TRƯỜNG ĐẠI HỌC PHENIKAA KHOA CÔNG NGHỆ THÔNG TIN

**PHÂN TÍCH THIẾT KẾ**

**PHẦN MỀM**

**(NO5)**

**Nhóm 13**

***Dư án: Hệ thống Quản lý Đăng ký Học theo Tín chỉ***

|  |  |
| --- | --- |
| 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

**Tuần 1: Phân tích Yêu cầu Hệ thống Quản lý Đăng ký Học theo Tín chỉ**

**1. Giới thiệu**

Trong bối cảnh các trường đại học ngày càng áp dụng hình thức đào tạo theo tín chỉ, nhu cầu xây dựng một hệ thống quản lý đăng ký học trực tuyến trở nên cấp thiết. Hệ thống cần đảm bảo hỗ trợ sinh viên, giảng viên và quản trị viên trong việc đăng ký môn học, quản lý lớp học, cập nhật điểm số và quản trị dữ liệu.

Mục tiêu của báo cáo tuần này là xác định yêu cầu ban đầu, thông qua việc nhận diện các Actor và Use Case cơ bản của hệ thống.

**2. Actor của Hệ thống**

Dựa trên phân tích yêu cầu, các Actor chính được xác định như sau:

Sinh viên

* + Đối tượng sử dụng hệ thống nhiều nhất.
  + Nhiệm vụ: đăng ký môn học, xem thông tin học tập, và quản lý lịch cá nhân.

Giảng viên

* + Chịu trách nhiệm giảng dạy và quản lý điểm số.
  + Nhiệm vụ: mở lớp học, nhập điểm, xem danh sách sinh viên đăng ký.

Quản trị viên (Admin)

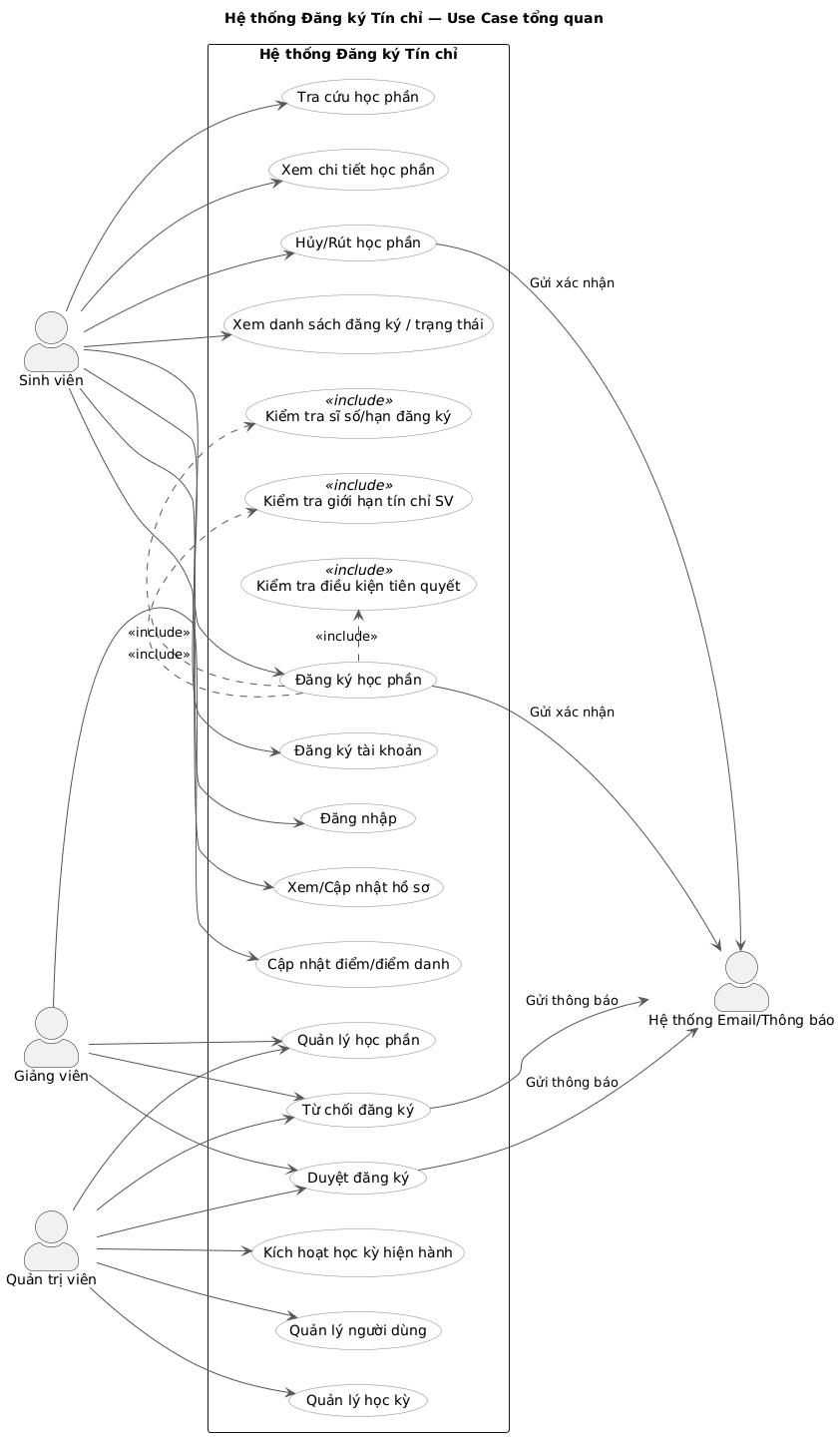
* + Người quản lý toàn bộ hệ thống.
  + Nhiệm vụ: quản lý thông tin sinh viên, môn học, lớp học; cấp quyền và duy trì hoạt động hệ thống.

Hệ thống

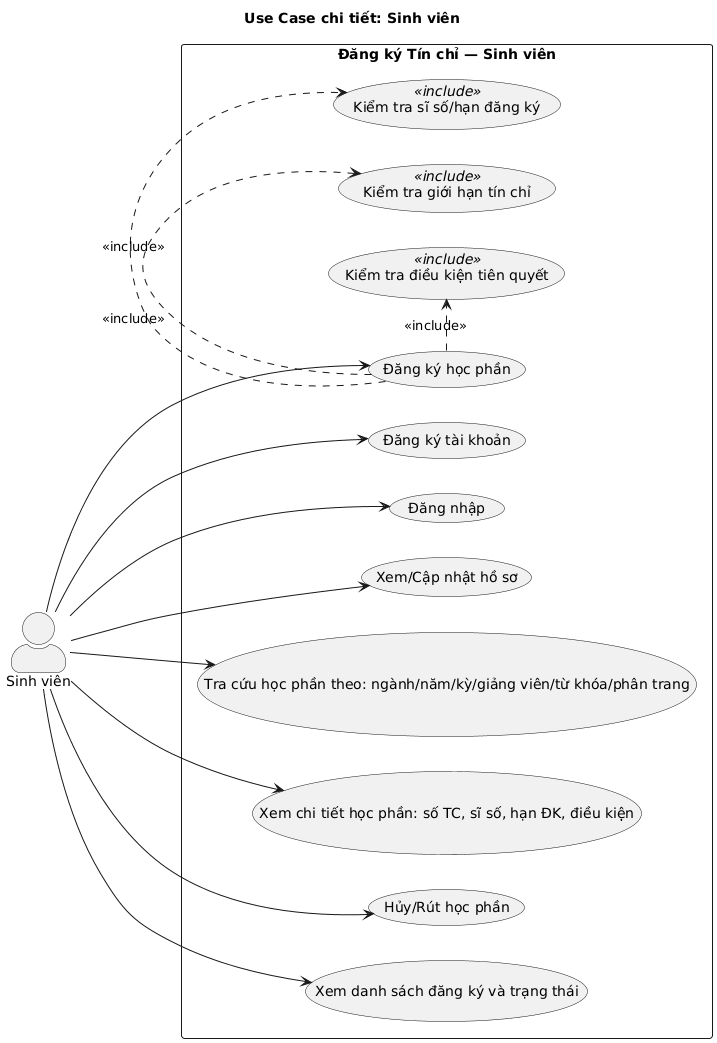
* + Là phần mềm trung gian, xử lý yêu cầu từ các Actor.
  + Nhiệm vụ: cung cấp giao diện, xác thực thông tin, lưu trữ và xử lý dữ liệu.

**3. Use Case của Hệ thống**

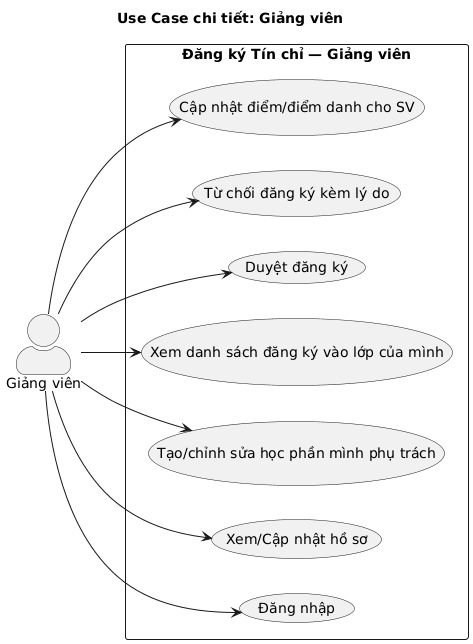
**Use case tổng quan của hệ thống**



**Use case chi tiết cho sinh viên**

****

**Use case chi tiết cho giảng viên**

****

**Đặc tả các use case**

**3.1. UC\_Register — Đăng ký tài khoản**

**Mục tiêu:** Sinh viên/Giảng viên/Quản trị tạo tài khoản để truy cập hệ thống.

**Tác nhân chính:** Sinh viên, Giảng viên, (Quản trị có thể tạo thay trong UC\_ManageUser).

**Bên liên quan:** Nhà trường, Phòng đào tạo, Hệ thống email/OTP (nếu có), Bảo mật.

**Tiền điều kiện:**

* Hệ thống đang khả dụng; chưa đăng nhập.
* Email/MSSV chưa tồn tại trong hệ thống.

**Hậu điều kiện:**

* Tài khoản ở trạng thái *active* hoặc *pending-verify* (tùy thiết lập).

**Dòng chính:**

1. Tác nhân mở màn hình Đăng ký.
2. Nhập họ tên, email/MSSV, mật khẩu (và vai trò nếu cho phép tự chọn).
3. Hệ thống kiểm tra hợp lệ (định dạng, độ mạnh mật khẩu, trùng lặp).
4. (Tùy chọn) Gửi email xác thực/OTP.
5. Tạo bản ghi User và trả thông báo thành công.

**Luồng thay thế/ngoại lệ:**

* 3a. Email/MSSV đã tồn tại → hiển thị lỗi, gợi ý quên mật khẩu.
* 4a. Xác thực thất bại/hết hạn → yêu cầu gửi lại mã.

**Quy tắc:** Mật khẩu tối thiểu n ký tự, có ký tự đặc biệt/số; Email thuộc domain nhà trường (tùy chọn).

**Dữ liệu:** User{id, name, email, role, status, password\_hash, created\_at}.

**3.2 UC\_Login — Đăng nhập**

**Mục tiêu:** Cấp phiên làm việc.

**Tác nhân:** Tất cả người dùng.

**Tiền điều kiện:** Tài khoản *active*; đã đăng ký.

**Hậu điều kiện:** Token/JWT hợp lệ; ghi nhận lần đăng nhập gần nhất.

**Dòng chính:**

1. Nhập email/MSSV + mật khẩu.
2. Hệ thống xác thực (so khớp hash), kiểm tra trạng thái.
3. Cấp JWT, trả thông tin hồ sơ tối thiểu (id, role, name).

**Ngoại lệ:** Sai mật khẩu; tài khoản bị khóa; hết lượt/lockout tạm thời.

**Quy tắc:** Chống brute-force (rate limit), ghi log.

**3.3 UC\_Profile — Xem/Cập nhật hồ sơ**

**Mục tiêu:** Người dùng xem/cập nhật thông tin cá nhân.

**Tác nhân:** SV, GV, Admin.

**Tiền điều kiện:** Đã đăng nhập.

**Hậu điều kiện:** Hồ sơ cập nhật và được audit log.

**Dòng chính:**

1. Người dùng mở trang Hồ sơ.
2. Xem thông tin; chỉnh sửa các trường cho phép (điện thoại, địa chỉ, avatar…).
3. Lưu; hệ thống validate và cập nhật.

**Ngoại lệ:** Định dạng không hợp lệ; quyền không đủ khi đổi vai trò/trạng thái.

**Dữ liệu:** UserProfile (mở rộng của User).

**3.4 UC\_SearchCourse — Tra cứu học phần**

**Mục tiêu:** Tìm học phần theo bộ lọc và phân trang.

**Tác nhân:** SV, GV, Admin (tất cả có thể tra cứu).

**Tiền điều kiện:** Đã đăng nhập; có Học kỳ *active* (nếu lọc theo học kỳ hiện hành).

**Hậu điều kiện:** Danh sách học phần phù hợp, kèm thuộc tính hiển thị.

**Dòng chính:**

1. Tác nhân nhập từ khóa/bộ lọc: ngành, năm, kỳ, giảng viên, mã HP.
2. Hệ thống truy vấn và trả danh sách (page, pageSize, total).
3. Tác nhân có thể mở chi tiết học phần (UC\_CourseDetail).

**Ngoại lệ:** Không tìm thấy; bộ lọc mâu thuẫn.

**Dữ liệu:** Course{id, code, name, credits, capacity, enrolled, deadline, prerequisites, lecturer\_id, semester\_id}.

**3.5 UC\_CourseDetail — Xem chi tiết học phần**

**Mục tiêu:** Xem thông tin đầy đủ một học phần.

**Tác nhân:** SV, GV, Admin.

**Tiền điều kiện:** Học phần tồn tại; người dùng đăng nhập.

**Hậu điều kiện:** Hiển thị nội dung, cung cấp hành động liên quan (Đăng ký/Hủy dành cho SV, Chỉnh sửa dành cho GV/Admin).

**Dòng chính:**

1. Tác nhân chọn một học phần từ danh sách.
2. Hệ thống hiển thị thông tin: mô tả, số tín chỉ, sĩ số/còn chỗ, hạn đăng ký, điều kiện tiên quyết, lịch học (nếu có), giảng viên phụ trách.

**Ngoại lệ:** Học phần bị ẩn/không thuộc kỳ *active* (tùy chính sách).

**3.6 UC\_RegisterCourse — Đăng ký học phần**

**Mục tiêu:** SV ghi danh vào học phần.

**Tác nhân:** Sinh viên.

**Bên liên quan:** Giảng viên (duyệt), Admin, Phòng đào tạo.

**Tiền điều kiện:**

* SV đăng nhập; Học kỳ *active* mở cổng đăng ký.
* Học phần cho phép đăng ký; chưa quá hạn.

**Hậu điều kiện:**

* Tạo Registration ở trạng thái *pending*/*approved* tùy mô hình (phổ biến: pending → duyệt bởi GV/Admin).
* Sĩ số cập nhật; gửi thông báo (nếu cấu hình).

**Dòng chính:**

1. SV nhấn **Đăng ký** tại học phần.
2. Hệ thống chạy các kiểm tra bao hàm: UC\_CheckPrereq, UC\_CheckCapacity, UC\_CheckCreditLimit.
3. Nếu qua hết kiểm tra, tạo bản ghi Registration và (a) set *pending* để chờ duyệt, hoặc (b) tự động *approved* nếu chính sách cho phép.
4. Trả kết quả và gửi thông báo.

**Luồng thay thế/ngoại lệ:**

* 2a. **Thiếu tiên quyết** → từ chối, nêu rõ học phần thiếu.
* 2b. **Đầy sĩ số** → vào danh sách chờ (nếu có) hoặc từ chối.
* 2c. **Vượt giới hạn tín chỉ** (theo kỳ) → từ chối.
* 2d. **Quá hạn đăng ký** → từ chối.
* 3a. **Trùng lịch** (nếu có dữ liệu lịch) → từ chối hoặc cảnh báo.

**Quy tắc:**

* credits\_registered + credits\_course ≤ credit\_limit\_semester.
* now ≤ course.deadline.
* completed(prereq\_course) == true cho tất cả prerequisites.
* enrolled < capacity.

**Dữ liệu:** Registration{id, student\_id, course\_id, status[pending|approved|rejected|dropped], reason, created\_at}.

**3.7 UC\_DropCourse — Hủy/Rút học phần**

**Mục tiêu:** SV rút khỏi học phần đã đăng ký.

**Tác nhân:** Sinh viên.

**Tiền điều kiện:** Có bản ghi Registration ở trạng thái *approved*/*pending*; còn trong **hạn rút** (nếu thiết lập).

**Hậu điều kiện:** Cập nhật trạng thái *dropped*; giảm sĩ số; ghi nhận lịch sử.

**Dòng chính:**

1. SV mở danh sách đăng ký → chọn Hủy.
2. Hệ thống kiểm tra hạn rút; cập nhật trạng thái; điều chỉnh enrolled của học phần.
3. Gửi thông báo xác nhận.

**Ngoại lệ:** Quá hạn rút; học phần đã chấm điểm → không cho hủy.

**3.8 UC\_ViewRegs — Xem danh sách đăng ký & trạng thái**

**Mục tiêu:** SV theo dõi các đăng ký trong kỳ.

**Tác nhân:** Sinh viên.

**Tiền điều kiện:** Đã đăng nhập.

**Hậu điều kiện:** Hiển thị danh sách kèm trạng thái và điểm (nếu có).

**Dòng chính:**

1. SV mở trang Đăng ký của tôi.
2. Hệ thống trả danh sách: học phần, tín chỉ, trạng thái, ghi chú, điểm số.

**3.9 UC\_ViewCourseRegs — Xem danh sách SV trong lớp (Giảng viên)**

**Mục tiêu:** GV xem các đăng ký vào lớp mình phụ trách.

**Tác nhân:** Giảng viên.

**Tiền điều kiện:** GV đăng nhập; có học phần phụ trách.

**Hậu điều kiện:** Danh sách đăng ký hiển thị (pending/approved/rejected).

**Dòng chính:**

1. GV chọn học phần của mình.
2. Hệ thống trả danh sách đăng ký và thao tác duyệt/từ chối.

**3.10 UC\_Approve — Duyệt đăng ký**

**Mục tiêu:** GV/Admin phê duyệt yêu cầu đăng ký pending.

**Tác nhân:** Giảng viên, Quản trị.

**Tiền điều kiện:** Bản ghi Registration ở *pending*; còn chỗ; chưa quá hạn.

**Hậu điều kiện:** Trạng thái *approved*; cập nhật sĩ số; gửi thông báo.

**Dòng chính:**

1. GV/Admin mở danh sách pending.
2. Chọn đăng ký → **Duyệt**.
3. Hệ thống kiểm tra lại capacity/deadline; cập nhật trạng thái, sĩ số.

**Ngoại lệ:** Lớp đầy/đã hết hạn → từ chối (xem UC\_Reject).

**3.11 UC\_Reject — Từ chối đăng ký**

**Mục tiêu:** Từ chối yêu cầu pending, có lý do.

**Tác nhân:** Giảng viên, Quản trị.

**Tiền điều kiện:** Bản ghi pending tồn tại.

**Hậu điều kiện:** Trạng thái *rejected*; lưu reason; gửi thông báo.

**Dòng chính:**

1. Chọn đăng ký pending.
2. Nhập lý do từ chối.
3. Cập nhật trạng thái.

**3.12 UC\_Grade — Cập nhật điểm/điểm danh**

**Mục tiêu:** GV nhập điểm/điểm danh cho SV trong lớp.

**Tác nhân:** Giảng viên.

**Tiền điều kiện:** Đăng nhập; là GV phụ trách học phần; có danh sách SV đã approved.

**Hậu điều kiện:** Lưu điểm/điểm danh; tính trạng thái đạt/không đạt (tùy thang điểm/quy chế).

**Dòng chính:**

1. GV mở lớp → tab Điểm.
2. Nhập/Import điểm từng SV.
3. Hệ thống validate biên (0–10, 0–100…); lưu.

**Ngoại lệ:** Giá trị ngoài biên; khóa điểm sau thời hạn.

**Dữ liệu:** Registration.grade, attendance (nếu có).

**3.13 UC\_ManageCourse — Quản lý học phần**

**Mục tiêu:** Tạo/sửa học phần; gán giảng viên; cập nhật capacity, deadline, tiên quyết.

**Tác nhân:** GV (chỉ học phần của mình), Admin (toàn cục).

**Tiền điều kiện:** Đăng nhập; có quyền.

**Hậu điều kiện:** Học phần được tạo/cập nhật; ràng buộc dữ liệu nhất quán.

**Dòng chính:**

1. Tạo học phần mới hoặc mở để sửa.
2. Nhập/đổi: mã, tên, credits, capacity, deadline, prerequisites, lịch học, lecturer, semester.
3. Lưu thay đổi; hệ thống validate và cập nhật liên quan.

**Ngoại lệ:** Trùng mã; deadline trước ngày hiện tại; capacity < enrolled.

**3.14 UC\_ManageUser — Quản lý người dùng**

**Mục tiêu:** Admin tạo/sửa/xóa người dùng; phân vai; kích hoạt/khóa.

**Tác nhân:** Quản trị viên.

**Tiền điều kiện:** Đăng nhập với vai trò Admin.

**Hậu điều kiện:** Hồ sơ người dùng cập nhật; tác động tới quyền.

**Dòng chính:**

1. Tìm người dùng → mở chi tiết.
2. Sửa trường: vai trò, trạng thái, reset mật khẩu, thông tin cơ bản.
3. Lưu.

**Ngoại lệ:** Không thể xóa người dùng đang là giảng viên của học phần đang mở.

**3.15 UC\_ManageSemester — Quản lý học kỳ**

**Mục tiêu:** Tạo/sửa học kỳ, các mốc thời gian đăng ký/rút/chấm điểm.

**Tác nhân:** Admin.

**Tiền điều kiện:** Quyền Admin.

**Hậu điều kiện:** Học kỳ được tạo/cập nhật; dùng trong filter và ràng buộc thời gian.

**Dòng chính:**

1. Tạo học kỳ với mã, tên (VD: 2025-1), ngày bắt đầu/kết thúc.
2. Khai báo mốc registration\_open, registration\_deadline, drop\_deadline, v.v.
3. Lưu.

**Ngoại lệ:** Mốc thời gian chồng chéo; end < start.

**3.16 UC\_ActivateSemester — Kích hoạt học kỳ hiện hành**

**Mục tiêu:** Chỉ định một học kỳ là *active*.

**Tác nhân:** Admin.

**Tiền điều kiện:** Có ít nhất một học kỳ hợp lệ.

**Hậu điều kiện:** Cờ *active* set cho đúng học kỳ; các use case tra cứu/đăng ký dựa theo học kỳ này.

**Dòng chính:**

1. Chọn học kỳ → **Kích hoạt**.
2. Hệ thống bỏ cờ active ở kỳ cũ (nếu có), set active cho kỳ mới.

**Ngoại lệ:** Không cho kích hoạt kỳ đã kết thúc (tùy chính sách).

**3.17 UC\_SetCreditLimit — Thiết lập ngưỡng tín chỉ**

**Mục tiêu:** Quy định tổng tín chỉ tối đa SV được đăng ký trong một kỳ.

**Tác nhân:** Admin.

**Tiền điều kiện:** Có học kỳ; có chính sách tín chỉ.

**Hậu điều kiện:** Lưu cấu hình, dùng trong UC\_RegisterCourse.

**Dòng chính:**

1. Nhập giá trị ngưỡng mặc định theo kỳ (và/hoặc theo nhóm đối tượng).
2. Lưu cấu hình.

**Ngoại lệ:** Giá trị ngoài biên hợp lệ.

**4. Yêu cầu Hệ thống**

**4.1. Yêu cầu Chức năng**

1. Hệ thống phải cho phép sinh viên đăng ký và hủy đăng ký môn học trong thời gian quy định.
2. Hệ thống phải cung cấp cho sinh viên khả năng xem thời khóa biểu theo học kỳ.
3. Hệ thống phải cho phép sinh viên xem kết quả học tập (điểm số).
4. Hệ thống phải cho phép giảng viên mở lớp học và cập nhật điểm số.
5. Hệ thống phải cung cấp cho giảng viên danh sách sinh viên đã đăng ký lớp.
6. Hệ thống phải cho phép quản trị viên quản lý thông tin sinh viên, môn học và lớp học.
7. Hệ thống phải cung cấp chức năng tìm kiếm môn học, lớp học nhanh chóng.

**4.2. Yêu cầu Phi chức năng**

1. Hiệu năng: Thời gian phản hồi khi tra cứu thời khóa biểu không quá 2 giây.
2. Bảo mật: Hệ thống phải yêu cầu đăng nhập bằng tài khoản hợp lệ (mã sinh viên, mã giảng viên).
3. Khả dụng: Hệ thống phải sẵn sàng 24/7, ngoại trừ thời gian bảo trì.
4. Dễ sử dụng: Giao diện trực quan, hỗ trợ đa ngôn ngữ (ít nhất là tiếng Việt và tiếng Anh).
5. Khả năng mở rộng: Có thể dễ dàng tích hợp với các hệ thống khác như quản lý học phí, thư viện.

**Tuần 2: Mô hình hóa Use Case và Kịch bản**

**1. Mục tiêu tuần 2**

* Trực quan hóa các Use Case đã xác định từ tuần 1 bằng sơ đồ UML.
* Mô tả chi tiết các Use Case quan trọng dưới dạng **kịch bản (scenario)**.
* Xây dựng tài liệu có thể dùng làm cơ sở cho thiết kế chi tiết ở tuần sau.

**2. Công việc chi tiết**

**Bước 1: Xây dựng Biểu đồ Use Case UML**

1. **Xác định lại Actor**:
   * Sinh viên
   * Giảng viên
   * Quản trị viên (Admin)
   * Hệ thống (nếu cần, đóng vai trò phụ trợ)
2. **Xác định Use Case** (từ tuần 1):
   * **Sinh viên**: Đăng ký môn học, Hủy đăng ký môn học, Xem thời khóa biểu, Xem kết quả học tập.
   * **Giảng viên**: Mở lớp học, Cập nhật điểm, Xem danh sách sinh viên.
   * **Quản trị viên**: Quản lý sinh viên, Quản lý môn học, Quản lý khóa học.
3. **Bổ sung quan hệ include/extend**:
   * “Đăng ký môn học” <<include>> “Kiểm tra điều kiện tiên quyết”.
   * “Đăng ký môn học” <<extend>> “Thông báo lỗi khi lớp đã đầy”.
   * “Hủy đăng ký” <<include>> “Cập nhật danh sách lớp”.

Kết quả: Một **biểu đồ Use Case tổng quan** thể hiện đầy đủ Actor, Use Case, và quan hệ.

**Bước 2: Mô tả chi tiết kịch bản Use Case (Scenario)**

**Use Case 1: Đăng ký môn học**

* **Actor chính**: Sinh viên
* **Mục tiêu**: Đăng ký thành công một môn học trong học kỳ.
* **Tiền điều kiện**: Sinh viên đã đăng nhập và trong thời gian đăng ký.
* **Hậu điều kiện**: Môn học được thêm vào danh sách của sinh viên, danh sách lớp được cập nhật.

**Luồng chính (Primary Scenario):**

1. Sinh viên chọn chức năng "Đăng ký môn học".
2. Hệ thống hiển thị danh sách môn học đang mở.
3. Sinh viên chọn môn học muốn đăng ký.
4. Hệ thống kiểm tra điều kiện tiên quyết và số lượng tín chỉ tối đa.
5. Hệ thống kiểm tra sĩ số lớp còn chỗ trống.
6. Hệ thống lưu kết quả và cập nhật danh sách lớp.
7. Hệ thống thông báo: “Đăng ký thành công”.

**Luồng phụ (Alternative Scenarios):**

* 4a. Nếu sinh viên chưa đạt môn tiên quyết → báo: “Không đủ điều kiện đăng ký”.
* 5a. Nếu lớp đã đầy → báo: “Lớp đã đủ sĩ số”.
* 3a. Nếu sinh viên chọn trùng lịch với môn khác đã đăng ký → báo: “Bị trùng lịch học”.
* 2a. Nếu ngoài thời hạn đăng ký → báo: “Hết hạn đăng ký”.

A diagram of a diagram

AI-generated content may be incorrect.

**Use Case 2: Hủy đăng ký môn học**

* **Actor chính**: Sinh viên
* **Mục tiêu**: Hủy một môn học đã đăng ký.
* **Tiền điều kiện**: Sinh viên đã đăng nhập, môn học nằm trong danh sách đã đăng ký, và còn trong thời hạn hủy.
* **Hậu điều kiện**: Môn học được xóa khỏi danh sách, danh sách lớp được cập nhật.

**Luồng chính:**

1. Sinh viên chọn chức năng "Hủy đăng ký môn học".
2. Hệ thống hiển thị danh sách các môn sinh viên đã đăng ký.
3. Sinh viên chọn môn học cần hủy.
4. Hệ thống xác nhận và xóa môn học khỏi danh sách.
5. Hệ thống cập nhật sĩ số lớp học.
6. Hệ thống thông báo: “Hủy đăng ký thành công”.

**Luồng phụ:**

* 3a. Nếu môn học đã quá hạn hủy → báo: “Không thể hủy, đã quá hạn”.
* 3b. Nếu môn học đã có điểm giữa kỳ → báo: “Không thể hủy, đã có điểm”.

A diagram of a diagram

AI-generated content may be incorrect.

**Use Case 3: Mở lớp học (Giảng viên)**

* **Actor chính**: Giảng viên
* **Mục tiêu**: Tạo một lớp học phần mới cho môn giảng dạy.
* **Tiền điều kiện**: Giảng viên đã đăng nhập và môn học thuộc phạm vi phụ trách.
* **Hậu điều kiện**: Lớp học phần mới được thêm vào hệ thống.

**Luồng chính:**

1. Giảng viên chọn “Mở lớp học”.
2. Hệ thống hiển thị danh sách môn học giảng viên có thể mở lớp.
3. Giảng viên nhập thông tin lớp (thời gian, phòng học, sĩ số).
4. Hệ thống kiểm tra phòng học và lịch giảng dạy.
5. Hệ thống lưu lớp học mới.
6. Hệ thống thông báo: “Mở lớp thành công”.

**Luồng phụ:**

* 3a. Trùng lịch giảng dạy → báo: “Lịch giảng dạy bị trùng”.
* 3b. Phòng học không khả dụng → báo: “Phòng học đã có lớp khác”.

A diagram of a diagram

AI-generated content may be incorrect.

**Bước 3: Viết tài liệu kịch bản tổng hợp**

* Danh sách Use Case + sơ đồ UML.
* Bảng mô tả chi tiết cho từng Use Case quan trọng.
* Luồng chính + luồng phụ + tiền/hậu điều kiện.

**Báo cáo tuần 3: Thiết kế Lớp và Tạo cơ sở code**

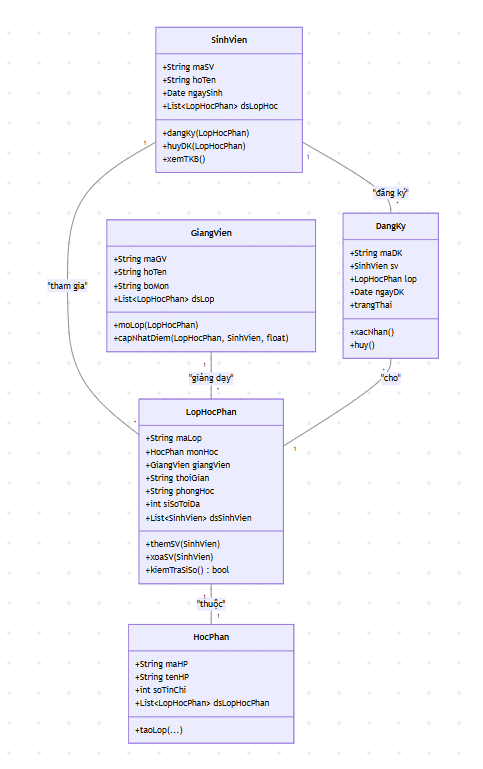
**Mục tiêu tuần 3**

* Xây dựng cấu trúc dữ liệu và các đối tượng chính của hệ thống.
* Thiết kế biểu đồ lớp để thể hiện rõ mối quan hệ giữa các thực thể.
* Tạo cơ sở mã nguồn (skeleton code) cho các lớp chính của hệ thống.

**Nội dung thực hiện**

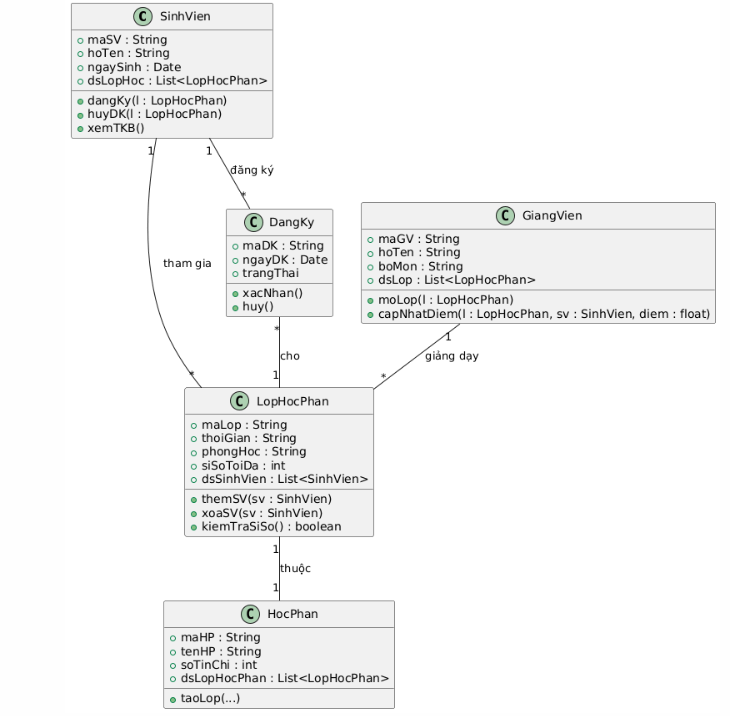
**1. Thiết kế Biểu đồ Lớp**

* Dựa trên kịch bản Use Case đã xây dựng ở tuần 2, nhóm đã phân tích và xác định được các lớp đối tượng chính gồm:
  + **SinhVien**: quản lý thông tin sinh viên.
  + **GiangVien**: quản lý thông tin giảng viên.
  + **HocPhan**: mô tả thông tin môn học.
  + **LopHocPhan**: thể hiện lớp học phần cụ thể.
  + **DangKy**: quản lý việc đăng ký học phần.
* Đã xác định được:
  + **Thuộc tính (attributes)** cơ bản cho từng lớp (ví dụ: mã sinh viên, tên sinh viên, ngày sinh,...).
  + **Phương thức (methods)** cần thiết để quản lý dữ liệu (ví dụ: thêm, xóa, cập nhật, tìm kiếm).
  + **Quan hệ giữa các lớp**:
    - SinhVien ↔ LopHocPhan (quan hệ tham gia).
    - GiangVien ↔ LopHocPhan (quan hệ giảng dạy).
    - HocPhan ↔ LopHocPhan (quan hệ thuộc).
    - SinhVien ↔ DangKy ↔ LopHocPhan (quan hệ đăng ký học).



**2. Tạo cơ sở code**

* Các lớp chính đã được tạo trong dự án với tên file tương ứng:
  + SinhVien.js
  + GiangVien.js
  + HocPhan.js
  + LopHocPhan.js
  + DangKy.js
* Trong tuần này, mỗi lớp mới chỉ được khai báo các **thuộc tính và phương thức cơ bản** ở dạng khung (chưa triển khai logic chi tiết).
* Đây sẽ là nền tảng để các tuần tiếp theo nhóm có thể mở rộng và phát triển chức năng cụ thể.



**Phân công công việc**

* **Đỗ Hoài Nam**
  + Thực hiện phân tích và xác định thuộc tính, phương thức cho các lớp **SinhVien, GiangVien, HocPhan**.
  + Thiết kế mối quan hệ giữa các lớp trên biểu đồ.
* **Nguyễn Văn Minh**
  + Thực hiện xây dựng cấu trúc lớp **LopHocPhan, DangKy**.
  + Hỗ trợ triển khai sơ bộ mã nguồn (skeleton code) và commit code lên GitHub.

**Kết quả đạt được**

* Hoàn thành **biểu đồ lớp** mô tả các đối tượng chính và mối quan hệ.
* Tạo thành công **các file lớp cơ bản** cho hệ thống.
* Có nền tảng vững chắc cho việc phát triển logic chi tiết ở các tuần tiếp theo.

**Đánh giá**

* Tiến độ tuần 3 được hoàn thành **đúng kế hoạch**.
* Cấu trúc hệ thống bước đầu đã rõ ràng, thuận lợi cho việc mở rộng.
* Nhóm đã phối hợp hiệu quả trong việc chia sẻ và phân công công việc.

**Tuần 4: Thiết kế Tương tác**

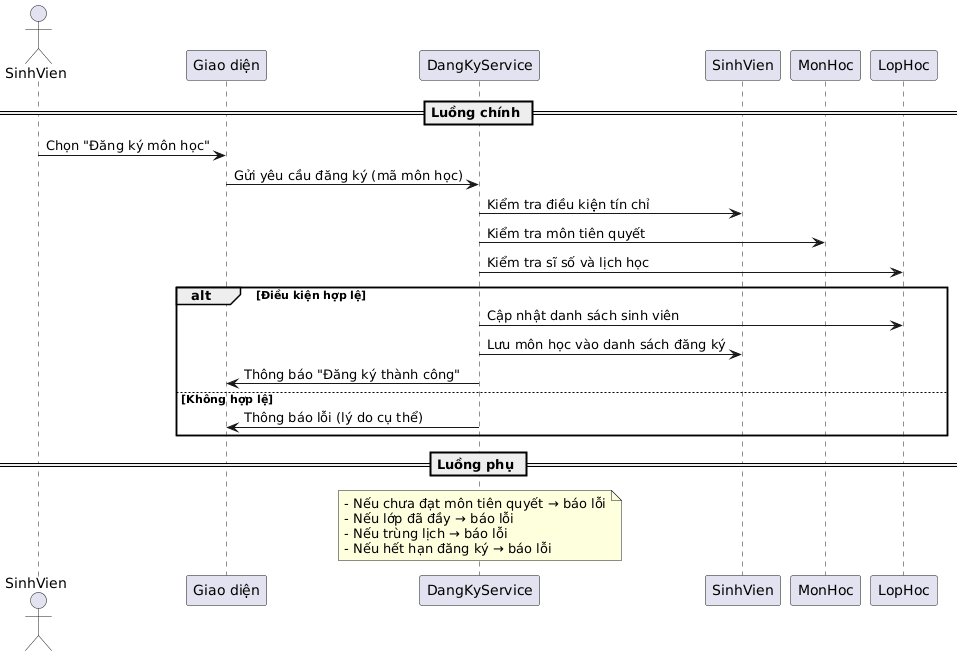
**1. Mục tiêu**

Trong tuần 4, mục tiêu chính của nhóm là:

* Thiết kế **Biểu đồ Trình tự (Sequence Diagram)** cho Use Case "Đăng ký môn học" để mô tả chi tiết các luồng tương tác giữa các đối tượng.
* Phác thảo **giao diện người dùng** cho chức năng "Đăng ký môn học".
* Đảm bảo rằng mô hình thiết kế thể hiện rõ ràng mối quan hệ giữa **Actor**, **Giao diện**, **Service** và các lớp dữ liệu liên quan.

**2. Nội dung công việc**

**2.1. Vẽ Biểu đồ Trình tự (Sequence Diagram)**

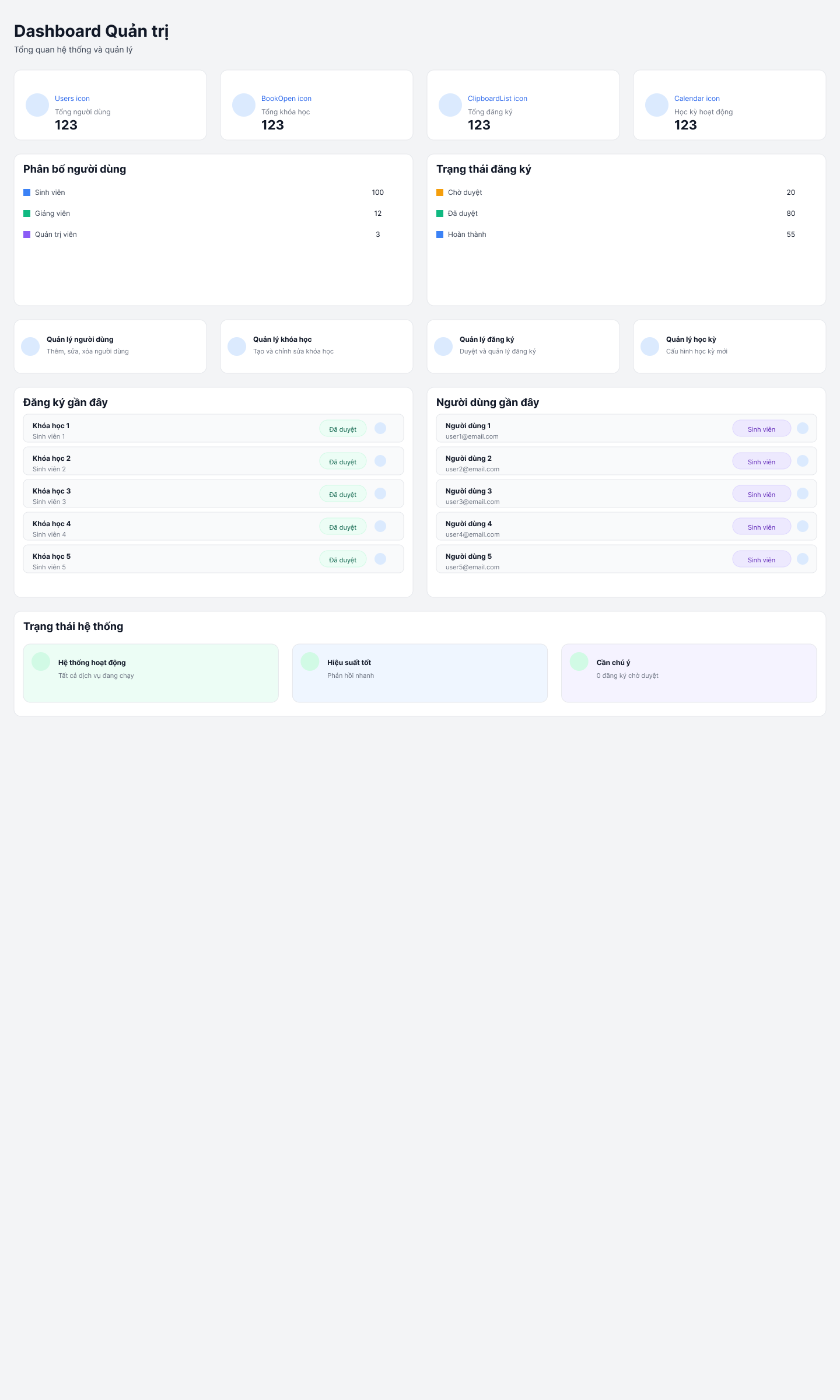
****

* **Use Case được chọn:** *Đăng ký môn học*.
* **Các đối tượng tham gia:**
  1. **Actor:** SinhVien (người dùng hệ thống).
  2. **Giao diện (UI):** Màn hình Đăng ký môn học.
  3. **Service:** DangKyService (xử lý nghiệp vụ).
  4. **Các lớp dữ liệu:**
     + SinhVien (kiểm tra thông tin sinh viên và điều kiện tín chỉ).
     + MonHoc (kiểm tra điều kiện tiên quyết).
     + LopHoc (kiểm tra sĩ số và lịch học).
* **Luồng chính trong biểu đồ trình tự:**
  1. Sinh viên chọn chức năng "Đăng ký môn học".
  2. Giao diện gửi yêu cầu đến **DangKyService**.
  3. DangKyService gọi các đối tượng liên quan (SinhVien, MonHoc, LopHoc) để kiểm tra:
     + Điều kiện tiên quyết của môn học.
     + Giới hạn số tín chỉ.
     + Sĩ số lớp còn trống.
     + Tránh trùng lịch học.
  4. Nếu hợp lệ, DangKyService cập nhật thông tin đăng ký và lưu kết quả.
  5. Hệ thống phản hồi lại cho giao diện, hiển thị thông báo “Đăng ký thành công”.
* **Luồng phụ (ngoại lệ) trong biểu đồ:**
  1. Nếu chưa đạt môn tiên quyết → thông báo lỗi.
  2. Nếu lớp đã đầy → thông báo lỗi.
  3. Nếu trùng lịch → thông báo lỗi.
  4. Nếu hết hạn đăng ký → thông báo lỗi.

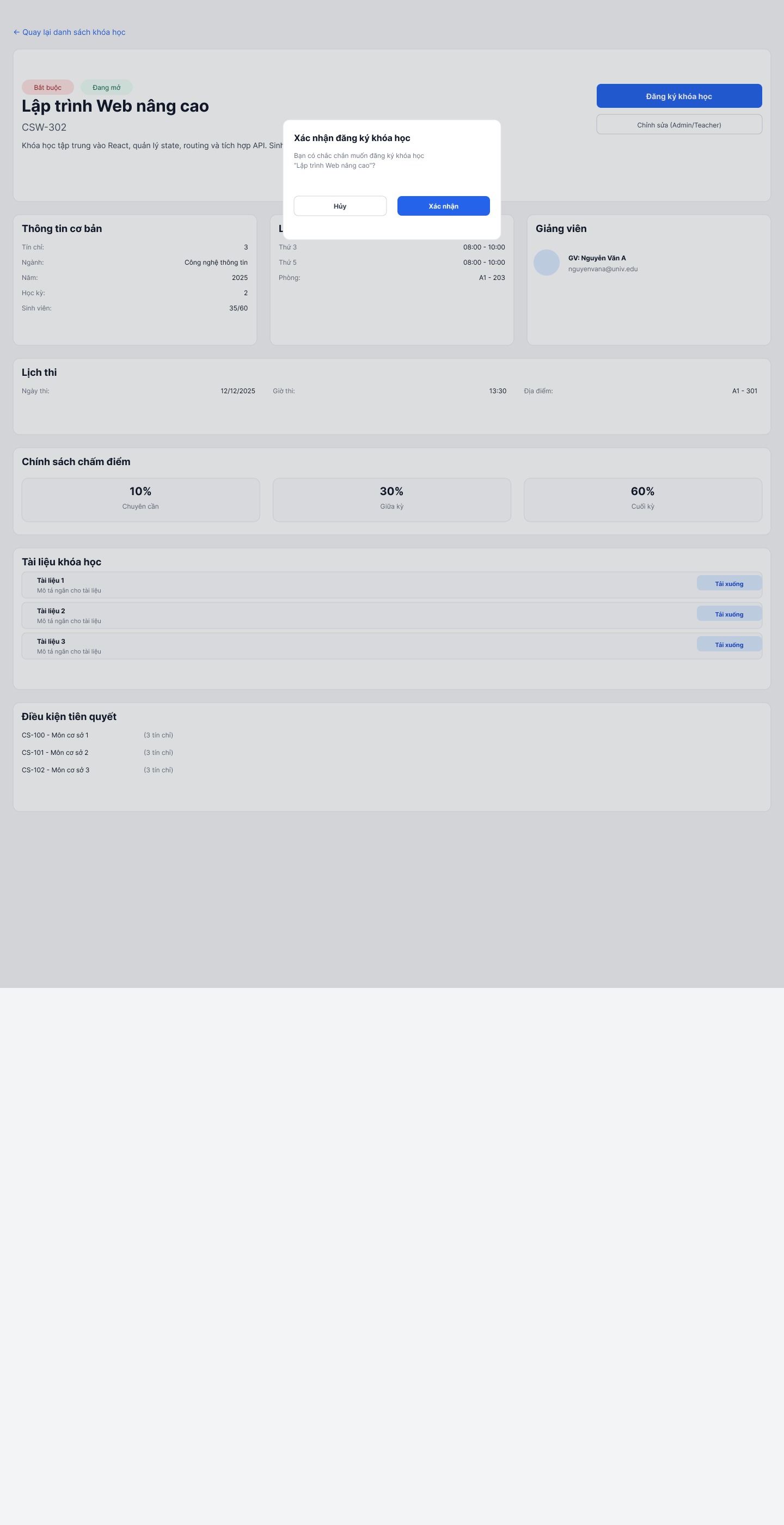
**2.2. Thiết kế Giao diện người dùng (UI)**

* **Giao diện được phác thảo:**
  + Màn hình chính có chức năng "Đăng ký môn học".
  + Khi chọn, hiển thị **danh sách môn học mở** (dạng bảng hoặc danh sách cuộn).
  + Sinh viên chọn môn học, hệ thống sẽ hiển thị:
    - Tên môn học.
    - Mã môn học.
    - Số tín chỉ.
    - Sĩ số tối đa và số chỗ còn trống.
    - Lịch học (thứ, tiết, phòng học).
  + Có nút **“Đăng ký”** → Khi nhấn, giao diện gửi yêu cầu đến hệ thống và phản hồi kết quả (thành công hoặc thất bại).
  + Khu vực thông báo hiển thị các lỗi (VD: “Trùng lịch”, “Lớp đã đầy”, …).
* **Công cụ sử dụng:**
  + Figma / Balsamiq để phác thảo giao diện.
  + Nếu chưa có điều kiện, nhóm có thể vẽ tay trên giấy rồi chụp ảnh đưa vào báo cáo.

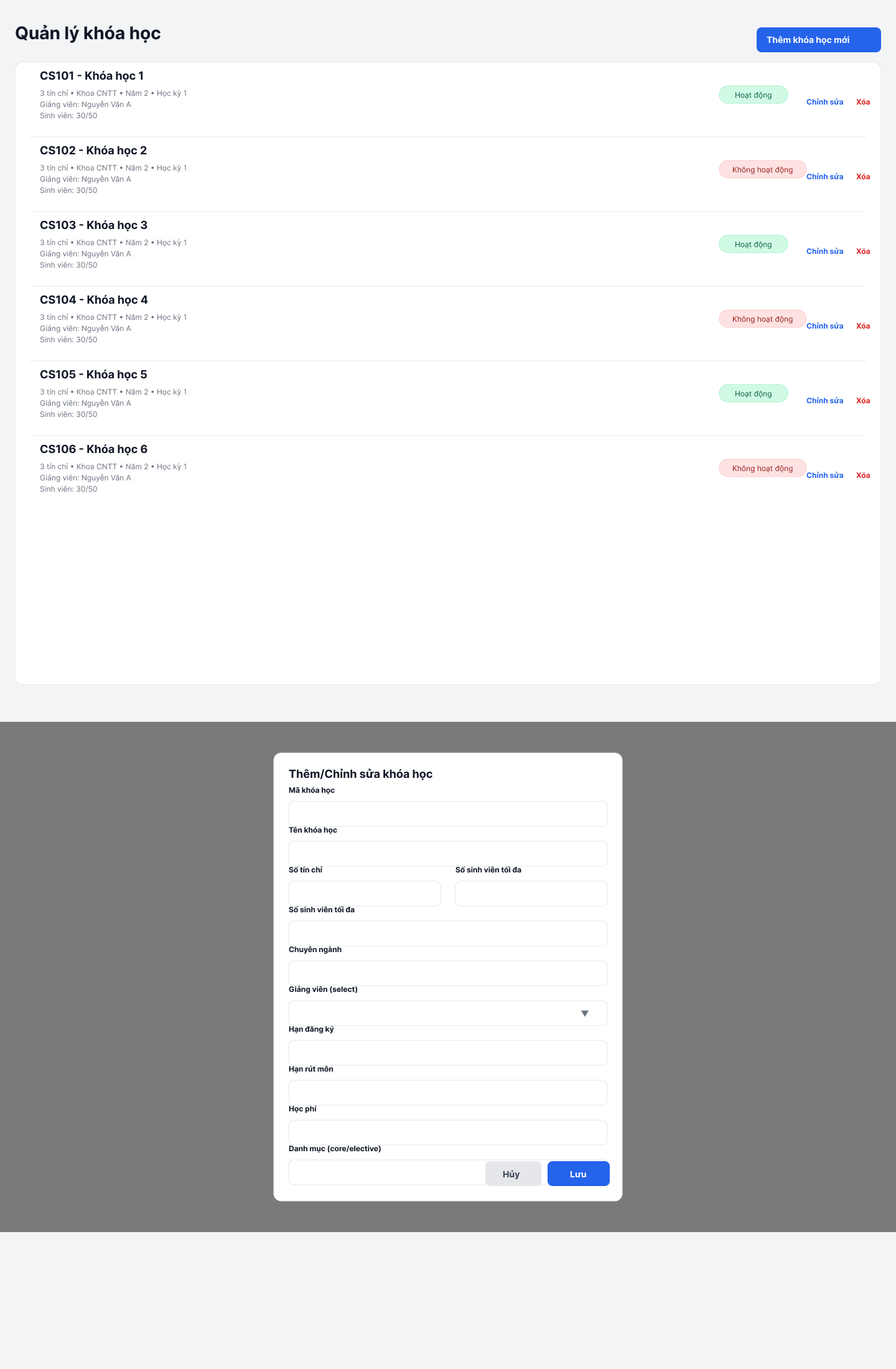
**2.2.1 Giao diện dashboard hệ thống**



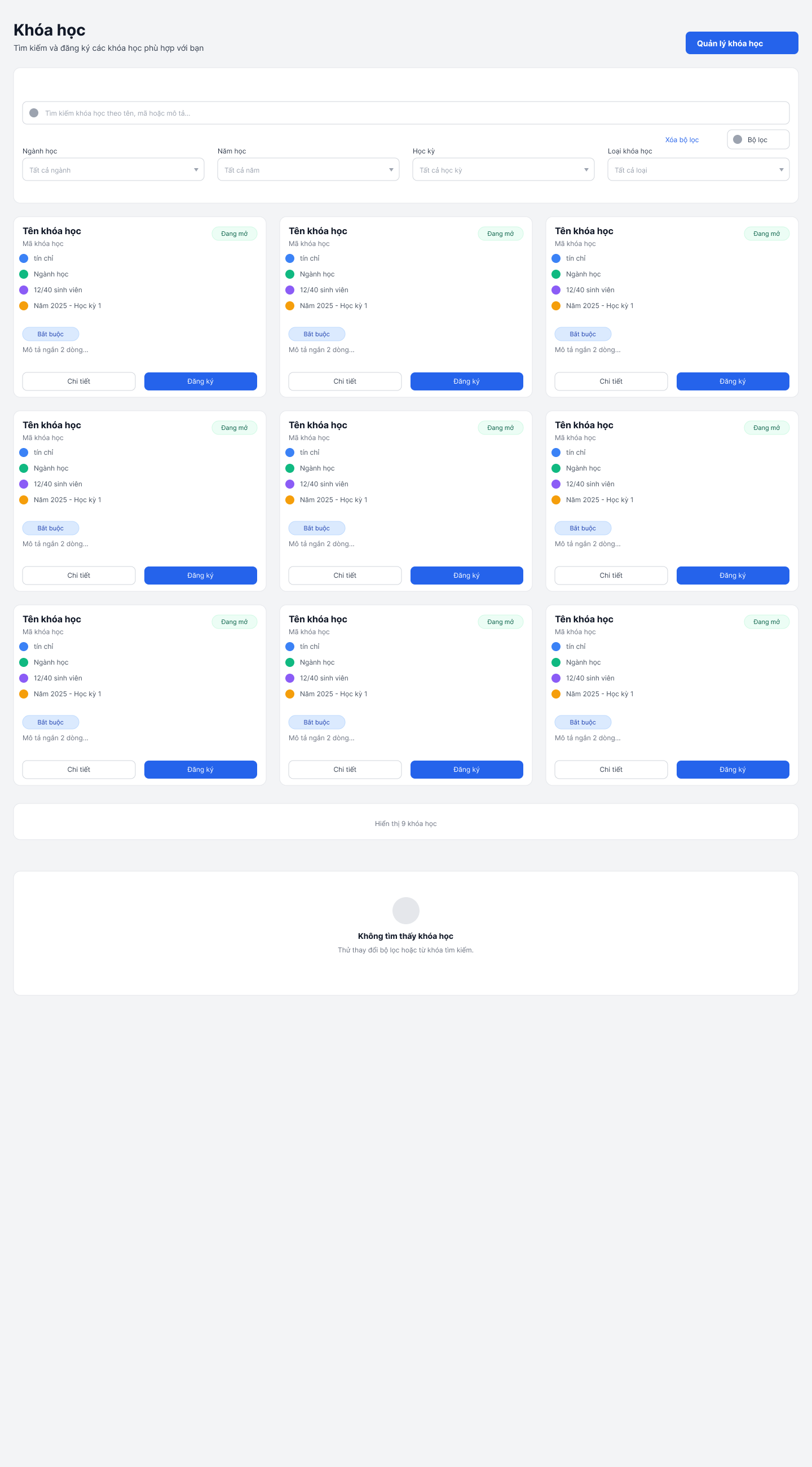
**2.2.1 Giao diện chi tiết đăng kí khóa học**

****

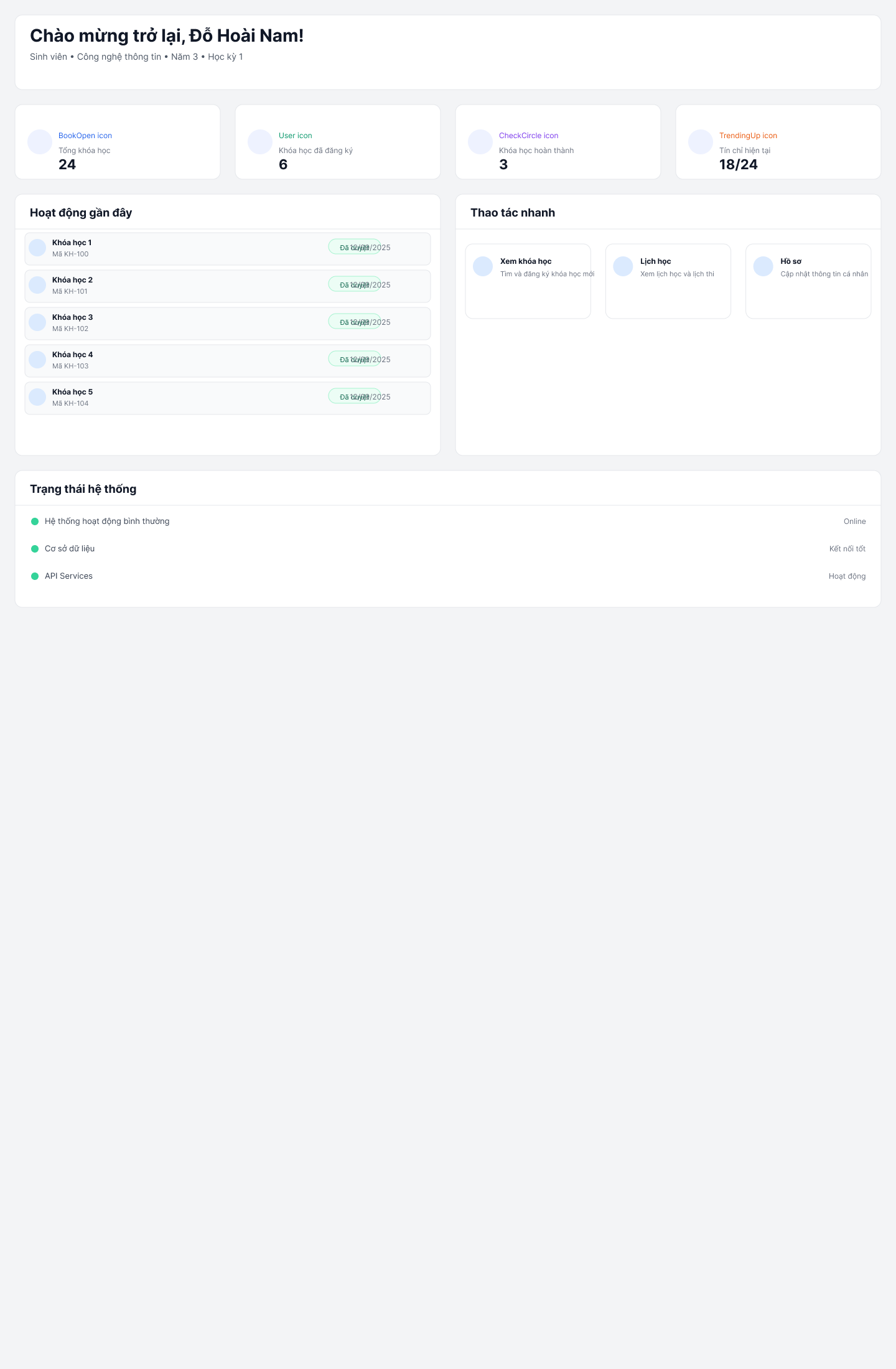
**2.2.3 Giao diện thêm chỉnh sửa khóa học**

****

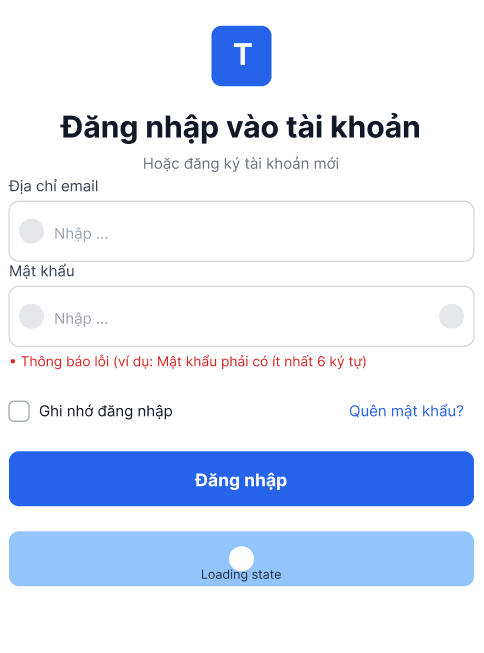
**2.2.4 Giao diện Quản lý Khóa học**

****

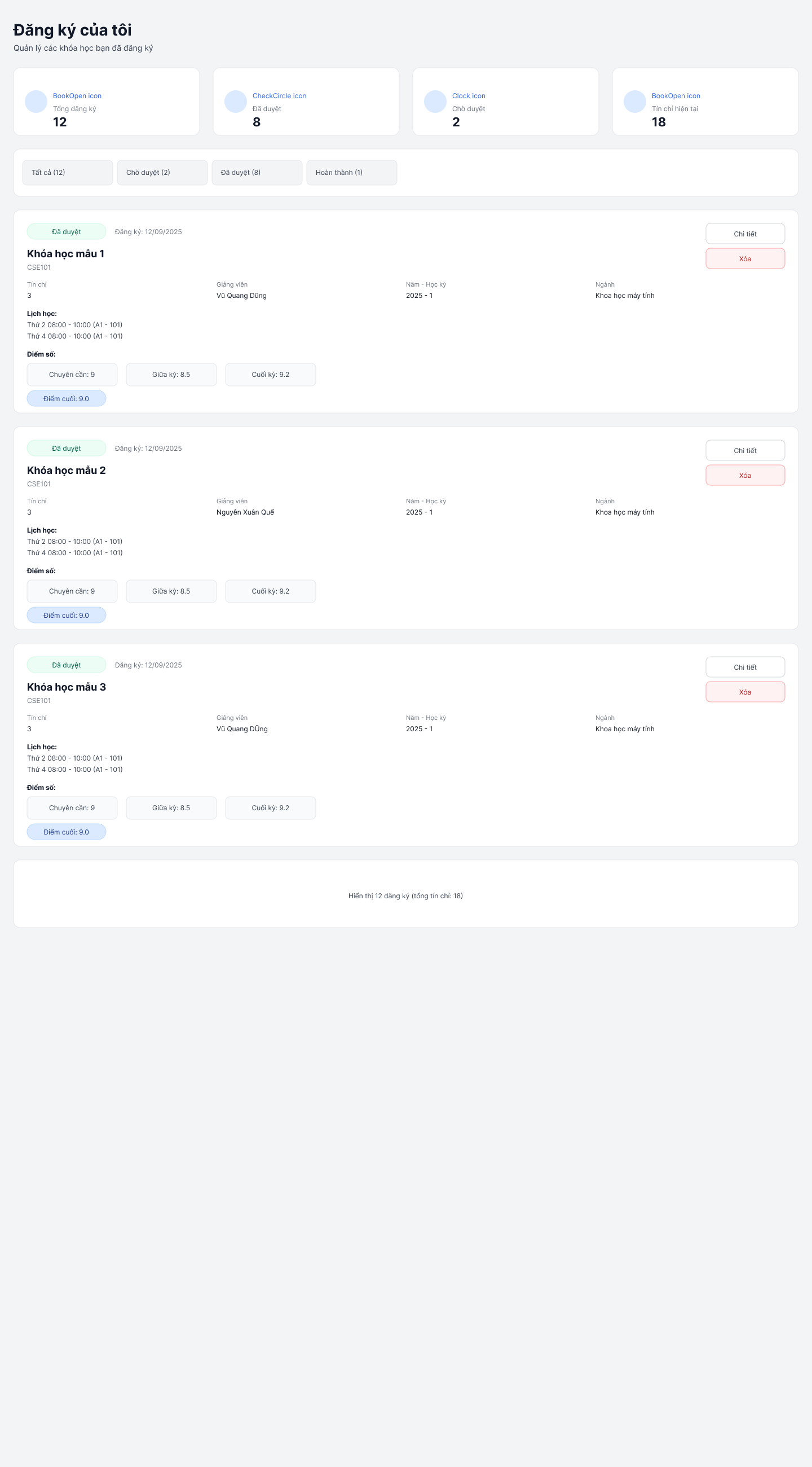
**2.2.5 Giao diện Chính**

****

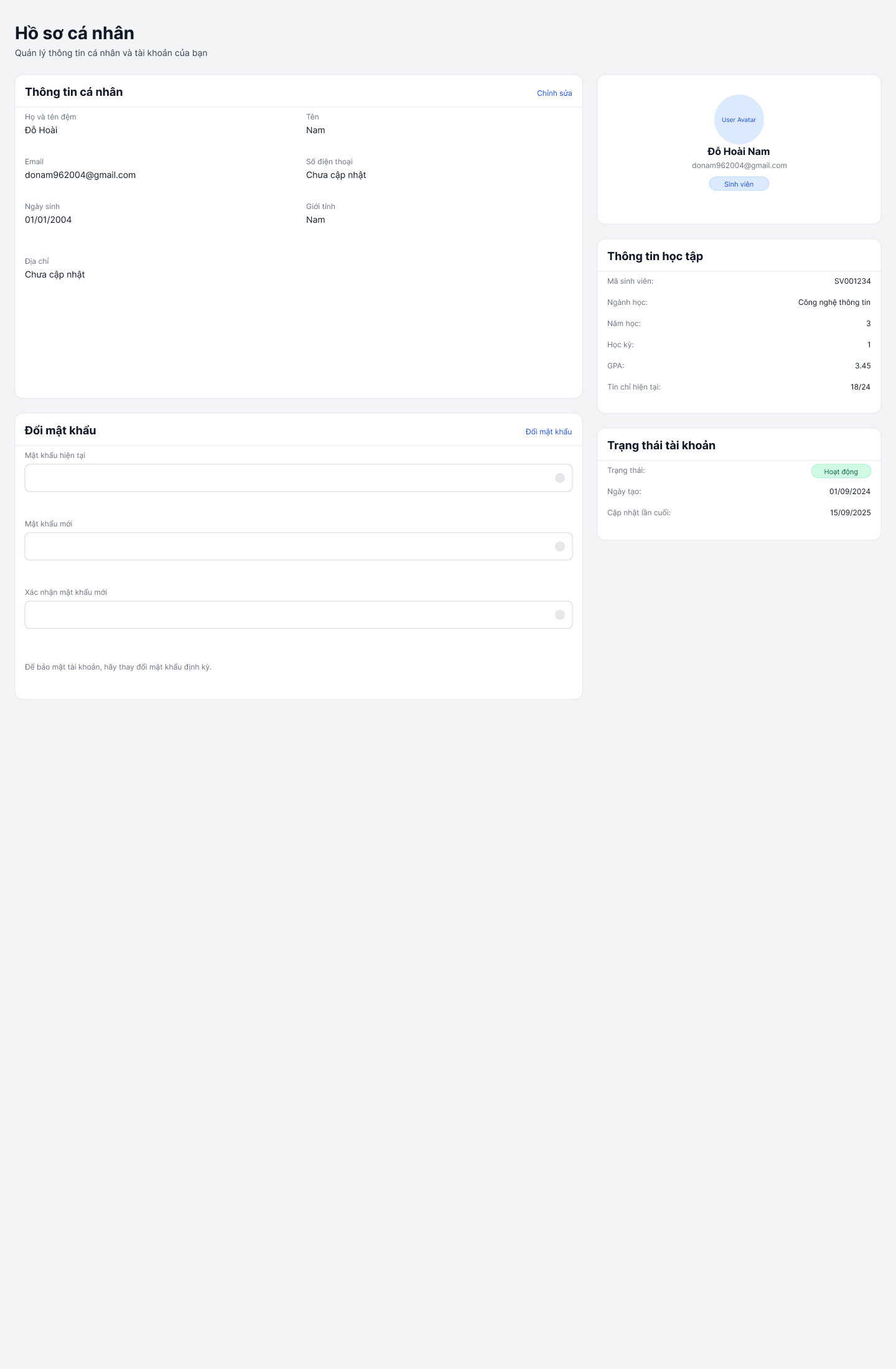
**2.2.6 Giao diện Login**

****

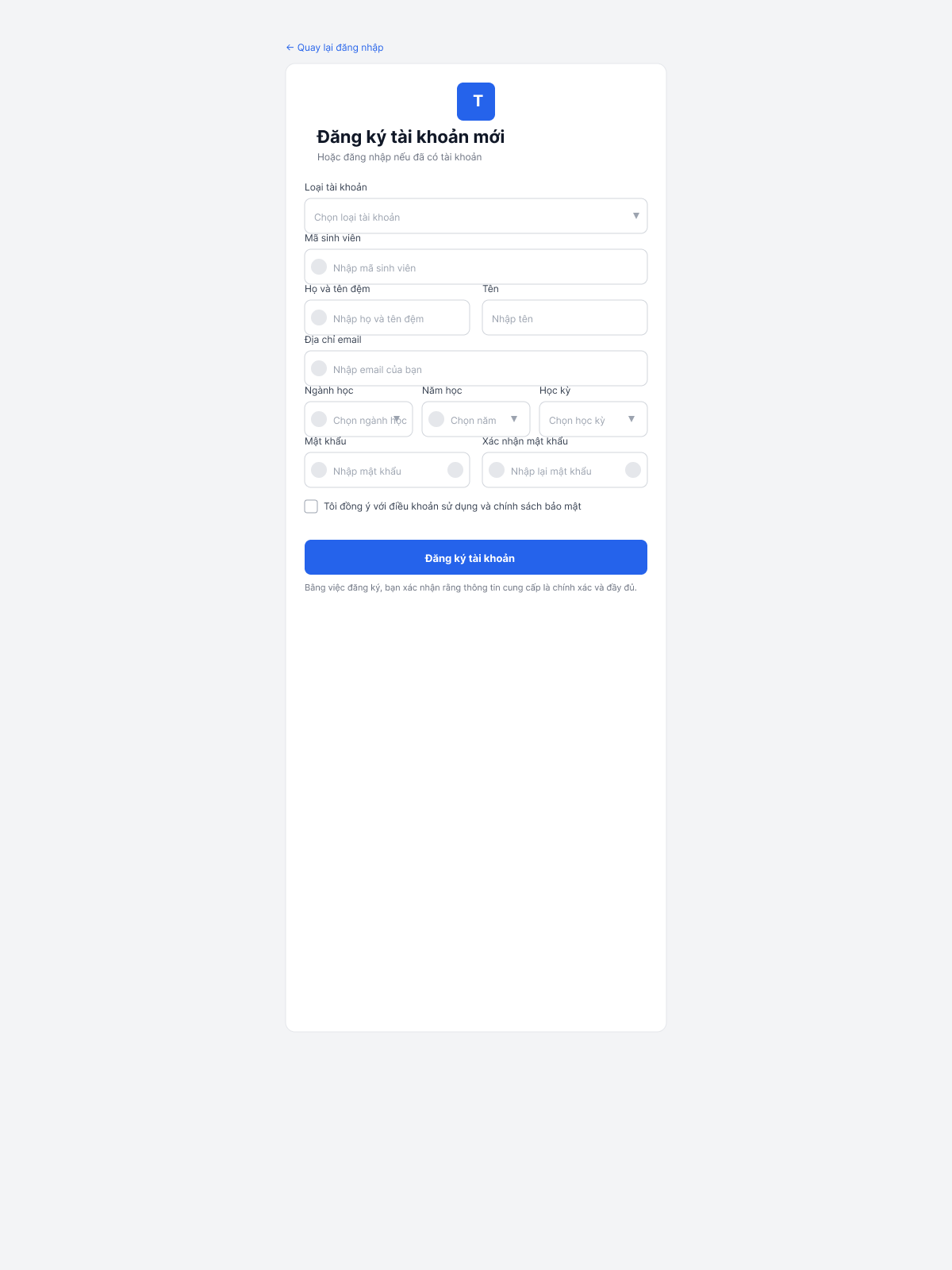
**2.2.7 Giao diện hiển thị các khóa học đã đăng kí**

****

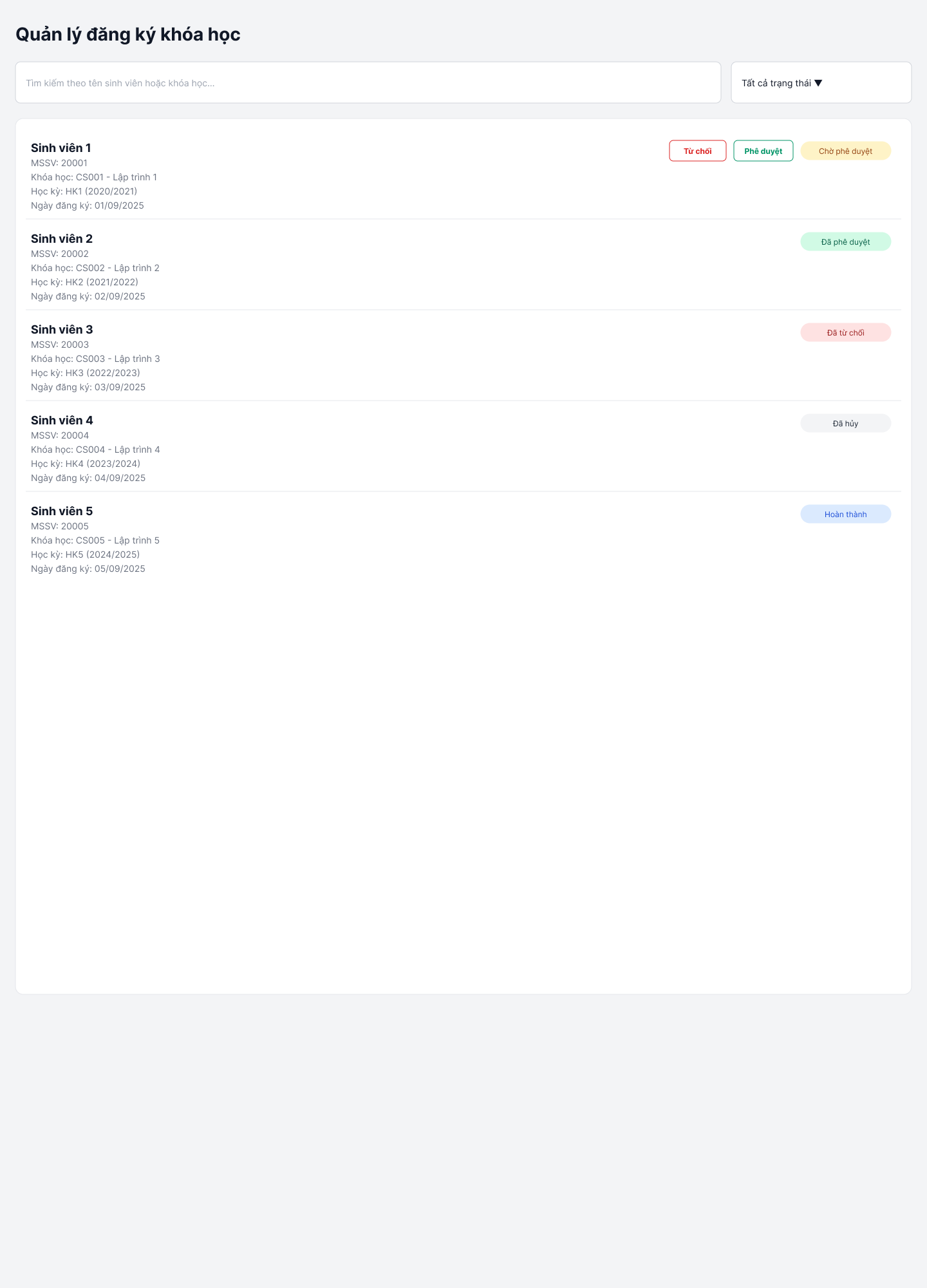
**2.2.8 Giao diện hồ sơ đăng kí**

****

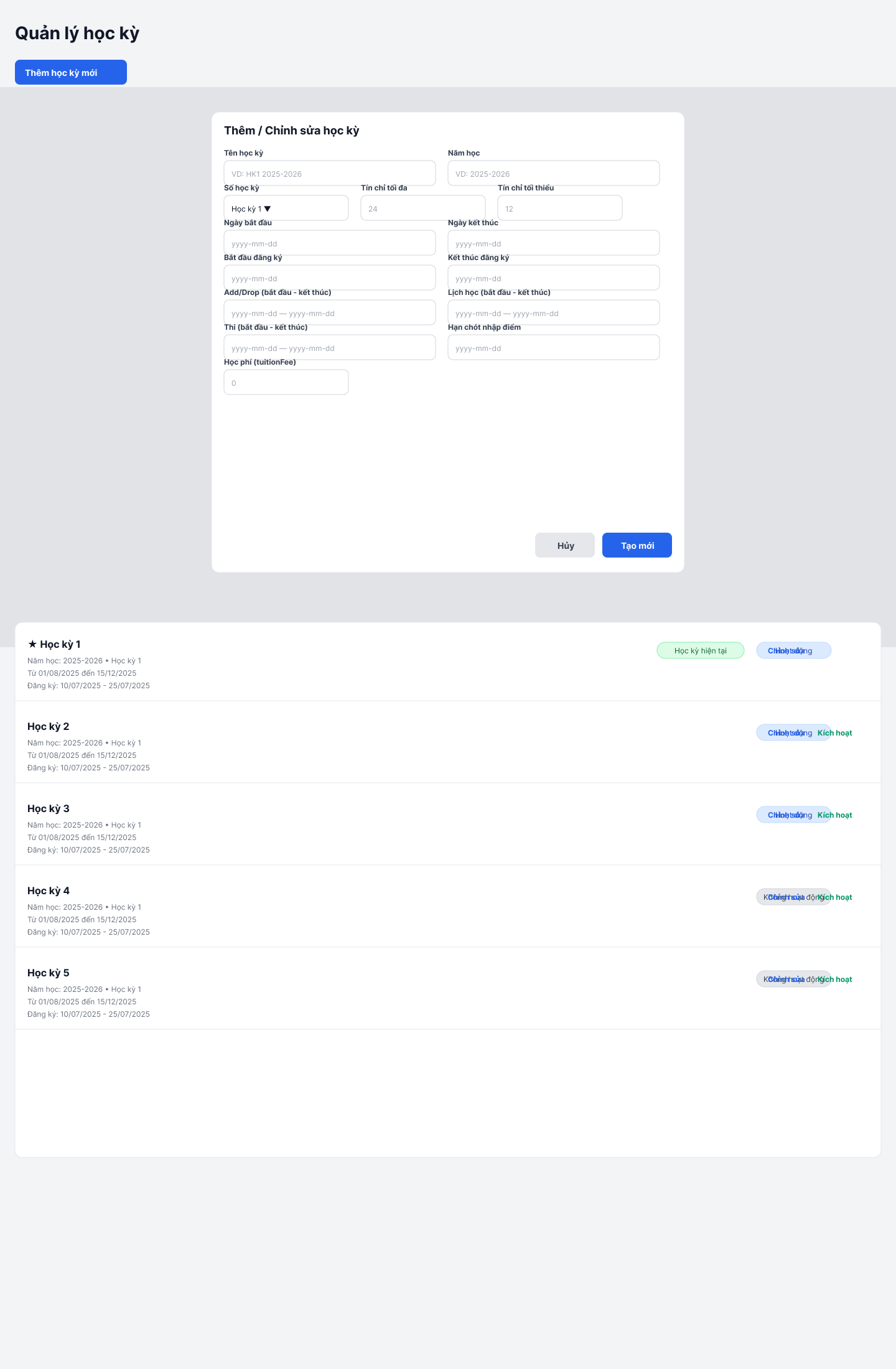
**2.2.9 Giao diện đăng kí**

****

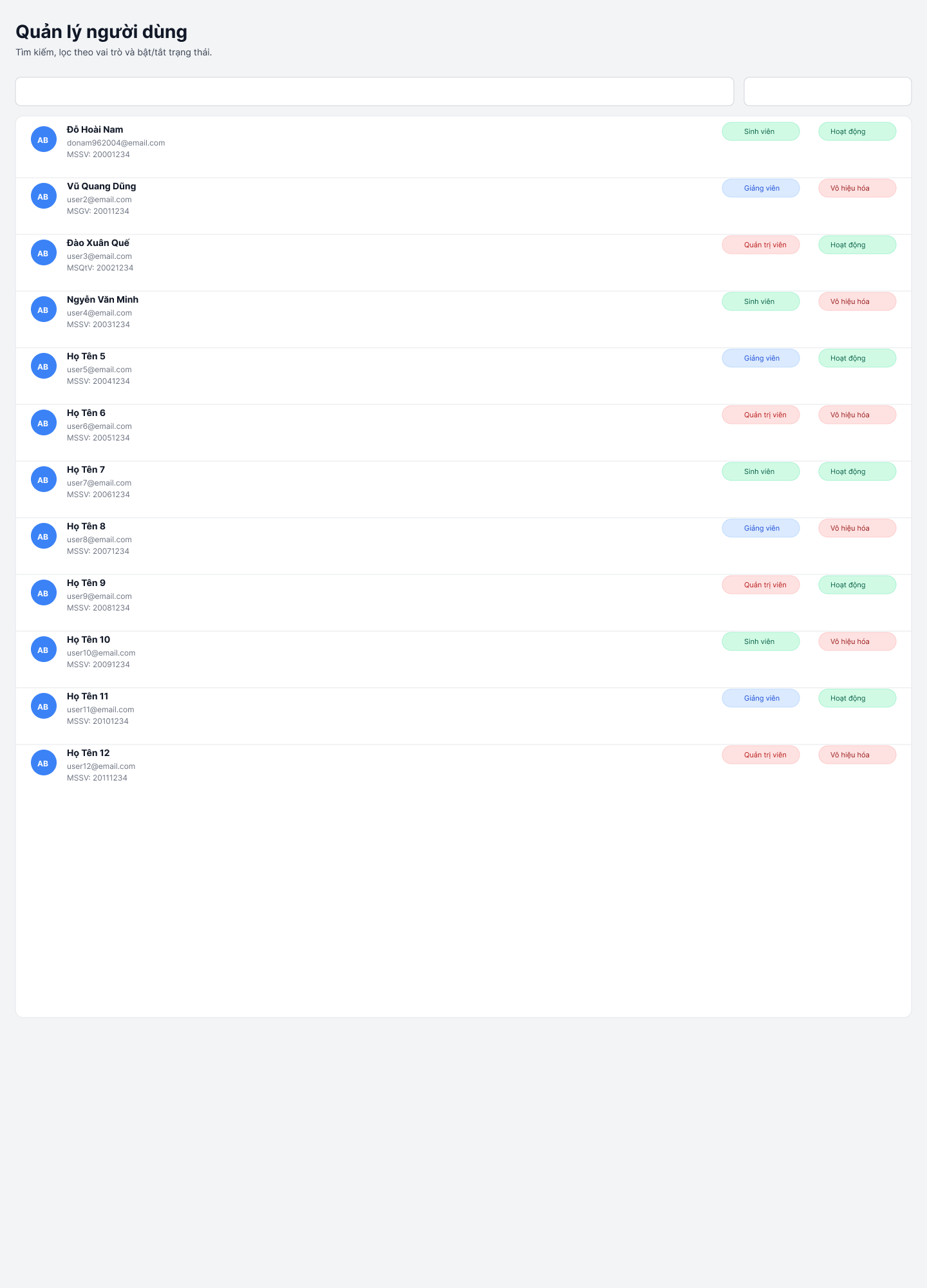
**2.2.10 Giao diện quản lí đăng kí**

****

**2.2.11 Giao diện quản lí kì học**

****

**2.2.12 Giao diện quản lí người dùng**

****

**3. Kết quả đạt được**

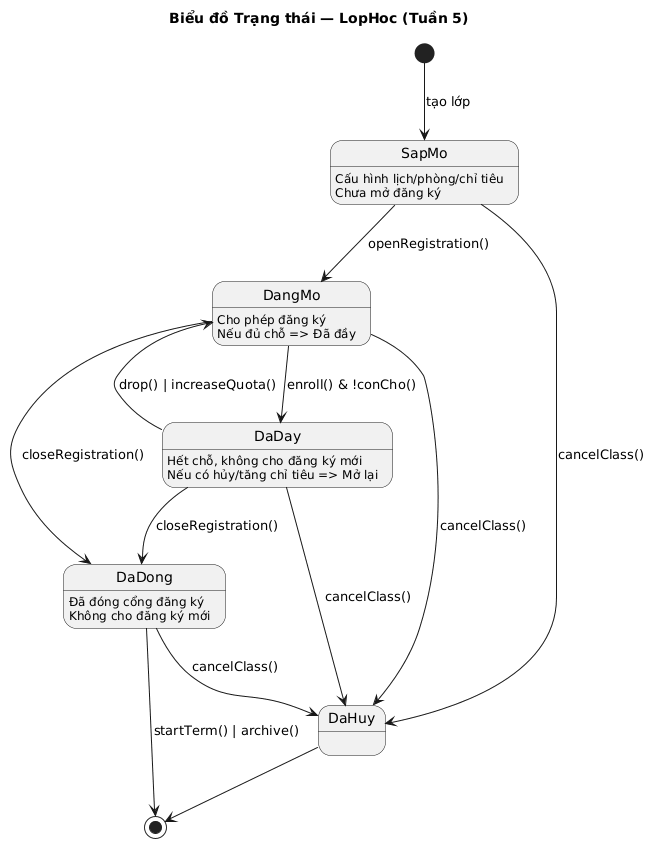
* Hoàn thành **Biểu đồ Trình tự cho Use Case Đăng ký môn học**, thể hiện rõ các đối tượng, thông điệp và điều kiện kiểm tra.
* Hoàn thành **bản phác thảo giao diện đăng ký môn học**, trực quan và dễ hiểu cho người dùng.
* Xác định được các tình huống lỗi và xử lý ngoại lệ trong cả biểu đồ trình tự lẫn giao diện.

**4. Đánh giá và khó khă**

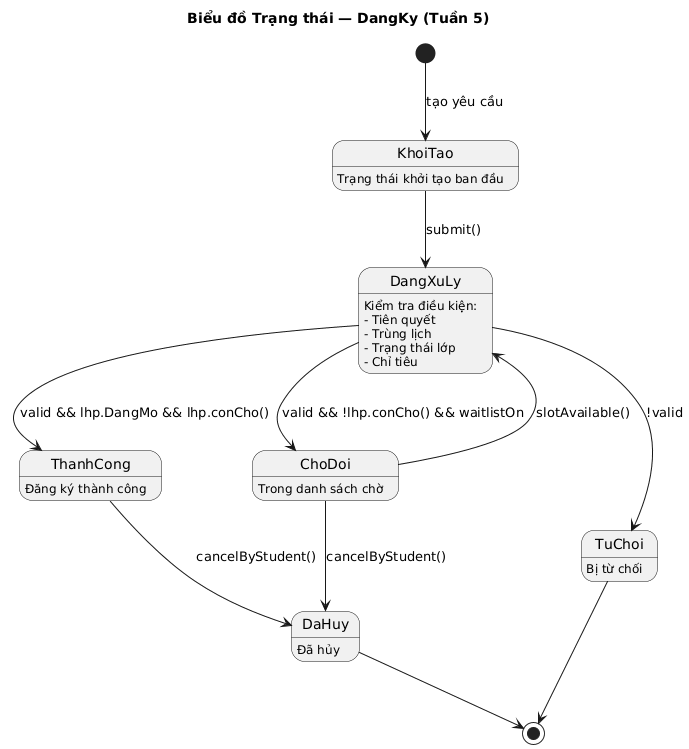
* **Ưu điểm:**
  + Nhóm đã mô tả được chi tiết cách các đối tượng tương tác.
  + Giao diện được thiết kế trực quan, dễ thao tác.
  + Mối liên hệ giữa **thiết kế UML** và **giao diện người dùng** được thể hiện rõ ràng.
* **Khó khăn:**
  + Việc vẽ Biểu đồ Trình tự có nhiều đối tượng, cần thống nhất cách đặt tên.
  + Thiết kế giao diện ban đầu còn đơn giản, cần cải tiến thêm về trải nghiệm người dùng (UX).

**Tuần 5: Thiết kế Hành vi và Trạng thái**

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

****

**LopHoc**: vòng đời từ **Sắp mở** → **Đang mở** → **(Đã đầy** ↔ **Đang mở)** → **Đã đóng / Đã hủy.**Sự kiện chính**: openRegistration()**, **enroll()**, **drop()**, **increaseQuota()**, **closeRegistration()**, **cancelClass()**.

****

**DangKy**: vòng đời từ **Khởi tạo** → **Đang xử lý** → **(Thành công / Chờ đợi / Từ chối)** → **Đã hủy (khi SV tự hủy).**Sự kiện chính: **submit()**, **markValid()**, **confirm()**, **waitlist()**, **resumeFromWaitlist()**, **reject()**, **cancel().**

**2. Cập nhật code để quản lý trạng thái**

Cập nhật **src/models/Dangky.js**:

* Thêm **enum trạng thái** DangKyStates (KHOI\_TAO, DANG\_XU\_LY, THANH\_CONG, CHO\_DOI, TU\_CHOI, DA\_HUY).
* Đặt trangThai **default** KHOI\_TAO, ngayDangKy default NOW.
* Thêm **instance methods** để điều khiển state:
  + submit(), markValid(), confirm(), waitlist(), resumeFromWaitlist(), reject(reason), cancel().

Cập nhật **src/models/LopHocPhan.js**:

* Thêm **enum** LopHocPhanStates (SAP\_MO, DANG\_MO, DA\_DAY, DA\_DONG, DA\_HUY).
* Bổ sung cột **chiTieu**, **siSo**, **trangThai** (default SAP\_MO).
* Thêm **logic & helper**:
  + conCho(), openRegistration(), closeRegistration(), cancelClass(reason),
  + themSinhVien() (tự chuyển **DANG\_MO → DA\_DAY** khi vừa đầy),
  + xoaSinhVien() (tự chuyển **DA\_DAY → DANG\_MO** khi có chỗ),
  + setChiTieu(newQuota) (nếu đang đầy mà tăng quota → về **DANG\_MO**).

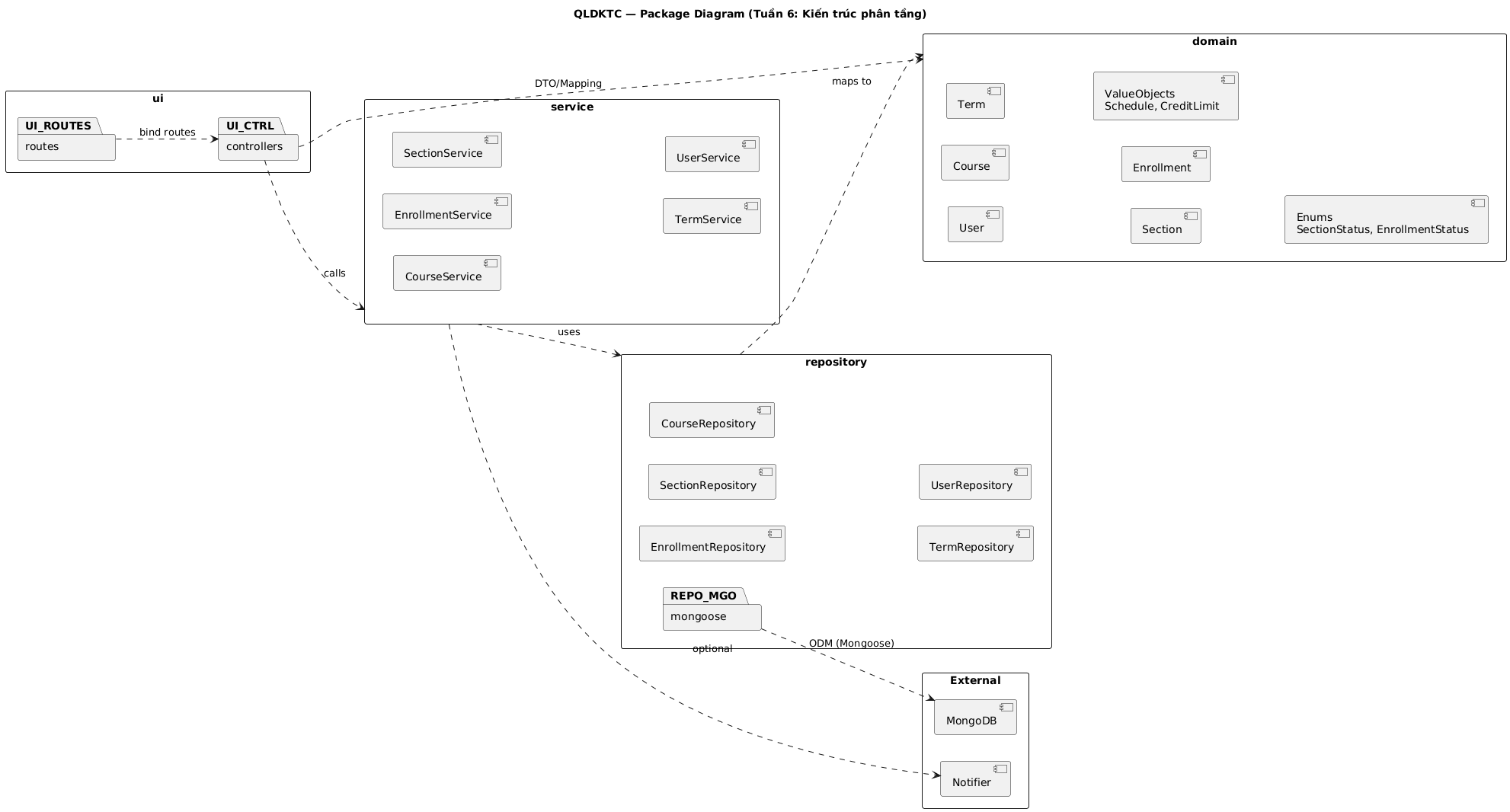
**Báo cáo tuần 6. Thiết kế kiến trúc hệ thống**

**1 Tổng quan kiến trúc & phạm vi**

Hệ thống được tổ chức theo **kiến trúc phân tầng** nhằm tách biệt giao diện – nghiệp vụ – truy cập dữ liệu – miền nghiệp vụ:

* **ui**: giao tiếp HTTP (backend) và màn hình SPA (frontend).
* **service**: xử lý nghiệp vụ thuần (không phụ thuộc framework).
* **repository**: tương tác cơ sở dữ liệu (ẩn chi tiết Mongoose).
* **domain**: định nghĩa entity/value object/enum độc lập hạ tầng.
* **external**: MongoDB, Notifier (Email/SMS)…

**2 Thiết kế biểu đồ gói**



**2.1 Các gói & trách nhiệm**

* **ui**
  + routes/: định tuyến REST (không chứa logic).
  + controllers/: nhận request, validate, gọi service, trả response.
* **service**
  + Thực thi **quy tắc nghiệp vụ**: kiểm tra cửa sổ đăng ký, quota, waitlist, promote từ waitlist khi hủy…
  + Chỉ “biết” đến **repository** qua interface → dễ mock để test.
* **repository**
  + Che giấu **ODM (Mongoose)**, ánh xạ Domain ↔ MongoDB.
  + Trả về DTO/record thuần, không “rò rỉ” document Mongoose ra ngoài.
* **domain**
  + Enums: SectionStatus, EnrollmentStatus.
  + (khuyến nghị) ValueObjects: Schedule, CreditLimit.
  + (tùy chọn) Entity nếu cần bất biến/logic riêng trong Domain.
* **external**
  + MongoDB: lưu toàn bộ dữ liệu.
  + Notifier: gửi email/SMS/in-app (mở rộng sau).

**2.2 Quan hệ phụ thuộc (dep)**

* ui → service → repository → domain.
* repository → MongoDB (Mongoose); service → Notifier (tùy chọn).
* ui.controllers → domain (mapping DTO).

**3 Domain (lõi nghiệp vụ)**

**3.1 domain/valueobjects/Email.java**

* **Mục đích:** Biểu diễn email như một **Value Object** bất biến, tự kiểm tra hợp lệ.
* **Thiết kế:**
  + Dùng record Email(String value) để bất biến, so sánh theo giá trị, dễ dùng làm key.
  + Validate ngay trong canonical constructor (regex đơn giản) → **đảm bảo bất biến nghiệp vụ** “email phải hợp lệ” tại **biên domain**.
* **Hệ quả tích cực:**
  + Mọi nơi tạo Email đều được “bảo hiểm” tính hợp lệ → giảm rò rỉ lỗi xuống DB/UI.
* **Giới hạn:**
  + Regex tối giản; cần thay thế bằng validator mạnh hơn nếu có i18n hoặc quy tắc nội bộ.

**3.2 domain/entities/User.java**

* **Mục đích:** Thực thể cốt lõi với danh tính (id), thuộc tính (email), trạng thái (active).
* **Thiết kế:**
  + Kiểm tra id rỗng ngay trong constructor → bất biến danh tính.
  + Trạng thái kích hoạt (active) mặc định true, có hành vi deactivate() → thể hiện **hành vi domain**, không chỉ là data bag.
* **Quy tắc biên (Boundaries):**
  + Không biết gì về hạ tầng (DB, HTTP).
  + Không tham chiếu sang tầng service/ui.

**4 Repository (truy cập dữ liệu)**

**4.1 repository/interfaces/IUserRepository.java**

* **Vai trò (Contract):** Abstraction cho truy xuất User trong domain.
* **Phương thức:**
  + save(User user) → trả về Entity domain sau khi lưu.
  + findById(String id) → Optional<User>.
  + existsByEmail(Email email) → phục vụ quy tắc duy nhất (unique) do **service** quyết định.
* **Nguyên tắc:**
  + **Trả về/nhận vào Entity/VO domain** (không DTO), giữ sạch ranh giới.

**4.2 repository/implementations/UserRepositoryImpl.java**

* **Mục đích:** Triển khai tối thiểu **in-memory** để minh hoạ và phục vụ test.
* **Thiết kế:**
  + Lưu trữ Map<String, User> db → thay thế nhanh bằng ORM/SQL sau này.
  + Duy trì “duy nhất theo email” bằng cách duyệt values() trong existsByEmail.
* **Rủi ro/giới hạn:**
  + Không thread-safe (dùng HashMap). Nếu tích hợp thật, thay bằng:
    - Lớp infra/DAO với JPA/JDBC hoặc client HTTP, và/hoặc
    - ConcurrentHashMap cho demo đa luồng.
* **Mở rộng:**
  + Khi dùng ORM, thêm lớp persistence model + mapper 2 chiều (Entity ↔ Persistence).

**4.3 repository/mappers/UserMapper.java**

* **Trạng thái:** Placeholder.
* **Ý nghĩa:** Chỗ đặt logic mapping khi persistence model ≠ domain entity (ví dụ có bảng, cột, lazy fields, …).

**5 Service (use-cases ứng dụng)**

**5.1 DTOs**

* service/dto/CreateUserRequest.java
  + **Input từ UI vào Service**. Chứa id, email.
* service/dto/UserDto.java
  + **Output từ Service ra UI**. Chứa id, email, active.
* **Quy tắc:** Không để Entity domain “lọt” ra UI; luôn chuyển đổi qua DTO.

**5.2 service/interfaces/IUserService.java**

* **Vai trò:** Định nghĩa các **use-case** cho người dùng:
  + create(CreateUserRequest req) → UserDto
  + get(String id) → Optional<UserDto>
* **Ý nghĩa:** Tầng UI chỉ biết đến **hợp đồng** này; không quan tâm repo chạy bằng gì.

**5.3 service/validators/UserValidator.java**

* **Mục đích:** Validation ở **application-level** (khác domain-level):
  + Kiểm tra rỗng/format input trước khi chuyển sang domain.
* **Phân tầng validation:**
  + UI-level: kiểm tra form cơ bản (ở Controller/Client).
  + Service-level: ràng buộc use-case (trường bắt buộc).
  + Domain-level: bất biến (Email regex, id rỗng…).

**5.4 service/implementations/UserService.java**

* **Luồng create:**
  1. UserValidator.validate(req) → chặn input xấu sớm.
  2. Khởi tạo User + Email (domain) → domain tự kiểm tra bất biến.
  3. repo.existsByEmail(email) → enforce “duy nhất” trước save.
  4. repo.save(user) → lưu, trả về entity.
  5. Map sang UserDto → trả lên UI.
* **Luồng get:**
  1. repo.findById(id) → map sang UserDto nếu có.
* **Quyết định thiết kế quan trọng:**
  1. Service **chỉ** phụ thuộc IUserRepository (abstraction) → dễ thay thế hạ tầng, dễ mock trong unit test.
  2. **Không** để DTO đi xuống repository; **Không** để Entity đi lên UI.
* **Xử lý lỗi:**
  1. IllegalStateException("Duplicated email") cho case trùng email (demo). Thực tế: nên dùng **Exception nghiệp vụ** riêng và map sang HTTP 409 ở UI/API.

**6 UI (giao diện/adapter)**

**6.1 ui/controllers/UserController.java**

* **Mục đích:** Lớp controller mô phỏng, **không phụ thuộc framework** (giúp skeleton chạy độc lập).
* **Hành vi:**
  + create(CreateUserRequest) → gọi IUserService.create().
  + get(String id) → gọi IUserService.get().
* **Biên giới rõ ràng:** Controller **không** viết logic nghiệp vụ; chỉ uỷ quyền và chuyển đổi request/response.

**7 Tests (đảm bảo chất lượng)**

**7.1 tests/unit/domain/UserTest.java**

* **Kiểm tra bất biến domain:**
  + Tạo User hợp lệ → active == true, id đúng.
  + Tạo Email sai regex → ném IllegalArgumentException.
* **Ý nghĩa:** Bất biến domain **không phụ thuộc** UI/Service.

**7.2 tests/unit/service/UserServiceTest.java**

* **Kiểm tra luồng use-case:**
  + Dùng repo in-memory thật (UserRepositoryImpl) để đơn giản.
  + create rồi get lại → dữ liệu, trạng thái đúng.
* **Ý nghĩa:** Chứng minh service **kết hợp đúng** giữa validation, domain, repository và mapping DTO.

**7.3 tests/integration/repository/UserRepositoryTest.java**

* **Kiểm tra persistence tối thiểu:**
  + save rồi findById → trả về entity đúng.
* **Ý nghĩa:** Dù là in-memory, test này đặt **khung** để sau thay ORM/DB thật.