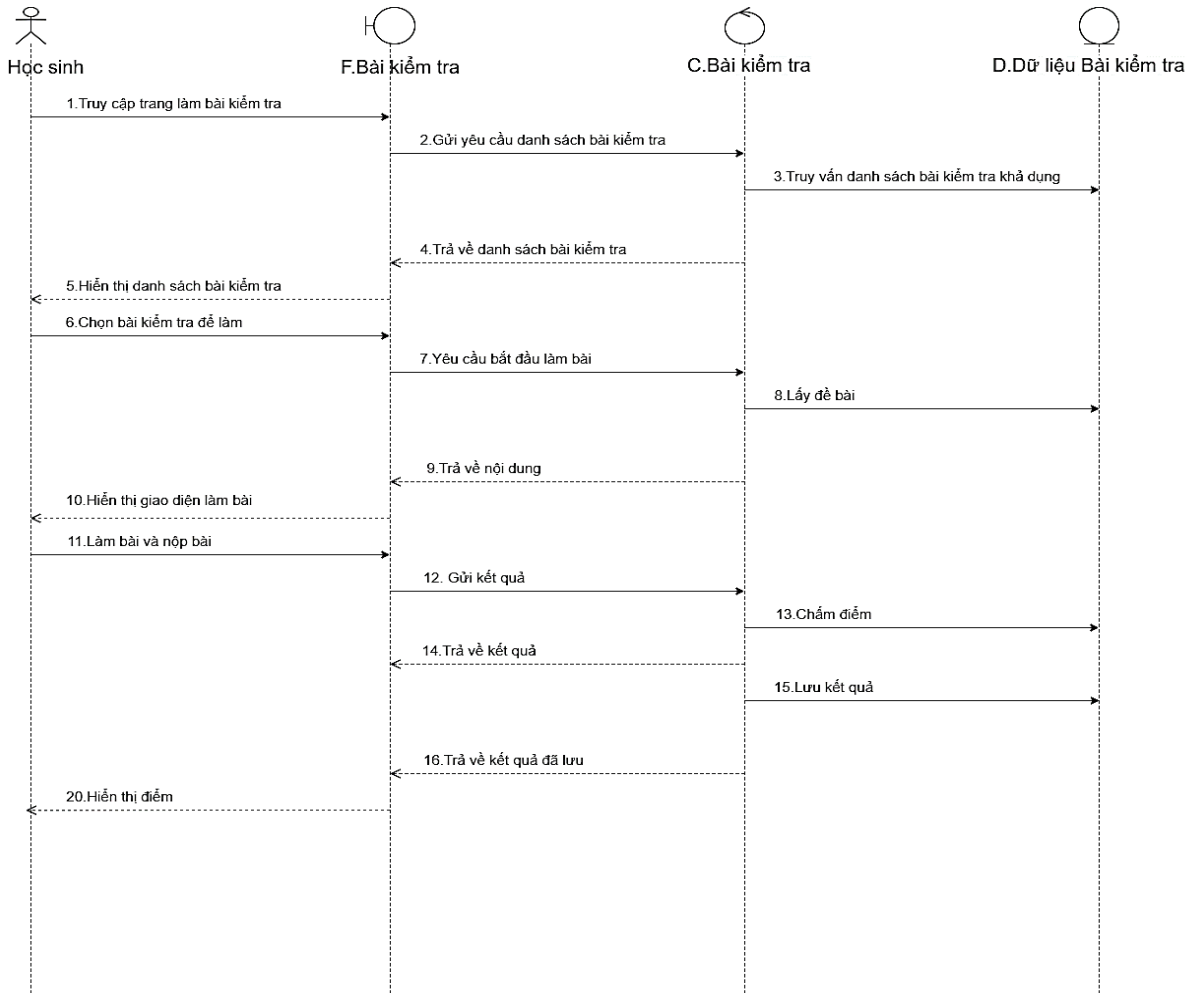
- Kết thúc

## 2.4. Biểu đồ tuần tự thực thi các Use Case


Biểu đồ tuần tự Use Case Học sinh đăng kí đăng nhập



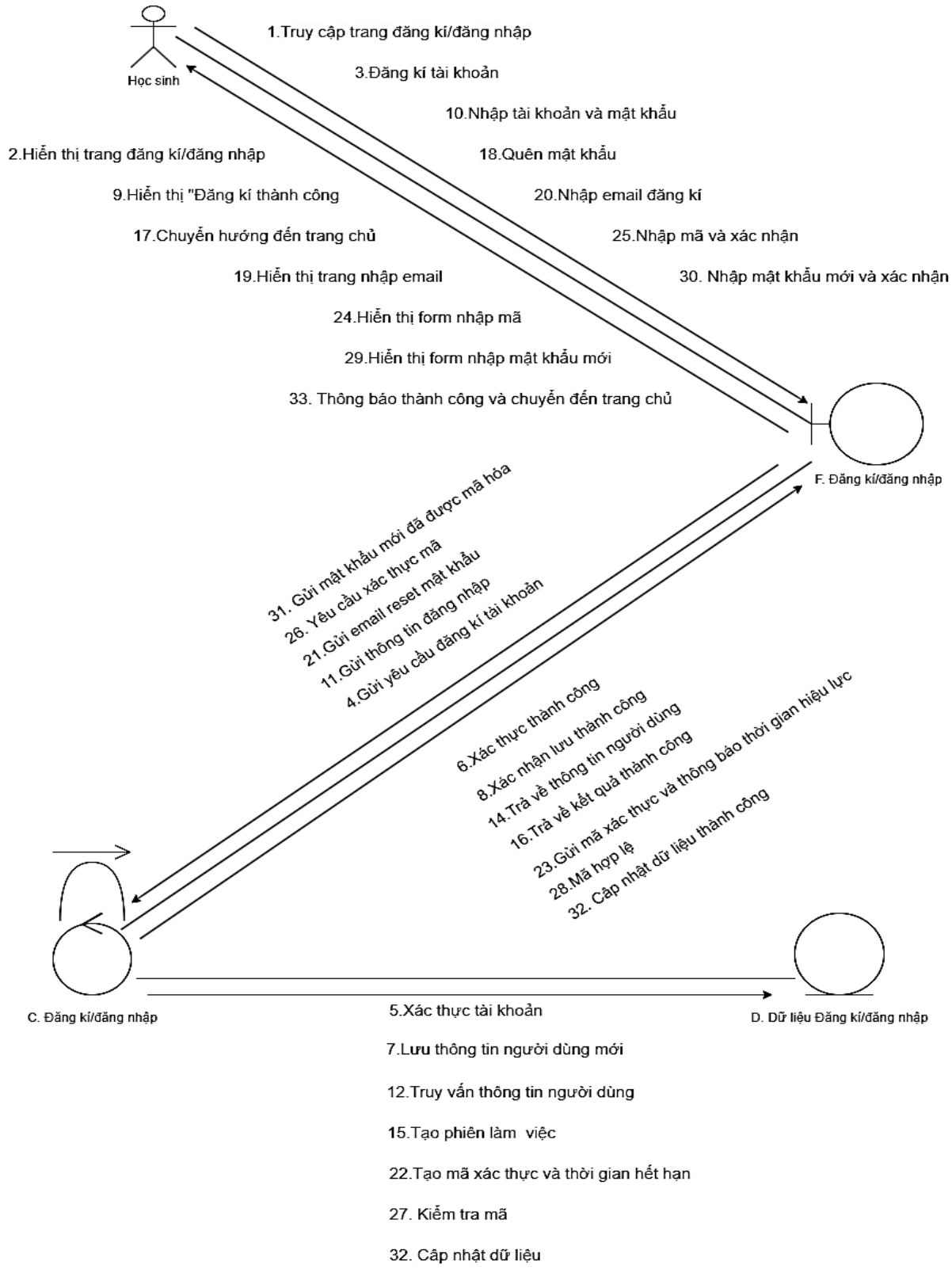

Biểu đồ tuần tự Use Case Quản trị viên đăng nhập



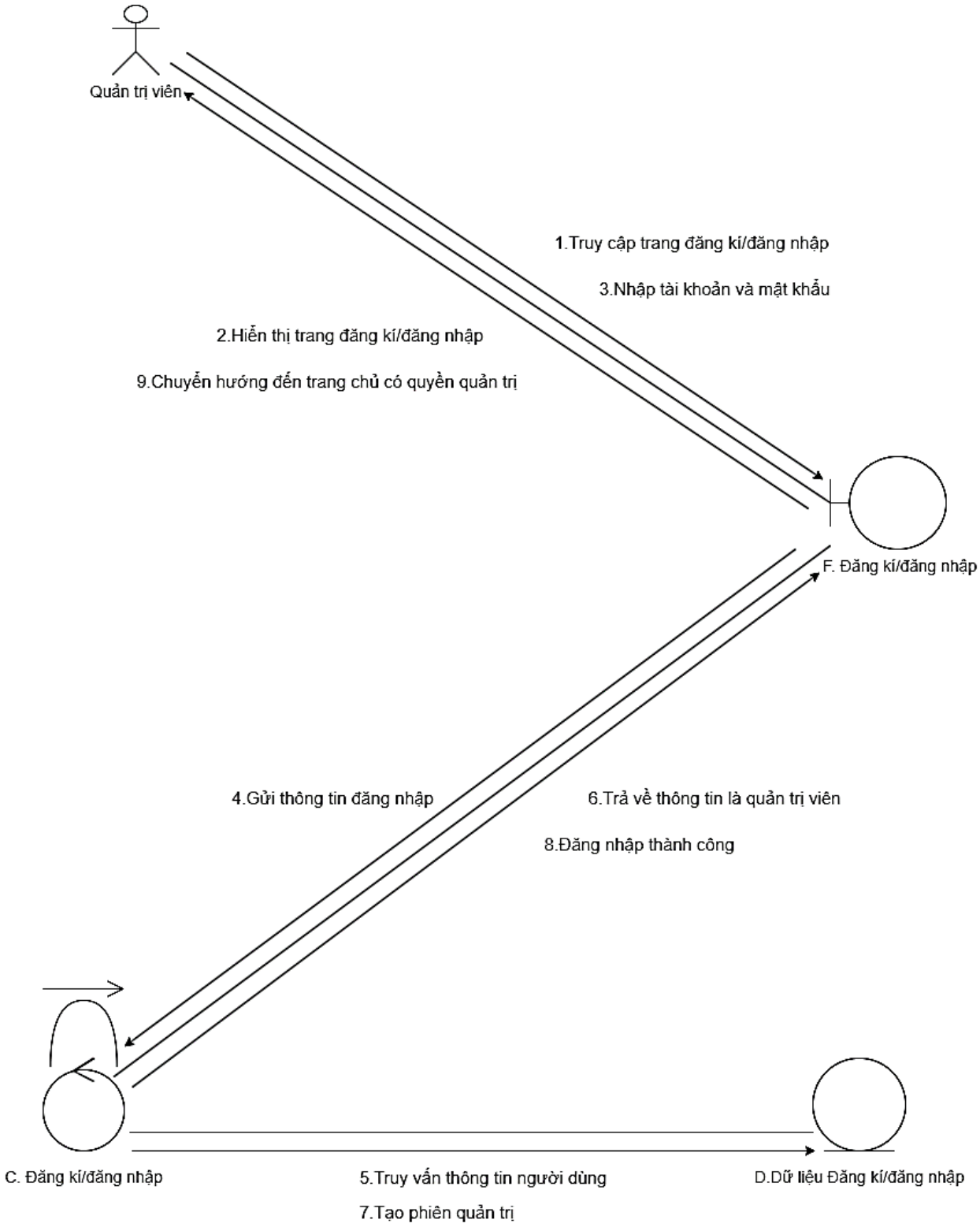

Biểu đồ tuần tự Use Case Học sinh làm bài kiểm tra


Biểu đồ tuần tự Use Case Nghiệp vụ của Quản trị viên

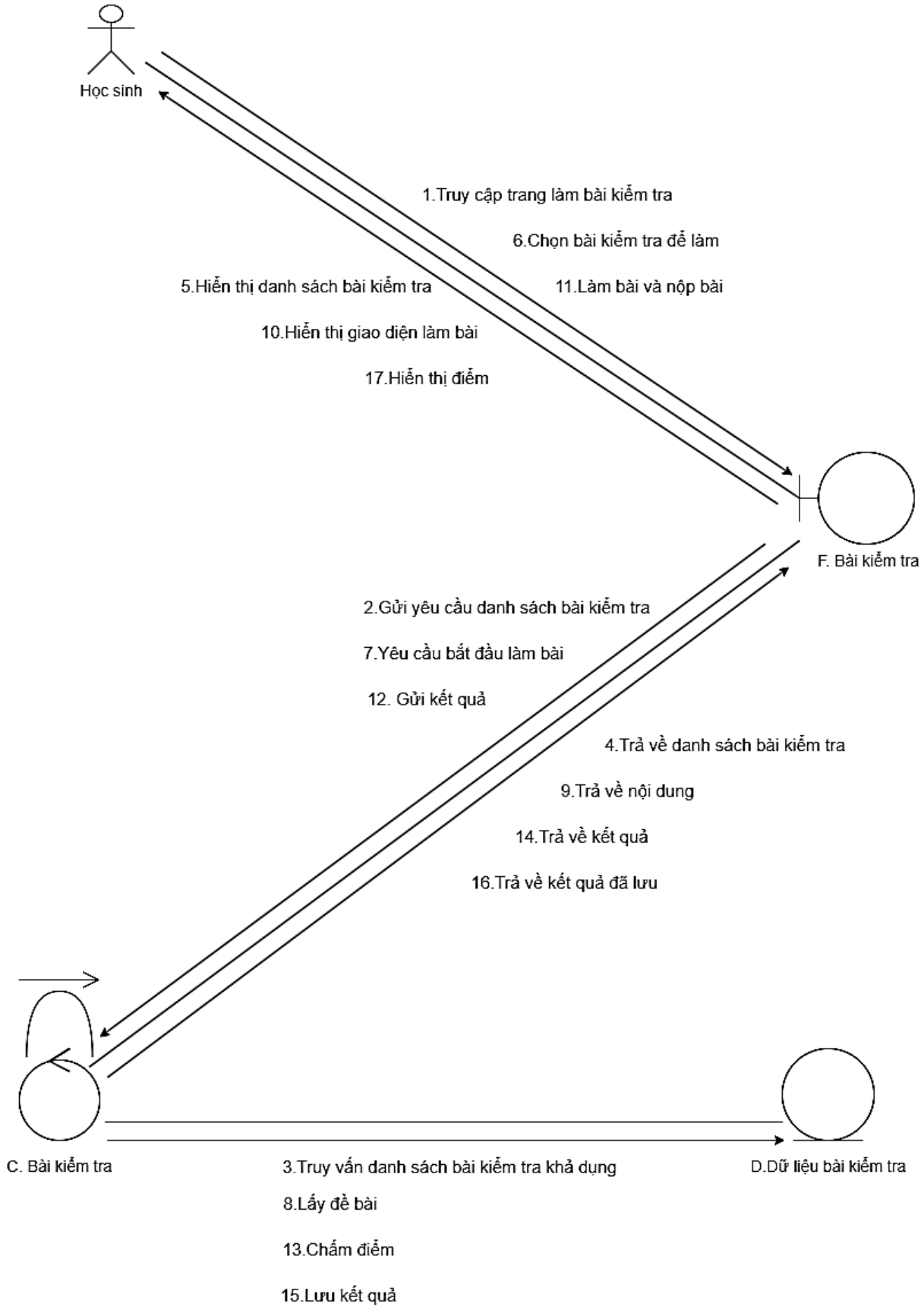
## 2.5. Biểu đồ cộng tác thực thi các Use Case



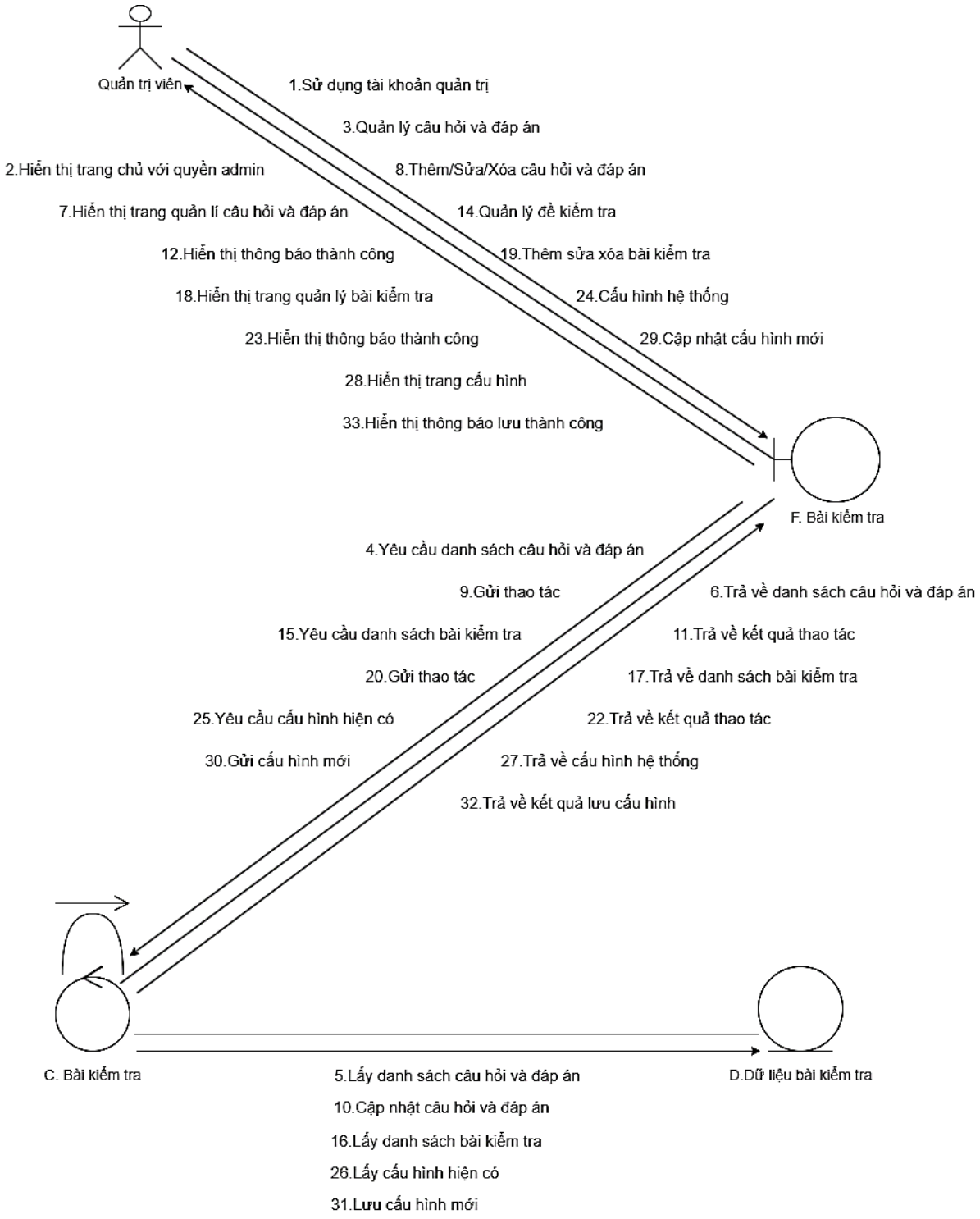

Biểu đồ cộng tác Use Case Người dùng đăng kí đăng nhập




Biểu đồ cộng tác Use Case Quản trị viên đăng kí đăng nhập




Biểu đồ cộng tác Use Case Học sinh làm bài kiểm tra

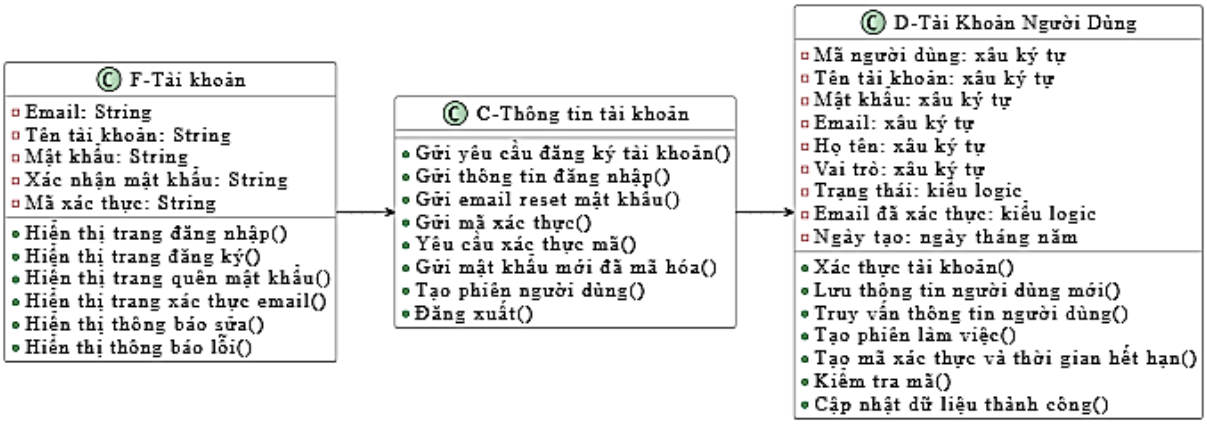



Biểu đồ cộng tác Use Case Nghiệp vụ Quản trị viên

# III. THIẾT KẾ HỆ THỐNG

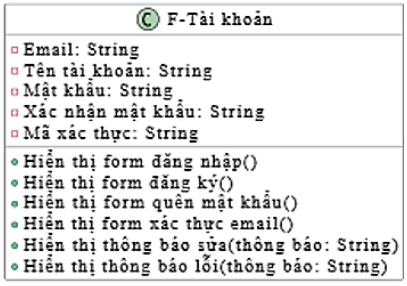
## 3.1. Biểu đồ lớp (Class Diagram) các UseCase

### Biểu đồ 1




Biểu đồ thiết kế Use Case "Tài khoản người dùng"

## Đặc tả biểu đồ 1




**1.Hiển thị form đăng nhập()**: Gửi yêu cầu hiển thị form đăng nhập đến lớp C-Đăng Ký Đăng Nhập

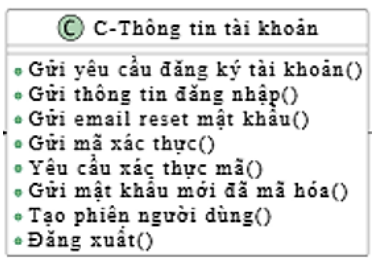
**2.Hiển thị form đăng ký()**: Gửi yêu cầu hiển thị form đăng ký đến lớp C-Đăng Ký Đăng Nhập

**3.Hiển thị form quên mật khẩu()**: Gửi yêu cầu hiển thị form quên mật khẩu đến lớp C-Đăng Ký Đăng Nhập

**4.Hiển thị form xác thực email()**: Gửi yêu cầu hiển thị form xác thực email đến lớp C-Đăng Ký Đăng Nhập

**5.Hiển thị thông báo sửa()**: Gửi thông báo thành công đến lớp C-Đăng Ký Đăng Nhập

**6.Hiển thị thông báo lỗi()**: Gửi thông báo lỗi đến lớp C-Đăng Ký Đăng Nhập




**7.Gửi yêu cầu đăng ký tài khoản()**: Yêu cầu tạo tài khoản mới đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**8.Gửi thông tin đăng nhập()**: Yêu cầu xác thực thông tin đăng nhập đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**9.Gửi email reset mật khẩu()**: Yêu cầu gửi email khôi phục mật khẩu đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**10.Gửi mã xác thực()**: Yêu cầu gửi mã xác thực qua email đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**11.Yêu cầu xác thực mã()**: Yêu cầu kiểm tra mã xác thực đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**12.Gửi mật khẩu mới đã mã hóa()**: Yêu cầu cập nhật mật khẩu mới đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**13.Tạo phiên người dùng()**: Yêu cầu tạo phiên làm việc đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**14.Đăng xuất()**: Yêu cầu kết thúc phiên làm việc đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập


**15.Xác thực tài khoản():** Kiểm tra thông tin đăng nhập trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**16.Lưu thông tin người dùng mới():** Thêm người dùng mới trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**17.Truy vấn thông tin người dùng():** Tìm kiếm thông tin người dùng trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

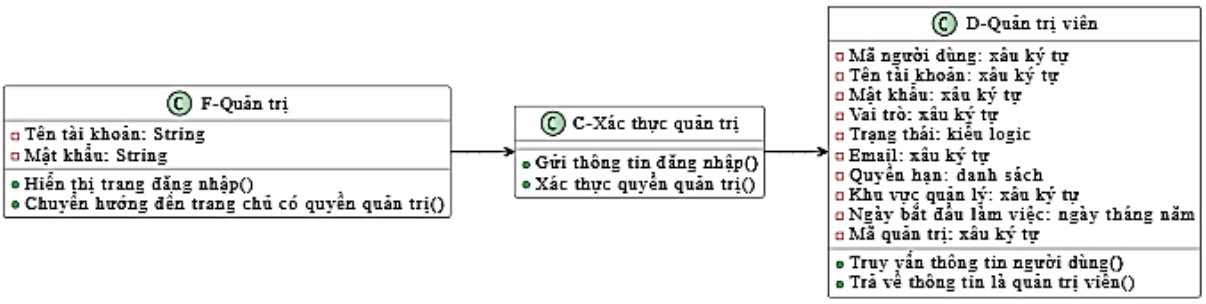
**18.Tạo phiên làm việc():** Tạo bản ghi phiên đăng nhập trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**19.Tạo mã xác thực và thời gian hết hạn():** Tạo mã xác thực có thời hạn trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

**20.Kiểm tra mã():** Xác thực mã xác thực trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

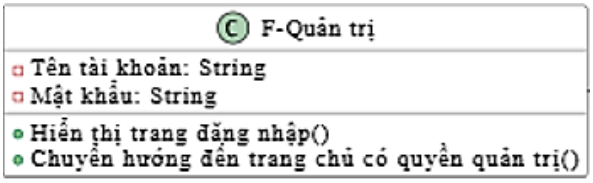
**21.Cập nhật dữ liệu thành công():** Cập nhật thông tin trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

## Biểu đồ 2



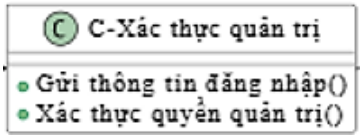

Biểu đồ thiết kế Use Case "Tài khoản quản trị"

## Đặc tả biểu đồ 2



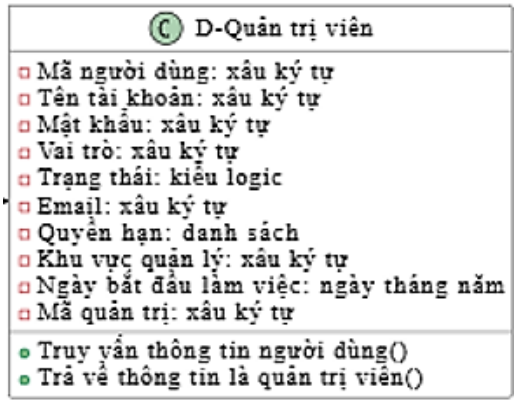

**1.Hiển thị trang đăng ký đăng nhập()**: Gửi yêu cầu hiển thị trang đăng nhập đến lớp C-Đăng Ký Đăng Nhập

**2.Chuyển hướng đến trang chủ có quyền quản trị()**: Gửi yêu cầu chuyển hướng sau đăng nhập thành công đến lớp C-Đăng Ký Đăng Nhập




**3.Gửi thông tin đăng nhập()**: Yêu cầu xác thực thông tin đăng nhập quản trị đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập

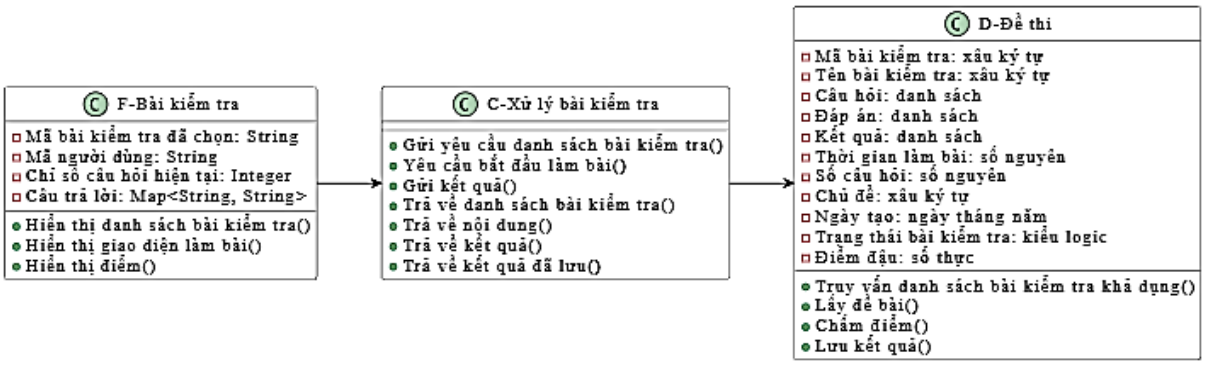
**4.Xác thực quyền quản trị ()**:Xác thực quyền quản trị đến lớp D-Dữ Liệu Đăng Ký Đăng Nhập




**5.Truy vấn thông tin người dùng():** Tìm thông tin user trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

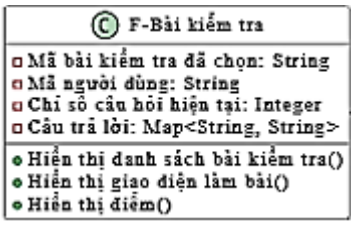
**6.Trả về thông tin là quản trị viên():** Xác nhận quyền quản trị trong lớp D-Dữ Liệu Đăng Ký Đăng Nhập

## Biểu đồ 3




Biểu đồ thiết kế Use Case "Làm bài kiểm tra"

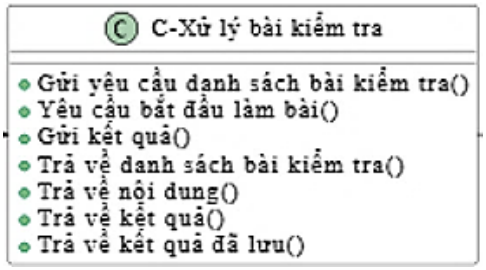
## Đặc tả biểu đồ 3




**1.Hiển thị danh sách bài kiểm tra()**: Gửi yêu cầu hiển thị danh sách bài kiểm tra đến lớp C-Bài Kiểm Tra

**2.Hiển thị giao diện làm bài()**: Gửi yêu cầu hiển thị giao diện làm bài đến lớp C-Bài Kiểm Tra

**3.Hiển thị điểm()**: Gửi yêu cầu hiển thị kết quả bài kiểm tra đến lớp C-Bài Kiểm Tra




**4.Gửi yêu cầu danh sách bài kiểm tra()**: Yêu cầu lấy danh sách bài kiểm tra đến lớp D-Dữ Liệu Bài Kiểm Tra

**5.Yêu cầu bắt đầu làm bài()**: Yêu cầu bắt đầu phiên làm bài đến lớp D-Dữ Liệu Bài Kiểm Tra

**6.Gửi kết quả()**: Yêu cầu nộp bài kiểm tra đến lớp D-Dữ Liệu Bài Kiểm Tra

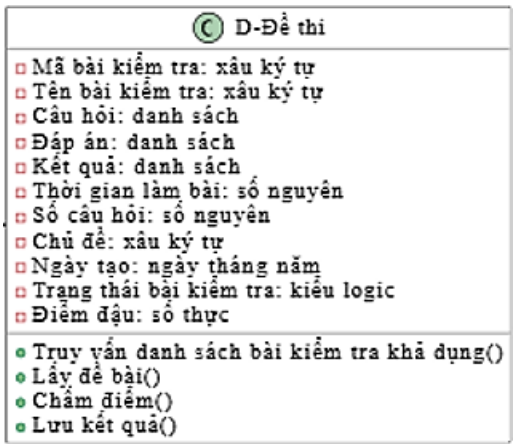
**7.Trả về danh sách bài kiểm tra()**: Yêu cầu trả về danh sách bài kiểm tra đến lớp D-Dữ Liệu Bài Kiểm Tra

**8.Trả về nội dung()**: Yêu cầu trả về nội dung bài kiểm tra đến lớp D-Dữ Liệu Bài Kiểm Tra

**9.Trả về kết quả()**: Yêu cầu trả về kết quả chấm bài đến lớp D-Dữ Liệu Bài Kiểm Tra

**10.Trả về kết quả đã lưu()**: Yêu cầu xác nhận kết quả đã được lưu đến lớp D-Dữ Liệu Bài Kiểm Tra

**11.Truy vấn danh sách bài kiểm tra khả dụng():** Lấy bài kiểm tra đang hoạt động trong lớp D-Dữ Liệu Bài Kiểm Tra

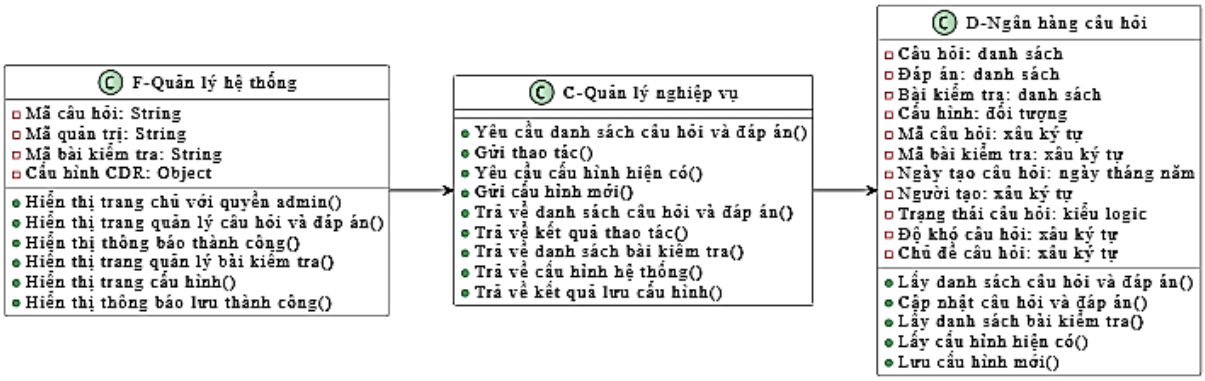



**12.Lấy đề bài():** Lấy nội dung bài kiểm tra trong lớp D-Dữ Liệu Bài Kiểm Tra

**13.Chấm điểm():** Tính điểm bài kiểm tra trong lớp D-Dữ Liệu Bài Kiểm Tra

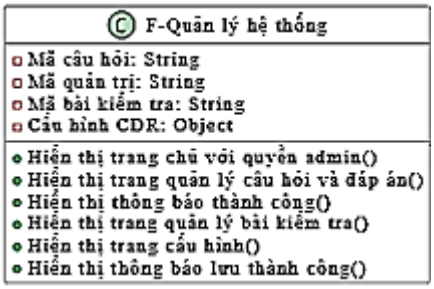
**14.Lưu kết quả():** Lưu kết quả trong lớp D-Dữ Liệu Bài Kiểm Tra

## Biểu đồ 4




Biểu đồ thiết kế Use Case "Nghiệp vụ quản trị"

## Đặc tả biểu đồ 4




**1.Hiển thị trang chủ với quyền admin()**: Gửi yêu cầu hiển thị dashboard quản trị đến lớp C-Bài Kiểm Tra

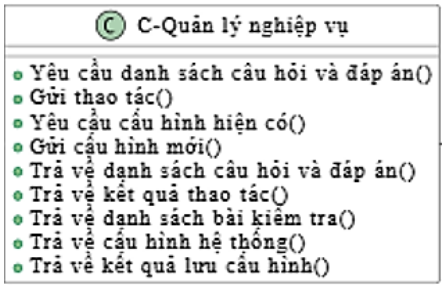
**2.Hiển thị trang quản lý câu hỏi và đáp án()**: Gửi yêu cầu hiển thị quản lý câu hỏi đến lớp C-Bài Kiểm Tra

**3.Hiển thị thông báo thành công()**: Gửi thông báo thao tác thành công đến lớp C-Bài Kiểm Tra

**4.Hiển thị trang quản lý bài kiểm tra()**: Gửi yêu cầu hiển thị quản lý bài kiểm tra đến lớp C-Bài Kiểm Tra

**5.Hiển thị trang cấu hình()**: Gửi yêu cầu hiển thị cấu hình hệ thống đến lớp C-Bài Kiểm Tra

**6.Hiển thị thông báo lưu thành công()**: Gửi xác nhận lưu cấu hình thành công đến lớp C-Bài Kiểm Tra




**7.Yêu cầu danh sách câu hỏi và đáp án()**: Yêu cầu lấy danh sách câu hỏi và đáp án đến lớp D-Dữ Liệu Bài Kiểm Tra

**8.Gửi thao tác()**: Yêu cầu thực hiện thao tác (thêm, sửa, xóa) đến lớp D-Dữ Liệu Bài Kiểm Tra

**9.Yêu cầu cấu hình hiện có()**: Yêu cầu lấy cấu hình hiện tại đến lớp D-Dữ Liệu Bài Kiểm Tra

**10.Gửi cấu hình mới()**: Yêu cầu cập nhật cấu hình mới đến lớp D-Dữ Liệu Bài Kiểm Tra

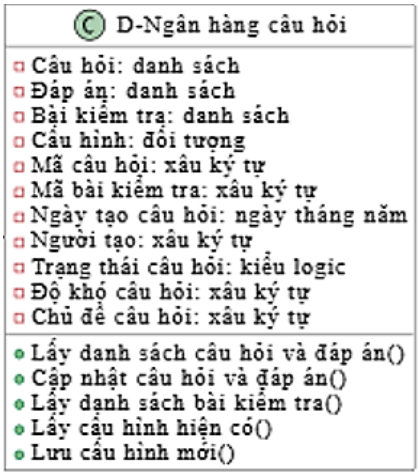
**11.Trả về danh sách câu hỏi và đáp án()**: Yêu cầu trả về danh sách câu hỏi đến lớp D-Dữ Liệu Bài Kiểm Tra

**12.Trả về kết quả thao tác()**: Yêu cầu trả về kết quả thao tác đến lớp D-Dữ Liệu Bài Kiểm Tra

**13.Trả về danh sách bài kiểm tra()**: Yêu cầu trả về danh sách bài kiểm tra đến lớp D-Dữ Liệu Bài Kiểm Tra

**14.Trả về cấu hình hệ thống()**: Yêu cầu trả về cấu hình hệ thống đến lớp D-Dữ Liệu Bài Kiểm Tra

**15.Trả về kết quả lưu cấu hình()**: Yêu cầu xác nhận lưu cấu hình đến lớp D-Dữ Liệu Bài Kiểm Tra




**16.Lấy danh sách câu hỏi và đáp án()**: Truy vấn câu hỏi và đáp án trong lớp D-Dữ Liệu Bài Kiểm Tra

**17.Cập nhật câu hỏi và đáp án()**: Cập nhật thông tin câu hỏi trong lớp D-Dữ Liệu Bài Kiểm Tra

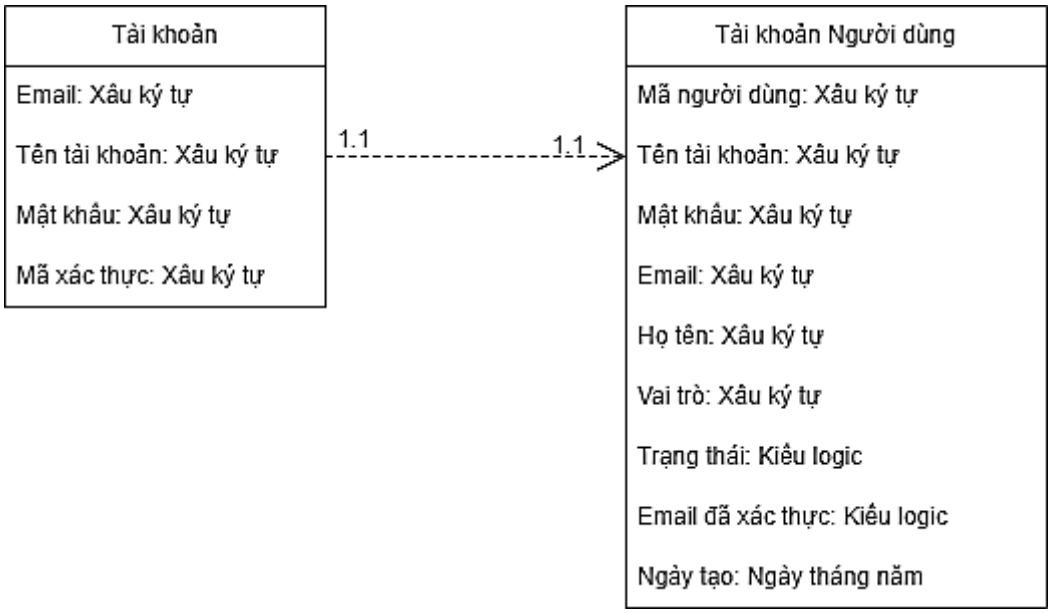
**18.Lấy danh sách bài kiểm tra()**: Truy vấn bài kiểm tra trong lớp D-Dữ Liệu Bài Kiểm Tra

**19.Lấy cấu hình hiện có()**: Lấy cấu hình hiện tại trong lớp D-Dữ Liệu Bài Kiểm Tra

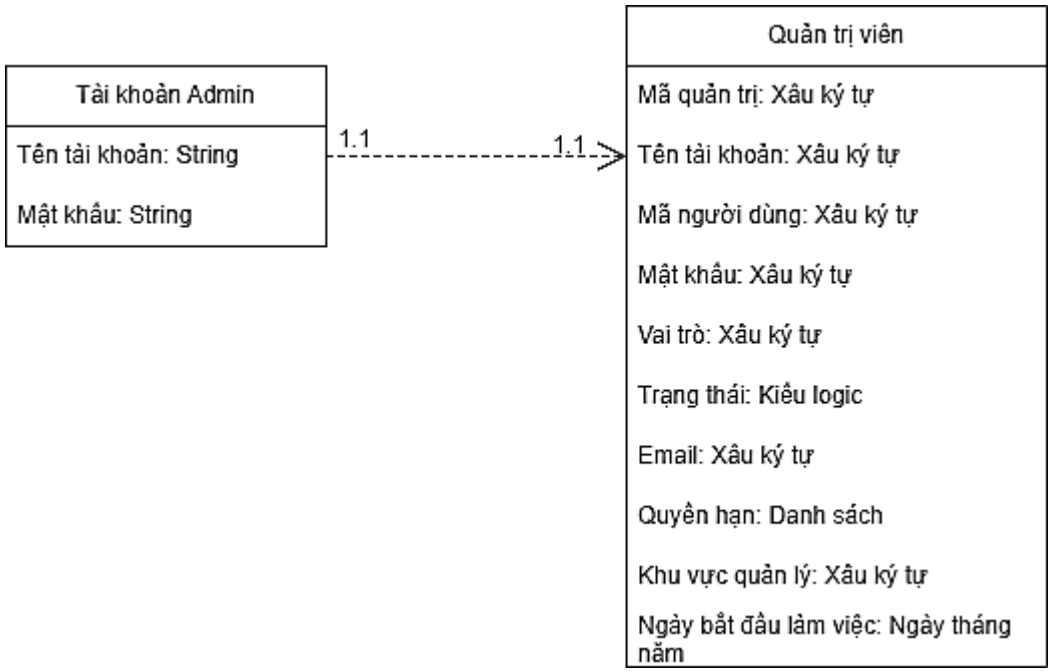
**20.Lưu cấu hình mới()**: Lưu cấu hình mới trong lớp D-Dữ Liệu Bài Kiểm Tra

## 3.2. Thiết kế cơ sở dữ liệu

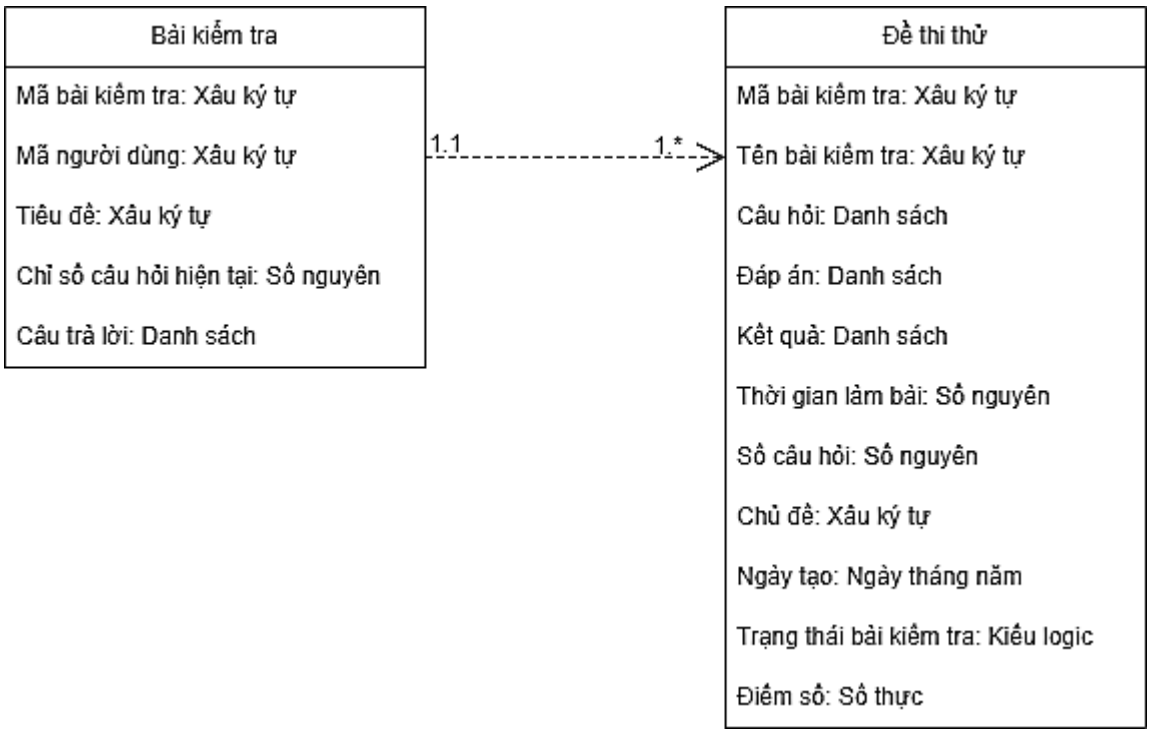
*a) Biểu đồ lớp thiết kế UML*



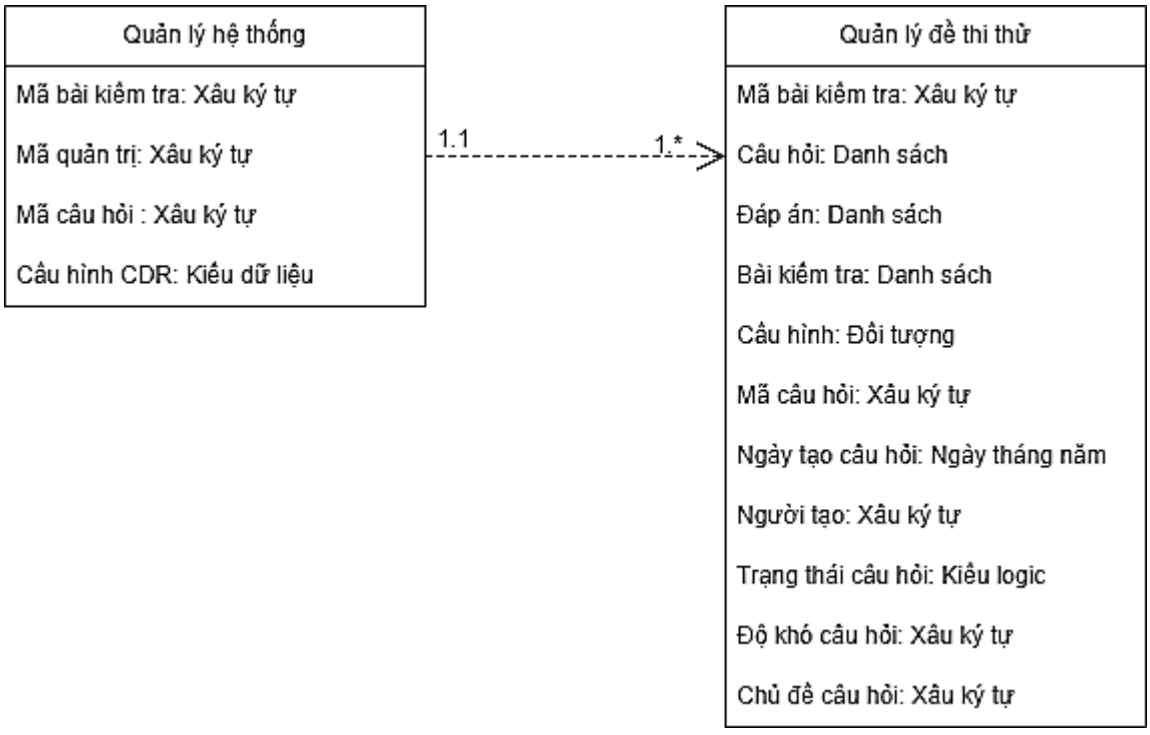

Lớp thiết kế UML "Tài khoản người dùng"



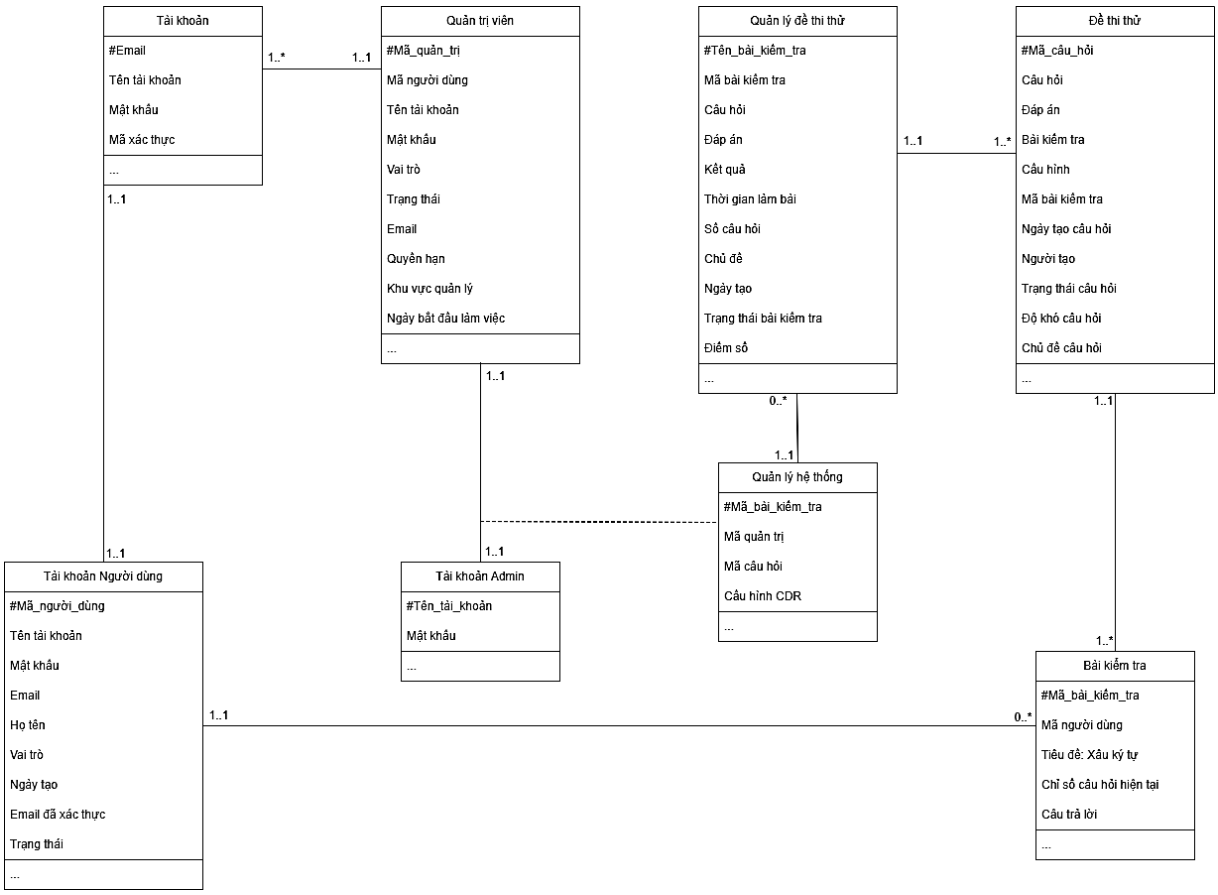

Lớp thiết kế UML "Tài khoản quản trị"




Lớp thiết kế UML "Làm bài kiểm tra"

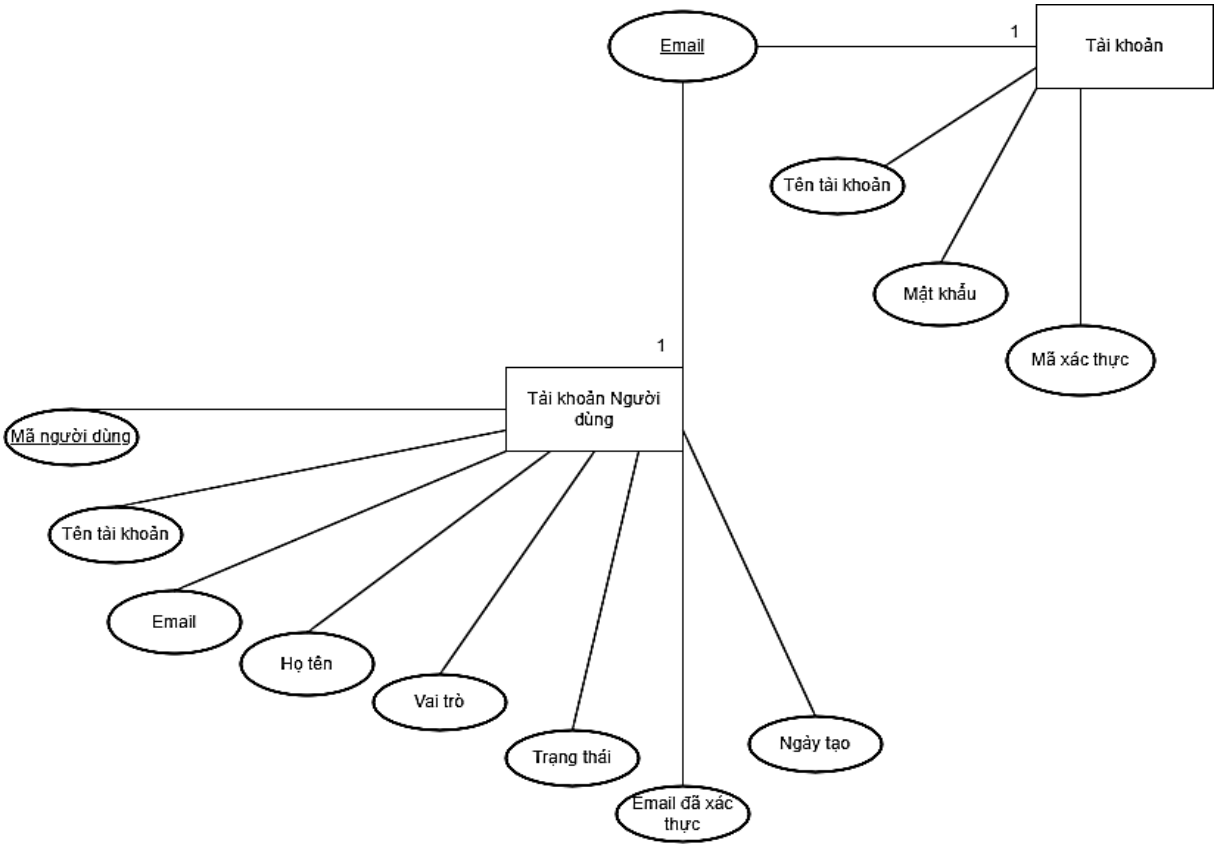



Lớp thiết kế UML "Nghiệp vụ quản trị

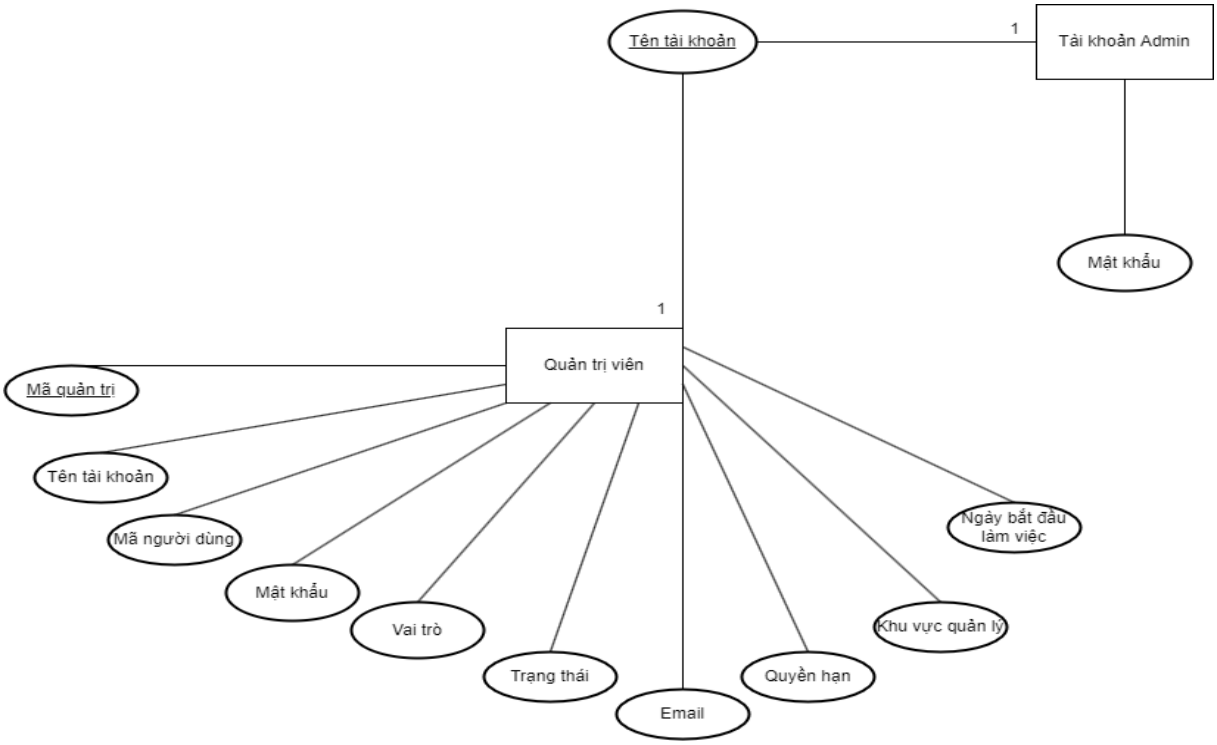



Biểu đồ lớp thiết kế UML tổng thể

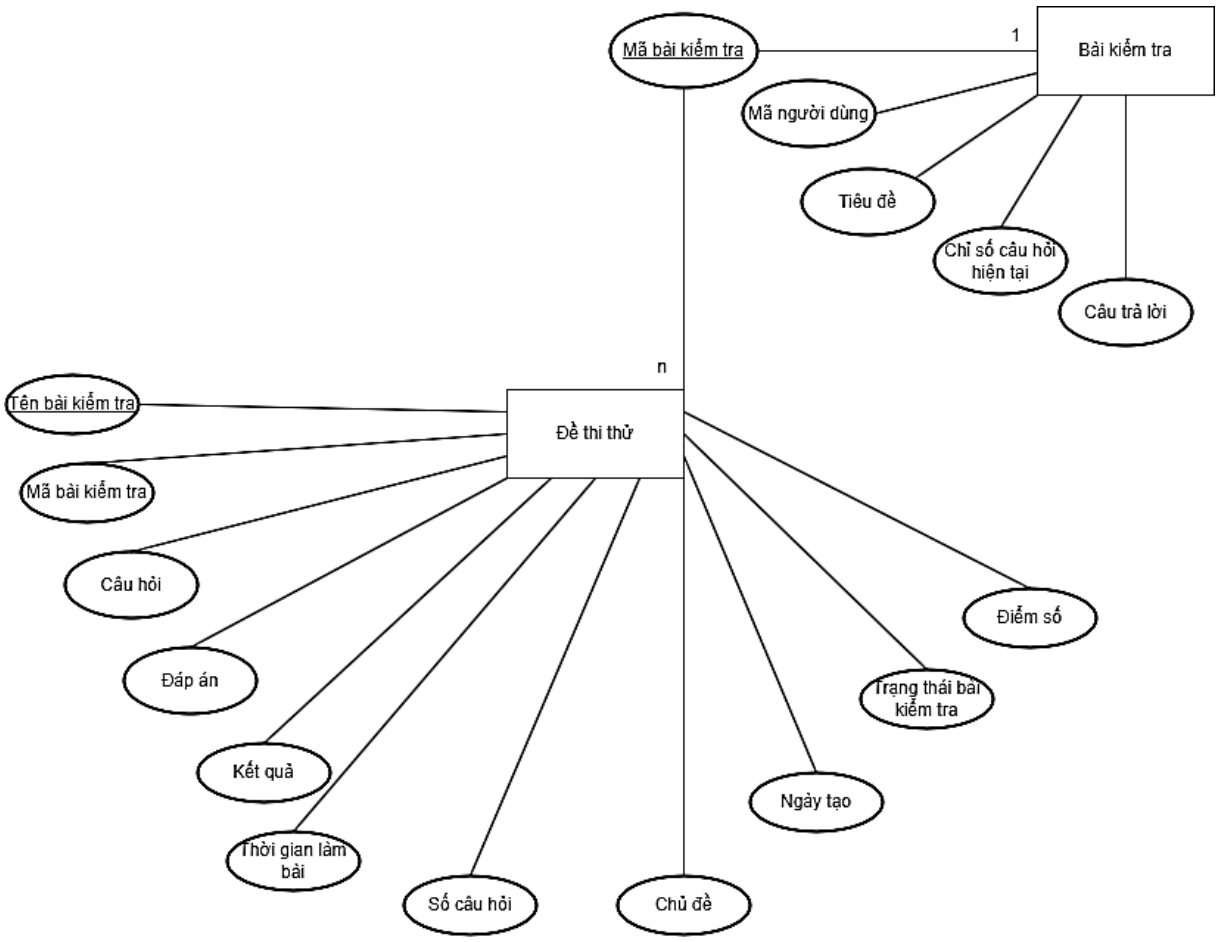
*b) Chuyển đổi biểu đồ lớp thiết kế UML sang mô hình EER*



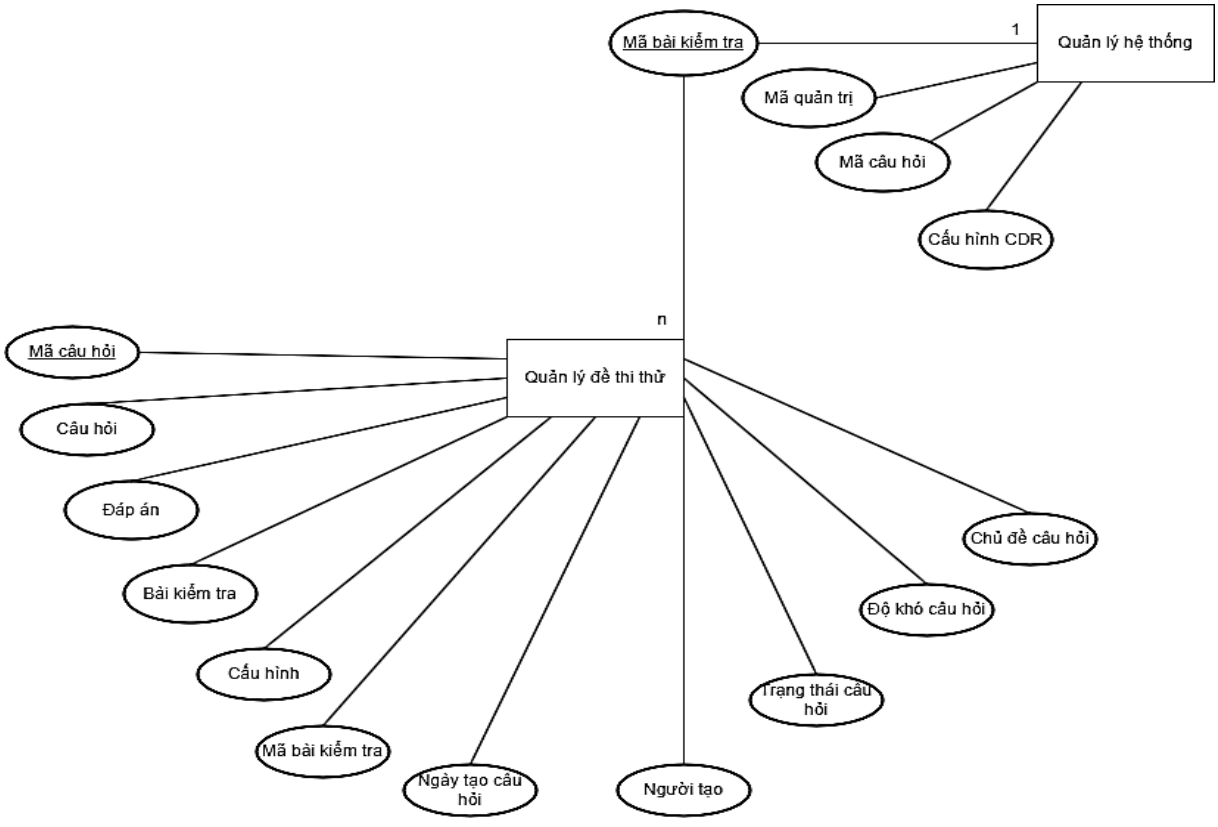

Chuyển đổi UML "Tài khoản người dùng" sang EER



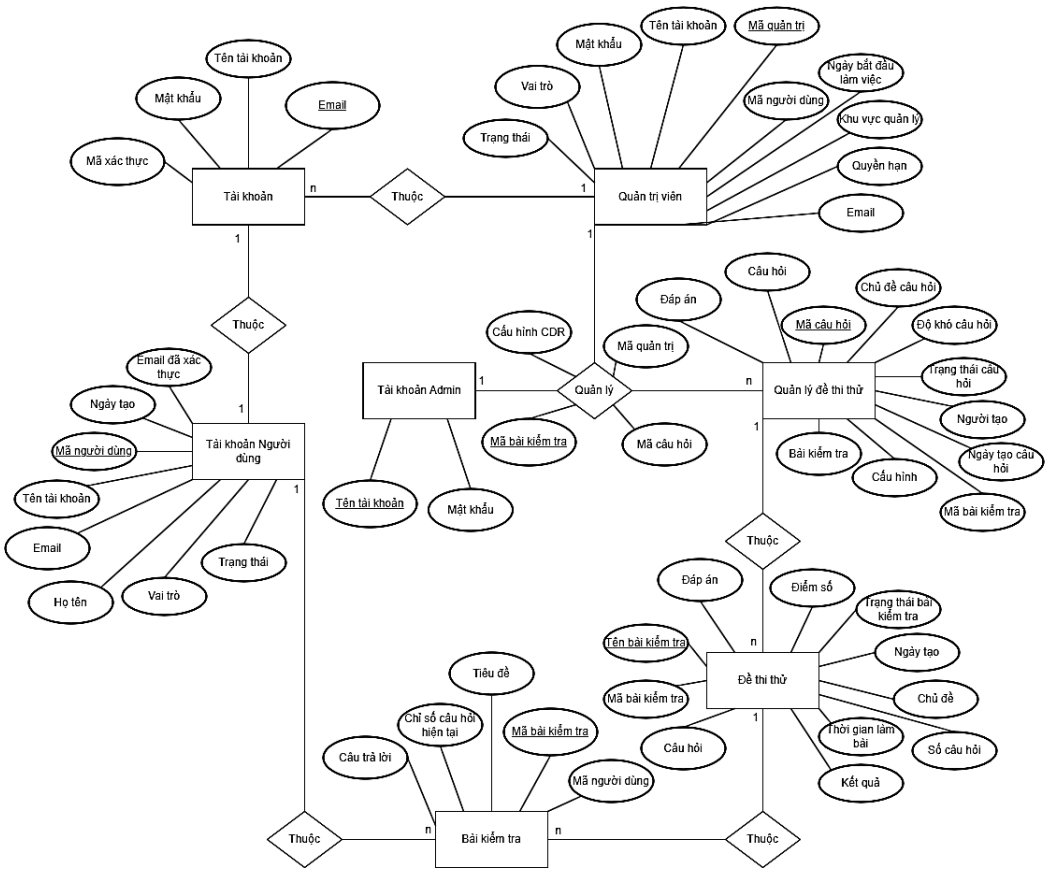

Chuyển đổi UML "Tài khoản quản trị" sang EER




Chuyển đổi UML "Làm bài kiểm tra" sang EER




Chuyển đổi UML "Nghiệp vụ quản trị" sang EER




Mô hình EER chuyển đổi từ biểu đồ lớp thiết kế UML tổng thể

## *c) Chuyển đổi mô hình EER sang mô hình quan hệ*

Tài khoản người dùng:

| Tài khoản | | | |
|---|---|---|---|
| **Email** | Tên tài khoản | Mật khẩu | Mã xác thực |

| Tài khoản Người dùng | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Mã người dùng** | Tên tài khoản | Email | Họ tên | Vai trò | Trạng thái | Email đã xác thực | Ngày tạo |

Tài khoản quản trị:

| Tài khoản Admin | |
|---|---|
| **Tên tài khoản** | Mật khẩu |

| Quản trị viên | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Mã quản trị** | Tên tài khoản | Mã người dùng | Mật khẩu | Vai trò | Trạng thái | Email | Quyền hạn | Khu vực quản lý | Ngày bắt đầu làm việc |

Làm bài kiểm tra:

| Bài kiểm tra | | | | |
|---|---|---|---|---|
| **Mã bài kiểm tra** | Mã người dùng | Tiêu đề | Chỉ số câu hỏi hiện tại | Câu trả lời |

| Đề thi thử | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Tên bài kiểm tra** | Mã bài kiểm tra | Câu hỏi | Đáp án | Kết quả | Thời gian làm bài | Số câu hỏi | Chủ đề | Ngày tạo | Trạng thái bài kiểm tra | Điểm số |

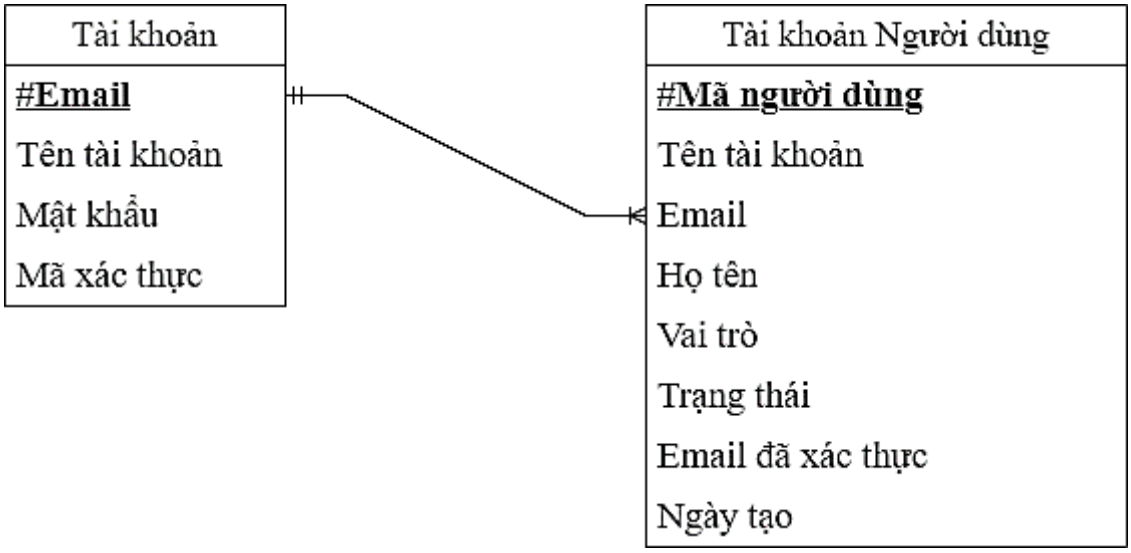
Nghiệp vụ quản trị:

| Quản lý hệ thống | | | |
|---|---|---|---|
| **Mã bài kiểm tra** | Mã quản trị | Mã câu hỏi | Cấu hình CDR |

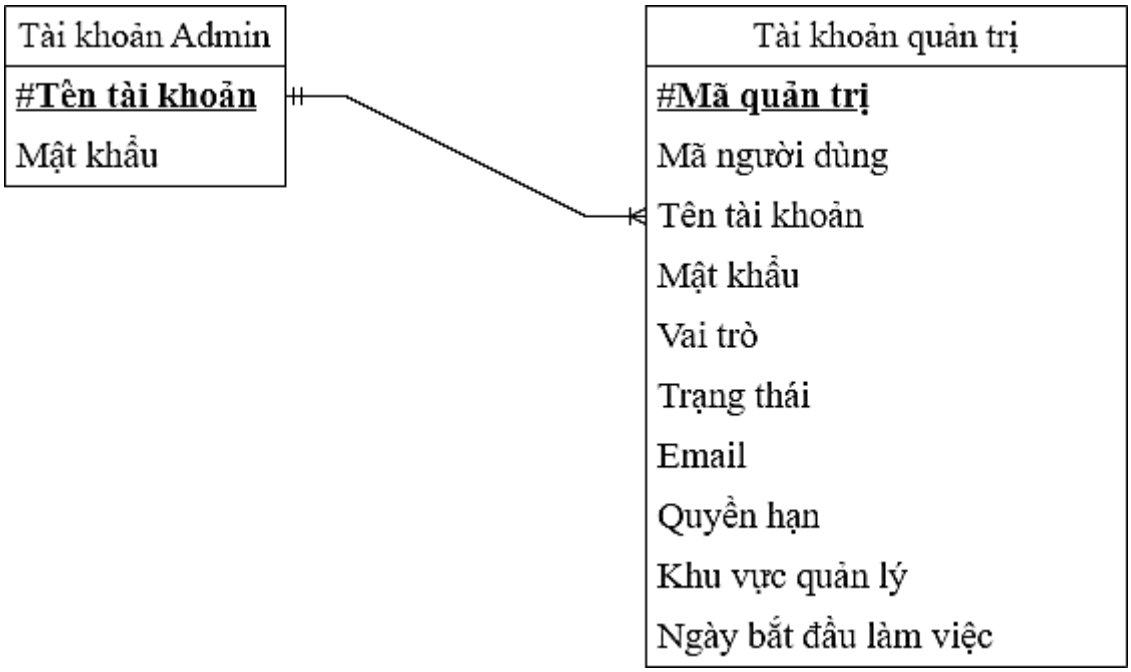
| Quản lý đề thi thử | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Mã câu hỏi** | Câu hỏi | Đáp án | Bài kiểm tra | Cấu hình | Mã bài kiểm tra | Ngày tạo câu hỏi | Người tạo | Trạng thái câu hỏi | Độ khó câu hỏi | Chủ đề câu hỏi |

*d) Mô tả các bảng quan hệ*

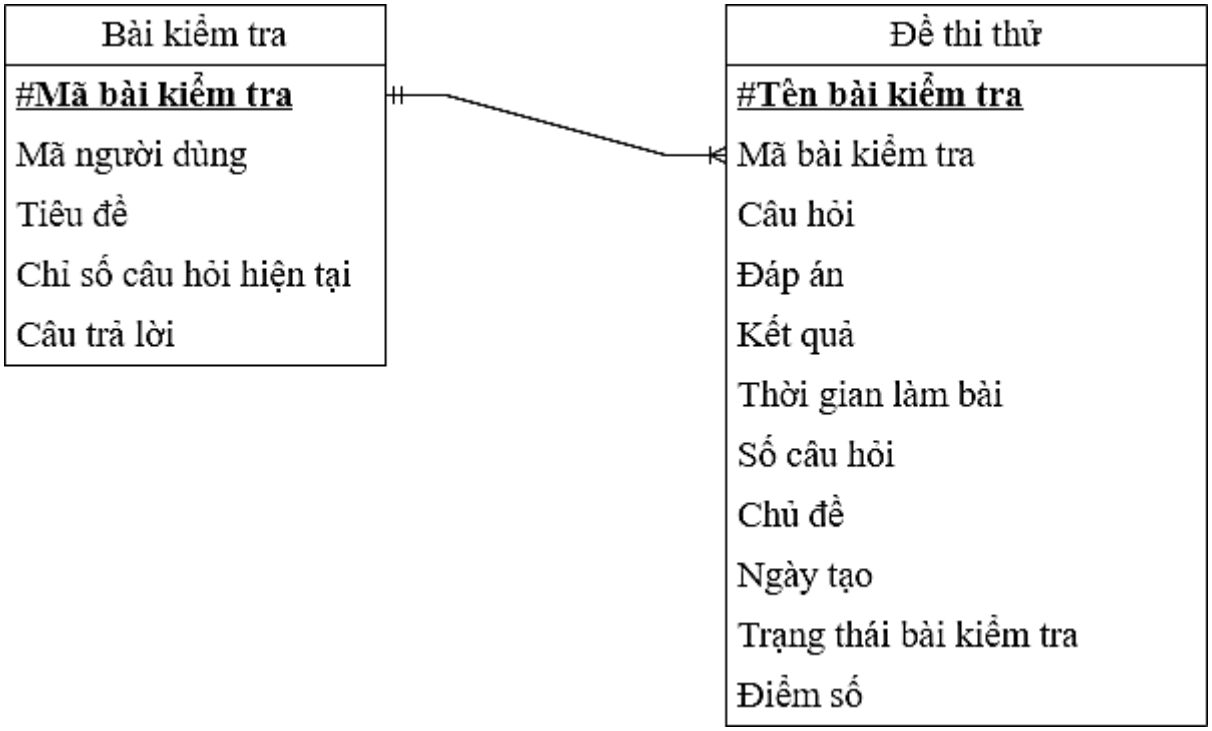
Tài khoản người dùng:



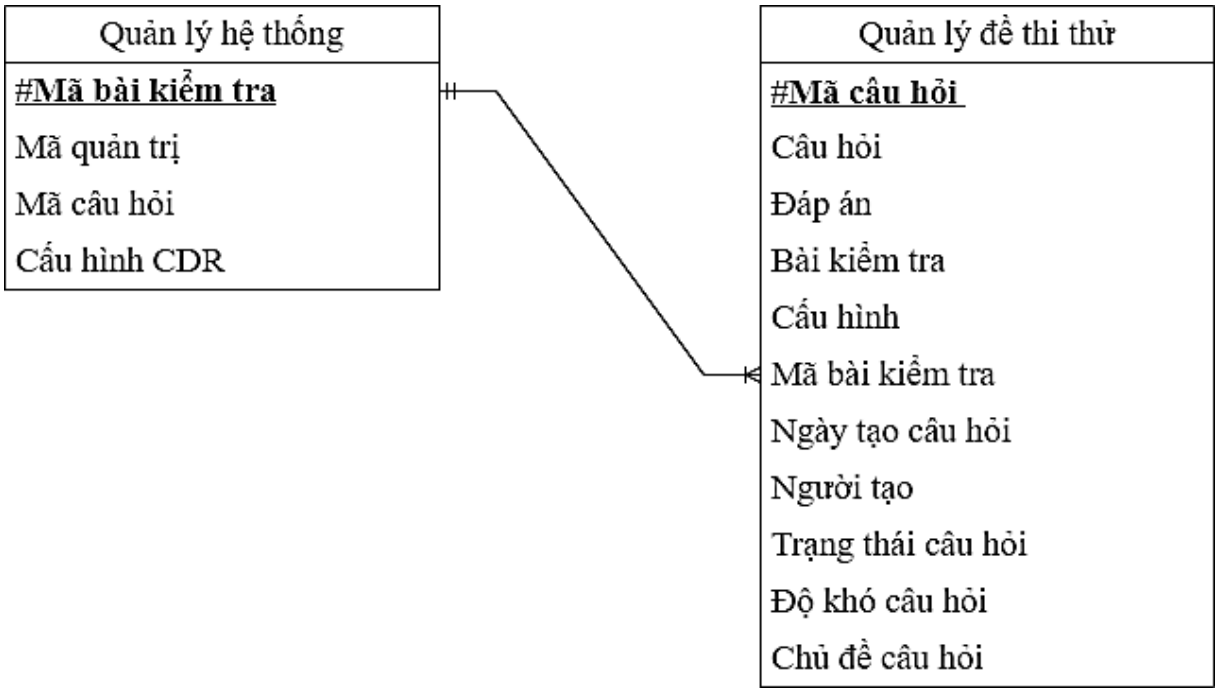

Tài khoản quản trị:

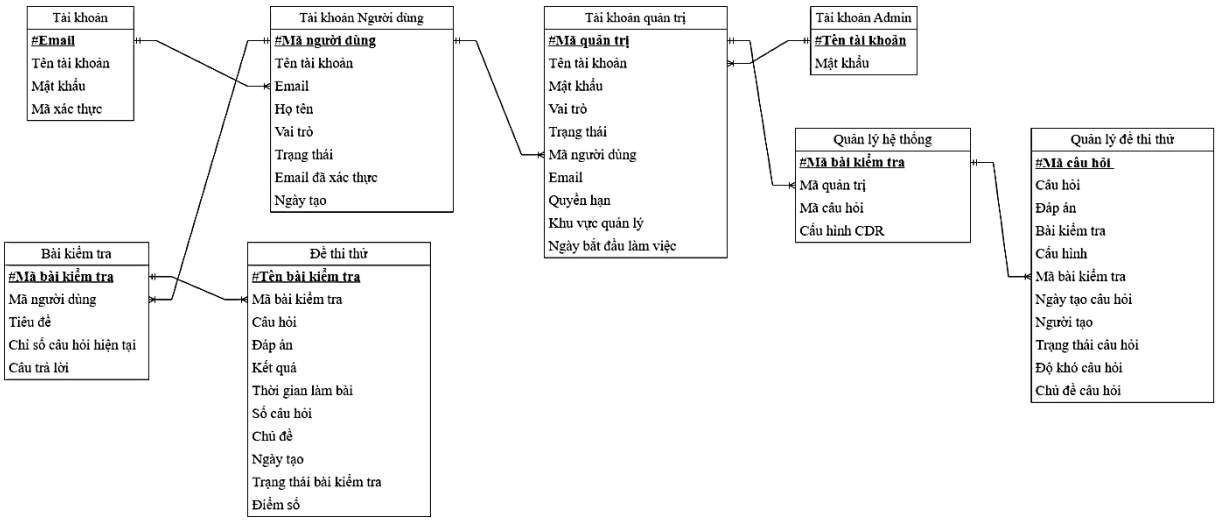


Làm bài kiểm tra:




Nghiệp vụ quản trị:






Biểu đồ mô hình quan hệ

# Chương 3 : Cơ sở lý thuyết

## I. Công cụ sử dụng

### 1.1.Visual Studio Code (Phần mềm phát triển)

Visual Studio Code là trình soạn thảo mã nguồn miễn phí, mã nguồn mở được phát triển bởi Microsoft. Ra mắt năm 2015, VS Code nhanh chóng trở thành công cụ phát triển phổ biến nhất nhờ tính năng mạnh mẽ, nhẹ, và hỗ trợ đa nền tảng (Windows, macOS, Linux).Tính năng nổi bật:

+ **IntelliSense:** Gợi ý mã thông minh, tự động hoàn thành code
+ **Debugging tích hợp:** Hỗ trợ debug trực tiếp trong editor
+ **Git tích hợp:** Quản lý version control ngay trong VS Code
+ **Extensions phong phú:** Hàng nghìn extension mở rộng chức năng
+ **Terminal tích hợp:** Chạy lệnh terminal ngay trong editor

### 1.2. Phát triển giao diện người dùng (Frontend Development)

HTML:

- HTML (HyperText Markup Language) là ngôn ngữ đánh dấu tiêu chuẩn để tạo cấu trúc cho các trang web. Nó không phải là ngôn ngữ lập trình mà là cách mô tả nội dung như văn bản, hình ảnh, liên kết để trình duyệt web hiển thị.

- HTML được Tim Berners-Lee tạo ra năm 1991 tại CERN như nền tảng để xây dựng các trang web. Phiên bản HTML5 từ năm 2014 đã mở rộng với hỗ trợ multimedia, canvas cho vẽ đồ họa, cùng các API như Geolocation để định vị. HTML sử dụng các thẻ để định nghĩa cấu trúc tài liệu, kèm thuộc tính để thêm chi tiết. Khi chạy, trình duyệt sẽ phân tích HTML thành DOM – một cây cấu trúc giúp hiển thị và tương tác với nội dung.

- Cách hoạt động: Trình duyệt phân tích (parse) HTML thành DOM (Document Object Model), một cây cấu trúc để hiển thị và tương tác.

- Ưu điểm: Dễ học, tương thích cao, và là nền tảng cho web. HTML kết hợp với CSS và JavaScript để tạo giao diện người dùng thân thiện và responsive trên mọi thiết bị.

CSS:

- CSS (Cascading Style Sheets) là ngôn ngữ định kiểu cho tài liệu HTML, kiểm soát cách hiển thị như màu sắc, font chữ, khoảng cách, bố cục.

- CSS ra đời năm 1996 nhờ ý tưởng của Håkon Wium Lie tại W3C, nhằm tách biệt phần thiết kế khỏi cấu trúc HTML. Phiên bản CSS3 từ năm 2011 đã thêm animation cùng responsive design để thích ứng với màn hình di động. CSS hoạt động theo nguyên tắc "cascading" (xếp chồng), nghĩa là các quy tắc kiểu được áp dụng theo thứ tự ưu tiên (inline, internal, external).
- Cách hoạt động: Trình duyệt áp dụng CSS lên DOM để render giao diện, hỗ trợ media queries cho responsive design trên thiết bị di động.
- Ưu điểm: Tách biệt cấu trúc (HTML) và kiểu dáng (CSS), dễ bảo trì. CSS giúp tạo giao diện responsive, đẹp mắt và chuyên nghiệp. Việc bảo trì code dễ dàng hơn khi chỉ cần chỉnh file CSS riêng mà không động đến HTML.

JavaScript - Ngôn ngữ lập trình phía client:

- JavaScript (JS) là ngôn ngữ lập trình động, được sử dụng chủ yếu ở phía client (trình duyệt) để tạo tương tác cho web.
- JavaScript được Brendan Eich phát triển năm 1995 tại Netscape để thêm tính tương tác cho web. Chuẩn ECMAScript 6 (ES6) từ năm 2015 hỗ trợ các tính năng hiện đại như arrow functions, classes, async/await. JavaScript là ngôn ngữ động và event-driven, thao tác trực tiếp với DOM để thay đổi nội dung thời gian thực, xử lý sự kiện người dùng, và gọi API.
- Cách hoạt động: Engine như V8 (Chrome) thực thi JavaScript trong trình duyệt, xử lý bất đồng bộ qua callback, Promise, async/await.
- Ưu điểm: Tích hợp trực tiếp với HTML/CSS, đa nền tảng. JavaScript tạo form validation, animation, gọi API, giúp trang web trở nên sống động và phản hồi nhanh mà không cần tải lại toàn bộ.

React

- React là thư viện JavaScript mã nguồn mở được phát triển bởi Facebook (nay là Meta) năm 2013 để xây dựng giao diện người dùng cho các ứng dụng web. React tập trung vào việc xây dựng UI component-based, cho phép tái sử dụng code hiệu quả.

- React sử dụng Virtual DOM - một bản sao nhẹ của DOM thật trong bộ nhớ. Khi state thay đổi, React so sánh Virtual DOM mới với cũ (diffing algorithm), chỉ cập nhật những phần thay đổi thật sự trên DOM, giúp tối ưu hiệu suất đáng kể. React áp dụng kiến trúc component-based, mỗi component là một khối độc lập có state và props riêng.

- Các khái niệm cốt lõi:

+ **Components**: Khối xây dựng UI, có thể là Class hoặc Function component
+ **Props**: Dữ liệu truyền từ component cha xuống con (read-only)
+ **State**: Dữ liệu nội bộ của component, khi thay đổi sẽ re-render
+ **Hooks**: useState, useEffect, useContext cho Function components
+ **Virtual DOM**: Tối ưu hiệu suất bằng cách chỉ cập nhật phần thay đổi

- Ưu điểm: Hiệu suất cao, component tái sử dụng, cộng đồng lớn, ecosystem phong phú. React đặc biệt phù hợp cho Single Page Applications (SPA).

### React Router DOM (Điều hướng trang)

- React Router DOM là thư viện điều hướng (routing) chính thức cho React, cho phép xây dựng Single Page Applications với nhiều trang/view mà không cần tải lại toàn bộ trang web.

- Thay vì mỗi URL yêu cầu tải trang mới từ server, React Router DOM sử dụng HTML5 History API để thay đổi URL trong trình duyệt mà không reload trang. Khi người dùng click vào link hoặc nhập URL, Router sẽ so khớp với các Route đã định nghĩa và render component tương ứng.

- Các thành phần chính:

- **BrowserRouter**: Wrapper component sử dụng HTML5 History API
- **Routes & Route**: Định nghĩa mapping giữa URL path và component
- **Link**: Thay thế thẻ a để điều hướng không reload trang
- **useNavigate**: Hook để điều hướng programmatically
- **useParams**: Hook lấy dynamic parameters từ URL

- Ưu điểm: Trải nghiệm người dùng mượt mà (không reload), quản lý lịch sử duyệt web, hỗ trợ nested routes, lazy loading components. Phù hợp cho SPA phức tạp với nhiều trang.

### Tailwind CSS (Framework CSS hiện đại)

- Tailwind CSS là framework CSS utility-first được tạo ra bởi Adam Wathan năm 2017, mang đến cách tiếp cận hoàn toàn mới so với các framework CSS truyền thống. Thay vì cung cấp các component UI có sẵn, Tailwind cung cấp hàng nghìn utility classes nhỏ để xây dựng UI trực tiếp trong HTML.

- Tailwind sử dụng JIT (Just-In-Time) compiler để chỉ generate CSS cho các class thực sự được dùng, giúp file CSS production cực nhỏ (thường chỉ 10-20KB sau

gzip). Framework này cung cấp hệ thống design tokens nhất quán (colors, spacing, typography) và hỗ trợ responsive design, dark mode ngay trong class names.

- Đặc điểm nổi bật:

+ **Utility-first**: Sử dụng các class nhỏ như flex, pt-4, text-center, bg-blue-500
+ **Responsive design**: Hỗ trợ breakpoints (mobile, tablet, desktop)
+ **Dark mode**: Tích hợp dark mode dễ dàng
+ **Customization**: Cấu hình tailwind.config.js cho colors, spacing, fonts
+ **PurgeCSS tích hợp**: Tự động xóa CSS không dùng

- Ưu điểm: Tốc độ phát triển nhanh, không phải đặt tên class, không CSS conflicts, file size nhỏ, dễ bảo trì. Đặc biệt phù hợp cho dự án cần UI custom và linh hoạt.

### Vite (Build tool)

- Vite (tiếng Pháp nghĩa là "nhanh") là build tool thế hệ mới được Evan You (tác giả Vue.js) tạo ra năm 2020 để thay thế các công cụ build chậm như Webpack. Vite tập trung vào tốc độ phát triển cực nhanh và trải nghiệm developer tuyệt vời.

- Điểm đột phá của Vite là sử dụng ES Modules (ESM) native của trình duyệt trong development mode. Thay vì bundle toàn bộ app, Vite chỉ transform file được request on-demand, khiến dev server khởi động gần như tức thì (cold start dưới 1s) và Hot Module Replacement cực nhanh. Trong production, Vite sử dụng Rollup để bundle code với tree-shaking tốt và tối ưu hóa cao.

- Tính năng chính:

- **Dev server nhanh**: Khởi động tức thì, HMR dưới 100ms
- **ES Modules native**: Không bundle trong dev mode
- **Build production**: Sử dụng Rollup, tree-shaking, code-splitting
- **Plugin ecosystem**: Tương thích Rollup plugins
- **Framework agnostic**: Hỗ trợ React, Vue, Svelte

- Ưu điểm: Tốc độ dev cực nhanh, config đơn giản, production build tối ưu, hỗ trợ TypeScript/JSX native. Vite đang trở thành tiêu chuẩn mới cho các dự án frontend hiện đại.

### Axios (HTTP client)

- Axios là thư viện HTTP client dựa trên Promise cho JavaScript, được phát triển năm 2016. Axios hoạt động trên cả browser và Node.js, cho phép gửi các HTTP requests một cách dễ dàng và linh hoạt hơn so với Fetch API native.

- Axios tự động transform JSON data, hỗ trợ interceptors để modify requests/responses globally, cancel requests, và xử lý errors tốt hơn Fetch. Axios cung cấp API đơn giản, hỗ trợ async/await, và có thể tạo instance với baseURL và headers mặc định.

- Tính năng nổi bật:

+ **Interceptors**: Modify request/response trước khi xử lý
+ **Auto JSON transform**: Tự động parse JSON response
+ **Request cancellation**: Hủy requests bằng AbortController
+ **Error handling**: Phân biệt network errors vs HTTP errors
+ **Timeout config**: Tự động timeout requests

Ưu điểm: API đơn giản, hỗ trợ interceptors mạnh mẽ, tương thích browser cũ, xử lý lỗi tốt. Phù hợp cho cả frontend (gọi API) và backend (scraping, API calls).

Lucide React - Thư viện icon vector:

- Lucide React là thư viện icon mã nguồn mở cung cấp hơn 1000 icon SVG có thể tùy chỉnh, được thiết kế đơn giản, nhất quán và hiện đại.

- Lucide được phát triển từ Feather Icons năm 2021, tối ưu hóa cho React với các props như size, color, strokeWidth. Icon được render dưới dạng SVG component, hỗ trợ tree-shaking để giảm bundle size, chỉ import icon thực sự sử dụng.

- Cách hoạt động: Import icon như component React, truyền props để tùy chỉnh kích thước và màu sắc, render trực tiếp vào JSX.

- Ưu điểm: Nhẹ, có thể tùy chỉnh, không cần tải file ảnh, hỗ trợ TypeScript, accessibility tốt. Lucide React giúp tạo giao diện chuyên nghiệp với icon vector sắc nét trên mọi màn hình.

React Syntax Highlighter - Thư viện tô màu code:

- React Syntax Highlighter là thư viện React component hiển thị code với syntax highlighting, hỗ trợ hơn 180 ngôn ngữ lập trình và 50+ theme màu sắc.

- Thư viện sử dụng Prism.js hoặc Highlight.js làm engine tô màu, tự động nhận diện ngôn ngữ qua language prop, parse code thành tokens và tô màu theo theme. Hỗ trợ line numbers, code wrapping, copy button, custom styles.

- Cách hoạt động: Truyền code string vào component, chọn language và theme, library tự động parse và render HTML có màu sắc.

- Ưu điểm: Hỗ trợ nhiều ngôn ngữ, nhiều theme đẹp, dễ tích hợp, responsive, có thể custom. React Syntax Highlighter giúp hiển thị code rõ ràng, dễ đọc trong trang bài học lập trình.

### 1.3. Phát triển máy chủ (Backend Development)

Node.js:

- Node.js là môi trường runtime mã nguồn mở cho JavaScript ở phía server, cho phép chạy JavaScript ngoài trình duyệt.
- Node.js do Ryan Dahl tạo năm 2009, nay được OpenJS Foundation quản lý, cho phép chạy JavaScript ở server-side với hiệu suất cao nhờ mô hình non-blocking I/O và event-driven. Xây dựng trên engine V8 của Chrome, nó sử dụng NPM (Node Package Manager) để quản lý package.
- Cách hoạt động: Sử dụng module system (CommonJS/ES modules), NPM để cài thư viện như Express cho web server. Node.js sử dụng mô hình event-driven, non-blocking I/O để xử lý nhiều kết nối đồng thời mà không chặn thread.
- Ưu điểm: Hiệu suất cao cho ứng dụng real-time (chat, API), full-stack development dễ dàng vì dùng chung ngôn ngữ JavaScript cho cả frontend lẫn backend. Node.js lý tưởng cho các hệ thống real-time như chat app hay API, giúp xử lý nhiều kết nối đồng thời mà không tốn tài nguyên.

JavaScript - Ngôn ngữ lập trình phía server:

- JavaScript ở phía server (Server-Side JavaScript - SSJS) sử dụng JavaScript để xử lý logic backend, thay vì chỉ client-side.
- JavaScript phía server bắt nguồn từ những năm 1990 với Netscape Enterprise Server, nhưng thực sự bùng nổ nhờ Node.js từ 2009, cho phép xử lý logic backend như yêu cầu HTTP hay truy vấn database. Nó giữ nguyên tính năng động của JavaScript client-side, nhưng tập trung vào routing và middleware qua framework như Express.
- Cách hoạt động: Sử dụng framework như Express để tạo API, xử lý routing, middleware.
- Ưu điểm: Đồng nhất ngôn ngữ frontend/backend (full-stack JavaScript), dễ scale. Ứng dụng phổ biến trong việc xây dựng API scalable, quản lý authentication hay data processing, giúp developer tiết kiệm thời gian vì không cần học ngôn ngữ mới khi chuyển từ frontend sang backend.

Express.js (Web Framework)

- Express.js là framework web minimalist và linh hoạt nhất cho Node.js, được TJ Holowaychuk tạo ra năm 2010. Express cung cấp các tính năng cơ bản để xây dựng web server và API mà không áp đặt cấu trúc cứng nhắc.

- Express hoạt động dựa trên middleware pattern, cho phép xử lý requests qua một chuỗi các hàm middleware. Mỗi middleware có thể modify request/response object, kết thúc request-response cycle, hoặc gọi middleware tiếp theo. Express cung cấp routing mạnh mẽ để map HTTP methods (GET, POST, PUT, DELETE) với URL paths và xử lý request tương ứng.

- Tính năng chính:

+ **Routing**: Định nghĩa routes cho các HTTP methods và URL patterns
+ **Middleware**: Chuỗi xử lý requests (body-parser, CORS, authentication...)
+ **Template engines**: Hỗ trợ render views (EJS, Pug, Handlebars...)
+ **Static files**: Serve static files (HTML, CSS, images)
+ **Error handling**: Xử lý lỗi tập trung

- Cách hoạt động: Request đi qua middleware chain, được route đến handler tương ứng, xử lý logic và trả về response.

### Bcryptjs (Mã hóa mật khẩu)

- Bcryptjs là thư viện mã hóa mật khẩu dựa trên thuật toán bcrypt, được viết hoàn toàn bằng JavaScript. Bcryptjs là phiên bản pure-JavaScript của bcrypt, phù hợp cho các môi trường không hỗ trợ C++ bindings.

- Bcrypt sử dụng thuật toán Blowfish cipher với salt ngẫu nhiên và cost factor (rounds) để làm chậm quá trình hash, giúp chống brute-force attacks. Mỗi lần hash cùng một password sẽ cho kết quả khác nhau nhờ salt ngẫu nhiên. Cost factor quy định số vòng lặp, mặc định là 10 (1024 vòng), có thể tăng lên 12-15 cho bảo mật cao hơn.

- Cách hoạt động: Tạo salt ngẫu nhiên, kết hợp với password và hash nhiều vòng lặp. Khi verify, sẽ hash password input với cùng salt và so sánh kết quả.

- Ưu điểm: Bảo mật cao (chống rainbow table, brute-force), salt tự động, adaptive hashing (có thể tăng cost factor theo thời gian). Bcryptjs là tiêu chuẩn cho password hashing trong Node.js applications.

### Jsonwebtoken (JWT authentication)

- Jsonwebtoken là thư viện triển khai JSON Web Token standard (RFC 7519) cho Node.js, được sử dụng rộng rãi cho authentication và authorization. JWT là một

chuẩn mở cho phép truyền thông tin an toàn giữa các bên dưới dạng JSON object được signed.

- Một JWT token gồm 3 phần cách nhau bởi dấu chấm: Header (thuật toán mã hóa), Payload (dữ liệu user), và Signature (chữ ký xác thực). Header và Payload được encode base64url, Signature được tạo bằng cách hash với thuật toán như HS256, RS256. Token được gửi kèm trong Authorization header cho mỗi request, server verify signature để xác thực.

- Cách hoạt động: Server tạo token sau login, client gửi trong header HTTP; server verify mà không lưu session (stateless).

- Ưu điểm: Bảo mật, stateless (server không cần lưu session), scalable, hỗ trợ cross-domain, compact size. JWT là giải pháp authentication phổ biến cho RESTful APIs và microservices.

### Dotenv (Environment variables)

- Dotenv là thư viện nhỏ gọn cho phép load các biến môi trường (environment variables) từ file .env vào process.env trong Node.js applications. Được phát triển năm 2013, dotenv giúp tách riêng cấu hình (credentials, API keys, URLs) ra khỏi code.

- Dotenv đọc file .env ở thư mục gốc project, parse các cặp key-value, và inject vào process.env. File .env thường chứa thông tin nhạy cảm như database credentials, API keys, JWT secrets, và được thêm vào .gitignore để không commit lên Git.

- Cách hoạt động: Load file .env vào process.env khi khởi động ứng dụng.

Ưu điểm: Bảo mật (không hardcode secrets), dễ config cho nhiều môi trường (dev/staging/prod), tuân thủ 12-factor app methodology. Dotenv là thư viện cơ bản trong hầu hết Node.js projects.

### Nodemon (Dev server auto-reload)

- Nodemon là công cụ giám sát (monitor) thay đổi file và tự động restart Node.js application trong quá trình development. Được phát triển bởi Remy Sharp, nodemon giúp tăng tốc độ phát triển bằng cách loại bỏ việc phải restart server thủ công sau mỗi lần sửa code.

- Nodemon watch các file trong project (mặc định là .js, .json, .mjs), khi phát hiện thay đổi sẽ tự động kill process cũ và start lại. Có thể config ignore files, delay

restart, và custom commands. Nodemon chỉ nên dùng trong development, không phù hợp cho production.

- Cách hoạt động: Giám sát thay đổi file, tự động restart Node.js application.

- Ưu điểm: Tăng năng suất dev (không cần restart thủ công), config linh hoạt, tích hợp tốt với các frameworks. Nodemon là công cụ thiết yếu cho Node.js development.

### Axios (HTTP client (cho scraping))

- Axios trong backend được sử dụng để gửi HTTP requests đến các website bên ngoài, thường dùng cho web scraping hoặc gọi API của bên thứ ba.

- Trong Node.js, Axios sử dụng http module thay vì XMLHttpRequest như trong browser. Axios cho phép scrape dữ liệu từ websites, gọi third-party APIs, và xử lý responses một cách đơn giản.

- Ưu điểm: API đơn giản, hỗ trợ interceptors, xử lý lỗi tốt, phù hợp cho scraping và API integration.

### Cheerio (Web scraping)

- Cheerio là thư viện fast, flexible và lean implementation của jQuery core cho server-side, được thiết kế đặc biệt để parsing và manipulating HTML/XML. Phát triển năm 2012, Cheerio cung cấp jQuery-like API nhưng không cần browser, chạy trên Node.js với hiệu suất cực cao.

- Cheerio sử dụng htmlparser2 để parse HTML thành DOM tree, sau đó cung cấp jQuery selectors (CSS selectors) để query và manipulate. Khác với Puppeteer/Playwright (headless browser), Cheerio không execute JavaScript, chỉ parse static HTML, nên rất nhanh và nhẹ. Cheerio phù hợp cho scraping các trang web render HTML server-side.

- Cách hoạt động: Parse HTML thành DOM tree, sử dụng jQuery selectors để query và extract data.

- Ưu điểm: Rất nhanh (10-50x nhanh hơn Puppeteer), API giống jQuery (dễ học), không cần browser, memory-efficient. Cheerio là lựa chọn tốt nhất cho scraping static websites.

### CORS (Cross-Origin Resource Sharing) - Middleware bảo mật:

- CORS là cơ chế bảo mật cho phép server kiểm soát resource nào có thể được truy cập từ domain khác, ngăn chặn cross-origin request không được phép.

- CORS được W3C chuẩn hóa năm 2014 dựa trên Same-Origin Policy của trình duyệt. Khi frontend khác origin gọi API, browser gửi preflight request (OPTIONS) để kiểm tra, server trả về header Access-Control-Allow-Origin, Access-Control-Allow-Methods, Access-Control-Allow-Headers để cho phép hoặc từ chối.

- Cách hoạt động: Server thêm header CORS vào response, browser kiểm tra header và quyết định cho phép request hay không.

- Ưu điểm: Bảo mật, linh hoạt, cho phép frontend-backend riêng domain, hỗ trợ credentials. CORS middleware trong Express giúp frontend gọi API backend mà không bị browser block.

Express Rate Limit - Middleware giới hạn request:

- Express Rate Limit là middleware Node.js giới hạn số lượng request từ một IP trong khoảng thời gian nhất định, bảo vệ API khỏi spam, brute-force, DDoS.

- Thư viện hoạt động theo thuật toán sliding window hoặc fixed window, lưu số lần request của mỗi IP vào memory hoặc Redis, kiểm tra mỗi request và reject nếu vượt giới hạn. Trả về status 429 Too Many Requests kèm header Retry-After.

- Cách hoạt động: Đếm số request theo IP trong windowMs, so sánh với max, block nếu vượt quá và trả về error message.

- Ưu điểm: Dễ cấu hình, bảo vệ server, tùy chỉnh được message và status code, hỗ trợ whitelist. Express Rate Limit ngăn chặn tấn công brute-force đăng nhập và spam API.

Nodemailer - Thư viện gửi email:

- Nodemailer là thư viện Node.js mạnh mẽ nhất để gửi email, hỗ trợ SMTP, OAuth2, AWS SES, Sendgrid và nhiều transport khác.

- Nodemailer phát triển từ năm 2010, sử dụng transport layer để kết nối SMTP server (Gmail, Outlook, custom), hỗ trợ HTML email, attachment, embedded images, Unicode, authentication qua username/password hoặc OAuth2. Tạo transporter với cấu hình SMTP, sau đó gọi sendMail() với mailOptions chứa to, subject, html.

- Cách hoạt động: Tạo transporter kết nối SMTP, định nghĩa mailOptions (from, to, subject, html), gọi sendMail() để gửi.

- Ưu điểm: Dễ sử dụng, hỗ trợ HTML rich email, attachment, nhiều transport, async/await, Unicode. Nodemailer gửi email xác thực tài khoản và reset mật khẩu với template HTML chuyên nghiệp.

### 1.4.Cơ sở dữ liệu(Database)

Docker

- Docker là nền tảng containerization mã nguồn mở được Solomon Hykes tạo ra năm 2013, giúp đóng gói ứng dụng và dependencies vào containers nhẹ, portable, và isolated. Docker đã cách mạng hóa cách deploy và quản lý ứng dụng, trở thành tiêu chuẩn trong DevOps và cloud computing.

- Container Docker là một lightweight, standalone package chứa everything để chạy ứng dụng: code, runtime, system tools, libraries, settings. Khác với Virtual Machines cần guest OS riêng, containers share kernel của host OS, nên nhẹ hơn và start nhanh hơn (giây thay vì phút).

- Kiến trúc Docker:

  + **Docker Engine**: Runtime chạy containers
  + **Dockerfile**: Script định nghĩa image
  + **Docker Image**: Template read-only để tạo containers
  + **Docker Container**: Instance đang chạy của image
  + **Docker Compose**: Tool orchestrate multi-container apps

- Ưu điểm: Consistency (dev = prod), portable (chạy mọi nơi), isolated (không conflict dependencies), scalable, efficient resource usage. Docker là công cụ thiết yếu cho modern application deployment.

Supabase

- Supabase là nền tảng Backend-as-a-Service (BaaS) mã nguồn mở được ra mắt năm 2020, được quảng cáo là "Firebase alternative" cho PostgreSQL. Supabase cung cấp database, authentication, storage, và real-time subscriptions out-of-the-box, giúp developers tập trung vào frontend mà không cần build backend từ đầu.

- Supabase sử dụng PostgreSQL (thay vì NoSQL như Firebase), cung cấp RESTful API tự động từ database schema, real-time API dựa trên PostgreSQL triggers, và Row Level Security (RLS) cho phân quyền chi tiết. Supabase Studio (dashboard) cho phép quản lý database, run SQL queries, xem logs.

Tính năng chính:

+ **PostgreSQL Database**: Relational database mạnh mẽ, hỗ trợ SQL đầy đủ
+ **Auto RESTful API**: Tự động generate API từ tables/views
+ **Real-time subscriptions**: Listen database changes qua websockets
+ **Authentication**: Email/password, OAuth (Google, GitHub...), JWT
+ **Row Level Security (RLS)**: Phân quyền ở database level
+ **Storage**: Upload/download files với access control
+ **Edge Functions**: Serverless functions chạy Deno runtime

- Ưu điểm: Setup nhanh (không cần backend), PostgreSQL mạnh mẽ, real-time built-in, auth đầy đủ, pricing hợp lý (free tier generous). Supabase phù hợp cho MVPs, startups, và apps cần database quan hệ với real-time.

PostgreSQL

- PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở mạnh mẽ nhất hiện nay, được phát triển từ năm 1986 tại University of California, Berkeley. PostgreSQL tuân thủ chuẩn SQL và cung cấp nhiều tính năng nâng cao.

- PostgreSQL hỗ trợ ACID (Atomicity, Consistency, Isolation, Durability) đầy đủ, đảm bảo tính toàn vẹn dữ liệu. Nó cung cấp các tính năng như JSON/JSONB cho dữ liệu semi-structured, Full-Text Search, Foreign Data Wrappers, Triggers, Stored Procedures. PostgreSQL có khả năng mở rộng cao với extensions như PostGIS (GIS data), pg_trgm (fuzzy search).

- Tính năng nổi bật:

+ **ACID compliance**: Đảm bảo tính toàn vẹn giao dịch
+ **Advanced data types**: JSON, Array, UUID, XML, Geographic data
+ **Full-Text Search**: Tìm kiếm văn bản mạnh mẽ
+ **Triggers & Stored Procedures**: Logic phức tạp ở database level
+ **Row Level Security (RLS)**: Phân quyền chi tiết đến từng row
+ **MVCC**: Multi-Version Concurrency Control cho performance cao

- Cách hoạt động: Lưu trữ dữ liệu dạng bảng (tables) với quan hệ (relationships), sử dụng SQL để query và manipulate data.

- Ưu điểm: Mạnh mẽ, reliable, hỗ trợ SQL đầy đủ, extensible, performance tốt, miễn phí. PostgreSQL là lựa chọn hàng đầu cho applications cần database quan hệ với tính năng nâng cao. Supabase sử dụng PostgreSQL làm database engine.

### 1.5.Các công nghệ sử dụng khác

MVC Pattern- Kiến trúc ứng dụng:

- MVC (Model-View-Controller) là mô hình kiến trúc phần mềm tách biệt ứng dụng thành ba phần: Model (dữ liệu/logic), View (giao diện), Controller (xử lý yêu cầu).

- MVC được Trygve Reenskaug đề xuất năm 1979 cho ngôn ngữ Smalltalk, giúp code dễ bảo trì, testable, và scalable. Model quản lý dữ liệu và business logic, View hiển thị giao diện cho người dùng, Controller xử lý input từ user và điều phối giữa Model và View.

- Cách hoạt động: Controller nhận input từ user, cập nhật Model, rồi render View.

- Ưu điểm: Separation of Concerns (SoC), dễ bảo trì, dễ test. MVC được sử dụng rộng rãi trong các framework web như Laravel, Spring, ASP.NET.

RESTful API - Kiểu kiến trúc API:

- REST (Representational State Transfer) là phong cách kiến trúc cho API, sử dụng HTTP để trao đổi dữ liệu giữa client và server.

- REST do Roy Fielding mô tả năm 2000 trong luận án tiến sĩ, dựa trên sáu nguyên tắc: Client-Server, Stateless, Cacheable, Uniform Interface, Layered System, Code on Demand. REST sử dụng HTTP methods (GET, POST, PUT, DELETE) để thao tác với resource qua URI.

- Cách hoạt động: Sử dụng methods HTTP (GET, POST, PUT, DELETE) với URI đại diện resource.

- Ưu điểm: Scalable, dễ tích hợp, stateless, cacheable. RESTful API là tiêu chuẩn phổ biến nhất cho việc xây dựng API, giúp kết nối frontend với backend một cách hiệu quả.

JSON - Định dạng trao đổi dữ liệu:

- JSON (JavaScript Object Notation) là định dạng dữ liệu nhẹ, dễ đọc, dùng để trao đổi dữ liệu giữa client và server.

- JSON được Douglas Crockford phổ biến năm 2001 và chuẩn hóa năm 2014. Dựa trên key-value pairs, array, object, dễ parse và nhỏ gọn hơn XML. Là định dạng dữ liệu độc lập với ngôn ngữ lập trình, được hỗ trợ bởi hầu hết các ngôn ngữ hiện đại.

- Cách hoạt động: Parse/stringify để chuyển đổi giữa JSON string và JavaScript object.

- Ưu điểm: Ngôn ngữ độc lập, nhỏ gọn hơn XML, dễ đọc cho con người và máy. JSON là format mặc định cho REST API, được sử dụng rộng rãi để truyền dữ liệu như user details, API responses một cách an toàn và hiệu quả.

JToken/JWT - Xác thực người dùng:

- JWT (JSON Web Token) là chuẩn mở (RFC 7519) để truyền thông tin an toàn dưới dạng token JSON, được sử dụng rộng rãi cho authentication và authorization.
- JWT được chuẩn hóa năm 2015. Token gồm ba phần: Header (algorithm), Payload (claims như user ID, email), Signature (ký để xác thực). JWT cho phép xác thực stateless, nghĩa là server không cần lưu session, chỉ cần verify signature của token.
- Cách hoạt động: Server tạo token sau login, client gửi token trong header HTTP (Authorization: Bearer token); server verify signature mà không lưu session (stateless).
- Ưu điểm: Bảo mật, scalable, stateless, hỗ trợ cross-domain. JWT là giải pháp authentication phổ biến cho API, cho phép user đăng nhập một lần và sử dụng token để xác thực các requests tiếp theo.

Groq AI (Artificial Intelligence API)

- Groq AI là nền tảng inference AI cung cấp API để chạy các Large Language Models (LLMs) với tốc độ cực nhanh. Groq được thành lập năm 2016, chuyên về phần cứng AI (Groq LPU) và cung cấp API public từ năm 2024 để developers tích hợp AI vào ứng dụng.

- Groq AI hỗ trợ các model như Llama 3.3 70B, Mixtral, Gemma với inference speed siêu nhanh (500+ tokens/giây), nhanh gấp 10-50 lần so với các providers khác. Groq API tương thích với OpenAI API format, cho phép dễ dàng migration. Groq cung cấp free tier generous với rate limits hợp lý cho development và production nhỏ.

Tính năng chính:

+ **Ultra-fast inference**: Tốc độ sinh text cực nhanh nhờ Groq LPU hardware
+ **Multiple models**: Llama 3.3 70B, Mixtral 8x7B, Gemma 7B
+ **OpenAI-compatible API**: Dễ dàng switch từ OpenAI sang Groq
+ **Streaming support**: Hỗ trợ streaming response real-time
+ **Function calling**: Cho phép LLM gọi external functions
+ **Free tier**: Miễn phí với rate limits hợp lý

Cách hoạt động: Send request với messages array (system prompt, user messages, assistant responses) đến Groq API, nhận response từ LLM. Hỗ trợ conversation history để duy trì context.

- Ưu điểm: Cực kỳ nhanh (nhanh nhất hiện tại), miễn phí với free tier generous, API đơn giản, hỗ trợ nhiều models mạnh. Groq phù hợp cho chatbots, AI assistants, content generation. Trong hệ thống, Groq AI được sử dụng để xây dựng ChatBot AI hỗ trợ học sinh học Tin học THPT, trả lời câu hỏi bằng tiếng Việt với độ chính xác cao và tốc độ phản hồi gần như tức thì.

Git - Quản lý phiên bản mã nguồn:

- Git là hệ thống kiểm soát phiên bản phân tán (DVCS - Distributed Version Control System) để theo dõi thay đổi code và collaboration trong team.
- Git do Linus Torvalds tạo năm 2005 cho Linux kernel. Git lưu trữ lịch sử thay đổi qua commit, hỗ trợ branch để phát triển tính năng độc lập, và merge để tích hợp code. Mỗi repository Git là bản sao đầy đủ của lịch sử project.
- Cách hoạt động: Sử dụng các lệnh như git init, add, commit, push, pull; hỗ trợ collaboration qua GitHub, GitLab, Bitbucket.
- Ưu điểm: Nhanh, an toàn dữ liệu (hash SHA-1), hỗ trợ offline, branching mạnh mẽ. Git giúp quản lý code đội nhóm, theo dõi thay đổi, revert lỗi, và collaboration hiệu quả. Git là công cụ thiết yếu trong phát triển phần mềm hiện đại.

# Chương 4: Phát triển chương trình hệ thống

## Phần I: Các công cụ áp dụng vào chương trình và cách hoạt động

### 1.Phát triển giao diện người dùng (Frontend Development)

HTML (HyperText Markup Language)

Tạo cấu trúc nền tảng cho toàn bộ giao diện web, bao gồm các trang chọn khối lớp, danh sách bài học, trang làm bài thi, trang xem kết quả. HTML định nghĩa các phần tử như form đăng nhập, nút bấm, bảng hiển thị câu hỏi trắc nghiệm.

CSS (Cascading Style Sheets)

Tạo giao diện đẹp mắt cho toàn bộ hệ thống, định dạng màu sắc cho từng khối lớp (lớp 10, 11, 12), tạo hiệu ứng hover cho các nút bấm, thiết kế bố cục responsive để hiển thị tốt trên điện thoại và máy tính.

JavaScript

Xử lý tương tác người dùng như đếm ngược thời gian làm bài thi, kiểm tra câu trả lời trước khi submit, hiển thị thông báo khi hoàn thành bài thi, chuyển trang giữa các câu hỏi. JavaScript cũng gửi request đến backend để lấy đề thi và nộp bài.

React

Xây dựng toàn bộ giao diện người dùng thành các component độc lập như LessonModal (modal quản lý bài học), QuestionModal (modal thêm/sửa câu hỏi), ChatBot (trợ lý AI), AdminDashboard (trang quản trị). React quản lý trạng thái đăng nhập qua AuthContext và tự động cập nhật giao diện khi dữ liệu thay đổi.

React Router DOM

Quản lý điều hướng giữa các trang như chọn khối lớp (/), trang chủ khối lớp (/grade/:grade), danh sách bài học (/grade/:grade/lessons), trang làm bài thi (/grade/:grade/exam/:testId), trang kết quả (/grade/:grade/result). Router cũng bảo vệ các trang admin và lịch sử chỉ cho phép người dùng đã đăng nhập truy cập.

Tailwind CSS

Tạo giao diện nhất quán cho toàn bộ hệ thống với các class có sẵn như button, card, modal, form. Tailwind giúp thiết kế nhanh các component như nút chọn khối lớp, bảng hiển thị danh sách đề thi, form đăng nhập, ChatBot modal mà không cần viết CSS thủ công.

Vite

Khởi chạy server phát triển frontend tại port 5173, tự động reload trang khi sửa code, build project thành file tối ưu khi deploy. Vite làm việc cùng Docker để chạy container frontend và đồng bộ code realtime.

### Axios (Frontend)

Gửi HTTP request từ frontend đến backend API như gọi /api/auth/login để đăng nhập, /api/exam/start/:id để lấy đề thi, /api/exam/submit để nộp bài, /api/lessons để lấy danh sách bài học, /api/chatbot/ask để chat với AI. Axios tự động xử lý JSON và gửi JWT token trong header để xác thực.

### Lucide React

Cung cấp hệ thống icon vector cho toàn bộ giao diện như icon Home, Book, FileText cho menu điều hướng, icon Bell cho thông báo, icon MessageCircle cho ChatBot, icon User cho trang cá nhân, icon LogOut cho đăng xuất. Lucide React giúp UI thống nhất và chuyên nghiệp mà không cần thiết kế icon thủ công.

### React Syntax Highlighter

Hiển thị code Python có màu sắc (syntax highlighting) trong trang chi tiết bài học, giúp học sinh dễ đọc và phân biệt các thành phần code như keyword, function, variable, string. Thư viện tự động nhận diện ngôn ngữ lập trình và tô màu chuẩn như trong IDE.

**2.Phát triển máy chủ (Backend Development)**

### Node.js

Chạy server backend tại port 5000, lắng nghe các request từ frontend, xử lý logic nghiệp vụ như chấm điểm bài thi, quản lý phiên đăng nhập, tạo JWT token, lưu kết quả thi vào database. Node.js chạy trong Docker container và kết nối với Supabase PostgreSQL.

### JavaScript (Server-side)

Viết toàn bộ logic backend như xác thực người dùng (auth.controller.js), quản lý đề thi (exam.controller.js), chấm điểm tự động (validation-optimized.controller.js), quản lý bài học (lessons.controller.js), xử lý ChatBot AI (chatbot.controller.js), thống kê admin (statistics.controller.js).

### Express.js

Tạo RESTful API với các endpoint như POST /api/auth/register (đăng ký), POST /api/auth/login (đăng nhập), GET /api/exam/start/:id (lấy đề thi), POST

/api/exam/submit (nộp bài), POST /api/chatbot/ask (chat AI). Express xử lý CORS, parse JSON request, routing, middleware xác thực JWT, rate limiting.

### Bcryptjs

Mã hóa mật khẩu người dùng trước khi lưu vào database bằng thuật toán Blowfish với salt rounds. Khi đăng nhập, Bcryptjs so sánh mật khẩu nhập vào với hash đã lưu để xác thực. Dùng phiên bản JavaScript thuần để tương thích Docker và mọi môi trường.

### JWT (JSON Web Token)

Tạo token xác thực sau khi đăng nhập thành công, token chứa user_id và role (student/admin). Frontend gửi JWT trong header Authorization để truy cập các API bảo mật như /api/history, /api/admin. Middleware auth.middleware.js verify token trước khi cho phép truy cập.

### Dotenv

Load biến môi trường từ file .env như SUPABASE_URL, SUPABASE_SERVICE_KEY, JWT_SECRET, GROQ_API_KEY, PORT. Dotenv giúp bảo mật thông tin nhạy cảm và dễ dàng thay đổi cấu hình giữa môi trường development và production.

### Nodemon

Tự động restart server backend khi có thay đổi code trong quá trình phát triển. Chạy qua lệnh npm run dev, giúp developer không phải tắt/mở server thủ công mỗi lần sửa code.

### Axios (Backend)

Gửi HTTP request từ backend đến các API bên ngoài như scrape dữ liệu câu hỏi từ website VietJack trong các script scrape_vietjack.js. Axios xử lý request/response và hỗ trợ async/await.

### Cheerio

Parse HTML từ các trang web giáo dục để scrape câu hỏi trắc nghiệm, bài học, đề thi. Cheerio hoạt động như jQuery trên server, giúp trích xuất nội dung từ các selector HTML và lưu vào database.

### CORS (Cross-Origin Resource Sharing)

Cho phép frontend (chạy tại http://localhost:5173) gọi API đến backend (chạy tại http://localhost:5000) mặc dù khác domain. CORS middleware xử lý preflight request và thêm header Access-Control-Allow-Origin để trình duyệt không block request từ frontend.

### Express Rate Limit

Giới hạn số lượng request mỗi IP được gửi đến API trong khoảng thời gian nhất định để chống spam, brute-force đăng nhập, DDoS. Rate limiter middleware áp dụng cho tất cả endpoint /api/ và trả về lỗi 429 Too Many Requests nếu vượt giới hạn.

### Nodemailer

Gửi email tự động cho các chức năng như xác thực tài khoản, reset mật khẩu khi quên, thông báo kết quả thi. Nodemailer kết nối với SMTP server để gửi email HTML có định dạng đẹp kèm link xác thực hoặc mã OTP.

## 3.Cơ sở dữ liệu (Database)

### Docker

Chạy toàn bộ hệ thống trong container gồm frontend (exam-frontend tại port 5173) và backend (exam-backend tại port 5000). Docker đảm bảo môi trường development nhất quán, dễ deploy, tự động mount volume để đồng bộ code, và tạo network bridge để frontend-backend giao tiếp.

### Supabase

Lưu trữ toàn bộ dữ liệu hệ thống trên cloud PostgreSQL gồm bảng users (tài khoản), subjects (môn học), lessons (bài học), tests (đề thi), questions (câu hỏi), test_results (kết quả thi), chatbot_qa (câu hỏi ChatBot). Supabase cung cấp API client (@supabase/supabase-js) để backend query, insert, update, delete dữ liệu.

### PostgreSQL

Hệ quản trị cơ sở dữ liệu quan hệ chạy trên nền tảng Supabase, lưu trữ dữ liệu có cấu trúc với các relationship như lessons thuộc subjects, questions thuộc tests, test_results lưu user_id và test_id. PostgreSQL hỗ trợ transaction, foreign key, index để đảm bảo tính toàn vẹn dữ liệu.

## 4.Công nghệ bổ trợ khác (Other Technologies)

### Mô hình MVC (Model-View-Controller)

Tổ chức code backend theo cấu trúc rõ ràng: Model (tương tác Supabase qua config/supabase.js), View (JSON response), Controller (xử lý logic trong

controllers/), Routes (định nghĩa endpoint trong routes/). Cấu trúc này giúp code dễ bảo trì và mở rộng.

### RESTful API

Thiết kế API theo chuẩn REST với HTTP methods: GET (lấy dữ liệu), POST (tạo mới), PUT (cập nhật), DELETE (xóa). Endpoint có cấu trúc rõ ràng như /api/lessons/:id, /api/tests/grade/:grade, /api/exam/submit. Response trả về JSON với status code chuẩn (200, 400, 401, 500).

### JSON (JavaScript Object Notation)

Định dạng dữ liệu trao đổi giữa frontend-backend và backend-database. JSON truyền tải thông tin đề thi, danh sách câu hỏi, kết quả thi, dữ liệu đăng nhập. Express tự động parse JSON request và response JSON.

### Git

Quản lý phiên bản code, theo dõi lịch sử thay đổi, tạo branch để phát triển tính năng mới, merge code, push lên GitHub. Git giúp team cộng tác và rollback khi có lỗi.

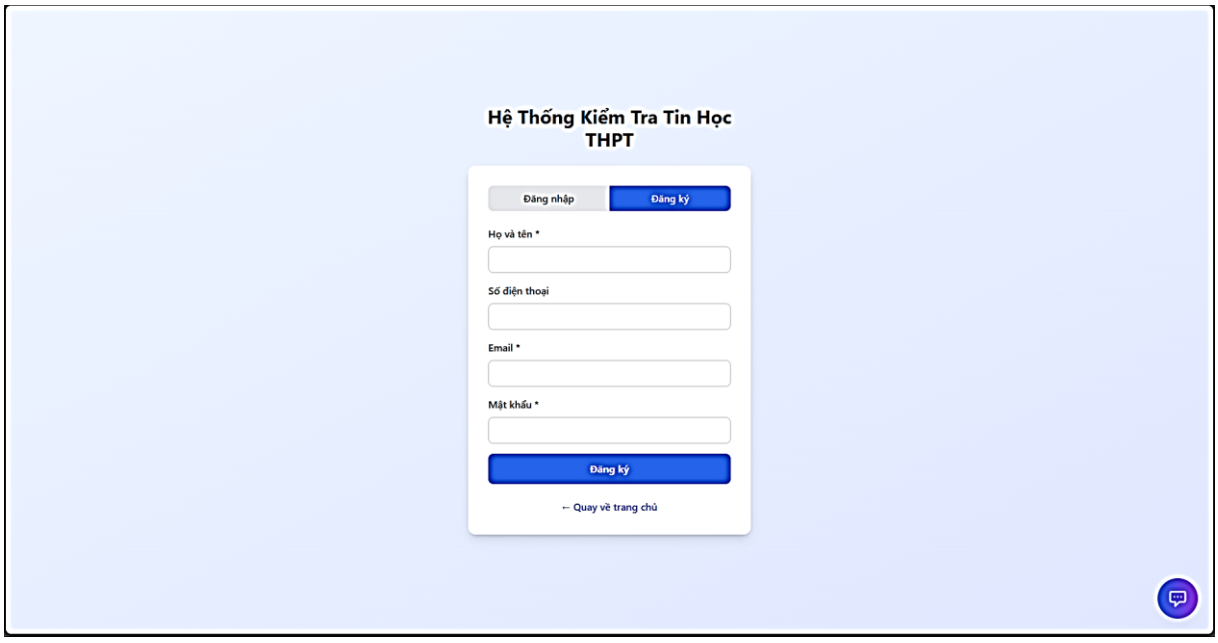
### Groq AI

Cung cấp API để ChatBot trả lời câu hỏi học sinh bằng model Llama 3.3 70B Versatile. Backend gửi request đến Groq API qua endpoint /api/chatbot/ask, kèm system prompt tiếng Việt và lịch sử hội thoại. Groq AI đọc kiến thức từ bảng chatbot_qa để trả lời chính xác hơn về môn Tin học THPT.

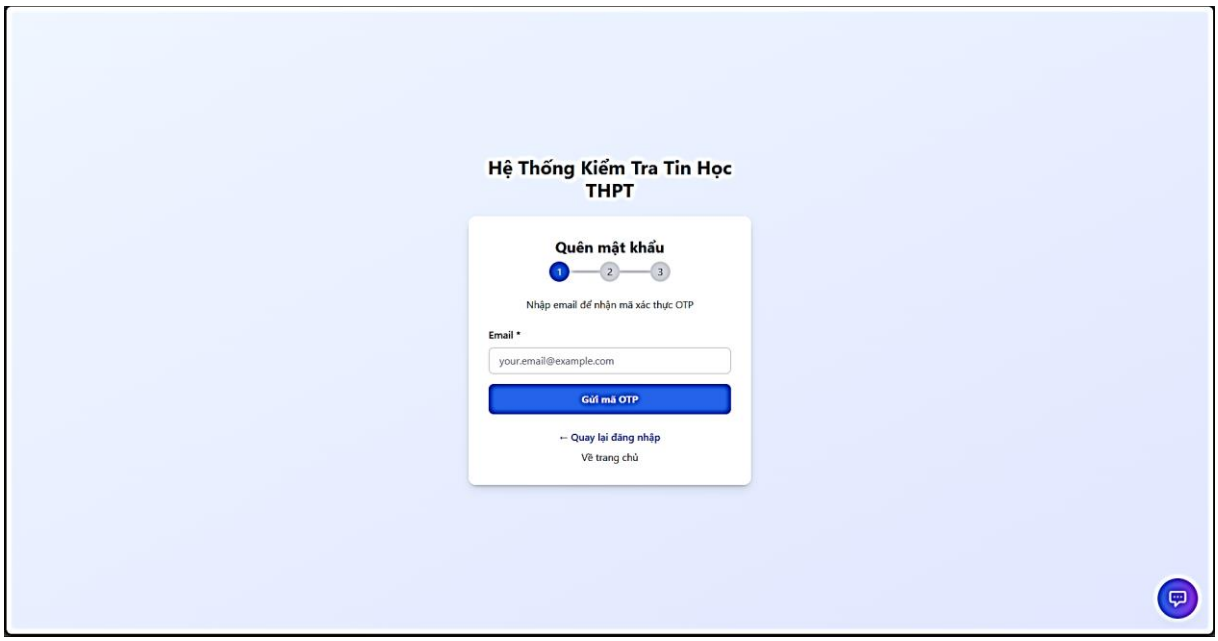
# Phần II: Giao diện của chương trình

## I. Giao diện login

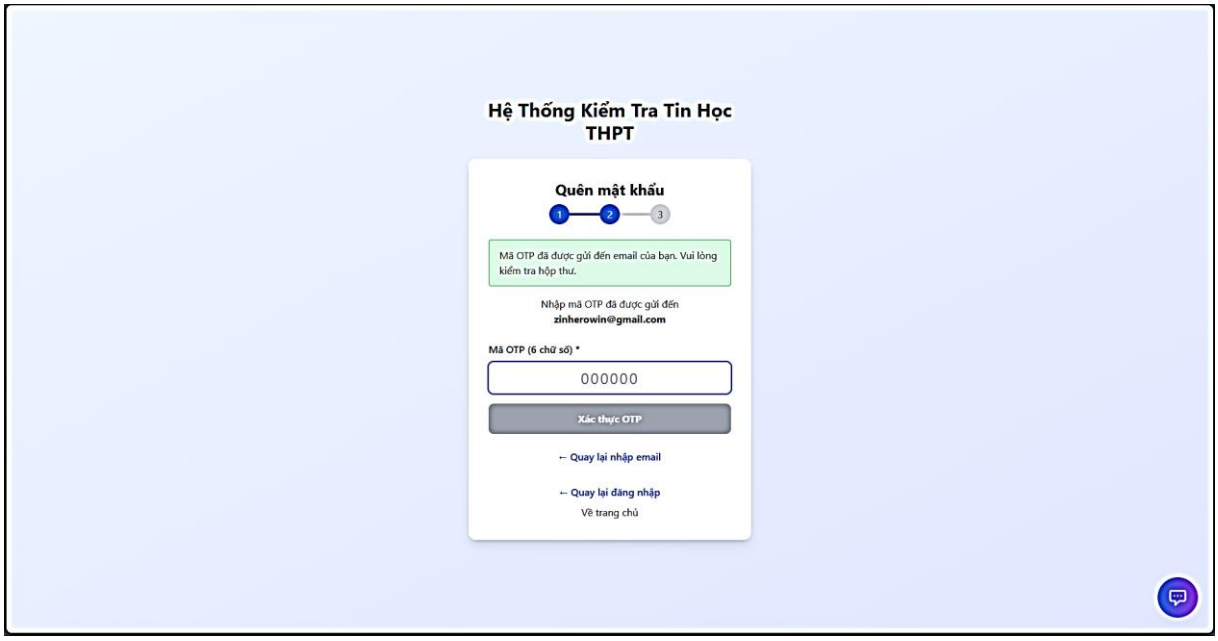

Giao diện đăng nhập



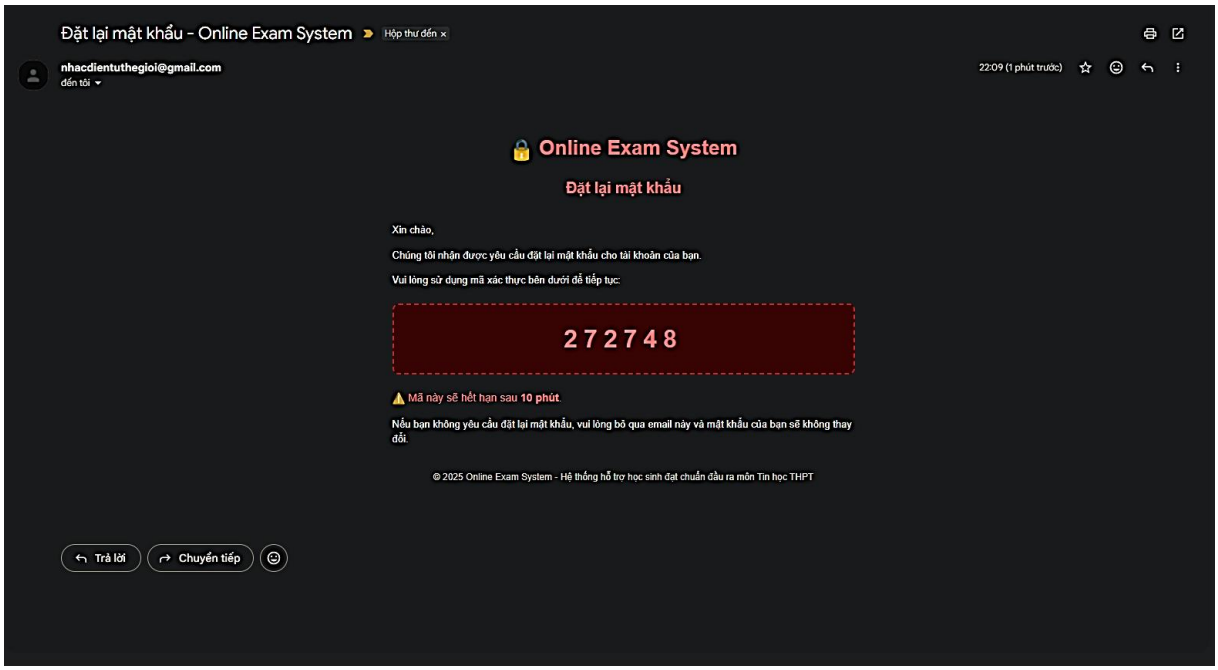

Giao diện đăng ký




Giao diện chức năng quên mật khẩu 1



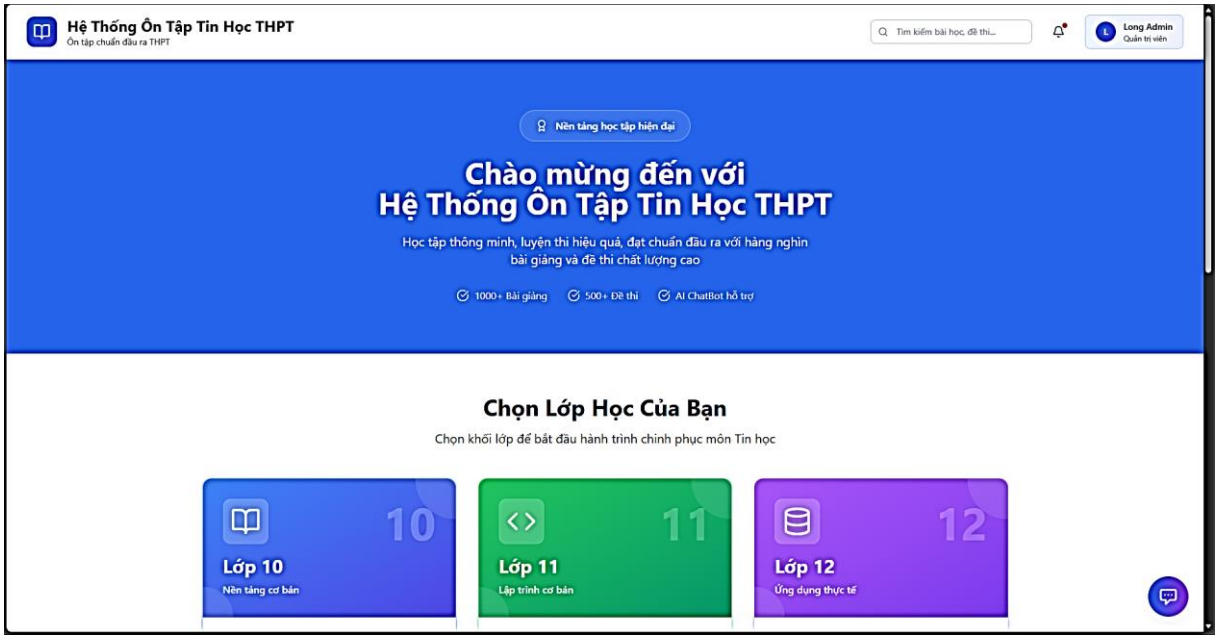

Giao diện chức năng quên mật khẩu 2



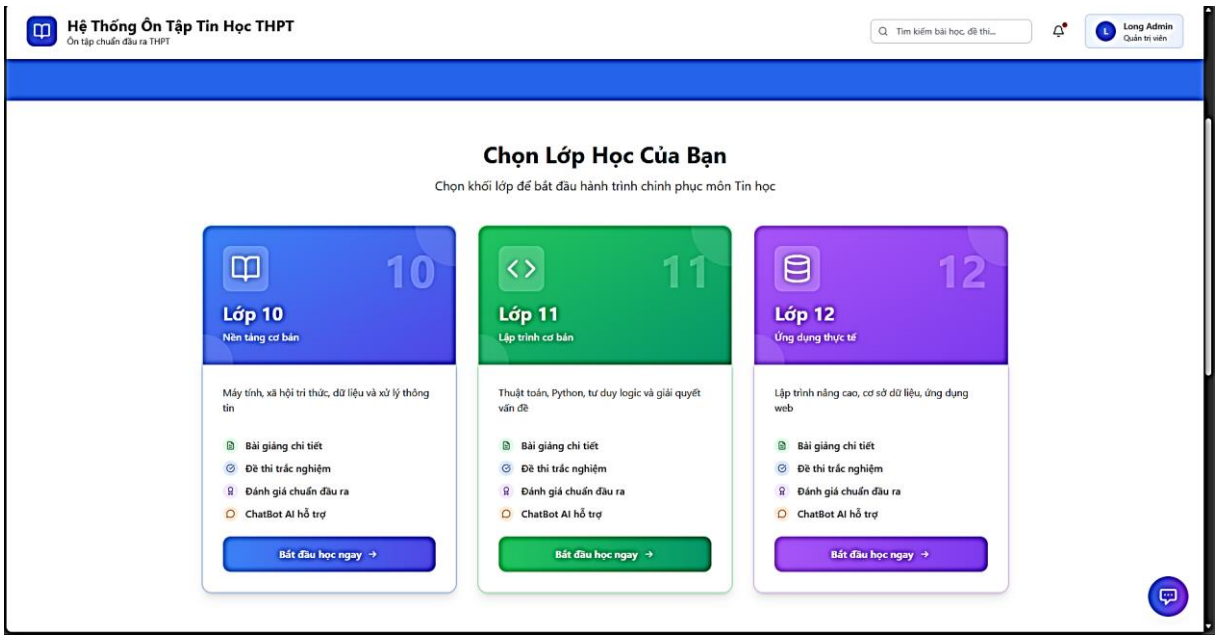

Hộp thư gửi mã OTP đến Gmail quên mật khẩu


Giao diện quên mật khẩu 3

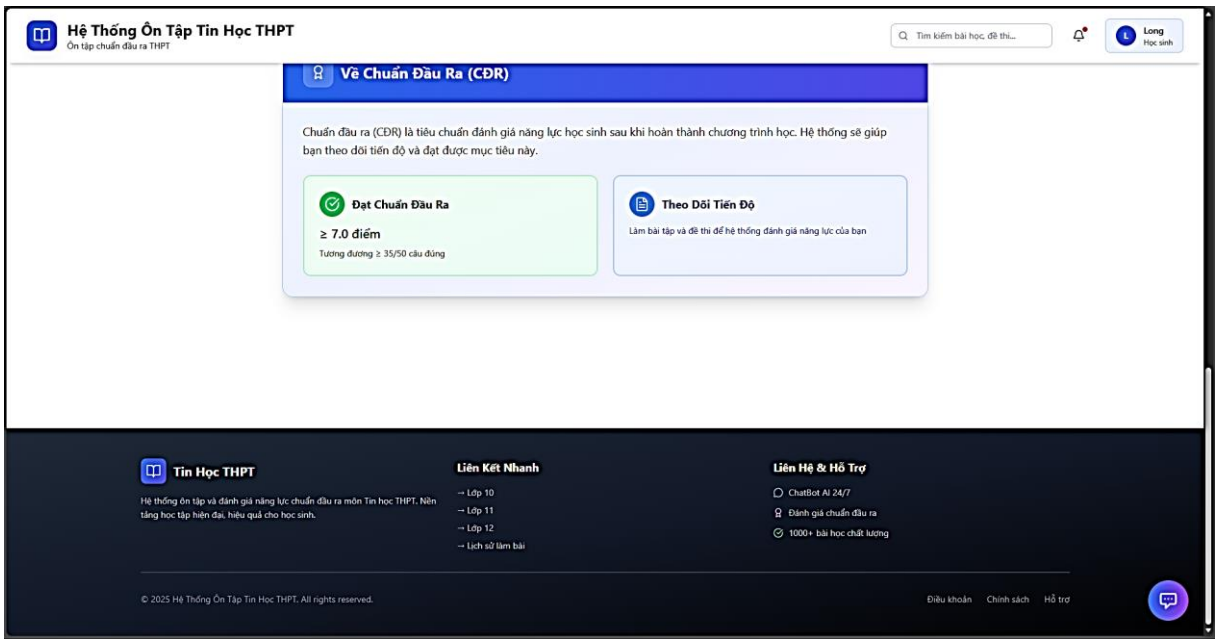
## II. Giao diện trang chủ



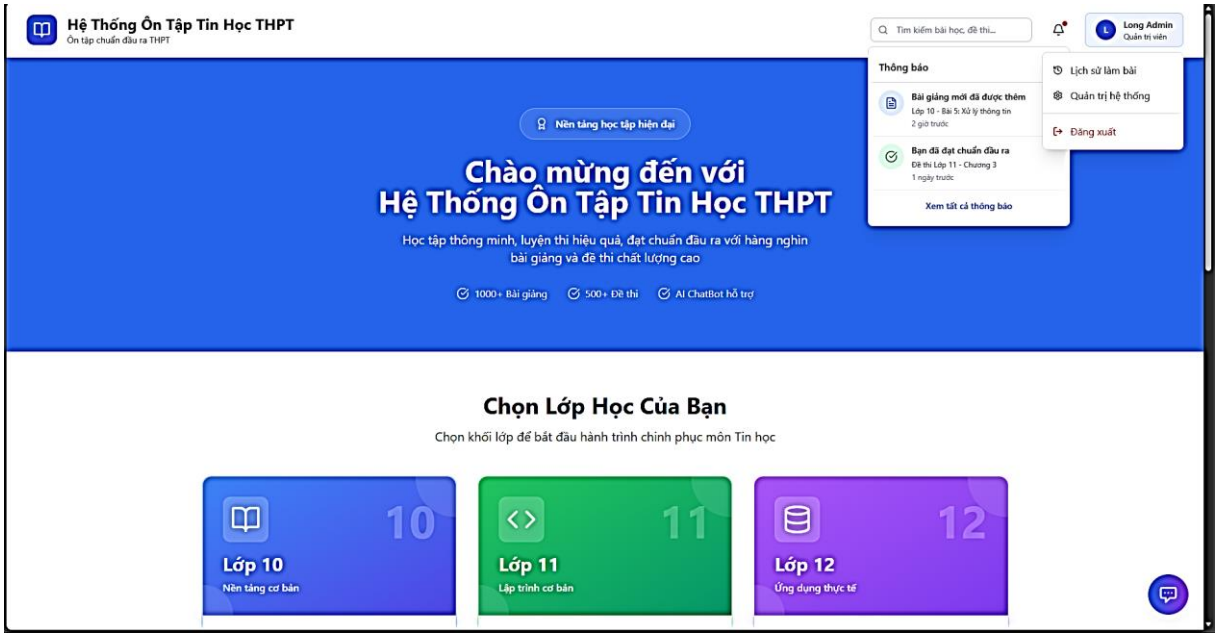

Giao diện trang chủ “Phần trên”



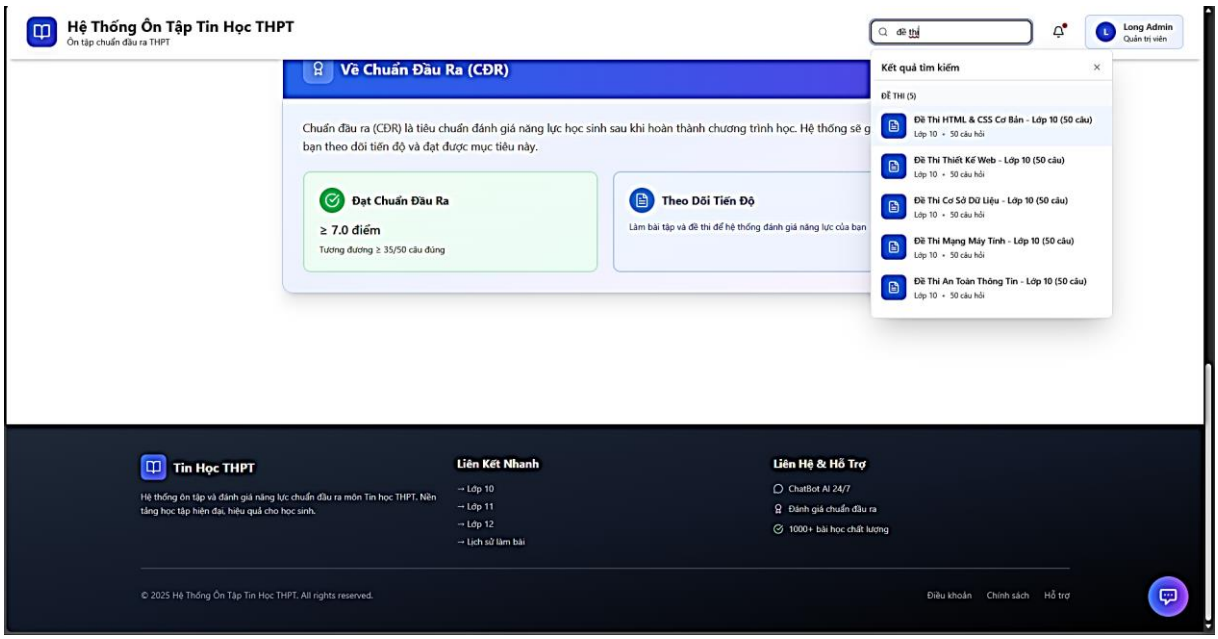

Giao diện trang chủ “Phần giữa”



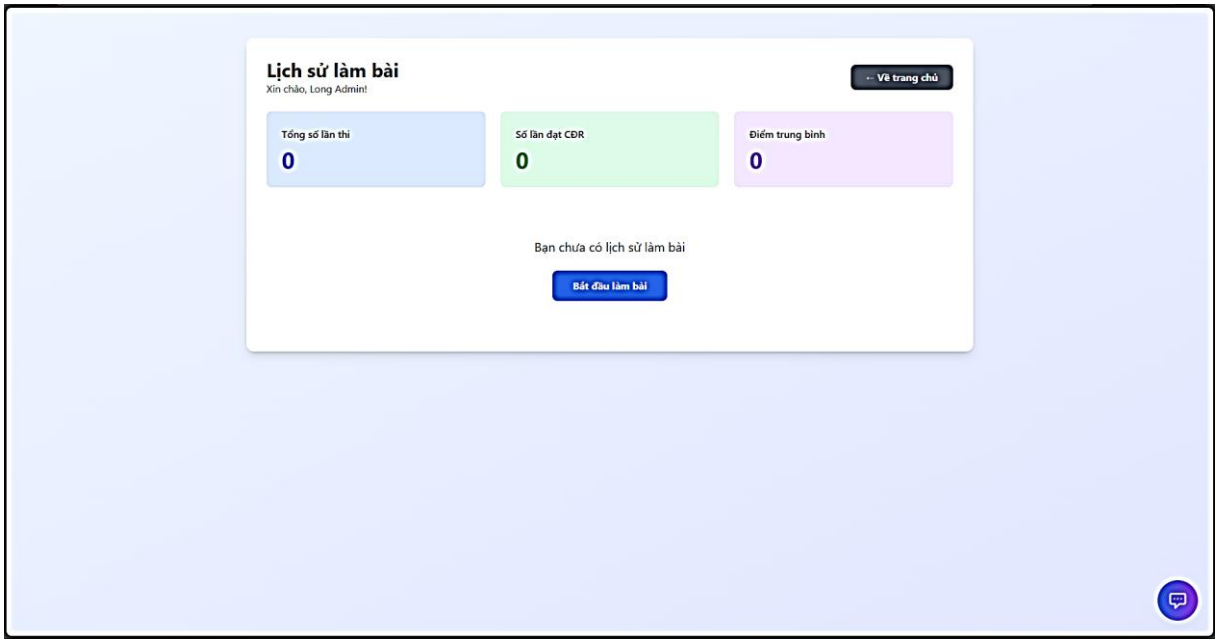

Giao diện trang chủ “Phần dưới cùng”



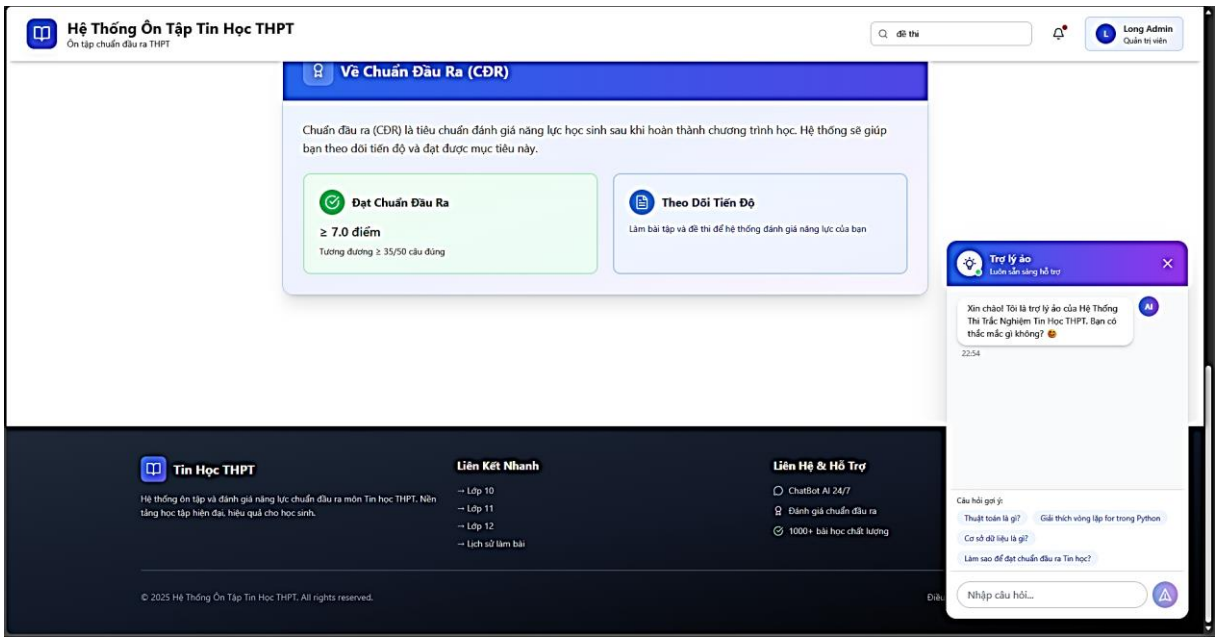

Chức năng của quản trị viên và chức năng thông báo



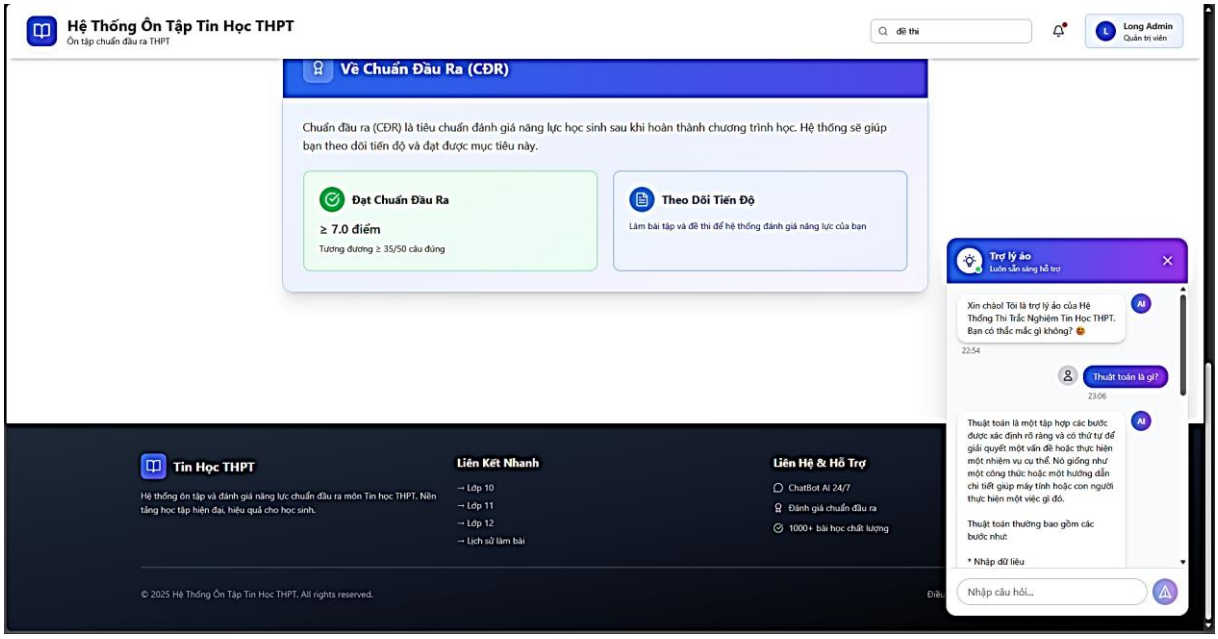

Chức năng tìm kiếm




Giao diện lịch sử làm bài

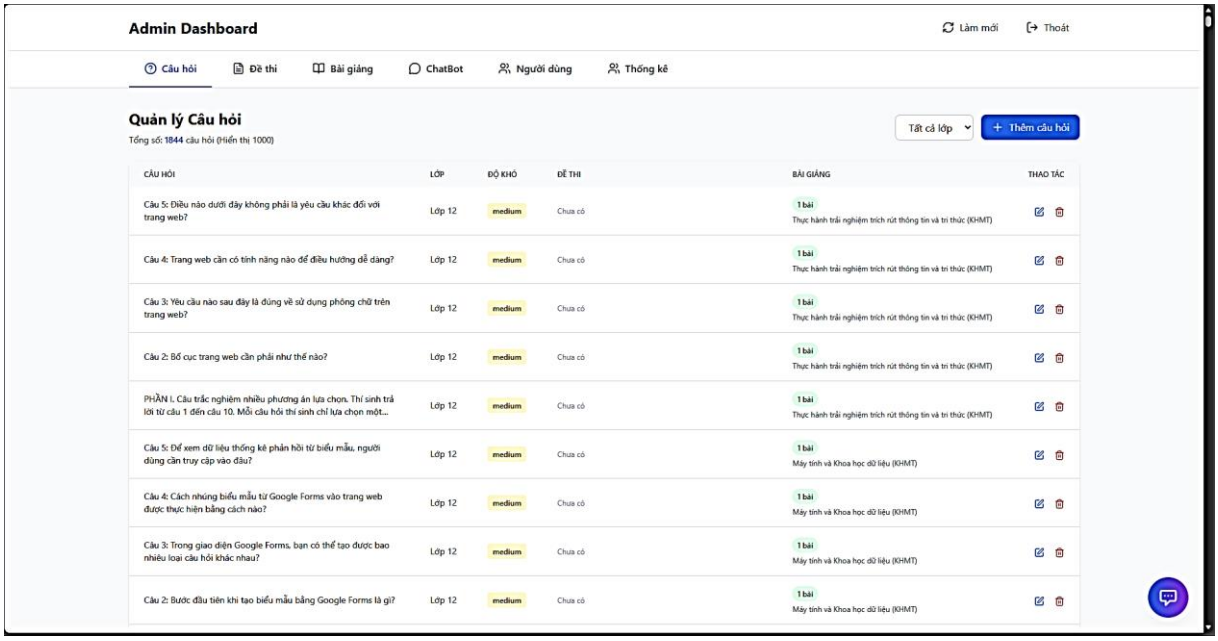



Giao diện trợ lý ảo

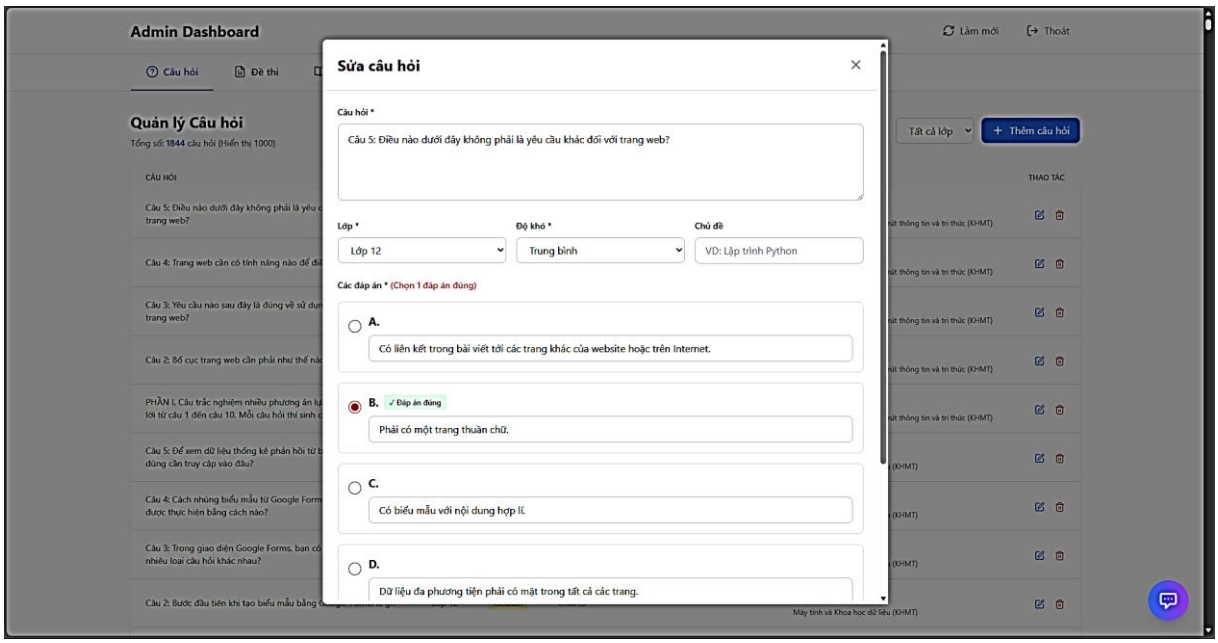



Test chức năng trợ lý ảo tích hợp Groq AI

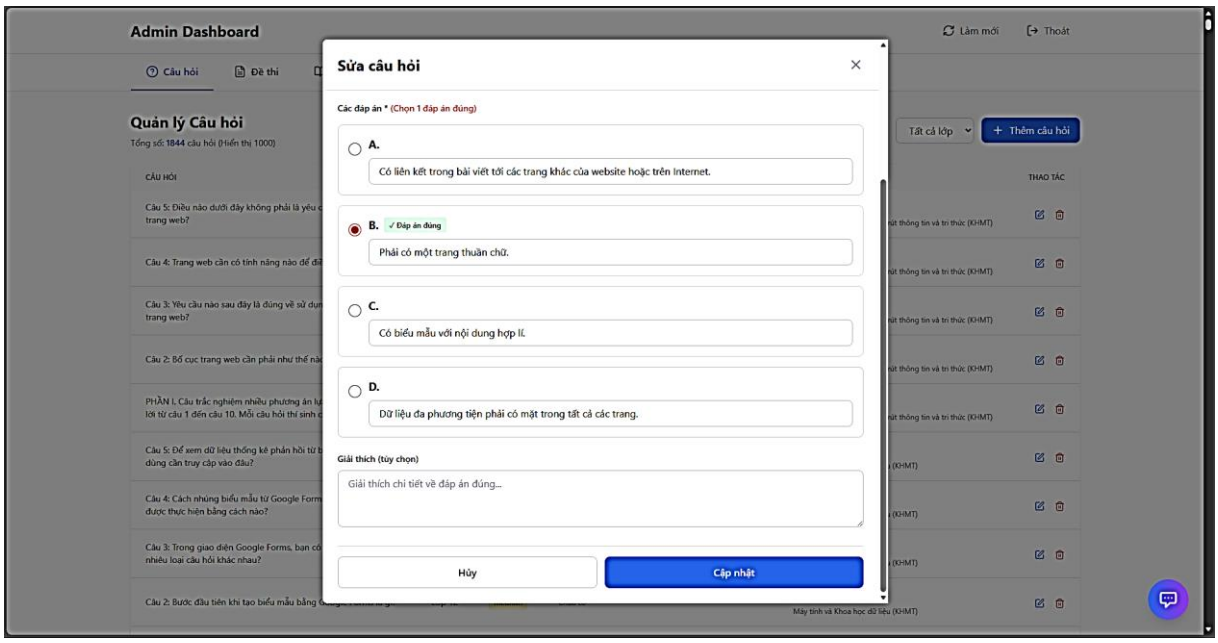
# III. Giao diện quản trị hệ thống



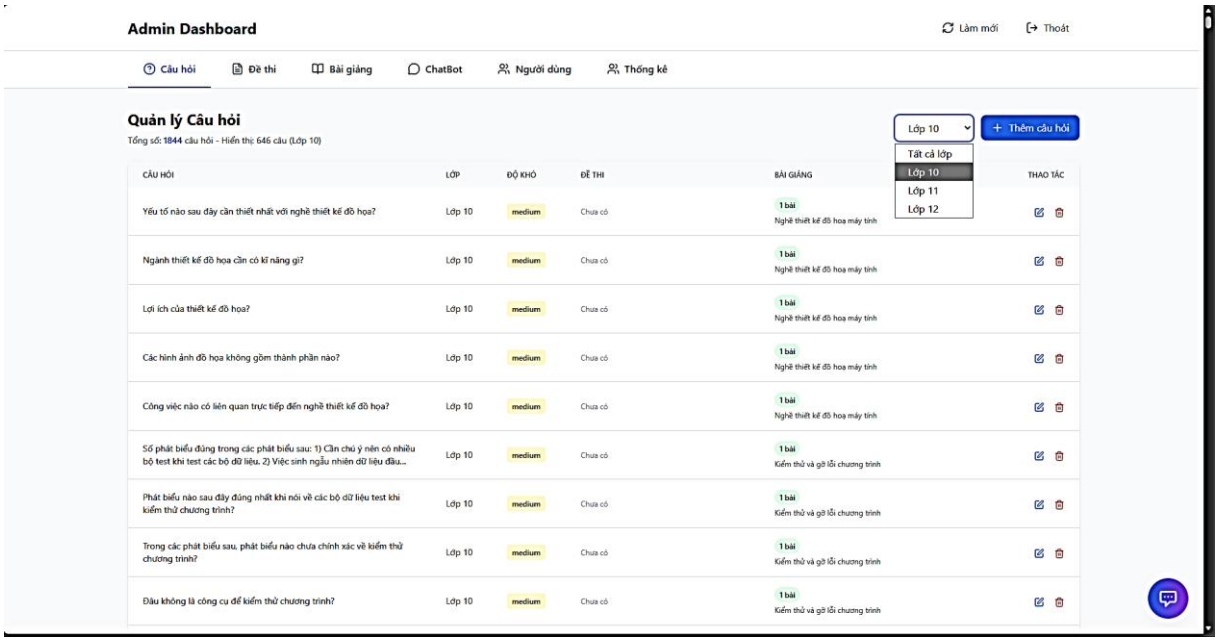

Giao diện quản lý câu hỏi



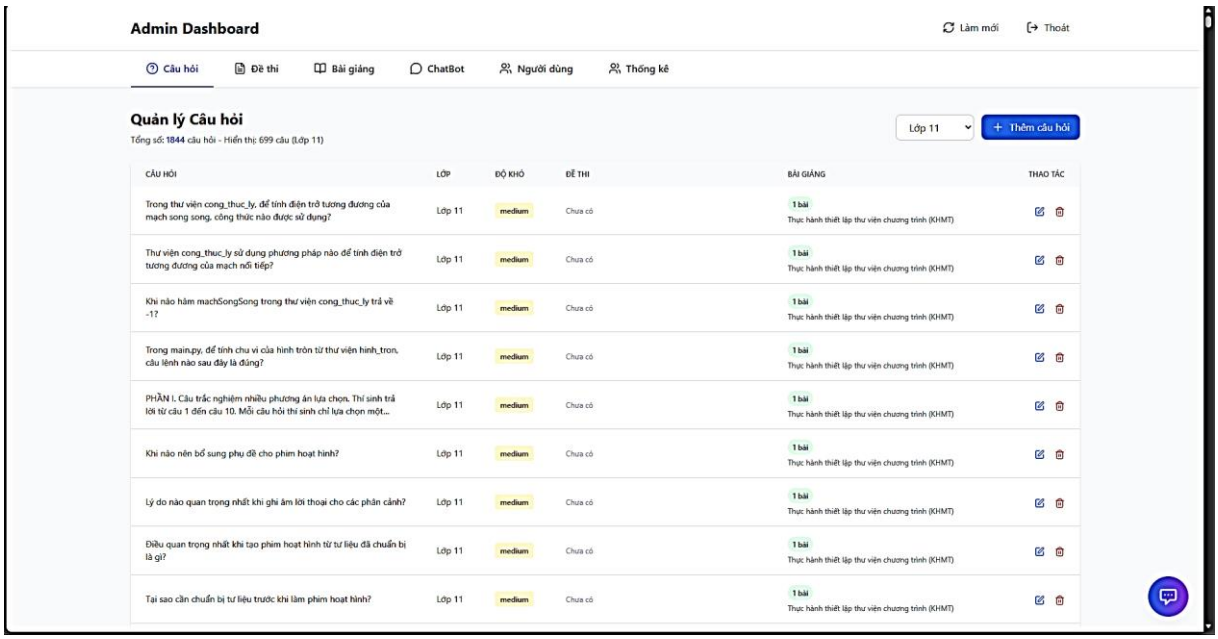

Chức năng sửa câu hỏi “Phần trên”



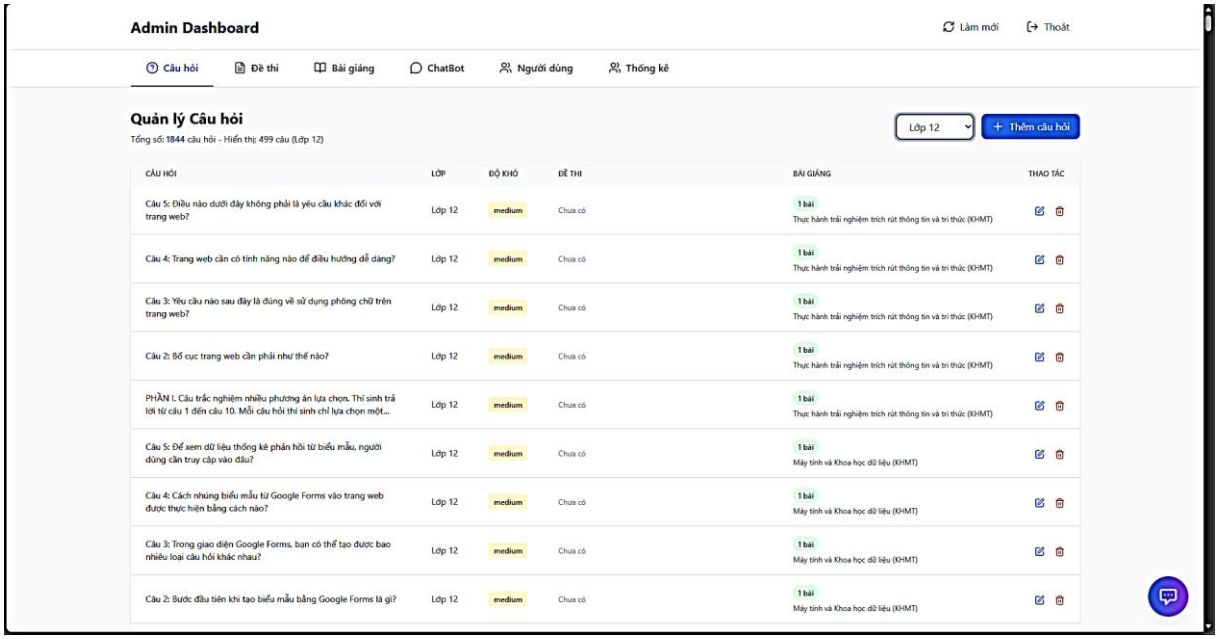

Chức năng sửa câu hỏi “Phần dưới”



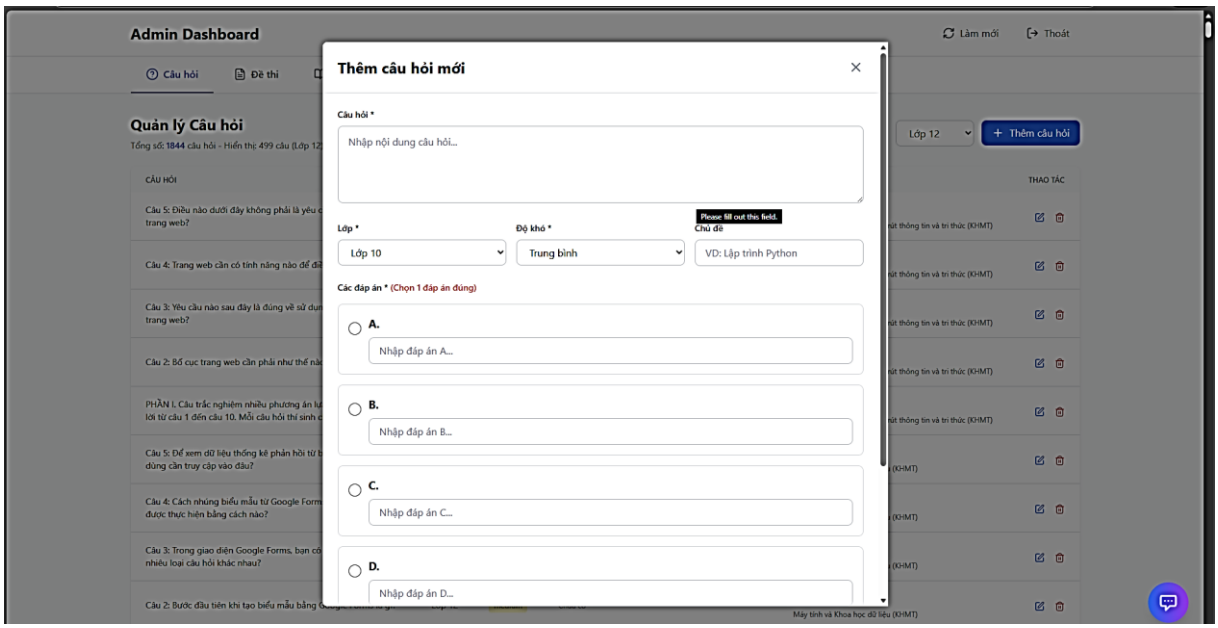

Giao diện quản lý câu hỏi lớp 10



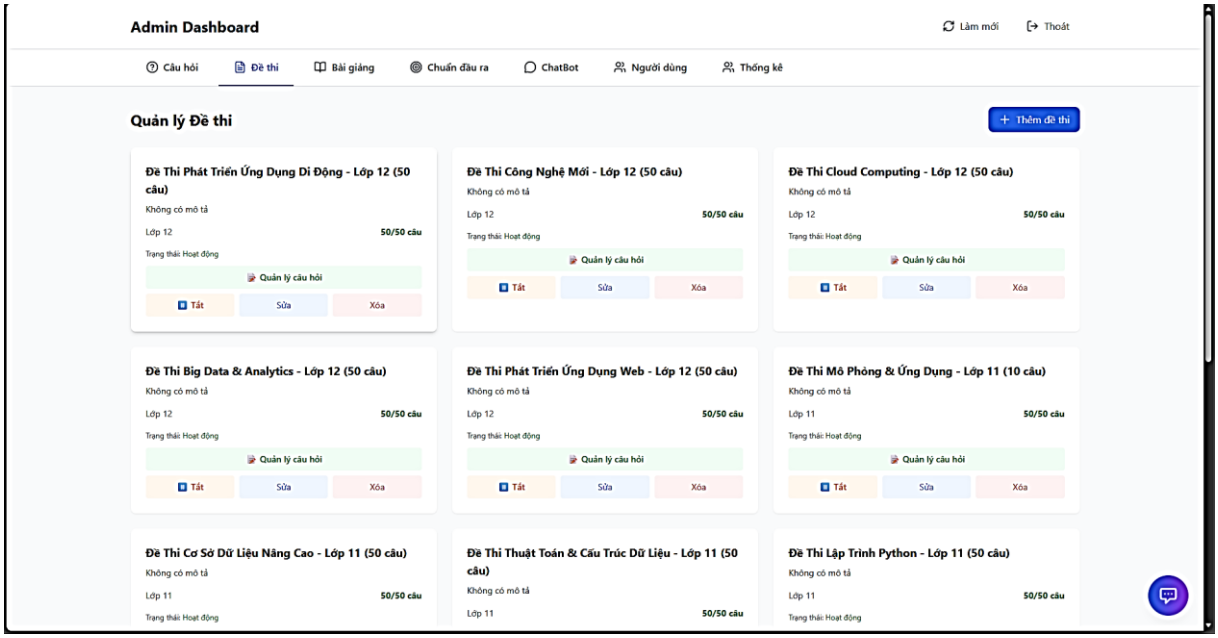

Giao diện quản lý câu hỏi lớp 11



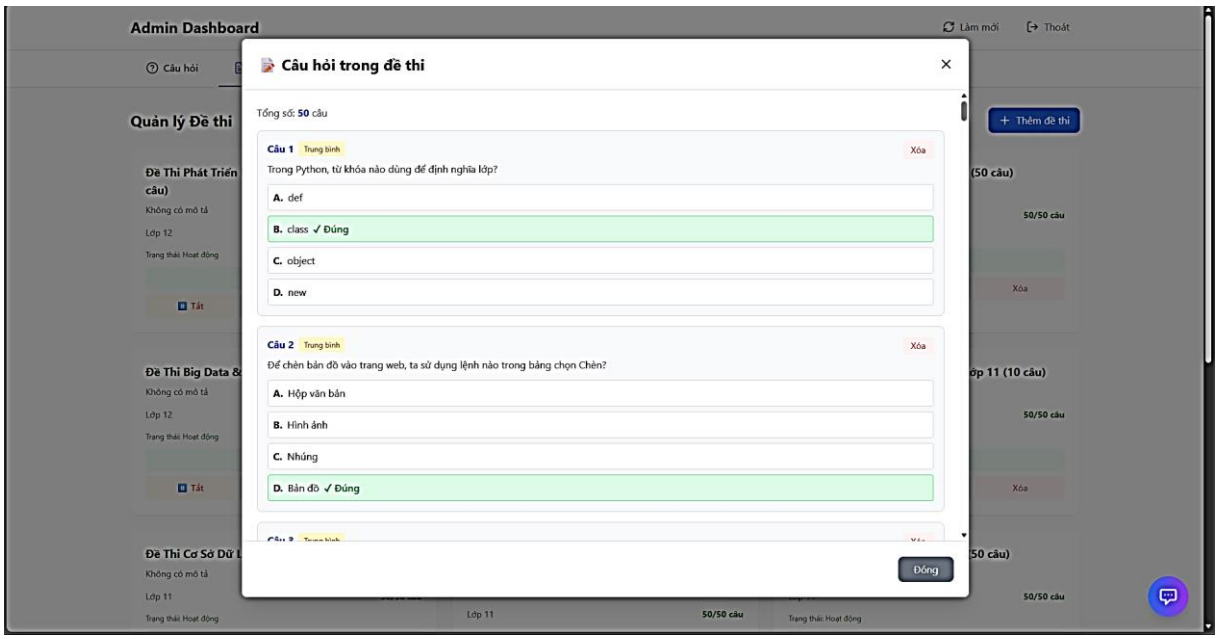

Giao diện quản lý câu hỏi lớp 12



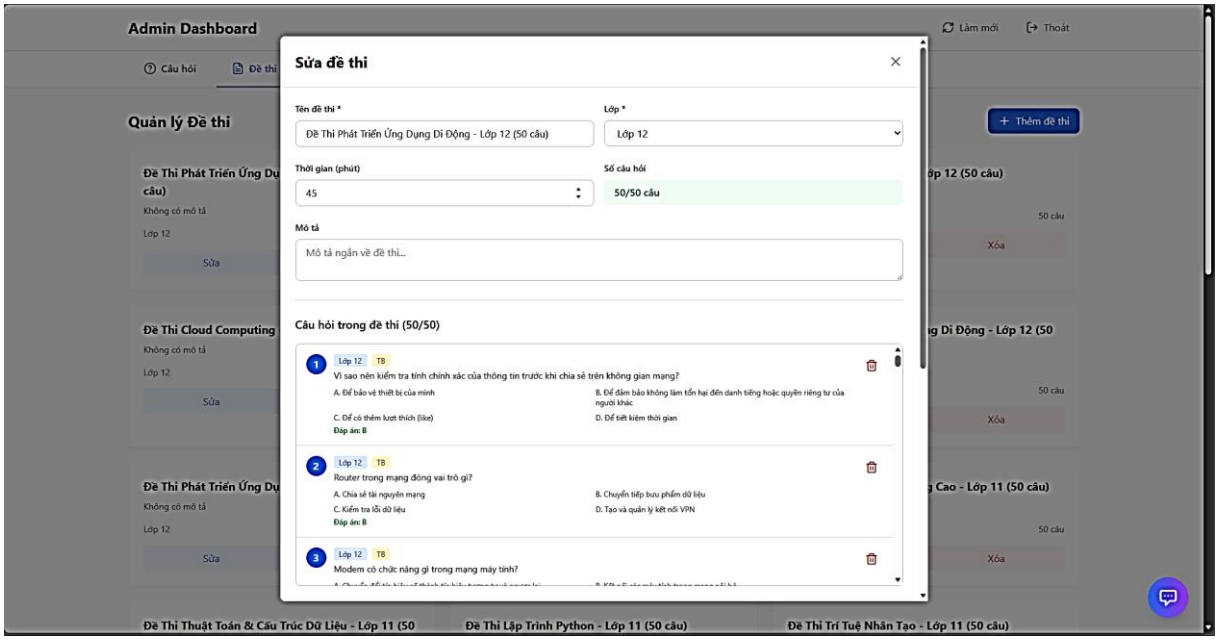

Chức năng thêm câu hỏi



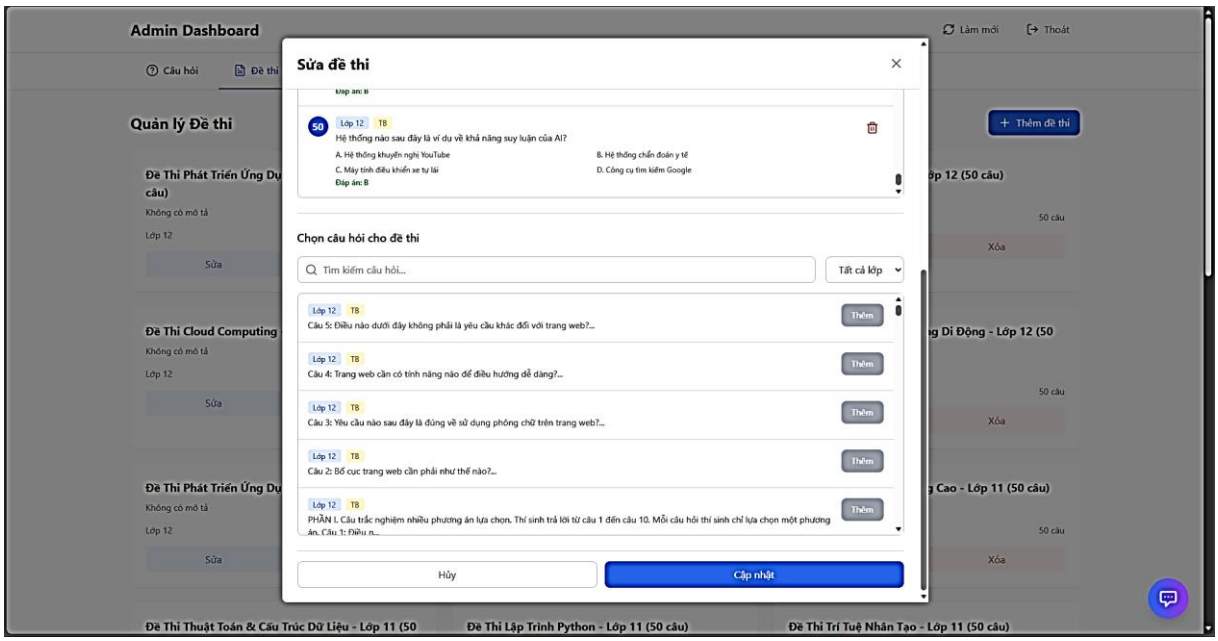

Giao diện quản lý đề thi



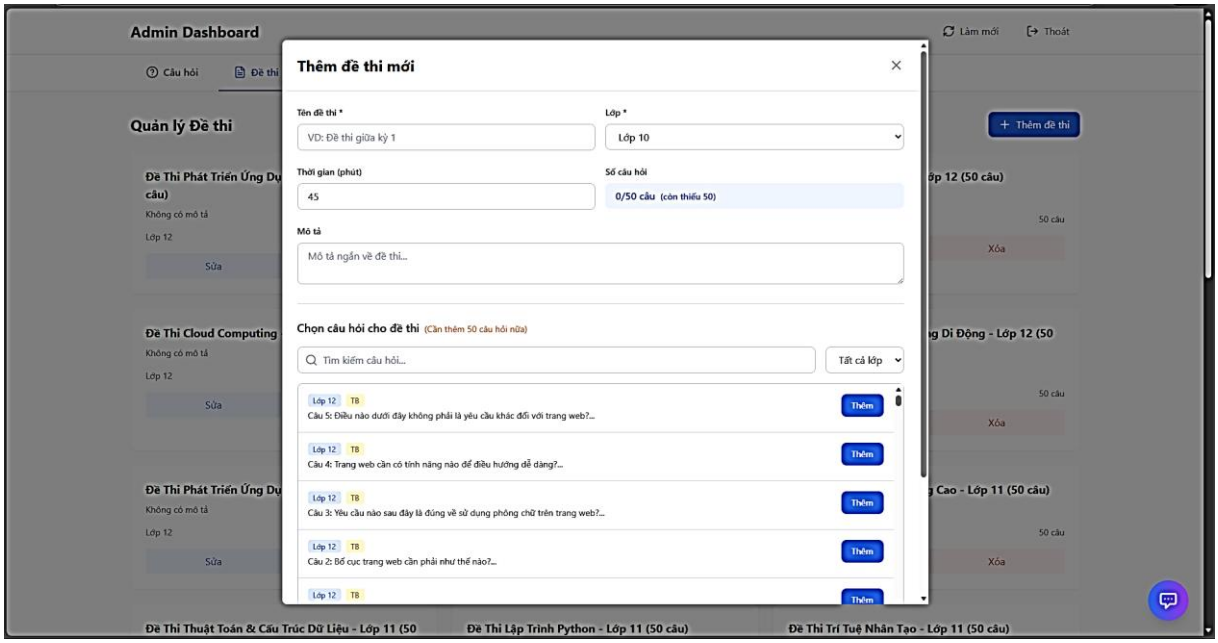

Chức năng quản lý câu hỏi của đề thi



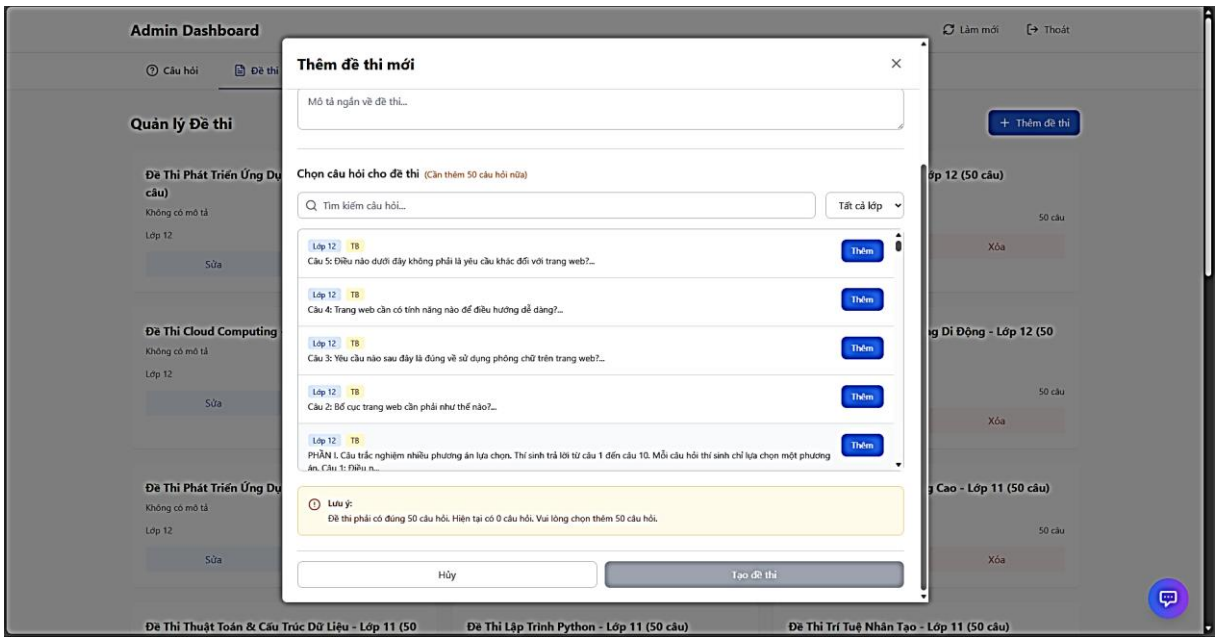

Chức năng sửa đề thi (hiển thị toàn bộ câu hỏi trong đề thi) “Phần trên”




Chức năng sửa đề thi (hiển thị toàn bộ câu hỏi trong đề thi) “Phần dưới”

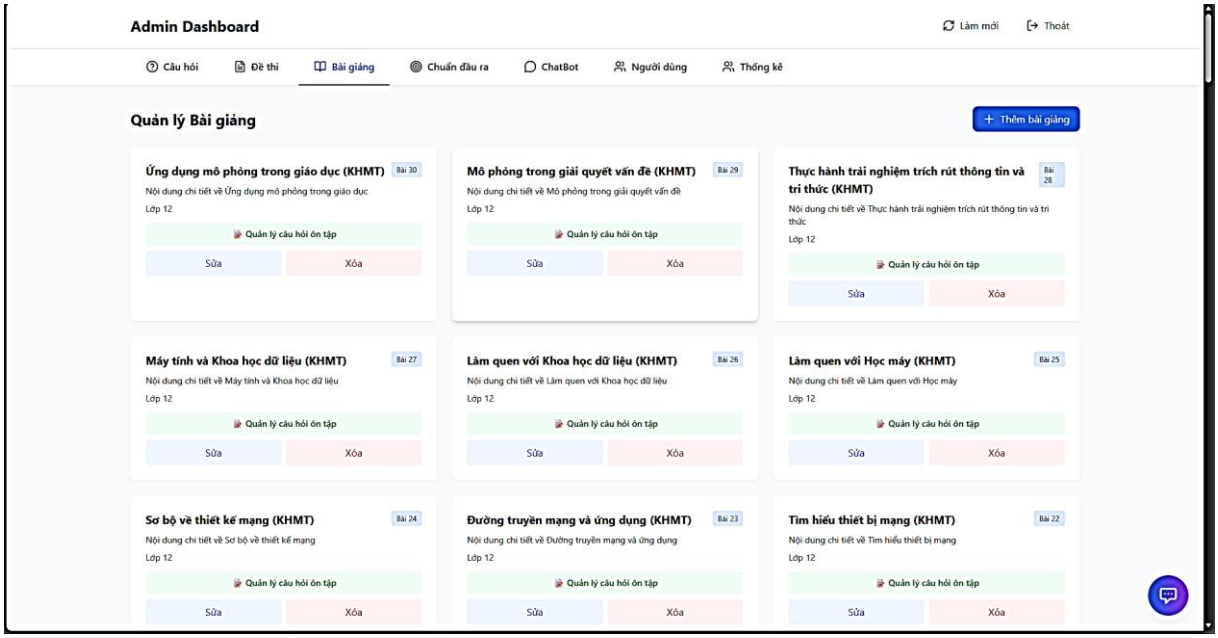



Chức năng thêm đề thi “Phần trên”

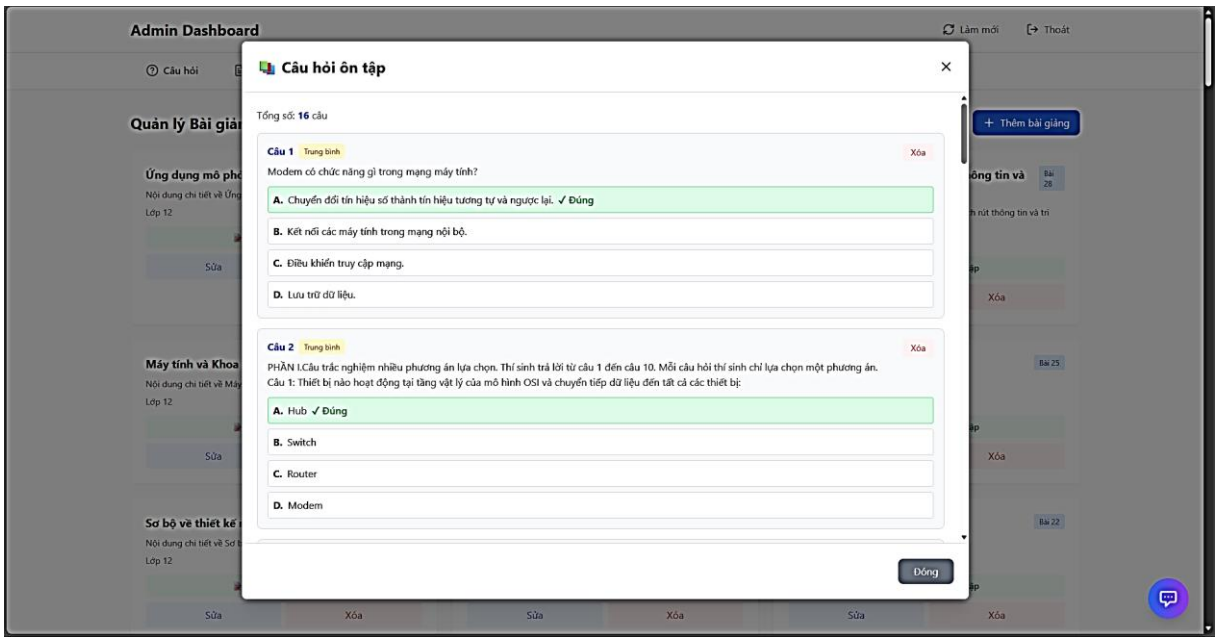



Chức năng thêm đề thi “Phần dưới”

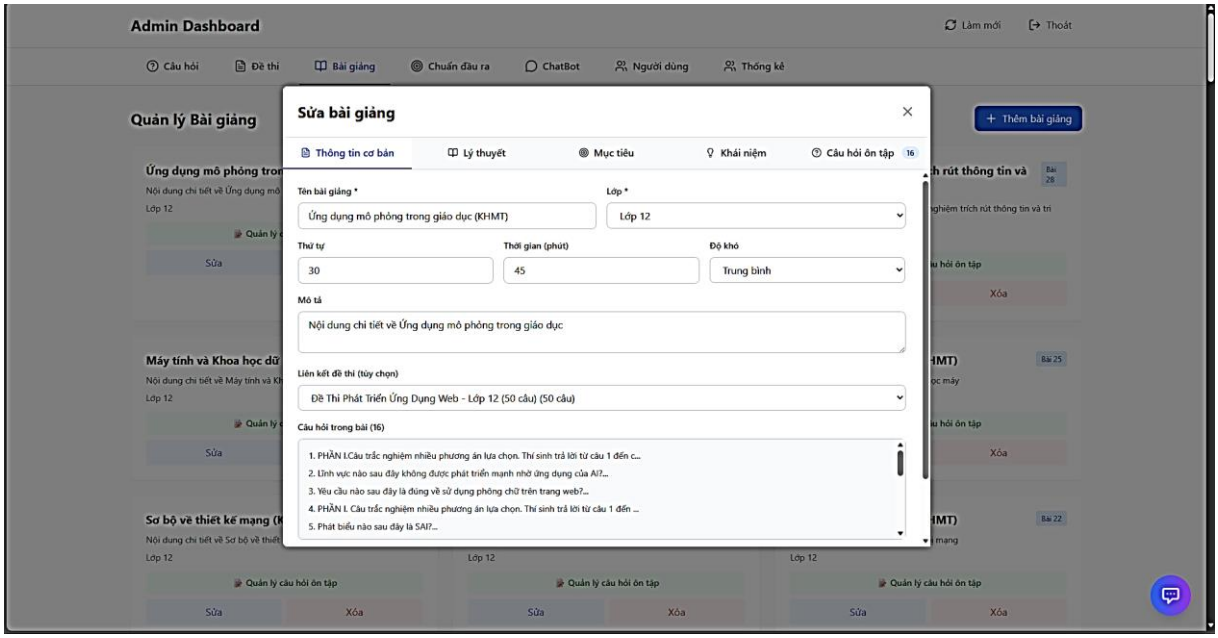
Lưu ý: chỉ cho tạo khi đủ 50 câu hỏi trong đề



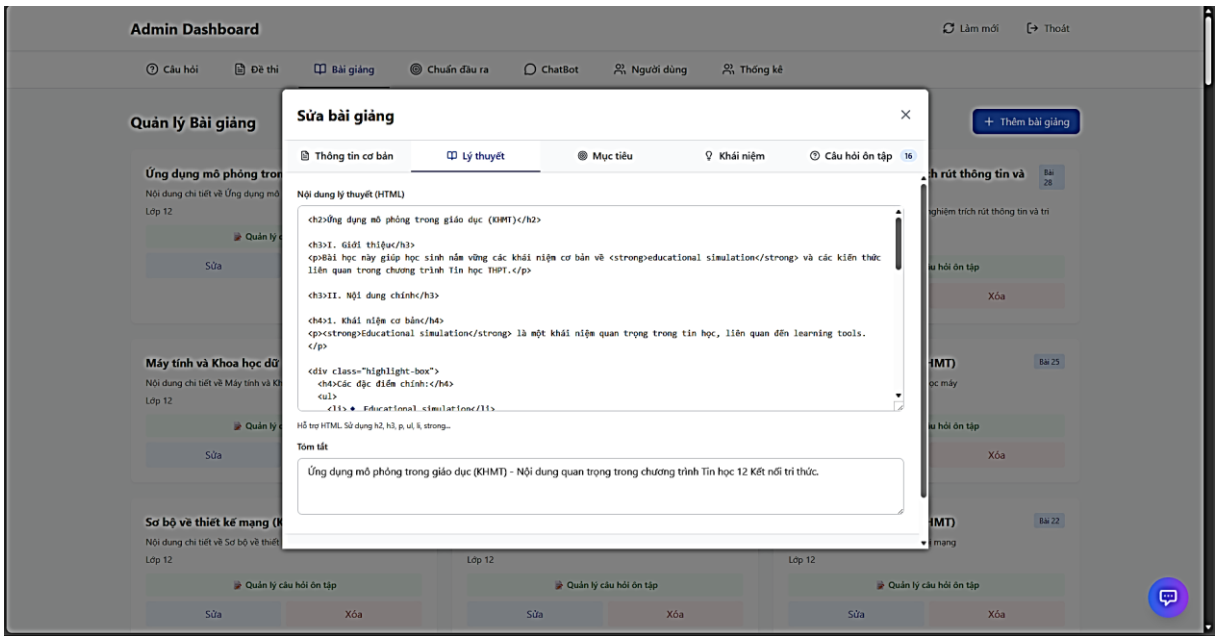

Chức năng quản lý bài giảng



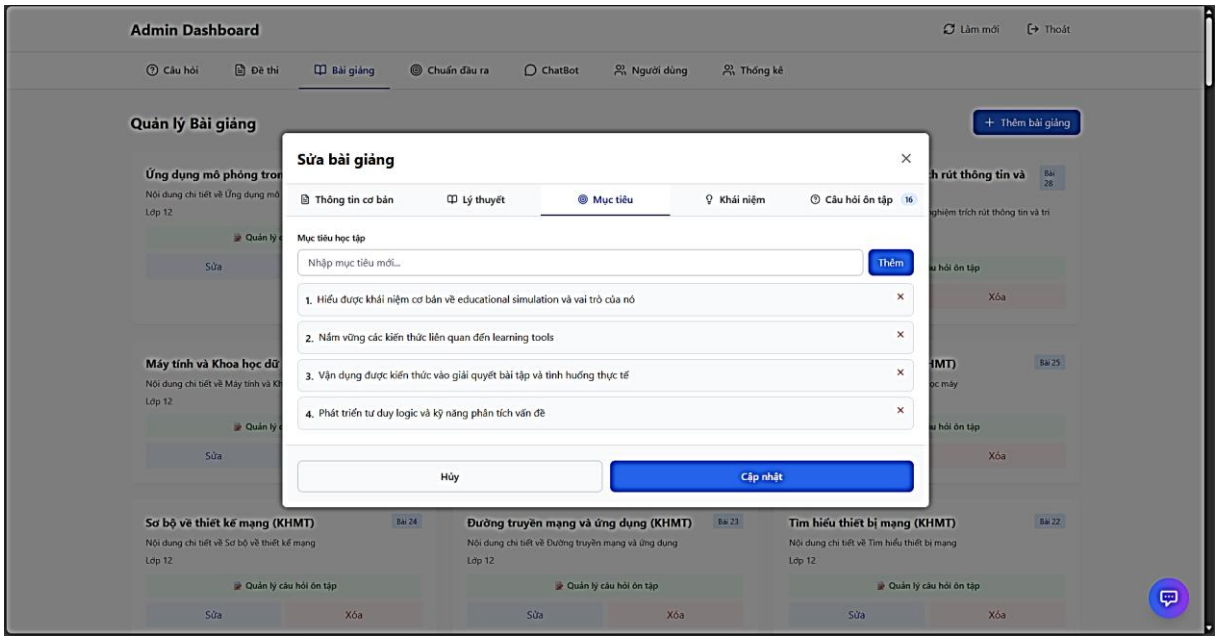

Chức năng quản lý câu hỏi ôn tập của bài giảng



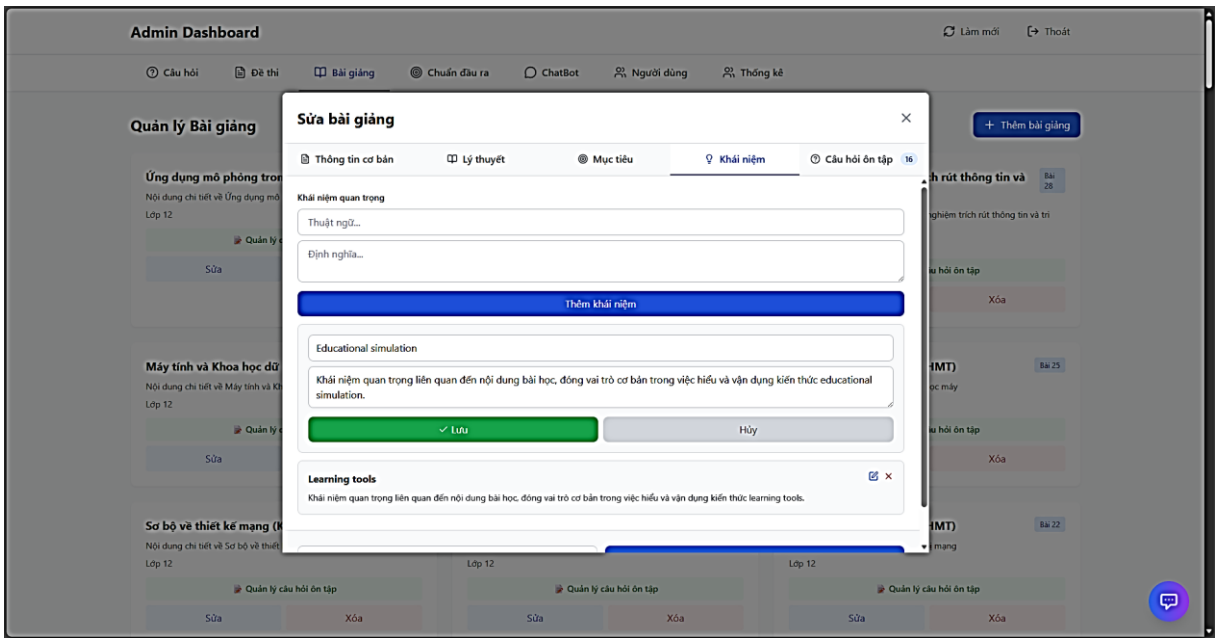

Chức năng sửa bài giảng “phần thông tin cơ bản”



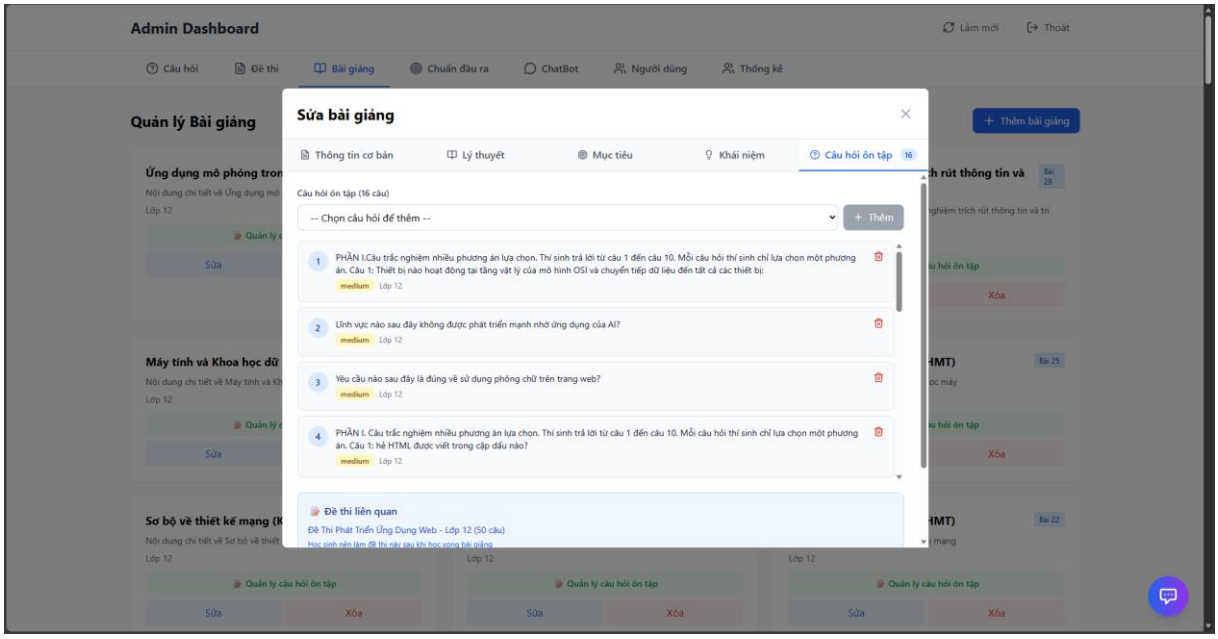

Chức năng sửa bài giảng “phần lý thuyết”



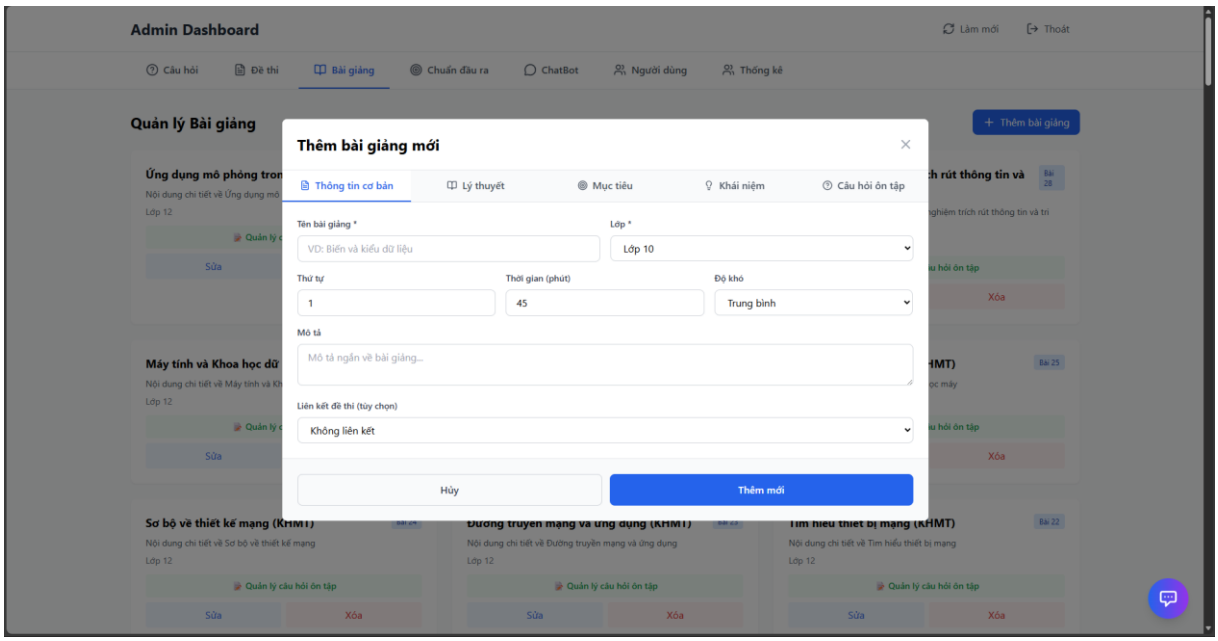

Chức năng sửa bài giảng “phần mục tiêu”



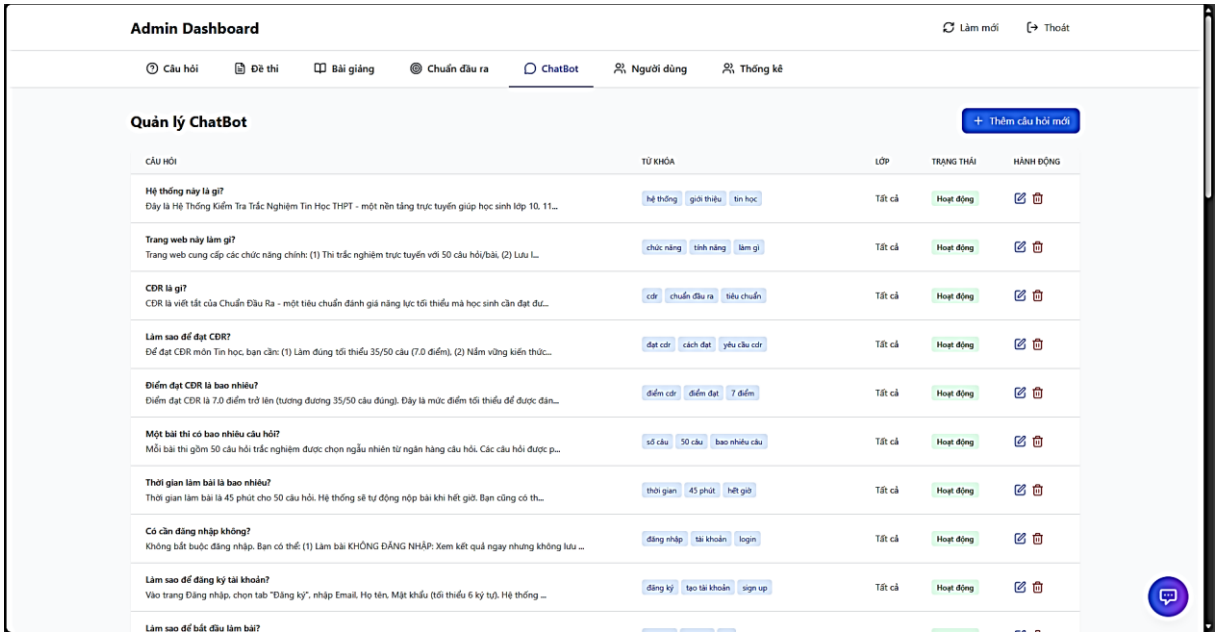

Chức năng sửa bài giảng “phần khái niệm”



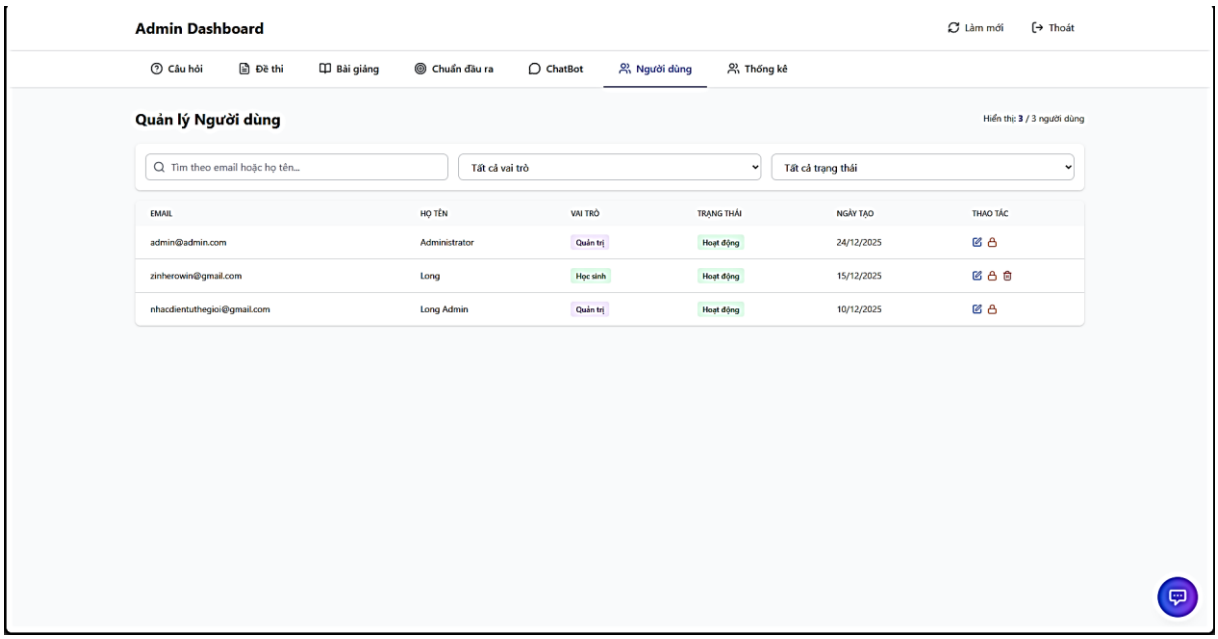

Chức năng sửa bài giảng “phần câu hỏi ôn tập”



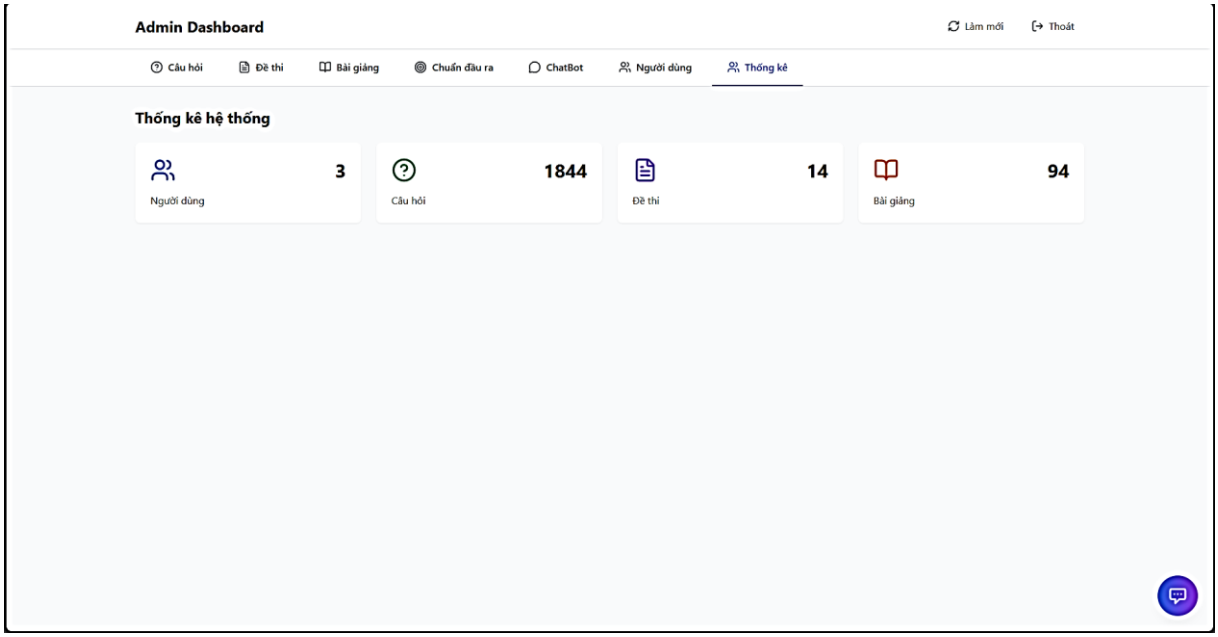

Giao diện thêm bài giảng



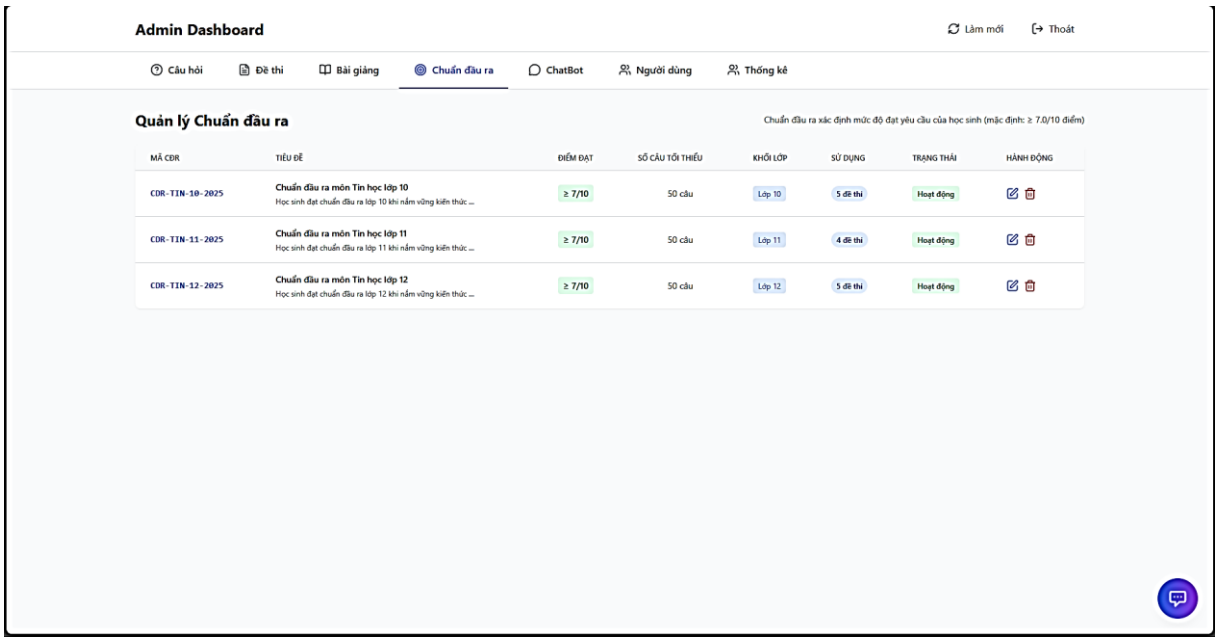

Giao diện quản lý chat bot (AI lấy thông tin ở đây)




Giao diện quản lý người dùng



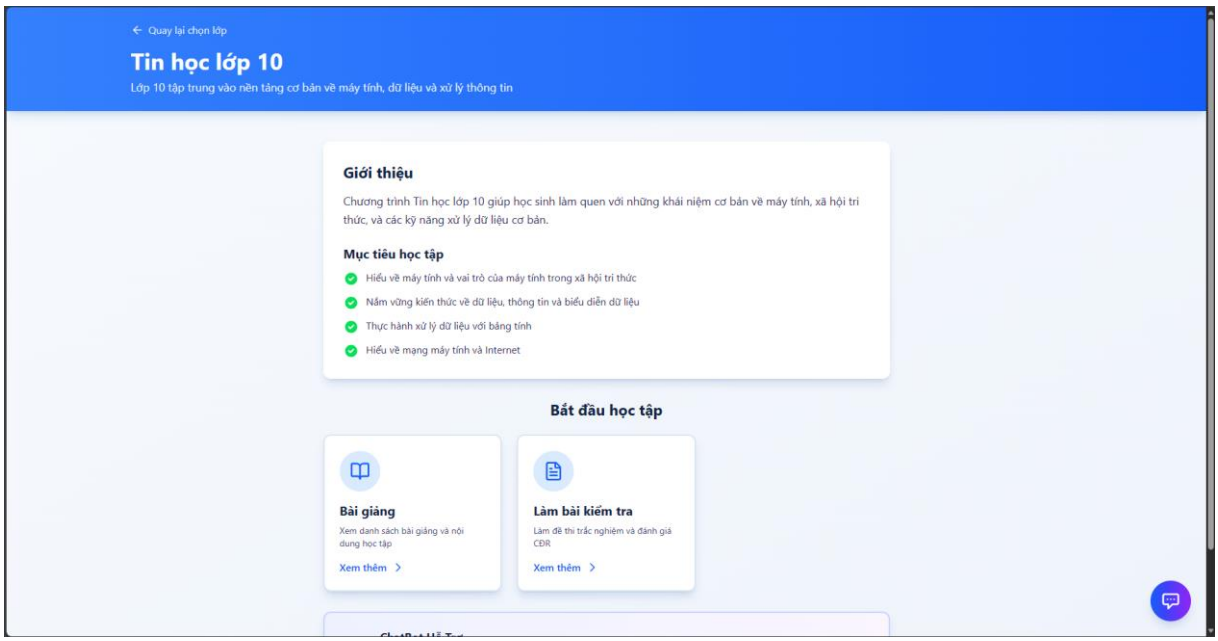

Giao diện thống kê hệ thống



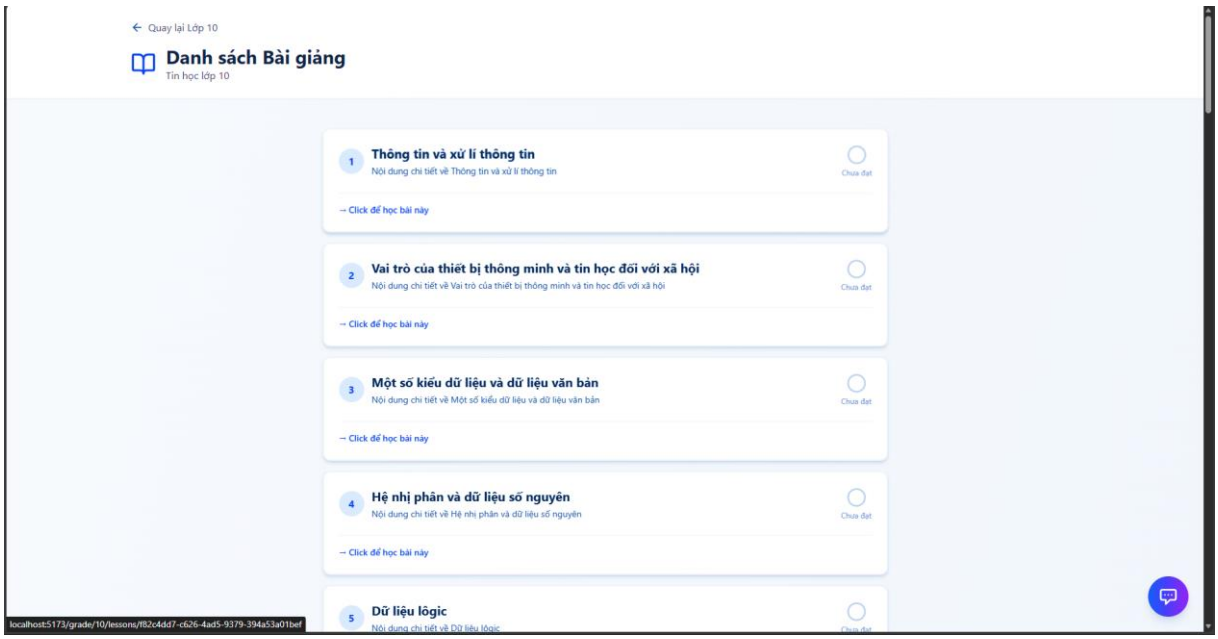

Giao diện quản lý chuẩn đầu ra

# IV.Giao diện lớp

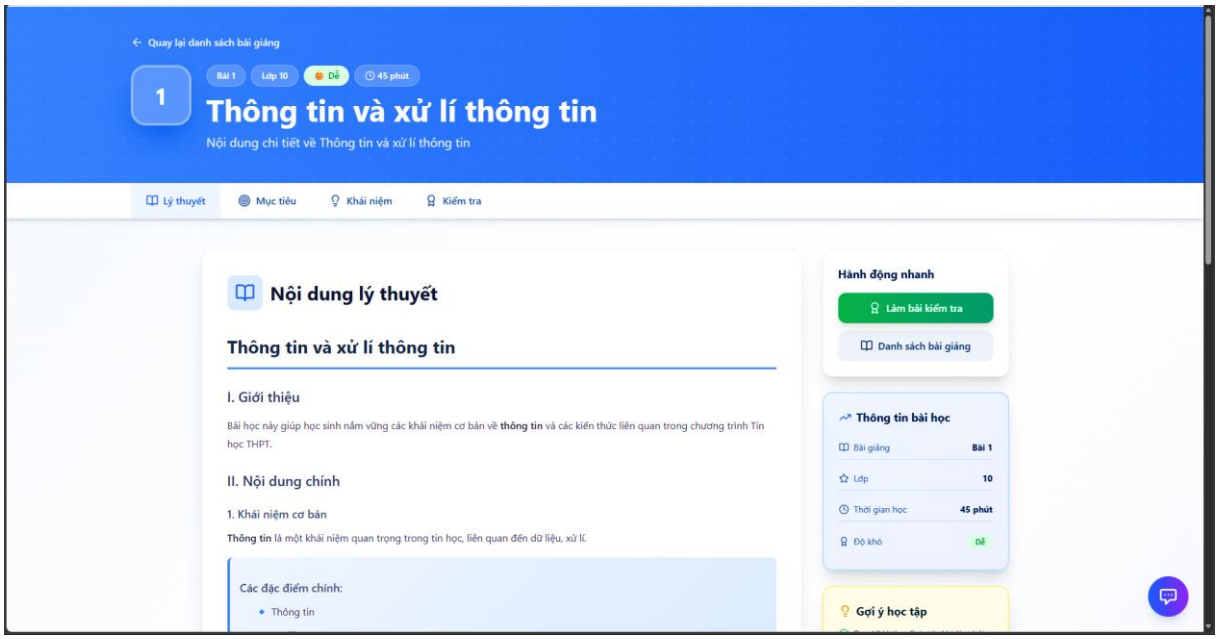
*Lớp 10:*



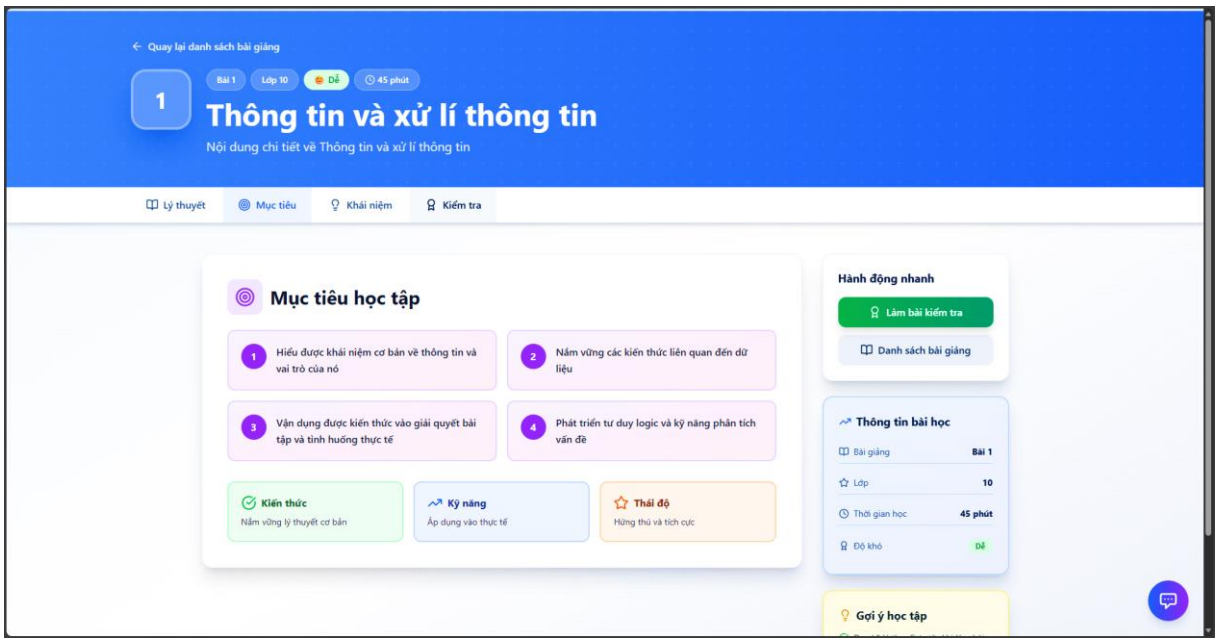

Giao diện chính



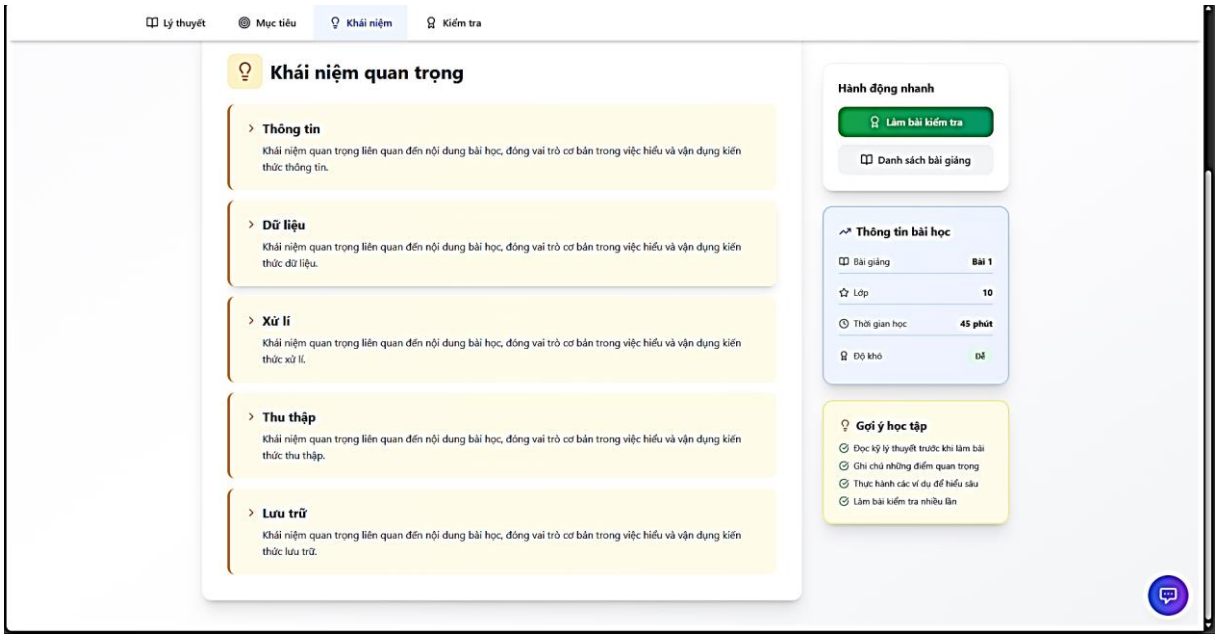

Giao diện bài giảng của lớp 10



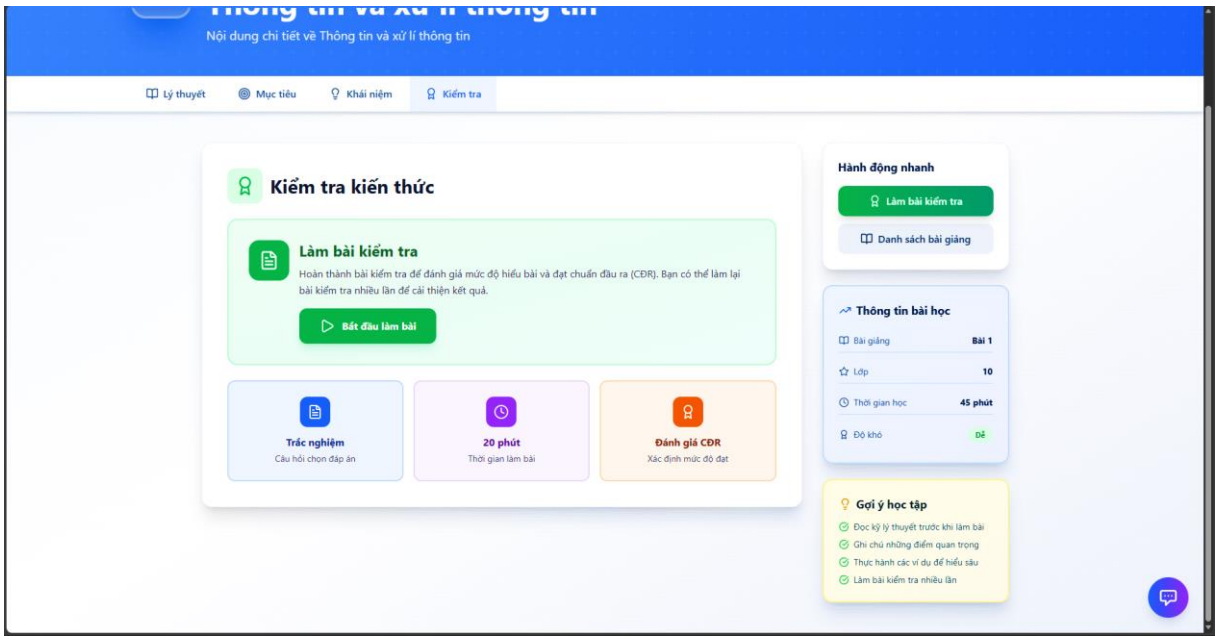

Giao diện trong 1 bài giảng lớp 10 “Phần lý thuyết”



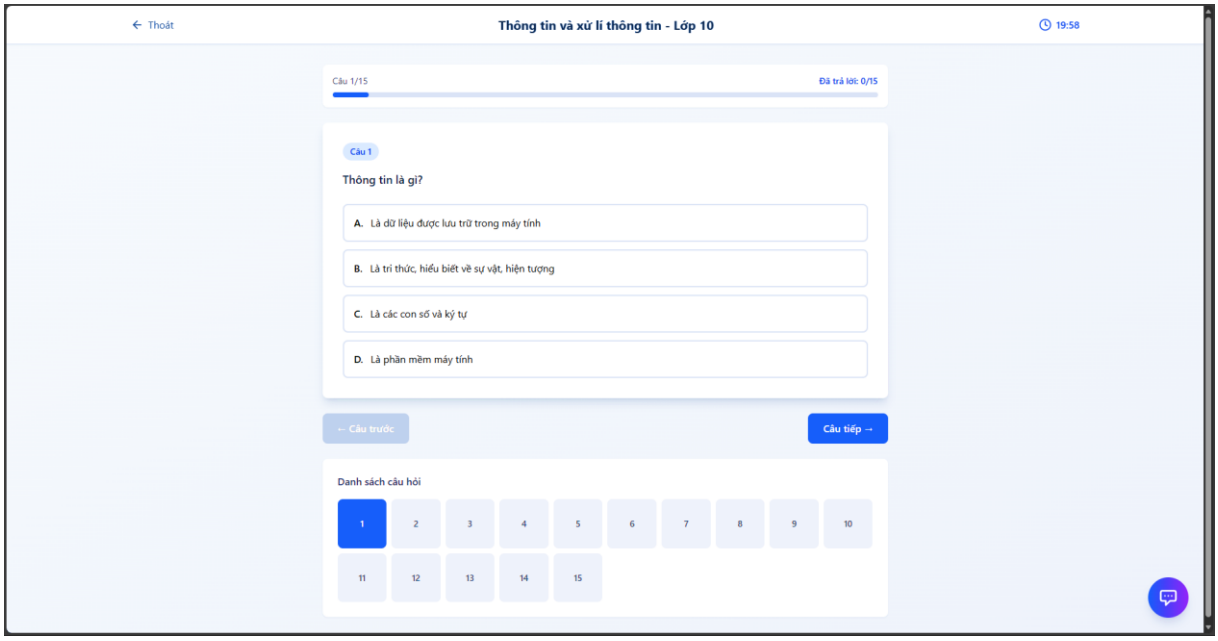

Giao diện trong 1 bài giảng lớp 10 “Phần mục tiêu”



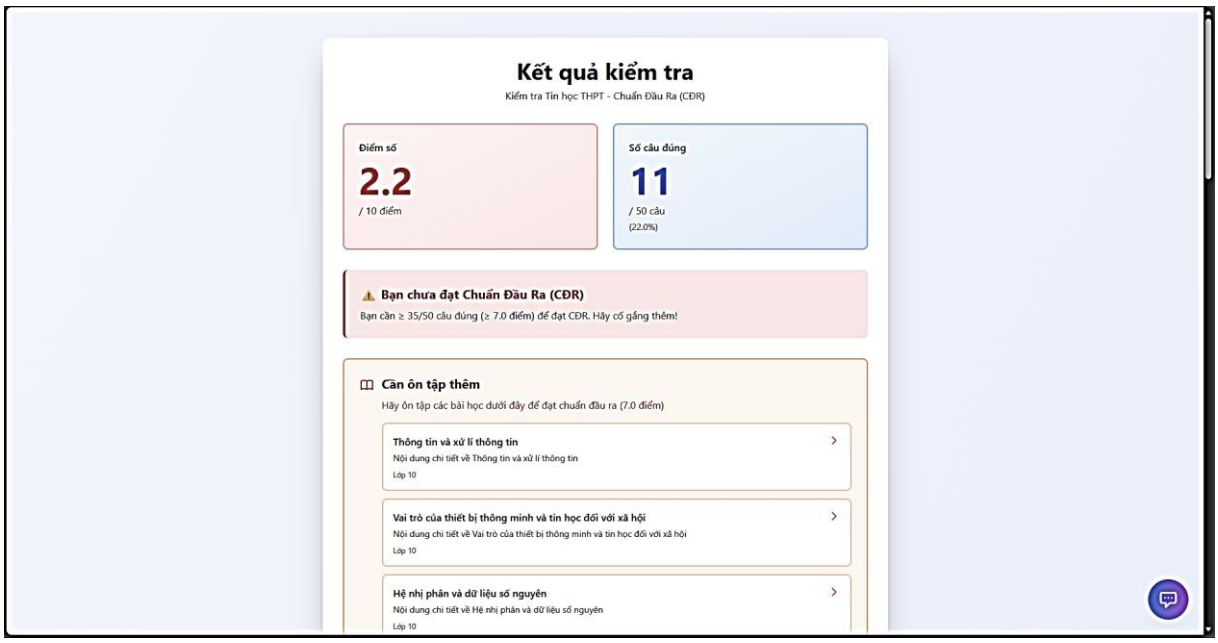

Giao diện trong 1 bài giảng lớp 10 “Phần khái niệm”



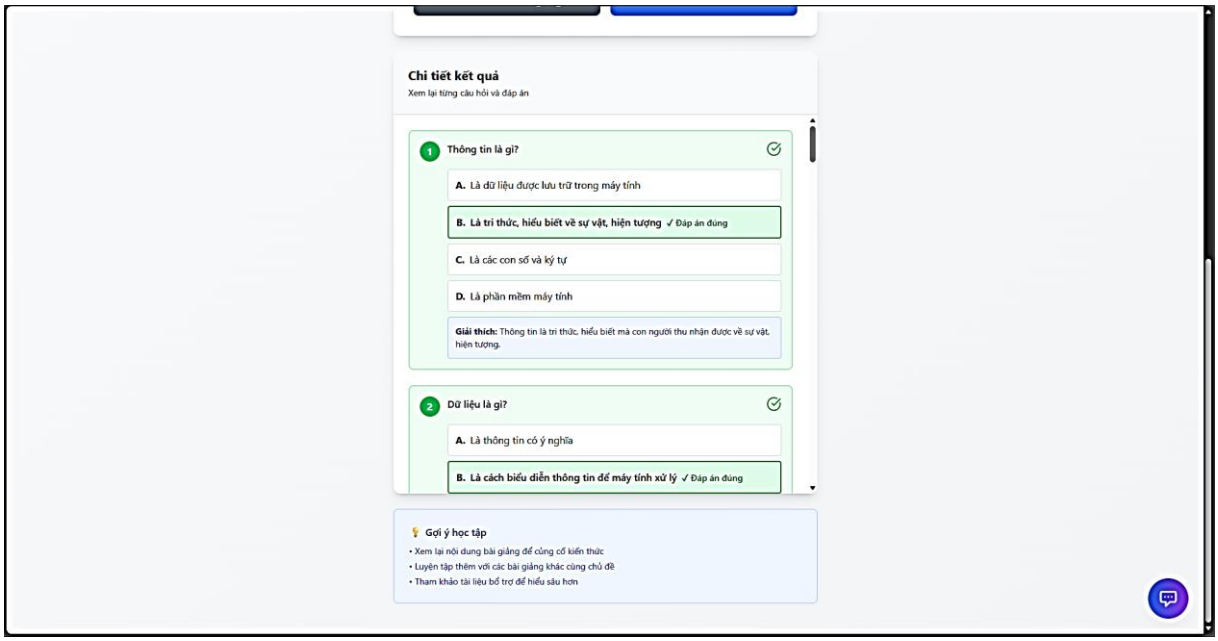

Giao diện trong 1 bài giảng lớp 10 “Phần kiểm tra”



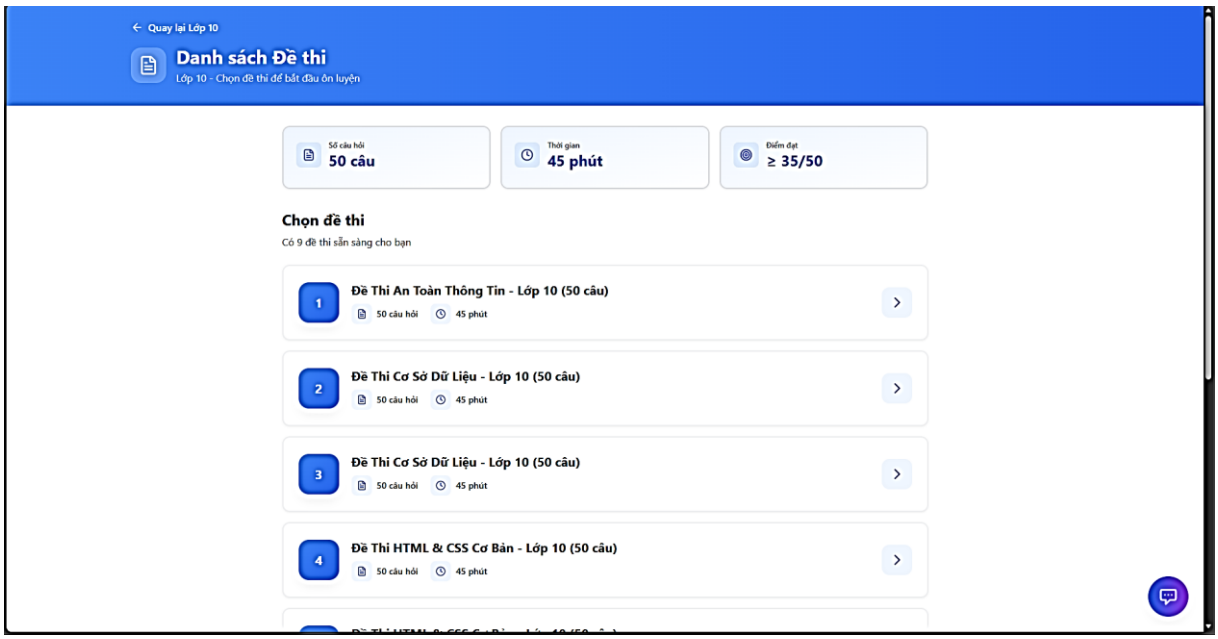

Giao diện ôn tập trong phần bài giảng



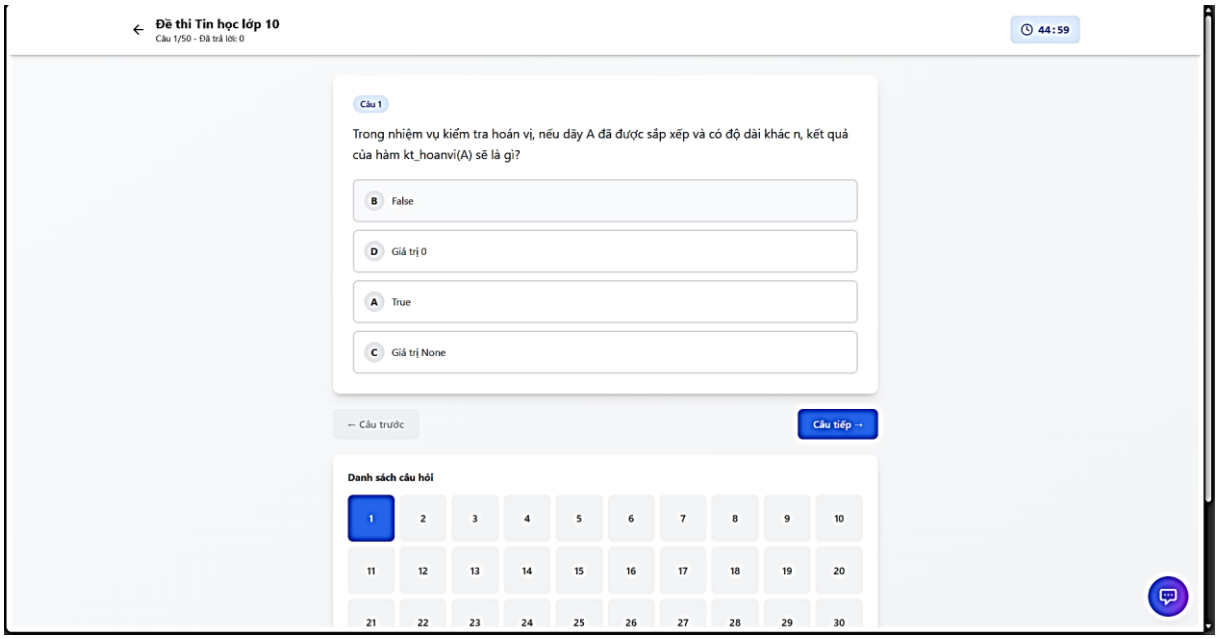

Giao diện chấm điểm và đánh giá chuẩn đầu ra , gợi ý ôn tập



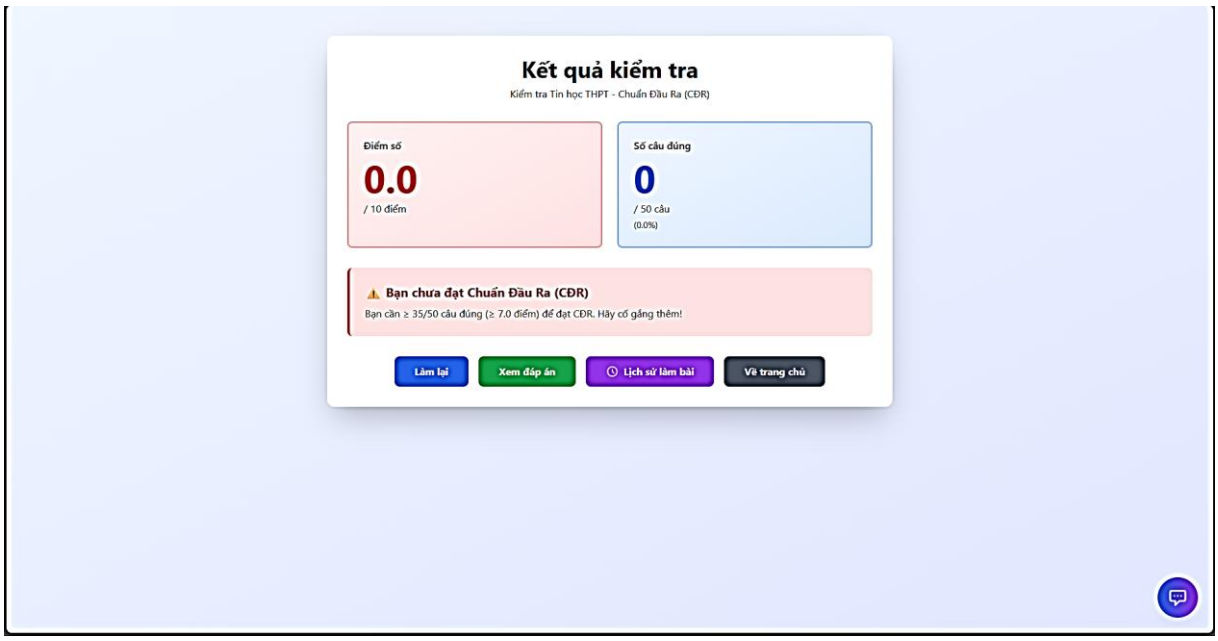

Hiển thị câu trả lời và đáp án để học sinh ôn tập



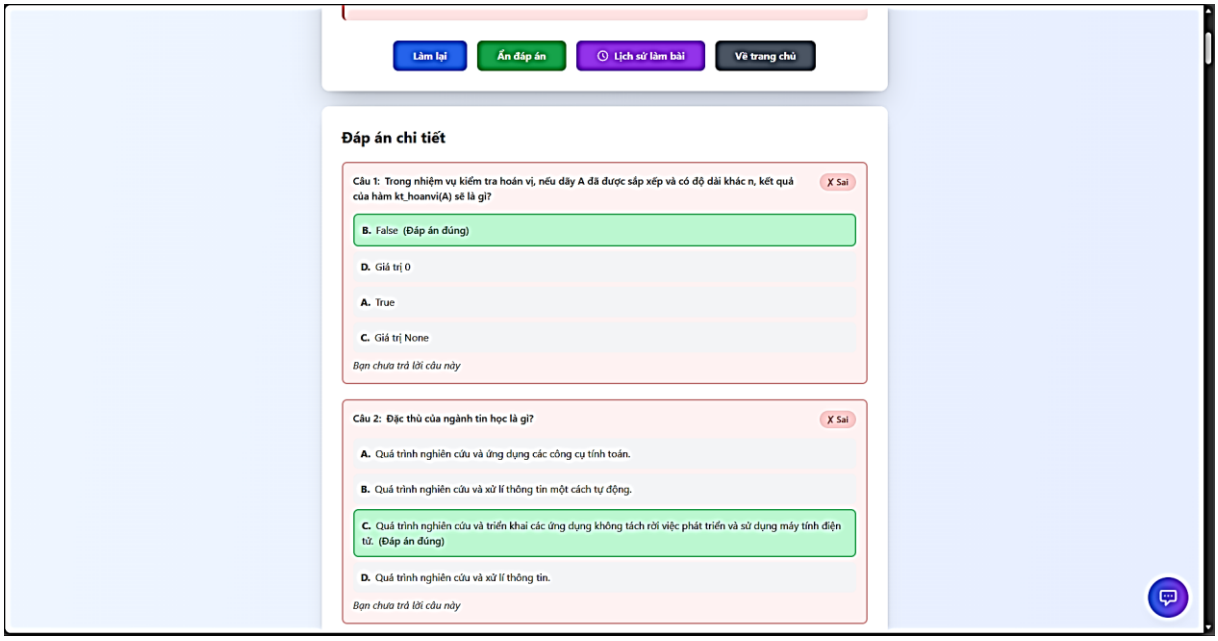

Giao diện danh sách đề thi



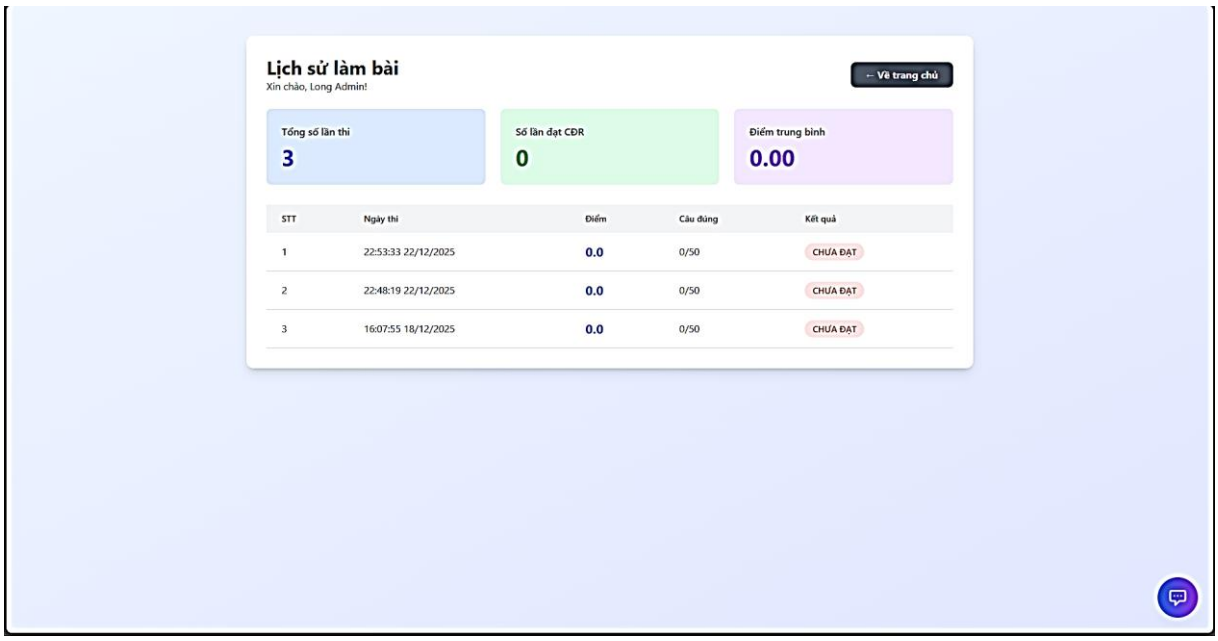

Giao diện làm đề thi đánh giá chuẩn đầu ra




Giao diện sau khi nộp bài ở đề thi

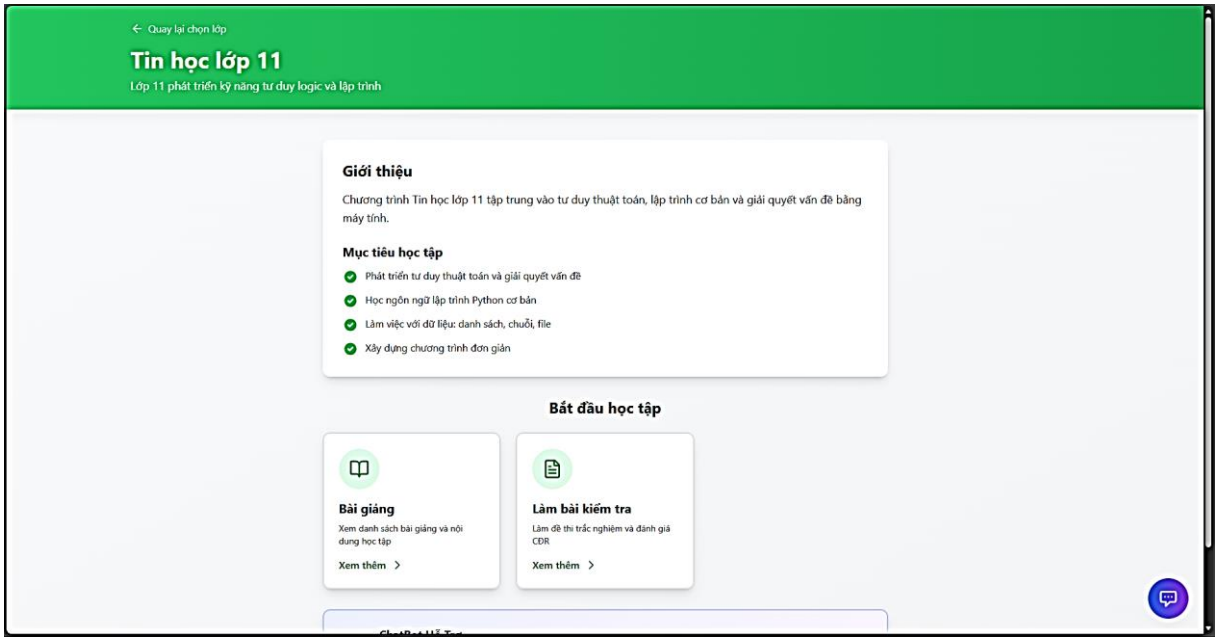



Chức năng xem đáp án

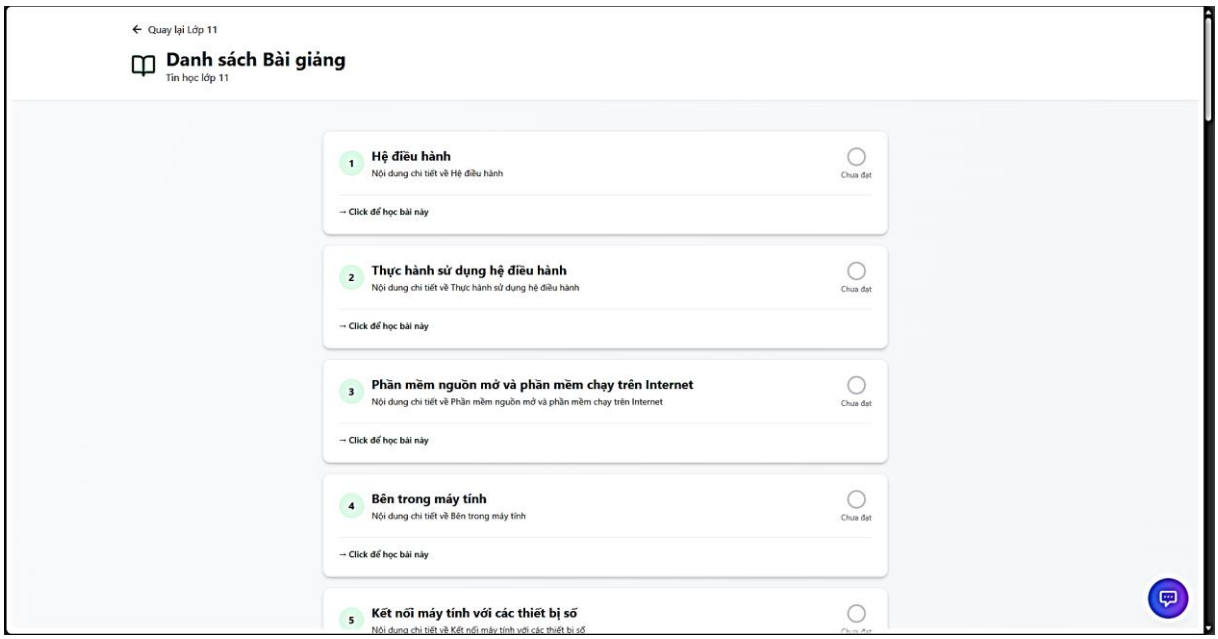



Giao diện lịch sử sau khi làm đề thi

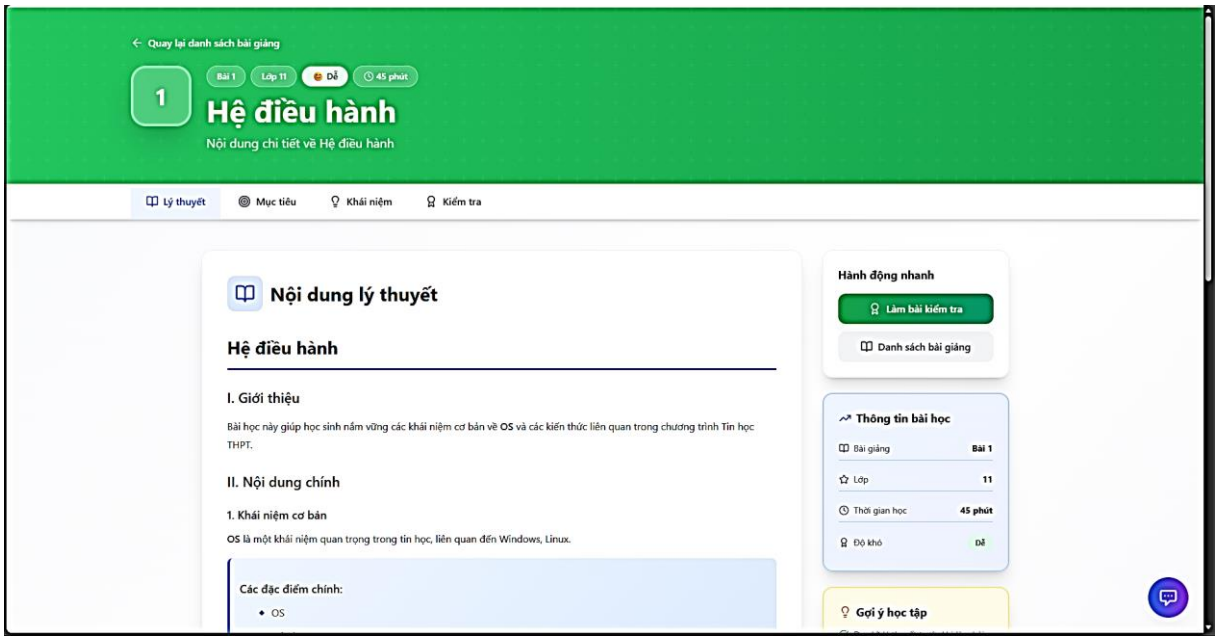
*Lớp 11:*



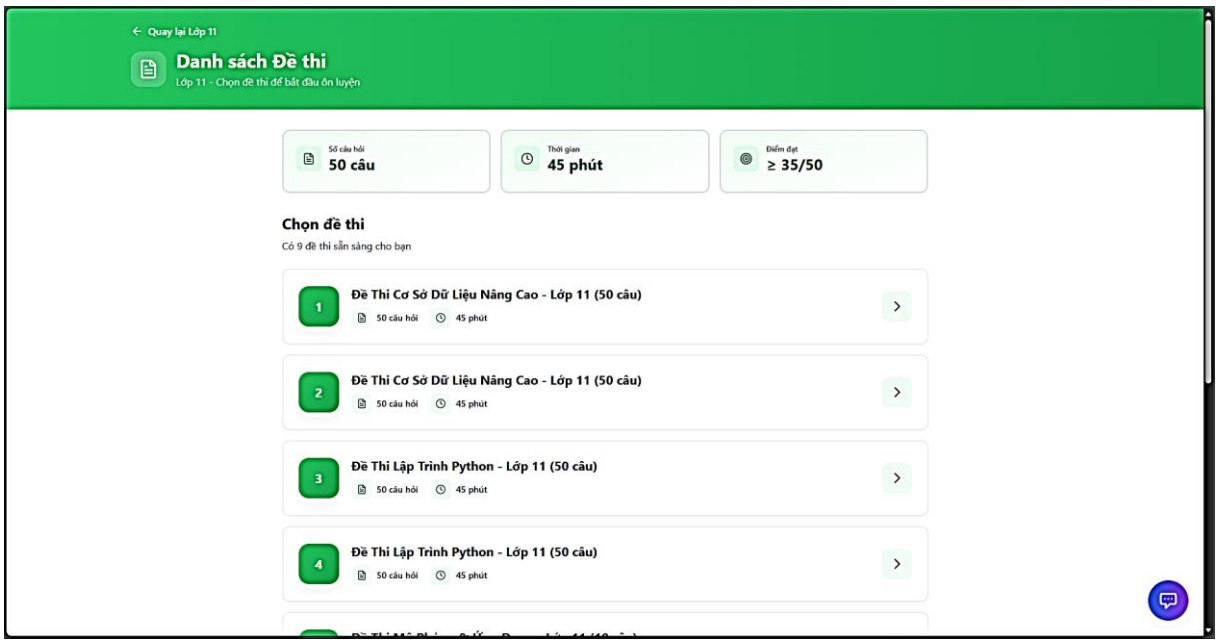

Giao diện chính




Giao diện bài giảng

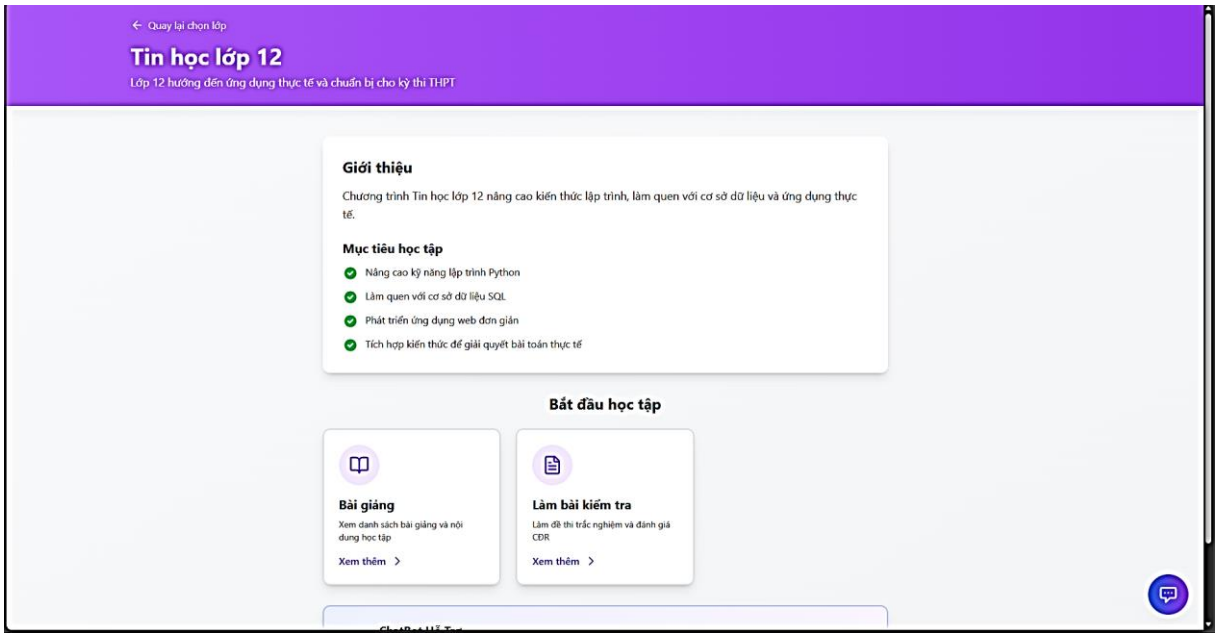



Giao diện trong 1 bài giảng của lớp 11

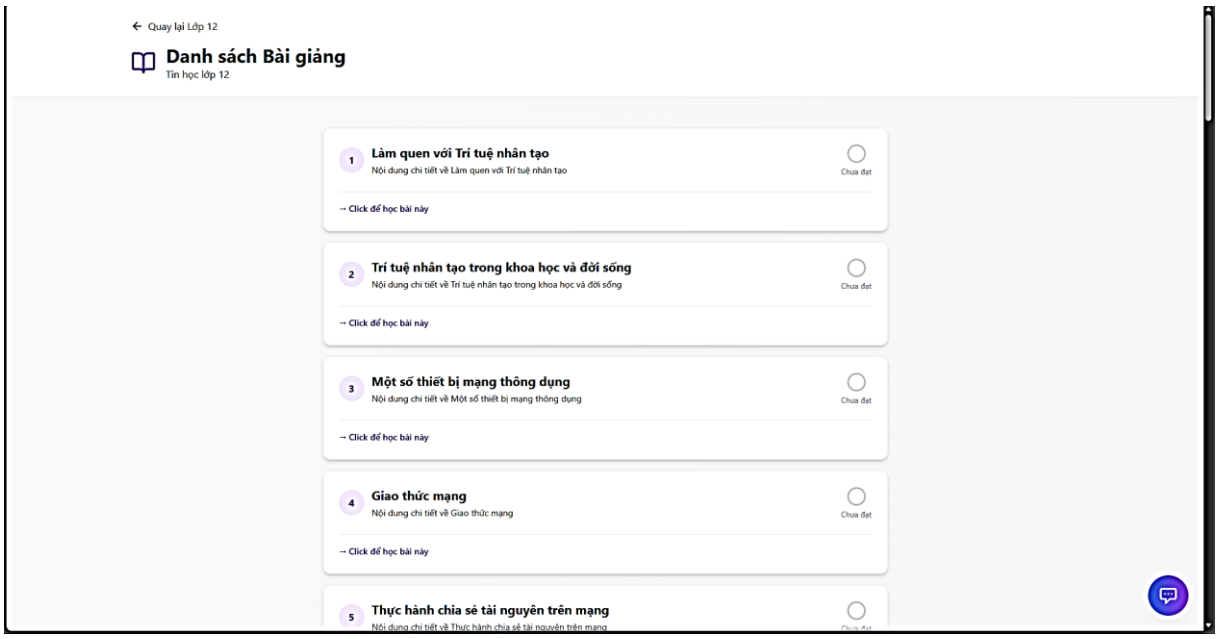



Giao diện danh sách đề thi

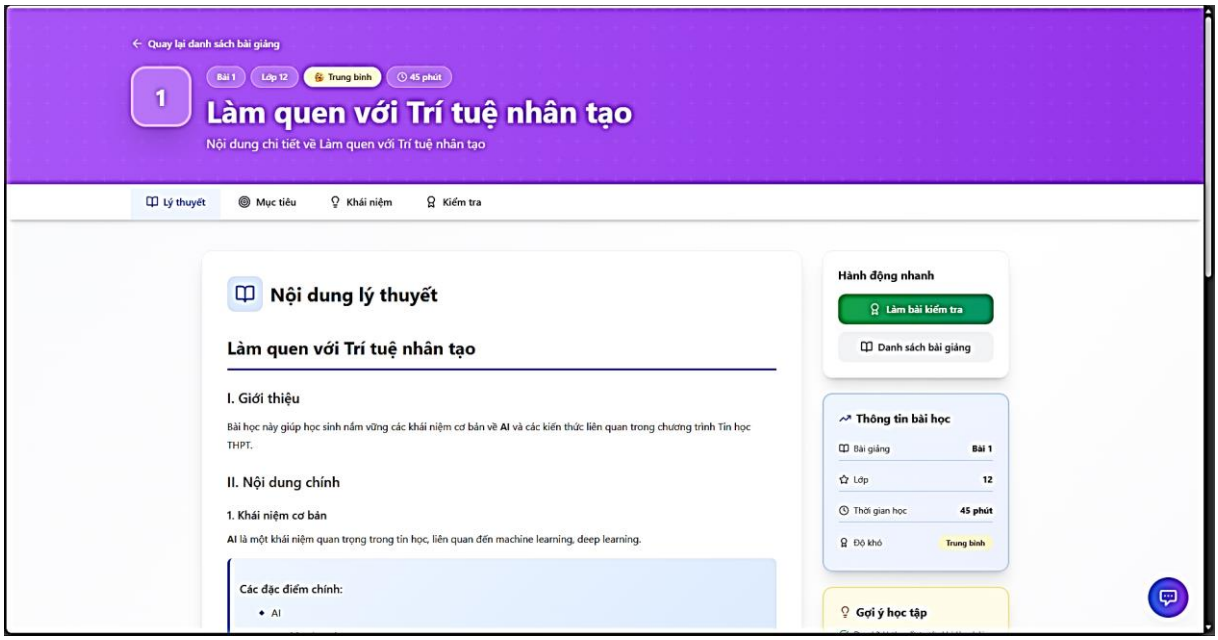
*Lớp 12:*



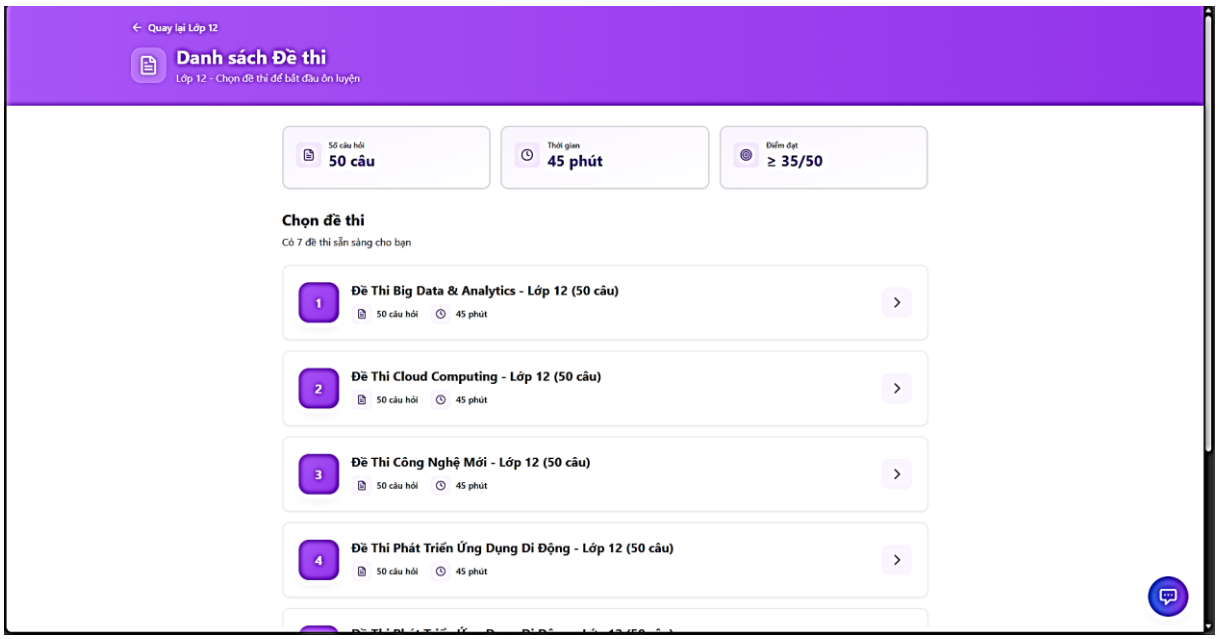

Giao diện chính




Giao diện danh sách bài giảng




Giao diện 1 bài giảng của lớp 12




Giao diện danh sách đề thi

## Phần III: Những điểm còn tồn tại và phương hướng phát triển

### 1. Những điểm còn tồn tại

#### 1.1. Về chức năng

- Chưa có chức năng xuất kết quả ra file PDF
- Chưa có tính năng gửi email thông báo kết quả
- Chưa hỗ trợ nhiều định dạng câu hỏi (chỉ có trắc nghiệm 4 đáp án)
- Chưa có chức năng so sánh kết quả giữa các lần thi

#### 1.2. Về giao diện

- Một số màn hình chưa responsive tốt trên tablet
- Chưa có chế độ dark mode
- Chưa có thông báo realtime khi admin thay đổi đề thi

#### 1.3. Về hiệu năng

- Chưa implement caching cho dữ liệu tĩnh
- Chưa tối ưu query database khi có nhiều user đồng thời

#### 1.4. Về bảo mật

- Chưa có cơ chế phát hiện gian lận (switch tab, copy/paste)
- Chưa mã hóa câu trả lời khi lưu trữ

### 2. Phương hướng phát triển

#### 2.1. Ngắn hạn

- Bổ sung thêm đề thi cho các lớp 10, 11, 12
- Thêm chức năng export kết quả PDF
- Tối ưu responsive trên tablet
- Thêm validation chặt chẽ hơn cho form

#### 2.2. Trung hạn

- Phát triển mobile app (React Native)
- Thêm video bài giảng
- Tích hợp email service để gửi thông báo
- Xây dựng forum hỏi đáp

#### 2.3. Dài hạn

- Mở rộng sang các môn học khác
- Áp dụng AI để đề xuất lộ trình học
- Tích hợp với hệ thống quản lý học sinh của trường

# Kết luận

Sau quá trình nghiên cứu và phát triển, website "Hỗ trợ học sinh ôn thi môn Tin THPT theo chuẩn đầu ra" đã hoàn thành và đi vào hoạt động với đầy đủ các chức năng cơ bản.

1. Kết quả đạt được

Hệ thống đã triển khai thành công các chức năng chính:

- Hệ thống thi trắc nghiệm với 14 đề thi (5 đề lớp 10, 4 đề lớp 11, 5 đề lớp 12)
- Chấm điểm tự động với công thức linh hoạt theo số câu hỏi
- Đánh giá chuẩn đầu ra (CĐR) tự động với ngưỡng 7.0 điểm
- Gợi ý bài học dựa trên kết quả thi
- Chatbot AI hỗ trợ học sinh 24/7
- Quản lý đề thi, câu hỏi, người dùng cho admin
- 42 bài học chi tiết cho 3 khối lớp

2. Công nghệ đã sử dụng

- Frontend: React.js, Tailwind CSS
- Backend: Node.js, Express.js
- Database: PostgreSQL (Supabase)
- AI: OpenAI GPT-4 API
- Deployment: Docker, Docker Compose
- Authentication: JWT

3. Đóng góp của đề tài

- Cung cấp công cụ ôn thi miễn phí cho học sinh
- Giảm tải công việc chấm bài cho giáo viên
- Áp dụng thành công AI vào giáo dục
- Triển khai hệ thống đánh giá chuẩn đầu ra tự động

4. Hạn chế

- Số lượng đề thi còn hạn chế (14 đề)
- Chưa có mobile app
- Chưa tối ưu hoàn toàn cho nhiều người dùng đồng thời

5. Triển vọng

Với nền tảng đã xây dựng, hệ thống có thể phát triển thêm nhiều tính năng mới như video bài giảng, forum học tập, và mở rộng sang các môn học khác.

# Tài liệu tham khảo

[1] Bộ Giáo dục và Đào tạo (2018), "Chương trình giáo dục phổ thông môn Tin học".

[2] React Documentation, https://react.dev/

[3] Node.js Documentation, https://nodejs.org/en/docs/

[4] Express.js Documentation, https://expressjs.com/

[5] Supabase Documentation, https://supabase.com/docs

[6] PostgreSQL Documentation, https://www.postgresql.org/docs/

[7] OpenAI API Documentation, https://platform.openai.com/docs/

[8] Tailwind CSS Documentation, https://tailwindcss.com/docs

[9] Docker Documentation, https://docs.docker.com/

[10] JWT Introduction, https://jwt.io/introduction