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**

**3.1. Use Case cho Sinh viên**

**Use Case 1: Đăng ký môn học**

* **Actor**: Sinh viên
* **Mô tả**: Cho phép sinh viên đăng ký môn học trong học kỳ.
* **Điều kiện tiên quyết**: Sinh viên đã đăng nhập hệ thống và thời gian đăng ký còn hiệu lực.
* **Luồng sự kiến chính**:
  1. Sinh viên chọn chức năng “Đăng ký môn học”.
  2. Hệ thống hiển thị danh sách môn học mở trong kỳ.
  3. Sinh viên chọn môn học muốn đăng ký.
  4. Hệ thống kiểm tra điều kiện tiên quyết (học phần bắt buộc, số tín chỉ còn lại).
  5. Nếu hợp lệ, hệ thống xác nhận đăng ký và cập nhật vào cơ sở dữ liệu.
  6. Hệ thống hiển thị thông báo “Đăng ký thành công”.
* **Luồng thay thế**: Nếu môn học đã đầy hoặc sinh viên không đủ điều kiện, hệ thống báo lỗi.
* **Điều kiện sau**: Môn học được thêm vào danh sách học tập của sinh viên.

**Use Case 2: Hủy đăng ký môn học**

* **Actor**: Sinh viên
* **Mô tả**: Cho phép sinh viên hủy môn đã đăng ký trước khi hết hạn.
* **Điều kiện tiên quyết**: Sinh viên đã đăng nhập và môn học chưa quá hạn hủy.
* **Luồng sự kiến chính**:
  1. Sinh viên chọn “Hủy đăng ký”.
  2. Hệ thống hiển thị danh sách môn đã đăng ký.
  3. Sinh viên chọn môn muốn hủy.
  4. Hệ thống xác nhận và xóa môn khỏi danh sách đăng ký.
* **Điều kiện sau**: Môn học bị xóa khỏi lịch đăng ký của sinh viên.

**Use Case 3: Xem thời khóa biểu**

* **Actor**: Sinh viên
* **Mô tả**: Hiển thị thời khóa biểu theo học kỳ.
* **Điều kiện tiên quyết**: Sinh viên đã đăng nhập.
* **Luồng sự kiến chính**:
  1. Sinh viên chọn “Xem thời khóa biểu”.
  2. Hệ thống hiển thị danh sách môn học kèm lịch học chi tiết.
* **Điều kiện sau**: Sinh viên xem được lịch học cá nhân.

**Use Case 4: Xem kết quả học tập**

* **Actor**: Sinh viên
* **Mô tả**: Cho phép sinh viên tra cứu điểm các môn đã học.
* **Điều kiện tiên quyết**: Sinh viên đã đăng nhập.
* **Luồng sự kiến chính**:
  1. Sinh viên chọn “Kết quả học tập”.
  2. Hệ thống hiển thị điểm từng môn theo học kỳ.
* **Điều kiện sau**: Sinh viên có thể in hoặc lưu kết quả.

**3.2. Use Case cho Giảng viên**

**Use Case 5: Mở lớp học**

* **Actor**: Giảng viên
* **Mô tả**: Cho phép giảng viên đăng ký mở lớp học phần.
* **Điều kiện tiên quyết**: Giảng viên đã đăng nhập và được phân công môn.
* **Luồng sự kiến chính**:
  1. Giảng viên chọn “Mở lớp học”.
  2. Hệ thống hiển thị danh sách môn mà giảng viên phụ trách.
  3. Giảng viên nhập thông tin lớp (số lượng sinh viên, lịch học).
  4. Hệ thống xác nhận và mở lớp.
* **Điều kiện sau**: Lớp học xuất hiện trong danh sách đăng ký cho sinh viên.

**Use Case 6: Cập nhật điểm sinh viên**

* **Actor**: Giảng viên
* **Mô tả**: Cho phép nhập và cập nhật điểm cho sinh viên.
* **Điều kiện tiên quyết**: Giảng viên đã đăng nhập, lớp học đã kết thúc.
* **Luồng sự kiến chính**:
  1. Giảng viên chọn “Cập nhật điểm”.
  2. Hệ thống hiển thị danh sách sinh viên trong lớp.
  3. Giảng viên nhập điểm từng sinh viên.
  4. Hệ thống lưu dữ liệu.
* **Luồng thay thế**: Nếu nhập sai định dạng, hệ thống báo lỗi.
* **Điều kiện sau**: Điểm được lưu và hiển thị cho sinh viên.

**Use Case 7: Xem danh sách sinh viên**

* **Actor**: Giảng viên
* **Mô tả**: Hiển thị danh sách sinh viên đăng ký lớp.
* **Điều kiện tiên quyết**: Giảng viên đã đăng nhập.
* **Luồng sự kiến chính**:
  1. Giảng viên chọn “Danh sách sinh viên”.
  2. Hệ thống hiển thị danh sách đầy đủ kèm thông tin liên hệ.
* **Điều kiện sau**: Giảng viên có thể tải về danh sách.

**3.3. Use Case cho Quản trị viên (Admin)**

**Use Case 8: Quản lý sinh viên**

* **Actor**: Admin
* **Mô tả**: Cho phép thêm, sửa, xóa thông tin sinh viên.
* **Điều kiện tiên quyết**: Admin đã đăng nhập.
* **Luồng sự kiến chính**:
  1. Admin chọn “Quản lý sinh viên”.
  2. Nhập thông tin sinh viên mới hoặc chỉnh sửa.
  3. Hệ thống lưu dữ liệu.
* **Điều kiện sau**: CSDL sinh viên được cập nhật.

**Use Case 9: Quản lý môn học**

* **Actor**: Admin
* **Mô tả**: Quản lý danh mục môn học.
* **Luồng sự kiến chính**:
  1. Admin chọn “Quản lý môn học”.
  2. Thêm mới, sửa đổi hoặc xóa môn học.
  3. Hệ thống cập nhật danh mục.
* **Điều kiện sau**: Danh mục môn học được cập nhật.

**Use Case 10: Quản lý lớp học**

* **Actor**: Admin
* **Mô tả**: Quản lý thông tin các lớp học.
* **Luồng sự kiến chính**:
  1. Admin chọn “Quản lý lớp học”.
  2. Thêm mới, sửa đổi hoặc đóng lớp học.
  3. Phân công giảng viên cho lớp.
  4. Hệ thống cập nhật dữ liệu.
* **Điều kiện sau**: Thông tin lớp học và giảng viên được cập nhật.

**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**

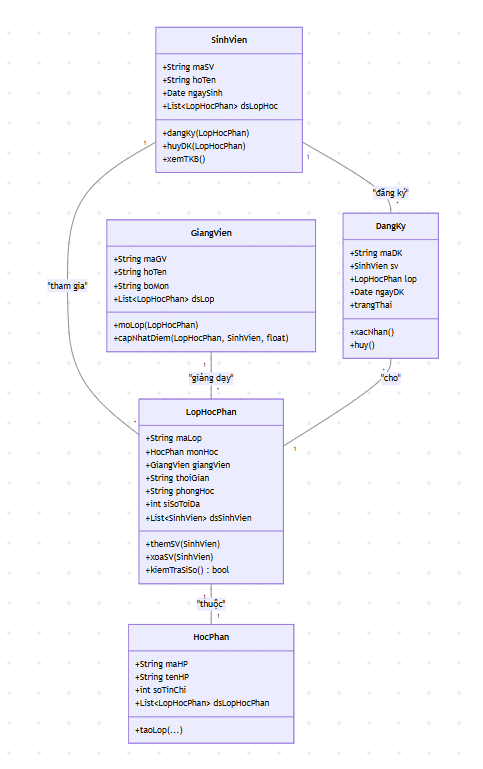
**Mục tiêu tuần 3**

* 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**

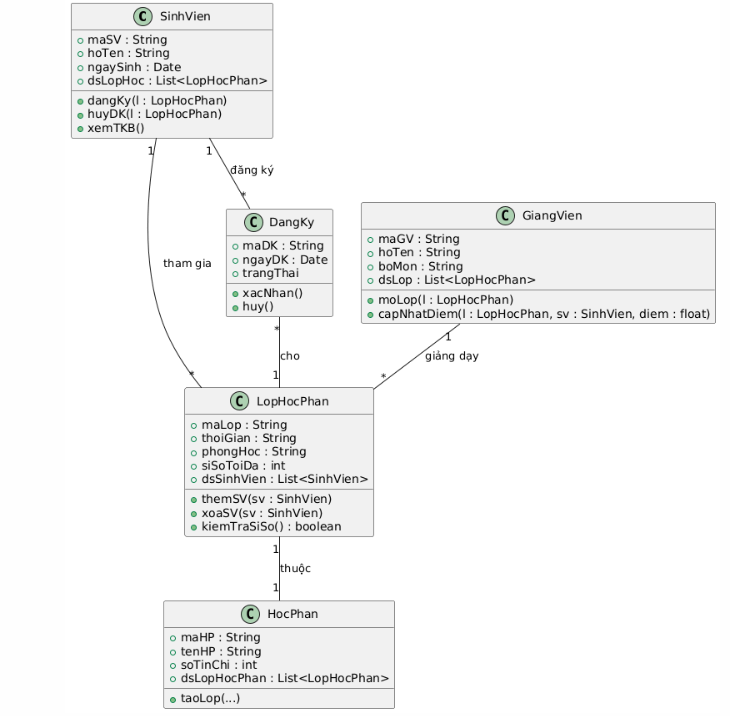
**1. Thiết kế Biểu đồ Lớp**

* 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**

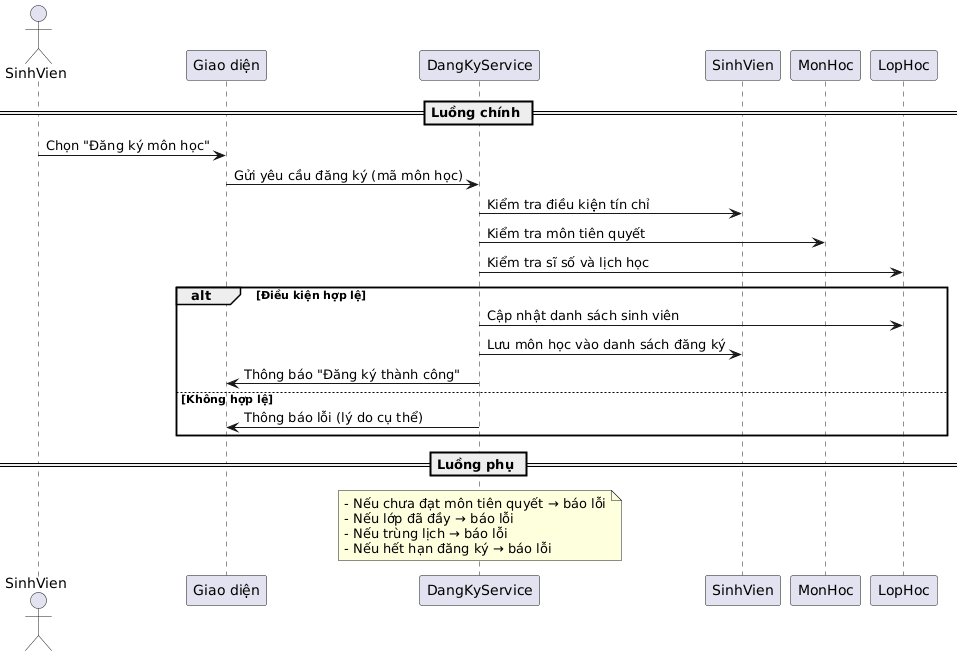
**1. Mục tiêu**

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)**

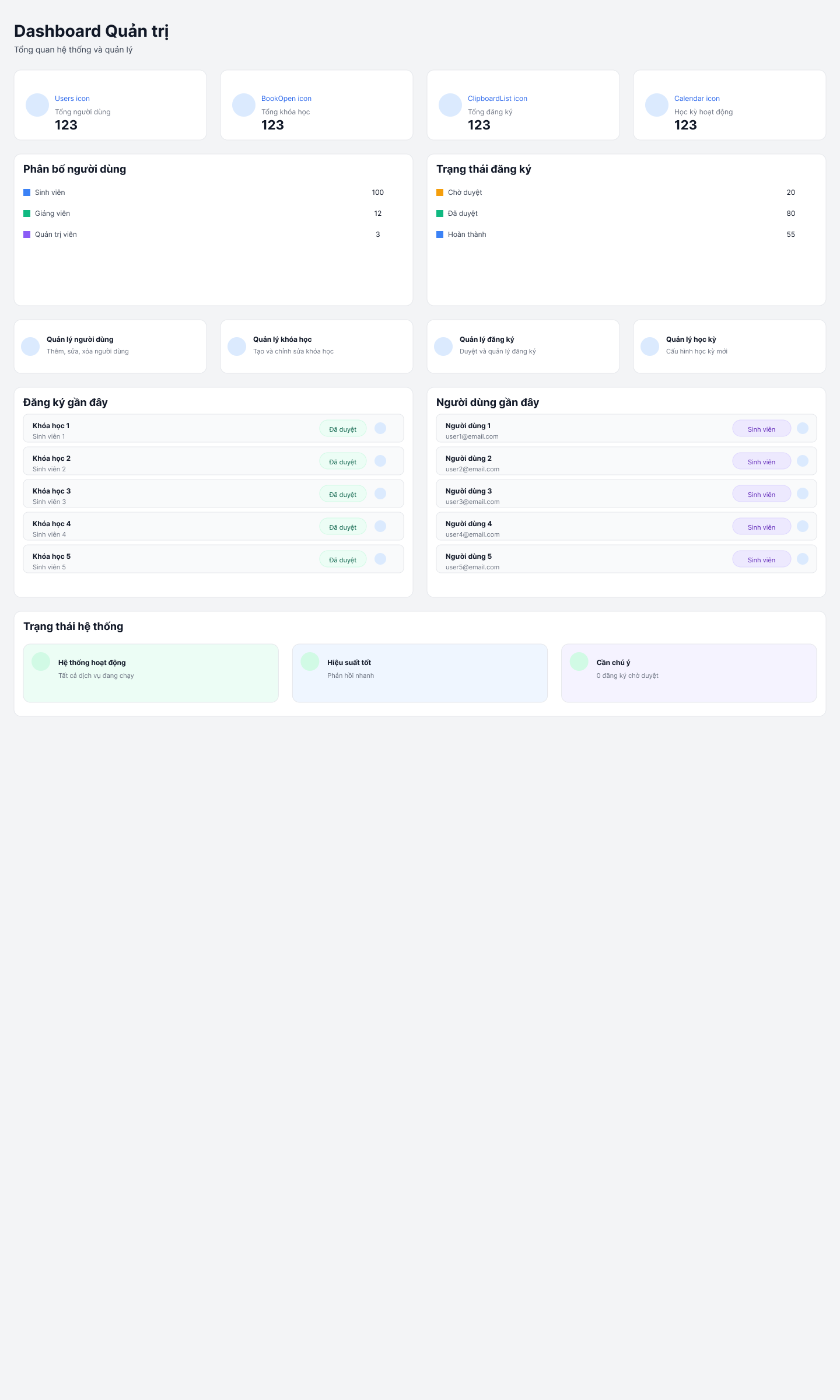
****

* **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.

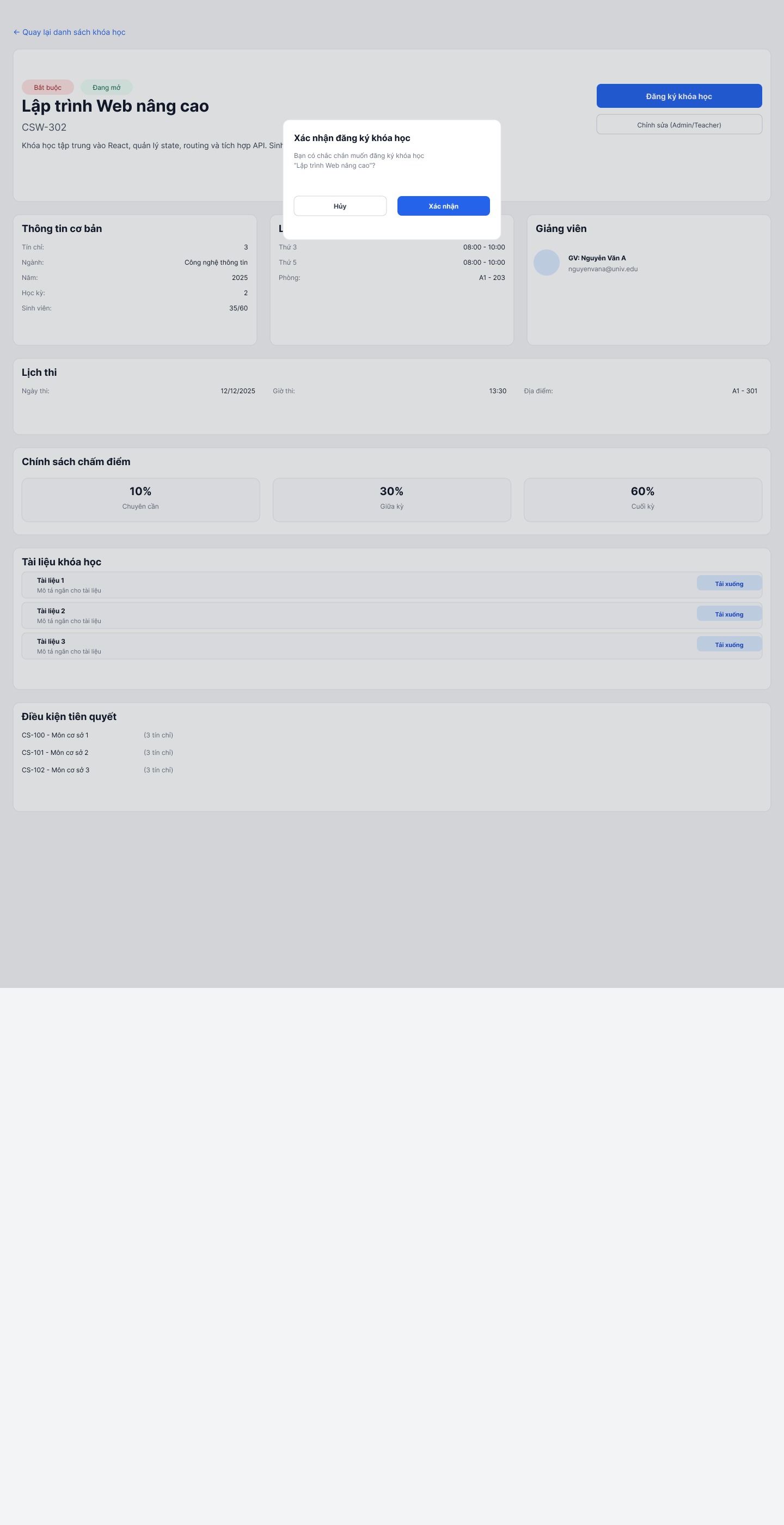
**2.2. Thiết kế Giao diện người dùng (UI)**

* **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.

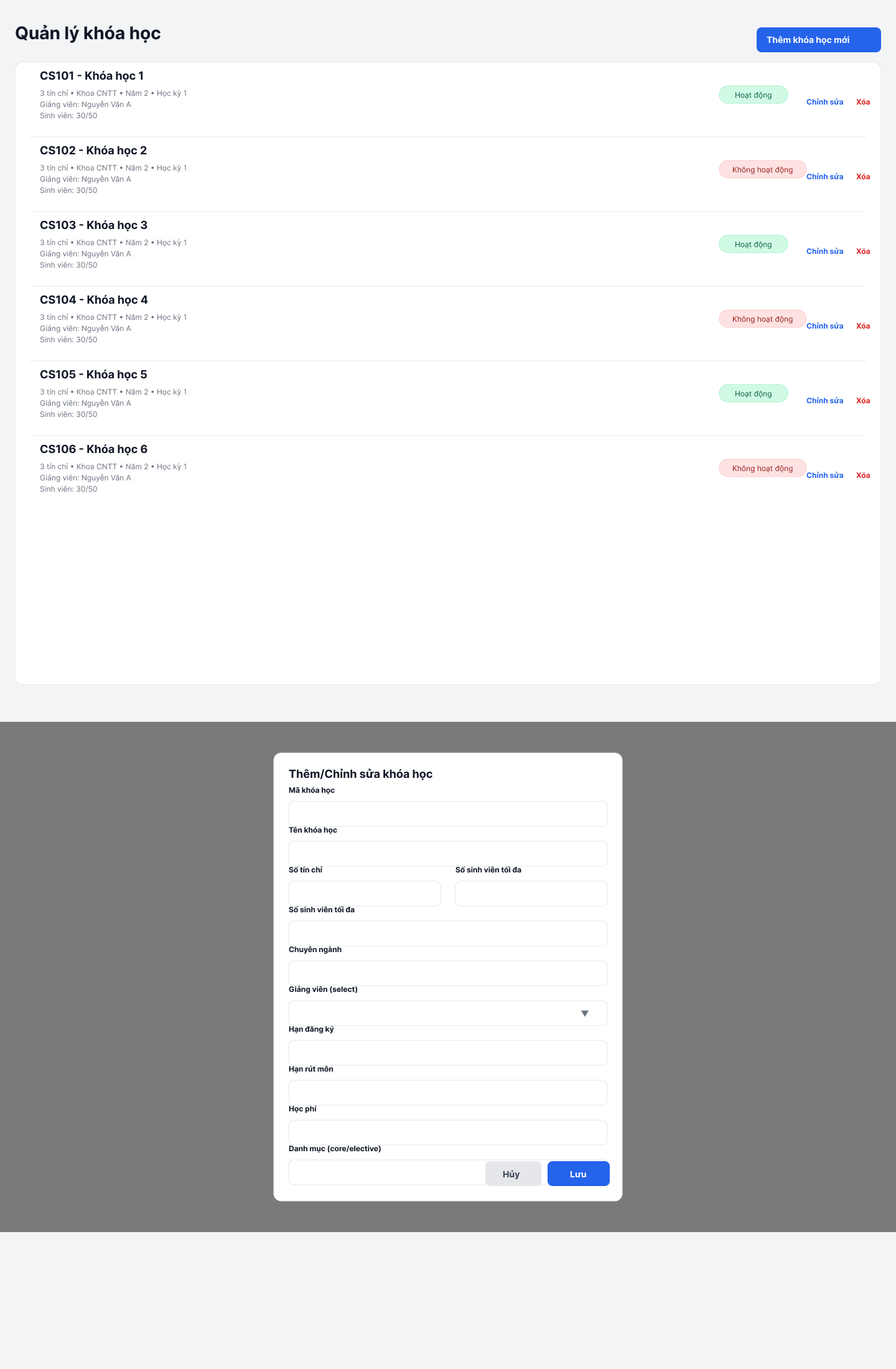
**2.2.1 Giao diện dashboard hệ thống**



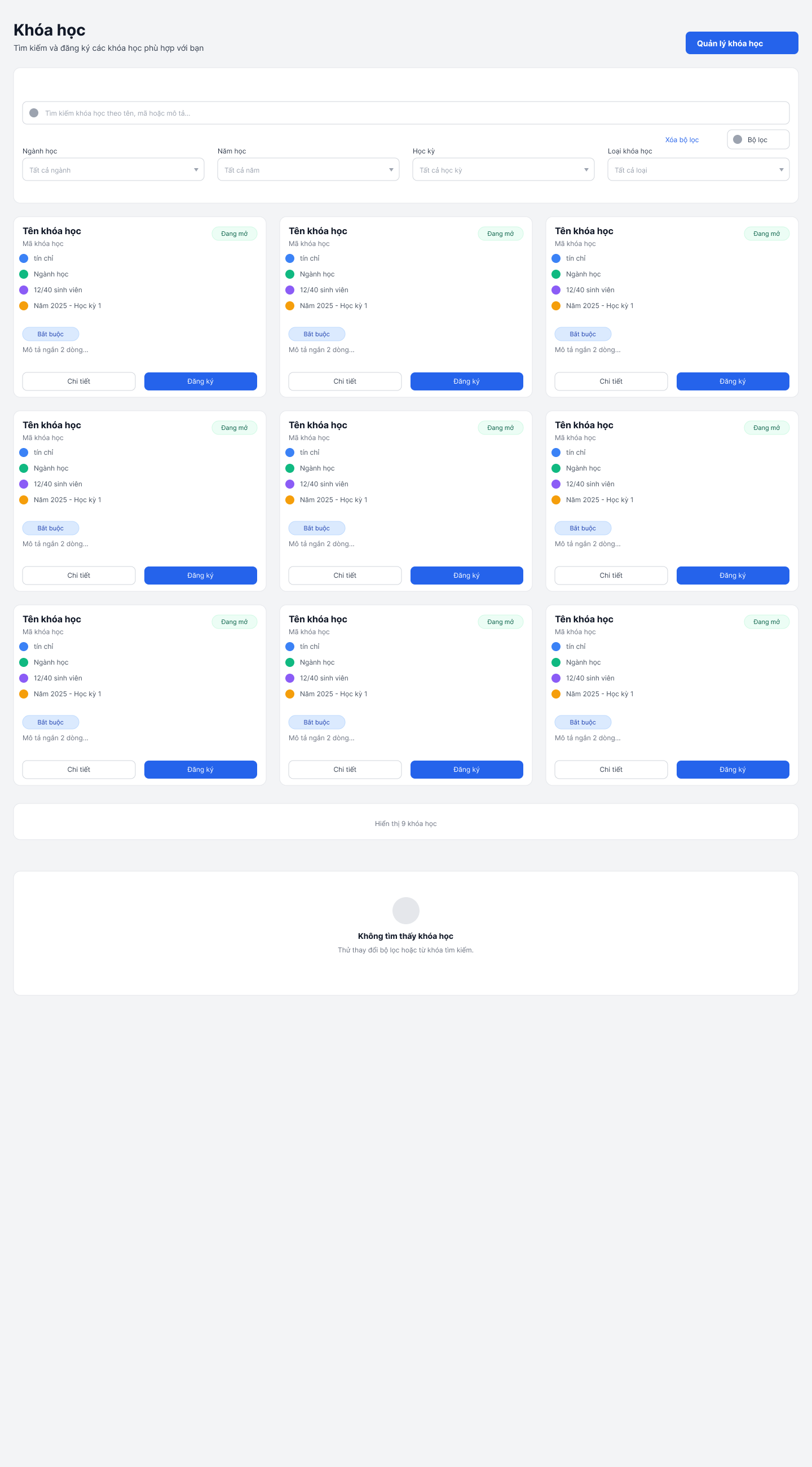
**2.2.1 Giao diện chi tiết đăng kí khóa học**

****

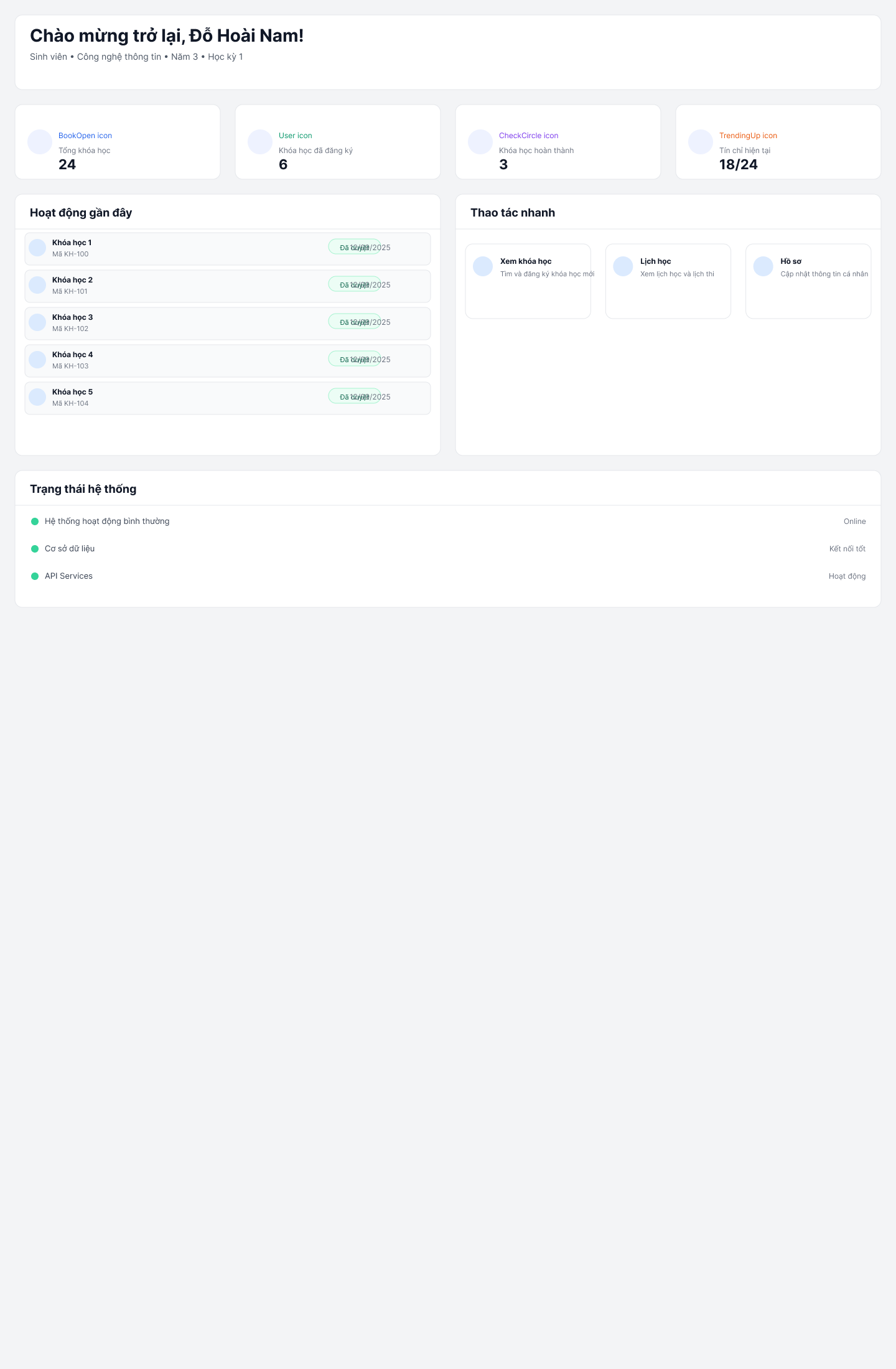
**2.2.3 Giao diện thêm chỉnh sửa khóa học**

****

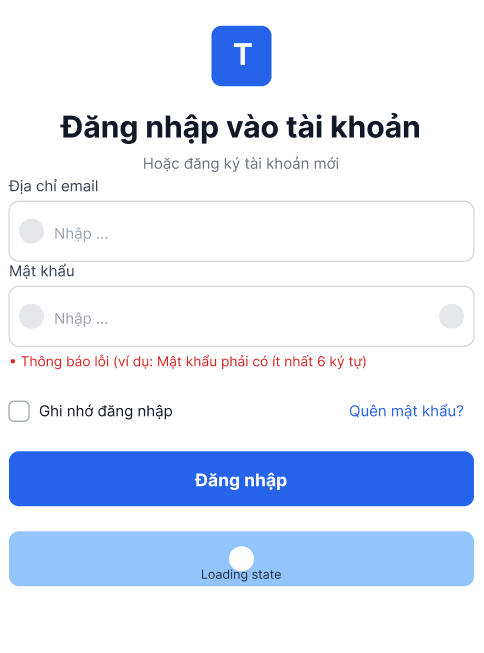
**2.2.4 Giao diện Quản lý Khóa học**

****

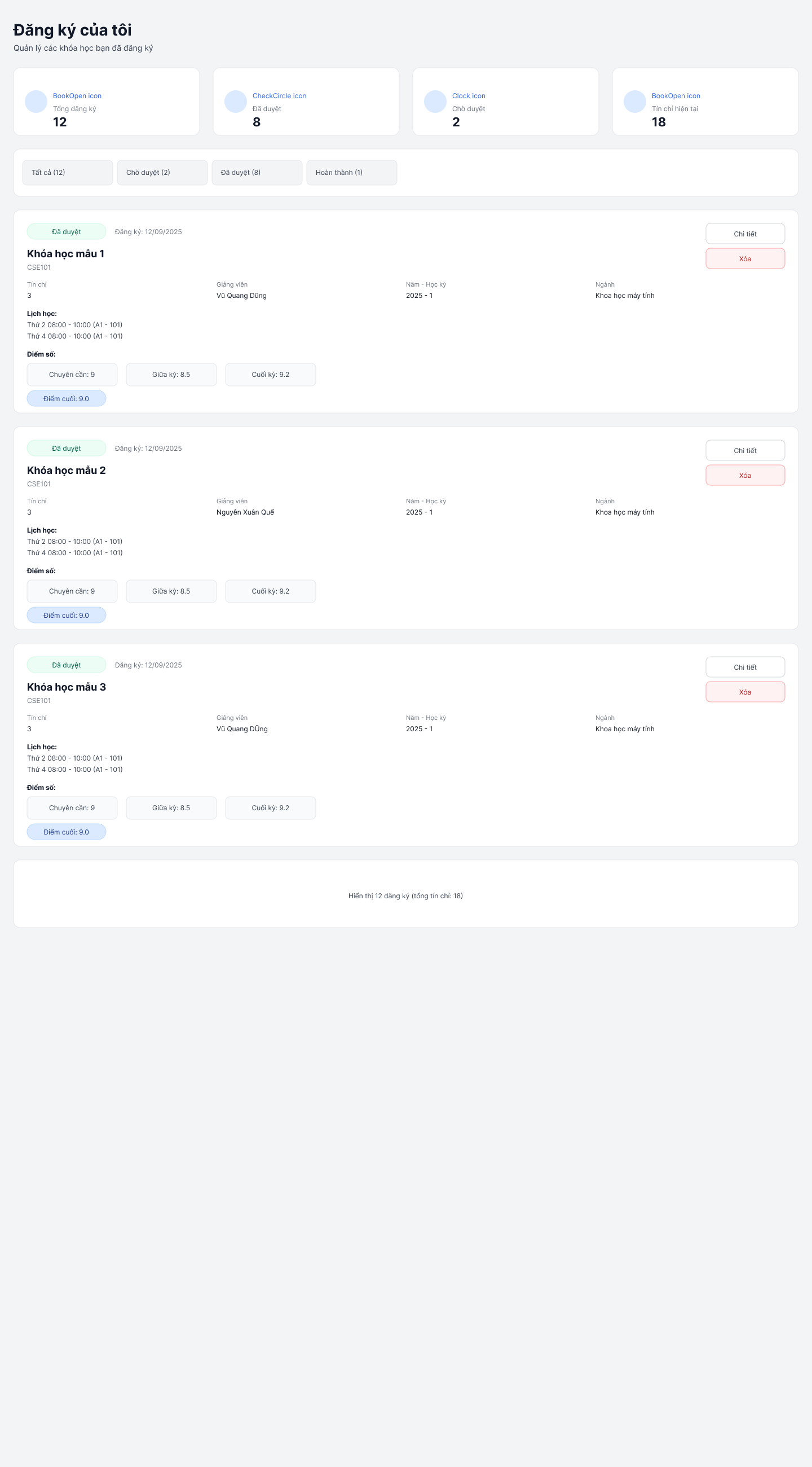
**2.2.5 Giao diện Chính**

****

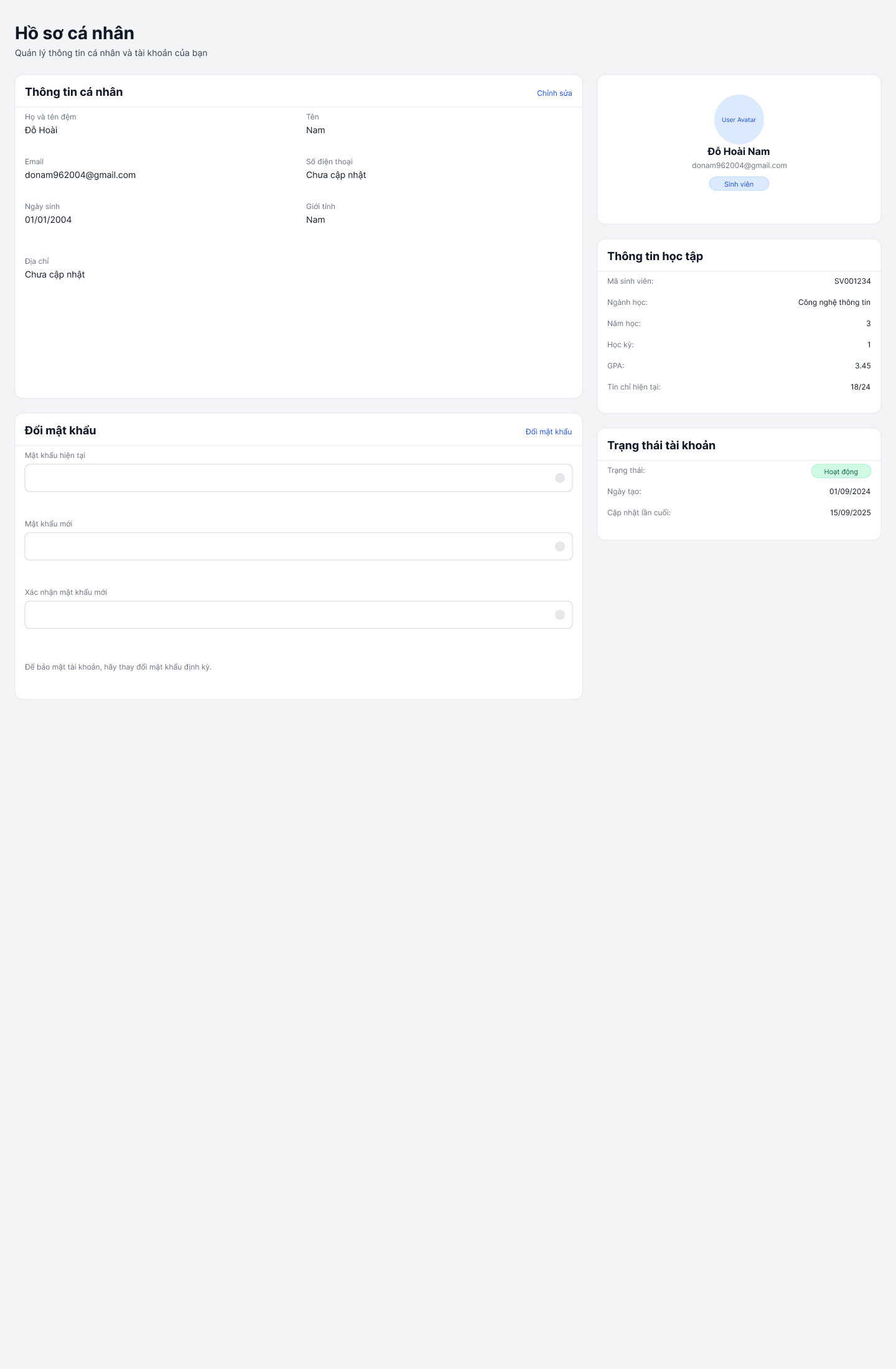
**2.2.6 Giao diện Login**

****

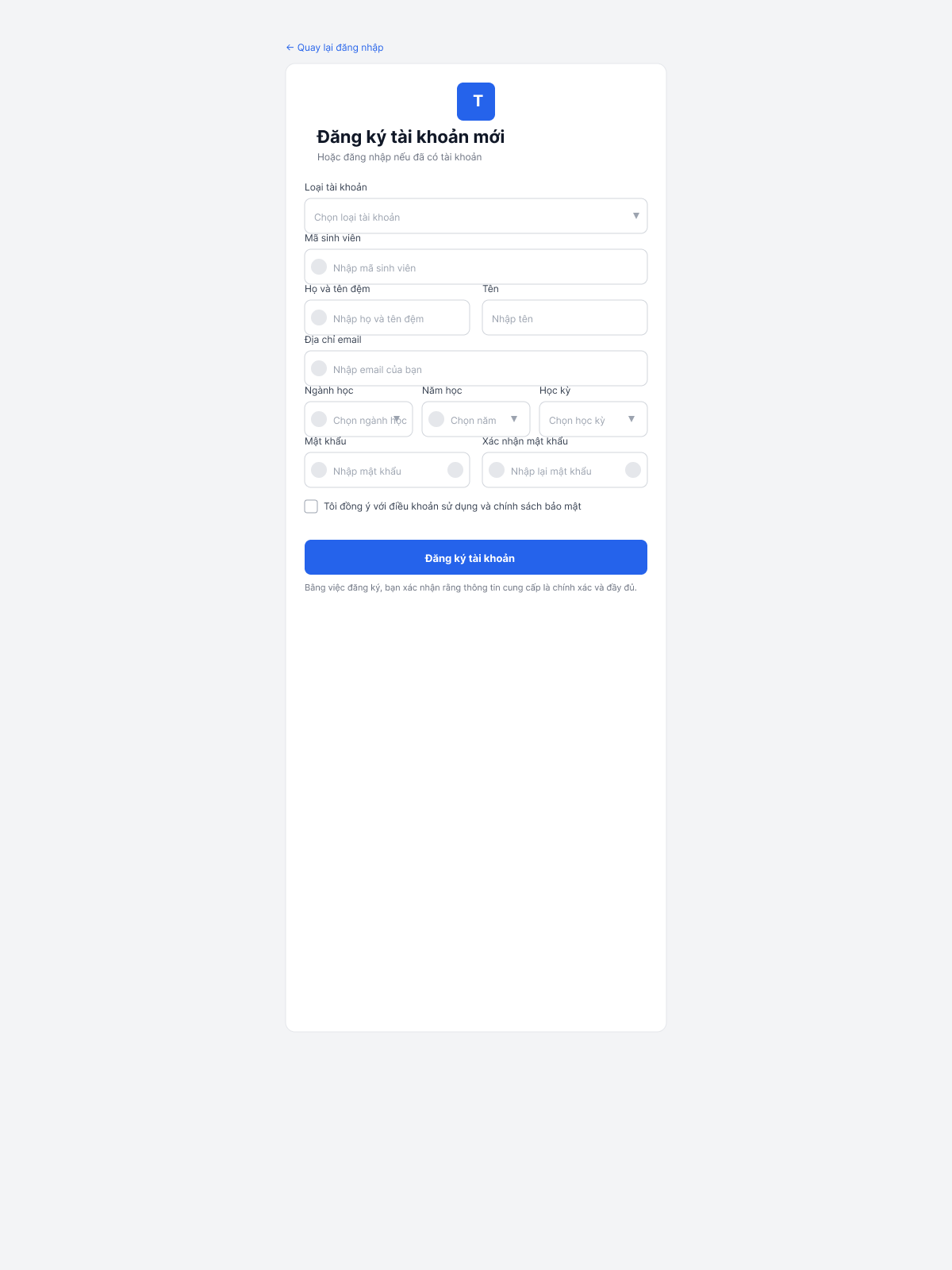
**2.2.7 Giao diện hiển thị các khóa học đã đăng kí**

****

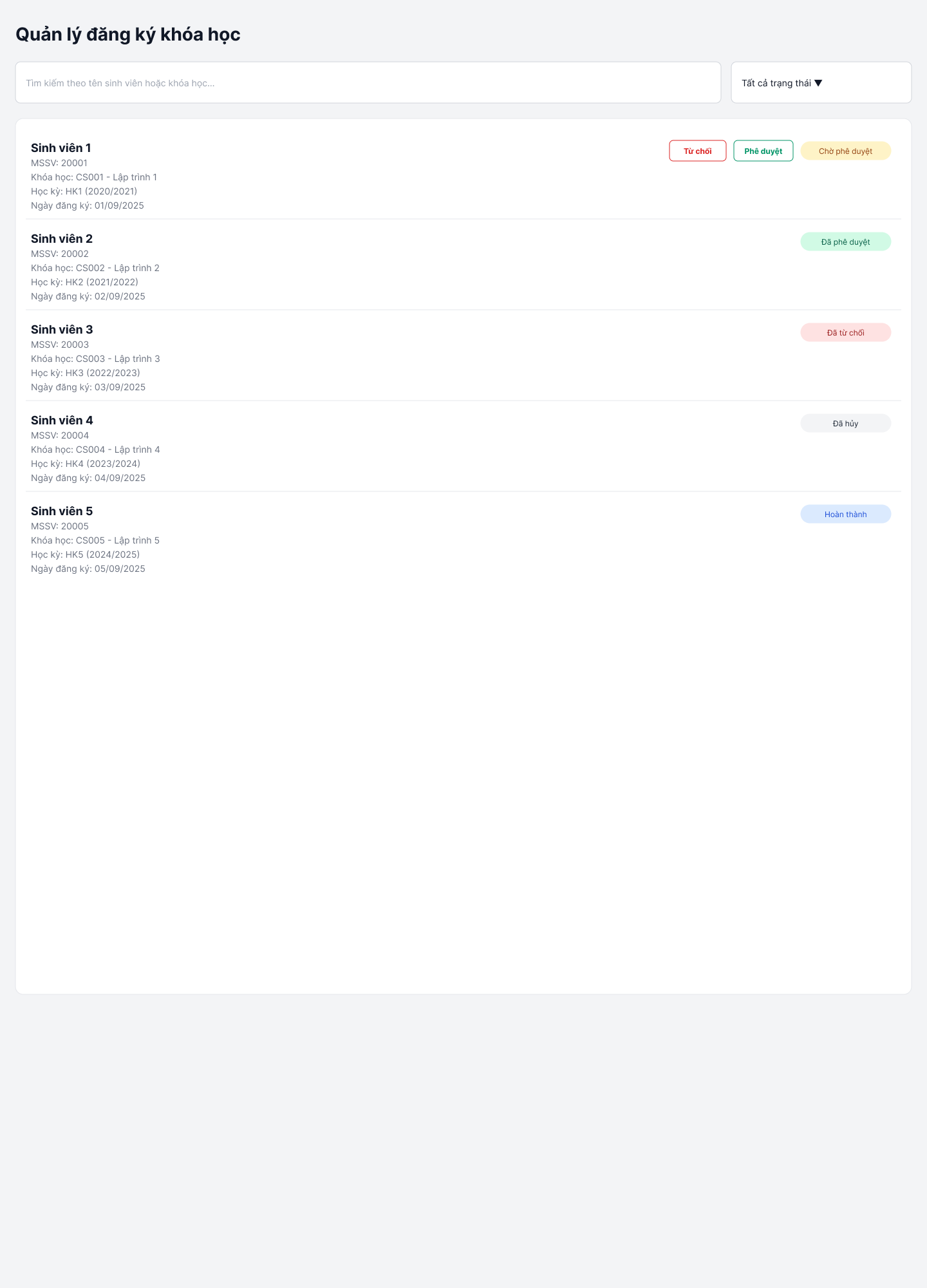
**2.2.8 Giao diện hồ sơ đăng kí**

****

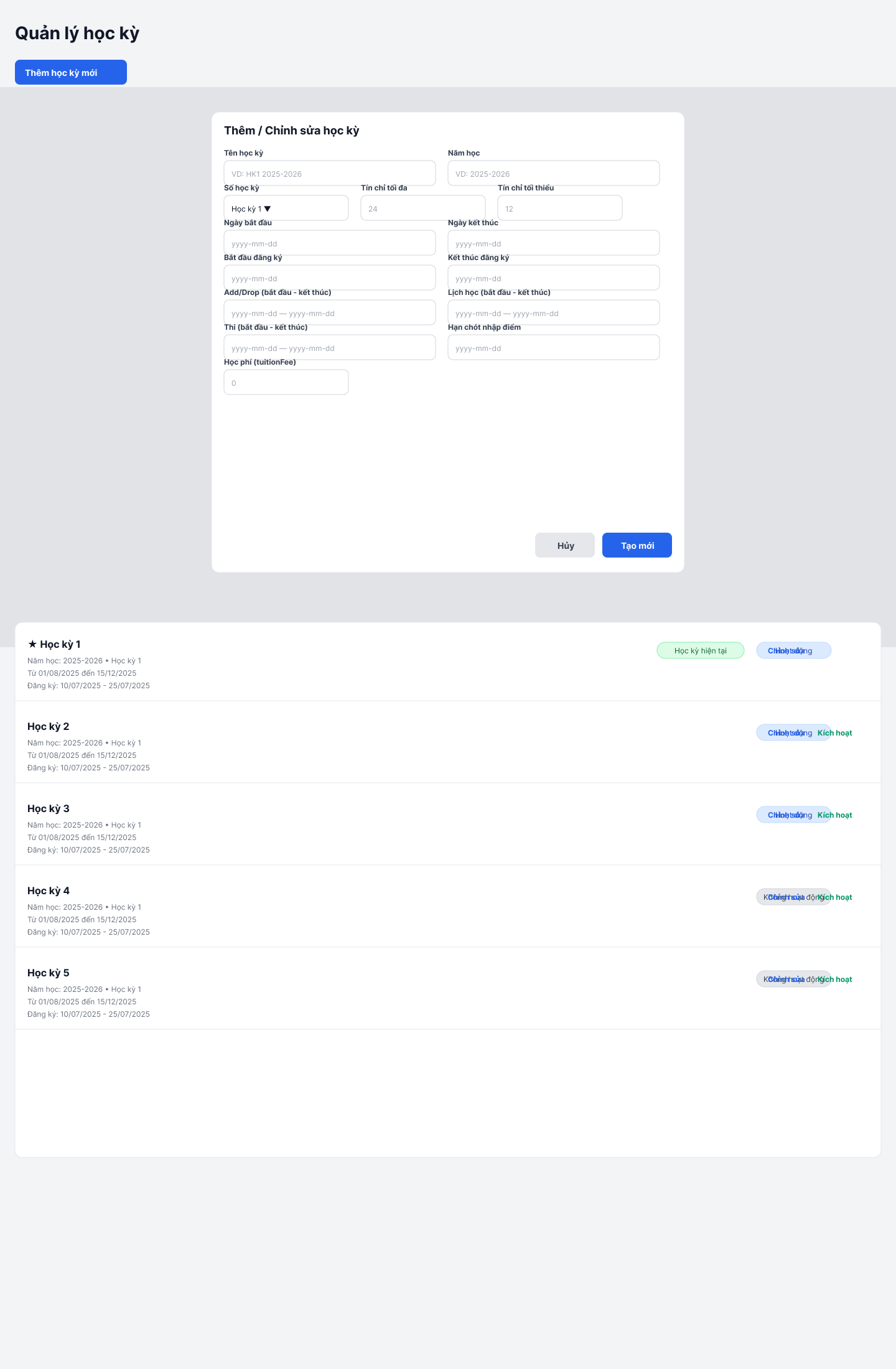
**2.2.9 Giao diện đăng kí**

****

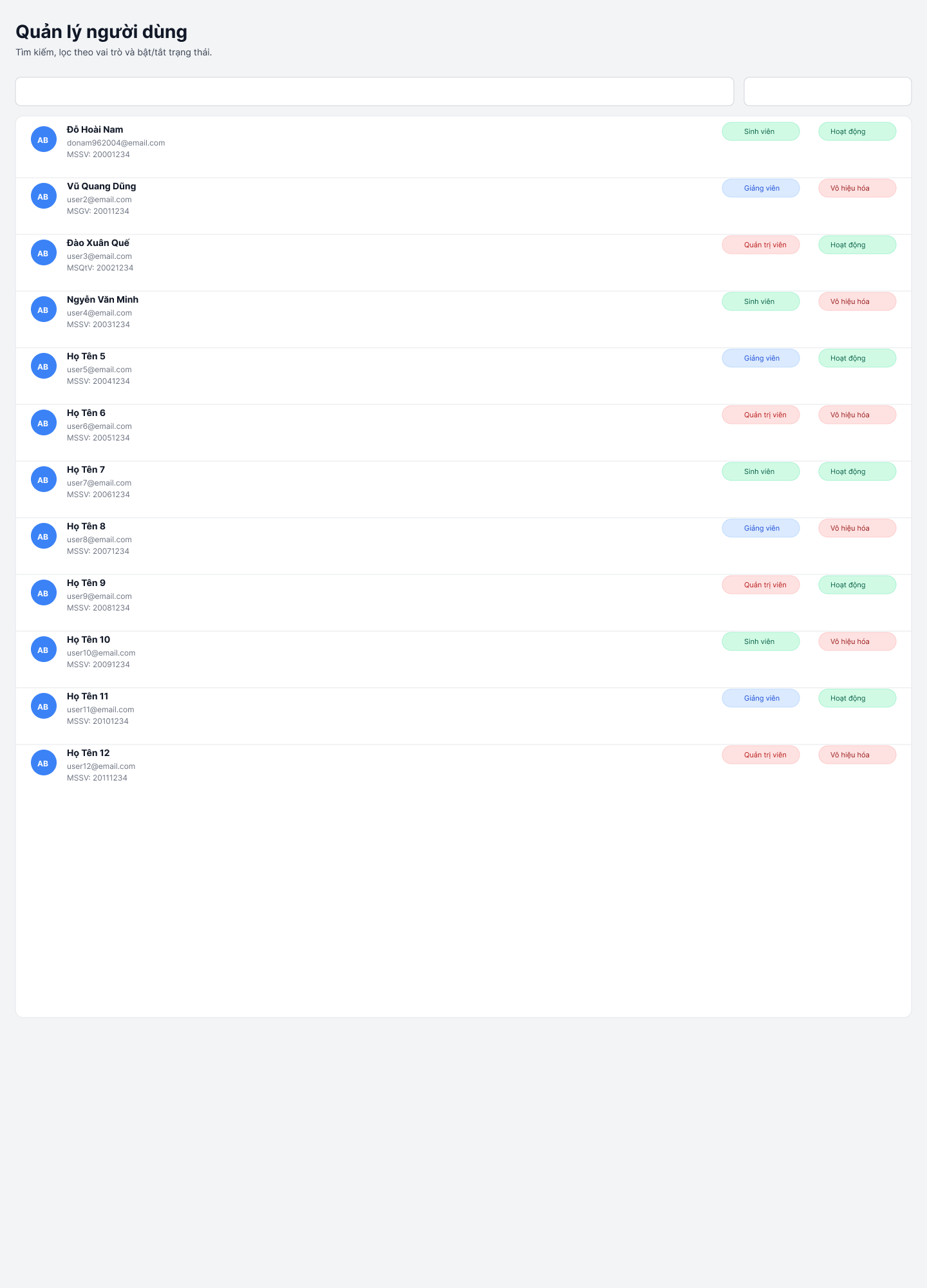
**2.2.10 Giao diện quản lí đăng kí**

****

**2.2.11 Giao diện quản lí kì học**

****

**2.2.12 Giao diện quản lí người dùng**

****

**3. Kết quả đạt được**

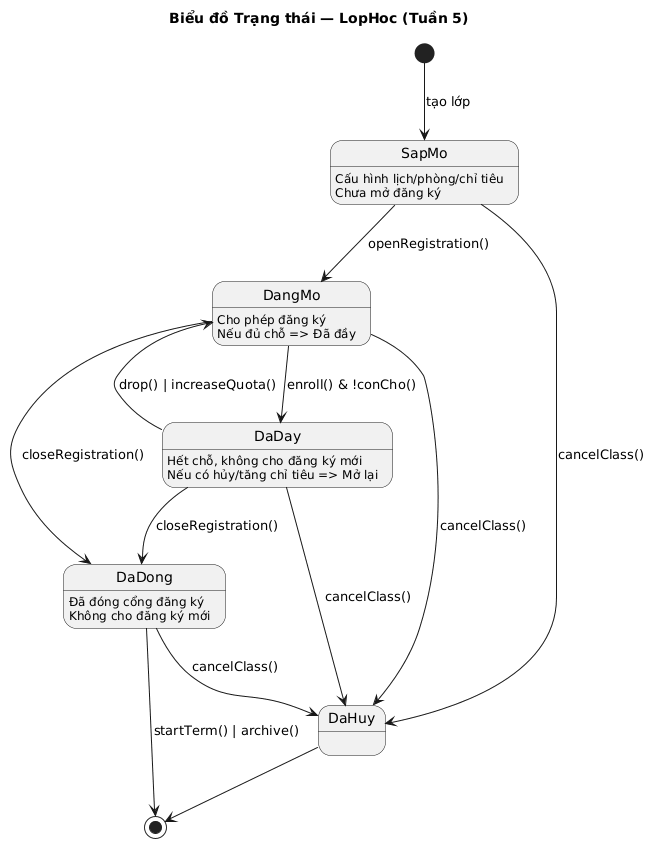
* Hoàn thành **Biểu đồ Trình tự cho Use Case Đăng ký môn học**, thể hiện rõ các đối tượng, thông điệp và điều kiện kiểm tra.
* Hoàn thành **bản phác thảo giao diện đăng ký môn học**, trực quan và dễ hiểu cho người dùng.
* Xác định được các tình huống lỗi và xử lý ngoại lệ trong cả biểu đồ trình tự lẫn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

**4. Đánh giá và khó khă**

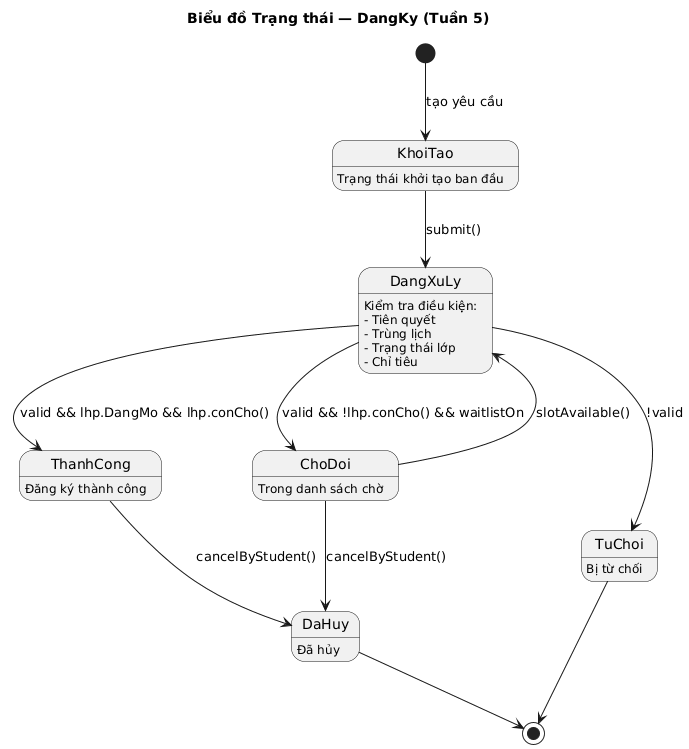
* **Ưu điểm:**
  + Nhóm đã mô tả được chi tiết cách các đối tượng tương tác.
  + Giao diện được thiết kế trực quan, dễ thao tác.
  + Mối liên hệ giữa **thiết kế UML** và **giao diện người dùng** được thể hiện rõ ràng.
* **Khó khăn:**
  + Việc vẽ Biểu đồ Trình tự có nhiều đối tượng, cần thống nhất cách 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**).

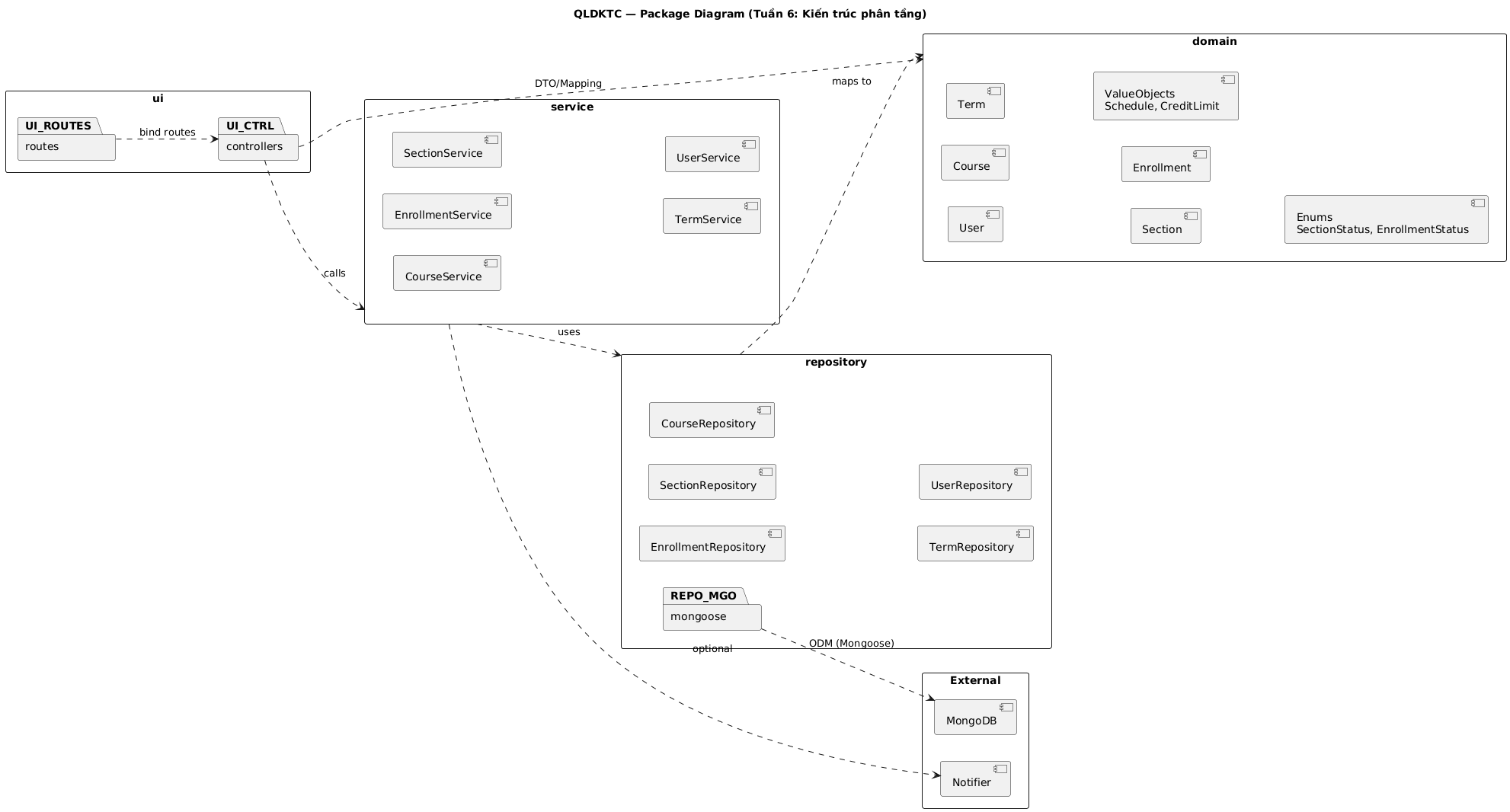
**Báo cáo tuần 6. Thiết kế kiến trúc hệ thống**

**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).