

**ĐẠI** **HỌC** **PHENIKAA**

**TRƯỜNG CÔNG NGỆ THÔNG TIN**

**PHÂN TÍCH THIẾT KẾ PHẦN MỀM**

**(N05)**

**“Dự án: Hệ thống Quản lý đăng kí học theo tín chỉ”**

**Nhóm 13**

|  |  |
| --- | --- |
| Đỗ Hoài Nam | [22014002@st.phenikaa-uni.edu.vn](mailto:22010392@st.phenikaa-uni.edu.vn) |
| Nguyễn Văn Minh | [22010487@st.phenikaa-uni.edu.vn](mailto:22010508@st.phenikaa-uni.edu.vn) |

**Giảng viên hướng dẫn:** ThS. Vũ Quang Dũng, ThS. Nguyễn Xuân Quế

Nội dung

[**Lời cảm ơn** 4](#_Toc211507081)

[**Các định nghĩa và từ viết tắt** 5](#_Toc211507082)

[**Chương 1. Giới thiệu đề tài** 8](#_Toc211507083)

[**1.1. Mục tiêu** 8](#_Toc211507084)

[**1.1.1. Mục tiêu tổng quát** 8](#_Toc211507085)

[**1.1.2. Mục tiêu chức năng** 8](#_Toc211507086)

[**1.1.3. Mục tiêu phi chức năng** 9](#_Toc211507087)

[**1.2. Phạm vi dự án (Rút gọn)** 10](#_Toc211507088)

[**1.2.1. Phạm vi chức năng:** 10](#_Toc211507089)

[**1.2.2. Phạm vi dữ liệu** 10](#_Toc211507090)

[**1.2.3. Triển khai & môi trường** 10](#_Toc211507091)

[**1.2.4. Người dùng & bên liên quan** 10](#_Toc211507092)

[**1.3. Mô hình quy trình đã chọn và lý do** 10](#_Toc211507093)

[**1.3.1. Mô hình quy trình** 10](#_Toc211507094)

[**1.3.2. Lý do lựa chọn** 11](#_Toc211507095)

[**Chương 2. Phân tích yêu cầu** 12](#_Toc211507096)

[**2.1. Mô tả dự án (tóm tắt)** 12](#_Toc211507097)

[**2.1.1. Các use case Diagram** 13](#_Toc211507098)

[**2.1.2. Mô tả actors** 16](#_Toc211507099)

[**2.2 Yêu cầu chức năng** 17](#_Toc211507100)

[**2.2.1. Danh mục Use Case** 17](#_Toc211507101)

[**2.2.2 Đặc tả chi tiết cho từng Use Case** 18](#_Toc211507102)

[**2.3. Yêu cầu phi chức năng** 28](#_Toc211507103)

[**2.3.1. Hiệu năng** 28](#_Toc211507104)

[**2.3.2. Bảo mật (phân quyền)** 28](#_Toc211507105)

[**2.3.3. Khả năng sử dụng (UI/UX)** 28](#_Toc211507106)

[**2.4. Đặc tả luồng dữ liệu (DFD)** 29](#_Toc211507107)

[**2.4.1. Phạm vi & Tác nhân** 29](#_Toc211507108)

[**2.4.2. Kho dữ liệu (Data Stores)** 29](#_Toc211507109)

[**2.4.3. Tiến trình (Processes) ↔ API (rút từ UI)** 29](#_Toc211507110)

[**2.4.4. DFD Level 0 (ngắn gọn)** 29](#_Toc211507111)

[**2.4.5. Ma trận CRUD (tóm tắt)** 30](#_Toc211507112)

[**2.4.6. Luồng điển hình** 30](#_Toc211507113)

[**2.4.7. Ràng buộc chính (tối thiểu)** 31](#_Toc211507114)

[**Chương 3. Thiết kế hệ thống** 31](#_Toc211507115)

[**3.1. Thiết kế kiến trúc** 31](#_Toc211507116)

[**3.1.1. Lựa chọn mô hình kiến trúc** 31](#_Toc211507117)

[**3.1.2. Mô hình triển khai (gợi ý)** 31](#_Toc211507118)

[**3.2. Thiết kế chi tiết (UML)** 32](#_Toc211507119)

[**3.2.1. Sơ đồ lớp (Class Diagram)** 32](#_Toc211507120)

[**3.2.2. Sơ đồ trình tự (Sequence Diagram)** 34](#_Toc211507121)

[**3.2.3. Biểu đồ Gói (Package Diagram)** 37](#_Toc211507122)

[**3.2.4. Biểu đồ Trạng thái (State Machine)** 38](#_Toc211507123)

[**3.2.5. Đặc tả DTO/Contract** 40](#_Toc211507124)

[**3.2.6. Thuật toán & kiểm tra nghiệp vụ (cụ thể hoá để code/test)** 41](#_Toc211507125)

[**3.2.7. Ràng buộc kiểm thử (testability)** 42](#_Toc211507126)

[**3.2.8. Liên kết mã nguồn (mapping nhanh để người đọc dò code)** 42](#_Toc211507127)

[**Chương 4. Thiết kế giao diện** 43](#_Toc211507128)

[**4.1. Mục tiêu & phạm vi** 43](#_Toc211507129)

[**4.1.1. Mục tiêu UX (Trải nghiệm người dùng)** 43](#_Toc211507130)

[**4.1.2. Mục tiêu UI (Giao diện)** 43](#_Toc211507131)

[**4.2. Giao diện chương trình (Figma)** 43](#_Toc211507132)

[**4.3. User Flow & Sitemap** 52](#_Toc211507133)

[**4.3.1. Gán mã màn hình (tham chiếu)** 52](#_Toc211507134)

[**4.3.2. Sitemap (cấu trúc điều hướng)** 52](#_Toc211507135)

[**4.3.3. Luồng thao tác chính(User Flow)** 53](#_Toc211507136)

[**Chương 5. Tổng kết và hướng phát triển** 54](#_Toc211507137)

[**5.1. Đánh giá** 54](#_Toc211507138)

[**5.1.1. Những điểm đã làm được** 54](#_Toc211507139)

[**5.1.2. Hạn chế & rút kinh nghiệm** 55](#_Toc211507140)

[**5.2. Hướng phát triển** 56](#_Toc211507141)

[**5.2.1. Ngắn hạn (ưu tiên cao)** 56](#_Toc211507142)

[**5.2.2. Trung hạn** 56](#_Toc211507143)

[**5.2.3. Dài hạn** 57](#_Toc211507144)

[**5.3. Kết luận** 57](#_Toc211507145)

[**Chương 6. Tài liệu tham khảo** 57](#_Toc211507146)

**Lời cảm ơn**

Đối với bất kì bất kì sinh viên nào thì bài tập lớn kết thúc môn luôn là thành quả quan trọng của một môn học. Và để bài tập lớn này được đạt kết quả tốt như hiện nay chúng em đã nhận được rất nhiều sự hỗ trợ, hướng dẫn vô cùng tận tâm và nhiệt huyết của ThS. Vũ Quang Dũng, ThS. Nguyễn Xuân Quế. Xuất phát từ sự chân thành, chúng em xin bày tỏ sự biết ơn sâu sắc nhất đến các thầy.

Qua những bài giảng vô cùng thú vị ở lớp, những tiết học cuốn hút, các thầy đã không ngừng tạo những điều kiện tốt để chúng em có được môi trường học tập tốt và là người trang bị cho chúng em tất cả những kiến thức, kỹ năng cần có để chúng em có thể hoàn thành tốt đồ án này.

Trong quá trình học tập và nghiên cứu, do bản thân chúng em vẫn còn chưa vững kiến thức về chuyên ngành và kinh nghiệm thực tế nên có nhiều điều chúng em còn thiếu sót và cần học hỏi nhiều hơn nữa.

Chúng em rất cảm ơn các thầy vẫn tận tâm, tỉ mỉ, dành nhiều thời gian để hướng dẫn, chỉ bảo cho chúng em, giúp chúng em ngày một hoàn thiện bản thân mình hơn. Chúng em kính chúc các thầy luôn luôn mạnh khỏe.

Chúng em xin chân thành cảm ơn!

**Các định nghĩa và từ viết tắt**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuật ngữ** | **Viết đầy đủ** | **Mô tả ngắn** | **Dịch nghĩa / Giải thích** |
| **2FA** | Two-Factor Authentication | Xác thực đăng nhập cần 2 yếu tố (mật khẩu + mã OTP, v.v.) | Xác thực hai lớp |
| **5xx** | HTTP 5xx Status Codes | Mã lỗi máy chủ trong HTTP (500–599) | Lỗi phía máy chủ |
| **95th percentile** | — | Ngưỡng hiệu năng tính theo phân vị 95% | 95% yêu cầu nhanh hơn mốc này |
| **AC** | Acceptance Criteria | Tiêu chí chấp nhận cho mỗi chức năng/use case | Tiêu chí chấp nhận |
| **ACID** | Atomicity, Consistency, Isolation, Durability | Tính chất giao dịch bảo đảm toàn vẹn dữ liệu | Giao dịch nguyên tử/nhất quán/cô lập/bền vững |
| **Admin** | — | Vai trò quản trị hệ thống | Quản trị viên |
| **API** | Application Programming Interface | Giao diện lập trình cho tích hợp hệ thống | Giao tiếp lập trình ứng dụng |
| **Approved** | — | Trạng thái đăng ký đã được duyệt | Đã duyệt |
| **Audit log** | — | Nhật ký ghi lại thao tác/biến đổi dữ liệu | Nhật ký kiểm toán |
| **Autoscale** | — | Tự động tăng/giảm tài nguyên theo tải | Tự động mở rộng |
| **BI** | Business Intelligence | Phân tích dữ liệu, báo cáo hỗ trợ quyết định | Trí tuệ doanh nghiệp |
| **BR** | Business Rule | Quy tắc nghiệp vụ điều khiển hành vi hệ thống | Quy tắc nghiệp vụ |
| **Cache** | — | Lưu tạm dữ liệu để truy cập nhanh | Bộ nhớ đệm |
| **Capacity** | — | Sức chứa tối đa của lớp/phòng | Chỉ tiêu/sức chứa |
| **CDN** | Content Delivery Network | Mạng phân phối nội dung tĩnh để tăng tốc | Mạng phân phối nội dung |
| **Completed** | — | Trạng thái đăng ký đã hoàn tất môn | Đã hoàn thành |
| **CRUD** | Create, Read, Update, Delete | Tập thao tác cơ bản với dữ liệu | Tạo/Đọc/Cập nhật/Xóa |
| **CSV** | Comma-Separated Values | Định dạng tệp dữ liệu dạng bảng | Tệp giá trị phân tách bằng dấu phẩy |
| **CTĐT** | — | Chương trình đào tạo, có phiên bản theo niên khóa | Curriculum |
| **CVHT** | — | Cố vấn học tập | Academic advisor |
| **DEV** | Development | Môi trường phát triển | Môi trường phát triển |
| **Dis/Overbooking** | — | Vượt quá số chỗ cho phép khi đăng ký | Đăng ký quá chỗ |
| **DR (site)** | Disaster Recovery | Hạ tầng dự phòng khắc phục thảm họa | Trung tâm dự phòng |
| **Drop/ Dropped** | — | Hủy đăng ký môn | Hủy đăng ký |
| **Dashboard** | — | Trang tổng quan số liệu | Bảng điều khiển |
| **E-mail RFC 5322** | — | Chuẩn định dạng địa chỉ email | Tiêu chuẩn RFC 5322 |
| **GET/POST/PUT/DELETE** | — | Các phương thức HTTP dùng trong API | Phương thức HTTP |
| **GPA** | Grade Point Average | Điểm trung bình học tập | Điểm trung bình tích lũy |
| **GV** | — | Giảng viên | Teacher/Instructor |
| **HA** | High Availability | Khả dụng cao của hệ thống | Tính sẵn sàng cao |
| **HTTPOnly cookie** | — | Cookie chỉ truy cập được từ HTTP, chống XSS | Cookie an toàn (HTTPOnly) |
| **Iterative & Incremental** | — | Quy trình phát triển lặp – tăng trưởng | Phát triển lặp & tăng dần |
| **LMS** | Learning Management System | Hệ thống quản lý học tập | Hệ quản trị học tập |
| **Lockout** | — | Khóa tạm tài khoản sau nhiều lần sai | Khóa tạm |
| **LMS/SSO/SIS** | — | Nhóm hệ tích hợp: học tập / đăng nhập tập trung / thông tin sinh viên | Tích hợp hệ thống |
| **NFR** | Non-Functional Requirements | Yêu cầu phi chức năng (hiệu năng, bảo mật…) | Yêu cầu phi chức năng |
| **On-prem** | On-premises | Triển khai tại hạ tầng nội bộ | Tại chỗ |
| **Pending** | — | Trạng thái đăng ký chờ duyệt | Đang chờ |
| **PDF** | Portable Document Format | Định dạng tài liệu phổ biến | Tệp PDF |
| **PĐT** | — | Phòng Đào tạo | Academic Affairs |
| **PROD** | Production | Môi trường sản xuất | Môi trường vận hành |
| **RPO** | Recovery Point Objective | Mức mất dữ liệu tối đa cho phép khi sự cố | Mục tiêu điểm khôi phục |
| **RTO** | Recovery Time Objective | Thời gian khôi phục dịch vụ tối đa cho phép | Mục tiêu thời gian khôi phục |
| **Scale-out** | — | Mở rộng theo chiều ngang (thêm máy) | Mở rộng ngang |
| **SIS** | Student Information System | Hệ thống thông tin sinh viên | Hệ quản trị thông tin SV |
| **SSO** | Single Sign-On | Đăng nhập một lần cho nhiều hệ thống | Đăng nhập một lần |
| **Staging/UAT** | User Acceptance Testing | Môi trường kiểm thử chấp nhận người dùng | Môi trường UAT/Staging |
| **SV** | — | Sinh viên | Student |
| **Token/Access token** | — | Chuỗi định danh phiên/ủy quyền | Mã truy cập |
| **TKB** | — | Thời khóa biểu | Timetable |
| **UI** | User Interface | Giao diện người dùng | Giao diện |
| **Uptime 99.9%** | — | Cam kết sẵn sàng dịch vụ trong mùa đăng ký | Thời gian hoạt động |
| **UUID/ULID** | — | Định danh duy nhất cho bản ghi | Mã định danh duy nhất |
| **Use case (UC-xx)** | — | Trường hợp sử dụng; danh mục UC-01…UC-31 | Trường hợp sử dụng |
| **Waitlist** | — | Danh sách chờ khi lớp đầy | Danh sách chờ |

**Chương 1. Giới thiệu đề tài**

**1.1. Mục tiêu**

Bản phân tích và thiết kế này cung cấp mô tả chi tiết về **Hệ thống Quản lý Đăng ký Tín chỉ Học phần** (sau đây gọi tắt là “Hệ thống”), bao gồm mục tiêu, phạm vi, nhóm người dùng chính, các chức năng cốt lõi và các tiêu chí chất lượng cần đạt. Hệ thống hướng đến việc chuẩn hóa và tự động hóa toàn bộ quy trình đăng ký học phần theo học chế tín chỉ, đảm bảo tính công bằng, minh bạch, chính xác và trải nghiệm sử dụng thuận tiện cho sinh viên, giảng viên và bộ phận quản lý đào tạo.

Trong bối cảnh số hóa giáo dục diễn ra mạnh mẽ, số lượng môn học, lớp học phần, ràng buộc tiên quyết – song hành – tương đương, cùng các chính sách học vụ ngày càng đa dạng khiến việc đăng ký thủ công, phân lớp và kiểm soát sĩ số trở nên phức tạp, dễ sai sót và tốn nguồn lực. Hệ thống được xây dựng nhằm giải quyết các vấn đề đó, hỗ trợ **đăng ký – kiểm tra điều kiện – xếp lớp** một cách chính xác theo thời gian thực, đồng thời giảm tải cho phòng đào tạo.

Với mục tiêu trở thành nền tảng đăng ký tín chỉ trực tuyến **hữu ích – tiện lợi – mở rộng linh hoạt**, Hệ thống cung cấp giao diện thân thiện, quy trình đơn giản và phản hồi tức thì để sinh viên dễ dàng tra cứu chương trình đào tạo, lập kế hoạch học kỳ, đăng ký/hủy/muộn đăng ký theo quy định; giảng viên quản lý lớp học phần, sĩ số, lịch học – lịch thi; và phòng đào tạo kiểm soát năng lực phòng học, lịch biểu toàn trường, chỉ tiêu, chuẩn đầu ra cùng các báo cáo phục vụ ra quyết định.

**1.1.1. Mục tiêu tổng quát**

* **Chuẩn hóa quy trình** đăng ký tín chỉ theo quy chế học vụ và chương trình đào tạo của nhà trường.
* **Tự động hóa kiểm tra ràng buộc** (tiên quyết, học trước, song hành, tương đương, trùng lịch, giới hạn tín chỉ/học kỳ, trùng lớp, trùng ca thi…).
* **Tối ưu phân bổ nguồn lực** (chỉ tiêu lớp, phòng học, lịch giảng, lịch thi) và giảm thiểu tình trạng nghẽn hệ thống trong “giờ cao điểm”.
* **Minh bạch & thời gian thực**: cập nhật sĩ số, danh sách chờ (waitlist), trạng thái đăng ký, thông báo thay đổi tức thì.

**1.1.2. Mục tiêu chức năng**

* **Quản lý danh mục đào tạo**
  + Chương trình/khóa học, học phần, phiên bản CTĐT theo niên khóa.
  + Điều kiện ràng buộc: tiên quyết, song hành, tương đương, tái đăng ký, khống chế số lần học/thi lại.
* **Kế hoạch** 
  + Kế hoạch mở lớp học phần theo học kỳ (giảng viên, thời khóa biểu, phòng học, chỉ tiêu, hình thức dạy-học).
* **Đăng ký học phần cho sinh viên**
  + Đăng ký/hủy/đổi lớp, danh sách chờ tự động khi đầy lớp; áp dụng cửa sổ thời gian theo đợt/mốc ưu tiên.
* **Quản lý lớp học phần**
  + Theo dõi sĩ số, danh sách sinh viên, phân nhóm thực hành/thảo luận, điểm danh, xuất danh sách.
  + Điều chỉnh chỉ tiêu (theo phê duyệt), chuyển phòng/đổi lịch có thông báo đến người liên quan.
* **Tích hợp – mở rộng**
  + Tích hợp cổng định danh/SSO, học phí & hóa đơn (liên kết công nợ học phí thì ghi nhận tín chỉ đăng ký), LMS, thư viện.
  + API nội bộ phục vụ đồng bộ dữ liệu và phân tích BI.

**1.1.3. Mục tiêu phi chức năng**

* **Hiệu năng & khả dụng cao:** chịu tải giờ cao điểm (hàng nghìn yêu cầu/giây), thời gian phản hồi < 2s cho thao tác đăng ký; uptime ≥ 99.9% trong mùa đăng ký.
* **Mở rộng tuyến tính:** kiến trúc có thể mở rộng theo chiều ngang (scale-out), cân bằng tải, cache chiến lược.
* **Tính nhất quán & toàn vẹn dữ liệu:** đảm bảo giao dịch đăng ký nguyên tử (ACID) để tránh “overbooking”.
* **Bảo mật & riêng tư:** tuân thủ bảo vệ dữ liệu cá nhân, phân quyền theo vai trò (SV/giảng viên/cán bộ), ghi nhật ký (audit log), mã hóa dữ liệu nhạy cảm.

**1.2. Phạm vi dự án (Rút gọn)**

**1.2.1. Phạm vi chức năng:**

* **Danh mục đào tạo:** Quản lý CTĐT (phiên bản theo khóa), khối kiến thức, chuẩn đầu ra; học phần & nhóm tương đương; ràng buộc học vụ (tiên quyết/học trước/song hành/cấm trùng, theo ngành/khóa).
* **Kế hoạch & đăng ký:** Lập kế hoạch/mở lớp (chỉ tiêu, GV, lịch–phòng), lịch thi tránh trùng; quy trình đăng ký của SV với giỏ đăng ký/hủy/đổi, kiểm tra ràng buộc tức thì, danh sách chờ & giữ chỗ, cảnh báo trùng lịch/giới hạn tín chỉ.

**1.2.2. Phạm vi dữ liệu**

Gồm người dùng (SV/GV/Cán bộ), CTĐT, học phần/nhóm tương đương, ràng buộc, lớp học phần, lịch học–thi, phòng, đăng ký, công nợ/học phí (nếu có), nhật ký. Bảo đảm toàn vẹn (khóa ngoại), phiên bản hóa CTĐT, lịch sử thay đổi, kiểm soát đồng thời. Lưu trữ dữ liệu cho các khóa đang học và tối thiểu N năm sau tốt nghiệp.

**1.2.3. Triển khai & môi trường**

DEV, UAT/Staging, PROD với dữ liệu ẩn danh; hạ tầng on-prem hoặc cloud, ưu tiên container, autoscale, HA; web responsive, có thể bổ sung app di động nhẹ; đảm bảo hiệu năng giờ cao điểm với CDN/cache đọc.

**1.2.4. Người dùng & bên liên quan**

Sinh viên theo học chế tín chỉ; giảng viên giảng dạy; cán bộ đào tạo/khoa/bộ môn lập kế hoạch, vận hành và báo cáo.

**1.3. Mô hình quy trình đã chọn và lý do**

**1.3.1. Mô hình quy trình**

Chọn mô hình lặp–tăng trưởng (Iterative & Incremental) theo Agile Scrum, có quản trị rủi ro kiểu Spiral nhẹ và kiểm thử theo V-Model trong từng vòng lặp. Cấu trúc chính:

* **Scrum** làm khung quản lý (kế hoạch, vai trò, nghi lễ, nhịp độ).
* **Phát hành tăng trưởng** theo từng **Increment** có thể triển khai thực tế.
* **Quản trị rủi ro** (hiệu năng giờ cao điểm, nhất quán giao dịch, tích hợp SSO/học phí) theo chu kỳ **Identify → Mitigate → Validate** trong mỗi sprint.
* **Kiểm thử sớm & liên tục**: mỗi hạng mục đều có kiểm thử đơn vị/tích hợp/phi chức năng kèm tiêu chí chấp nhận.

**1.3.2. Lý do lựa chọn**

* **Nhu cầu thay đổi thường xuyên** (quy chế học vụ, ràng buộc tiên quyết, cửa sổ đăng ký) ⇒ Agile thích ứng tốt, cấu hình hóa rule.
* **Ràng buộc kỹ thuật phức tạp** (ACID, chống overbooking, scale giờ cao điểm) ⇒ lặp–tăng trưởng giúp **thử–đo–học** sớm với dữ liệu giả lập, tránh dồn rủi ro về cuối.
* **Phạm vi rộng, nhiều bên liên quan** (SV, giảng viên, phòng đào tạo, tài vụ) ⇒ phát hành theo **module** để nhận phản hồi thật, giảm sai lệch kỳ vọng.
* **Yêu cầu phi chức năng nghiêm ngặt** (99.9% uptime mùa đăng ký, TTR<2s) ⇒ kiểm thử hiệu năng/độ tin cậy **lặp lại liên tục** trước mỗi mốc phát hành.
* **Cửa sổ thời gian gấp** (chuẩn bị trước mỗi học kỳ) ⇒ Increment có giá trị sử dụng sớm, tránh “big-bang”.

**Chương 2. Phân tích yêu cầu**

**2.1. Mô tả dự án (tóm tắt)**

**Mục tiêu:** Hệ thống quản lý đăng ký tín chỉ hỗ trợ trọn quy trình học vụ theo học chế tín chỉ: mở lớp → tư vấn lộ trình → đăng ký/huỷ/đổi → kiểm tra tiên quyết & xung đột TKB → tính/chốt học phí → theo dõi kết quả.  
Tập trung 3 đích đến:

1. Sinh viên đăng ký nhanh, chính xác, đúng lộ trình;
2. Giảng viên/Phòng Đào tạo vận hành, kiểm duyệt và tối ưu sĩ số;
3. Nhà trường giám sát số liệu thời gian thực để ra quyết định.

**Chức năng theo vai trò**

Sinh viên

* Tìm & đăng ký học phần: Lọc theo kỳ/khoa/GV/khung giờ/cơ sở; giỏ đăng ký–hủy–đổi; trạng thái (đã đăng ký/chờ duyệt/waitlist/đã hủy).

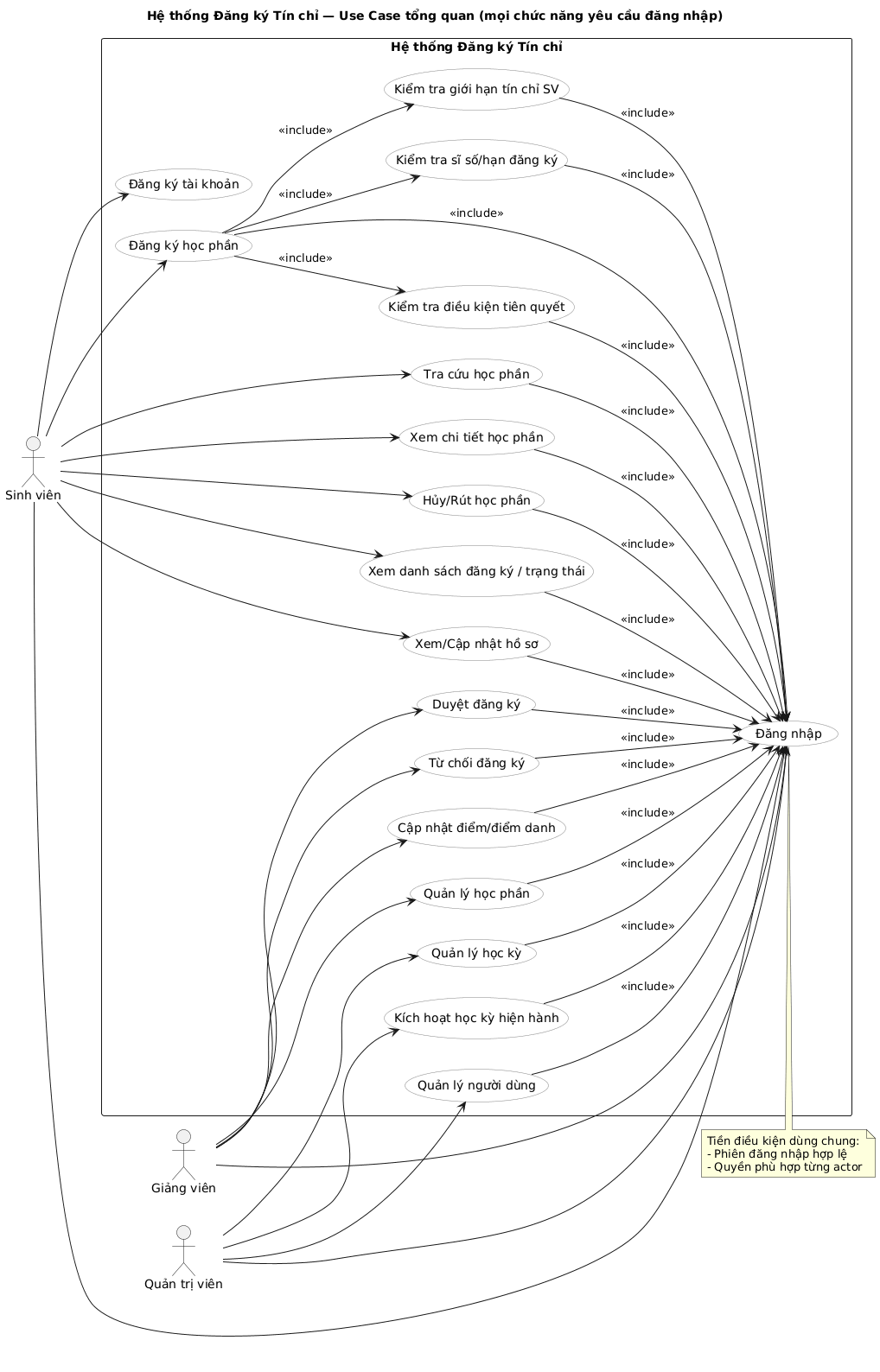
Giảng viên

* Quản lý lớp phụ trách: Phê duyệt sinh viên đã đăng ký học phần ở lớp giáo viên phụ trách
* Theo dõi & báo cáo nhanh: Tình trạng đăng ký, mức độ quan tâm học phần, tỉ lệ trùng lịch sinh viên.

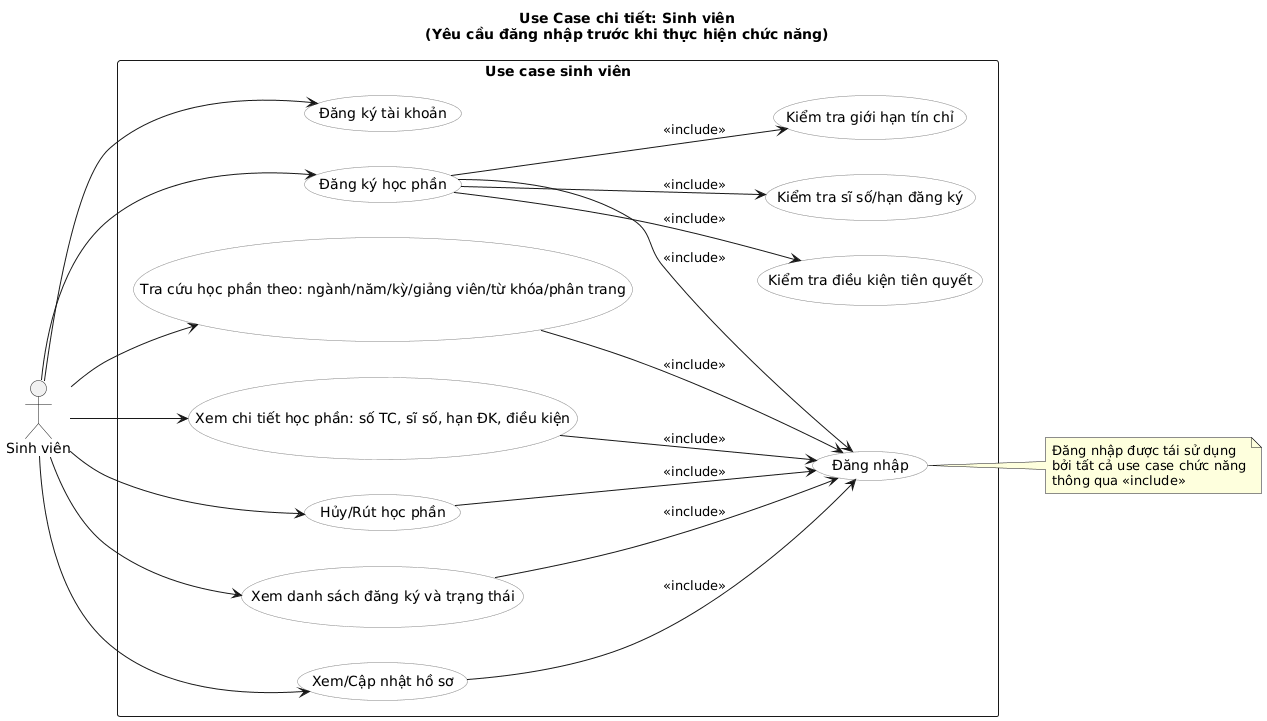
Quản trị viên

* Cấu hình & lập kế hoạch: Tạo/cập nhật kỳ học, mốc mở–đóng đăng ký; quản lý CTĐT, học phần, ràng buộc (tiên quyết/song hành/cấm trùng), chỉ tiêu–lịch–phòng; chính sách waitlist.
* Điều phối & kiểm duyệt: Phân công giảng viên; xét duyệt ngoại lệ (vượt trần tín chỉ, học chéo khoa… nếu áp dụng); phát hiện & xử lý xung đột phòng/GV; gộp/huỷ/mở thêm lớp; phát hành thông báo.

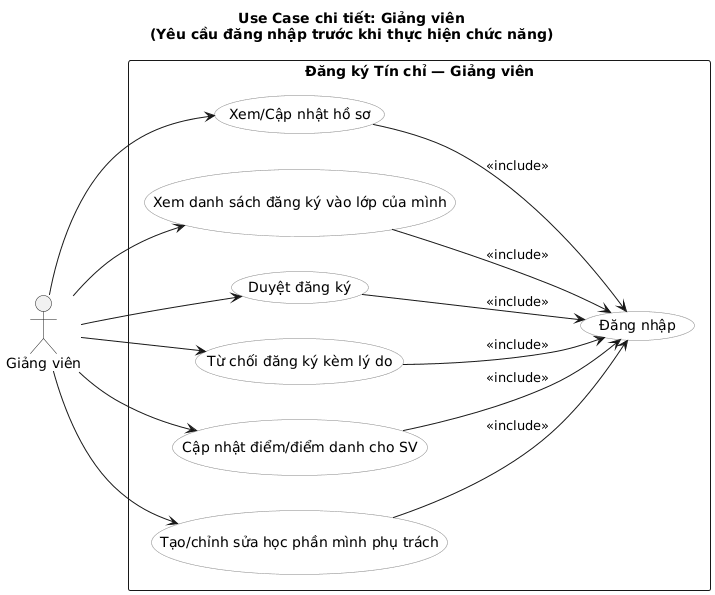
**2.1.1. Các use case Diagram**

****

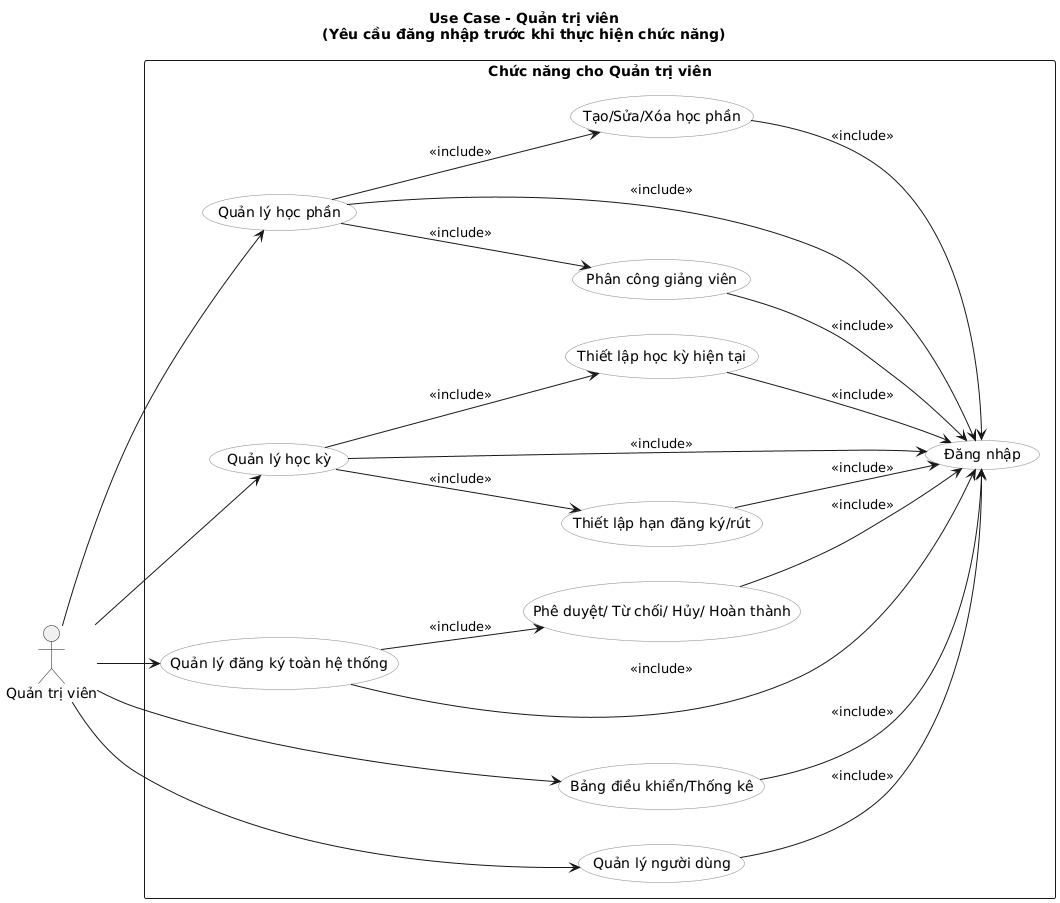
*Hình 1. Use case tổng quát hệ thống*

**

*Hình 2. Use case sinh viên*

**

*Hình 3. Use case giảng viên*

**

*Hình 4. Use case quản trị viên*

**2.1.2. Mô tả actors**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Actor** | **Nhiệm vụ chính** | **Quyền / Thao tác** | **Luồng thao tác** | **Ràng buộc / Quy tắc** | **Màn hình chính** |
| Sinh viên | Tìm & xem môn; đăng ký/hủy; theo dõi trạng thái; xem điểm. | Lọc/tìm môn; xem sĩ số, GV, lịch; Đăng ký; Hủy trước hạn; Xem điểm & biên nhận. | Đăng nhập → Tìm môn → Chi tiết → Đăng ký → Theo dõi 'Đăng ký của tôi' → Hủy (nếu cần) → Xem điểm khi hoàn tất. | Chỉ khi kỳ/môn mở; không vượt sĩ số; trong hạn đăng ký/rút; điểm xuất hiện khi completed. | Danh sách môn; Chi tiết môn; Đăng ký của tôi; Hồ sơ cá nhân. |
| Giảng viên | Quản lý lớp phụ trách; duyệt/từ chối; cập nhật tiến độ & điểm. | Cập nhật thông tin lớp (lịch, hạn, sĩ số); Duyệt/Từ chối yêu cầu; Nhập điểm; đánh dấu completed. | Vào quản lý → Xem yêu cầu → Phê duyệt/Từ chối → Cập nhật lớp → Nhập điểm & hoàn tất. | Chỉ lớp được phân công; tuân thủ hạn; trạng thái pending/approved/rejected/dropped/completed. | QL khóa học; QL đăng ký; Dashboard giảng viên. |
| Quản trị viên | Quản trị người dùng, học kỳ, học phần, đăng ký; theo dõi số liệu. | Quản lý người dùng & vai trò; tạo/sửa học kỳ; CRUD học phần; gán GV/kỳ; đặt sĩ số, hạn đăng ký/rút; duyệt/điều chỉnh đăng ký; xem dashboard. | Dashboard → Thiết lập học kỳ → Tạo/cấu hình học phần → Theo dõi/duyệt đăng ký → Quản lý tài khoản. | Toàn quyền dữ liệu; hạn & sĩ số chi phối SV/GV; đảm bảo nhất quán trạng thái & kỳ mở/đóng. | Dashboard quản trị; QL người dùng; QL học kỳ; QL học phần; QL đăng ký. |

**2.2 Yêu cầu chức năng**

**2.2.1. Danh mục Use Case**

**Nhóm Xác thực & Hồ sơ**

UC-01: Đăng nhập

UC-02: Đăng ký tài khoản

UC-03: Đăng xuất

UC-04: Xem & cập nhật hồ sơ cá nhân (bao gồm đổi mật khẩu)

**Nhóm Khóa học (Sinh viên/Khách xem; GV/QL trị quản lý)**

UC-05: Xem chi tiết học phần  
UC-06: Đăng ký học phần  
UC-07: Hủy/Drop đăng ký học phần của tôi  
UC-08: Xem các đăng ký của tôi (lọc theo trạng thái)

**Nhóm Quản trị học phần/học kỳ (Admin/Giảng viên)**

UC-09: Tạo học phần  
 UC-10: Cập nhật học phần  
 UC-11: Xóa học phần  
 UC-12: Quản lý giảng viên phụ trách học phần

**Nhóm Quản trị đăng ký (Admin/Giảng viên)**

UC-13: Duyệt/Phê duyệt đăng ký  
 UC-14: Từ chối đăng ký  
 UC-15: Đánh dấu hoàn thành/hủy/báo cáo tình trạng đăng ký

**Nhóm Quản trị học kỳ (Admin)**

UC-16: Tạo học kỳ  
 UC-17: Cập nhật học kỳ  
 UC-18: Đóng/Mở học kỳ, thiết lập cửa sổ đăng ký  
 UC-19: Xem học kỳ hiện hành

**Nhóm Quản trị người dùng (Admin)**  
 UC-20: Xem chi tiết người dùng  
 UC-21: (Tuỳ chọn) Gán/đổi vai trò người dùng

**Nhóm Dashboard & Báo cáo**

UC-22: Dashboard Sinh viên (tổng tín chỉ, tình trạng đăng ký gần đây)  
 UC-23: Dashboard Giảng viên (lớp phụ trách, đăng ký môn mình dạy)  
 UC-24: Dashboard Quản trị (thống kê người dùng/khóa học/đăng ký)

**2.2.2 Đặc tả chi tiết cho từng Use Case**

**UC-01: Đăng nhập**

* **Mục tiêu:** Cho phép người dùng hợp lệ truy cập hệ thống.
* **Phạm vi:** Xác thực phiên dựa trên email/mật khẩu.
* **Tác nhân:** Khách.
* **Tiền điều kiện:** Tài khoản tồn tại, trạng thái *active*.
* **Kích hoạt:** Người dùng nhấn **Login** trên trang Login.
* **Hậu điều kiện (Success):** Phiên hợp lệ được tạo; chuyển hướng **Dashboard** theo vai trò.  
  **(Failure):** Không tạo phiên; hiển thị thông báo lỗi cụ thể.
* **Luồng chính:**
  1. Người dùng nhập **email** và **mật khẩu**.
  2. Hệ thống kiểm tra định dạng email, độ dài mật khẩu (≥8 ký tự—cấu hình).
  3. Gửi yêu cầu xác thực (Auth service).
  4. Nếu hợp lệ: tạo accessToken/session; lưu an toàn (HTTPOnly cookie).
  5. Điều hướng tới Dashboard tương ứng (SV/GV/Admin).
* **Luồng thay thế/ngoại lệ:**
  1. A1: Sai thông tin → tăng bộ đếm thất bại; **BR:** Email hợp lệ; lockout sau N lần; token hết hạn sau T phút; buộc 2FA (nếu bật chính sách).
* **Dữ liệu & Ràng buộc:** email (RFC 5322), password (≥8, có chữ & số—tùy chính sách).
* **Quyền truy cập:** Public endpoint.
* **Thông báo/Lỗi:** Sai email hoặc mật khẩu; .
* **AC:**
  1. AC1: Nhập đúng thông tin → vào Dashboard đúng vai trò.
  2. AC2: Token lưu dưới cookie HTTPOnly.
* **Audit:** Ghi thời điểm đăng nhập, IP, user-agent, kết quả (success/fail).
* **NFR:** Thời gian phản hồi < 2s trong 95th percentile.
* **UI/API:** Login.js; (Auth service nội bộ).

**UC-02: Đăng ký tài khoản**

* **Mục tiêu:** Cho phép tạo tài khoản mới.
* **Phạm vi:** Tự đăng ký người dùng (mặc định role student; teacher qua admin duyệt nếu chính sách yêu cầu).
* **Tác nhân:** Khách.
* **Tiền điều kiện:** Email chưa tồn tại.
* **Kích hoạt:** Nhấn **Create account** trên Register.
* **Hậu điều kiện (Success):** Tài khoản ở trạng thái *pending* hoặc *active* (theo cấu hình xác minh email).  
  **(Failure):** Không tạo tài khoản, hiển thị lỗi cụ thể.
* **Luồng chính:**
  1. Nhập họ tên, email, mật khẩu, (mã SV, khoa, niên khóa...).
  2. Kiểm tra client-side định dạng & bắt buộc trường.
  3. Gửi đăng ký; server kiểm tra *uniqueness* email, studentCode.
  4. Tạo tài khoản; gửi email xác minh (nếu bật).
  5. Hiển thị hướng dẫn đăng nhập/xác minh.
* **Ngoại lệ:** Email trùng; studentCode trùng; mật khẩu yếu; lỗi mailer.
* **BR:** Password policy; studentCode duy nhất; có thể yêu cầu email tổ chức (@...edu.vn).
* **Dữ liệu/Ràng buộc:** fullName (≤100), email (unique), password (policy), studentCode (regex mã SV), faculty, intakeYear.
* **Quyền:** Public endpoint.
* **Thông báo:** Email đã tồn tại; Vui lòng xác minh email.
* **AC:**
  1. AC1: Đăng ký với dữ liệu hợp lệ → tạo user.
  2. AC2: Email trùng → bị từ chối với mã lỗi 409.
  3. AC3: Nhận email xác minh khi bật policy.
* **Audit:** Log tạo tài khoản (by=system, ip).
* **NFR:** Chống spam (rate limit, CAPTCHA nếu cần).
* **UI/API:** Register.js.

**UC-03: Đăng xuất**

* **Mục tiêu:** Kết thúc phiên làm việc an toàn.
* **Tác nhân:** Mọi vai trò.
* **Tiền điều kiện:** Đang đăng nhập.
* **Kích hoạt:** Chọn **Logout** trên navbar.
* **Hậu điều kiện:** Xóa session/token; điều hướng Login.
* **Luồng chính:** Xóa cookie token (server/invalidate); redirect.
* **Ngoại lệ:** Lỗi mạng → vẫn xóa token client; hiển thị đã đăng xuất.
* **BR:** Invalidate phía server để vô hiệu hóa token bị rò rỉ.
* **AC:** Sau logout, truy cập trang bảo vệ → bị yêu cầu đăng nhập.
* **Audit:** Log logout (by user, time).

**UC-04: Xem & cập nhật hồ sơ cá nhân**

* **Mục tiêu:** Cho phép người dùng tự quản hồ sơ, đổi mật khẩu.
* **Tác nhân:** SV/GV/Admin.
* **Tiền điều kiện:** Đăng nhập.
* **Kích hoạt:** Mở trang Profile.
* **Hậu điều kiện:** Hồ sơ được cập nhật; mật khẩu thay đổi (nếu thao tác).
* **Luồng chính:**
  1. Tải dữ liệu hồ sơ hiện tại.
  2. Người dùng sửa các trường cho phép; bấm **Lưu**.
  3. Server xác thực giá trị; cập nhật; trả về bản mới.
* **Luồng thay thế:** Đổi mật khẩu → yêu cầu oldPassword, newPassword, xác nhận khớp.
* **Ngoại lệ:** oldPassword sai; định dạng điện thoại sai; kích thước ảnh > 2MB.
* **BR:** Một số trường chỉ đọc (email, mã SV); đổi mật khẩu yêu cầu lại đăng nhập theo chính sách.
* **Dữ liệu/Ràng buộc:** phone (regex VN), avatar (≤2MB, jpg/png), displayName (≤100).
* **AC:** Cập nhật hợp lệ phản ánh ngay UI; sai định dạng hiển thị message cạnh trường.
* **Audit:** Log thay đổi trường nhạy cảm; che giá trị.

**UC-05: Xem chi tiết học phần**

* **Mục tiêu:** Hiển thị đầy đủ thông tin 1 học phần và trạng thái của SV.
* **Tác nhân:** Tất cả.
* **Tiền điều kiện:** courseId hợp lệ.
* **Kích hoạt:** Nhấp vào 1 dòng trong Courses.
* **Hậu điều kiện:** Thông tin chi tiết + nút hành động tương ứng trạng thái/role.
* **Luồng chính:**
  1. GET /api/courses/:id tải chi tiết (mô tả, credits, lịch, phòng, capacity, enrolled, tiên quyết, GV…).
  2. GET /api/semesters/current để xác định cửa sổ đăng ký.
  3. Nếu là SV đã đăng nhập, GET /api/registrations?studentId=me&courseId=:id để lấy trạng thái.
* **Ngoại lệ:** 404 course; server lỗi.
* **BR:** Ngoài cửa sổ → disable **Đăng ký**; đã đăng ký → hiển thị **Drop** nếu cho phép.
* **AC:** Hiển thị chuẩn xác capacity/enrolled; trạng thái nút đúng bối cảnh.

**UC-06: Đăng ký học phần**

* **Mục tiêu:** Ghi nhận yêu cầu đăng ký môn của SV.
* **Tác nhân:** Sinh viên.
* **Tiền điều kiện:** Trong cửa sổ; không trùng lịch; chưa vượt tín chỉ; còn chỗ; đạt tiên quyết.
* **Kích hoạt:** Nút **Đăng ký** tại CourseDetail.
* **Hậu điều kiện:** Tạo bản ghi Registration (pending/approved).
* **Luồng chính:**
  1. Client gửi POST /api/registrations {courseId, semesterId(current)}.
  2. Server **re-validate** các ràng buộc (cửa sổ, capacity, trùng lịch, tín chỉ, tiên quyết).
  3. Nếu đạt: tạo record; nếu auto-approve và còn chỗ → approved & tăng enrolled.
* **Ngoại lệ:** Vi phạm bất kỳ ràng buộc → trả mã lỗi chuyên biệt (OUT\_OF\_WINDOW, CAPACITY\_FULL, CLASH, CREDIT\_LIMIT, PREREQ\_FAIL, DUPLICATE).
* **BR (chính):** BR-07.1…BR-07.6 như đã nêu ở mục 4.
* **Dữ liệu/Ràng buộc:** courseId bắt buộc; semesterId = current; id hợp lệ UUID/ULID.
* **AC:** Gặp CAPACITY\_FULL → hiển thị "Lớp đã đủ chỗ"; thành công → thẻ trạng thái chuyển approved/pending tức thì.
* **Audit:** Ghi by=student, courseId, kết quả.

**UC-07: Hủy/Drop đăng ký học phần**

* **Mục tiêu:** Cho phép SV tự hủy đăng ký còn hiệu lực.
* **Tác nhân:** Sinh viên.
* **Tiền điều kiện:** Có đăng ký ở trạng thái cho phép drop; trong hạn drop.
* **Kích hoạt:** Nút **Drop** tại MyRegistrations/CourseDetail.
* **Hậu điều kiện:** status → dropped; giảm enrolled nếu previously approved.
* **Luồng chính:** PUT /api/registrations/:id/drop → server kiểm tra quyền sở hữu + hạn drop → cập nhật.
* **Ngoại lệ:** Quá hạn; status không cho drop; không phải chủ sở hữu.
* **BR:** Giải phóng suất khi drop thành công; ghi lý do (tùy chọn).
* **AC:** Sau drop, bản ghi biến mất khỏi danh sách *đang học*; xuất hiện ở tab *đã hủy*.

**UC-08: Xem các đăng ký của tôi**

* **Mục tiêu:** Cung cấp danh sách đăng ký và tổng tín chỉ của SV.
* **Tác nhân:** Sinh viên.
* **Tiền điều kiện:** Đăng nhập.
* **Kích hoạt:** Mở MyRegistrations.
* **Hậu điều kiện:** Hiển thị danh sách + tổng tín chỉ theo trạng thái được tính.
* **Luồng chính:** GET /api/registrations?studentId=me → render; bộ lọc client theo status.
* **Ngoại lệ:** Không dữ liệu; lỗi mạng.
* **BR:** Tổng tín chỉ chỉ tính approved (hoặc cấu hình khác).
* **AC:** Khi đổi bộ lọc, danh sách cập nhật trong <300ms (client).

**UC-09: Tạo học phần**

* **Mục tiêu:** Thêm lớp/học phần mới vào kỳ phù hợp.
* **Tác nhân:** Admin/GV (được phân quyền).
* **Tiền điều kiện:** Có học kỳ *active*; dữ liệu tối thiểu sẵn có.
* **Kích hoạt:** Nút **Thêm học phần** tại CourseManagement.
* **Hậu điều kiện:** Học phần mới sẵn sàng để SV đăng ký (khi cửa sổ mở).
* **Luồng chính:**
  1. Nhập form: courseCode, courseName, credits, department, yearLevel, schedule[], room, capacity, teachers[], semesterId.
  2. Server kiểm tra: mã duy nhất, capacity≥0, lịch hợp lệ.
  3. Lưu DB; trả về đối tượng mới.
* **Ngoại lệ:** Trùng mã; capacity<0; lịch sai định dạng.
* **BR:** Không được tạo lớp ngoài kỳ cho phép (hoặc phải gắn rõ semesterId).
* **AC:** Tạo xong hiển thị trong danh sách quản trị và trang *Courses* (nếu publish=true).

**UC-10: Cập nhật học phần**

* **Mục tiêu:** Sửa thông tin học phần hiện hữu.
* **Tác nhân:** Admin/GV (được phân quyền).
* **Tiền điều kiện:** Học phần tồn tại.
* **Kích hoạt:** Hành động **Sửa**.
* **Hậu điều kiện:** Dữ liệu mới có hiệu lực.
* **Luồng chính:** PUT /api/courses/:id với các trường thay đổi.
* **Ngoại lệ:** Giảm capacity < approved; lịch gây xung đột phòng.
* **BR:** Không giảm capacity thấp hơn số đã *approved*; thay đổi lịch phải thông báo SV.
* **AC:** Sau cập nhật, SV/ GV thấy lịch mới; đăng ký không bị mất dữ liệu.

**UC-11: Xóa học phần**

* **Mục tiêu:** Gỡ bỏ học phần (xóa mềm khuyến nghị).
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Không có ràng buộc hoạt động (hoặc cho phép hủy lớp theo chính sách).
* **Kích hoạt:** **Xóa** tại CourseManagement.
* **Hậu điều kiện:** Học phần ẩn khỏi tìm kiếm; SV được thông báo.
* **Luồng chính:** Đặt status=archived (soft delete) qua PUT hoặc DELETE nếu chưa có đăng ký.
* **Ngoại lệ:** Đã có SV *approved* → buộc hủy lớp, hoàn tiền theo chính sách (nếu có).
* **BR:** Gửi thông báo đến SV đã đăng ký; lưu lý do hủy.
* **AC:** Học phần không còn hiển thị trong Courses.

**UC-12: Quản lý giảng viên phụ trách học phần**

* **Mục tiêu:** Gán/đổi giảng viên cho học phần.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Giảng viên tồn tại, role=teacher.
* **Kích hoạt:** Chọn GV trong CourseManagement.
* **Hậu điều kiện:** Quan hệ course–teacher cập nhật.
* **Luồng chính:** GET /api/users/teachers → chọn → PUT /api/courses/:id cập nhật teachers[].
* **Ngoại lệ:** GV không hợp lệ; gv trùng lặp.
* **BR:** Ít nhất 1 GV phụ trách.
* **AC:** CourseDetail hiển thị GV mới.

**UC-13: Duyệt/Phê duyệt đăng ký**

* **Mục tiêu:** Xác nhận yêu cầu pending thành approved.
* **Tác nhân:** Admin/(GV nếu chính sách cho phép).
* **Tiền điều kiện:** Đăng ký ở trạng thái pending.
* **Kích hoạt:** Nút **Approve** tại RegistrationManagement.
* **Hậu điều kiện:** status=approved; cập nhật enrolled.
* **Luồng chính:**
  1. Lọc pending.
  2. Chọn một/multiple registrations.
  3. Server re-validate capacity/cửa sổ/trùng lịch/tín chỉ.
  4. Nếu đạt → cập nhật trạng thái; gửi thông báo cho SV.
* **Ngoại lệ:** Hết chỗ; phát hiện clash; quá cửa sổ → reject kèm mã lỗi tương ứng.
* **BR:** Duyệt hàng loạt phải đảm bảo **tính nguyên tử** từng bản ghi; ghi audit.
* **AC:** approved tăng; capacity giảm tương ứng.

**UC-14: Từ chối đăng ký**

* **Mục tiêu:** Đánh dấu thất bại kèm lý do.
* **Tác nhân:** Admin/(GV).
* **Tiền điều kiện:** Đăng ký pending.
* **Kích hoạt:** **Reject** tại RegistrationManagement.
* **Hậu điều kiện:** status=rejected; lưu reason.
* **Ngoại lệ:** —
* **BR:** Lý do bắt buộc (≥10 ký tự); thông báo SV.
* **AC:** Bản ghi hiển thị ở tab *rejected* với lý do.

**UC-15: Cập nhật trạng thái đăng ký (hoàn thành/hủy…)**

* **Mục tiêu:** Cập nhật completed, dropped, v.v. theo tiến trình học.
* **Tác nhân:** Admin/(GV).
* **Tiền điều kiện:** Đăng ký tồn tại.
* **Kích hoạt:** Hành động cập nhật trạng thái.
* **Hậu điều kiện:** Trạng thái phản ánh đúng thực tế; có log.
* **Luồng chính:** PUT/PATCH /api/registrations/:id với status mục tiêu; kiểm tra hợp lệ theo state machine.
* **Ngoại lệ:** Chuyển trái phép (ví dụ từ rejected → approved không qua quy trình lại).
* **BR:** Biểu đồ trạng thái cho phép: pending→approved|rejected; approved→completed|dropped.
* **AC:** Không thể cập nhật sang trạng thái không hợp lệ (HTTP 409).

**UC-16: Tạo học kỳ**

* **Mục tiêu:** Khởi tạo học kỳ mới với cửa sổ đăng ký.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Không chồng chéo với kỳ khác cùng năm học.
* **Kích hoạt:** **Thêm học kỳ** tại SemesterManagement.
* **Hậu điều kiện:** Kỳ mới có thể được đặt *current*.
* **Luồng chính:** POST /api/semesters {name, code, academicYear, semesterNumber, registrationStart, registrationEnd, isCurrent?}.
* **Ngoại lệ:** registrationStart≥registrationEnd; code trùng.
* **BR:** code duy nhất; ngày giờ theo timezone hệ thống.
* **AC:** Sau tạo, GET /api/semesters trả về bản ghi mới.

**UC-17: Cập nhật học kỳ**

* **Mục tiêu:** Điều chỉnh thông tin/cửa sổ đăng ký của kỳ.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Học kỳ tồn tại.
* **Kích hoạt:** **Sửa** tại SemesterManagement.
* **Hậu điều kiện:** Thông tin kỳ cập nhật.
* **Luồng chính:** PUT /api/semesters/:id.
* **Ngoại lệ:** Thu hẹp cửa sổ gây ảnh hưởng đăng ký đang diễn ra → yêu cầu xác nhận mạnh.
* **BR:** Khi thay đổi isCurrent, đảm bảo chỉ một kỳ current=true.
* **AC:** GET /api/semesters/current phản ánh đúng sau cập nhật.

**UC-18: Đóng/Mở học kỳ, thiết lập cửa sổ đăng ký**

* **Mục tiêu:** Bật/tắt khả năng đăng ký cho kỳ hiện hành.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Có kỳ được đánh dấu current.
* **Kích hoạt:** Toggle trên SemesterManagement.
* **Hậu điều kiện:** Cửa sổ áp dụng ngay cho UC-07/16.
* **Luồng chính:** Cập nhật thuộc tính thời gian/ trạng thái mở-đóng.
* **Ngoại lệ:** —
* **BR:** current là duy nhất; khi mở kỳ mới phải đóng kỳ cũ.
* **AC:** Nút **Đăng ký** bị vô hiệu hóa khi ngoài cửa sổ.

**UC-19: Xem học kỳ hiện hành**

* **Mục tiêu:** Cho client và server biết kỳ đang áp dụng quy tắc đăng ký.
* **Tác nhân:** All/Hệ thống.
* **Tiền điều kiện:** —
* **Kích hoạt:** Tải CourseDetail/Dashboard.
* **Hậu điều kiện:** Nhận đối tượng semester hiện hành.
* **Luồng chính:** GET /api/semesters/current.
* **AC:** Trả về 200 với bản ghi duy nhất hoặc 204 nếu chưa cấu hình.

**UC-20: Xem chi tiết người dùng**

* **Mục tiêu:** Xem hồ sơ chi tiết từng user.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Người dùng tồn tại.
* **Kích hoạt:** Nhấn vào hàng/biểu tượng xem chi tiết.
* **Hậu điều kiện:** Modal chi tiết hiển thị.
* **Luồng chính:** GET /api/users/:id.
* **BR:** Che dữ liệu riêng tư; chỉ hiển thị trường cần thiết.
* **AC:** Thông tin khớp với thực tế; audit hiển thị (nếu có).

**UC-21: Gán/đổi vai trò người dùng (tuỳ chính sách)**

* **Mục tiêu:** Thay đổi vai trò tài khoản.
* **Tác nhân:** Admin.
* **Tiền điều kiện:** Người dùng tồn tại; không phải admin cuối cùng.
* **Kích hoạt:** Sửa role và **Lưu**.
* **Hậu điều kiện:** Vai trò cập nhật, session có thể bị vô hiệu để user đăng nhập lại.
* **Luồng chính:** PUT /api/users/:id {role}.
* **Ngoại lệ:** Cố hạ quyền admin cuối cùng → chặn.
* **BR:** Yêu cầu 2FA/xác nhận lại mật khẩu admin; lưu audit.
* **AC:** Quyền trên UI thay đổi theo role mới sau khi login lại.

**UC-22: Dashboard Sinh viên**

* **Mục tiêu:** Cung cấp bức tranh tổng quan cho SV.
* **Tác nhân:** Sinh viên.
* **Nội dung:** Tổng tín chỉ đã duyệt, đăng ký gần đây, cảnh báo trùng lịch/chưa đủ tiên quyết, kỳ hiện hành.
* **Luồng dữ liệu:** GET /api/registrations?studentId=me, GET /api/semesters/current.
* **BR:** Chỉ hiển thị dữ liệu của chính SV; **không** lộ thông tin người khác.
* **AC:** Số liệu trùng khớp với MyRegistrations.

**UC-23: Dashboard Giảng viên**

* **Mục tiêu:** Tổng quan lớp được phân công và tình trạng đăng ký.
* **Tác nhân:** Giảng viên.
* **Nội dung:** Danh sách lớp mình dạy, số SV đăng ký/approved, thông tin kỳ.
* **Luồng dữ liệu:** GET /api/courses/teacher/:id, GET /api/registrations?courseId=....
* **BR:** Chỉ thấy lớp mình phụ trách.
* **AC:** Số approved khớp với trang quản trị đăng ký.

**UC-24: Dashboard Quản trị**

* **Mục tiêu:** Thống kê hệ thống: người dùng, học phần, đăng ký.
* **Tác nhân:** Admin.
* **Nội dung:** Tổng số user theo vai trò, số học phần theo khoa/kỳ, số đăng ký theo trạng thái, top lớp đầy chỗ.
* **Luồng dữ liệu:** GET /api/users, GET /api/courses, GET /api/registrations.
* **BR:** Có thể cần cache/ETL đối với thống kê nặng.
* **AC:** Các số liệu khớp với từng module nguồn.

**2.3. Yêu cầu phi chức năng**

**2.3.1. Hiệu năng**

* **Tải dữ liệu song song**: Màn hình tổng quan gọi nhiều API đồng thời bằng Promise.all → giảm độ trễ khi hợp nhất số liệu.
* **Trạng thái tải rõ ràng**: Nhiều trang có cờ loading và hiển thị **spinner** trong lúc gọi API; các nút có trạng thái **disabled** khi đang xử lý.
* **Lọc/tìm kiếm phía client**: Danh sách (khóa học, đăng ký, người dùng) có tìm kiếm/lọc trực tiếp trên dữ liệu đã tải.
* **Chưa thấy**: cơ chế **phân trang (pagination)**, **debounce** tìm kiếm, **cache**/prefetch, hay bất kỳ ràng buộc mục tiêu **“< 2s”** trong mã giao diện.

**2.3.2. Bảo mật (phân quyền)**

* **Ngữ cảnh đăng nhập**: Có trang **đăng nhập**, dùng useAuth() để thực hiện login.
* **Phân quyền hiển thị theo vai trò**: UI điều kiện hóa theo role (**student / teacher / admin**).
  + Sinh viên mới thấy nút **đăng ký/hủy đăng ký** học phần.
  + Quản trị/giảng viên mới thấy các thao tác **phê duyệt/từ chối** đăng ký, quản trị học kỳ/khóa học/người dùng.
* **Đổi mật khẩu**: Trang hồ sơ cho phép **đổi mật khẩu**, kiểm tra tối thiểu độ dài và **xác nhận mật khẩu** khớp.
* **Chưa thấy**: chặn truy cập theo vai trò ở tầng route hay kiểm tra phân quyền phía server trong phần giao diện này (UI chỉ ẩn/hiện nút; không khẳng định kiểm soát truy cập phía backend).

**2.3.3. Khả năng sử dụng (UI/UX)**

* **Phản hồi thao tác**: Dùng **toast** báo lỗi/thành công khi tải dữ liệu, lưu thay đổi, phê duyệt/từ chối, đăng ký/hủy đăng ký.
* **Trạng thái rỗng & cảnh báo**: Có thông điệp khi **không có dữ liệu** hoặc khóa học không tồn tại/không hoạt động.
* **Biểu mẫu thân thiện**:
  + **Xác thực client**: yêu cầu bắt buộc, tối thiểu độ dài, mẫu năm học (YYYY-YYYY), thông báo lỗi cạnh trường.
  + **Hiển thị/ẩn mật khẩu** cho đăng nhập/đăng ký/đổi mật khẩu.
* **Khả năng thao tác**: có **tìm kiếm**, **lọc** (ví dụ lọc ngành, giảng viên, trạng thái), các **nút hành động** rõ ràng; bố cục **responsive** với lưới; sử dụng **icon** nhất quán.
* **Chưa thấy**: thuộc tính **ARIA**/phím tắt, hướng dẫn truy cập (accessibility) chuyên sâu.

**2.4. Đặc tả luồng dữ liệu (DFD)**

**2.4.1. Phạm vi & Tác nhân**

* **Tác nhân**: Sinh viên, Quản trị viên (GV dùng vai trò teacher, hiện UI chưa có thao tác riêng).
* **Chức năng chính (từ code pages/\*.js)**  
  Duyệt/Tìm kiếm học phần, Xem chi tiết, Đăng ký–Hủy; Quản trị Học phần, Học kỳ, Người dùng, Duyệt đơn; Dashboard thống kê.

**2.4.2. Kho dữ liệu (Data Stores)**

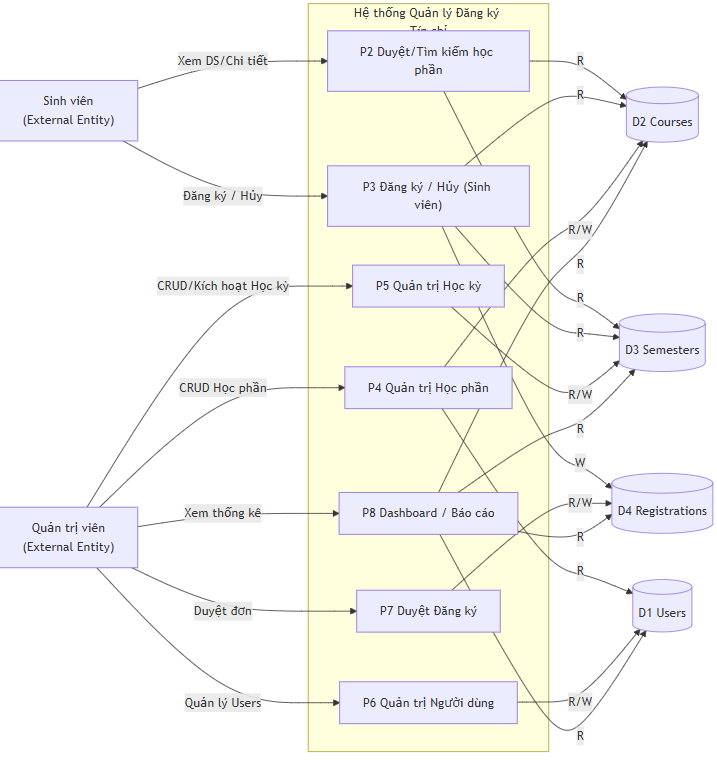
* **D1 – Users**: \_id, firstName, lastName, email, role, isActive, studentId?, major?, year?, semester?, currentCredits?, maxCredits?, phone?, address?, dateOfBirth?
* **D2 – Courses**: \_id, courseCode, courseName, credits, description?, department?, major?, yearLevel?, semesterNumber?, maxStudents, teacher(ref User), semester(ref Semester), registrationDeadline?, withdrawalDeadline?, schedule{dayOfWeek,startTime,endTime,room}, courseType
* **D3 – Semesters**: \_id, name, code, academicYear, semesterNumber, startDate, endDate, registrationStartDate?, registrationEndDate?, withdrawalDeadline?, maxCreditsPerStudent?, minCreditsPerStudent?, description?, isActive
* **D4 – Registrations**: \_id, student(ref User), course(ref Course), semester(ref Semester), status(pending|approved|rejected|dropped|completed), reason?, createdAt, updatedAt

**2.4.3. Tiến trình (Processes) ↔ API (rút từ UI)**

* **P1 Auth/Profile** *(ẩn trong code trang khác)*: đăng nhập/đăng ký, xem hồ sơ.
* **P2 Duyệt học phần**: GET /api/courses?…, GET /api/courses/:id, GET /api/semesters/current.
* **P3 Đăng ký/Hủy (SV)**: POST /api/registrations, PUT /api/registrations/:id/drop, GET /api/registrations.
* **P4 Quản trị Học phần**: GET/POST/PUT/DELETE /api/courses, GET /api/users/teachers, GET /api/semesters.
* **P5 Quản trị Học kỳ**: GET/POST/PUT /api/semesters, PUT /api/semesters/:id/activate.
* **P6 Quản trị Người dùng**: GET /api/users, PUT /api/users/:id, GET /api/users/students/:id.
* **P7 Duyệt Đăng ký**: GET /api/registrations, PUT /api/registrations/:id/approve|reject.
* **P8 Dashboard**: tổng hợp users/courses/registrations/semesters.

**2.4.4. DFD Level 0 (ngắn gọn)**

* **SV** → (P2) xem Courses (R **D2**, **D3**) → (P3) đăng ký/hủy (R **D2/D3**, R/W **D4**).
* **Admin** → (P4) CRUD Courses (R/W **D2**, R **D1**) ; (P5) CRUD/Activate Semesters (R/W **D3**) ; (P6) quản lý Users (R/W **D1**) ; (P7) duyệt Registrations (R/W **D4**).
* **Dashboard (P8)** đọc **D1–D4** để thống kê.



*Hình 5. DFD level 0*

**2.4.5. Ma trận CRUD (tóm tắt)**

* **P2**: R **D2/D3** | **P3**: R **D2/D3**, R/W **D4**
* **P4**: R/W **D2**, R **D1** | **P5**: R/W **D3**
* **P6**: R/W **D1** | **P7**: R/W **D4** | **P8**: R **D1–D4**

**2.4.6. Luồng điển hình**

* **SV đăng ký học phần**: xem chi tiết (P2) → lấy học kỳ hiện hành → POST /registrations (P3) ⇒ ghi **D4** trạng thái pending.
* **Admin duyệt**: PUT /registrations/:id/approve|reject (P7) ⇒ cập nhật **D4** (và đồng bộ số lượng lớp nếu backend hỗ trợ).

**2.4.7. Ràng buộc chính (tối thiểu)**

* Mỗi student–course–semester chỉ có **1** đăng ký mở.
* semester.isActive là duy nhất.
* Duy trì tính toàn vẹn maxStudents, max/minCreditsPerStudent, deadline đăng ký/rút.

**Chương 3. Thiết kế hệ thống**

**3.1. Thiết kế kiến trúc**

**3.1.1. Lựa chọn mô hình kiến trúc**

**Kiến trúc N-Tier (3 lớp)**:

* **UI (Client/Web App)**: Ứng dụng React (thư mục client/src) hiển thị màn hình đăng nhập/đăng ký, danh sách học phần, chi tiết học phần, đăng ký tín chỉ, trang quản trị (quản lý môn học, học kỳ, người dùng, duyệt đăng ký…).
* **Service/API (Server)**: REST API xây dựng bằng **Node.js + Express** (server.js, routes/\*.js, middleware/auth.js) triển khai các nghiệp vụ: xác thực, tra cứu học phần, kiểm tra điều kiện & xung đột, ghi nhận đăng ký, quản trị học kỳ…
* **Data (Repository/DB)**: **MongoDB** (Mongoose Model trong models/\*.js: User, Course, Semester, Registration) lưu người dùng, học phần, học kỳ, và các phiếu đăng ký.

**Vì sao phù hợp?**

* **Phân tách rõ ràng** giao diện – nghiệp vụ – dữ liệu → dễ bảo trì/mở rộng.
* **Phù hợp đặc thù CRUD + kiểm tra quy tắc** của bài toán đăng ký tín chỉ (REST chuẩn mực, middleware dễ cấy thêm rule).
* **Khả năng mở rộng**: có thể scale theo chiều ngang từng lớp (SPA tĩnh, API nhiều instance, MongoDB Replica Set/Sharding khi tải cao).
* **Bảo mật**: dùng **JWT** (thấy trong middleware/auth.js) để bảo vệ các route; dễ áp dụng RBAC (student/admin) cho endpoint.

**Note\***

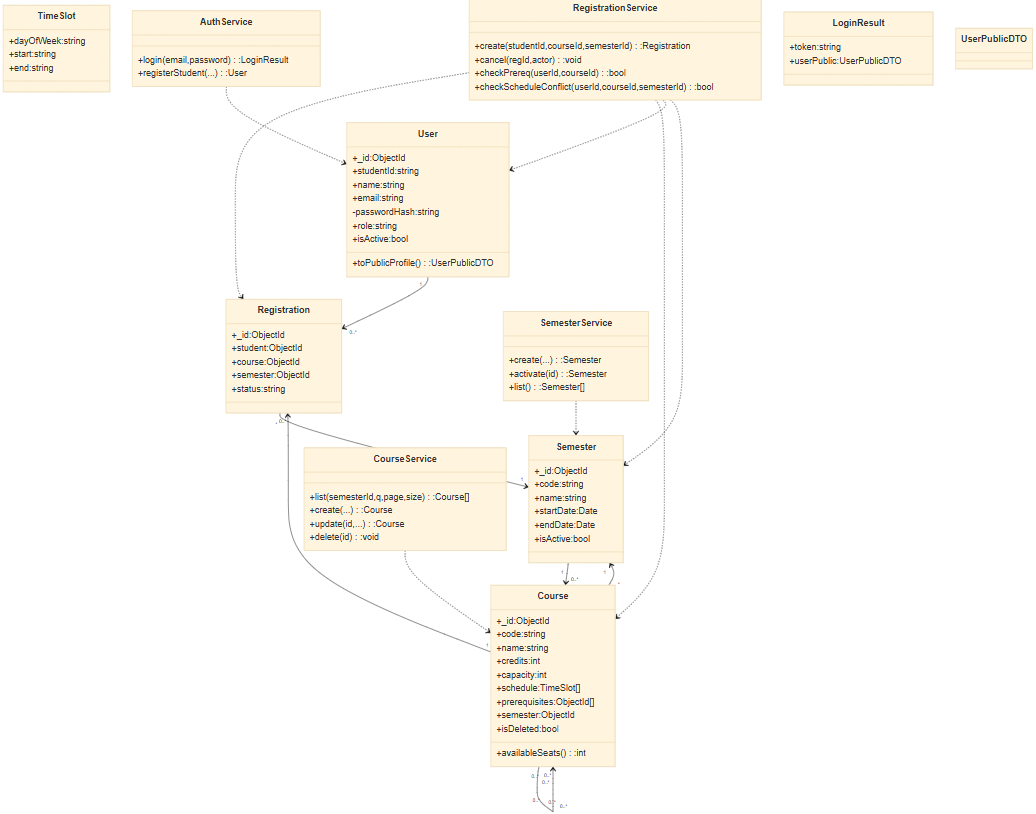
Hệ thống hiện là **micro-frontend/monolith API**. Nếu tải rất lớn theo học kỳ, có thể tiến hóa thành **microservices** (ví dụ tách CourseService, RegistrationService, UserService) nhưng hiện tại monolith API giúp **đơn giản triển khai** và **giảm độ phức tạp vận hành**.

**3.1.2. Mô hình triển khai (gợi ý)**

* **Client** build tĩnh (SPA) phục vụ qua CDN/Nginx.
* **API** chạy Node.js (PM2/Docker) sau Nginx (reverse proxy, TLS).
* **MongoDB** chạy Replica Set (3 node) để tăng sẵn sàng; backup theo lịch.
* **Biến môi trường**: config.env, client/.env; tách khóa JWT, URI Mongo.

**3.2. Thiết kế chi tiết (UML)**

**3.2.1. Sơ đồ lớp (Class Diagram)**



*Hình 6: Sơ đồ lớp*

**(Sơ đồ lớp UML)** mô tả cấu trúc lớp của hệ thống đăng ký tín chỉ: các lớp/DTO, thuộc tính, phương thức và quan hệ phụ thuộc/liên kết giữa chúng

**3.2.1.1. Các nhóm lớp**

* **Domain entities (thực thể miền):**
  + **User**  
    Thuộc tính: \_id:ObjectId, studentId:string, name:string, email:string, passwordHash:string, role:string, isActive:bool  
    Phương thức: toPublicProfile():UserPublicDTO
  + **Course**  
    Thuộc tính: \_id:ObjectId, code:string, name:string, credits:int, capacity:int, schedule:TimeSlot[], prerequisites:ObjectId[], semester:ObjectId, isDeleted:bool  
    Phương thức: availableSeats():int
  + **Semester**  
    Thuộc tính: \_id:ObjectId, code:string, name:string, startDate:Date, endDate:Date, isActive:bool
  + **Registration**  
    Thuộc tính: \_id:ObjectId, student:ObjectId, course:ObjectId, semester:ObjectId, status:string
  + **TimeSlot**  
    Thuộc tính: dayOfWeek:string, start:string, end:string
* **Service layer (dịch vụ nghiệp vụ):**
  + **AuthService**  
    login(email,password):LoginResult • registerStudent(...):User
  + **CourseService**  
    list(semesterId,q,page,size):Course[] • create(...):Course • update(id,...):Course • delete(id):void
  + **SemesterService**  
    create(...):Semester • activate(id):Semester • list():Semester[]
  + **RegistrationService**  
    create(studentId,courseId,semesterId):Registration •  
    cancel(regId,actor):void •  
    checkPrereq(userId,courseId):bool •  
    checkScheduleConflict(userId,courseId,semesterId):bool
* **DTO/Result (đối tượng trao đổi dữ liệu):**
  + **LoginResult**: token:string, userPublic:UserPublicDTO
  + **UserPublicDTO**: (thông tin công khai của User)

**3.2.1.2. Các quan hệ chính (mũi tên/đường nối trong sơ đồ)**

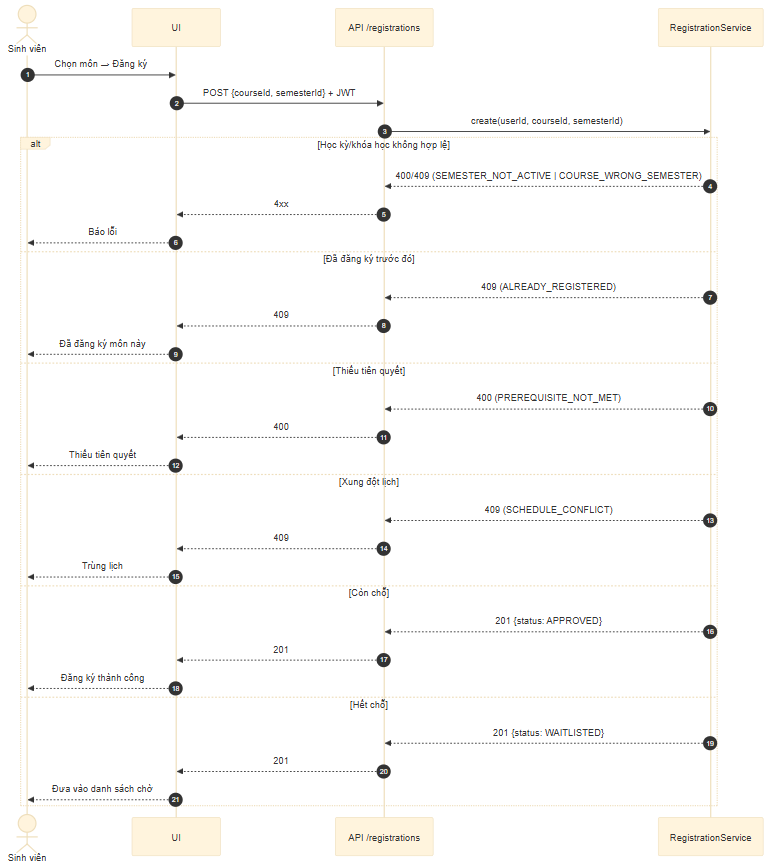
* **Registration ↔ User/Course/Semester:** Registration liên kết đến 3 thực thể này (đăng ký của sinh viên cho một học phần trong một học kỳ).
* **Course ↔ Semester:** Mỗi Course thuộc về một Semester (mũi tên từ Course sang Semester).
* **Course ↔ TimeSlot:** Course có danh sách lịch học schedule: TimeSlot[].
* **Course prerequisites:** prerequisites:ObjectId[] thể hiện quan hệ tiên quyết giữa các Course.
* **Services → Entities/DTOs (đường nét đứt với mũi tên):** các service **phụ thuộc** vào thực thể/DTO tương ứng:
  + AuthService trả về LoginResult, khởi tạo User.
  + CourseService thao tác/hoàn trả Course.
  + SemesterService thao tác/hoàn trả Semester.
  + RegistrationService tạo/hủy Registration, kiểm tra điều kiện trên User/Course/Semester.
* **AuthService → UserPublicDTO:** qua LoginResult.userPublic.

**3.2.1.3. Ý nghĩa thiết kế**

* Phân tách rõ **miền (entities)** và **nghiệp vụ (services)**; service chứa logic như kiểm tra tiên quyết, xung đột lịch, sức chứa; entity giữ dữ liệu và một số hành vi cơ bản (ví dụ Course.availableSeats()).
* DTO tách dữ liệu trả ra client (ẩn passwordHash, …).
* Quan hệ giữa các lớp phản ánh đúng nghiệp vụ: một sinh viên (User) đăng ký (Registration) một học phần (Course) trong học kỳ (Semester), và Course có lịch (TimeSlot) cùng danh sách môn tiên quyết.

**3.2.2. Sơ đồ trình tự (Sequence Diagram)**

**3.2.2.1. Sinh viên đăng ký học phần**



*Hình 7: Sơ đồ trình tự đăng kí học phần*

**Mục tiêu**

Mô tả luồng sinh viên đăng ký học phần và các quyết định chính (APPROVED/WAITLISTED, mã lỗi 4xx).

**Thành phần**

* **SV** (Sinh viên) → **UI** (React) → **API** (POST /registrations) → **RegistrationService** (SVC).
* Xác thực bằng **JWT** (Auth MW ẩn).

**Luồng chính**

1. SV bấm **Đăng ký** trên UI.
2. UI gửi POST /registrations {courseId, semesterId} kèm **Bearer JWT**.
3. API xác thực JWT → lấy userId → gọi SVC.create(userId, courseId, semesterId).
4. SVC kiểm tra theo thứ tự:
   * **Học kỳ active** & **môn thuộc học kỳ**.
   * **Chưa đăng ký trùng** (user–course–semester).
   * **Đủ tiên quyết**.
   * **Không xung đột lịch**.
   * **Sức chứa** của môn.
5. Kết quả:
   * **Còn chỗ** → tạo **APPROVED** → 201.
   * **Hết chỗ** → tạo **WAITLISTED** → 201.

**Nhánh lỗi & mã trả về**

* 400/409 SEMESTER\_NOT\_ACTIVE / COURSE\_WRONG\_SEMESTER
* 409 ALREADY\_REGISTERED
* 400 PREREQUISITE\_NOT\_MET
* 409 SCHEDULE\_CONFLICT (có thể kèm slot)

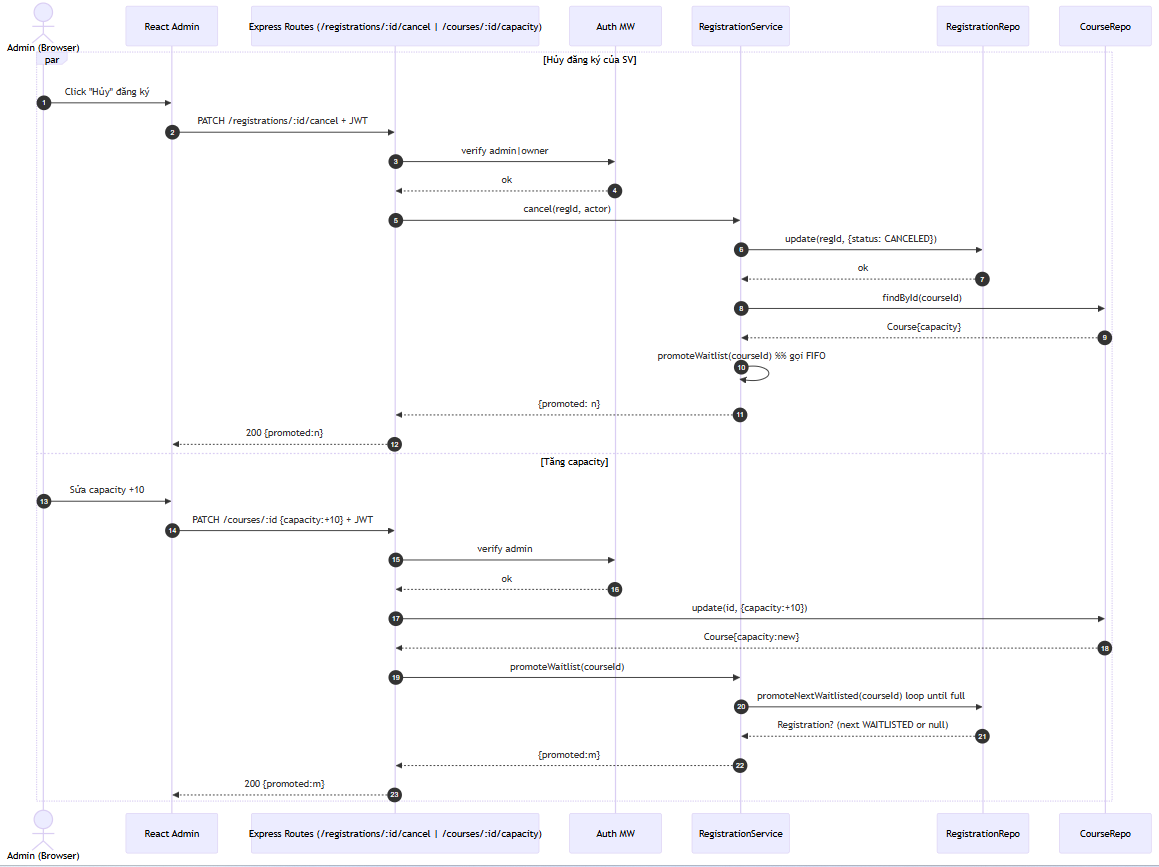
**Quy tắc dữ liệu (tóm tắt)**

* semester.isActive = true
* course.semester == semesterId
* Không cho phép trùng đăng ký cùng bộ khóa.
* Lịch không trùng dayOfWeek + time range.
* approvedCount < capacity → **APPROVED**, ngược lại **WAITLISTED**.

**Bảo mật & UX**

* Chỉ dùng userId từ JWT; không nhận userId trong payload.
* UI hiển thị thông điệp theo mã lỗi hoặc trạng thái trả về.

**3.2.2.2. Tự động chuyển trạng thái đăng ký**



*Hình 8: Sơ đồ trình tự tự đông chuyển trạng thái đăng kí*

**Mục tiêu**

Tự động nâng SV từ **WAITLISTED → APPROVED** khi có chỗ trống do **hủy đăng ký** hoặc **tăng capacity** của học phần.

**Thành phần**

* Admin (Browser) → UI (React) → API (Express)
  + PATCH /registrations/:id/cancel
  + PATCH /courses/:id/capacity
* Auth MW (JWT, kiểm tra admin/owner)
* RegistrationService (RegSvc), RegistrationRepo (RegRepo), CourseRepo

**Luồng chính**

1. **Hủy đăng ký của SV**
2. Admin bấm “Hủy” trên UI.
3. UI gọi PATCH /registrations/:id/cancel kèm Bearer JWT.
4. API → Auth MW xác thực **admin hoặc chủ sở hữu** → OK.
5. API gọi RegSvc.cancel(regId, actor).
6. RegSvc cập nhật đăng ký sang **CANCELED**.
7. RegSvc đọc course liên quan → **promoteWaitlist(courseId)** (FIFO).
8. API trả 200 { promoted: n }.
9. **Tăng capacity**
10. Admin nhập tăng capacity:+10 trên UI.
11. UI gọi PATCH /courses/:id { capacity:+10 } kèm JWT.
12. API → Auth MW xác thực **admin** → OK.
13. API cập nhật capacity tại CourseRepo.
14. API gọi RegSvc.promoteWaitlist(courseId) (lặp **FIFO** cho đến khi đầy/ hết chờ).
15. API trả 200 { promoted: m }.

**Nhánh lỗi & mã trả về**

* 401 UNAUTHENTICATED (không có/ JWT sai).
* 403 FORBIDDEN (không phải admin; hủy không phải **owner** cũng không phải **admin**).
* 404 NOT\_FOUND (registration/course không tồn tại).
* 409 CONFLICT
  + REG\_ALREADY\_CANCELED (đã CANCELED).
  + COURSE\_CAPACITY\_INVALID (delta không hợp lệ/ capacity âm).
  + COURSE\_INACTIVE\_OR\_MISMATCH (khi kiểm tra sau hủy).
* 422 UNPROCESSABLE\_ENTITY (payload không hợp lệ).

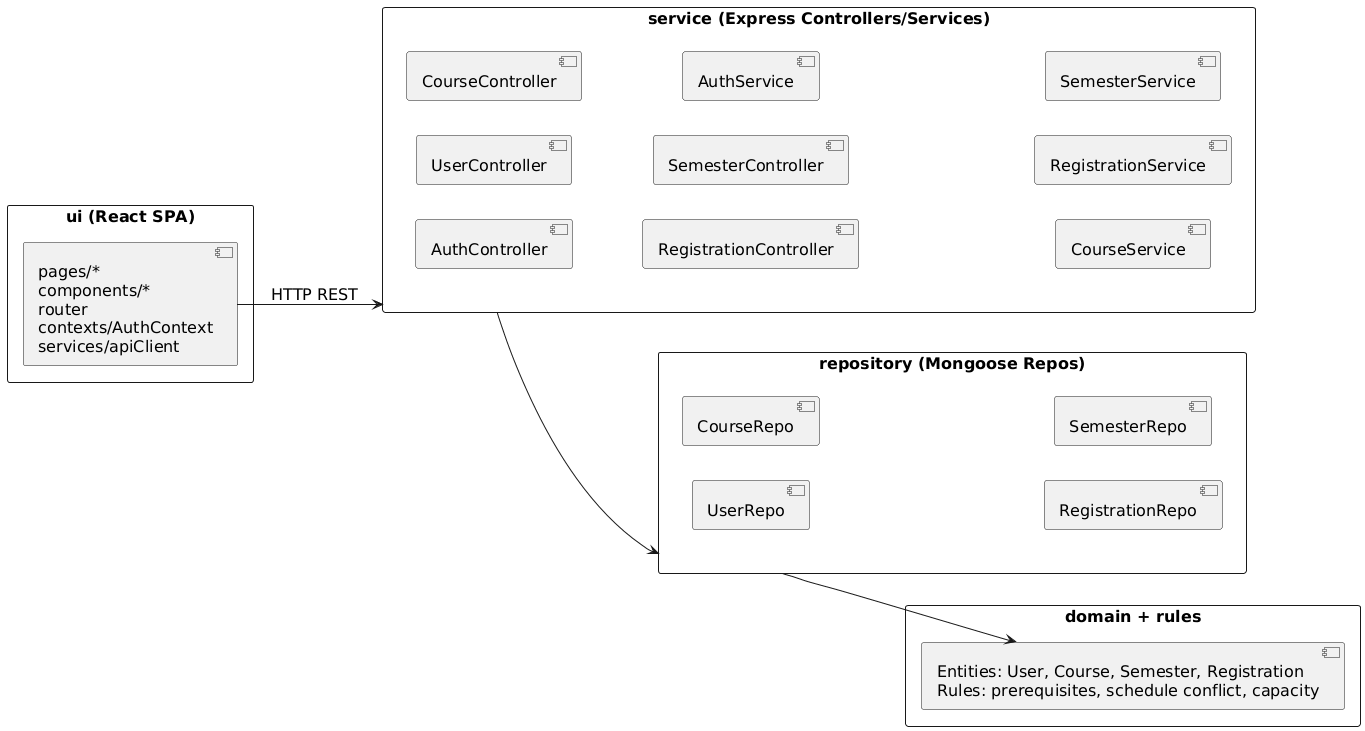
**Quy tắc dữ liệu (tóm tắt)**

* Hủy đăng ký: chuyển trạng thái **CANCELED**, ghi nhận updatedAt, actor.
* Nâng danh sách chờ: **FIFO** theo createdAt; chỉ nâng **WAITLISTED** cùng courseId.
* Dừng nâng khi approvedCount >= capacity hoặc **hết hàng chờ**.
* Tăng capacity: newCapacity = oldCapacity + delta, delta có thể âm/dương nhưng không để capacity < 0.
* Trả về tổng số lượng đã được nâng: promoted.

**Bảo mật & UX**

* Chỉ dùng **userId** từ JWT; không nhận userId trong payload.
* Hủy: chỉ **admin** hoặc **chính SV** được phép. Tăng capacity: **chỉ admin**.
* UI hiển thị số lượng promoted sau thao tác và thông báo theo **mã lỗi**; làm mới danh sách APPROVED/WAITLISTED.
* Log phục vụ audit: actor, regId/courseId, số người được nâng, thời điểm.

**3.2.3. Biểu đồ Gói (Package Diagram)**

****

*Hình 9: Biểu đồ gói*

**Các gói (packages)**

* **ui (React SPA):** pages, components, router, AuthContext, apiClient. Gửi HTTP, mang token, hiển thị kết quả.
* **service (Express Controllers/Services):** Controllers nhận request/validate, gọi **Services** để xử lý **business logic** (prerequisite, trùng lịch, còn chỗ…).
* **repository (Mongoose Repos):** Truy cập DB (CRUD, truy vấn, phân trang, đếm/chờ), **không** chứa nghiệp vụ.
* **domain + rules:** Entities (User, Course, Semester, Registration) và các **rules**: prerequisites, schedule conflict, capacity.

**Luồng chính**

1. **UI → Service (HTTP REST)**: SPA gọi API kèm token.
2. **Service → Repository**: Service áp dụng rules rồi đọc/ghi DB qua Repo.
3. **Repository → Domain**: Repo thao tác entity, trả dữ liệu domain cho Service → UI.

**Ví dụ ngắn (đăng ký môn)**

* UI gọi POST /registrations → Service kiểm tra **prereq**, **xung đột lịch**, **sức chứa** → ghi APPROVED/WAITLISTED → trả về UI.

**Lợi ích thiết kế**

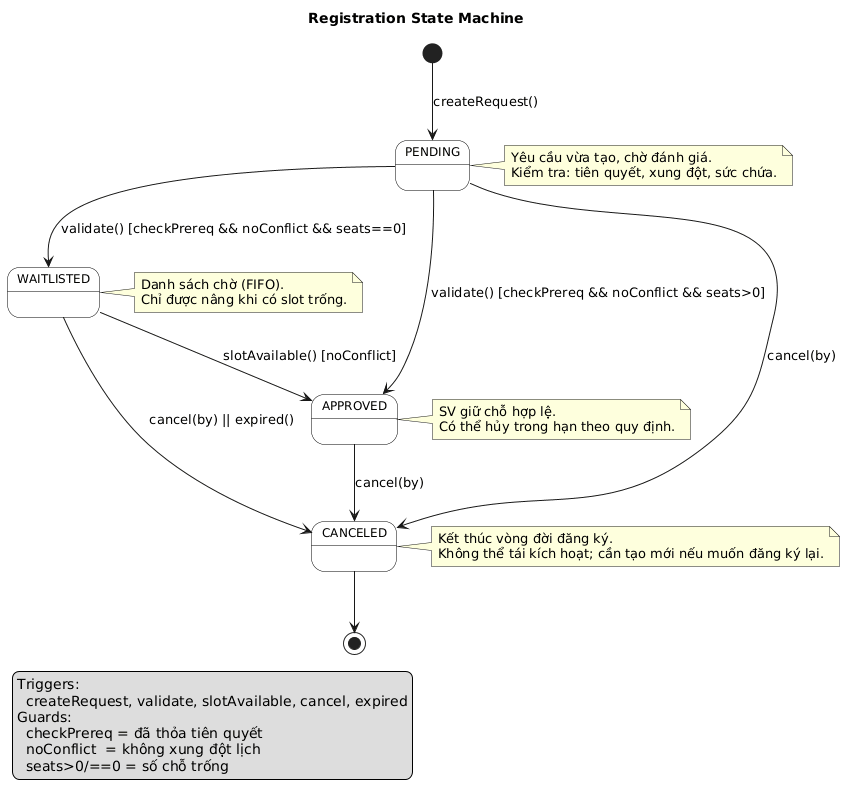
* Tách lớp rõ ràng → dễ bảo trì, test (mock Repo, test rule ở Service).
* Đổi DB/ORM chỉ ảnh hưởng Repo; mở rộng tính năng thêm controller/service/repo mà không chạm UI/domain.

**3.2.4. Biểu đồ Trạng thái (State Machine)**

**3.2.4.1. Registration – vòng đời đăng ký**

**Mô tả ngắn gọn**

* PENDING: vừa tạo yêu cầu đăng ký.
* APPROVED: đủ điều kiện và còn chỗ.
* WAITLISTED: đủ điều kiện nhưng hết chỗ, xếp hàng FIFO.
* CANCELED: sinh viên hủy, admin hủy, hoặc hệ thống rollback.
* Chuyển trạng thái:
  + PENDING → APPROVED khi: checkPrereq && noConflict && seats>0.
  + PENDING → WAITLISTED khi: checkPrereq && noConflict && seats==0.
  + WAITLISTED → APPROVED khi có chỗ (do hủy/tăng chỉ tiêu) **và** vẫn noConflict.
  + \* → CANCELED theo quy định (hạn hủy/actor).

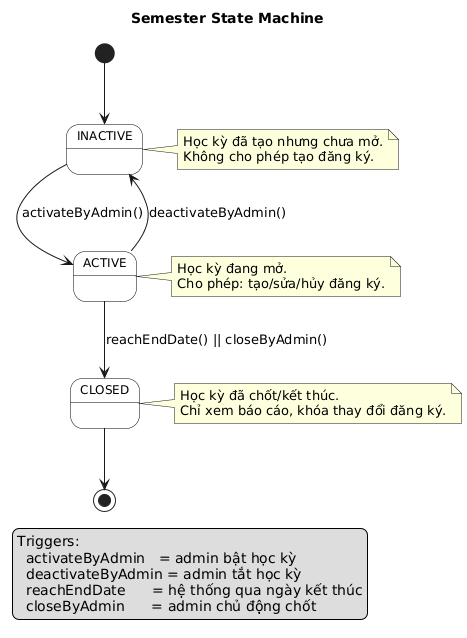


*Hình 10: Biểu đồ trạng thái Registration*

**3.2.4.2. Semester – quản lý mở/đóng học kỳ**

**Mô tả ngắn gọn**

* INACTIVE: tạo xong nhưng chưa mở cho SV đăng ký.
* ACTIVE: đang mở đăng ký/chỉnh sửa.
* CLOSED: đã chốt, chỉ xem báo cáo; cấm thao tác đăng ký.
* Chuyển trạng thái:
  + INACTIVE → ACTIVE: admin “activate”.
  + ACTIVE → INACTIVE: admin “deactivate”.
  + ACTIVE → CLOSED: đến endDate hoặc admin chốt.



*Hình 11: Biểu đồ trạng thái Semester*

**3.2.5. Đặc tả DTO/Contract**

**Auth**

* POST /auth/login  
  **Req** { email, password }  
  **Res** { token, user: { id, name, email, role } }

**Courses**

* GET /courses?semesterId=&q=&page=&size= → phân trang.  
  **Res(item)** { id, code, name, credits, capacity, semester, prerequisites[], schedule[] }
* POST /courses (admin)  
  **Req** CourseCreateDTO = { code, name, credits, capacity, semesterId, prerequisites?: ObjectId[], schedule: [{ dayOfWeek, start, end, room }] }  
  **Res** Course

**Registrations**

* GET /registrations/me?semesterId=  
  **Res(item)** { id, course: { id, code, name }, semester, status, createdAt }
* POST /registrations  
  **Req** { courseId, semesterId }  
  **Res** { id, status } (APPROVED/WAITLISTED)
* PATCH /registrations/:id/cancel → { ok: true }

**Semesters**

* GET /semesters → danh sách
* POST /semesters (admin) → tạo
* PATCH /semesters/:id { isActive }

**3.2.6. Thuật toán & kiểm tra nghiệp vụ (cụ thể hoá để code/test)**

1. **Kiểm tra tiên quyết**
   * Input: userId, courseId.
   * B1: Lấy Course.prerequisites[].
   * B2: Kiểm tra mỗi course tiên quyết có **“đã hoàn thành”** (tuỳ hệ thống: có thể kiểm bằng Registration trạng thái APPROVED + flag completed hoặc bảng điểm).
   * Output: true/false, kèm danh sách môn thiếu.
2. **Xung đột lịch**
   * Input: userId, courseId, semesterId.
   * B1: Lấy schedule[] của courseId.
   * B2: Lấy toàn bộ Registration của SV trong semesterId có status ∈ {APPROVED, WAITLISTED} → với mỗi course đã có, so sánh từng TimeSlot.
   * Hai khoảng **giao nhau** nếu dayOfWeek trùng và max(startA, startB) < min(endA, endB).
   * Output: noConflict | conflict[].
3. **Sức chứa**
   * Input: courseId.
   * B1: enrolled = count(Registration where courseId & status=APPROVED).
   * B2: available = course.capacity - enrolled.
   * Output: { available: available>0, left: max(0, available) }.
4. **Nâng waitlist (FIFO)**
   * Input: courseId.
   * B1: Trong vòng lặp, nếu availableSeats()==0 → dừng.
   * B2: Lấy WAITLISTED cũ nhất (sort createdAt asc limit 1).
   * B3: Re-check **xung đột**; nếu **ok** → set APPROVED và giảm available; nếu **ko** → bỏ qua và lấy người kế tiếp.
   * Output: promotedCount.

**3.2.7. Ràng buộc kiểm thử (testability)**

* **Unit test** cho:
  + TimeSlot.overlaps() (biên: trùng biên, lồng nhau, lệch ngày).
  + checkScheduleConflict() với nhiều lớp, nhiều slot.
  + checkPrereq() với cây tiên quyết N-N.
  + promoteWaitlist() trong các kịch bản: có xung đột phát sinh, tăng capacity, hủy.
* **Integration test** cho luồng POST /registrations (đủ/thiếu tiên quyết, xung đột, hết chỗ).
* **API contract test** cho DTO (mã lỗi & payload).

**3.2.8. Liên kết mã nguồn (mapping nhanh để người đọc dò code)**

* **Domain/Models**: models/User.js, models/Course.js, models/Semester.js, models/Registration.js
* **Middleware**: middleware/auth.js (JWT, RBAC)
* **Routes/Controllers**: routes/auth.js, routes/users.js, routes/courses.js, routes/registrations.js, routes/semesters.js
* **Frontend**: client/src/pages/\* (màn hình SV/Admin), client/src/components/\*, client/src/contexts/AuthContext.js, client/src/services/api.js

**Chương 4. Thiết kế giao diện**

Sử dụng công cụ figma để thiết kế giao diện UI cho hệ thống

**4.1. Mục tiêu & phạm vi**

* Mục tiêu: Thiết kế giao diện cho Hệ thống Quản lý Tín Chỉ (CREDIT) theo hướng dễ dùng, nhất quán và phù hợp bối cảnh sử dụng của sinh viên và quản trị.
* Phạm vi: Bộ Design System (màu, chữ, lưới, component), cấu trúc thông tin, luồng thao tác, màn hình chính và quy tắc tương tác.
* Ghi chú: Do giới hạn thời gian, phần kiểm thử hộp đen sẽ được thực hiện ở giai đoạn sau.

**4.1.1. Mục tiêu UX (Trải nghiệm người dùng)**

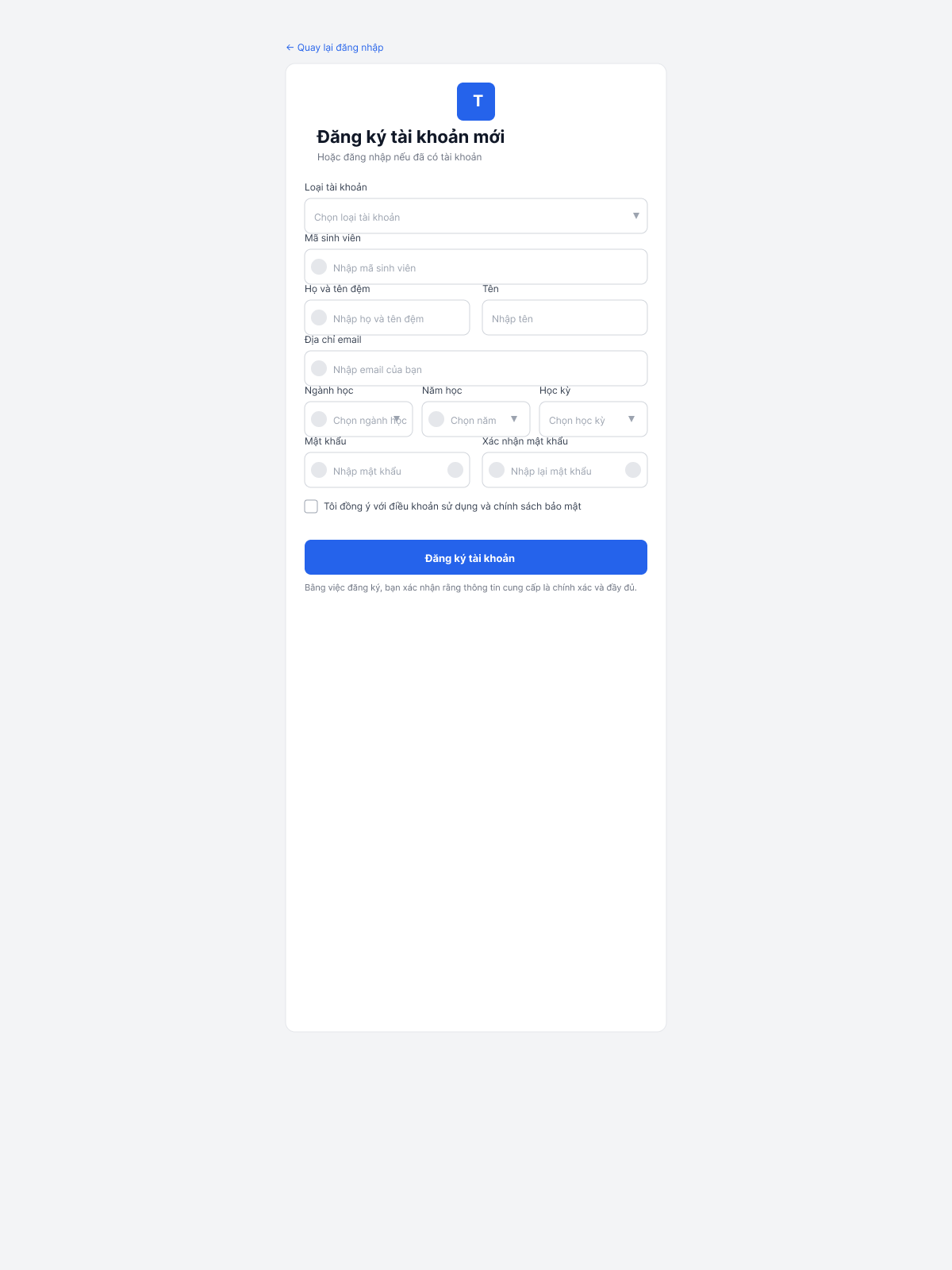
* Rút ngắn thao tác: Hoàn thành tác vụ chính (Đăng ký môn, Hủy môn, Tra cứu TKB) trong ≤ 4 bước.
* Tốc độ hoàn thành: Người dùng mới đăng ký 1 học phần trong ≤ 30–45 giây.
* Dễ tìm chức năng: ≥ 85% người thử nghiệm tìm đúng tính năng ngay lần đầu (không cần hướng dẫn).
* Giảm lỗi thao tác: Tỷ lệ nhập sai/misclick ≤ 5% mỗi phiên; có nhắc lỗi và gợi ý sửa tại chỗ.

**4.1.2. Mục tiêu UI (Giao diện)**

* Nhất quán thị giác: Màu, font, spacing, icon tuân thủ Design System trên 100% màn hình của CREDIT.
* Khả năng truy cập: Tương phản văn bản ≥ 4.5:1; vùng chạm tối thiểu 44×44 px; hỗ trợ bàn phím với focus ring rõ ràng.
* Responsive chuẩn: 12 màn hình trọng yếu hiển thị đúng trên Desktop/Tablet/Mobile (không vỡ layout, không tràn chữ).
* Đủ trạng thái hệ thống: Mỗi màn hình có Loading / Empty / Error / Success theo chuẩn thành phần dùng chung.

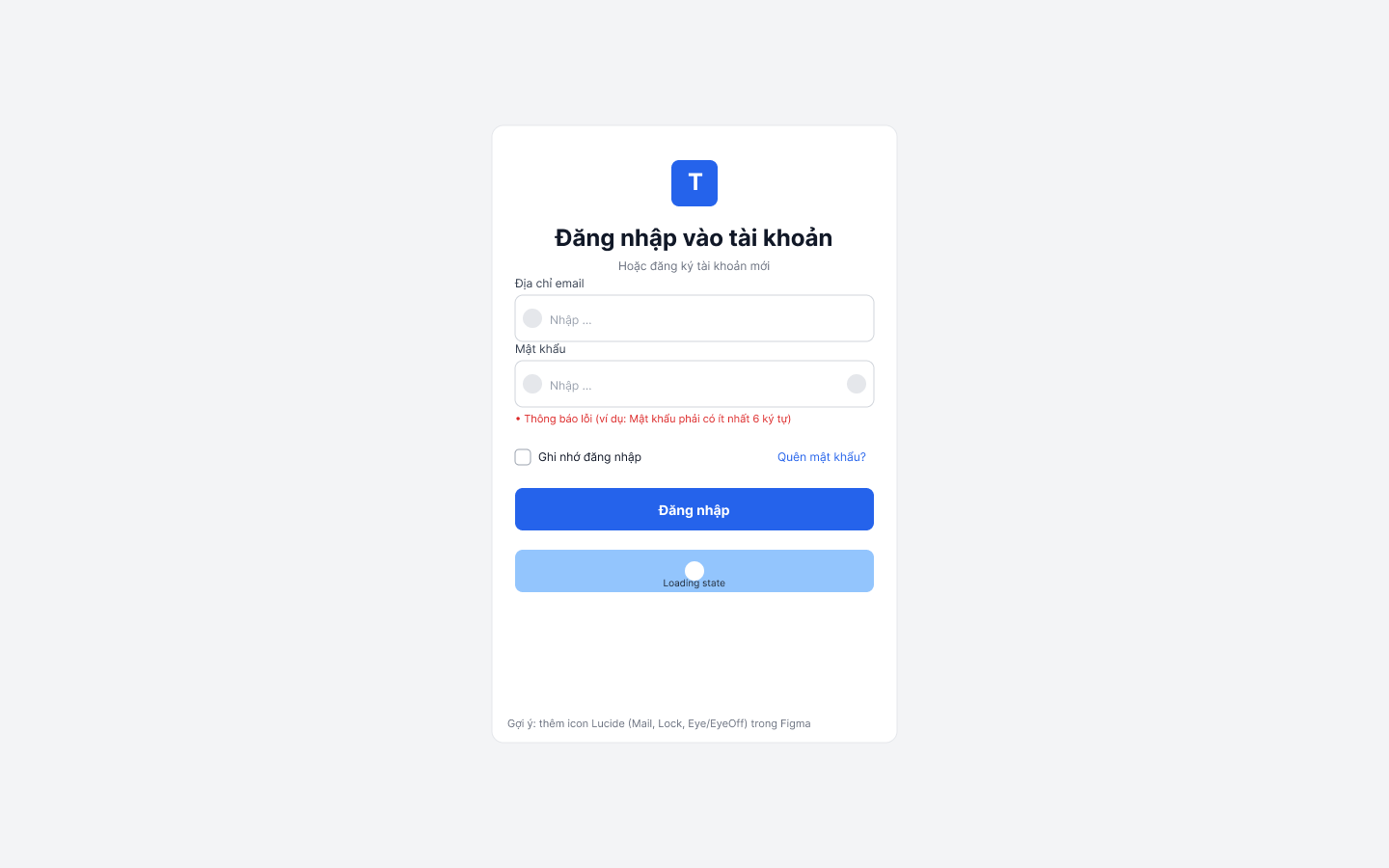
**4.2. Giao diện chương trình (Figma)**

**Trang đăng kí tài khoản**

****

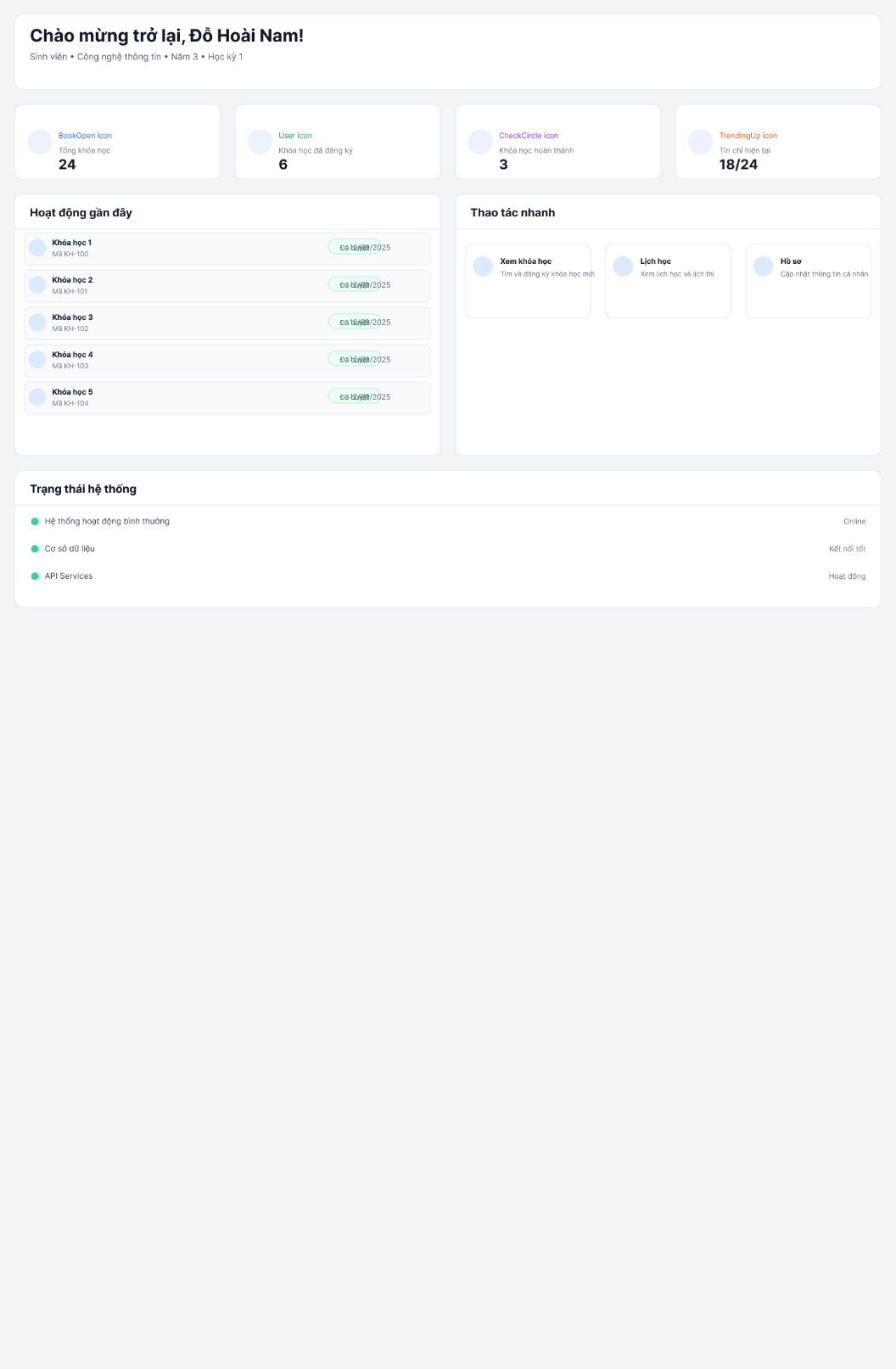
*Hình 12: Giao diện đăng kí tài khoản*

**Trang đăng nhập tài khoản**

****

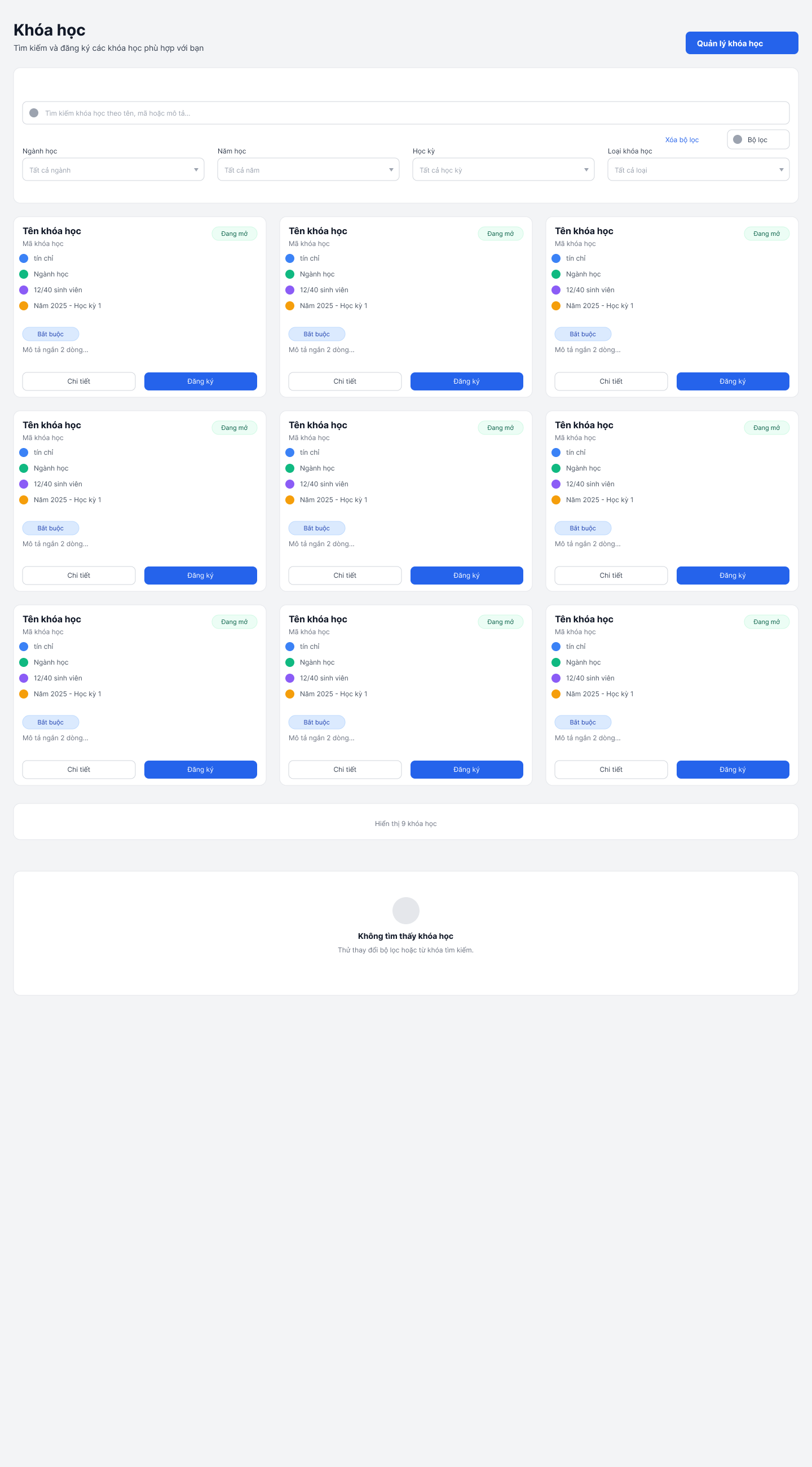
*Hình 13: Giao diện đăng nhập tài khoản*

**Trang dashboard sinh viên**

****

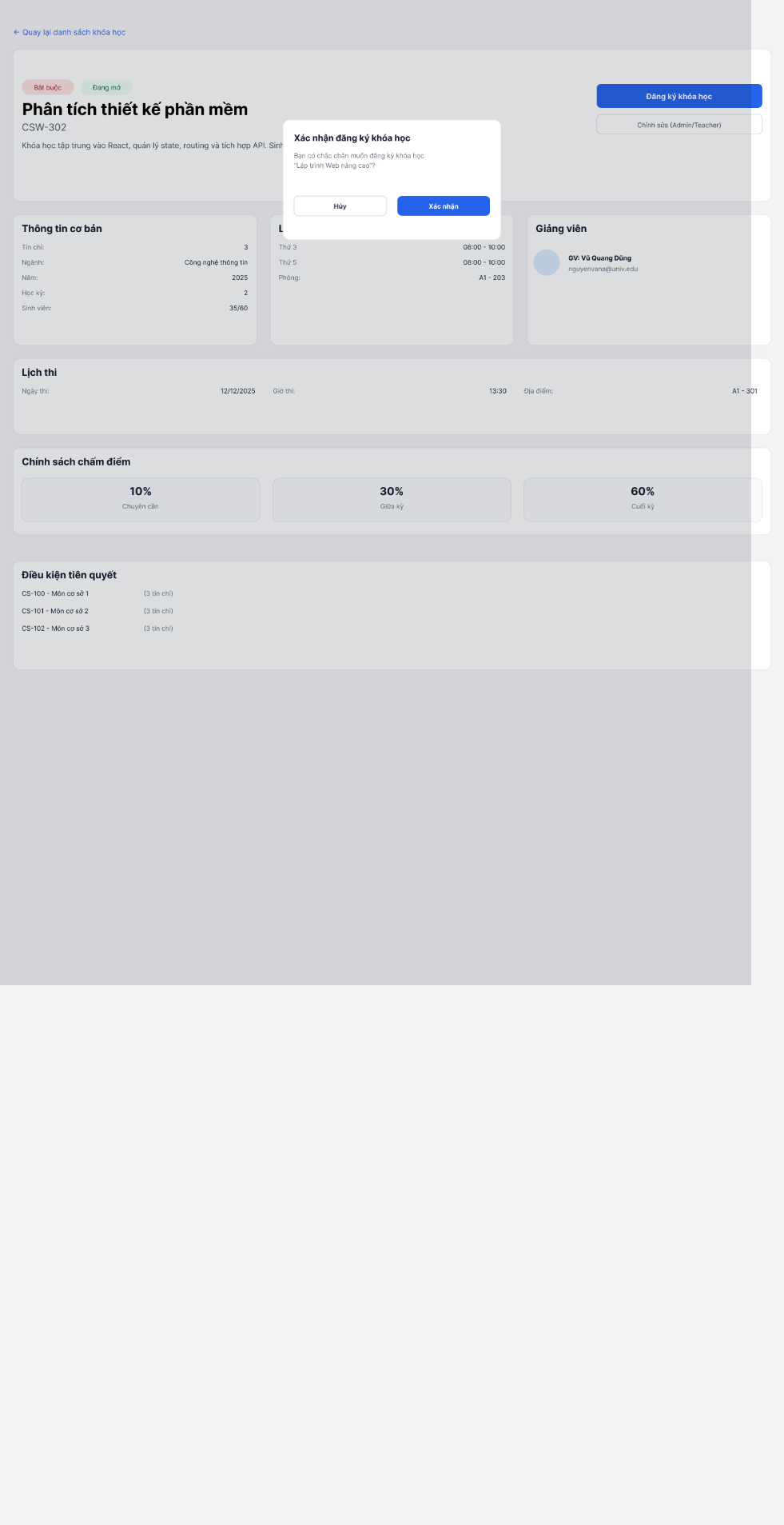
*Hình 14: Giao diện dashboard sinh viên*

**Trang danh sách khóa học**

****

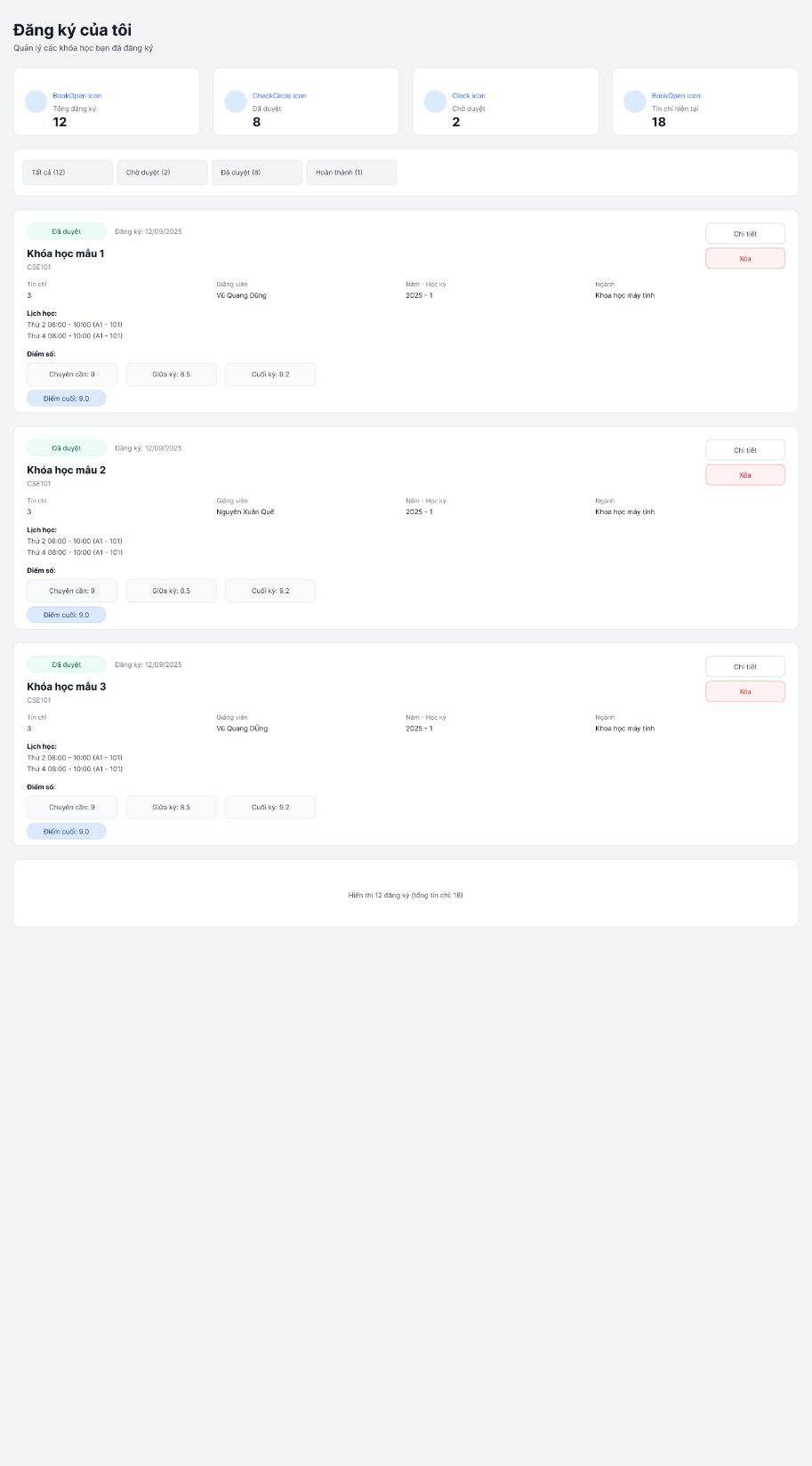
*Hình 15: Giao diện danh sách khóa học*

**Trang đăng kí khóa học**

****

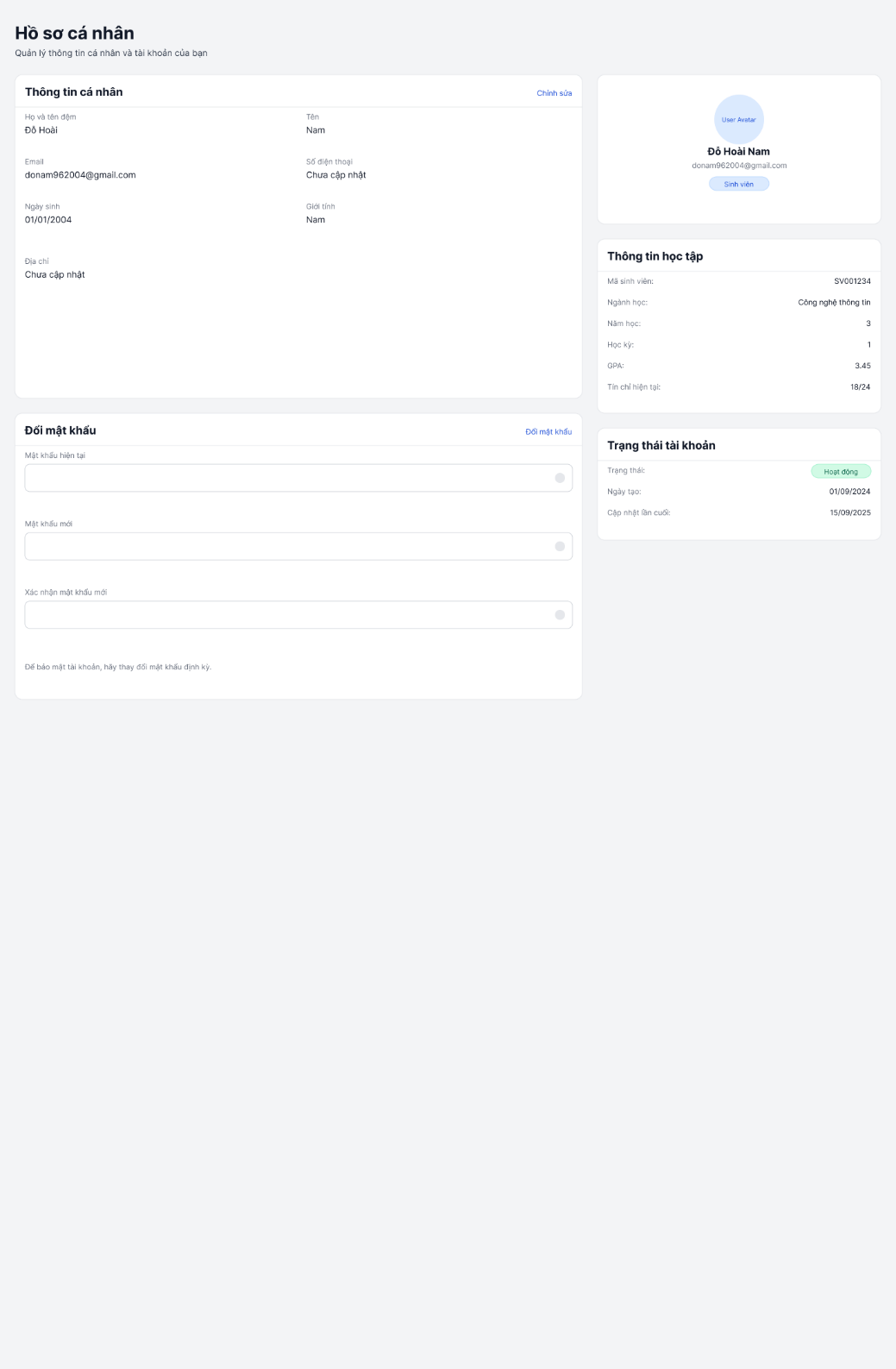
*Hình 16: Giao diện đăng kí khóa học*

**Trang xem các khóa học đã đăng kí (Sinh viên)**

****

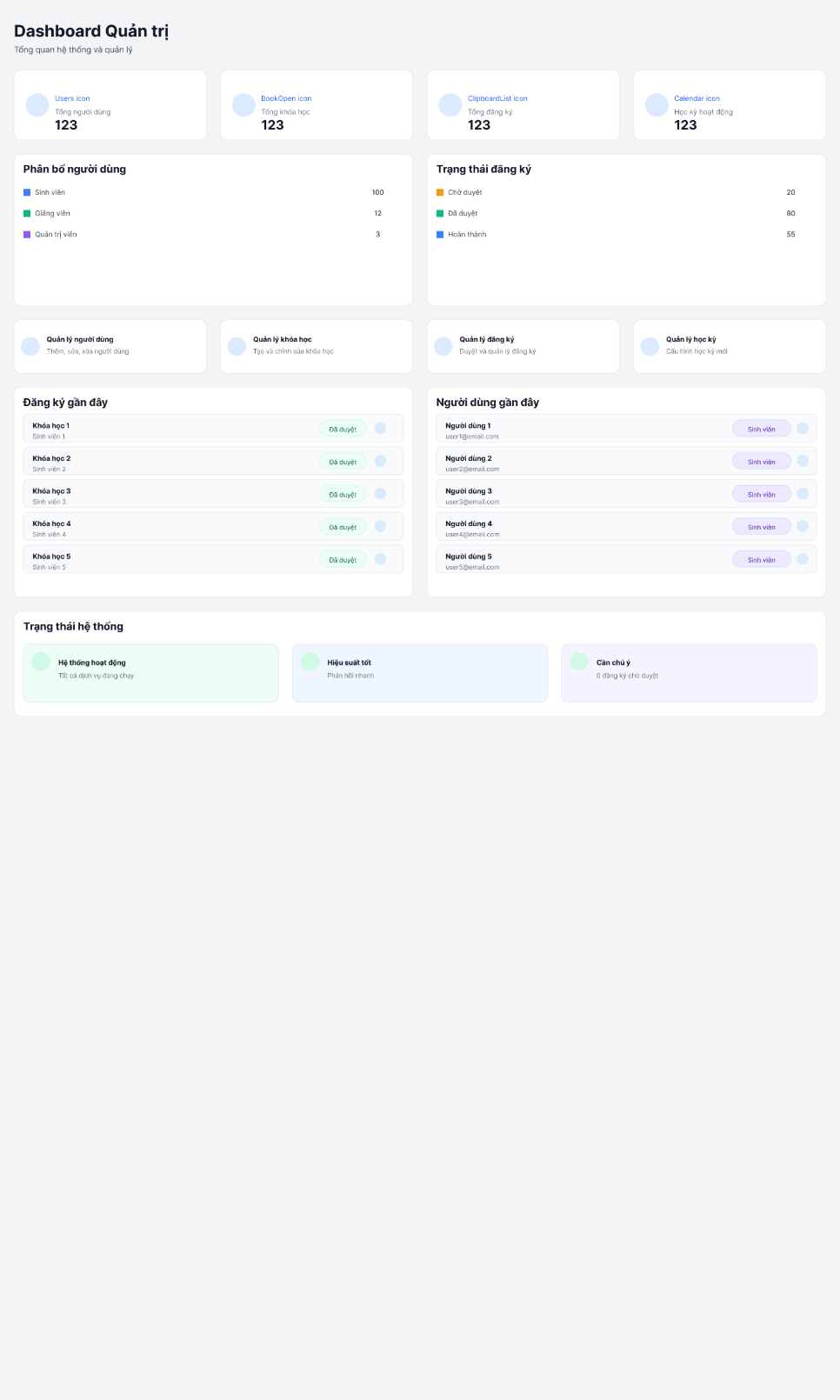
*Hình 17: Giao diện các khóa học đăng kí*

**Trang hồ sơ cá nhân**

****

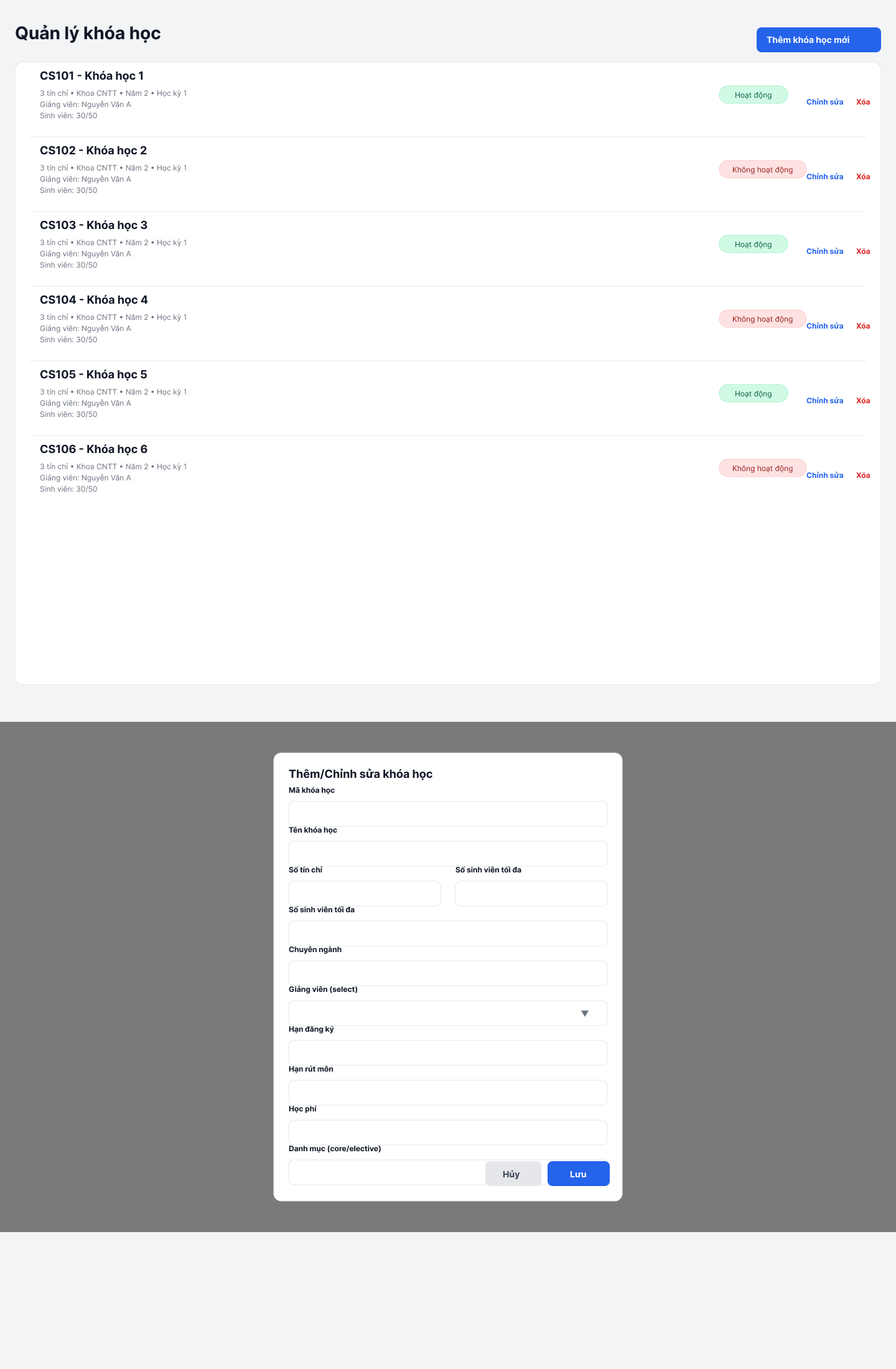
*Hình 18: Giao diện hồ sơ cá nhân*

**Trang dashboard quản trị**

****

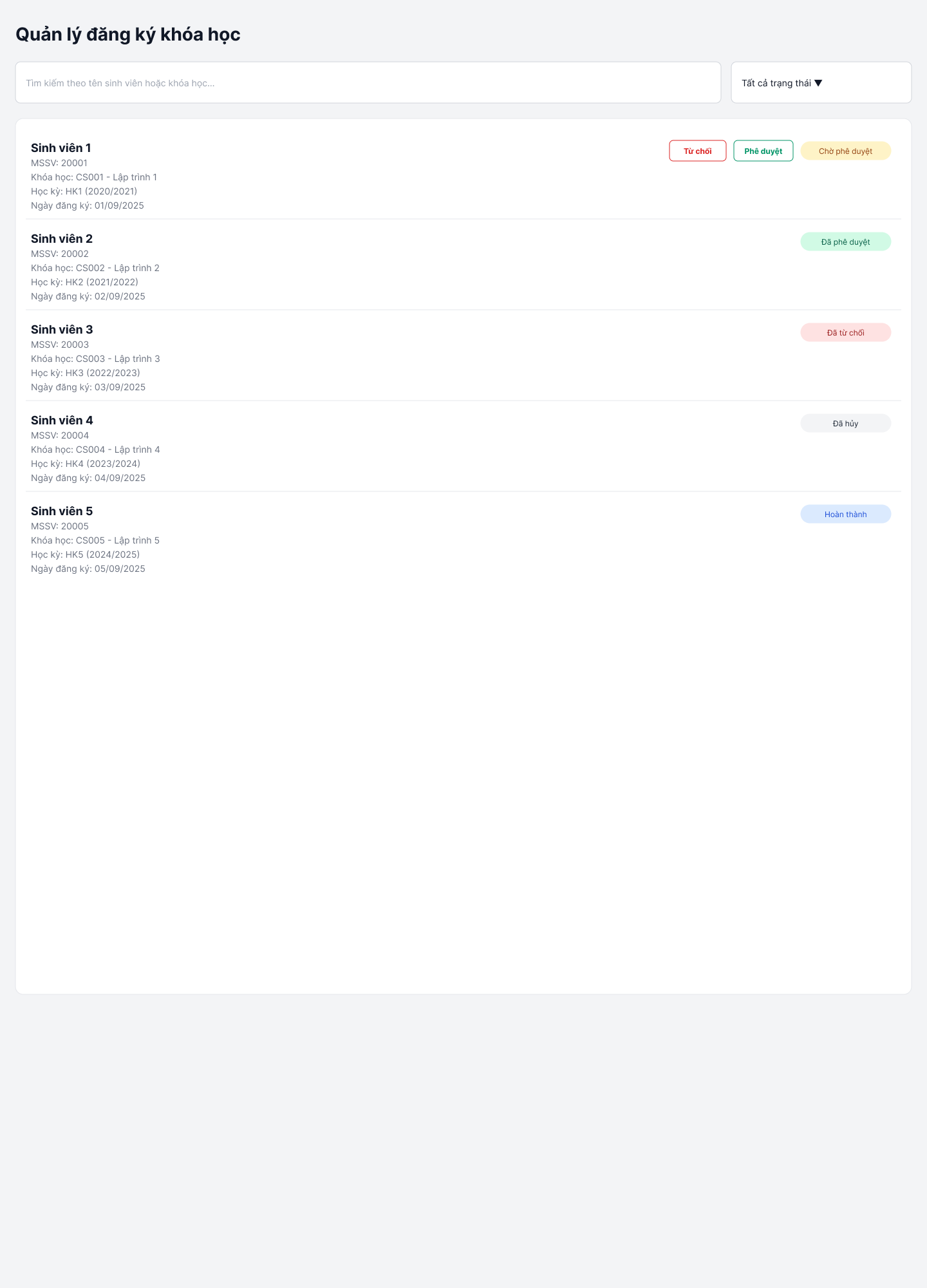
*Hình 19: Giao diện dashboard quản trị*

**Trang chỉnh sửa khóa học(Quản trị viên)**

****

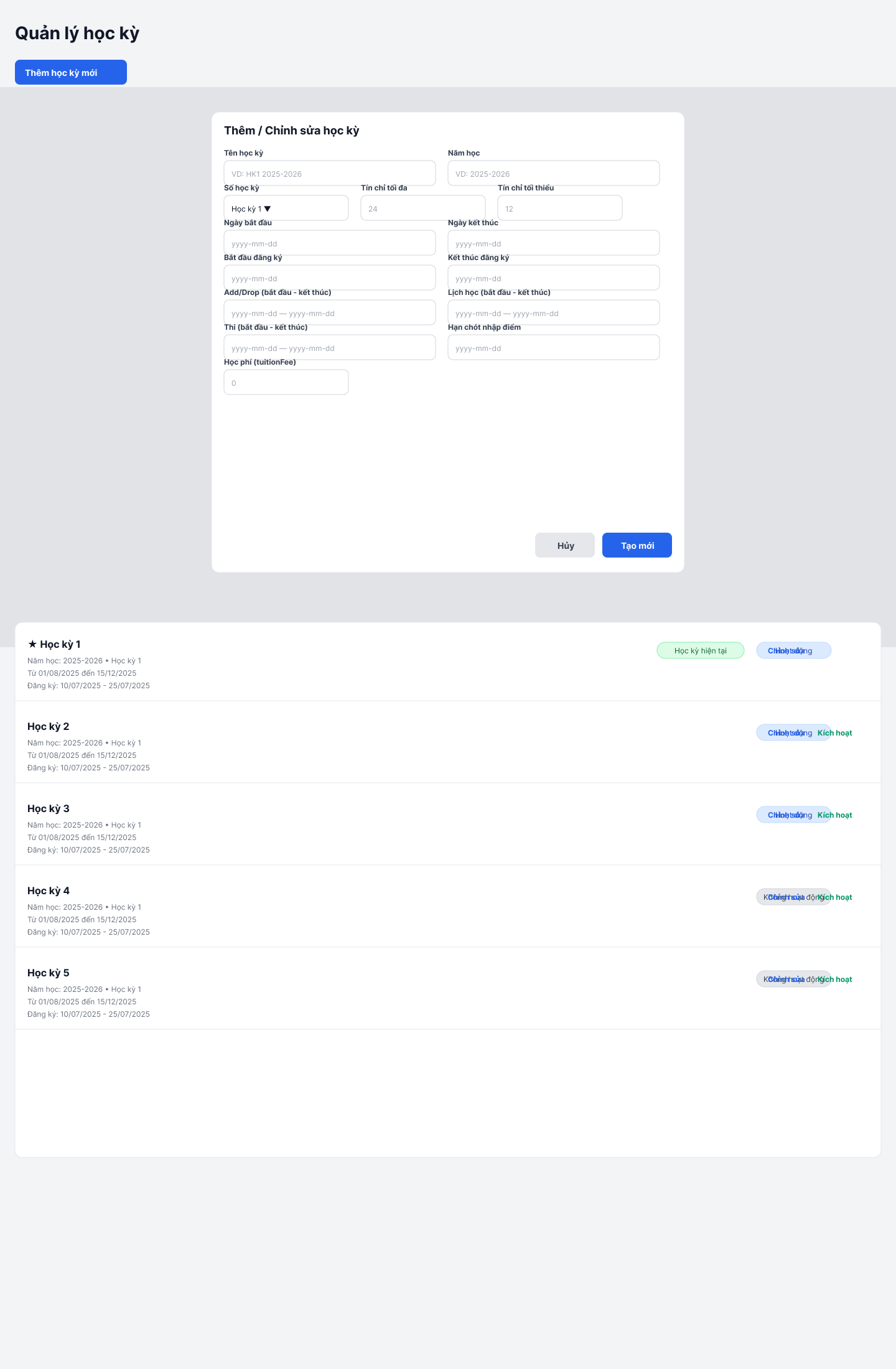
*Hình 20: Giao diện chỉnh sửa khóa học*

**Trang quản lý đăng kí khóa học**

****

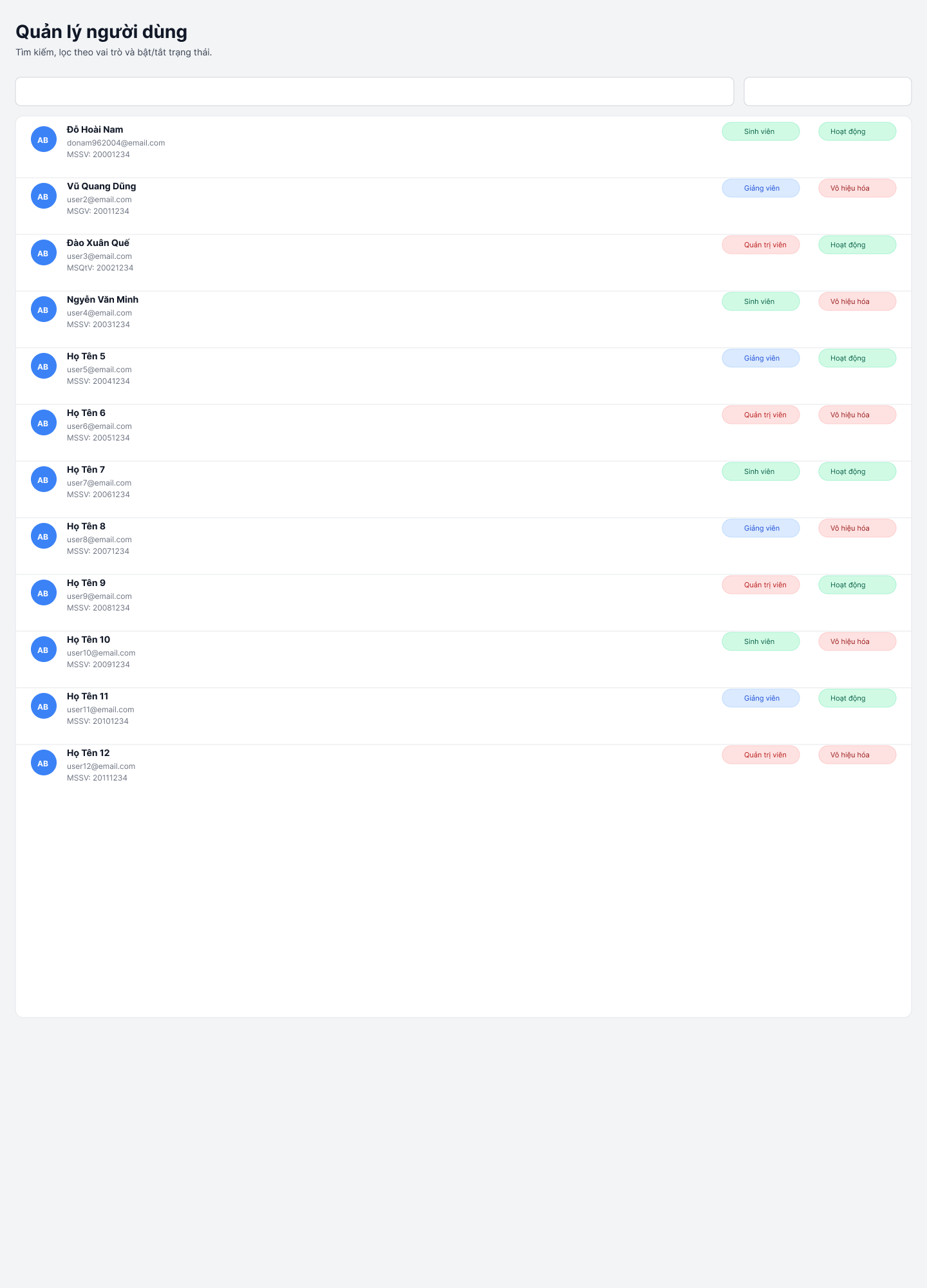
*Hình 21: Giao diện quản lí đăng kí khóa học*

**Trang quản lí học kì**

****

Hình 22: Giao diện quản lí học kì

Trang quản lí người dùng



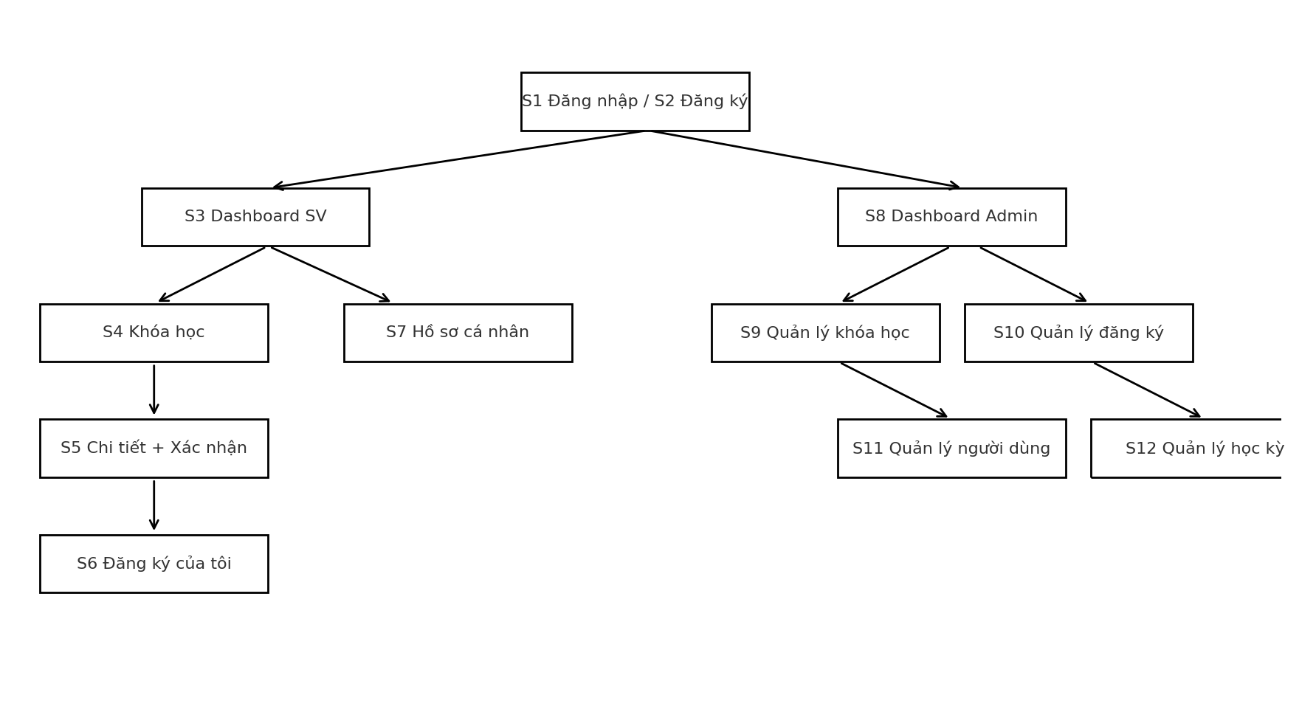
*Hình 23: Giao diện quản lí người dùng*

**4.3. User Flow & Sitemap**

**4.3.1. Gán mã màn hình (tham chiếu)**

* S1 Đăng nhập
* S2 Đăng ký tài khoản
* S3 Dashboard Sinh viên
* S4 Danh sách Khóa học (lọc/tìm/empty state)
* S5 Chi tiết khóa học + Modal Xác nhận đăng ký
* S6 Đăng ký của tôi (theo dõi/kết quả học phần)
* S7 Hồ sơ cá nhân (đổi mật khẩu)
* S8 Dashboard Quản trị
* S9 Quản lý khóa học (list + form Thêm/Chỉnh sửa)
* S10 Quản lý đăng ký khóa học (duyệt/từ chối)
* S11 Quản lý người dùng
* S12 Quản lý học kì

**4.3.2. Sitemap (cấu trúc điều hướng)**

****

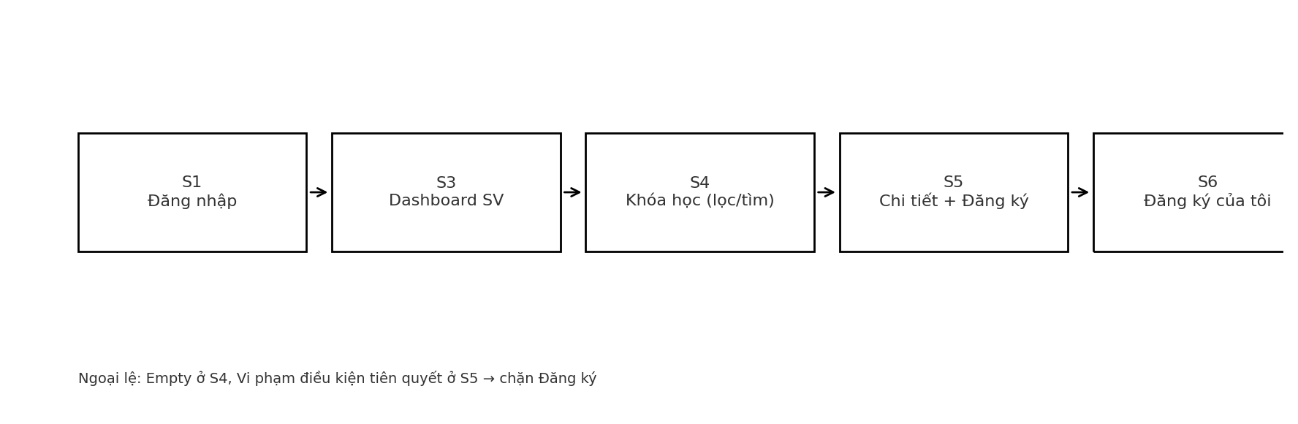
*Hình 24: Site map hệ thống*

Sơ đồ thể hiện hai nhánh chính theo **vai trò**:

* **Sinh viên:** S1/S2 → **S3 Dashboard SV** → S4 (Khóa học) → S5 (Chi tiết + Đăng ký) → **S6 (Đăng ký của tôi)**, kèm **S7 (Hồ sơ)**.
* **Quản trị:** S1 → **S8 Dashboard Admin** → S9 (Quản lý khóa học) • S10 (Quản lý đăng ký) • S11 (Quản lý người dùng) • **S12 (Quản lý học kỳ)**.

**4.3.3. Luồng thao tác chính(User Flow)**

**a) Flow A — Sinh viên đăng ký học phần**

****

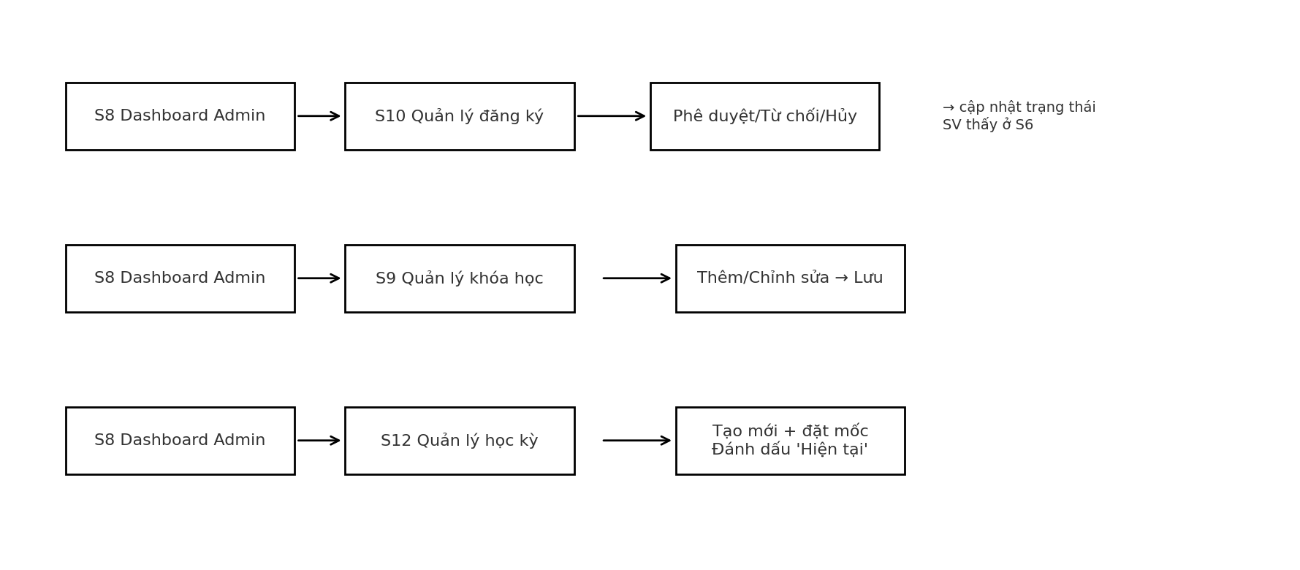
*Hình 25: Flow đăng kí học phần*

1. **S1** Đăng nhập → **S3** Dashboard SV
2. **S4** Tìm/Lọc khóa học phù hợp
3. **S5** Xem chi tiết → **Đăng ký** → Xác nhận
4. **S6** Kiểm tra kết quả (**Chờ duyệt → Đã duyệt/Hoàn thành**)

**Ngoại lệ:**

* *Empty state* ở S4 → gợi ý thay bộ lọc.
* *Không đạt điều kiện tiên quyết* ở S5 → chặn nút Đăng ký + thông báo.

**b) Flow B – Quản trị**

****

*Hình 26: Flow dành cho quản trị*

**Chương 5. Tổng kết và hướng phát triển**

**5.1. Đánh giá**

**5.1.1. Những điểm đã làm được**

1. **Mô hình miền rõ ràng, đủ thực thể cốt lõi**
   * Hệ thống đã tách bạch các thực thể chính: người dùng (phân vai *student/teacher/admin* với thông tin học tập của sinh viên), học phần (mã, tên, số tín chỉ, mô tả), học kỳ (năm học, học kỳ 1/2/3, mốc thời gian), và đăng ký (liên kết sinh viên–môn–học kỳ, trạng thái vòng đời).
   * Dữ liệu học phần phong phú: ràng buộc **tiên quyết** (cả tham chiếu và theo mã), lịch học (thứ, giờ bắt đầu/kết thúc, phòng), **sĩ số tối đa/hiện tại**, mốc **đăng ký** và **rút học**.
2. **Luồng nghiệp vụ đăng ký có kiểm soát**
   * Đăng ký theo **học kỳ** với trạng thái *pending/approved/rejected/withdrawn/completed*.
   * Có **duyệt** (approve/reject) bởi quản trị/giảng viên, ghi nhận người duyệt/lý do từ chối; hỗ trợ **rút môn** (withdraw).
   * Kiểm tra **tiên quyết** dựa trên các môn đã *completed* với mức điểm tối thiểu; chặn **trùng đăng ký** cùng sinh viên–môn–kỳ bằng ràng buộc duy nhất.
   * Trường thông tin **học phí** theo môn và ghi nhận **thanh toán** ở mức đăng ký (paymentAmount/paymentStatus).
   * Có cơ chế **chờ** (waitlist, vị trí chờ, độ ưu tiên) để giảm thất bại khi lớp đầy.
3. **Bảo mật và phân quyền cơ bản hợp lý**
   * Sử dụng xác thực **JWT**, băm mật khẩu, kiểm tra quyền truy cập theo vai trò (admin/teacher/student) ở các API quan trọng.
4. **Tối ưu hóa truy vấn và khả năng mở rộng dữ liệu**
   * Áp dụng nhiều **index** tổng hợp cho đăng ký (theo sinh viên, môn, học kỳ, trạng thái, thời điểm), cho người dùng (email, mã SV, role, ngành–năm–kỳ), và cho học kỳ (isActive/isCurrent, mốc thời gian) → cải thiện tốc độ lọc, phân trang.
5. **Kiểm tra dữ liệu đầu vào & tính nhất quán**
   * Ràng buộc kiểu dữ liệu, kiểm tra hợp lệ giờ học (HH:MM), khoảng ngày bắt đầu/kết thúc, giới hạn tín chỉ môn, sĩ số… giúp giảm lỗi do nhập liệu.
6. **Trường thông tin phục vụ quản lý học tập**
   * Lưu **điểm** (thang chữ + điểm quy đổi 0–4), **điểm chuyên cần** (số buổi/đã dự, tỷ lệ), **phản hồi/ghi chú** quản trị.
   * Điều này hỗ trợ đánh giá kết quả và theo dõi tiến độ học tập về sau.

**5.1.2. Hạn chế & rút kinh nghiệm**

1. **Xử lý cạnh tranh (race condition) khi giữ chỗ/sĩ số**
   * Chưa thấy cơ chế *atomic/transaction* rõ ràng cho tình huống nhiều sinh viên đăng ký cùng lúc một lớp gần đầy (tăng currentStudents và xét isWaitlisted).
   * **Bài học:** Cần dùng khóa/transaction hoặc thao tác cập nhật có điều kiện (optimistic concurrency) để tránh **overbook**.
2. **Kiểm soát trần tín chỉ & mốc thời gian ở cấp hệ thống**
   * Đang kiểm soát theo học kỳ và điều kiện trong mã, nhưng chưa thấy bộ **policy** tập trung, chưa có đồng bộ múi giờ/UTC rõ ràng.
   * **Bài học:** Chuẩn hóa múi giờ (UTC+7), gom logic mốc **add/drop** vào lớp chính sách để dễ bảo trì và kiểm toán.
3. **Bao phủ kiểm thử tự động còn hạn chế**
   * Hiện thiên về kiểm soát dữ liệu ở model + validate ở route; chưa thấy bộ **unit/integration/E2E** tự động hóa đầy đủ.
   * **Bài học:** Thêm test với dữ liệu mẫu đa dạng (tiên quyết–xung đột lịch–xếp lớp đầy–rút môn sau hạn).
4. **Khía cạnh an toàn, chống lạm dụng**
   * Chưa bật **rate limiting**, **helmet**, kiểm soát kích thước payload/upload, **CAPTCHA** ở các đầu mối nhạy cảm.
   * **Bài học:** Gia cố ngay những lớp bảo vệ tiêu chuẩn để vận hành production an toàn.
5. **Quan sát hệ thống & nhật ký kiểm toán**
   * Thiếu **audit log** tập trung (ai làm gì, lúc nào) và **giám sát** (metrics/alerts) để điều tra sự cố.
   * **Bài học:** Bổ sung logging có cấu trúc, tracing request-id; dashboard cảnh báo cho lỗi 5xx/tắc nghẽn.
6. **Khả năng tích hợp & tài liệu hoá**
   * Chưa có mô tả API chuẩn hóa (OpenAPI/Swagger), thiếu quy trình đồng bộ với hệ thống nhân sự/đào tạo khác (SIS/LMS).
   * **Bài học:** Xuất bản tài liệu API, tiêu chuẩn dữ liệu (mã khoa, mã môn, chuẩn điểm), lộ trình tích hợp.
7. **Trải nghiệm người dùng & khả năng tiếp cận**
   * UI hợp lý nhưng còn dư địa nâng cấp: minh hoạ TKB dạng lưới, cảnh báo xung đột theo thời gian thực, trạng thái học phí/điểm/điểm danh nhất quán.
   * **Bài học:** Tăng tính trực quan, hỗ trợ **i18n**, **a11y** (bàn phím, screen reader).

**5.2. Hướng phát triển**

**5.2.1. Ngắn hạn (ưu tiên cao)**

1. **Chốt chỗ an toàn & xử lý waitlist tự động**
   * Dùng **transaction** (hoặc cập nhật có điều kiện) khi đăng ký/hủy để không vượt sĩ số.
   * Tự động **đẩy danh sách chờ** lên *enrolled* khi có chỗ (kèm thông báo).
2. **Củng cố bảo mật & độ tin cậy vận hành**
   * Bật **helmet**, **rate limiting**, giới hạn kích thước request, kiểm soát CORS theo domain.
   * Thêm **2FA** cho quản trị, nhật ký **audit** cho hành động quản trị/duyệt, cơ chế **lockout** khi đoán mật khẩu.
3. **Chuẩn hóa mốc thời gian và trần tín chỉ**
   * Gom logic hạn đăng ký/rút môn, trần tín chỉ theo học kỳ vào *policy service*; chuẩn UTC+7; đồng bộ hiển thị ở UI.
   * Cảnh báo sớm khi gần hết hạn; khóa thao tác sau hạn ở cả API và UI.
4. **Tài liệu hoá & kiểm thử tự động**
   * Công bố **OpenAPI/Swagger**; bổ sung **unit/integration** (đăng ký/vượt trần/tiên quyết/xung đột lịch) và **E2E** kịch bản quan trọng.
   * Thiết lập **CI** chạy test và kiểm tra chất lượng mã.
5. **Giám sát & cảnh báo**
   * Logging có cấu trúc (request-id), theo dõi lỗi (Sentry), **metrics** (thời gian phản hồi, tỉ lệ lỗi), cảnh báo qua email/chat.

**5.2.2. Trung hạn**

1. **Thanh toán & công nợ học phí**
   * Chuẩn hoá *billing* theo tín chỉ/học phần, tích hợp cổng thanh toán, hoá đơn điện tử, đối soát; báo cáo công nợ.
2. **Trải nghiệm đăng ký nâng cao**
   * Tìm môn thông minh (lọc theo lịch rảnh), **gợi ý lộ trình** dựa vào tiên quyết/điểm đạt.
   * TKB dạng **calendar grid**; cảnh báo xung đột **real-time** khi chọn.
3. **Phân tích & báo cáo**
   * Dashboard cho quản trị: tỉ lệ đầy lớp, tỉ lệ rớt/đạt tiên quyết, tải giảng viên, nhu cầu môn theo kỳ; xuất **CSV/PDF**.
4. **Quy trình học vụ mở rộng**
   * Phúc khảo/đổi nhóm/đổi lớp có kiểm duyệt nhiều bước; **phỏng vấn/duyệt** đặc cách khi thiếu tiên quyết.
5. **Quản lý giảng viên & lịch phòng**
   * Quản lý định mức, xếp lịch theo **phòng/thiết bị**; cảnh báo trùng lịch dạy.

**5.2.3. Dài hạn**

1. **Tích hợp sinh thái giáo dục**
   * Kết nối **SIS/LMS** (điểm danh tự động, đồng bộ kết quả), **SSO** (OAuth2/SAML), đồng bộ hồ sơ sinh viên.
2. **Quản trị đa cơ sở & đa học kỳ**
   * Hỗ trợ **multi-tenant** theo cơ sở/khoa; phân quyền chi tiết hơn; cấu hình chính sách theo từng đơn vị.
3. **Di động & thông báo đa kênh**
   * Ứng dụng **mobile**; thông báo đẩy khi duyệt đăng ký, mở lớp, đổi lịch, hạn nộp.
4. **Nâng cao chất lượng & hiệu năng**
   * Cache truy vấn phổ biến; tách dịch vụ (microservices) khi quy mô lớn; kiểm thử tải; **sharding/replica** cho CSDL.
5. **Quản trị dữ liệu & tuân thủ**
   * Vòng đời dữ liệu, ẩn/ẩn danh hóa khi cần; đáp ứng tiêu chuẩn **bảo vệ dữ liệu**; cơ chế sao lưu/khôi phục định kỳ.

**5.3. Kết luận**

Hệ thống đã hiện thực tương đối đầy đủ các chức năng cốt lõi của quản lý tín chỉ: mô hình miền rõ ràng, đăng ký có kiểm soát, ràng buộc học vụ (tiên quyết/hạn đăng ký/sĩ số), và phân quyền theo vai trò. Để tiến lên mức sẵn sàng **production** quy mô lớn, trọng tâm nên là: (i) **an toàn dữ liệu và giao dịch** khi cạnh tranh chỗ, (ii) **bảo mật vận hành** và **giám sát**, (iii) **kiểm thử tự động** bao phủ luồng quan trọng, và (iv) **trải nghiệm người dùng** trực quan với lịch–cảnh báo theo thời gian thực. Trên nền tảng đó, các hướng phát triển về thanh toán, phân tích, tích hợp hệ sinh thái giáo dục, và mở rộng đa cơ sở sẽ giúp hệ thống đáp ứng tốt hơn nhu cầu thực tế.

**Chương 6. Tài liệu tham khảo**

**6.1. Tài liệu kỹ thuật chính**

1. Node.js Foundation, “Node.js Documentation,” Online: <https://nodejs.org/en/docs>
2. Express.js, “Express Guide & API,” Online: <https://expressjs.com>
3. React Team, “React Docs,” Online: <https://react.dev>
4. MongoDB Inc., “MongoDB Manual,” Online: <https://www.mongodb.com/docs>
5. Mongoose, “Mongoose ODM Docs,” Online: <https://mongoosejs.com/docs>
6. Jones, M. et al., “JSON Web Token (JWT),” *RFC 7519*, IETF, 2015. Online: <https://www.rfc-editor.org/rfc/rfc7519>
7. bcrypt, “bcrypt Library Documentation,” Online: <https://github.com/kelektiv/node.bcrypt.js>
8. Axios, “Axios HTTP Client,” Online: <https://axios-http.com>
9. Postman, “API Platform Docs,” Online: <https://learning.postman.com>
10. Jest, “Jest Testing Framework,” Online: <https://jestjs.io/docs>
11. Cypress, “Cypress E2E Testing,” Online: <https://docs.cypress.io>
12. OWASP, “Cheat Sheet Series (Authentication, Authorization, Input Validation…),” Online: <https://cheatsheetseries.owasp.org>

**6.2. Chuẩn & thực hành khuyến nghị**  
13. OWASP, “ASVS – Application Security Verification Standard,” Online: https://owasp.org/www-project-application-security-verification-standard  
14. Mozilla, “HTTP Security Guidelines (CORS, Cookies, Headers),” Online: https://developer.mozilla.org