**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO MÔN HỌC**

**CÔNG NGHỆ PHẦN MỀM**

**WEBSITE ĐĂNG KÝ**

**KHÓA HỌC TRỰC TUYẾN**

|  |  |
| --- | --- |
| *Giảng viên hướng dẫn:* **TS. Nguyễn Bảo Ân** | *Sinh viên thực hiện:*  **Phạm Thế Vinh – 110122208 –** **DA22TTC**  **Nguyễn Phúc Vinh –** **110122205 –** **DA22TTC** |

**Vĩnh Long, tháng 7 năm 2025**

**TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO MÔN HỌC**

**CÔNG NGHỆ PHẦN MỀM**

**WEBSITE ĐĂNG KÝ**

**KHÓA HỌC TRỰC TUYẾN**

|  |  |
| --- | --- |
| *Giảng viên hướng dẫn:* **TS. Nguyễn Bảo Ân** | *Sinh viên thực hiện:*  **Phạm Thế Vinh – 110122208 –** **DA22TTC**  **Nguyễn Phúc Vinh –** **110122205 –** **DA22TTC** |

**Vĩnh Long, tháng 7 năm 2025**

**LỜI CẢM ƠN**

Chúng tôi xin gửi lời cảm ơn chân thành đến TS. Nguyễn Bảo Ân, người đã giảng dạy môn Công nghệ phần mềm và hướng dẫn chúng tôi thực hiện đề tài này. Nhờ sự giảng dạy nhiệt tình và chia sẻ kinh nghiệm quý giá từ thầy, chúng tôi đã nắm vững hơn những kiến thức cần thiết để thực hiện đề tài. Những lời góp ý và sự hỗ trợ của thầy đã giúp chúng tôi hoàn thành đề tài và học được nhiều điều bổ ích.

Trong quá trình thực hiện đề tài, do kiến thức còn hạn chế, chúng tôi khó tránh khỏi những thiếu sót trong việc tìm hiểu, phân tích và thực hiện. Chúng tôi rất mong nhận được những ý kiến đóng góp quý báu từ thầy/cô để có thể khắc phục, cải thiện và từng bước hoàn thiện hơn.

Chúng tôi xin chân thành cảm ơn!

**MỤC LỤC**

[CHƯƠNG 1. TỔNG QUAN 1](#_Toc204285703)

[1.1. Giới thiệu đề tài 1](#_Toc204285704)

[1.1.1. Lý do chọn đề tài 1](#_Toc204285705)

[1.1.2. Mục tiêu của đề tài 1](#_Toc204285706)

[1.2. Khái niệm cơ bản về công nghệ phần mềm 1](#_Toc204285707)

[1.3. Các mô hình phát triển phần mềm 2](#_Toc204285708)

[1.4. Sự cần thiết của phần mềm phân tán, microservices và đám mây 3](#_Toc204285709)

[1.5. Các khía cạnh về bảo mật và an toàn của phần mềm 4](#_Toc204285710)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 5](#_Toc204285711)

[2.1. Phân tích yêu cầu 5](#_Toc204285712)

[2.1.1. Các chức năng chính 5](#_Toc204285713)

[2.1.2. Các yêu cầu phi chức năng 6](#_Toc204285714)

[2.2. Thiết kế hệ thống 6](#_Toc204285715)

[2.2.1. Kiến trúc tổng thể 6](#_Toc204285716)

[2.2.2. Thiết kế cơ sở dữ liệu 7](#_Toc204285717)

[2.2.3. Thiết kế API 10](#_Toc204285718)

[2.2.4. Thiết kế giao diện 17](#_Toc204285719)

[CHƯƠNG 3. TRIỂN KHAI DỰ ÁN 23](#_Toc204285720)

[3.1. Các công nghệ đã sử dụng 23](#_Toc204285721)

[3.2. Quy trình CI/CD với GitHub Actions 24](#_Toc204285722)

[3.3. Cấu hình Docker 25](#_Toc204285723)

[CHƯƠNG 4. QUẢN LÝ DỰ ÁN 28](#_Toc204285724)

[4.1. Mô hình phát triển Agile 28](#_Toc204285725)

[4.2. Phân công nhiệm vụ 28](#_Toc204285726)

[CHƯƠNG 5. KIỂM THỬ 32](#_Toc204285727)

[5.1. Chiến lược kiểm thử 32](#_Toc204285728)

[5.2. Kết quả kiểm thử API với Postman 32](#_Toc204285729)

[CHƯƠNG 6. ĐÁNH GIÁ VÀ KẾT LUẬN 40](#_Toc204285730)

[6.1. Khó khăn gặp phải 40](#_Toc204285731)

[6.2. Bài học rút ra 40](#_Toc204285732)

[6.3. Đề xuất cải thiện 40](#_Toc204285733)

[DANH MỤC TÀI LIỆU THAM KHẢO 41](#_Toc204285734)

[PHỤ LỤC 42](#_Toc204285735)

**DANH MỤC HÌNH ẢNH VÀ BẢNG**

[Hình 2.1. Sơ đồ kiến trúc tổng thể 7](#_Toc204285736)

[Hình 2.2. Mô hình thực thể kết hợp 8](#_Toc204285737)

[Hình 2.3. Mô hình dữ liệu quan hệ 8](#_Toc204285738)

[Hình 2.4. Giao diện trang chủ chưa đăng nhập 17](#_Toc204285739)

[Hình 2.5. Giao diện đăng ký 18](#_Toc204285740)

[Hình 2.6. Giao diện đăng nhập 18](#_Toc204285741)

[Hình 2.7. Giao diện yêu cầu đặt lại mật khẩu 18](#_Toc204285742)

[Hình 2.8. Giao diện đặt mật khẩu mới 19](#_Toc204285743)

[Hình 2.9. Giao diện trang chủ đã đăng nhập 19](#_Toc204285744)

[Hình 2.10. Giao diện xem thông tin cá nhân 20](#_Toc204285745)

[Hình 2.11. Giao diện xem thông tin khóa học 20](#_Toc204285746)

[Hình 2.12. Giao diện soạn thảo khóa học 21](#_Toc204285747)

[Hình 2.13. Giao diện xem các học viên đã đăng ký 22](#_Toc204285748)

[Hình 4.1. Burndown chart của Sprint 1 29](#_Toc204285749)

[Hình 4.2. Burndown chart của Sprint 2 30](#_Toc204285750)

[Hình 4.3. Burndown chart của Sprint 3 30](#_Toc204285751)

[Hình 4.4. Burndown chart của Sprint 4 31](#_Toc204285752)

[Hình 5.1. Kết quả kiểm thử API đăng ký tài khoản 33](#_Toc204285753)

[Hình 5.2, Kết quả kiểm thử API đăng nhập 33](#_Toc204285754)

[Hình 5.3. Kết quả kiểm thử API yêu cầu đặt lại mật khẩu 34](#_Toc204285755)

[Hình 5.4. Kết quả kiểm thử API đặt lại mật khẩu 34](#_Toc204285756)

[Hình 5.5. Kết quả kiểm thử API lấy một người dùng bằng id 34](#_Toc204285757)

[Hình 5.6. Kết quả kiểm thử API cập nhật thông tin người dùng bằng id 35](#_Toc204285758)

[Hình 5.7. Kết quả kiểm thử API tạo khóa học 35](#_Toc204285759)

[Hình 5.8. Kết quả kiểm thử API cập nhật khóa học bằng id 36](#_Toc204285760)

[Hình 5.9. Kết quả kiểm thử API xóa khóa học bằng id 36](#_Toc204285761)

[Hình 5.10. Kết quả kiểm thử API tạo danh mục khóa học 37](#_Toc204285762)

[Hình 5.11. Kết quả kiểm thử API lấy các danh mục khóa học 37](#_Toc204285763)

[Hình 5.12. Kết quả kiểm thử API đăng ký khóa học bằng id 38](#_Toc204285764)

[Hình 5.13. Kết quả kiểm thử API hủy đăng ký khóa học bằng id 38](#_Toc204285765)

[Hình 5.14. Kết quả kiểm thử API lấy danh sách học viên đăng ký bằng id khóa học 39](#_Toc204285766)

[Hình 5.15. Kết quả kiểm thử API lấy một khóa học bằng id 39](#_Toc204285767)

[Bảng 2.1. Mô tả bảng users 8](#_Toc204285768)

[Bảng 2.2. Mô tả bảng courseCategories 9](#_Toc204285769)

[Bảng 2.3. Mô tả bảng courses 9](#_Toc204285770)

[Bảng 2.4. Mô tả bảngenrollments 10](#_Toc204285771)

[Bảng 4.1. Bảng phân công nhiệm vụ ở các Sprint 28](#_Toc204285772)

# TỔNG QUAN

## Giới thiệu đề tài

### Lý do chọn đề tài

Sự phát triển của giáo dục trực tuyến đã mở ra nhiều cơ hội mới cho việc tổ chức và tham gia các khóa học từ xa. Với nhu cầu học tập ngày càng linh hoạt và đa dạng, việc xây dựng một nền tảng hỗ trợ đăng ký và theo dõi khóa học trở nên cần thiết, kể cả ở quy mô nhỏ. Các trung tâm đào tạo, lớp học ngắn hạn hay các nhóm giảng dạy tự tổ chức cũng đang dần chuyển sang hình thức trực tuyến để tiếp cận học viên một cách thuận tiện hơn.

Xuất phát từ thực tế đó, đề tài tập trung xây dựng một website đơn giản nhưng hiệu quả, cho phép người học dễ dàng tìm kiếm và đăng ký khóa học, đồng thời giúp người dạy tổ chức và chia sẻ nội dung học tập một cách thuận tiện. Đây là một bước tiếp cận thực tế, giúp nhóm sinh viên rèn luyện kỹ năng thiết kế và phát triển ứng dụng web, đồng thời góp phần giải quyết nhu cầu cụ thể trong môi trường học tập hiện nay.

### Mục tiêu của đề tài

Tạo một nền tảng cho phép người dùng dễ dàng tìm kiếm thông tin khóa học, xem chi tiết và thực hiện đăng ký trực tuyến.

Hỗ trợ giảng viên đăng tải nội dung khóa học, theo dõi số lượng người tham gia và cập nhật thông tin một cách thuận tiện.

Áp dụng các công nghệ hiện đại trong phát triển phần mềm, như ReactJS cho giao diện người dùng, NodeJS và Express cho phía máy chủ, MySQL để lưu trữ dữ liệu, và tích hợp các công cụ như Docker, GitHub Actions cho triển khai tự động.

Giúp sinh viên trong nhóm thực hành toàn diện quy trình phát triển ứng dụng web từ thiết kế, lập trình, kiểm thử đến triển khai và vận hành.

## ****Khái niệm cơ bản về công nghệ phần mềm****

Công nghệ phần mềm (Software Engineering - SE) là một lĩnh vực quan trọng trong ngành công nghệ thông tin, chuyên nghiên cứu và ứng dụng các nguyên tắc kỹ thuật vào việc xây dựng, phát triển, vận hành và bảo trì phần mềm một cách có hệ thống, có kỷ luật và định lượng được. Theo định nghĩa của IEEE, đây là quá trình áp dụng kỹ thuật vào phần mềm nhằm tạo ra các sản phẩm phần mềm có chất lượng cao, hoạt động ổn định và có khả năng bảo trì trong suốt vòng đời sử dụng.

Phần mềm không chỉ đơn thuần là các dòng mã lập trình, mà còn là sự kết hợp của mã nguồn, tài liệu kỹ thuật, dữ liệu cấu hình và các công cụ hỗ trợ triển khai, bảo trì. Trong thực tế, phần mềm có thể được phân loại thành:

* Phần mềm chuyên dụng (Custom Software): Phát triển theo yêu cầu riêng của một tổ chức hay doanh nghiệp cụ thể, thường được điều chỉnh và cập nhật theo nhu cầu thay đổi trong nội bộ.
* Phần mềm thông dụng (Generic Software): Là phần mềm thương mại phổ biến, được phát triển để bán đại trà cho nhiều người dùng cuối (ví dụ: Microsoft Office, Zoom...).
* Phần mềm nhúng (Embedded Software): Là phần mềm được tích hợp bên trong các thiết bị phần cứng như TV, tủ lạnh, thiết bị IoT, giúp thiết bị đó thực hiện một số chức năng nhất định.

Ngày nay, công nghệ phần mềm hiện đại thường kết hợp nhiều phương pháp và công cụ tiên tiến như:

* Phương pháp phát triển linh hoạt (Agile) để rút ngắn thời gian phản hồi và cải tiến liên tục;
* Kiến trúc microservices để chia nhỏ hệ thống thành các thành phần độc lập;
* Công nghệ điện toán đám mây (Cloud) để triển khai linh hoạt trên nhiều hạ tầng;
* DevOps và CI/CD để tự động hóa quy trình kiểm thử, đóng gói và triển khai phần mềm.

## ****Các mô hình phát triển phần mềm****

Việc lựa chọn mô hình phát triển phù hợp là bước khởi đầu quan trọng trong bất kỳ dự án phần mềm nào. Mỗi mô hình đều có ưu điểm riêng và được lựa chọn dựa trên quy mô dự án, mức độ yêu cầu thay đổi, thời gian, và nguồn lực.

Một số mô hình phát triển phần mềm phổ biến bao gồm:

* Mô hình thác nước (Waterfall): Là mô hình tuyến tính truyền thống, trong đó các giai đoạn phát triển diễn ra theo trình tự từ phân tích, thiết kế, triển khai, kiểm thử đến bảo trì. Mô hình này phù hợp với các dự án có yêu cầu ổn định, ít thay đổi.
* Mô hình xoắn ốc (Spiral): Kết hợp giữa phát triển lặp và phân tích rủi ro. Mô hình này phù hợp với các dự án lớn, phức tạp, có độ rủi ro cao và yêu cầu kiểm soát kỹ lưỡng.
* Mô hình nguyên mẫu (Prototyping): Tập trung vào việc nhanh chóng xây dựng một phiên bản thử nghiệm (prototype) để người dùng trải nghiệm và phản hồi, từ đó hoàn thiện sản phẩm.
* Mô hình phát triển linh hoạt (Agile): Đây là mô hình được sử dụng phổ biến hiện nay, đặc biệt trong môi trường phát triển nhanh và cần thích nghi với sự thay đổi liên tục. Agile chia nhỏ quá trình phát triển thành các vòng lặp (sprint), mỗi vòng đều có thể đưa sản phẩm ra thử nghiệm và nhận phản hồi từ người dùng.

## ****Sự cần thiết của phần mềm phân tán, microservices và đám mây****

Trong kiến trúc hiện đại, việc triển khai hệ thống trên nền đám mây (cloud) mang lại nhiều lợi ích rõ rệt:

* Tính linh hoạt: Dễ dàng thay đổi cấu hình hạ tầng theo nhu cầu sử dụng thực tế.
* Tính mở rộng (scalability): Có thể tăng số lượng máy chủ hoặc tài nguyên theo số lượng người dùng tăng.
* Tính đàn hồi (elasticity): Cho phép hệ thống tự động co giãn tài nguyên theo lưu lượng.
* Tiết kiệm chi phí: Chỉ cần trả tiền cho tài nguyên sử dụng, không cần đầu tư hạ tầng ban đầu.

Kiến trúc phân tán (distributed software) cũng rất phù hợp vì các thành phần như frontend, backend và cơ sở dữ liệu có thể triển khai độc lập trên nhiều máy chủ khác nhau, giao tiếp với nhau thông qua API. Điều này giúp hệ thống dễ bảo trì, nâng cấp từng phần và tránh sự phụ thuộc lẫn nhau.

Cuối cùng, microservices là một hướng tiếp cận lý tưởng cho các hệ thống hiện đại. Thay vì xây dựng một hệ thống lớn nguyên khối, microservices chia hệ thống thành các dịch vụ nhỏ độc lập. Các dịch vụ này có thể triển khai, nâng cấp và tái sử dụng riêng biệt, rất phù hợp để thực hành triển khai phần mềm.

## Các khía cạnh về bảo mật và an toàn của phần mềm

**Tính an toàn (Safety):** Tính an toàn phần mềm đề cập đến khả năng hệ thống hoạt động mà không gây hại cho người dùng, dữ liệu hoặc môi trường. Đặc biệt quan trọng trong các hệ thống có tác động thực như y tế, giao thông, tài chính… Trong các hệ thống thông tin, an toàn thường liên quan đến việc ngăn chặn hành vi không mong muốn, xử lý lỗi đúng cách và đảm bảo hệ thống không bị sập hoặc phản ứng nguy hiểm khi gặp sự cố.

**Tính riêng tư (Privacy):** Tính riêng tư là khả năng bảo vệ thông tin cá nhân và nhạy cảm khỏi truy cập trái phép. Phần mềm cần đảm bảo rằng dữ liệu người dùng như email, số điện thoại, mật khẩu hoặc thông tin thanh toán không bị rò rỉ hoặc bị sử dụng sai mục đích. Các biện pháp thường dùng bao gồm mã hóa, ẩn danh dữ liệu và kiểm soát truy cập theo vai trò (RBAC).

**Đảm bảo chất lượng phần mềm (Software Quality Assurance – SQA):** Đảm bảo chất lượng phần mềm là tập hợp các hoạt động giúp kiểm soát và duy trì chất lượng trong suốt vòng đời phát triển phần mềm. Điều này bao gồm việc tuân thủ quy trình, thực hiện kiểm thử, kiểm tra mã nguồn, đánh giá hiệu năng và xác minh rằng sản phẩm đáp ứng yêu cầu người dùng về mặt chức năng và phi chức năng như bảo mật, hiệu suất và khả năng mở rộng.

Một số phương pháp giúp bảo mật và an toàn của phần mềm

* Xác thực và phân quyền: Đảm bảo người dùng chỉ truy cập được các chức năng tương ứng với vai trò.
* Mã hóa dữ liệu: Bảo vệ thông tin khi truyền và lưu trữ, đặc biệt là mật khẩu và thông tin cá nhân.
* Bảo vệ API: Giới hạn truy cập API bằng các cơ chế như token, kiểm soát CORS, rate limiting.
* Ngăn ngừa lỗ hổng phổ biến: Chống lại các tấn công như SQL Injection, XSS, CSRF thông qua việc kiểm tra đầu vào và cấu hình bảo mật đúng cách.
* Sao lưu và phục hồi: Đảm bảo dữ liệu có thể được khôi phục trong trường hợp xảy ra sự cố.
* Kiểm thử bảo mật: Kết hợp kiểm thử chức năng và phi chức năng để phát hiện lỗ hổng bảo mật từ sớm.

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Phân tích yêu cầu

Từ những nhu cầu thực tế về đăng ký các khóa học online cho thấy:

* Người học cần một nền tảng dễ sử dụng để tìm kiếm, đăng ký khóa học với giao diện thân thiện và khả năng tích hợp đăng nhập nhanh như Google.
* Người dạy cần công cụ quản lý khóa học và học viên hiệu quả, với khả năng tùy chỉnh thông tin khóa học và theo dõi danh sách học viên.
* Hệ thống cần đảm bảo bảo mật thông tin, hiệu suất tốt và khả năng tích hợp với các dịch vụ bên thứ ba.

### Các chức năng chính

Các chức năng của hệ thông được xác định như sau:

* Người học:
* Xem danh sách khóa học: Hiển thị danh sách khóa học trên trang chủ với thông tin chi tiết như tên khóa học, mô tả, ngày diễn ra, số lượng học viên tối đa, giá, và nơi tổ chức học tập.
* Tìm kiếm khóa học: Tìm kiếm theo tên khóa học hoặc từ khóa liên quan.
* Lọc khóa học: Lọc theo danh mục và giá khóa học (miễn phí và trả phí).
* Đăng ký/hủy đăng ký khóa học: Cho phép đăng ký hoặc hủy đăng ký khóa học, với kiểm tra ràng buộc như số lượng học viên tối đa và hạn đăng ký.
* Nhận thông báo: Hiển thị thông báo về các khóa học sắp diễn ra.
* Quản lý tài khoản: Đăng ký tài khoản, đăng nhập, chỉnh sửa thông tin cá nhân, khôi phục mật khẩu và đăng nhập qua Google OAuth.
* Người dạy:
* Quản lý khóa học: Tạo, chỉnh sửa, xóa khóa học với các thông tin như tên, mô tả, danh mục, ngày diễn ra, số lượng học viên tối đa, giá, nơi tổ chức,...
* Xem danh sách học viên: Xem danh sách học viên đã đăng ký cho từng khóa học.
* Quản lý danh mục khóa học: Tạo và lấy danh sách danh mục khóa học.
* Hệ thống:
* Quản lý dữ liệu: Lưu trữ thông tin người dùng, khóa học, danh mục, và đăng ký trong cơ sở dữ liệu MySQL.
* Giao tiếp API: Sử dụng RESTful API để kết nối frontend và backend, đảm bảo tính mô-đun và dễ mở rộng.

### Các yêu cầu phi chức năng

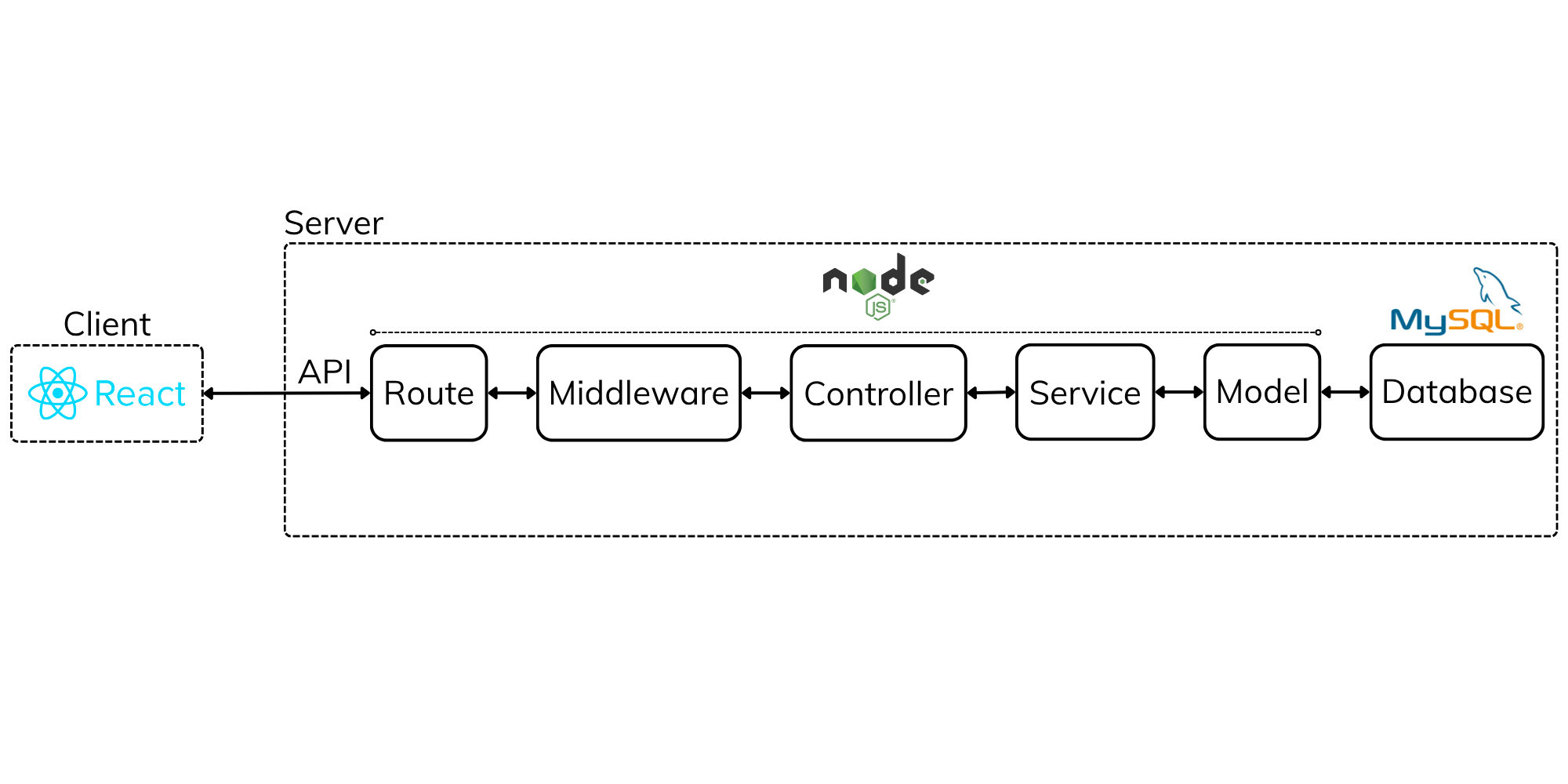
Để đảm bảo chất lượng, phần mềm cần đáp ứng cách tính chất sau:

* Hiệu suất: Hệ thống cần phản hồi nhanh chóng trong, đảm bảo trải nghiệm người dùng mượt mà với nhiều yêu cầu đồng thời.
* Bảo mật:
* Mã hóa dữ liệu: Sử dụng bcrypt để mã hóa mật khẩu người dùng, đảm bảo an toàn thông tin đăng nhập.
* Xác thực: Sử dụng JSON Web Token (JWT) để xác thực người dùng trong các yêu cầu API, đảm bảo chỉ người dùng hợp lệ mới truy cập được tài nguyên.
* Tính riêng tư: Đảm bảo thông tin cá nhân (email, số điện thoại) được lưu trữ an toàn và chỉ hiển thị với người dùng sở hữu hoặc có quyền liên quan.
* Khả năng mở rộng: Kiến trúc client-server cho phép dễ dàng mở rộng sang môi trường đám mây hoặc kiến trúc microservices..
* Tính dễ sử dụng: Giao diện người dùng được thiết kế trực quan, đảm bảo tính responsive trên nhiều thiết bị (desktop, tablet, mobile).
* Tính bảo trì:
* Sử dụng Git/GitHub để quản lý mã nguồn, đảm bảo lịch sử thay đổi được theo dõi rõ ràng.
* Tự động hóa kiểm thử bằng GitHub Actions để giảm thiểu lỗi trong quy trình phát triển.
* Tính nhất quán: Container hóa bằng Docker đảm bảo môi trường phát triển và thử nghiệm giống nhau, tránh lỗi do khác biệt cấu hình.

## Thiết kế hệ thống

### Kiến trúc tổng thể

Hệ thống được thiết kế theo mô hình client-server, với các thành phần chính là frontend và backend. Sau đây là sơ đồ kiến trúc tổng thể:



Hình 2.1. Sơ đồ kiến trúc tổng thể

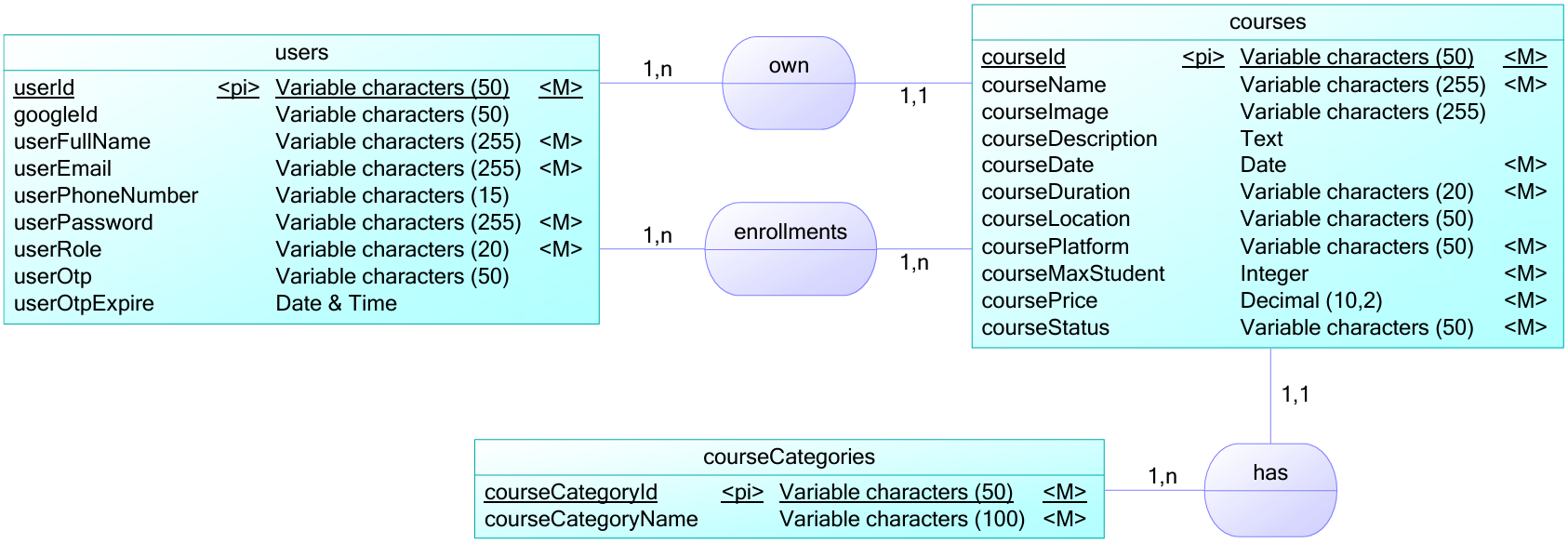
Mô tả chức năng trong kiến trúc:

* Client (ReactJS):
* Chức năng: Hiển thị giao diện người dùng, xử lý tương tác người dùng và gửi yêu cầu API đến backend.
* Công nghệ: ReactJS sử dụng Styled Component để xây dựng giao diện, dùng Axios để gọi API từ backend.
* Server (Node.js/Express):
* Chức năng: Xử lý logic nghiệp vụ (xác thực, quản lý khóa học, đăng ký), tương tác với cơ sở dữ liệu và trả về dữ liệu qua RESTful API.
* Công nghệ: Express để định tuyến API, bcrypt để mã hóa mật khẩu, JWT để xác thực, MySQL driver để truy vấn cơ sở dữ liệu.
* Cơ sở dữ liệu (MySQL):
* Chức năng: Lưu trữ thông tin người dùng, khóa học, danh mục và đăng ký.
* Công nghệ: MySQL với mô hình quan hệ, sử dụng khóa chính, khóa ngoại và các ràng buộc để đảm bảo toàn vẹn dữ liệu.
* Dịch vụ bên thứ ba (Google OAuth):
* Chức năng: Hỗ trợ đăng nhập qua Google, giảm thời gian xác thực cho người dùng.
* Công nghệ: Google API để xác thực và lấy thông tin người dùng.

### Thiết kế cơ sở dữ liệu

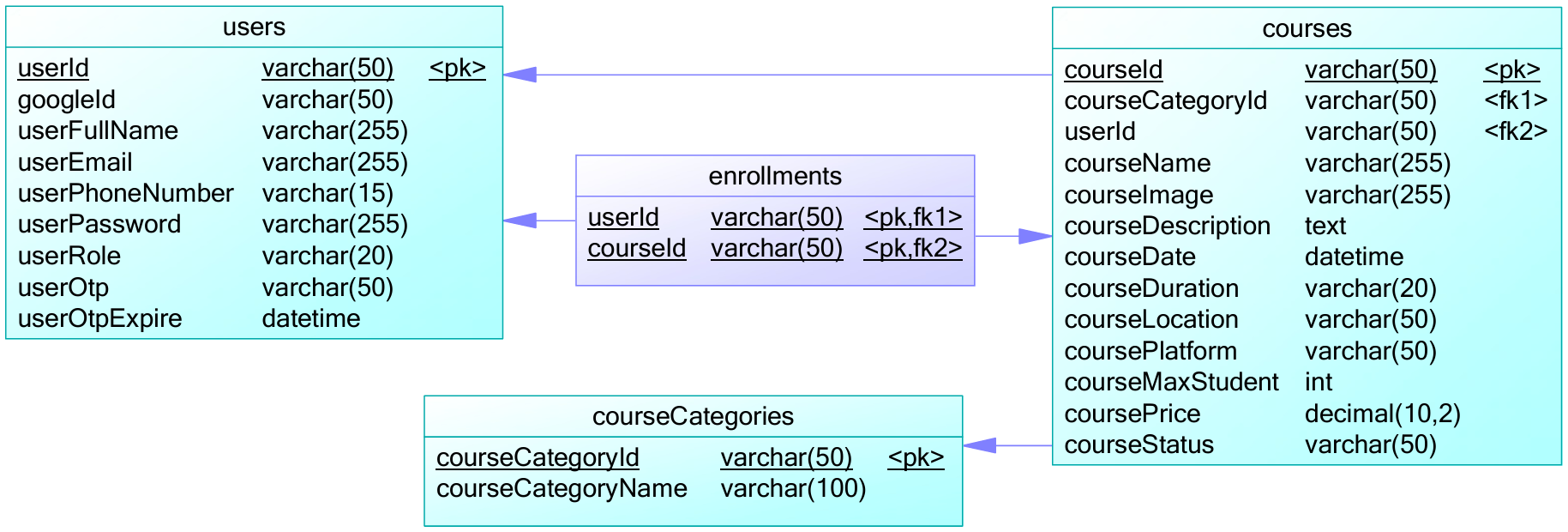
Hệ thống sử dụng cơ sở dữ liệu quan hệ MySQL để lưu trữ dữ liệu.

Mô hình thực thể kết hợp:



Hình 2.2. Mô hình thực thể kết hợp

Mô hình dữ liệu quan hệ:



Hình 2.3. Mô hình dữ liệu quan hệ

Mô tả các bảng dữ liệu:

* Bảng **users**

Bảng 2.1. Mô tả bảng users

| **STT** | **Tên tắt** | **Diễn giải** | **Loại giá trị** | **Kiểu dữ liệu** | **Miền giá trị** | **Chiều dài** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | userId | Mã người dùng | Bắt buộc | Varchar | Khóa chính | 50 |
| 2 | googleId | ID Google liên kết |  | Varchar |  | 50 |
| 3 | userFullName | Họ và tên người dùng | Bắt buộc | Varchar |  | 255 |
| 4 | userEmail | Email người dùng | Bắt buộc | Varchar |  | 255 |
| 5 | userPhoneNumber | Số điện thoại |  | Varchar |  | 15 |
| 6 | userPassword | Mật khẩu đã mã hóa | Bắt buộc | Varchar |  | 255 |
| 7 | userRole | Vai trò người dùng | Bắt buộc | Varchar |  | 20 |
| 8 | userOtp | Mã OTP xác thực |  | Varchar |  | 50 |
| 9 | userOtpExpire | Hạn hiệu lực OTP |  | Datetime |  |  |

* **Bảng courseCategories**

Bảng 2.2. Mô tả bảng **courseCategories**

| **STT** | **Tên tắt** | **Diễn giải** | **Loại giá trị** | **Kiểu dữ liệu** | **Miền giá trị** | **Chiều dài** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | courseCategoryId | Mã danh mục khóa học | Bắt buộc | Varchar | Khóa chính | 50 |
| 2 | courseCategoryName | Tên danh mục | Bắt buộc | Varchar |  | 100 |

* **Bảng courses**

Bảng 2.3. Mô tả bảng courses

| **STT** | **Tên tắt** | **Diễn giải** | **Loại giá trị** | **Kiểu dữ liệu** | **Miền giá trị** | **Chiều dài** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | courseId | Mã khóa học |  | Varchar | Khóa chính | 50 |
| 2 | userId | Mã người tạo khóa học |  | Varchar | Khóa ngoại | 50 |
| 3 | courseCategoryId | Mã danh mục khóa học |  | Varchar | Khóa ngoại | 50 |
| 4 | courseName | Tên khóa học | Bắt buộc | Varchar |  | 255 |
| 5 | courseImage | Ảnh đại diện khóa học |  | Varchar |  | 255 |
| 6 | courseDescription | Mô tả chi tiết |  | Text |  | - |
| 7 | courseDate | Ngày khai giảng |  | Date |  | - |
| 8 | courseDuration | Thời lượng |  | Varchar |  | 20 |
| 9 | courseLocation | Địa điểm học |  | Varchar |  | 50 |
| 10 | coursePlatform | Nền tảng học |  | Varchar |  | 50 |
| 11 | courseMaxStudent | Số học viên tối đa |  | Smallint |  | - |
| 12 | coursePrice | Học phí |  | Decimal |  | 10 |
| 13 | courseStatus | Trạng thái khóa học | Bắt buộc | Varchar |  | 50 |

* **Bảng enrollments**

Bảng 2.4. Mô tả bảng **enrollments**

| **STT** | **Tên tắt** | **Diễn giải** | **Loại giá trị** | **Kiểu dữ liệu** | **Miền giá trị** | **Chiều dài** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | userId | Mã người học | Bắt buộc | Varchar | Khóa chính, khóa ngoại | 50 |
| 2 | courseId | Mã khóa học đăng ký | Bắt buộc | Varchar | Khóa chính, khóa ngoại | 50 |

### Thiết kế API

Sau đây là mô tả các endpoint chính, cấu trúc request/response của API:

* Nhóm API xác thực bao gồm đăng ký, đăng nhập, quên mật khẩu và đặt lại mật khẩu.

Endpoint đăng ký tài khoản mới (POST /api/auth/register):

Request Body

{

"userFullName": "Nguyễn Văn A",

"userEmail": "user@example.com",

"userPassword": "matkhau123",

"userPhoneNumber": "0123456789",

"userRole": "Student"

}

Responses

201

{

"message": "Đăng ký thành công",

"user": {}

}

400

{

"message": "Email đã được sử dụng"

}

Endpoint đăng nhập (POST /api/auth/login):

Request Body

{

"userEmail": "user@example.com",

"userPassword": "matkhau123"

}

Responses

200

{

"message": "Đăng nhập thành công",

"token": "jwt\_token",

"user": {}

}

400

{

"message": "Email hoặc mật khẩu không đúng"

}

Endpoint yêu cầu đặt lại mật khẩu (POST /api/auth/forgot-password):

Request Body

{

"userEmail": "user@example.com"

}

Responses

200

{

"message": "Đã gửi email đặt lại mật khẩu"

}

400

{

"message": "Email không tồn tại"

}

Endpoint đặt lại mật khẩu (POST /api/auth/reset-password):

Request Body

{

"token": "reset\_token",

"newPassword": "matkhaumoi123"

}

Responses

200

{

"message": "Đặt lại mật khẩu thành công"

}

400

{

"message": "Yêu cầu đặt lại mật khẩu không hợp lệ hoặc đã hết hiệu lực"

}

* Nhóm API khóa học bao gồm danh mục khóa học, tạo, cập nhật, xóa, đăng ký và hủy đăng ký khóa học. Tất cả các endpoint yêu cầu xác thực JWT trù GET /api/course và GET /api/course/:courseId (hỗ trợ truy cập khách).

Endpoint tạo danh mục khóa học (POST /api/course/category):

Request Body

{

"courseCategoryName": "Lập trình"

}

Responses

201

{

"message": "Tạo danh mục thành công",

"category": {}

}

400

{

"message": "Danh mục đã tồn tại"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint lấy tất cả danh mục khóa học (GET /api/course/category)

Responses:

200

{

"categories": [

{}

]

}

Endpoint tạo khóa học mới (POST /api/course)

Request Body

{

"courseCategoryId": "null",

"courseName": "Lập trình NodeJS",

"courseImage": "nodejs.png",

"courseDescription": "Khóa học thực hành NodeJS",

"courseDate": "2025-08-01",

"courseDuration": "08:00 - 10:00",

"courseLocation": "https://zoom.us/j/123456789",

"coursePlatform": "Zoom",

"courseMaxStudent": 30,

"coursePrice": 1500000,

"courseStatus": "Draft"

}

Responses

201

{

"message": "Tạo khóa học thành công",

"course": {}

}

400

{

"message": "Tên khóa học đã tồn tại"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint: Lấy tất cả khóa học (GET /api/course)

Responses:

200

{

"courses": [

{

"isEnrolled": false

}

]

}

401

{

"message": "Thiếu hoặc sai token xác thực"

}

Endpoint lấy thông tin một khóa học (GET /api/course/:courseId)

Parameters: courseId

Responses

200

{

"course": {

"isEnrolled": false

}

}

401

{

"message": "Thiếu hoặc sai token xác thực"

}

404

{

"message": "Không tìm thấy khóa học hoặc không đủ quyền"

}

Endpoint cập nhật khóa học (PUT /api/course/:courseId)

Parameters: courseId

Request Body

{

"courseCategoryId": "null",

"courseName": "Lập trình NodeJS",

"courseImage": "nodejs.png",

"courseDescription": "Khóa học thực hành NodeJS",

"courseDate": "2025-08-01",

"courseDuration": "08:00 - 10:00",

"courseLocation": "https://zoom.us/j/123456789",

"coursePlatform": "Zoom",

"courseMaxStudent": 30,

"coursePrice": 1500000,

"courseStatus": "Draft"

}

Responses

200

{

"message": "Cập nhật khóa học thành công",

"course": {}

}

400

{

"message": "Không có thay đổi nào được cập nhật"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint: Đăng ký khóa học (POST /api/course/:courseId/enroll)

Parameters: courseId

Responses

200

{

"message": "Đăng ký khóa học thành công"

}

```

400

{

"message": "Bạn đã đăng ký khóa học này hoặc khóa học đã bắt đầu"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint hủy đăng ký khóa học (DELETE /api/course/:courseId/enroll)

Parameters: courseId

Responses

200

{

"message": "Hủy đăng ký khóa học thành công"

}

400

{

"message": "Bạn chưa đăng ký khóa học này hoặc khóa học đã bắt đầu"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint: Xóa khóa học (DELETE /api/course/:courseId)

Parameters: courseId

Responses

200

{

"message": "Xóa khóa học thành công"

}

400

{

"message": "Không thể xóa khóa học này"

}

403

{

"message": "Không có quyền truy cập"

}

Endpoint lấy danh sách học viên bằng id khóa học (POST /api/course/:courseId/students)

Parameters: courseId

Responses

200

{

"students": [

{

"userFullName": "string",

"userEmail": "string",

"userPhoneNumber": "string"

}

]

}

500

{

"message": "Lỗi server"

}

403

{

"message": "Không có quyền truy cập"

}

* Nhóm API người dùng bao gồm lấy và cập nhật thông tin.

Endpoint lấy thông tin người dùng (GET /api/user/:userId):

Parameters: userId

Responses

200

{

"user": {}

}

401

{

"message": "Thiếu hoặc sai token xác thực"

}

403

{

"message": "Không có quyền truy cập"

}

404

{

"message": "Không tìm thấy user"

}

Endpoint cập nhật thông tin người dùng (PUT /api/user/:userId)

Parameters: userId

Request Body

{

"userFullName": "Nguyễn Văn A",

"userPhoneNumber": "0123456789"

}

Responses

200

{

"message": "Cập nhật thành công",

"user": {}

}

400

{

"message": "Không có thay đổi nào được cập nhật"

}

401

{

"message": "Thiếu hoặc sai token xác thực"

}

403

{

"message": "Không có quyền truy cập"

}

404

{

"message": "Không tìm thấy user"

}

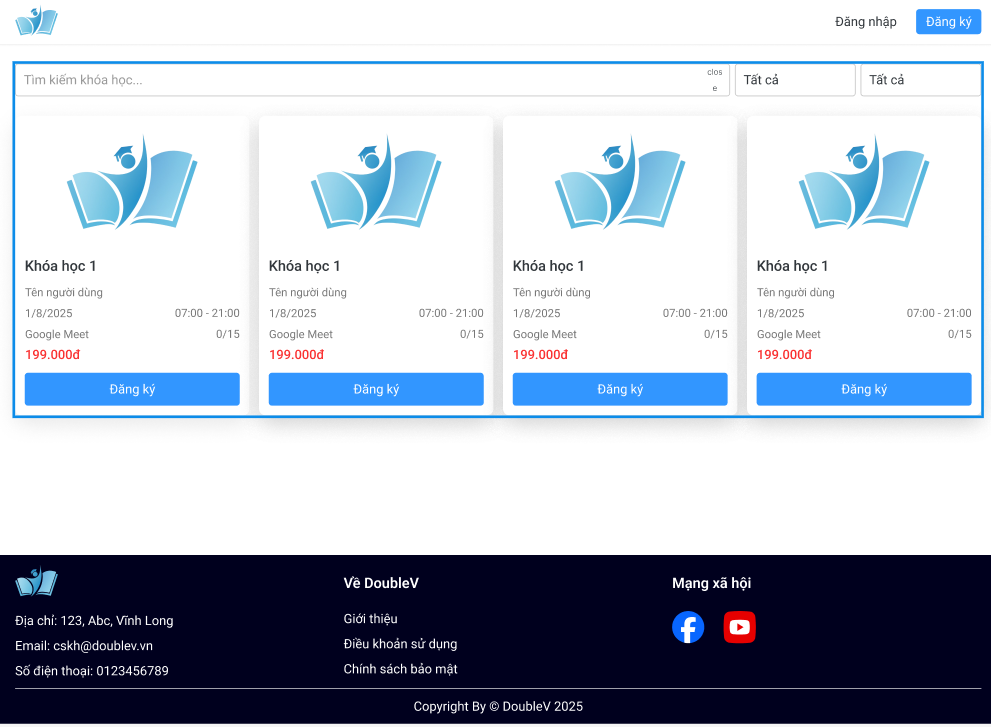
### Thiết kế giao diện

Giao diện được thiết kế trên Figma, với các tiêu chí:

* Tính trực quan: Sử dụng bố cục rõ ràng, phông chữ dễ đọc và màu sắc hài hòa.
* Tính responsive: Đảm bảo giao diện hoạt động tốt trên desktop, tablet và mobile.
* Tính tương tác: Các nút bấm và chức năng như bộ lọc được thiết kế để dễ sử dụng, với hiệu ứng hover và loading để tăng trải nghiệm.

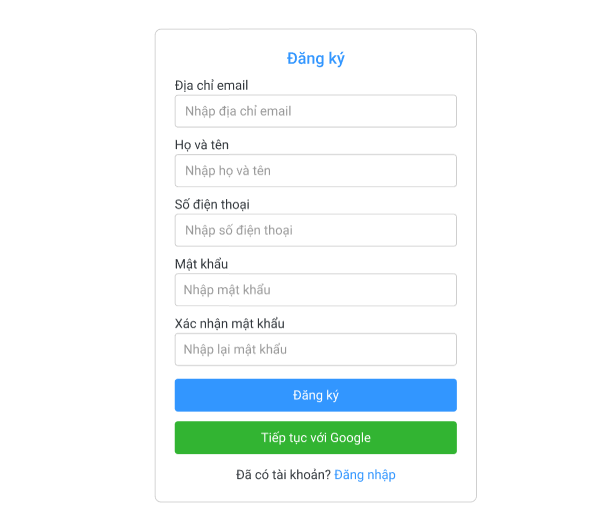
Sau đây là các giao diện chính đã được thiết kế:

* Giao diện trang chủ chưa đăng nhập



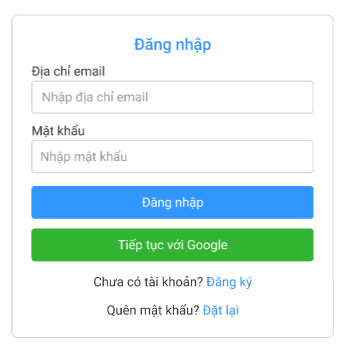
Hình 2.4. Giao diện trang chủ chưa đăng nhập

* Giao diện đăng ký



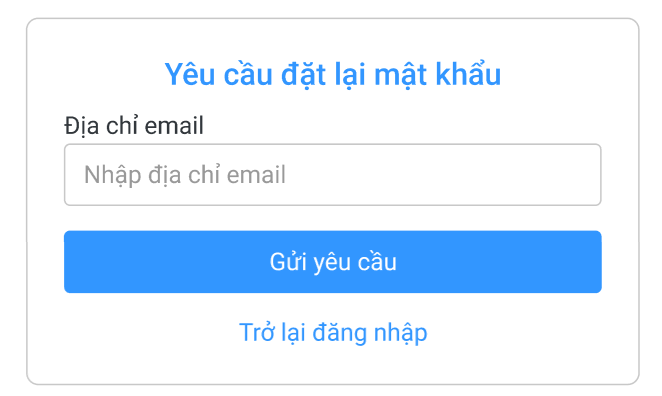
Hình 2.5. Giao diện đăng ký

* Giao diện đăng nhập



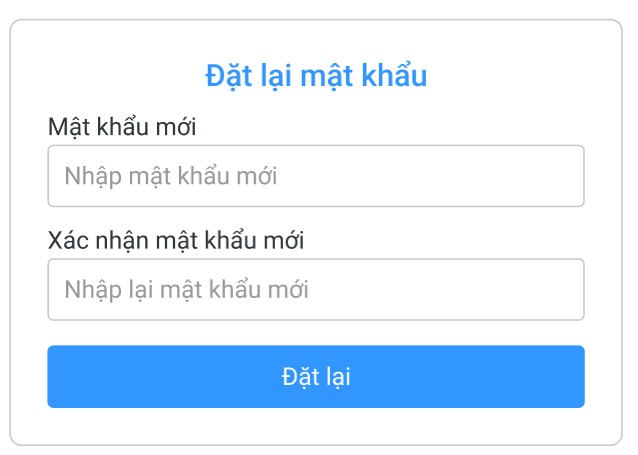
Hình 2.6. Giao diện đăng nhập

* Giao diện yêu cầu đặt lại mật khẩu



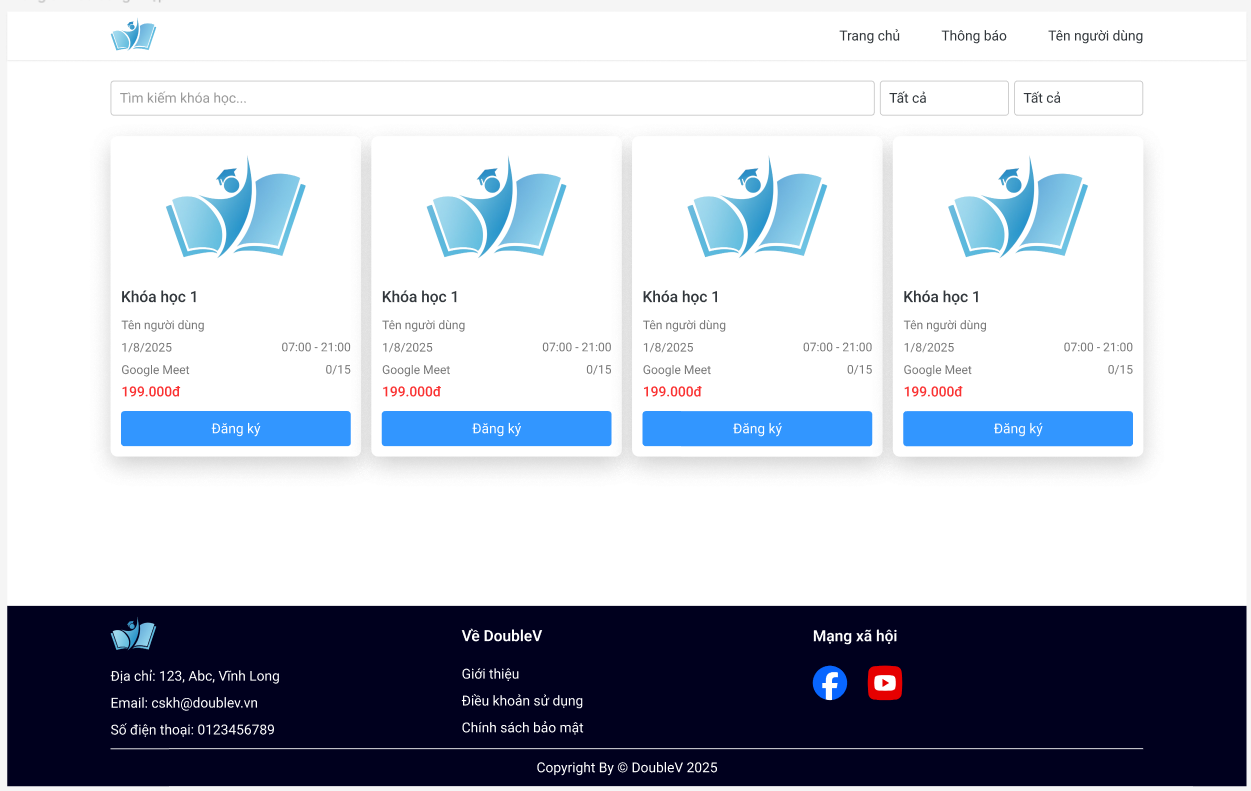
Hình 2.7. Giao diện yêu cầu đặt lại mật khẩu

* Giao diện đặt mật khẩu mới



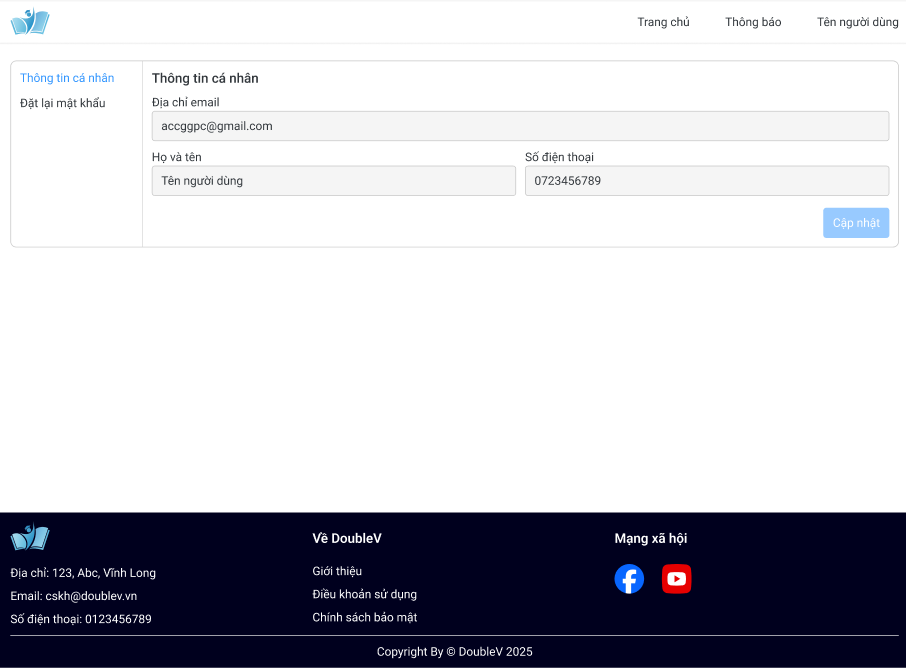
Hình 2.8. Giao diện đặt mật khẩu mới

* Giao diện trang chủ đã đăng nhập



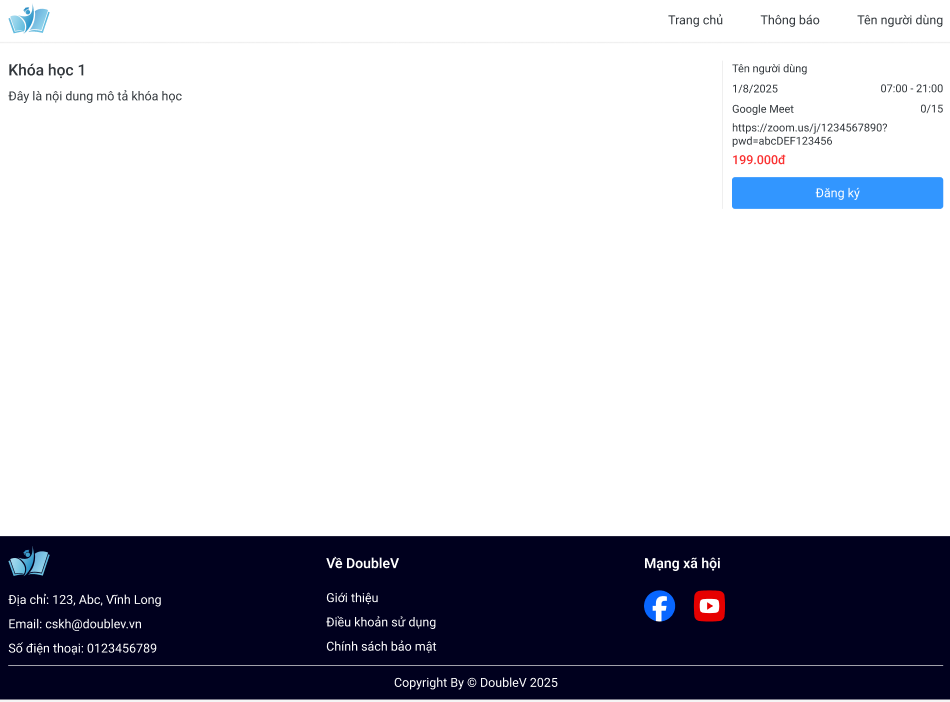
Hình 2.9. Giao diện trang chủ đã đăng nhập

* Giao diện xem thông tin cá nhân



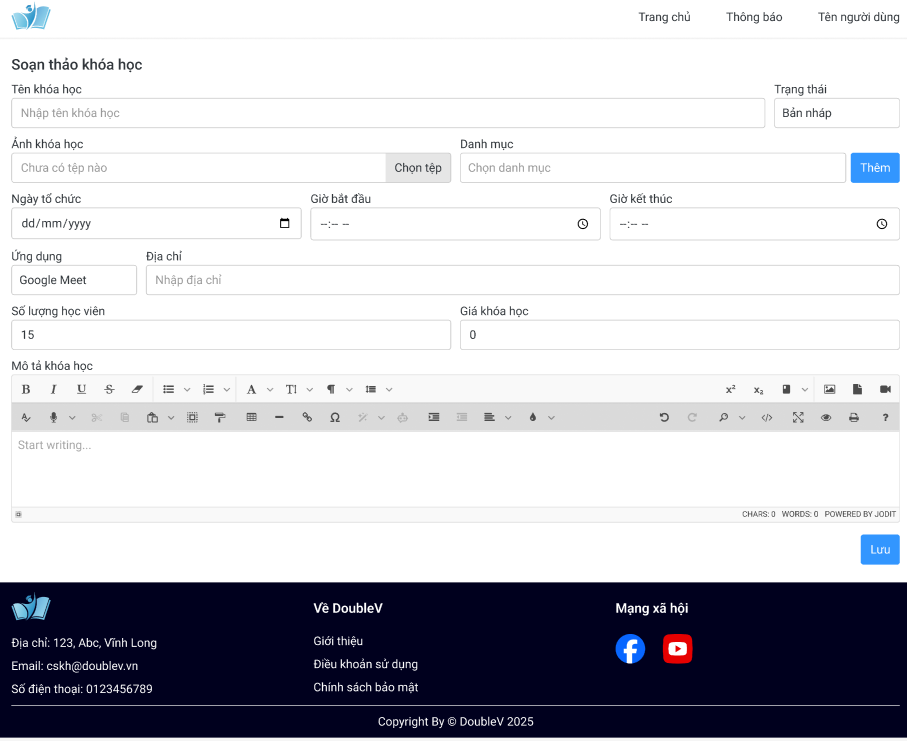
Hình 2.10. Giao diện xem thông tin cá nhân

* Giao diện xem thông tin khóa học



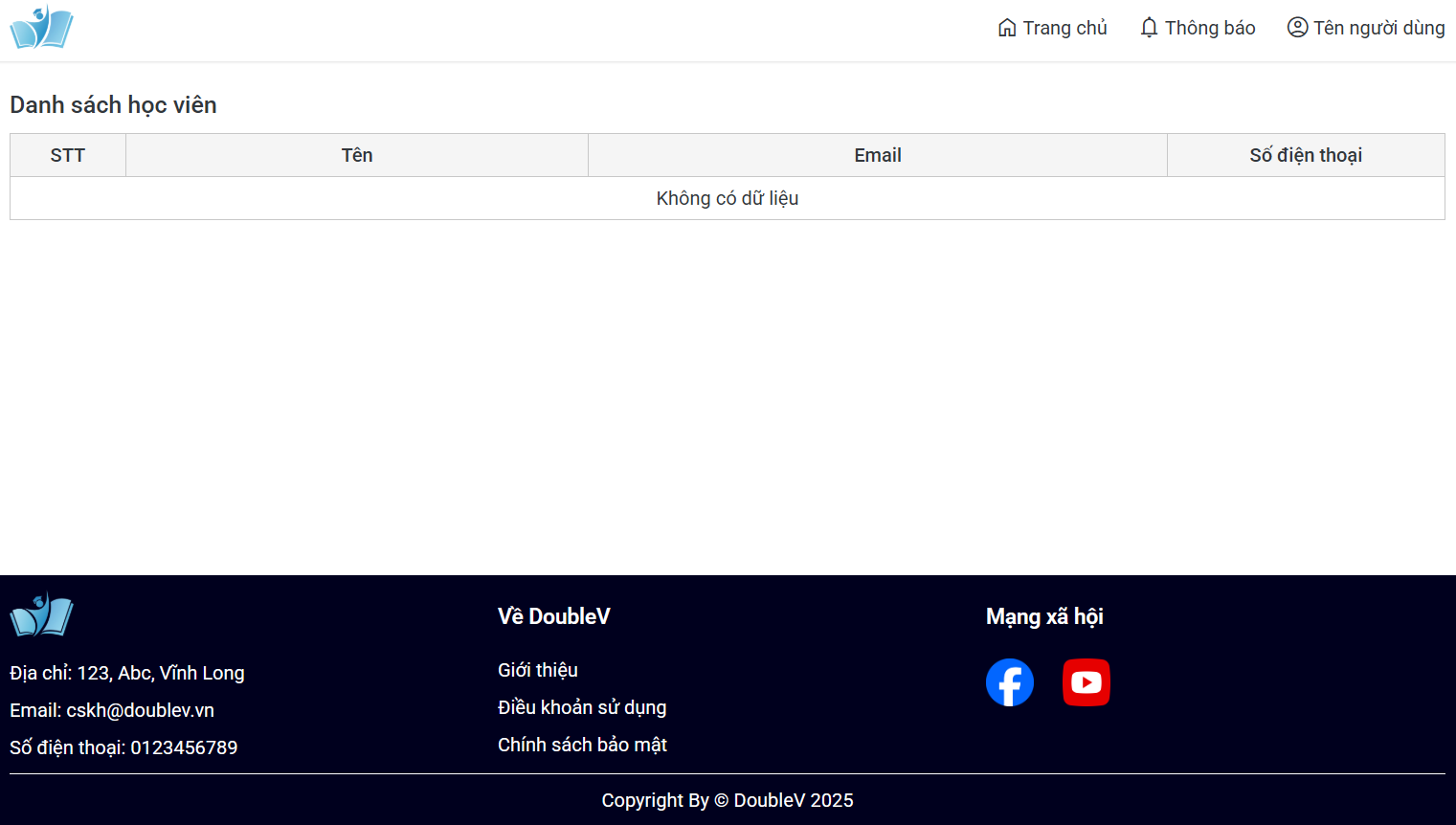
Hình 2.11. Giao diện xem thông tin khóa học

* Giao diện soạn thảo khóa học



Hình 2.12. Giao diện soạn thảo khóa học

* Giao diện xem danh sách học viên đã đăng ký



Hình 2.13. Giao diện xem các học viên đã đăng ký

# TRIỂN KHAI DỰ ÁN

## Các công nghệ đã sử dụng

Dự án sử dụng các công nghệ phổ biến như sau:

* Frontend:
* ReactJS: Framework JavaScript để xây dựng giao diện động, sử dụng component-based architecture để tái sử dụng mã và tăng hiệu quả phát triển.
* Axios: Thư viện để gọi API từ frontend, hỗ trợ xử lý HTTP requests và quản lý lỗi.
* Backend:
* Node.js: Môi trường chạy JavaScript phía server, nhẹ và hiệu quả cho các ứng dụng thời gian thực.
* Express: Framework để xây dựng RESTful API, xử lý định tuyến và middleware.
* MySQL: Cơ sở dữ liệu quan hệ để lưu trữ thông tin người dùng, khóa học, và đăng ký.
* Công cụ quản lý mã nguồn: dùng GitHub để quản lý mã nguồn, theo dõi lịch sử thay đổi và hỗ trợ làm việc nhóm.
* Kiểm thử:
* Postman: Kiểm thử API thủ công, kiểm tra request/response và mã trạng thái.
* Swagger: Tài liệu hóa API, cung cấp giao diện để kiểm tra endpoint.
* Tự động hóa: dùng GitHub Actions tự động hóa quy trình CI/CD (kiểm thử, đóng gói).
* Container hóa:
* Docker: Container hóa ứng dụng và cơ sở dữ liệu để đảm bảo tính nhất quán.
* Docker Compose: Quản lý nhiều container (frontend, backend, MySQL).
* Quản lý dự án: dùng Jira lập kế hoạch và theo dõi tiến độ theo mô hình Agile, quản lý Product Backlog và Sprint Backlog.
* Thiết kế giao diện: dùng Figma để thiết kế giao diện, hỗ trợ làm việc nhóm và chia sẻ thiết kế.

## Quy trình CI/CD với GitHub Actions

Quy trình CI/CD của dự án được thiết lập để tự động hóa các bước sau:

* Kiểm thử: Chạy unit test để đảm bảo mã nguồn không có lỗi.
* Build: Xây dựng Docker Image cho frontend và backend, đảm bảo ứng dụng được đóng gói đúng cách.
* Đóng gói: Đóng gói ứng dụng thành container để đảm bảo tính nhất quán giữa các môi trường.

Cấu hình GitHub Actions

name: CI/CD Pipeline

on:

  push:

    branches:

      - main

  pull\_request:

    branches:

      - main

jobs:

  test:

    runs-on: ubuntu-latest

    steps:

      - name: Checkout code

        uses: actions/checkout@v4

      - name: Setup Node.js

        uses: actions/setup-node@v4

        with:

          node-version: "18"

      - name: Install backend dependencies

        run: |

          cd course\_enrollment/backend

          npm ci

      - name: Run backend unit tests

        run: |

          cd course\_enrollment/backend

          npm test

        env:

          CI: true

  docker:

    needs: test

    if: github.event\_name == 'push' && github.ref == 'refs/heads/main'

    runs-on: ubuntu-latest

    steps:

      - name: Checkout code

        uses: actions/checkout@v4

      - name: Login to Docker Hub

        run: echo "${{ secrets.DOCKER\_ACCESS\_TOKEN }}" | docker login -u "${{ secrets.DOCKER\_USERNAME }}" --password-stdin

      - name: Build & Push Backend Image

        run: |

          docker build -t ${{ secrets.DOCKER\_USERNAME }}/course-enrollment-backend:latest -f course\_enrollment/backend/Dockerfile course\_enrollment/backend

          docker push ${{ secrets.DOCKER\_USERNAME }}/course-enrollment-backend:latest

      - name: Build & Push Frontend Image

        run: |

          docker build -t ${{ secrets.DOCKER\_USERNAME }}/course-enrollment-frontend:latest -f course\_enrollment/frontend/Dockerfile course\_enrollment/frontend

          docker push ${{ secrets.DOCKER\_USERNAME }}/course-enrollment-frontend:latest

## Cấu hình Docker

Docker được sử dụng để container hóa ứng dụng, đảm bảo tính nhất quán giữa môi trường phát triển và thử nghiệm. Các file cấu hình bao gồm:

* Dockerfile cho backend:

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install --production

COPY . .

EXPOSE 5000

CMD ["node", "src/start.js"]

* Dockerfile cho frontend:

FROM node:18-alpine AS build

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

RUN npm run build

FROM node:18-alpine

WORKDIR /app

COPY --from=build /app/dist ./dist

COPY package\*.json ./

RUN npm install -g serve

EXPOSE 3000

CMD ["serve", "-s", "dist", "-l", "3000"]

* Docker Compose:

version: "3.8"

services:

  mysql:

    image: mysql:8.0

    restart: always

    environment:

      MYSQL\_ROOT\_PASSWORD: root

      MYSQL\_DATABASE: course\_enrollment

      MYSQL\_USER: admin

      MYSQL\_PASSWORD: password

    ports:

      - "3306:3306"

    volumes:

      - mysql\_data:/var/lib/mysql

      - ./course\_enrollment/backend/init.sql:/docker-entrypoint-initdb.d/init.sql

  adminer:

    image: adminer

    restart: always

    ports:

      - "8080:8080"

    depends\_on:

      - mysql

  backend:

    image: pthevinh/course-enrollment-backend:latest

    restart: always

    environment:

      DB\_HOST: mysql

      DB\_USER: admin

      DB\_PASSWORD: password

      DB\_NAME: course\_enrollment

      DB\_PORT: 3306

      JWT\_SECRET: 3e05a1f9-86ff-4171-bfaf-b37aa2b92570

    ports:

      - "5000:5000"

    depends\_on:

      - mysql

  frontend:

    image: pthevinh/course-enrollment-frontend:latest

    restart: always

    ports:

      - "3000:3000"

    depends\_on:

      - backend

volumes:

  mysql\_data:

# QUẢN LÝ DỰ ÁN

## Mô hình phát triển Agile

Dự án được phát triển theo mô hình Agile với framework Scrum vì:

* Tính lặp lại: Mỗi sprint kéo dài 2 tuần, cho phép nhóm phát triển và kiểm thử các chức năng theo từng giai đoạn nhỏ.
* Tập trung vào giá trị: Ưu tiên các chức năng cốt lõi (đăng ký, quản lý khóa học) để đảm bảo sản phẩm đáp ứng nhu cầu người dùng.
* Hợp tác nhóm: Các buổi họp sprint planning, daily stand-up, sprint review, và sprint retrospective giúp tăng cường giao tiếp và phối hợp.

Các hoạt động chính trong Scrum bao gồm:

* Sprint Planning: Xác định các user stories sẽ thực hiện trong sprint, phân chia thành các task cụ thể và ước lượng thời gian.
* Daily Stand-up: Họp hàng ngày để cập nhật tiến độ, thảo luận khó khăn, và điều chỉnh kế hoạch.
* Sprint Review: Đánh giá sản phẩm sau mỗi sprint, trình bày các chức năng hoàn thành và thu thập phản hồi.
* Sprint Retrospective: Rút ra bài học kinh nghiệm và đề xuất cải thiện cho sprint tiếp theo.

Jira được sử dụng để quản lý Product Backlog, Sprint Backlog, và theo dõi tiến độ. Mỗi user story được gắn với các task cụ thể, có điểm ưu tiên (story points) và thời gian ước lượng.

## Phân công nhiệm vụ

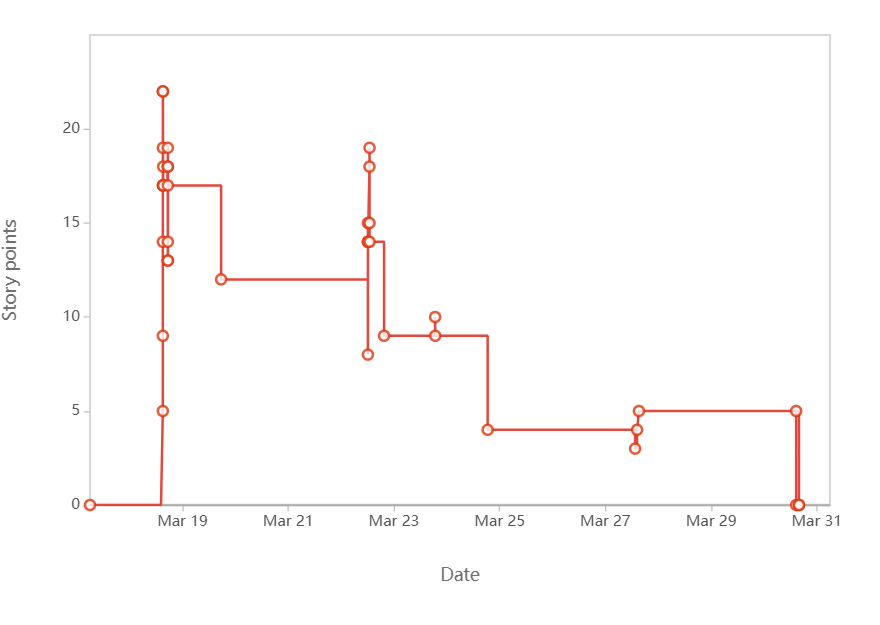
Dự án được thưc thực hiện với 4 Sprint. Sau đây là bảng phân công nhiệm vục cho từng thành viên ở các Sprint:

Bảng 4.1. Bảng phân công nhiệm vụ ở các Sprint

|  |  |  |
| --- | --- | --- |
| **Sprint 1** | | |
| **Nhiệm vụ** | **Thành viên** | **Story points** |
| Phân tích yêu cầu, thiết kế sơ đồ kiến trúc | Nguyễn Phúc Vinh | 4 |
| Thiết kế mô hình dữ liệu, viết script MySQL | Phạm Thế Vinh | 4 |
| Chức năng đăng ký | Nguyễn Phúc Vinh | 5 |
| Khởi tạo dự án, viết unit test, kiểm thử và build Docker Image bằng Github Actions | Phạm Thế Vinh | 5 |
| **Sprint 2** | | |
| **Nhiệm vụ** | **Thành viên** | **Story points** |
| Chức năng đăng nhập, trang chủ cơ bản | Nguyễn Phúc Vinh | 4 |
| Xem, sửa thông tin cá nhân | Phạm Thế Vinh | 4 |
| Chức năng đặt lại khi quên mật khẩu | Nguyễn Phúc Vinh | 4 |
| Chức năng tiếp tục với Google | Phạm Thế Vinh | 3 |
| **Sprint 3** | | |
| **Nhiệm vụ** | **Thành viên** | **Story points** |
| Tạo, xem khóa học | Phạm Thế Vinh | 4 |
| Xem danh sách, sửa, xóa khóa học | Nguyễn Phúc Vinh | 5 |
| Xem danh sách học viên đã đăng ký | Phạm Thế Vinh | 4 |
| Đăng ký/hủy đăng ký khóa học | Nguyễn Phúc Vinh | 3 |
| **Sprint 4** | | |
| **Nhiệm vụ** | **Thành viên** | **Story points** |
| Tìm kiếm và lọc khóa học | Nguyễn Phúc Vinh | 3 |
| Bổ sung các ràng buộc thông tin khóa học | Phạm Thế Vinh | 3 |
| Thông báo khóa học sắp diễn ra | Nguyễn Phúc Vinh | 3 |
| Tạo và lấy các danh mục khóa học | Phạm Thế Vinh | 3 |

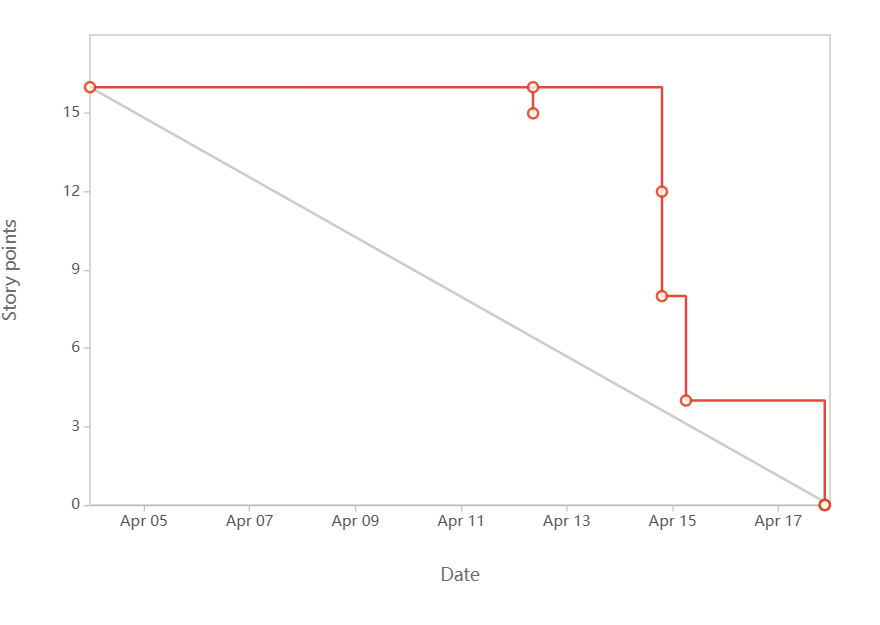
Sau đây là Burndown chart theo Story point của các Sprint đã thực hiện:

* Burndown chart của Sprint 1



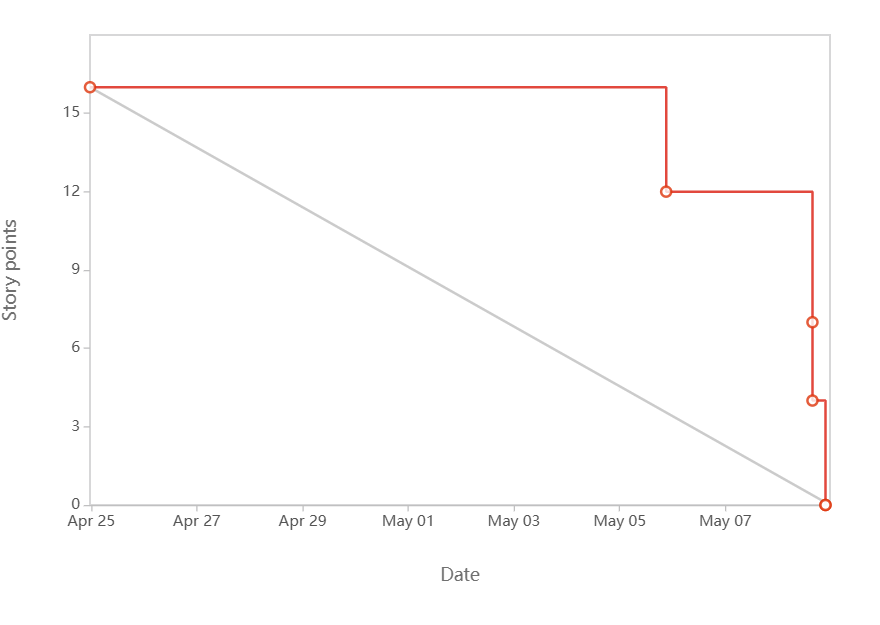
Hình 4.1. Burndown chart của Sprint 1

* Burndown chart của Sprint 2



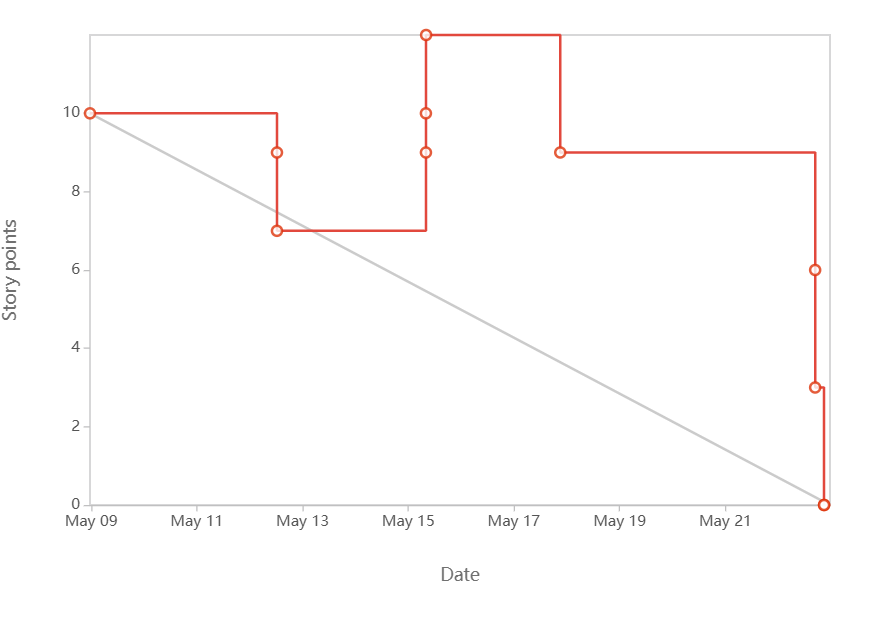
Hình 4.2. Burndown chart của Sprint 2

* Burndown chart của Sprint 3



Hình 4.3. Burndown chart của Sprint 3

* Burndown chart của Sprint 4



Hình 4.4. Burndown chart của Sprint 4

# KIỂM THỬ

## ****Chiến lược kiểm thử****

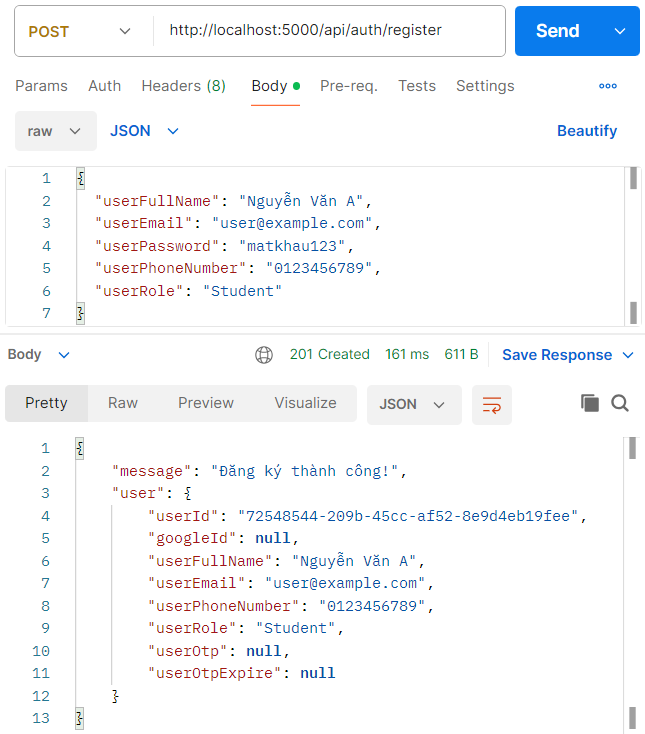
Chiến lược kiểm thử trong dự án được xây dựng dựa trên hướng tiếp cận theo từng lớp chức năng. Thay vì kiểm thử toàn bộ hệ thống một cách tích hợp từ đầu, các chức năng được chia nhỏ để kiểm tra độc lập, từ cấp độ đơn vị đến các điểm tương tác bên ngoài. Cách tiếp cận này giúp phát hiện sớm các vấn đề logic, đảm bảo từng thành phần nhỏ đều hoạt động đúng trước khi đưa vào tổng thể.

Để hỗ trợ quá trình kiểm thử hiệu quả, một số công cụ hiện đại đã được tích hợp vào quy trình phát triển phần mềm:

* **Postman:** là công cụ kiểm thử API phổ biến, Postman được sử dụng để gửi các yêu cầu mô phỏng đến hệ thống và kiểm tra phản hồi trả về. Giao diện trực quan của Postman hỗ trợ xây dựng, lưu trữ và tổ chức các bộ kiểm thử, phục vụ cả kiểm thử thủ công lẫn bán tự động.
* **Unit Test:** các kiểm thử đơn vị được viết để xác minh tính đúng đắn của các hàm xử lý logic nội bộ liên quan đến luồng API. Điều này giúp phát hiện lỗi sớm ngay trong quá trình phát triển và tăng mức độ tin cậy cho từng chức năng nhỏ.
* **GitHub Actions**: đóng vai trò tự động hóa quá trình kiểm thử mỗi khi có thay đổi mã nguồn. Với tích hợp kiểm thử vào quy trình CI, các kịch bản API test hoặc unit test sẽ được thực thi liên tục nhằm đảm bảo chất lượng và độ ổn định của hệ thống qua từng lần cập nhật.

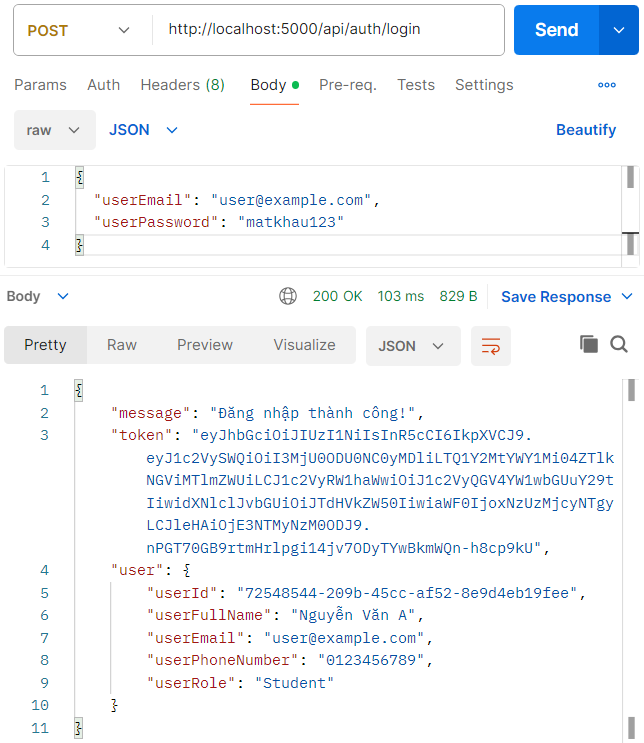
## Kết quả kiểm thử API với Postman

API đăng ký tài khoản (POST /api/auth/register):



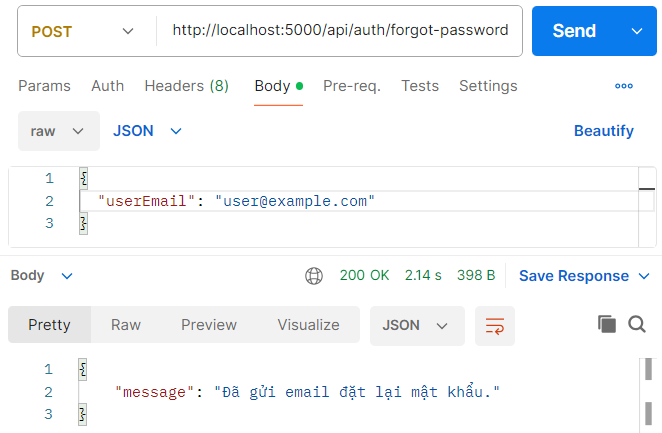
Hình 5.1. Kết quả kiểm thử API đăng ký tài khoản

API đăng nhập (POST /api/auth/login):



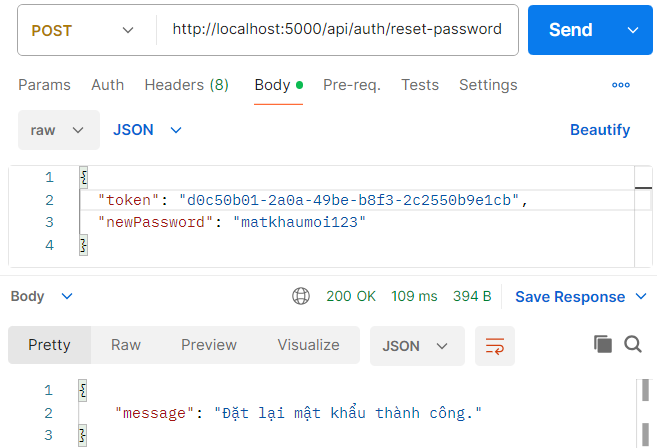
Hình 5.2, Kết quả kiểm thử API đăng nhập

API yêu cầu đặt lại mật khẩu (POST /api/auth/forgot-password)



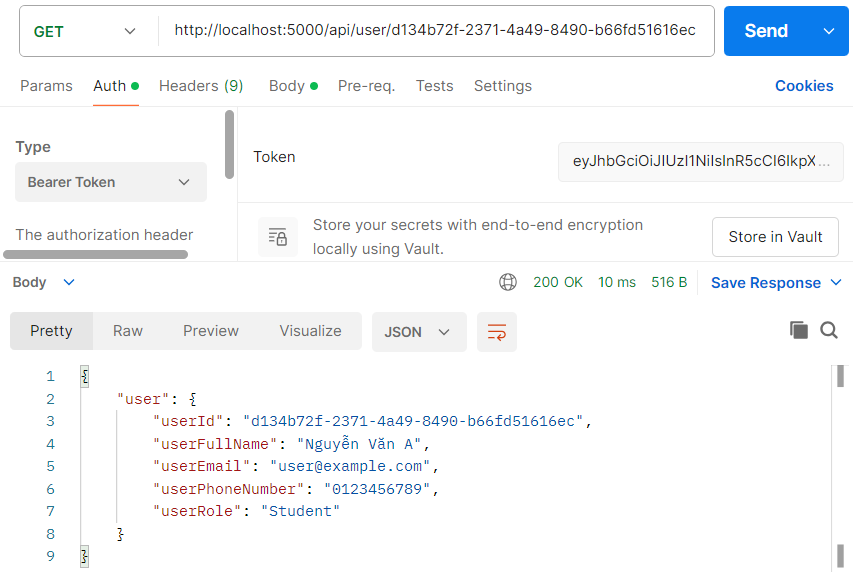
Hình 5.3. Kết quả kiểm thử API yêu cầu đặt lại mật khẩu

API đặt lại mật khẩu (POST /api/auth/reset-password):



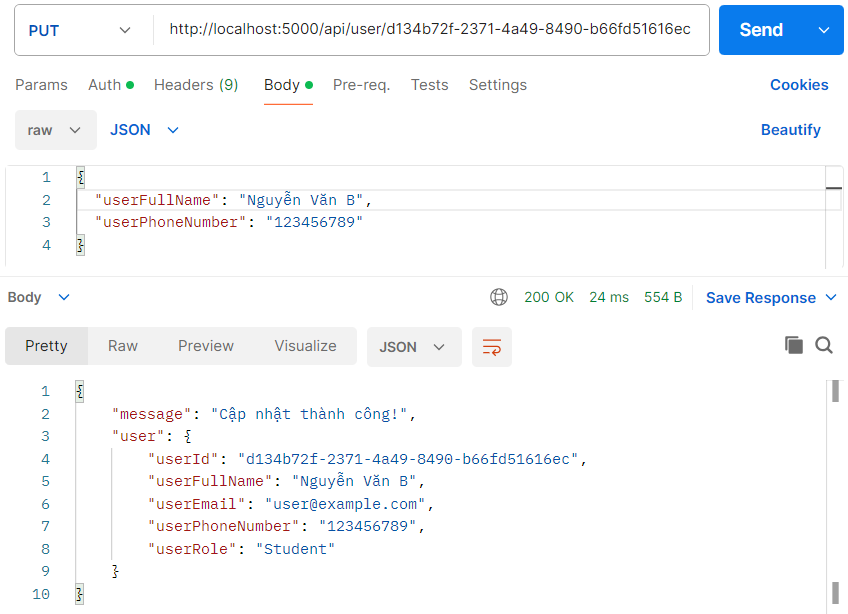
Hình 5.4. Kết quả kiểm thử API đặt lại mật khẩu

API lấy một người dùng bằng id (GET /api/user/{userId}):



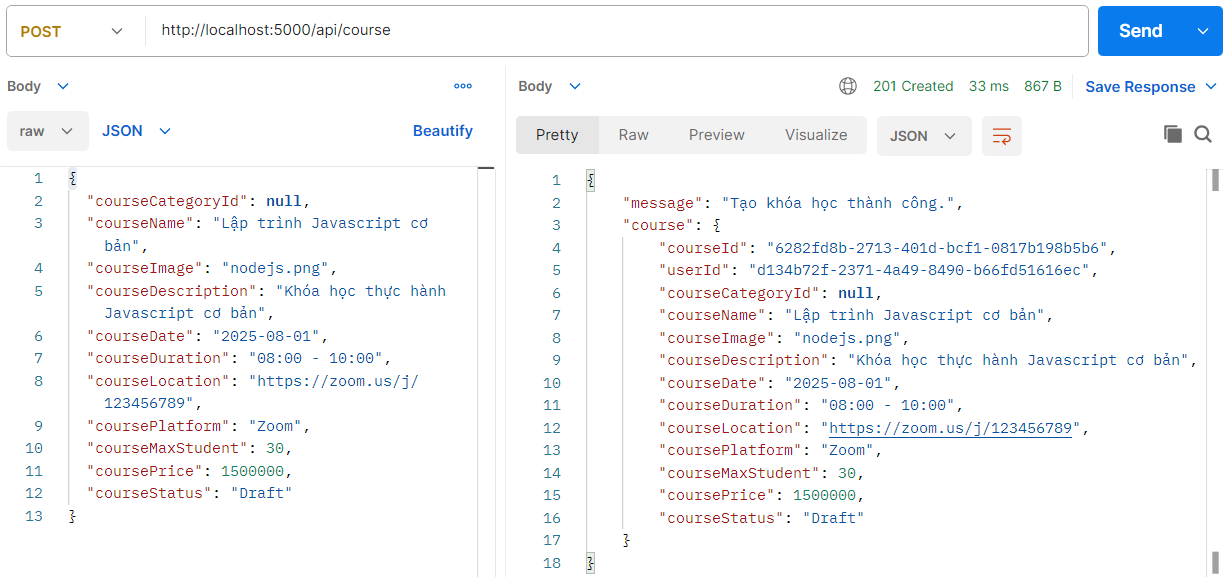
Hình 5.5. Kết quả kiểm thử API lấy một người dùng bằng id

API cập nhật thông tin người dùng bằng id (PUT /api/user/{userId}):



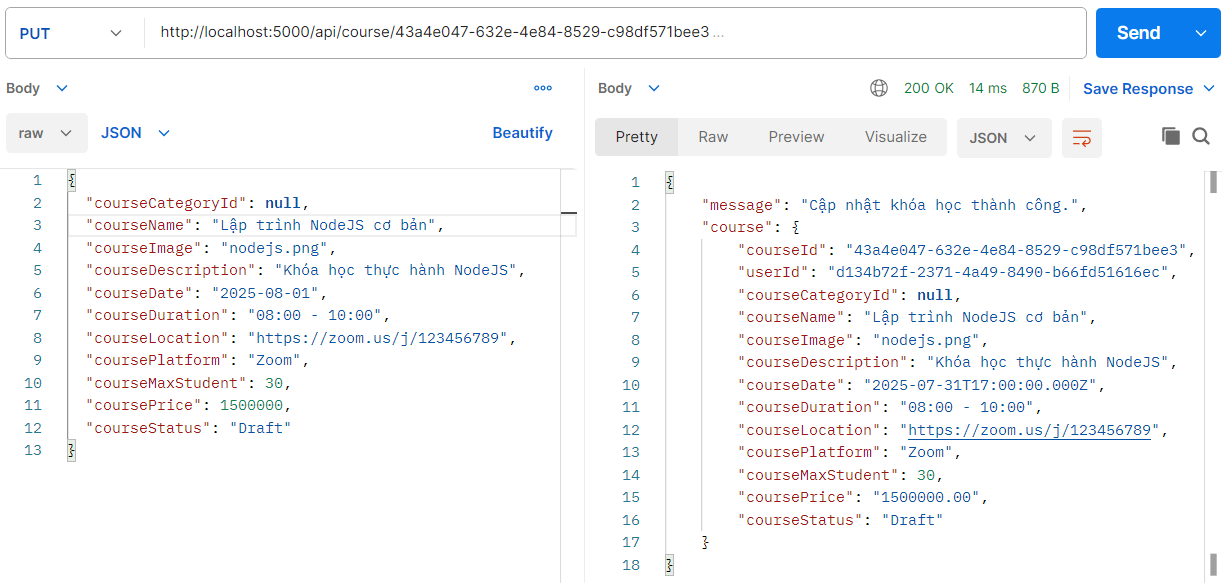
Hình 5.6. Kết quả kiểm thử API cập nhật thông tin người dùng bằng id

API tạo khóa học (POST /api/course):



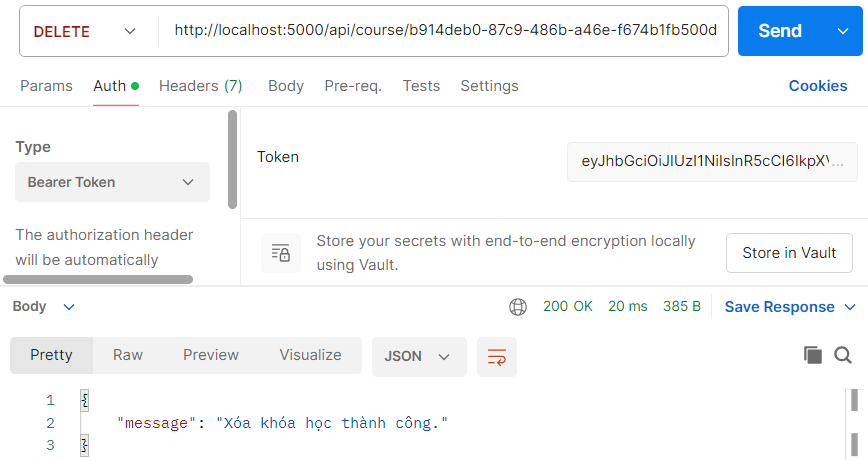
Hình 5.7. Kết quả kiểm thử API tạo khóa học

API cập nhật khóa học bằng id (PUT /api/course/{courseId}):



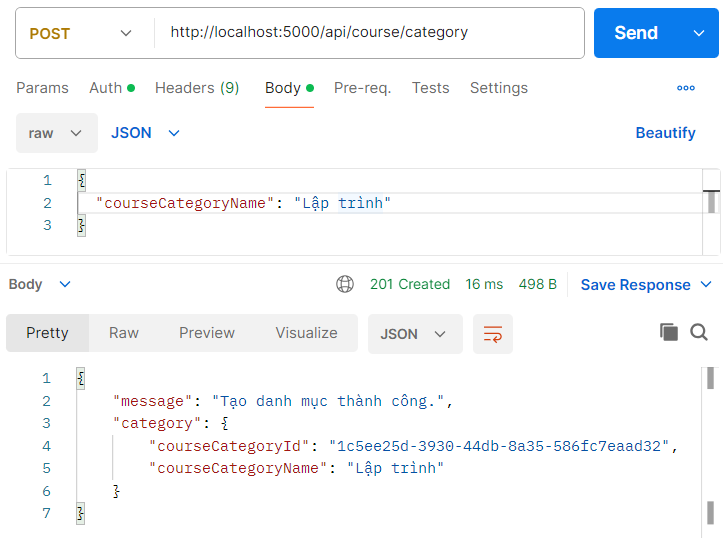
Hình 5.8. Kết quả kiểm thử API cập nhật khóa học bằng id

API xóa khóa học bằng id (DELETE /api/course/{courseId}):



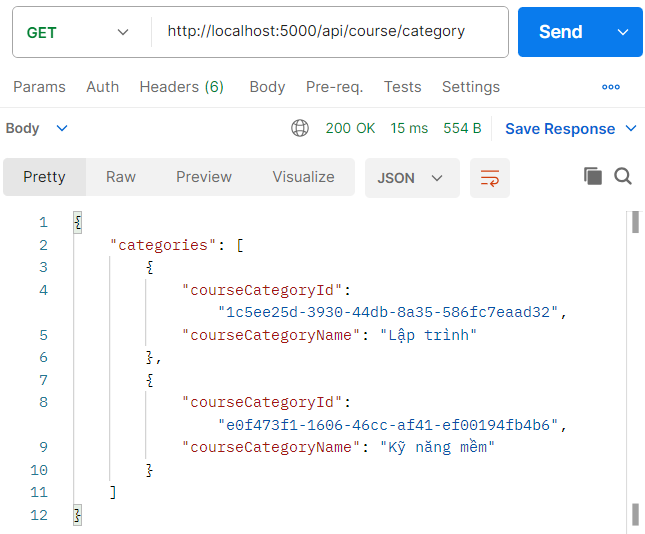
Hình 5.9. Kết quả kiểm thử API xóa khóa học bằng id

API tạo danh mục khóa học (POST /api/course/category):



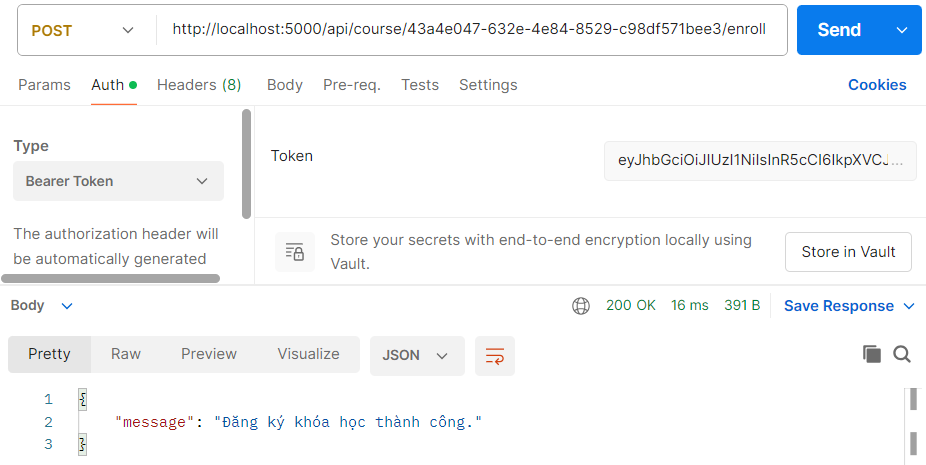
Hình 5.10. Kết quả kiểm thử API tạo danh mục khóa học

API lấy các danh mục khóa học (GET /api/course/category):



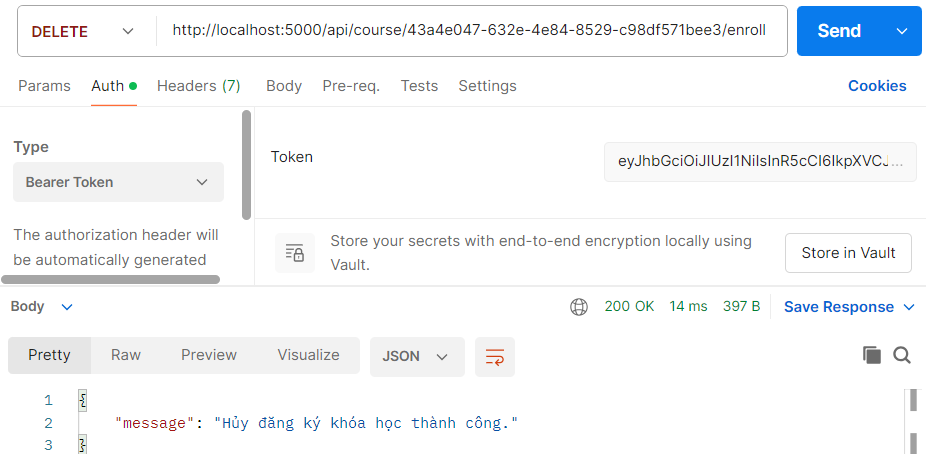
Hình 5.11. Kết quả kiểm thử API lấy các danh mục khóa học

API đăng ký khóa học bằng id (POST /api/course/{courseId}/enroll):



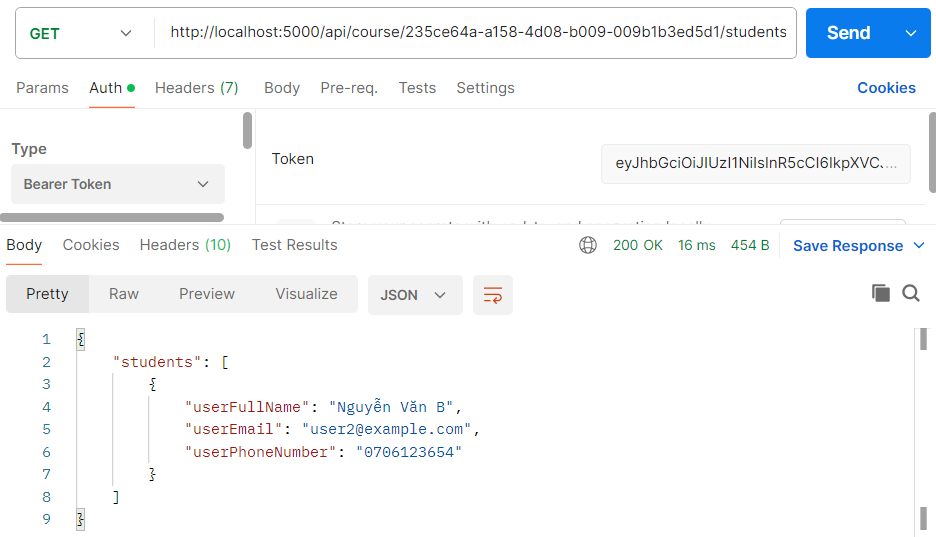
Hình 5.12. Kết quả kiểm thử API đăng ký khóa học bằng id

API hủy đăng ký khóa học bằng id (DELETE /api/course/{courseId}/enroll):



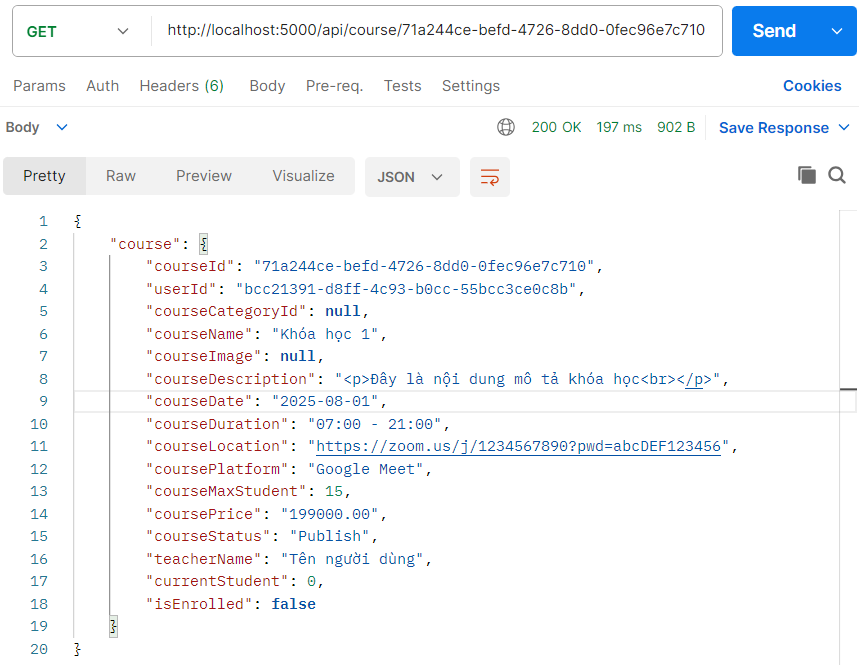
Hình 5.13. Kết quả kiểm thử API hủy đăng ký khóa học bằng id

API lấy danh sách học viên đăng ký bằng id khóa học (GET /api/course/{courseId}/students):



Hình 5.14. Kết quả kiểm thử API lấy danh sách học viên đăng ký bằng id khóa học

API lấy một khóa học bằng id (GET /api/course/{courseId}):



Hình 5.15. Kết quả kiểm thử API lấy một khóa học bằng id

# ĐÁNH GIÁ VÀ KẾT LUẬN

## Khó khăn gặp phải

* Thiết kế giao diện responsive: Đảm bảo giao diện hoạt động tốt trên nhiều thiết bị (desktop, tablet, mobile) yêu cầu nhiều lần điều chỉnh.
* Cấu hình CI/CD: Gặp lỗi khi tích hợp GitHub Actions với Docker do sai cấu hình biến môi trường hoặc quyền truy cập.
* Kiểm thử API: Một số endpoint trả về lỗi không mong muốn.

## Bài học rút ra

* Tài liệu hóa API: Sử dụng Swagger để tài liệu hóa API giúp việc kiểm thử và bảo trì dễ dàng hơn, đặc biệt khi nhóm có nhiều thành viên.
* Quy trình Agile: Sử dụng Jira và mô hình Agile giúp nhóm phân chia công việc rõ ràng, theo dõi tiến độ hiệu quả và nhanh chóng điều chỉnh khi có thay đổi yêu cầu.
* Container hóa: Docker đảm bảo tính nhất quán giữa các môi trường, giúp giảm lỗi do khác biệt cấu hình.
* Kiểm thử sớm: Viết unit test từ đầu giúp phát hiện lỗi sớm, tiết kiệm thời gian sửa lỗi ở các giai đoạn sau.
* Bảo mật: Áp dụng các biện pháp như mã hóa mật khẩu, JWT và kiểm tra đầu vào giúp nhóm hiểu rõ hơn về tầm quan trọng của bảo mật trong phát triển phần mềm.

## Đề xuất cải thiện

* Tích hợp thanh toán trực tuyến để hỗ trợ thanh toán khóa học.
* Thêm tính năng đánh giá và bình luận khóa học để người học cung cấp phản hồi.
* Phát triển ứng dụng mobile để mở rộng trải nghiệm người dùng.
* Thêm chức năng lịch học cá nhân để người học theo dõi lịch trình.
* Triển khai hệ thống trên nền tảng đám mây (AWS, Google Cloud) để hỗ trợ nhiều người dùng hơn và tăng tính sẵn sàng.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Research and Markets. (2023). *Global Online Education Market 2023-2028*. [Online]. Available at: <https://www.researchandmarkets.com/>
2. Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Doctoral dissertation, University of California, Irvine. [Online]. Available at: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
3. Fowler, M. (2014). *Microservices*. [Online]. Available at: <https://martinfowler.com/articles/microservices.html>
4. Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide*. [Online]. Available at: <https://scrumguides.org/>
5. Sommerville, I. (2015). *Software Engineering* (10th ed.). Pearson Education. ISBN: 978-0133943030.
6. Pressman, R. S., & Maxim, B. R. (2014). *Software Engineering: A Practitioner's Approach* (8th ed.). McGraw-Hill Education. ISBN: 978-0078022128.

# PHỤ LỤC

**Hướng dẫn cài đặt và chạy ứng dụng**

1. Clone dự án từ GitHub
2. Khởi động hệ thống

Tại thư mục course\_enrollment chạy lệnh docker compose up --build

1. Truy cập hệ thống

* Frontend: [http://localhost:3000](http://localhost:3000/)
* Backend API: [http://localhost:5000](http://localhost:5000/)
* Adminer (quản lý MySQL): [http://localhost:8080](http://localhost:8080/)

1. Dừng hệ thống

Chạy lệnh docker compose down

**Liên kết GitHub repository**

*https://github.com/phamthevinhtv/cong-nghe-phan-mem.git*