KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ II, NĂM HỌC 2024-2025**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ HỌC TẬP (LMS) BẰNG NEXT.JS**

*Giảng viên hướng dẫn:*

TS. Nguyễn Bảo Ân

*Sinh viên thực hiện:*

Họ tên: Phạm Hữu Lộc

MSSV: 110121055

Lớp: DA21TTB

***Trà Vinh, tháng 12 năm 2024***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**THỰC TẬP ĐỒ ÁN CHUYÊN NGÀNH**

**HỌC KỲ II, NĂM HỌC 2024-2025**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ HỌC TẬP (LMS) BẰNG NEXT.JS**

*Giảng viên hướng dẫn:*

TS. Nguyễn Bảo Ân

*Sinh viên thực hiện:*

Họ tên: Phạm Hữu Lộc

MSSV: 110121055

Lớp: DA21TTB

***Trà Vinh, tháng 12 năm 2024***

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Trước hết, em xin được gửi lời cảm ơn đến toàn thể quý thầy cô, giảng viên Trường Đại học Trà Vinh, đặc biệt là các thầy cô ở Khoa Kỹ thuật và Công nghệ, Bộ môn Công nghệ thông tin, đã tạo điều kiện để em hoàn thành đề tài này

Để hoàn thành đề tài này, em xin chân thành cảm ơn các thầy cô giảng viên đã tận tình hướng dẫn, giảng dạy trong suốt quá trình học tập, nghiên cứu ở Trường Đại học Trà Vinh. Đặc biệt xin gửi lời cảm ơn chân thành tới giảng viên hướng dẫn thầy Nguyễn Bảo Ân đã tận tình, chu đáo hướng dẫn em thực hiện đề tài này.

Mặc dù đã có nhiều cố gắng để thực hiện đề tài một cách hoàn chỉnh nhất. Song do thời gian có hạn và còn hạn chế về kiến thức cho nên trong đồ án không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự đóng góp ý kiến của thầy cô và bạn bè để em có thể hoàn thiện đồ án này tốt hơn. Một lần nữa cho em xin gửi lời cảm ơn đến quý thầy cô, chúc quý thầy cô thật nhiều sức khỏe, thành công và hạnh phúc.

Em xin chân thành cảm ơn!

|  |  |
| --- | --- |
|  | **Sinh viên thực hiện**  **Phạm Hữu Lộc** |

**MỤC LỤC**

[CHƯƠNG 1. TỔNG QUAN 1](#_Toc187270171)

[CHƯƠNG 2. NGHIÊN CỨU LÝ THUYẾT 2](#_Toc187270172)

[2.1 Next.js Framework 2](#_Toc187270175)

[2.1.1 Giới thiệu tổng quan về Next.js 2](#_Toc187270176)

[2.1.2 Kiến trúc và nguyên lý hoạt động của Next.js 2](#_Toc187270177)

[2.1.3 Các tính năng chính của Next.js 2](#_Toc187270178)

[2.1.4 Server Components và Client Components 4](#_Toc187270179)

[2.2 NodeJS 4](#_Toc187270180)

[2.2.1 Tổng quan về NodeJS 4](#_Toc187270181)

[2.2.2 Event-driven và non-blocking I/O 5](#_Toc187270182)

[2.2.3 ExpressJS Framework 6](#_Toc187270183)

[2.2.4 Vai trò của NodeJS trong hệ thống backend 6](#_Toc187270184)

[2.3 Prisma ORM 7](#_Toc187270185)

[2.3.1 Giới thiệu về Prisma 7](#_Toc187270186)

[2.3.2 Kiến trúc và cách hoạt động 7](#_Toc187270187)

[2.3.3 Schema Definition Language 8](#_Toc187270188)

[2.3.4 Prisma Client 8](#_Toc187270189)

[CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU 10](#_Toc187270190)

[3.1 Mô tả bài toán 10](#_Toc187270192)

[3.2 Kiến trúc hệ thống 11](#_Toc187270193)

[3.2.1 Sơ đồ usecase 11](#_Toc187270194)

[3.2.2 Mô hình dữ liệu 13](#_Toc187270195)

[3.2.3 Sơ đồ triển khai 17](#_Toc187270196)

[3.3 Thiết kế giao diện 17](#_Toc187270197)

[3.3.1 Màn hình trang chủ 17](#_Toc187270198)

[3.3.2 Màn hình khoá học 18](#_Toc187270199)

[3.3.1 Màn hình đăng nhập/ đăng xuất 19](#_Toc187270200)

[3.3.2 Triển khai 19](#_Toc187270201)

[CHƯƠNG 4. KẾT QUẢ NGHIÊN CỨU 22](#_Toc187270202)

[4.1 Thử nghiệm các API với Postman 22](#_Toc187270204)

[4.2 Giao diện chức năng người dùng 22](#_Toc187270205)

[4.3 Giao diện chức năng người quản trị 23](#_Toc187270206)

[4.3.1 Giao diện quản lý người dùng 23](#_Toc187270207)

[4.3.2 Giao diện quản lý chủ đề 25](#_Toc187270208)

[CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 27](#_Toc187270209)

[6.1 Kết quả đạt được 27](#_Toc187270212)

[6.2 Hướng phát triển 27](#_Toc187270213)

[CHƯƠNG 6. DANH MỤC TÀI LIỆU THAM KHẢO 28](#_Toc187270214)

[PHỤ LỤC 29](#_Toc187270215)

**DANH MỤC HÌNH ẢNH – BẢNG BIỂU**

[Hình 2.1 File-system Routing. 3](#_Toc187270274)

[Hình 2.2. Event-driven. 5](#_Toc187270275)

[Hình 2.3. Non-blocking I/O. 6](#_Toc187270276)

[Hình 3.1. Sơ đồ usecase. 11](#_Toc187270277)

[Hình 3.2. Mô hình dữ liệu. 13](#_Toc187270278)

[Hình 3.3. Màn hình trang chủ. 17](#_Toc187270279)

[Hình 3.4. Màn hình khoá học. 18](#_Toc187270280)

[Hình 3.5. Màn hình đăng nhập/ đăng xuất. 19](#_Toc187270281)

[Hình 4.1. Giao diện đăng nhập. 22](#_Toc187270282)

[Hình 4.2. Giao diện quản lý dự án (1). 22](#_Toc187270283)

[Hình 4.3. Giảo diện quản lý dự án (2). 23](#_Toc187270284)

[Hình 4.4. Giao diện quản lý người dùng. 23](#_Toc187270285)

[Hình 4.5. Giao diện chức năng tạo người dùng mới. 24](#_Toc187270286)

[Hình 4.6. Giao diện chức năng chỉnh sửa thông tin người dùng. 24](#_Toc187270287)

[Hình 4.7. Giao diện quản lý chủ đề. 25](#_Toc187270288)

[Hình 4.8. Giao diện chức năng thêm chủ đề mới. 25](#_Toc187270289)

[Hình 4.9. Giao diện chức năng chỉnh sửa thông tin chủ đề. 26](#_Toc187270290)

[Bảng 3.1. Bảng mô tả các Actor. 12](#_Toc187270254)

[Bảng 3.2. Bảng mô tả các usecase. 12](#_Toc187270255)

[Bảng 3.3. Mô tả bảng User. 14](#_Toc187270256)

[Bảng 3.4. Mô tả bảng Project. 14](#_Toc187270257)

[Bảng 3.5. Mô tả bảng Course. 15](#_Toc187270258)

[Bảng 3.6. Mô tả bảng Quiz. 15](#_Toc187270259)

[Bảng 3.7. Mô tả bảng Category. 16](#_Toc187270260)

**DANH MỤC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **Từ viết tắt** | **Ý nghĩa** |
| CRA | Create React App |
| CSR | Client Site Generation |
| LMS | Learning Management System |
| ORM | Object-Relational Mapping |
| SDL | Schema Definition Language |
| SEO | Search Engine Optimization |
| SSG | Static Site Generation |
| SSR | Server-Side Rendering |

**TÓM TẮT ĐỒ ÁN CHUYÊN NGÀNH**

Đề tài tập trung vào nghiên cứu Next.js framework để xây dựng một Hệ thống Quản lý Học tập trực tuyến hay Learning Management System (LMS). Hệ thống này được phát triển dựa trên nền tảng web, tập trung vào việc quản lý khóa học, tích hợp video học liệu và theo dõi tiến trình học tập.

Các công nghệ được sử dụng trong đồ án này:

* Cơ sở lý thuyết: Next.js framework, Prisma, NodeJS, ExpressJS.
* Front-end: Next.js framework
* Back-end: NodeJS, thư viện ExpressJS.
* Cơ sở dữ liệu: MySQL.

Phương pháp tiếp cận của đề tài được chia thành hai hướng chính:

* **Nghiên cứu lý thuyết:** Tìm hiểu sâu về các công nghệ được sử dụng và phân tích các hệ thống LMS hiện có.
* **Nghiên cứu thực nghiệm:** Triển khai và phát triển hệ thống thực tế, bao gồm xây dựng giao diện người dùng và phát triển các API cần thiết.

Kết quả mong đợi của đề tài là một hệ thống LMS hoàn chỉnh cho phép:

* Giảng viên có thể dễ dàng tạo và quản lý khóa học, đăng tải video bài giảng.
* Học viên có thể tương tác với nội dung học tập và theo dõi tiến trình của mình.
* Hệ thống có khả năng theo dõi và đánh giá hiệu quả học tập thông qua các báo cáo chi tiết.

Ngoài ra, đề tài cũng hướng đến việc tích lũy kinh nghiệm thực tế trong việc phát triển ứng dụng web hiện đại, đặc biệt là trong lĩnh vực E-learning, tạo nền tảng cho việc phát triển các dự án tương tự trong tương lai.

**MỞ ĐẦU**

**Lý do chọn đề tài:** Trong bối cảnh giáo dục hiện đại, việc số hóa và quản lý học tập trực tuyến đang trở thành một xu hướng tất yếu. Hệ thống quản lý học tập (LMS) đóng vai trò quan trọng trong việc tổ chức và triển khai các khóa học trực tuyến một cách hiệu quả. Tuy nhiên, việc xây dựng một hệ thống LMS đòi hỏi phải có một nền tảng công nghệ mạnh mẽ, linh hoạt và hiệu suất cao.

Next.js, một framework dựa trên React, đang nổi lên như một giải pháp tiên tiến cho việc phát triển ứng dụng web hiện đại. Với các tính năng như Server-Side Rendering (SSR), Static Site Generation (SSG), và API Routes tích hợp, Next.js cung cấp nền tảng lý tưởng để xây dựng các ứng dụng web phức tạp như hệ thống LMS. Đặc biệt, khả năng tối ưu hóa hiệu suất và Search Engine Optimization (SEO) của Next.js giúp cải thiện trải nghiệm người dùng và khả năng tiếp cận của hệ thống.

**Mục đích nghiên cứu:** Tìm hiểu sâu về kiến trúc và các tính năng của Next.js trong việc xây dựng ứng dụng web full-stack. Áp dụng Next.js và NodeJS để xây dựng một hệ thống LMS hoàn chỉnh.

Đối tượng nghiên cứu:

* Next.js framework.
* NodeJS và thư viện ExpressJS.
* Cơ sở dữ liệu MySQL, bộ công cụ Object-Relational Mapping (ORM) Prisma.

Phạm vi phiên cứu:

* Tập trung nhiên cứu vằn nắm vững các cơ sở lý thuyết của Next.js framework.
* Tập trung vào việc phát triển các chức năng cốt lõi của hệ thống LMS sử dụng Next.js

# TỔNG QUAN

Trong lĩnh vực phát triển web hiện đại, Next.js đã nổi lên như một framework mạnh mẽ dựa trên React, cung cấp giải pháp toàn diện cho việc xây dựng ứng dụng web full-stack. Trước Next.js, các developer thường phải sử dụng Create React App (CRA) hoặc các giải pháp tùy chỉnh khác để xây dựng ứng dụng React. Tuy nhiên, những giải pháp này thường thiếu các tính năng quan trọng như SSR, SSG, và routing tích hợp.

Next.js giải quyết những hạn chế này bằng cách cung cấp một framework toàn diện với nhiều tính năng tích hợp sẵn. Framework này cho phép các lập trình viên xây dựng ứng dụng web với hiệu suất cao và trải nghiệm người dùng tốt hơn thông qua các tính năng như SSR và SSG. Điều này đặc biệt quan trọng đối với các hệ thống LMS, nơi việc tối ưu hóa thời gian tải trang và SEO là yếu tố then chốt.

Trong việc phát triển hệ thống LMS, Next.js mang đến nhiều lợi thế quan trọng. Framework này cho phép xây dựng giao diện người dùng động và tương tác cao, đồng thời duy trì hiệu suất tốt thông qua các tối ưu hóa tích hợp. Khả năng tích hợp với các API và cơ sở dữ liệu của Next.js cũng giúp đơn giản hóa việc xây dựng backend cho hệ thống LMS.

# NGHIÊN CỨU LÝ THUYẾT



## Next.js Framework

### Giới thiệu tổng quan về Next.js

Next.js là một framework phát triển web mã nguồn mở được phát triển dựa trên nền tảng React. Framework này được thiết kế để mang lại trải nghiệm phát triển tốt nhất với tất cả các tính năng cần thiết được tích hợp sẵn như rendering phía máy chủ và tạo trang tĩnh cho phép chúng ta xây dựng các trang web tĩnh có tốc độ siêu nhanh. Next.js được ra đời vào năm 2016, thuộc sở hữu của Vercel. Next.js bắt đầu trở nên phổ biến vào năm 2018 và tiếp tục tăng trưởng mạnh mẽ trong cộng đồng phát triển web.

Tính đến tháng 11/2024, Next.js đã thu hút hơn 120.000 sao trên Github [1], cho thấy sự phổ biến và tin tưởng của cộng đồng developer. Framework này cung cấp một hệ sinh thái phong phú với nhiều tính năng tích hợp như file-system routing, API routes, và hỗ trợ TypeScript mặc định. Next.js cũng có khả năng tùy biến cao thông qua hệ thống cấu hình linh hoạt, cho phép developers điều chỉnh framework để phù hợp với yêu cầu cụ thể của dự án.

### Kiến trúc và nguyên lý hoạt động của Next.js

Next.js được xây dựng trên nền tảng của React và Node.js, sử dụng kiến trúc hybrid cho phép kết hợp linh hoạt giữa SSR và CSR. Framework này hoạt động dựa trên nguyên lý "convention over configuration", trong đó các quy ước về cấu trúc thư mục và đặt tên file được sử dụng để tự động tạo ra routing và các cấu hình khác. Kiến trúc này cho phép developers tập trung vào việc phát triển tính năng thay vì phải lo lắng về cấu hình cơ bản.

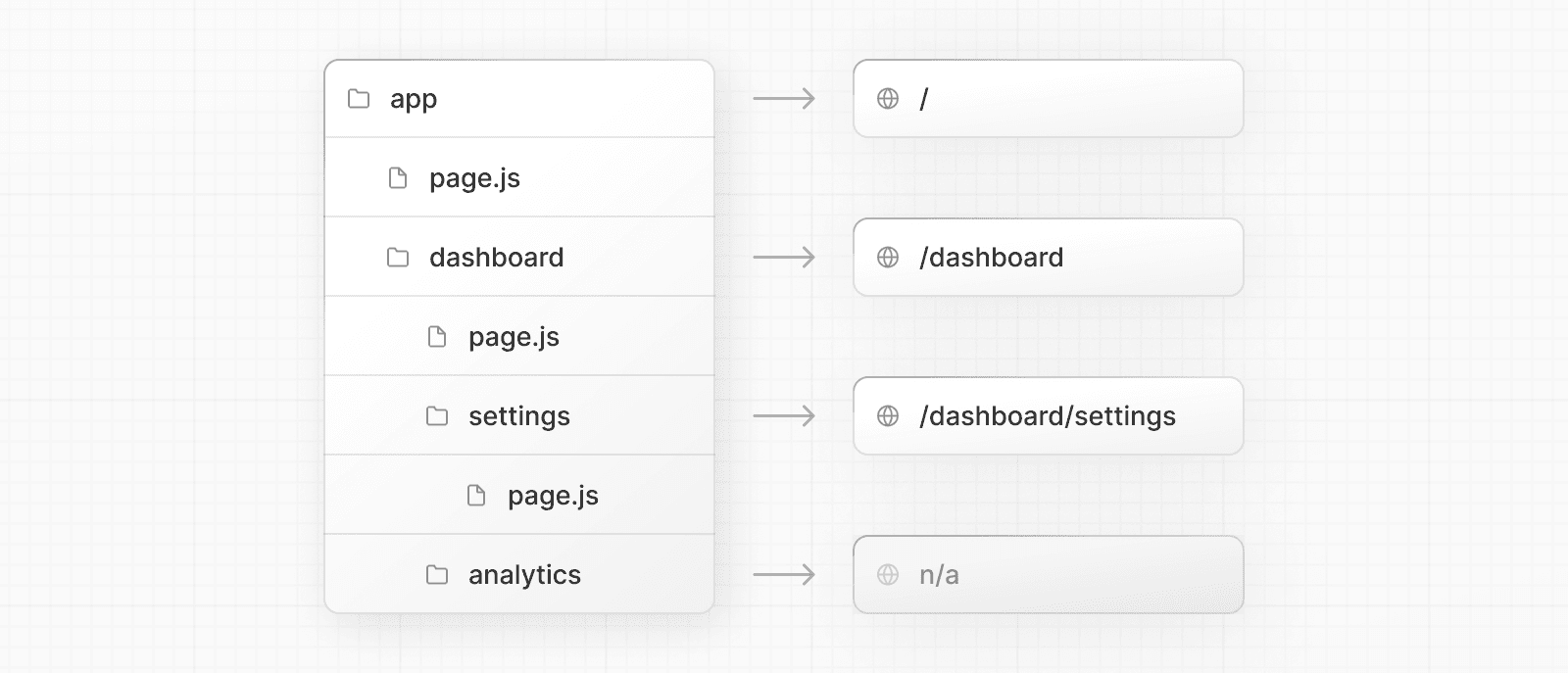
### Các tính năng chính của Next.js

**Server-Side Rendering (SSR):** Next.js cung cấp khả năng render các trang web trên server trước khi gửi đến client. Điều này giúp cải thiện thời gian tải trang đầu tiên và tối ưu hóa SEO. SSR đặc biệt hữu ích cho các trang web động có nội dung thường xuyên thay đổi.

**Static Site Generation (SSG):** Tính năng này cho phép tạo ra các trang HTML tĩnh tại thời điểm build. SSG giúp tăng hiệu suất và giảm tải cho server vì các trang đã được tạo sẵn và có thể được lưu cache.

**Client-Side Rendering (CSR):** Next.js vẫn hỗ trợ rendering phía client cho các tương tác động, cho phép ứng dụng hoạt động như một SPA sau lần tải đầu tiên.

**File-system Routing:** Next.js sử dụng hệ thống routing dựa trên cấu trúc thư mục, trong đó mỗi thư mục đại diện cho một router và ta có thể lòng các thư mục vào nhau. Ví dụ, nếu ta tạo một thư mục có tên là `dashboard` và thư mục `setting` được lồng bên trong. Thì Next.js sẽ tạo ra router `/dashboard` và `dashboard/settings`.



Hình 2.1 File-system Routing.

**Dynamic Routes:** Khi ta không biết chính xác tên của các segment và muốn tạo routes từ dữ liệu động, ta có thể sử dụng *Dynamic Routes* để tạo ra các router tại thời điểm request hoặc thời xây dựng. Ví dụ, một bài blog có route `app/blog/[slug]/page.js` trong đó `[slug]` là một *Dynamic Routes* cho các bài blog.

export default async function Page({

params,

}: {

params: Promise<{ slug: string }>

}) {

const slug = (await params).slug

return <div>My Post: {slug}</div>

}

**API Routes:** Framework cung cấp khả năng tạo API endpoints trực tiếp trong ứng dụng Next.js thông qua thư mục pages/api, cho phép xây dựng full-stack applications trong cùng một project.

### Server Components và Client Components

**Server Components:** Đây là một tính năng đột phá của React và Next.js, cho phép các thành phần của trang web được render hoàn toàn phía máy chủ. Các thành phần này không được gửi đến phía người dùng, giúp giảm kích thước bundle Javascript. Server Components đặc biệt tối ưu tốt cho các phần của ứng dụng không yêu cầu tương tác của người dùng như các trang tĩnh, hoặc các thành phần chứa logic xử lý phức tạp.

Ưu điểm của Server Components là:

* Tăng tốc độ tải ở lần đầu tiên.
* Truy cập trực tiếp vào tài nguyên máy chủ như cơ sở dữ liệu.
* Bảo mật tốt hơn do logic nhạy cảm được giữ ở phía server.

**Client Components:** Ngược lại, Client Components vẫn được render và tương tác hoàn toàn ở phía người dùng. Các thành phần này sử dụng các trạng thái (state), các sự kiện, và các hooks như useEffect, useState. Chúng được đánh dấu bằng chỉ thị "use client" ở đầu file, cho phép Next.js nhận biết và xử lý phù hợp.

Ưu điểm của Client Components:

* Hỗ trợ đầy đủ các tính năng tương tác của React.
* Sử dụng các hooks và quản lý trạng thái.
* Phù hợp cho các component yêu cầu tương tác động như form, nút bấm,…

## NodeJS

### Tổng quan về NodeJS

Node.js là một môi trường thực thi JavaScript mã nguồn mở, được phát triển bởi Ryan Dahl vào năm 2009. Mục tiêu ban đầu của Node.js là cho phép lập trình viên sử dụng JavaScript bên ngoài trình duyệt, nhằm xây dựng các ứng dụng web có khả năng mở rộng và hiệu suất cao. Node.js sử dụng V8 JavaScript engine của Google để biên dịch mã JavaScript thành mã máy, giúp tăng tốc độ thực thi [2] [3].

Đặc điểm và ưu điểm chính của Node.js bao gồm:

* **Mô hình I/O không đồng bộ:** Node.js cho phép xử lý nhiều yêu cầu đồng thời mà không bị tắc nghẽn.
* **Khả năng mở rộng:** Node.js dễ dàng mở rộng để xử lý hàng triệu kết nối đồng thời.
* **Cộng đồng lớn và thư viện phong phú:** Hệ sinh thái npm (Node Package Manager) cung cấp hàng triệu gói phần mềm, giúp lập trình viên dễ dàng tích hợp các tính năng mới vào ứng dụng.

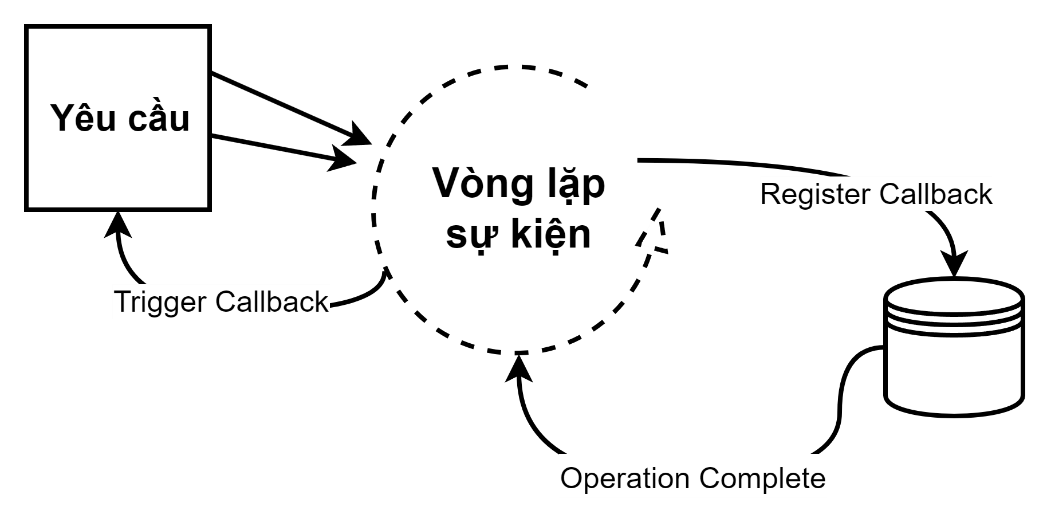
### Event-driven và non-blocking I/O

Mô hình event-driven trong Node.js cho phép ứng dụng phản hồi với các sự kiện mà không cần chờ đợi kết quả từ các tác vụ khác. Khi một sự kiện xảy ra, hàm callback tương ứng sẽ được gọi để xử lý sự kiện đó. Điều này giúp tối ưu hóa hiệu suất và giảm thiểu độ trễ trong quá trình xử lý yêu cầu.



Hình 2.2. Event-driven.

Cơ chế non-blocking I/O cho phép Node.js thực hiện các thao tác nhập/xuất mà không làm tắc nghẽn luồng chính của chương trình. Thay vì chờ đợi một thao tác hoàn thành trước khi tiếp tục, Node.js sẽ tiếp tục xử lý các yêu cầu khác trong khi chờ đợi kết quả từ tác vụ I/O.



Hình 2.3. Non-blocking I/O.

So với các mô hình truyền thống như multi-threading, nơi mỗi yêu cầu có thể tạo ra một luồng mới, mô hình event-driven của Node.js tiết kiệm tài nguyên hệ thống hơn, vì nó không cần tạo ra nhiều luồng cho mỗi yêu cầu.

### ExpressJS Framework

**ExpressJS** là một framework web nhẹ dành cho Node.js, được thiết kế để xây dựng các ứng dụng web và API một cách nhanh chóng và dễ dàng. Nó cung cấp một bộ công cụ mạnh mẽ để xử lý các yêu cầu HTTP và quản lý routing.

* Middleware trong **ExpressJS** là các hàm được gọi trong chuỗi xử lý yêu cầu trước khi đến route handler cuối cùng. Middleware có thể được sử dụng để thực hiện các tác vụ như xác thực người dùng, ghi log hoặc xử lý lỗi [4].
* Hệ thống Routing trong **ExpressJS** cho phép lập trình viên định nghĩa các endpoint cho ứng dụng của mình một cách rõ ràng và linh hoạt.
* Xây dựng **RESTful API** với **ExpressJS** rất đơn giản nhờ vào khả năng xử lý HTTP request và response một cách hiệu quả. Bằng cách sử dụng các phương thức HTTP như GET, POST, PUT, DELETE, lập trình viên có thể dễ dàng tạo ra các API phục vụ cho hệ thống LMS.

### Vai trò của NodeJS trong hệ thống backend

Node.js đóng vai trò quan trọng trong kiến trúc backend của hệ thống LMS nhờ vào khả năng xử lý đồng thời nhiều yêu cầu từ người dùng mà không làm giảm hiệu suất hệ thống.

* Vị trí của NodeJS trong kiến trúc hệ thống thường nằm ở lớp trung gian giữa frontend và cơ sở dữ liệu, nơi nó nhận và xử lý các yêu cầu từ client.
* Tương tác với cơ sở dữ liệu: Node.js có thể dễ dàng kết nối với nhiều loại cơ sở dữ liệu như MongoDB, MySQL hay PostgreSQL thông qua các thư viện như Mongoose hoặc Sequelize.
* Xử lý yêu cầu và phản hồi: Với khả năng non-blocking I/O, Node.js có thể xử lý hàng triệu yêu cầu từ người dùng cùng lúc mà không bị tắc nghẽn.
* Tích hợp với Next.js: Node.js thường được sử dụng như backend cho các ứng dụng Next.js, giúp tối ưu hóa việc truyền tải dữ liệu giữa server và client cũng như hỗ trợ render phía server (SSR).

## Prisma ORM

### Giới thiệu về Prisma

Prisma là một thư viện ORM hiện đại dành cho Node.js và TypeScript, giúp đơn giản hóa việc tương tác với cơ sở dữ liệu. Bằng cách tạo ra một lớp trừu tượng giữa ứng dụng và cơ sở dữ liệu, Prisma cho phép lập trình viên thao tác với dữ liệu mà không cần viết các câu truy vấn SQL phức tạp. Nó hỗ trợ nhiều loại cơ sở dữ liệu như PostgreSQL, MySQL, SQLite, SQL Server và MongoDB, mang lại sự linh hoạt cho các ứng dụng web hiện đại [2].

### Kiến trúc và cách hoạt động

Prisma hoạt động dựa trên ba thành phần chính: Prisma Client, Prisma Migrate, và Prisma Studio.

* **Prisma Client** là một trình tạo truy vấn an toàn và tự động được sinh ra từ định nghĩa mô hình trong tệp `schema.prisma`. Nó cho phép lập trình viên thực hiện các thao tác CRUD (Create, Read, Update, Delete) với dữ liệu một cách dễ dàng và an toàn về kiểu dữ liệu.
* **Prisma Migrate** là công cụ quản lý di chuyển cơ sở dữ liệu, giúp theo dõi và áp dụng các thay đổi trong cấu trúc cơ sở dữ liệu theo thời gian.
* **Prisma Studio** cung cấp giao diện người dùng trực quan để xem và chỉnh sửa dữ liệu trong cơ sở dữ liệu mà không cần viết mã

### Schema Definition Language

Schema Definition Language (SDL) của Prisma cho phép lập trình viên định nghĩa cấu trúc của cơ sở dữ liệu một cách rõ ràng và dễ hiểu trong tệp `schema.prisma`. Ví dụ về việc định nghĩa mô hình người dùng có thể như sau:

model User {

id Int @id @default(autoincrement())

name String

email String @unique

}

Trong ví dụ trên, mô hình User được định nghĩa với ba trường: id, name, và email, trong đó id là khóa chính tự động tăng và email là trường duy nhất

### Prisma Client

Prisma Client được sinh ra từ SDL và cung cấp một API mạnh mẽ để tương tác với cơ sở dữ liệu. Ví dụ về cách sử dụng Prisma Client để tạo một người dùng mới có thể được thực hiện như sau:

import { PrismaClient } from '@prisma/client';

const prisma = new PrismaClient();

async function main() {

const newUser = await prisma.user.create({

data: {

name: 'John Doe',

email: 'john.doe@example.com',

},

});

console.log(newUser);

}

main()

.catch(e => console.error(e))

.finally(async () => {

await prisma.$disconnect();

});

Trong đoạn mã trên, chúng ta khởi tạo một đối tượng PrismaClient, sau đó sử dụng phương thức create để thêm một người dùng mới vào cơ sở dữ liệu

Prisma ORM không chỉ là một công cụ mạnh mẽ giúp lập trình viên tương tác với cơ sở dữ liệu mà còn mang lại sự an toàn về kiểu dữ liệu thông qua TypeScript, cùng với khả năng quản lý di chuyển cơ sở dữ liệu hiệu quả.

# HIỆN THỰC HÓA NGHIÊN CỨU



## Mô tả bài toán

Để đáp ứng nhu cầu ngày càng tăng về học tập trực tuyến và quản lý học tập hiệu quả dự án này tập trung vào việc phát triển một nền tảng học tập trực tuyến, nơi giảng viên có thể dễ dàng tạo lập và quản lý nội dung giảng dạy, đồng thời học viên có thể tương tác hiệu quả với các tài liệu học tập.

Yêu cầu đặt ra của đề tài là: Xây dựng một hệ thống LMS sử dụng Next.js framework, trong đó:

**Tổ chức nội dung**

* Hệ thống cho phép tổ chức nội dung học tập theo dạng project, mỗi project có thể chứa nhiều khóa học (courses), bài kiểm tra (quizzes).
* Mỗi đơn vị nội dung (course, quiz) đều có các thuộc tính cơ bản như hình ảnh đại diện, tiêu đề, mô tả, thời gian tạo/chỉnh sửa/xuất bản, trạng thái, tác giả và danh mục.

**Quản lý khoá học**

* Người dùng có thể tìm kiếm khóa học theo tên, chủ đề, người tạo và lọc theo các chủ đề
* Giảng viên có thể tạo mới, chỉnh sửa và quản lý khóa học với đầy đủ thông tin như thumbnail, tên khóa học, chủ đề, mô tả và nội dung
* Học viên có thể tham gia hoặc hủy tham gia khóa học, xem video bài giảng và theo dõi tiến độ học tập

**Hệ thống kiểm tra trắc nghiệm**

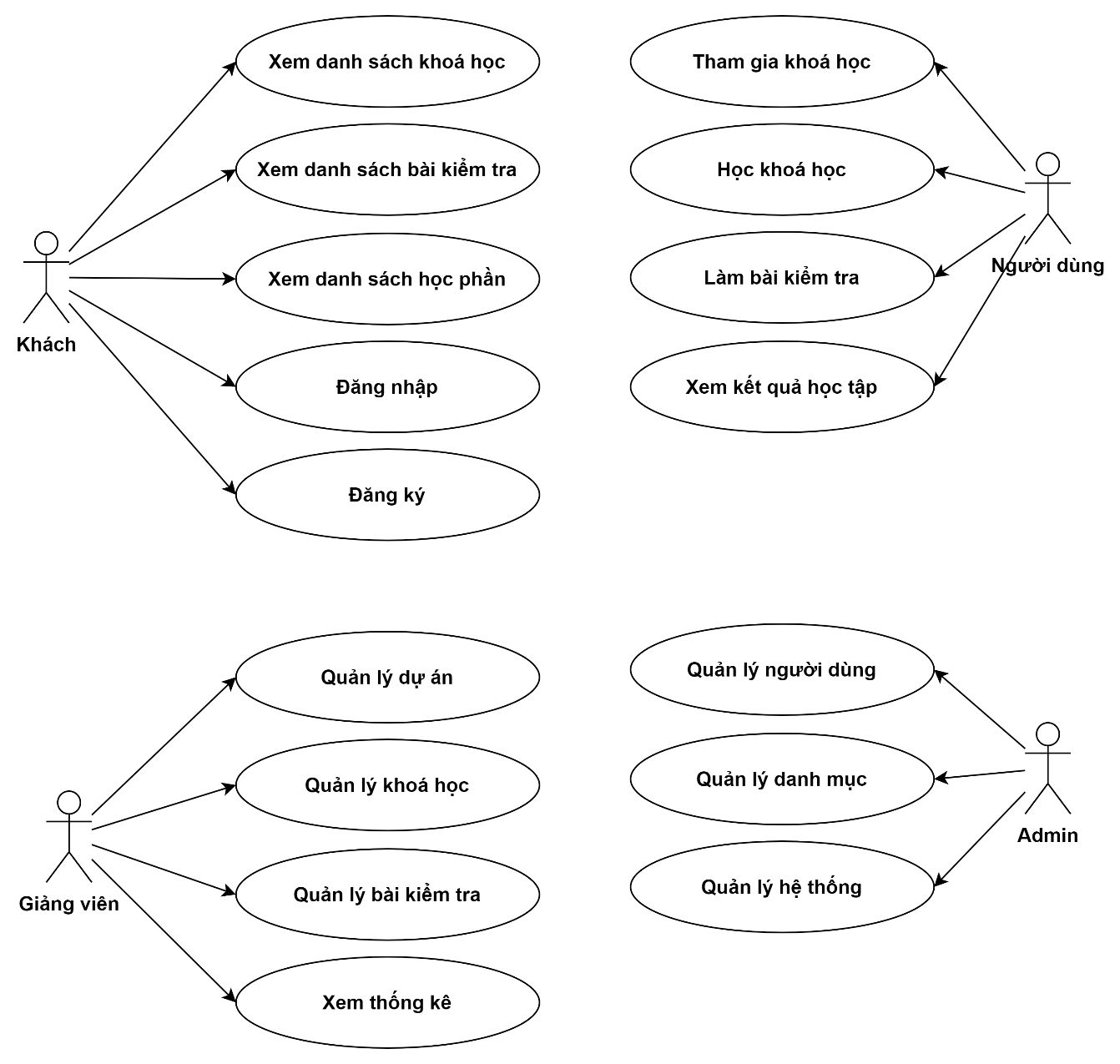
* Cho phép giảng viên tạo các bài kiểm tra với nhiều định dạng câu hỏi (trắc nghiệm, câu trả lời ngắn, câu trả lời dài)
* Hỗ trợ tùy chỉnh thời gian làm bài, thời gian bắt đầu/kết thúc, và điểm số cho từng câu hỏi
* Có khả năng thêm hình ảnh minh họa cho câu hỏi và tùy chọn xáo trộn thứ tự câu trả lời

**Về tổ chức dự án**

* Các khóa học và bài kiểm tra có thể được tổ chức độc lập hoặc thuộc về một dự án cụ thể
* Hệ thống quản lý phân quyền để kiểm soát việc truy cập nội dung dựa trên vai trò người dùng (Guest, Teacher, Student)
* Theo dõi và lưu trữ thông tin về thời gian tạo, chỉnh sửa, xuất bản và xóa của mọi nội dung

## Kiến trúc hệ thống

### Sơ đồ usecase



Hình 3.1. Sơ đồ usecase.

**\* Mô tả các Actor:**

Bảng 3.1. Bảng mô tả các Actor.

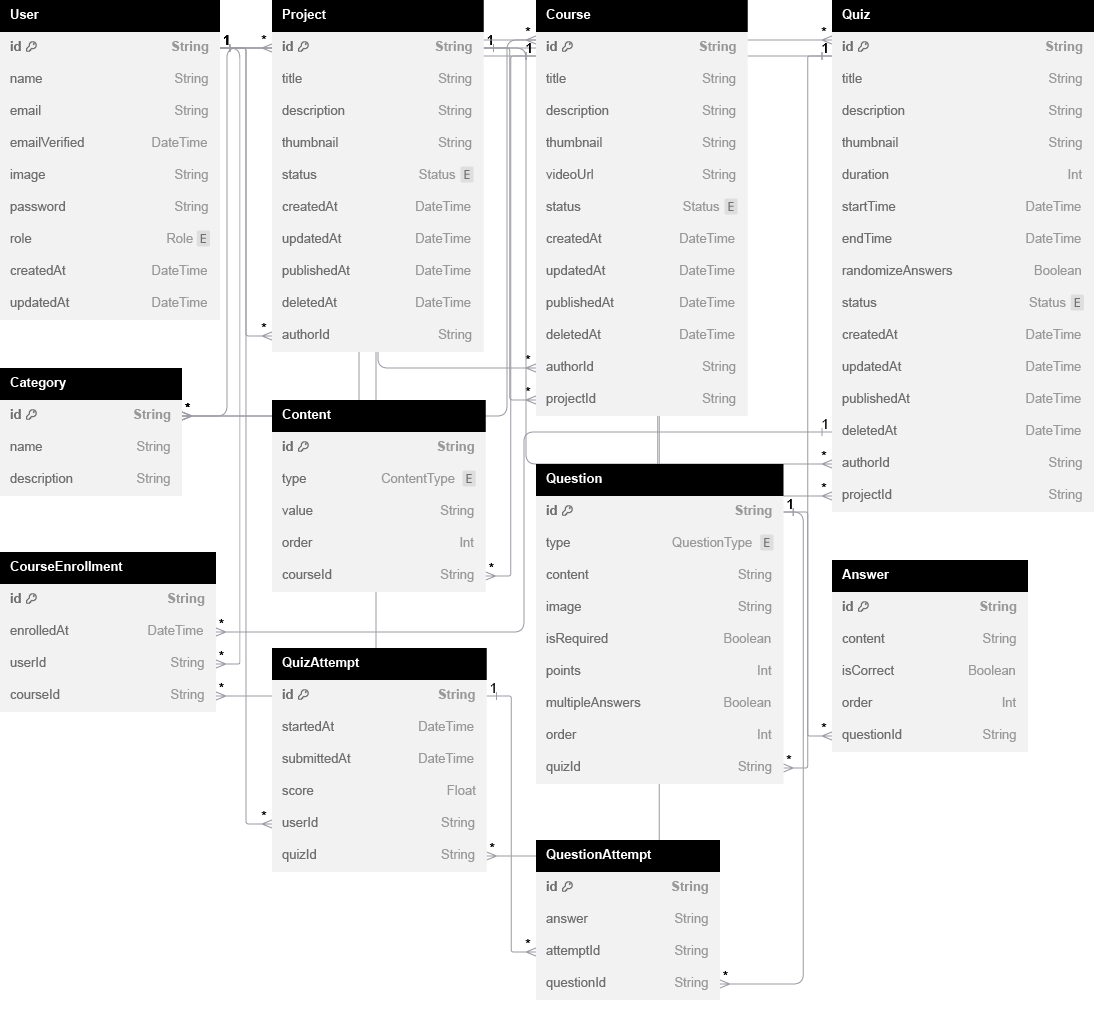
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên Actor** | **Ý nghĩa** |
| 1 | Khách | Người dùng thông thường (chưa đăng nhập) |
| 2 | Người dùng | Người dùng đã đăng nhập |
| 3 | Giảng viên | Người thực hiện giảng dạy trên hệ thống |
| 4 | Admin | Người quản trị hệ thống |

**\* Mô tả các usecase**

Bảng 3.2. Bảng mô tả các usecase.

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên usecase** | **Mô tả** |
| 1 | Xem danh sách khoá học | Người dùng có thể xem danh sách các khoá học, xem thông tin của khoá học. |
| 2 | Xem danh sách bài kiểm tra | Người dùng có thể xem danh sách bài kiểm tra. |
| 3 | Xem danh sách học phần | Xem danh sách các học phần, thông tin của từng học phần. |
| 4 | Đăng nhập/ Đăng ký | Người dùng có thể đăng nhập vào tài khoản của mình hoặc đăng ký tài khoảng mới. |
| 5 | Tham gia khoá học | Học viên có thể tham gia vào khoá học nào đó. |
| 6 | Học khoá học | Học viên có thể học khoá học mà mình đã giam gia. |
| 7 | Làm bài kiểm tra | Học viên có thể làm bài kiểm tra. |
| 8 | Xem kết quả kiểm tra | Học viên có thể xem kết quả kiểm tra sau khi kiểm tra xong. |
| 9 | Quản lý dự án | Giảng viên có thể tạo dự án và quản lý các thông tin của dự án mình tạo. |
| 10 | Quản lý khoá học | Giảng viên có thể tạo khoá học, quản lý các thông tin của khoá học. |
| 11 | Quản lý bài kiểm tra | Giảng viên có thể tạo bài kiểm tra, quản lý các thông tin của bài kiểm tra. |
| 12 | Xem thống kê | Giảng viên có thể xem được thống kê về tổng số học viên tham gia khoá học, thống kế về điểm số,… |
| 13 | Quản lý người dùng | Người quản trị có thể quản lý các thông tin về tài khoản người dùng. |
| 14 | Quản lý danh mục | Người quản trị có thể quản lý các thông tin về các danh mục. |

### Mô hình dữ liệu



Hình 3.2. Mô hình dữ liệu.

**Một số bảng chính trong dự án bao gồm:**

\* **Bảng User**

* Tên thực thể: User
* Mô tả: Lưu trữ thông tin người dùng

Bảng 3.3. Mô tả bảng User.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | id | String | Mã người dùng |
| 2 | name | String | Tên người dùng |
| 3 | email | String | Email đăng nhập |
| 4 | emailVerified | DateTime | Thời điểm xác thực email |
| 5 | image | String | URL ảnh đại diện |
| 6 | password | String | Mật khẩu đã mã hóa |
| 7 | role | Enum["GUEST", "STUDENT", "TEACHER", "ADMIN"] | Vai trò người dùng |
| 8 | createdAt | DateTime | Thời điểm tạo tài khoản |
| 9 | updatedAt | DateTime | Thời điểm cập nhật thông tin gần nhất |

\* **Bảng Project**

* Tên thực thể: Project
* Mô tả: Lưu trữ thông tin các dự án

Bảng 3.4. Mô tả bảng Project.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | id | String | Mã dự án |
| 2 | title | String | Tên dự án |
| 3 | description | String | Mô tả dự án |
| 4 | thumbnail | String | URL ảnh thu nhỏ |
| 5 | status | Enum["DRAFT", "PUBLISHED", "DELETED"] | Trạng thái dự án |
| 6 | createdAt | DateTime | Thời điểm tạo dự án |
| 7 | updatedAt | DateTime | Thời điểm cập nhật gần nhất |
| 8 | publishedAt | DateTime | Thời điểm xuất bản |
| 9 | deletedAt | DateTime | Thời điểm xóa |
| 10 | authorId | String | Mã người tạo dự án |

\* **Bảng Course**

* Tên thực thể: Course
* Mô tả: Lưu trữ thông tin các khoá học

Bảng 3.5. Mô tả bảng Course.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | id | String | Mã khóa học |
| 2 | title | String | Tên khóa học |
| 3 | description | String | Mô tả khóa học |
| 4 | thumbnail | String | URL ảnh thu nhỏ |
| 5 | videoUrl | String | URL video giới thiệu |
| 6 | status | Enum["DRAFT", "PUBLISHED", "DELETED"] | Trạng thái khóa học |
| 7 | createdAt | DateTime | Thời điểm tạo khóa học |
| 8 | updatedAt | DateTime | Thời điểm cập nhật gần nhất |
| 9 | publishedAt | DateTime | Thời điểm xuất bản |
| 10 | deletedAt | DateTime | Thời điểm xóa |
| 11 | authorId | String | Mã người tạo khóa học |
| 12 | projectId | String | Mã dự án chứa khóa học |

\* **Bảng Quiz**

* Tên thực thể: Quiz
* Mô tả: Lưu trữ thông tin bài kiểm tra trắc nghiệm

Bảng 3.6. Mô tả bảng Quiz.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | id | String | Mã bài kiểm tra |
| 2 | title | String | Tên bài kiểm tra |
| 3 | description | String | Mô tả bài kiểm tra |
| 4 | thumbnail | String | URL ảnh thu nhỏ |
| 5 | duration | Int | Thời gian làm bài (phút) |
| 6 | startTime | DateTime | Thời điểm bắt đầu |
| 7 | endTime | DateTime | Thời điểm kết thúc |
| 8 | randomizeAnswers | Boolean | Cho phép xáo trộn thứ tự câu trả lời |
| 9 | status | Enum["DRAFT", "PUBLISHED", "DELETED"] | Trạng thái bài kiểm tra |
| 10 | createdAt | DateTime | Thời điểm tạo |
| 11 | updatedAt | DateTime | Thời điểm cập nhật gần nhất |
| 12 | publishedAt | DateTime | Thời điểm xuất bản |
| 13 | deletedAt | DateTime | Thời điểm xóa |
| 14 | authorId | String | Mã người tạo |
| 15 | projectId | String | Mã dự án chứa bài kiểm tra |

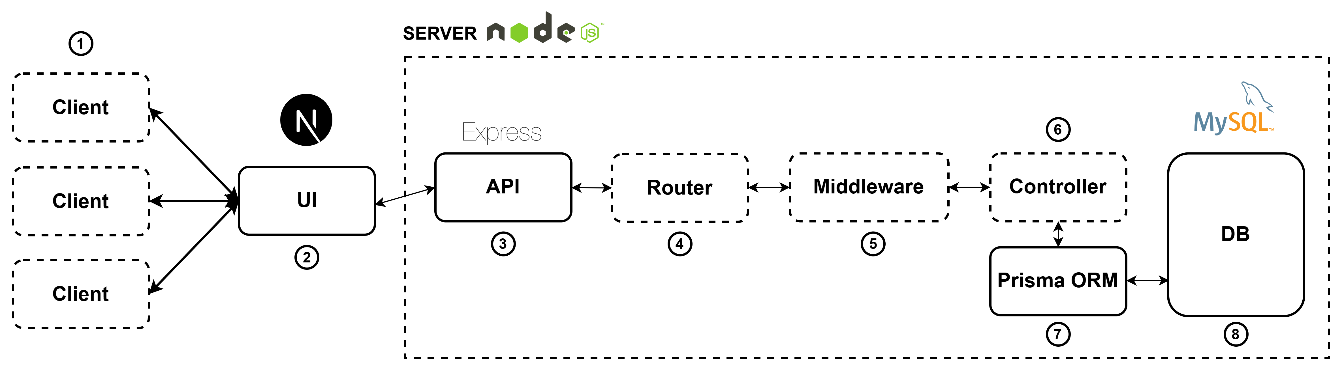
\* **Bảng Category**

* Tên thực thể: Category
* Mô tả: Lưu trữ thông tin các chủ đề

Bảng 3.7. Mô tả bảng Category.

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Tên thuộc tính** | **Kiểu dữ liệu** | **Ý nghĩa** |
| 1 | id | String | Mã chủ đề |
| 2 | name | String | Tên chủ đề |
| 3 | description | String | Mô tả chủ đề |

### Sơ đồ triển khai



Ảnh 3.1. Sơ đồ triển khai.

**Trong đó:**

(1): Các người dùng, trình duyệt.

(2): Giao diện người dùng (Next.js).

(3): Các API từ phía server (ExpressJS).

(4): Các định tuyến, dùng để truy xuất các API.

(5): Middleware, chứa các hàm dùng để xác thực người dùng, xác thực các yêu cầu được gửi đến API.

(6): Controller, chứa các hàm để xử lý các yêu cầu đến API.

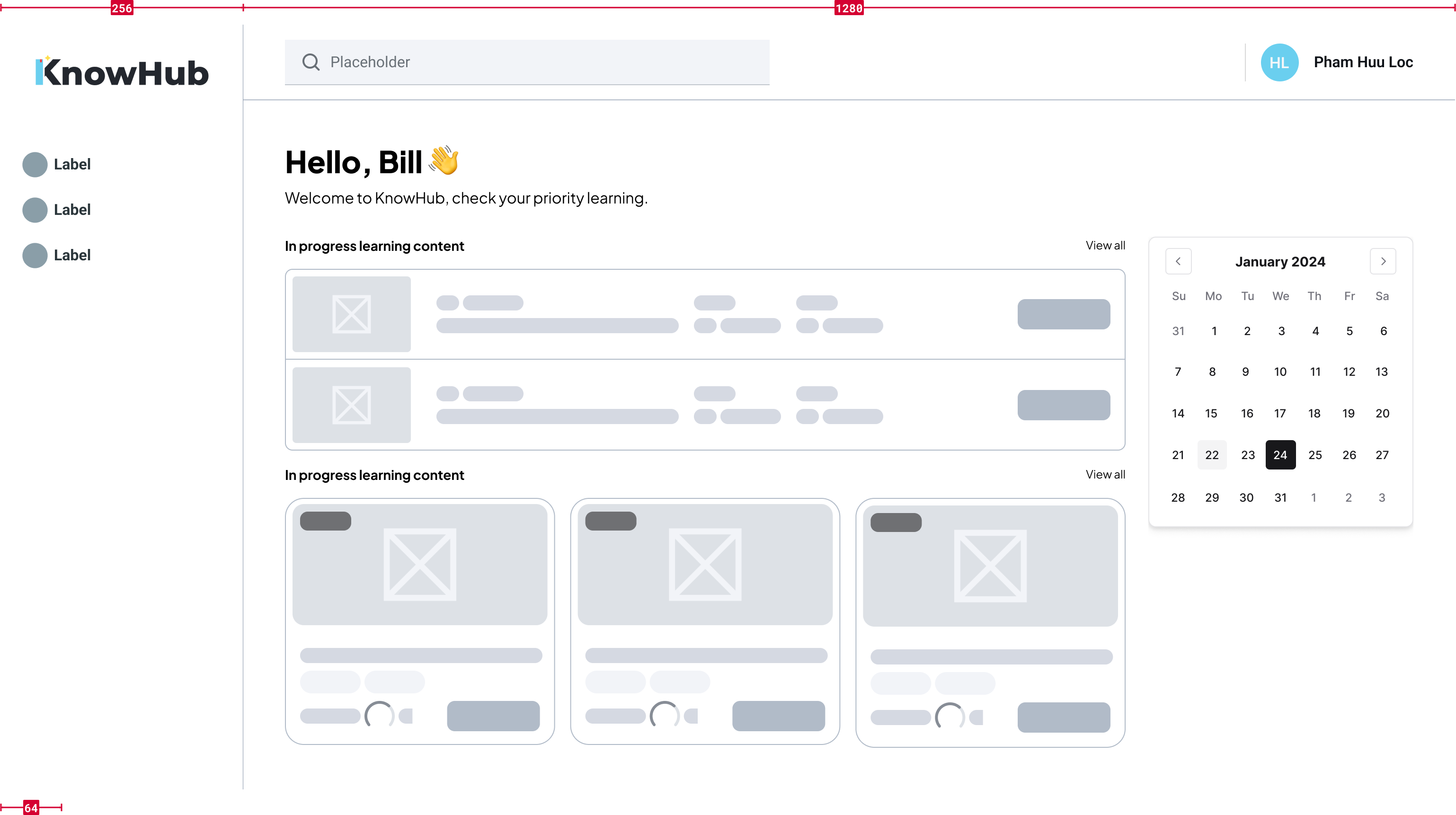
(7): Prisma ORM, cho phép thao tác với dữ liệu mà không cần viết các câu truy vấn SQL.

(8): Database (MySQL), nơi lưu trữ toàn bộ dữ liệu của trang web.

## Thiết kế giao diện

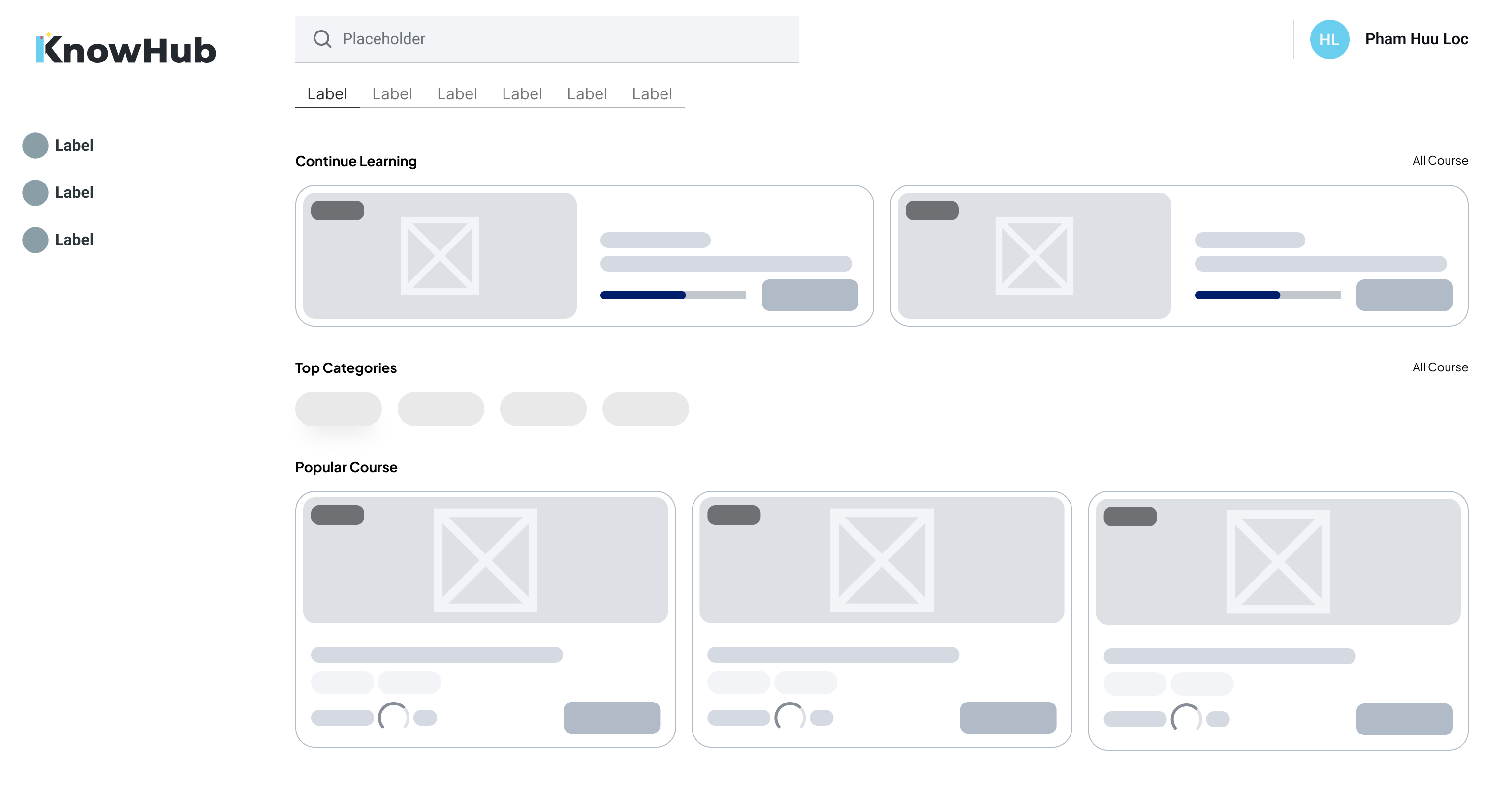
//TODO: Sơ đồ website

### Màn hình trang chủ



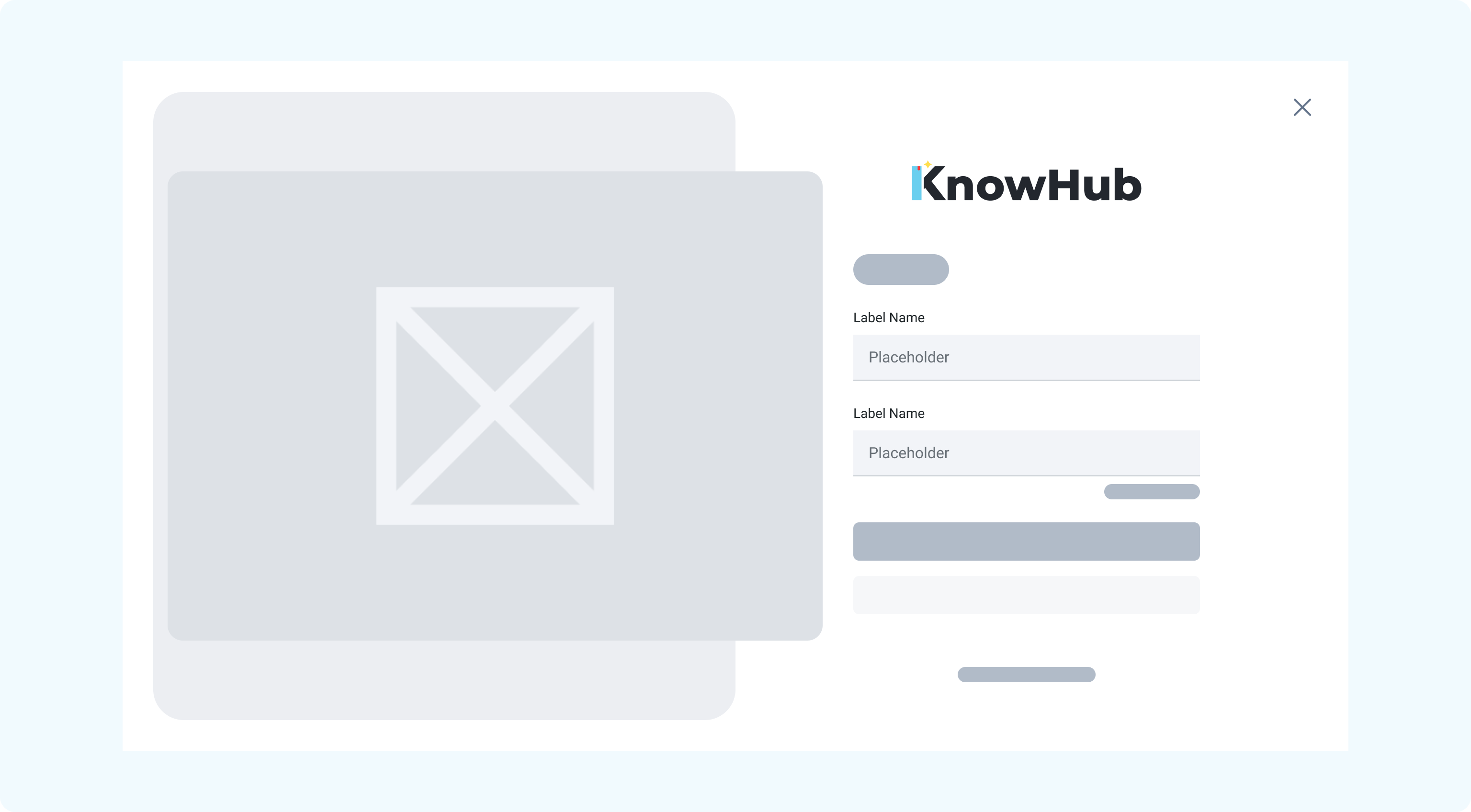
Hình 3.3. Màn hình trang chủ.

### Màn hình khoá học



Hình 3.4. Màn hình khoá học.

### Màn hình đăng nhập/ đăng xuất



Hình 3.5. Màn hình đăng nhập/ đăng xuất.

### Triển khai

Frontend: Dockerfile này sẽ xây dựng và khởi chạy Next.js ở cổng 3000.

# Build stage

FROM node:18-alpine AS builder

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

# Production stage

FROM node:18-alpine AS runner

WORKDIR /app

ENV NODE\_ENV=production

COPY --from=builder /app/next.config.js ./

COPY --from=builder /app/public ./public

COPY --from=builder /app/.next/standalone ./

COPY --from=builder /app/.next/static ./.next/static

EXPOSE 3000

CMD ["node", "server.js"]

Backend: Dockerfile này sẽ xây dựng và khởi chạy backend trên cổng 5000.

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npx prisma generate

EXPOSE 5000

CMD ["npm", "run", "dev"]

Docker Compose: Đóng vai trò quản lý và điều phối các container.

version: '3.8'

services:

frontend:

build:

context: ./src/frontend

dockerfile: Dockerfile

ports:

- "3000:3000"

environment:

- NEXT\_PUBLIC\_API\_URL=http://backend:5000

depends\_on:

- backend

backend:

build:

context: ./src/backend

dockerfile: Dockerfile

ports:

- "5000:5000"

environment:

- DATABASE\_URL=postgresql://postgres:postgres@db:5432/knowhub

- JWT\_SECRET=your\_jwt\_secret

depends\_on:

- db

db:

image: postgres:15

ports:

- "5432:5432"

environment:

- POSTGRES\_USER=postgres

- POSTGRES\_PASSWORD=postgres

- POSTGRES\_DB=knowhub

volumes:

- postgres\_data:/var/lib/postgresql/data

volumes:

postgres\_data:

# KẾT QUẢ NGHIÊN CỨU

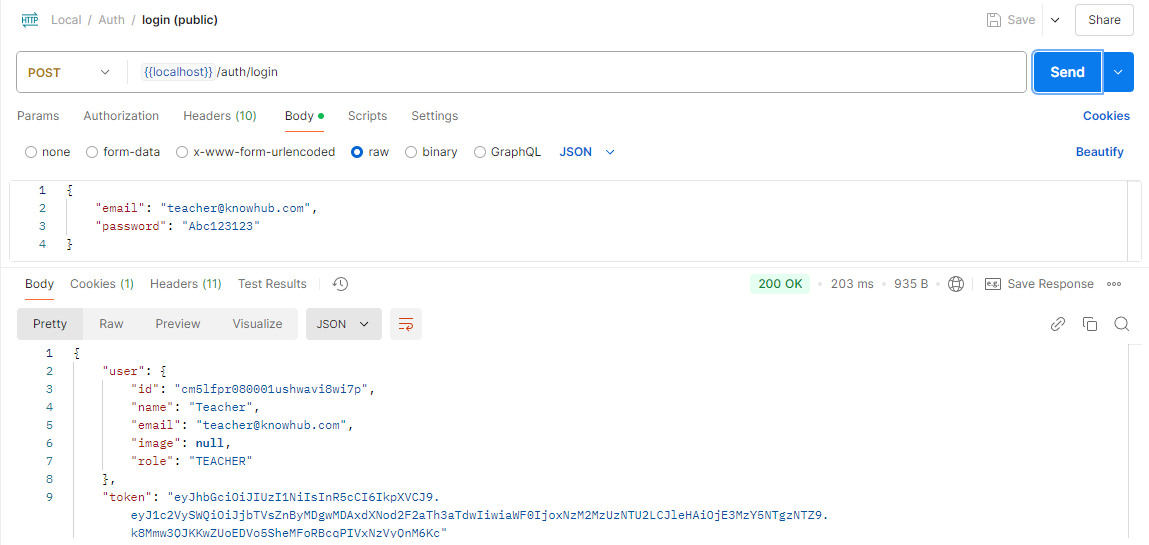
Trình bày các kết quả đạt được sau quá trình thực hiện đồ án. Có thể đánh giá về hiệu năng, trải nghiệm người dùng, hoặc trình bày các giao diện chức năng của nghiên cứu ở phần này.



## Thử nghiệm các API với Postman

### Các API xác thực

API đăng nhập với email và mật khẩu, dữ liệu trả về một chuỗi token JWT mã hoá các thông tin của người dùng. Token này được tự động lưu vào HTTP-only cookies với thời hạn 24 giờ và được gửi kèm trong mọi request tiếp theo để xác thực người dùng.

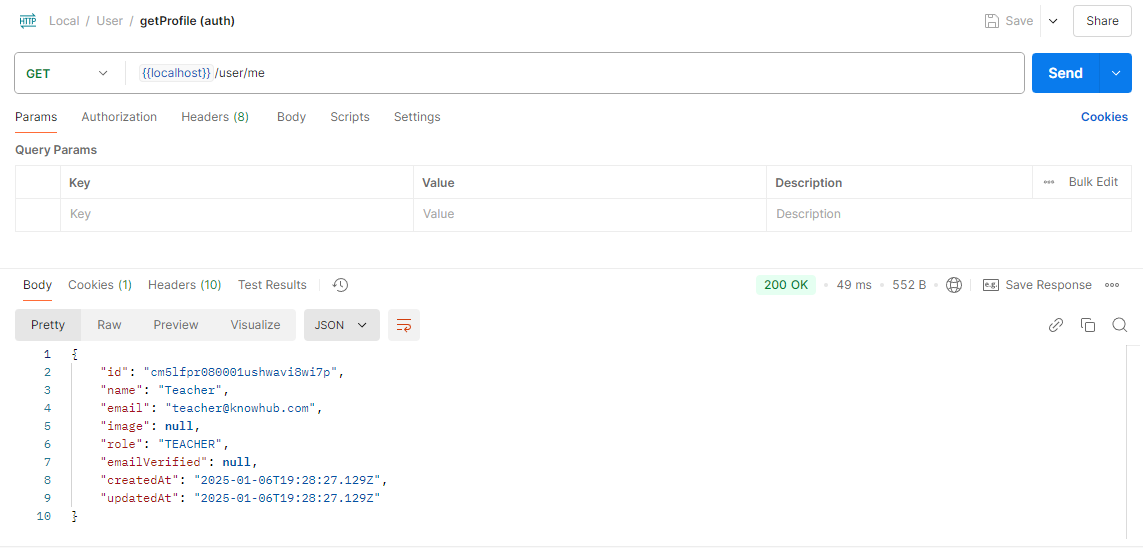


Hình 4.1. Thử nghiệm API đăng nhập với Postman.

API cho phép người dùng đăng nhập bằng tài khoản Google. Khi xác thực thành công với Google, hệ thống sẽ tự động tạo tài khoản mới nếu email chưa tồn tại trong database. Token JWT được tạo và lưu vào cookies với thời hạn 7 ngày.

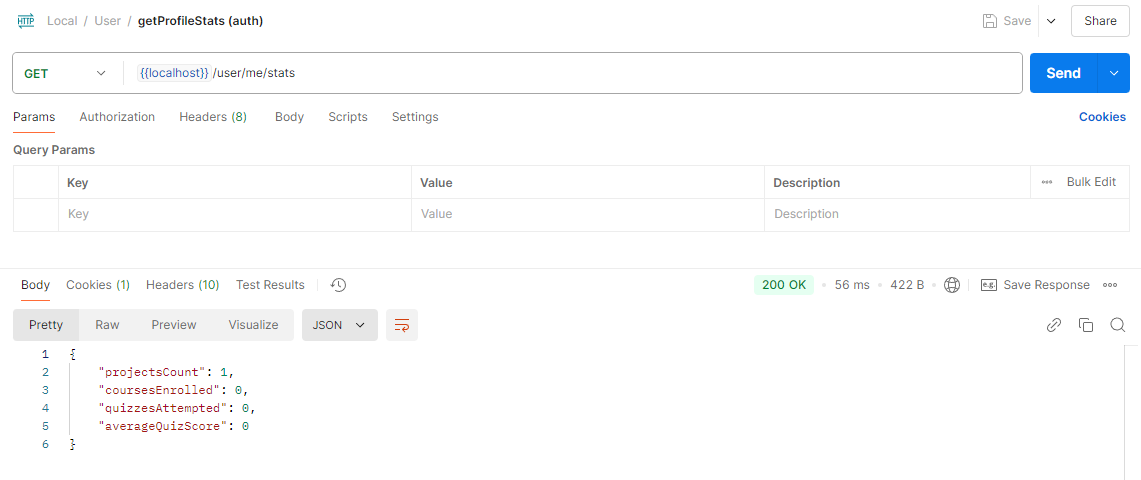
### Các API quản lý người dùng

API trả về thông tin chi tiết của người dùng đang đăng nhập, bao gồm các thông tin cơ bản và trạng thái xác thực email. API này yêu cầu token xác thực hợp lệ.



Hình 4.2. Kiểm thử API lấy thông tin người dùng đang đăng nhập với Postman.

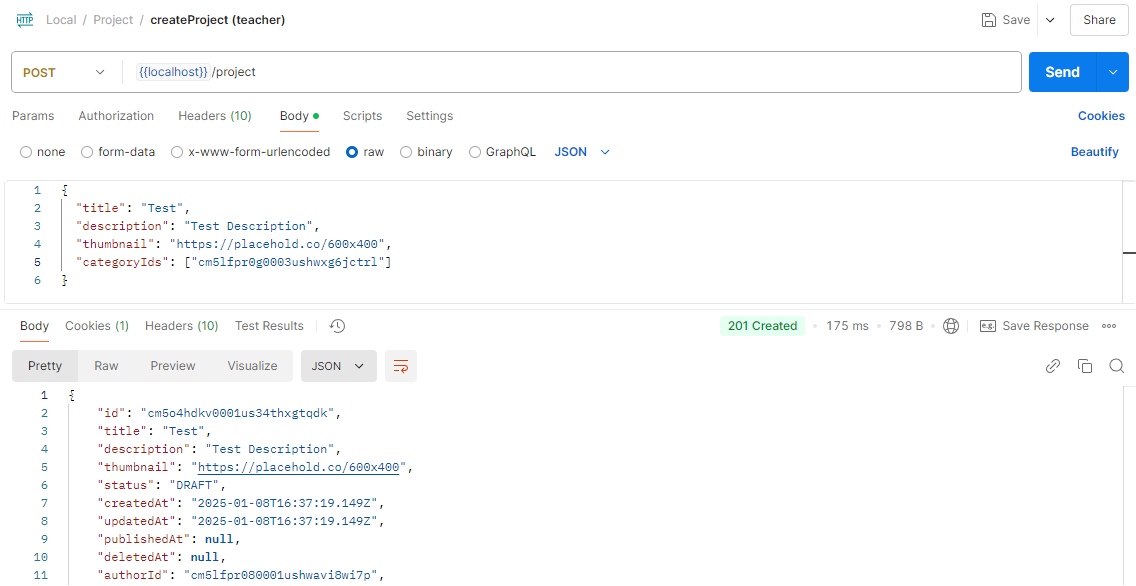
API trả về các thống kê về hoạt động của người dùng bao gồm: số lượng dự án đã tạo, số khóa học đã tham gia, số bài kiểm tra đã làm và điểm trung bình các bài kiểm tra.



Hình 4.3. Kiểm thử API lấy thông tin thống kê hoạt động người dùng.

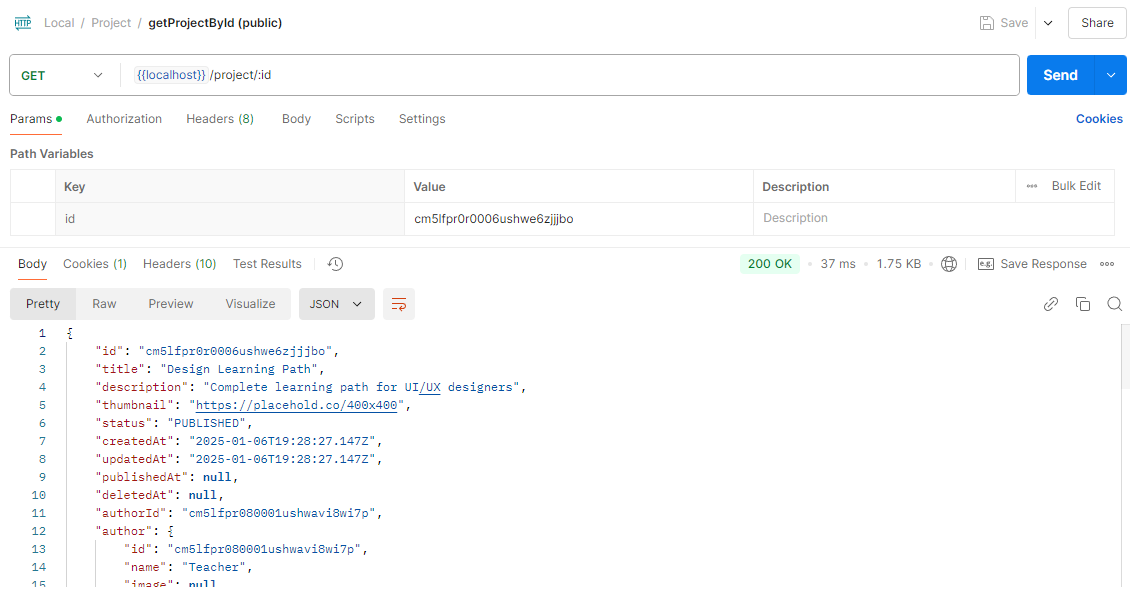
### Các API quản lý dự án, khoá học, bài kiểm tra

API cho phép giáo viên tạo một dự án mới với tiêu đề, mô tả, ảnh thumbnail và các danh mục liên quan. Dự án được tạo sẽ tự động liên kết với giáo viên tạo ra nó.



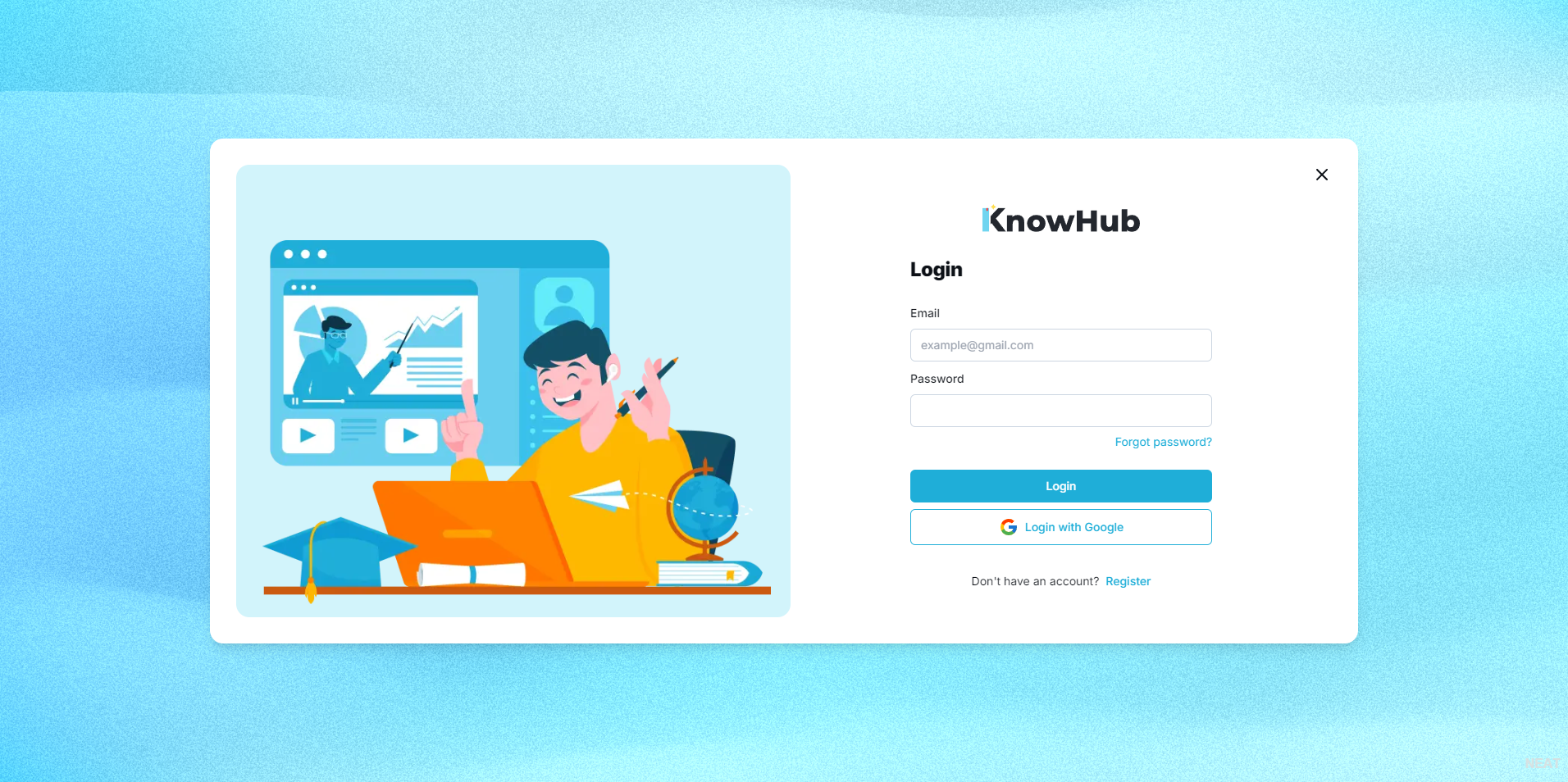
Hình 4.4. Kiểm thử API tạo mới dự án với Postman.

API trả về thông tin chi tiết của một dự án, bao gồm thông tin tác giả, danh mục và các khóa học, bài kiểm tra đã được xuất bản thuộc dự án đó.

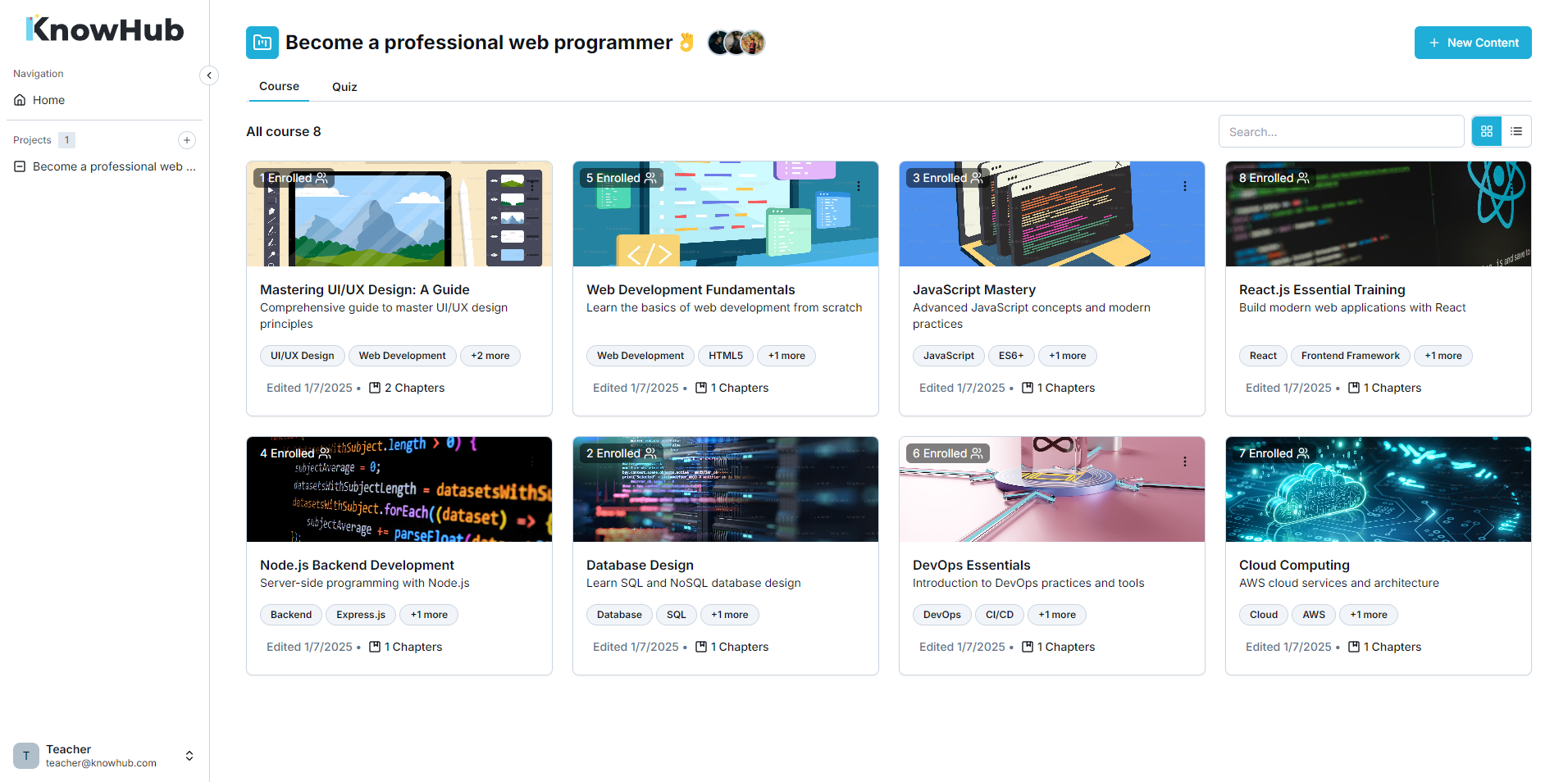


Hình 4.5. Kiểm thử API lấy thông tin chi tiết của một dự án với Postman.

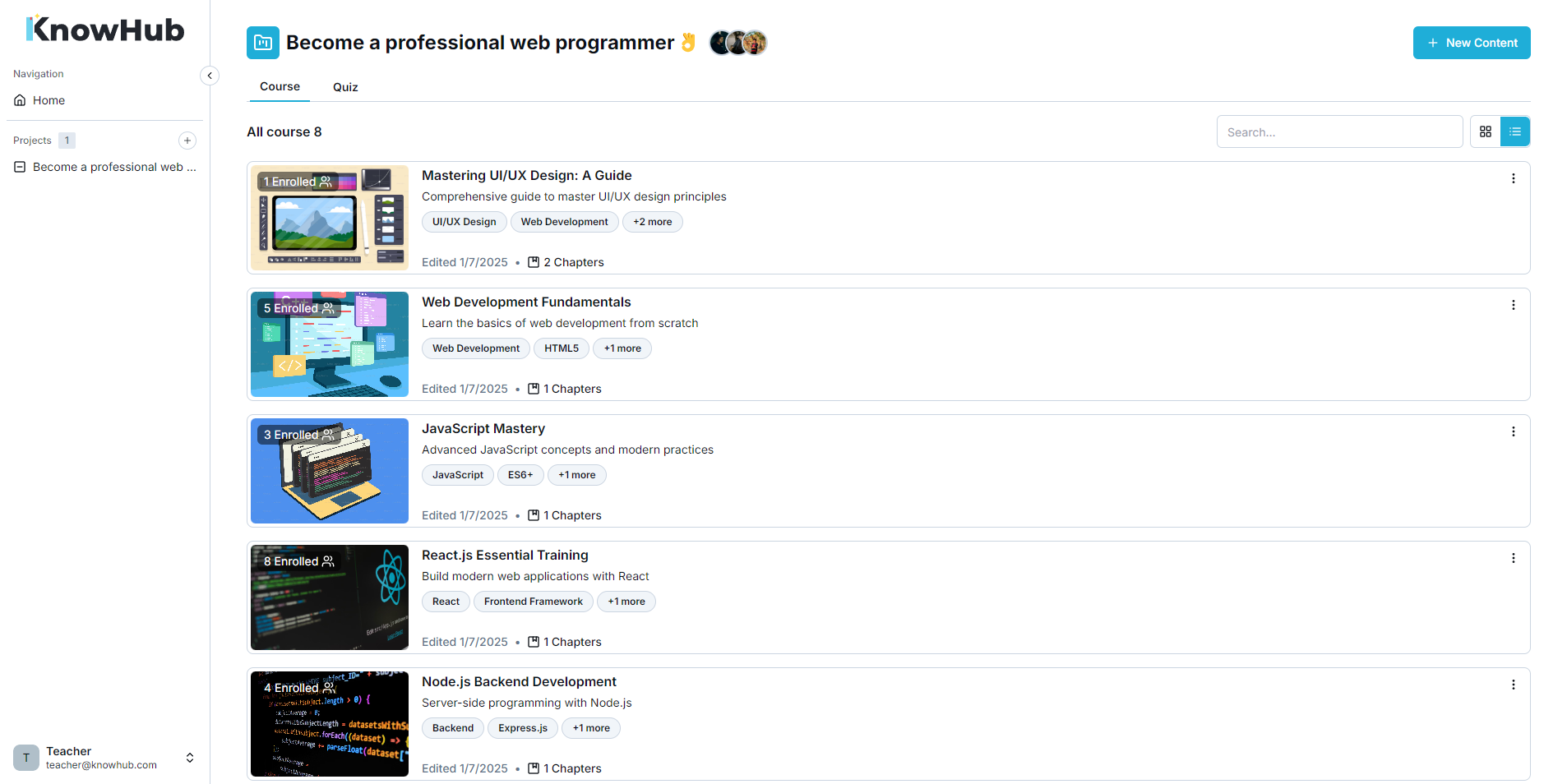
## Giao diện chức năng người dùng



Hình 4.6. Giao diện đăng nhập.



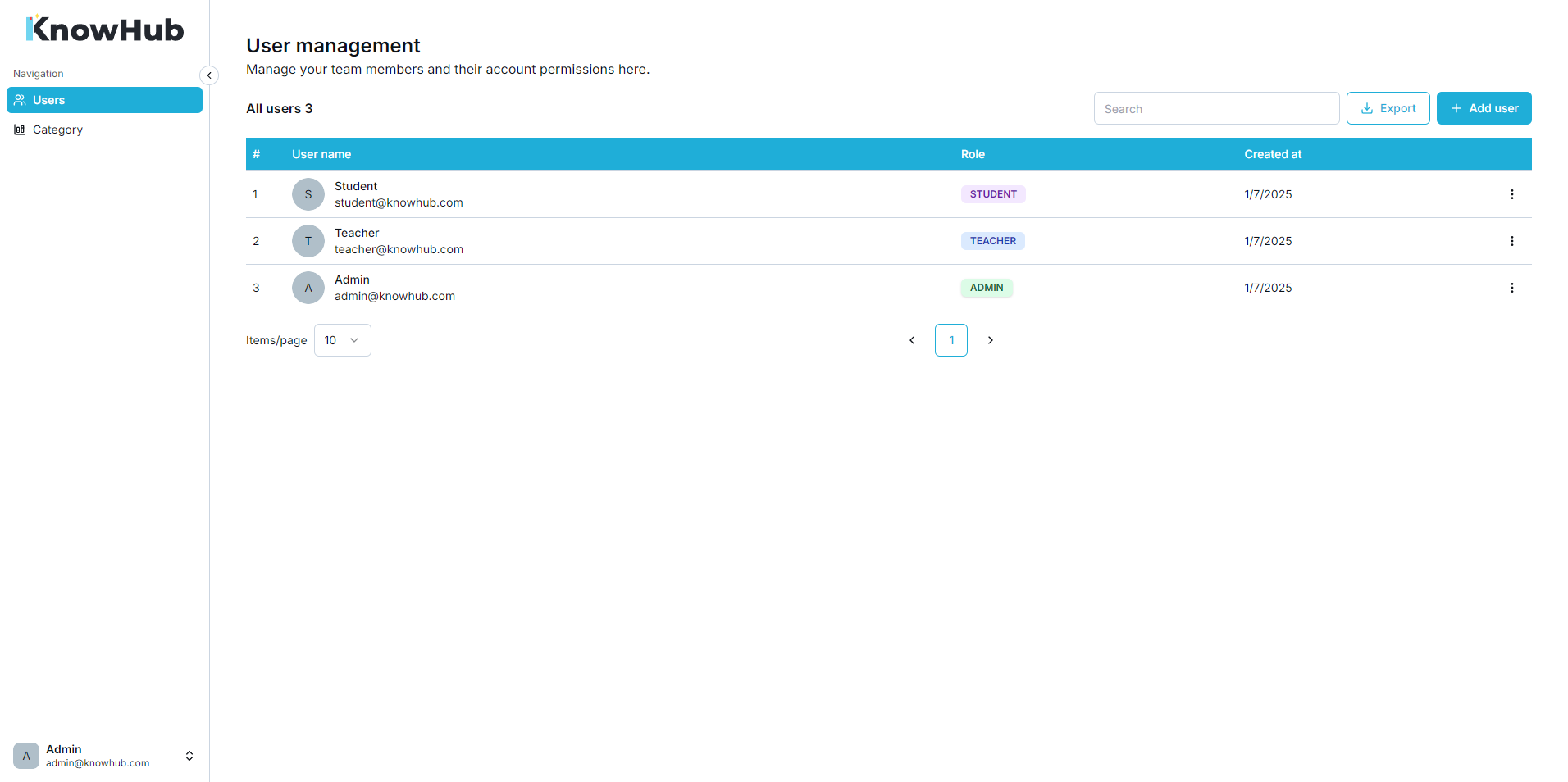
Hình 4.7. Giao diện quản lý dự án (1).



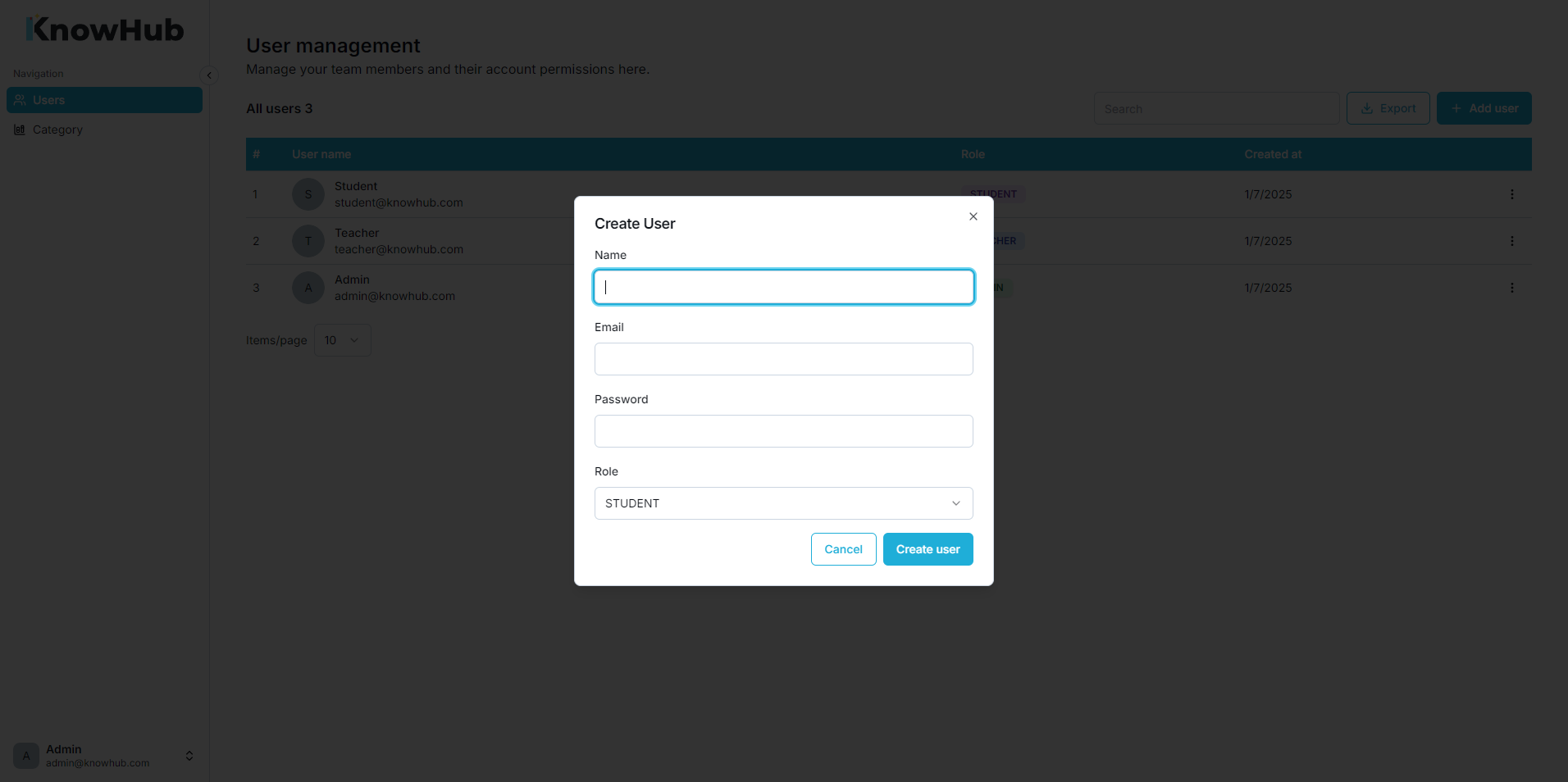
Hình 4.8. Giảo diện quản lý dự án (2).

## Giao diện chức năng người quản trị

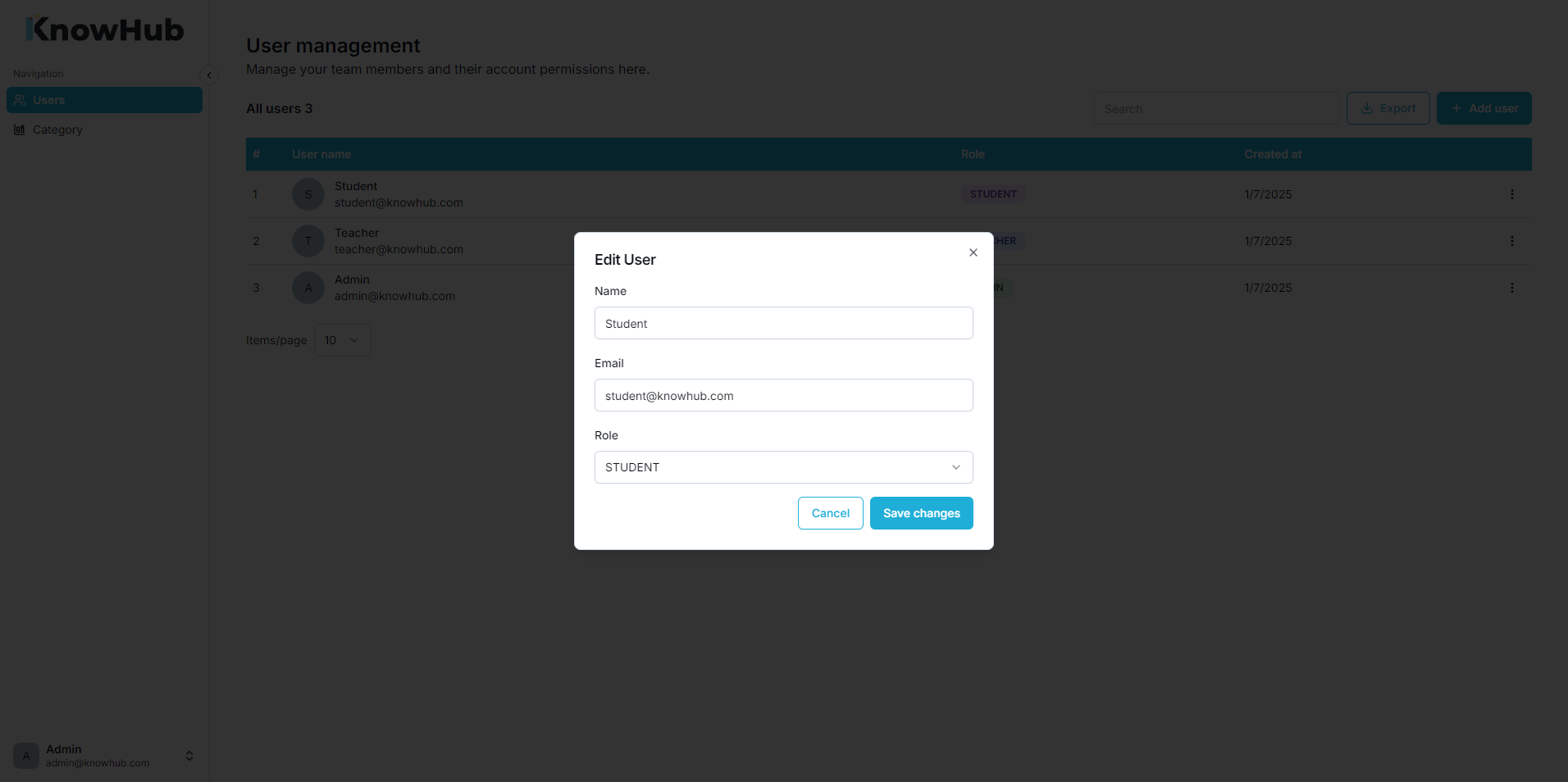
### Giao diện quản lý người dùng



Hình 4.9. Giao diện quản lý người dùng.

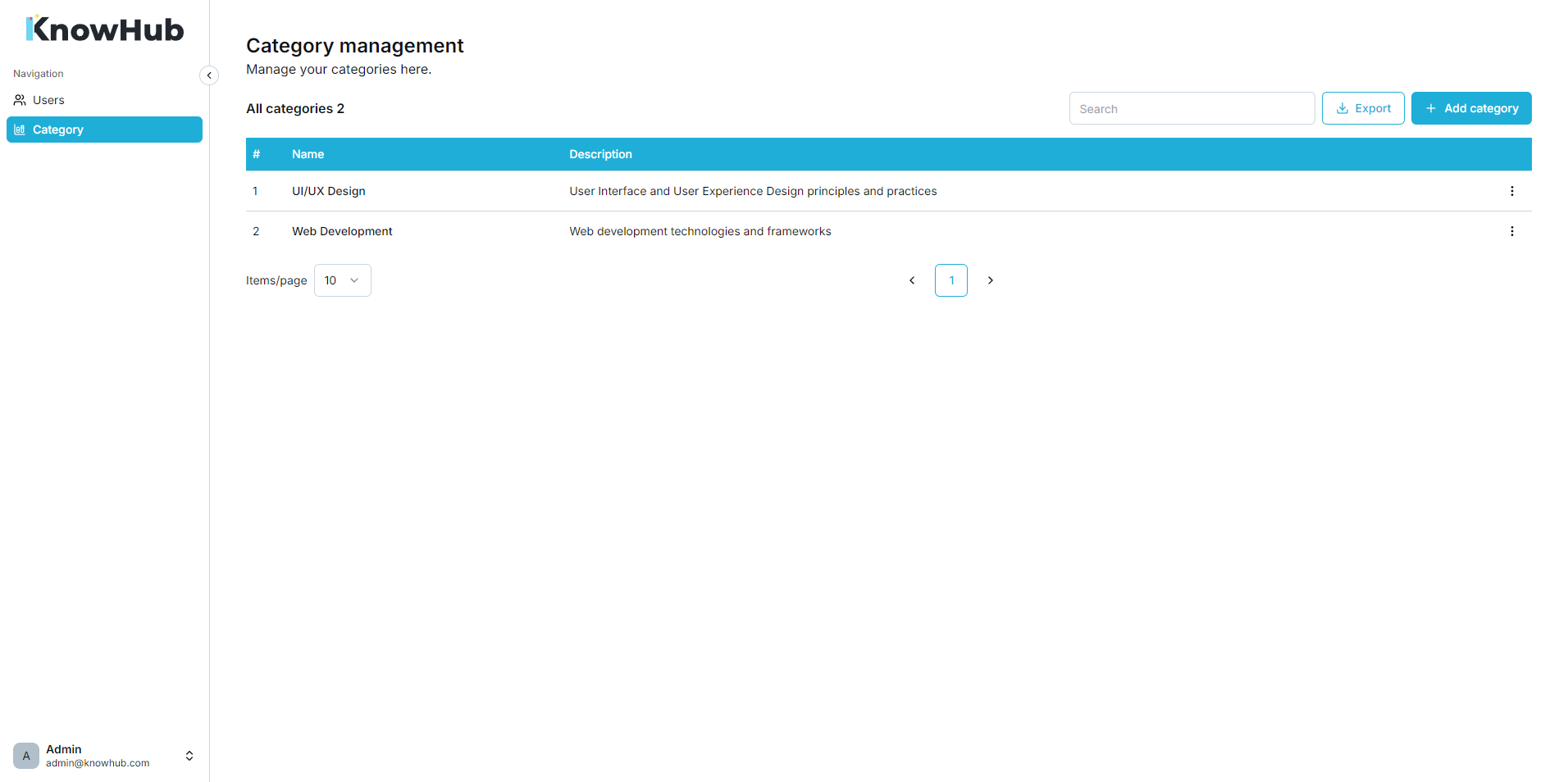


Hình 4.10. Giao diện chức năng tạo người dùng mới.

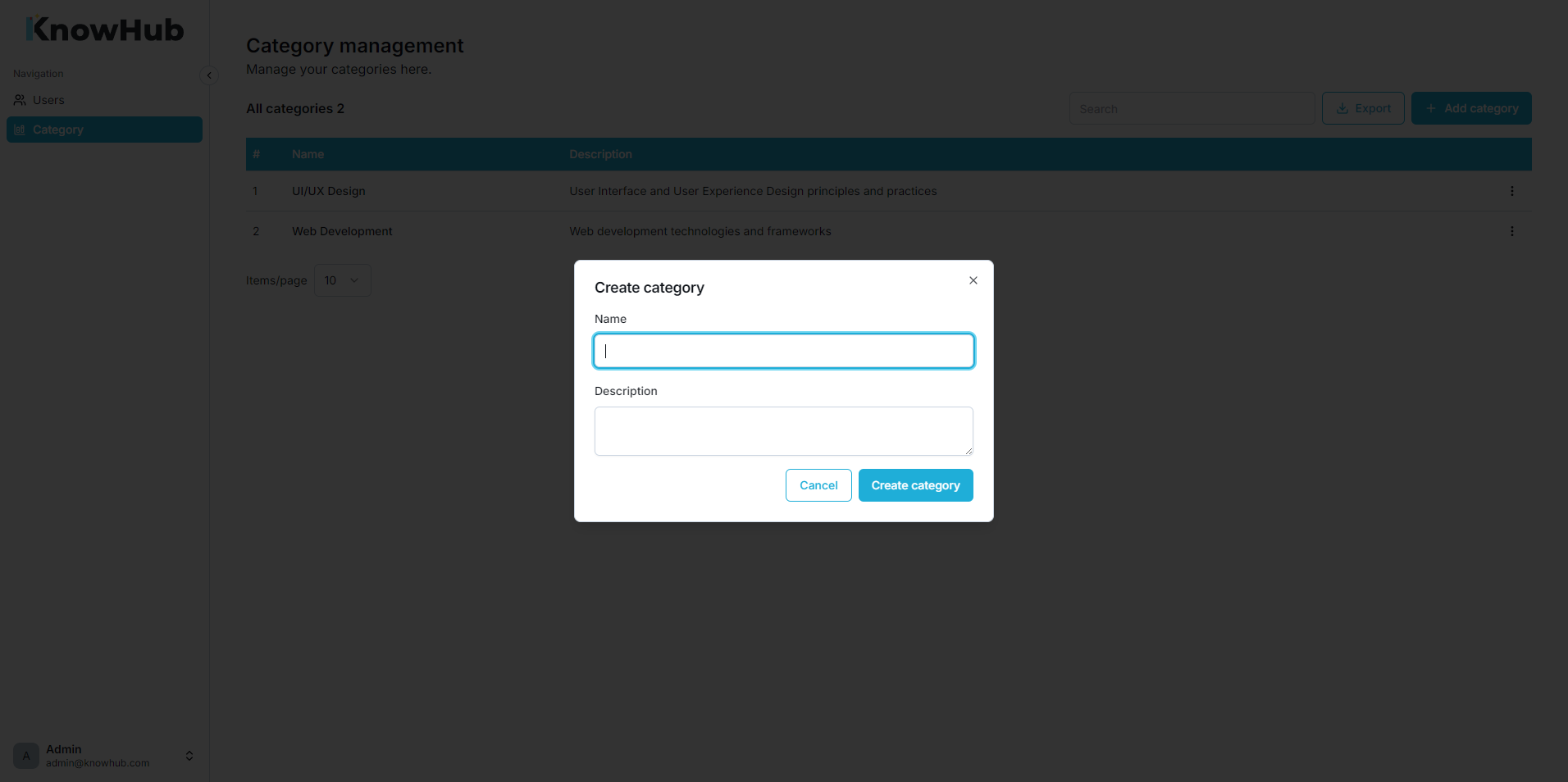


Hình 4.11. Giao diện chức năng chỉnh sửa thông tin người dùng.

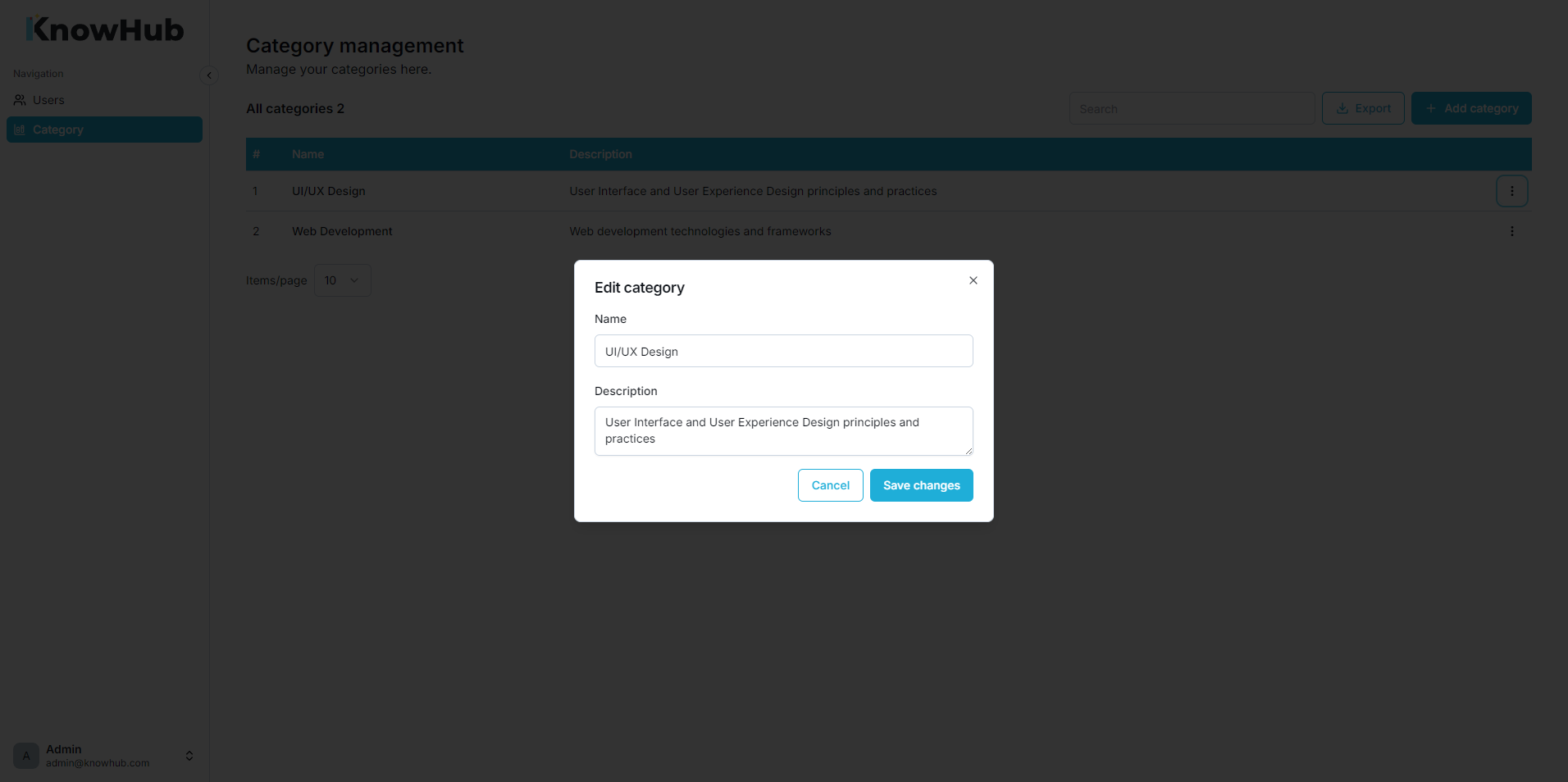
### Giao diện quản lý chủ đề



Hình 4.12. Giao diện quản lý chủ đề.



Hình 4.13. Giao diện chức năng thêm chủ đề mới.



Hình 4.14. Giao diện chức năng chỉnh sửa thông tin chủ đề.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN



## Kết quả đạt được

**Nghiên cứu sâu về Next.js:** Đã tiến hành nghiên cứu kỹ về Next.js framework. So sánh và đánh giá được ưu điểm và nhược điểm của Next.js.

**Xây dựng hệ thống quản lý học tập:** Áp dụng, xây dựng thành công hệ thống quản lý học tập.

## Hướng phát triển

Tiếp tục nghiên cứu về Next.js và mở rộng phạm vi ứng dụng của nó dự án khác nhau để đem lại giá trị cao hơn trong việc phát triển ứng dụng web.

Phát triển các chức năng như quản lý tài liệu học tập, cải thiện các tính năng phản hồi của người dùng.

# DANH MỤC TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "Github," 2024. [Online]. Available:  https://github.com/vercel/next.js. |
| [2] | "Techtarget," 2024. [Online]. Available:  https://www.techtarget.com/whatis/definition/Nodejs. |
| [3] | "Nodejs," 2024. [Online]. Available:  https://nodejs.org/en/about. |
| [4] | "ExpressJS," 2024. [Online]. Available:  https://expressjs.com/en/guide/using-middleware.html. |
| [5] | "Prisma," 2024. [Online]. Available:  https://www.prisma.io/docs/orm/overview/introduction/what-is-prisma. |