KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN KẾT THÚC MÔN**

**CÔNG NGHỆ PHẦN MỀM**

**HỌC KỲ II, NĂM HỌC 2024 – 2025**

**XÂY DỰNG ỨNG DỤNG**

**HỌC TỪ VỰNG TIẾNG ANH**

*Giáo viên hướng dẫn:*TS. Nguyễn Bảo Ân

*Sinh viên thực hiện:*

Tiêu Anh Huy – 110122007

Dương Bảo Khanh – 110122009

Nguyễn Huỳnh Yến Như – 110122018

***Trà Vinh, tháng 07 năm 2025***

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN KẾT THÚC MÔN**

**CÔNG NGHỆ PHẦN MỀM**

**HỌC KỲ II, NĂM HỌC 2024 – 2025**

**XÂY DỰNG ỨNG DỤNG**

**HỌC TỪ VỰNG TIẾNG ANH**



*Sinh viên thực hiện:*

Tiêu Anh Huy – 110122007

Dương Bảo Khanh – 110122009

Nguyễn Huỳnh Yến Như – 110122018

***Trà Vinh, tháng 07 năm 2025***

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*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)*

**MỤC LỤC**

[TỔNG QUAN 6](#_Toc204140801)

[CHƯƠNG 1: GIỚI THIỆU 8](#_Toc204140802)

[1.1 Giới thiệu về công nghệ phần mềm 8](#_Toc204140803)

[1.2 Giới thiệu về công nghệ phần mềm Agile 9](#_Toc204140804)

[1.2.1 Giới thiệu về Agile 9](#_Toc204140805)

[1.2.2 Giới thiệu về Scrum 9](#_Toc204140806)

[1.3 Giới thiệu về Microservices 10](#_Toc204140807)

[1.4 Tính an toàn, tính riêng tư và bảo đảm chất lượng phần mềm 11](#_Toc204140808)

[1.4.1 Tính an toàn 11](#_Toc204140809)

[1.4.2 Tính riêng tư 12](#_Toc204140810)

[1.4.3 Đảm bảo chất lượng phần mềm 13](#_Toc204140811)

[CHƯƠNG 2: PHÂN TÍCH YÊU CẦU 15](#_Toc204140812)

[2.1 Chức năng chính của hệ thống 15](#_Toc204140813)

[2.1.1 Chức năng trang Home 15](#_Toc204140814)

[2.1.2 Chức năng tìm kiếm 15](#_Toc204140815)

[2.1.3 Chức năng học 15](#_Toc204140816)

[2.1.4 Chức năng kiểm tra 16](#_Toc204140817)

[2.1.5 Chức năng quản lý tài khoản người dùng 16](#_Toc204140818)

[2.2 Yêu cầu phi chức năng 16](#_Toc204140819)

[CHƯƠNG 3: THIẾT KẾ HỆ THỐNG 18](#_Toc204140820)

[3.1 Kiến trúc hệ thống 18](#_Toc204140821)

[3.1.1 Kiến trúc 18](#_Toc204140822)

[3.1.2 Luồng hoạt động 19](#_Toc204140823)

[3.1.3 Lợi ích của Microservice 19](#_Toc204140824)

[3.2 Thiết kế cơ sở dữ liệu 20](#_Toc204140825)

[3.2.1 Cơ sở dữ liệu user-service 20](#_Toc204140826)

[3.2.2 Cơ sở dữ liệu topic-service 20](#_Toc204140827)

[3.2.3 Cơ sở dữ liệu test-service 21](#_Toc204140828)

[3.3 Thiết kế giao diện 22](#_Toc204140829)

[3.3.1 Thiết kế Figma 22](#_Toc204140830)

[3.3.2 Thiết kế giao diện với React 24](#_Toc204140831)

[CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ 27](#_Toc204140832)

[4.1 Danh sách các công nghệ đã sử dụng 27](#_Toc204140833)

[4.2 Quy trình CI/CD với GitHub Actions. 27](#_Toc204140834)

[4.3 Cấu hình Docker và quy trình triển khai ứng dụng. 28](#_Toc204140835)

[4.3.1 Cấu hình Docker 28](#_Toc204140836)

[4.3.2 Quy trình triển khai ứng dụng trên Docker 29](#_Toc204140837)

[CHƯƠNG 5: QUẢN LÝ DỰ ÁN 33](#_Toc204140838)

[CHƯƠNG 6: KIỂM THỬ 35](#_Toc204140839)

[6.1 Kiểm thử unit test 35](#_Toc204140840)

[6.2 Kiểm thử API với Postman: 36](#_Toc204140841)

[CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN 38](#_Toc204140842)

[DANH MỤC TÀI LIỆU THAM KHẢO 39](#_Toc204140843)

**DANH MỤC HÌNH ẢNH**

[Hình 3.1 Sơ đồ kiến trúc hệ thống 18](#_Toc204139783)

[Hình 3.2 User 20](#_Toc204139784)

[Hình 3.3 topic-service 20](#_Toc204139785)

[Hình 3.4 test service 21](#_Toc204139786)

[Hình 3.5 Giao diện Figma trang chủ 22](#_Toc204139787)

[Hình 3.6 Giao diện Figma trang chủ đề từ vựng 22](#_Toc204139788)

[Hình 3.7 Giao diện Figma trang học từ vựng 23](#_Toc204139789)

[Hình 3.8 Giao diện Figma trang các bài test 23](#_Toc204139790)

[Hình 3.9 Giao diện Figma trang bài test 24](#_Toc204139791)

[Hình 3.10 Giao diện trang chủ 24](#_Toc204139792)

[Hình 3.11 Giao diện trang chủ đề 25](#_Toc204139793)

[Hình 3.12 Giao diện các bài kiểm tra 25](#_Toc204139794)

[Hình 3.13 Giao diện học từ vựng 25](#_Toc204139795)

[Hình 3.14 Giao diện làm bài 26](#_Toc204139796)

[Hình 3.15 Giao diện hoàn thành bài kiểm tra 26](#_Toc204139797)

[Hình 3.16 Giao diện kết quả tìm kiếm 26](#_Toc204139798)

[Hình 5.1 SCRUM Sprint 1 33](#_Toc204139799)

[Hình 5.2 SCRUM Sprint 2 33](#_Toc204139800)

[Hình 5.3 SCRUM Sprint 3 33](#_Toc204139801)

[Hình 5.4 SCRUM Sprint 4 34](#_Toc204139802)

[Hình 5.5 SCRUM Sprint 5 34](#_Toc204139803)

[Hình 5.6 SCRUM Sprint 6 34](#_Toc204139804)

[Hình 6.1 Kiểm API đăng kí 36](#_Toc204139805)

[Hình 6.3 Lấy API đăng nhập 37](#_Toc204139806)

# TỔNG QUAN

**Lý do chọn đề tài**

Trong thời đại toàn cầu hóa, tiếng Anh đã trở thành ngôn ngữ phổ biến và đóng vai trò quan trọng trong học tập, công việc cũng như giao tiếp quốc tế. Trong quá trình học tiếng Anh, từ vựng là nền tảng cốt lõi để phát triển các kỹ năng ngôn ngữ như nghe, nói, đọc và viết. Do đó, việc học từ vựng một cách hiệu quả là nhu cầu thiết yếu đối với học sinh, sinh viên và người đi làm.

Với sự phát triển của công nghệ thông tin, hình thức học trực tuyến ngày càng trở nên phổ biến nhờ tính tiện lợi và linh hoạt. Người học chỉ cần đăng nhập tài khoản trên các nền tảng học tập để dễ dàng học tiếng Anh ngay tại nhà, tiết kiệm thời gian và chủ động hơn trong việc ôn luyện.

Xuất phát từ thực tiễn đó, nhóm chúng em đã lựa chọn và triển khai đề tài “Xây dựng ứng dụng web học từ vựng tiếng Anh” nhằm hỗ trợ người học tiếp cận từ vựng một cách trực quan, hiệu quả và thân thiện với người dùng.

**Mục tiêu**

Mục tiêu của đề tài nhằm:

- Xây dựng một phần mềm hỗ trợ học từ vựng tiếng Anh hiệu quả, với giao diện thân thiện và dễ sử dụng, phù hợp với nhiều đối tượng người học.

- Cung cấp các hình thức học đa dạng như Flashcard, bài trắc nghiệm,... giúp người học tiếp cận từ vựng một cách linh hoạt và ghi nhớ lâu hơn.

- Phân loại từ vựng theo từng chủ đề cụ thể (Family, Food, Animal,...), cho phép người dùng lựa chọn chủ đề phù hợp với nhu cầu học tập cá nhân.

- Áp dụng các kỹ thuật lập trình để lưu trữ và quản lý thông tin tài khoản người dùng, cũng như theo dõi tiến độ học tập, hỗ trợ việc học có hệ thống và liên tục.

**Đối tượng nghiên cứu**

- Người học tiếng Anh ở nhiều trình độ, đặc biệt là học sinh, sinh viên và người mới bắt đầu học từ vựng tiếng Anh, có nhu cầu mở rộng vốn từ một cách có hệ thống và hiệu quả.

- Phương pháp học từ vựng bằng ứng dụng, các từ vựng được phân loại thành nhiều chủ đề và mỗi từ vựng được học với flash card, cùng với các bài tập trắc nghiệm để người dùng có thể ôn tập lại từ vựng đã học.

- Công nghệ và kỹ thuật lập trình, cụ thể là các công nghệ liên quan đến phát triển giao diện người dùng, xử lý dữ liệu từ vựng, lưu trữ thông tin tài khoản trong cơ sở dữ liệu.

**Phạm vi nghiên cứu**

Về nội dung:

Phần mềm tập trung vào hỗ trợ học từ vựng tiếng Anh theo từng chủ đề thông dụng như: Gia đình, Động vật, Thời tiết, Màu sắc, Đồ vật,... Nội dung từ vựng ở mức cơ bản đến trung cấp, phù hợp với người học mới bắt đầu hoặc đang củng cố vốn từ.

Về chức năng:

Phần mềm cung cấp các chức năng chính như: học từ vựng qua flashcard, làm bài trắc nghiệm, chọn chủ đề học, theo dõi tiến độ học, quản lý tài khoản người dùng. Các tính năng nâng cao như phát âm, trò chơi từ vựng, hoặc tích hợp AI chưa được triển khai trong phạm vi đề tài.

Về đối tượng sử dụng:

Hướng đến người học tiếng Anh có nhu cầu rèn luyện và ghi nhớ từ vựng, đặc biệt là học sinh tiểu học.

CHƯƠNG 1: GIỚI THIỆU

## 1.1 Giới thiệu về công nghệ phần mềm

Công nghệ phần mềm (Software Engineering) là ngành khoa học và kỹ thuật chuyên nghiên cứu các nguyên lý, phương pháp, quy trình và công cụ để thiết kế, phát triển, kiểm thử, triển khai và bảo trì phần mềm một cách hiệu quả, chính xác và có thể mở rộng. Đó là những khái niệm trong ngành công nghệ thông tin, có sự liên quan mật thiết tới các khía cạnh của quá trình sản xuất phần mềm. Công nghệ được áp dụng một cách có hệ thống cho sự phát triển, sử dụng cũng như để bảo trì các phần mềm hệ thống.

Công nghệ phần mềm được xem là một bộ phận của quy trình công nghệ hệ thống, có liên quan tới sự phát triển của các ứng dụng, hạ tầng, cơ sở dữ liệu và điều khiển hệ thống. Các kỹ sư phần mềm luôn phải tuân thủ quy định của hệ thống, tổ chức trong công việc cũng như khi sử dụng kỹ thuật, công cụ phù hợp với từng vấn đề, tài nguyên sẵn có.

Khác với khoa học máy tính, công nghệ phần mềm không chỉ đề cập tới lý thuyết và các vấn đề cơ bản, mà nó còn tập trung vào hoạt động xây dựng chế tạo ra các sản phẩm phần mềm hệ thống hay phần mềm ứng dụng hữu ích với con người. Sự phát triển mạnh mẽ của ngành kỹ thuật phần mềm đã vượt xa hơn hẳn những lý thuyết khoa học máy tính tích góp nhỏ giọt. [1]

Công nghệ phần mềm bao gồm nhiều lĩnh vực nhỏ như:

- Phân tích và đặc tả yêu cầu phần mềm.

- Thiết kế phần mềm.

- Lập trình và kiểm thử.

- Quản lý dự án phần mềm.

-Bảo trì và cải tiến phần mềm.

Việc áp dụng công nghệ phần mềm giúp tăng hiệu suất phát triển, giảm lỗi trong quá trình xây dựng hệ thống, đồng thời giúp phần mềm dễ bảo trì và nâng cấp về sau.

Hai phần của Software Engineering: Phần mềm (S) và Kỹ thuật (E).

- Phần mềm (S) là tập hợp các mã, tài liệu và trình kích hoạt thực hiện một công việc cụ thể và đáp ứng một yêu cầu cụ thể.

- Kỹ thuật (E) là sự phát triển của các sản phẩm bằng cách sử dụng các phương pháp, nguyên tắc và phương pháp tốt nhất.

SE là việc áp dụng một cách tiếp cận có hệ thống, có kỷ luật, có thể định lượng để phát triển, vận hành và bảo trì phần mềm; nghĩa là ứng dụng kỹ thuật vào phần mềm (IEEE).

## 1.2 Giới thiệu về công nghệ phần mềm Agile

### Giới thiệu về Agile

Agile là một phương thức phát triển phần mềm linh hoạt (Agile software development) nhấn mạnh đến việc chia nhỏ dự án thành các phần nhỏ (gọi là iteration hoặc sprint), phát triển theo từng giai đoạn ngắn và lặp lại liên tục. Trong mỗi vòng lặp, nhóm phát triển sẽ thực hiện đầy đủ các bước như phân tích yêu cầu, thiết kế, lập trình, kiểm thử và có thể bàn giao sản phẩm hoàn chỉnh.

Mô hình Agile là một quá trình phát triển phần mềm gia tăng và lặp đi lặp lại. Không giống như mô hình thác nước (Waterfall) truyền thống tuân theo cách tiếp cận tuần tự, mô hình Agile chia dự án thành các phần nhỏ và quản lý được gọi là các bước lặp lại hoặc chạy nước rút. [2]

Công nghệ phần mềm agile tập trung vào việc:

- Cung cấp chức năng một cách nhanh chóng,

- Đáp ứng các thay đổi đặc tả kỹ thuật của sản phẩm, và

- Giảm thiểu chi phí phát triển.

### Giới thiệu về Scrum

Scrum là một trong những phương pháp Agile được sử dụng rộng rãi nhất. Nó tập trung vào việc quản lý các nhiệm vụ trong môi trường phát triển nhóm. Trong mô hình Scrum Agile, nhóm phát triển tuân theo một kế hoạch làm việc được xác định trước cho mỗi lần chạy nước rút. Các vai trò như Scrum Master (chủ sản phẩm) và Product Owner (chủ sản phẩm) được xác định để đảm bảo cộng tác hiệu quả và phân phối kịp thời các phần gia tăng.

Nhóm Scrum (Scrum Team) gồm 3 vai trò chính:

- Product Owner: Người đại diện cho khách hàng, xác định và sắp xếp độ ưu tiên các tính năng.

-Scrum Master: Người đảm bảo nhóm làm việc theo đúng Scrum, loại bỏ các trở ngại.

- Development Team: Nhóm kỹ thuật thực hiện công việc phát triển phần mềm.

Sự kiện trong Scrum:

- Sprint: Chu kỳ làm việc ngắn, lặp lại đều đặn.

- Sprint Planning: Lập kế hoạch cho Sprint.

- Daily Scrum: Cuộc họp ngắn hằng ngày (15 phút).

- Sprint Review: Trình bày sản phẩm sau mỗi Sprint.

- Sprint Retrospective: Rút kinh nghiệm và cải tiến sau mỗi Sprint.

Tài liệu (Artifacts):

- Product Backlog: Danh sách các tính năng cần phát triển.

- Sprint Backlog: Danh sách các công việc trong Sprint.

## Giới thiệu về Microservices

Microservices là một kiến trúc phần mềm trong đó ứng dụng được chia thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ thực hiện một chức năng cụ thể. Các dịch vụ này giao tiếp với nhau thông qua API. Với sự phát triển của công nghệ đám mây và DevOps, Microservices đã trở thành xu hướng hàng đầu trong phát triển phần mềm hiện đại. [3]

Đặc điểm của microservice:

- Khép kín: mỗi microservice quản lý dữ liệu và giao diện người dùng riêng, không phụ thuộc lẫn nhau.

- Nhẹ: giao tiếp bằng giao thức đơn giản, giúp truyền tải nhanh và ít chi phí.

- Triển khai độc lập: mỗi microservice là một quy trình riêng, có thể phát triển bằng ngôn ngữ và công nghệ khác nhau.

-Định hướng nghiệp vụ: mỗi microservice phục vụ một nhu cầu nghiệp vụ cụ thể, thay vì chỉ là kỹ thuật đơn thuần.

- Gắn kết cao – Ghép nối thấp: các chức năng bên trong microservice được liên kết chặt chẽ, nhưng liên kết lỏng với các dịch vụ khác để dễ bảo trì và thay đổi.

Một hệ thống xác thực có thể được chia thành các microservice như:

- Microservice quản lý UID: lưu trữ và xử lý định danh người dùng.

- Microservice quản lý mật khẩu: tạo, thay đổi, kiểm tra và khôi phục mật khẩu.

- Microservice xác thực hai yếu tố (2FA): gửi mã OTP qua SMS/email.

- Microservice đăng ký người dùng: xử lý thông tin đăng ký ban đầu và xác minh.

- Mỗi microservice trên có thể truy cập một cơ sở dữ liệu riêng hoặc sử dụng chung một trung tâm lưu trữ thông tin xác thực, tùy thuộc vào thiết kế.

Khi xây dựng hệ thống học từ vựng theo kiến trúc microservices, có thể chia các chức năng chính như: quản lý tài khoản, bài học từ vựng, bài kiểm tra, thống kê tiến độ học,... thành các dịch vụ riêng biệt. Điều này giúp:

- Dễ mở rộng từng phần (ví dụ thêm dạng học mới chỉ cần sửa service tương ứng).

-Hệ thống hoạt động ổn định ngay cả khi một phần nào đó gặp sự cố.

- Hỗ trợ triển khai nhanh theo hướng CI/CD và dễ bảo trì về sau.

## Tính an toàn, tính riêng tư và bảo đảm chất lượng phần mềm

### Tính an toàn

Tính an toàn phần mềm (Security Software) là khả năng của một hệ thống phần mềm trong việc bảo vệ thông tin và chức năng của nó khỏi các mối đe dọa, truy cập trái phép, và hành vi tấn công độc hại.

Tính an toàn phần mềm là tập hợp các cơ chế, kỹ thuật và chính sách nhằm đảm bảo rằng phần mềm:

- Chỉ cho phép người dùng hợp lệ truy cập.

- Dữ liệu không bị thay đổi trái phép (toàn vẹn).

- Luôn sẵn sàng hoạt động khi cần.

- Không bị lạm dụng hoặc tấn công gây gián đoạn hoặc thất thoát thông tin.

Tính an toàn đề cập đến khả năng của phần mềm trong việc bảo vệ dữ liệu và chức năng khỏi các truy cập trái phép, hành vi gây hại hoặc tấn công mạng. Một hệ thống an toàn phải đảm bảo:

- Xác thực (Authentication): chỉ người dùng hợp lệ mới có thể truy cập hệ thống.

- Phân quyền (Authorization): người dùng chỉ được phép thực hiện các hành động phù hợp với vai trò của họ.

- Bảo vệ dữ liệu (Data Protection): mã hóa thông tin nhạy cảm, đặc biệt là mật khẩu, thông tin cá nhân.

- Chống xâm nhập (Intrusion Prevention): hạn chế các lỗ hổng bảo mật như SQL injection, XSS,...

- Ghi nhật ký và giám sát (Logging & Monitoring): theo dõi hoạt động bất thường để kịp thời phát hiện sự cố.

Việc đảm bảo tính an toàn đặc biệt quan trọng trong các ứng dụng có liên quan đến thông tin cá nhân hoặc dữ liệu người dùng, như phần mềm học từ vựng tiếng Anh có lưu tài khoản và tiến độ học.

### Tính riêng tư

Tính riêng tư phần mềm (Software Privacy) là khả năng của phần mềm trong việc bảo vệ thông tin cá nhân của người dùng khỏi bị thu thập, sử dụng, lưu trữ hoặc chia sẻ trái phép.

Các nguyên tắc:

- Phần mềm cần giới hạn việc thu thập dữ liệu cá nhân ở mức tối thiểu cần thiết.

- Người dùng phải được thông báo rõ ràng về cách dữ liệu của họ sẽ được sử dụng.

- Cung cấp cơ chế để người dùng kiểm soát dữ liệu cá nhân như sửa đổi, xóa hoặc rút lại quyền truy cập.

- Hệ thống phải thực hiện bảo mật thông tin nhạy cảm, mã hóa dữ liệu, và tuân thủ các chuẩn bảo vệ dữ liệu.

Tính riêng tư đề cập đến cách phần mềm xử lý dữ liệu cá nhân sao cho phù hợp với quyền riêng tư của người dùng và các quy định pháp lý (như GDPR, CCPA). Điều này bao gồm thu thập hợp lý, sử dụng minh bạch và bảo vệ dữ liệu cá nhân khỏi truy cập trái phép.

### Đảm bảo chất lượng phần mềm

Chất lượng phần mềm là sự phù hợp của các yêu cầu cụ thể về hiệu năng và chức năng, các tiêu chuẩn phát triển phần mềm được ghi lại rõ ràng bằng tài liệu với các đặc tính ngầm định của tất cả các phần mềm được phát triển chuyên nghiệp.

Ba yêu cầu với chất lượng phần mềm cần được đáp ứng khi phát triển phần mềm:

- Các yêu cầu chức năng rõ ràng là nhân tố chính quyết định chất lượng đầu ra của phần mềm.

- Các tiêu chuẩn chất lượng phần mềm sẽ được nói đến trong hợp đồng.

- Các đặc tính ngầm định cần được đáp ứng trong quá trình phát triển cho dù không được nói đến rõ ràng trong hợp đồng. [4]

Đảm bảo chất lượng phần mềm (Software Quality Assurance – SQA) là một tập hợp các quy trình, phương pháp và hoạt động có hệ thống nhằm đảm bảo rằng phần mềm được phát triển đáp ứng các yêu cầu đã đặt ra về chức năng, hiệu năng, bảo mật và khả năng sử dụng.

Các hoạt động chính của SQA:

- Lập kế hoạch chất lượng (Quality Planning): Xác định tiêu chuẩn chất lượng và cách đo lường.

- Kiểm tra quy trình (Process Assurance): Giám sát việc tuân thủ quy trình phát triển phần mềm.

- Kiểm thử phần mềm (Product Assurance): Kiểm tra sản phẩm có đúng yêu cầu và hoạt động ổn định không.

- Rà soát và đánh giá (Reviews, Audits): Phát hiện lỗi sớm, đảm bảo tính đúng đắn và nhất quán.

- Tài liệu hóa quy trình: Đảm bảo mọi bước đều có thể theo dõi, lặp lại và cải tiến.

Trong ứng dụng học từ vựng tiếng Anh đảm bảo: Chạy kiểm thử các chức năng như tạo tài khoản, học từ, làm bài kiểm tra. Phần mềm chạy ổn định trên nhiều trình duyệt (Chrome, Edge). Sửa lỗi nhanh chóng khi phát hiện, nhờ có tài liệu rõ ràng và mã dễ bảo trì.

CHƯƠNG 2: PHÂN TÍCH YÊU CẦU

## Chức năng chính của hệ thống

### Chức năng trang Home

Carousel giới thiệu: Hiển thị các hình ảnh quảng bá dưới dạng trình chiếu tự động (slide).

Giới thiệu ứng dụng: Hiển thị đoạn văn mô tả ngắn gọn về mục tiêu, tiện ích và phạm vi sử dụng của ứng dụng.

Thanh điều hướng: Gồm các liên kết đến các trang chính như Trang chủ, Học từ vựng, Kiểm tra, và Tìm kiếm.

### Chức năng tìm kiếm

Tìm kiếm theo từ khóa: Hỗ trợ nhập từ khóa để tìm trong cả danh sách chủ đề và từ vựng.

Lọc kết quả: Cho phép lọc theo loại kết quả gồm: tất cả từ vựng hoặc chủ đề liên quan đến từ khóa tìm kiếm, chỉ chủ đề hoặc chỉ từ vựng.

Gợi ý từ khóa: Khi người dùng nhập ký tự, hệ thống gợi ý từ khoá liên quan theo thời gian thực.

Hiển thị kết quả: Kết quả được trình bày rõ ràng dưới dạng danh sách, mỗi mục bao gồm từ vựng, nghĩa và các thông tin liên quan.

### Chức năng học

Quản lý chủ đề: Quản lý danh sách các chủ đề: gồm tên chủ đề, hình ảnh cho chủ đề, và một số mô tả cho từng chủu đề.

Flashcard: Từ vựng được trình bày dưới dạng thẻ, cho phép người học xem từ và lật để xem nghĩa, giúp tăng cường ghi nhớ bằng tương tác.

Hệ thống bài học: Các từ vựng được phân chia theo các chủ đề, hỗ trợ lộ trình học tập rõ ràng.

### Chức năng kiểm tra

Giao diện bài kiểm tra: Hiển thị các câu hỏi dưới dạng trắc nghiệm.

Đồng hồ đếm thời gian: Quản thời gian làm bài.

Điều hướng câu hỏi: Người dùng có thể thực hiện thao tác chuyển đến các câu hỏi tiếp theo hoặc quay lại câu hỏi trước đó, nộp bài khi làm xong.

Lưới hiển thị các câu hỏi: Cho phép người dùng xem được số lượng câu hỏi số câu đã làm và chưa làm.

Hiển thị kết quả làm bài: Hiển thị số câu trả lời đúng và số điểm cho bài test.

### Chức năng quản lý tài khoản người dùng

Đăng ký/Đăng nhập: Hỗ trợ tạo và xác thực tài khoản người dùng.

Menu cài đặt: Cho phép người dùng thay đổi cài đặt hoặc đăng xuất khỏi hệ thống.

## Yêu cầu phi chức năng

Hiệu năng (Performance)

* Ứng dụng sử dụng API Gateway giúp tối ưu hóa routing và giảm độ trễ.
* Phân tách các service (auth, topic, test, user) giúp giảm tải cho từng thành phần.

Bảo mật (Security)

* Sử dụng JWT cho xác thực, mã hóa mật khẩu bằng bcrypt.
* Có kiểm tra quyền truy cập khi thực hiện các chức năng (ví dụ: chỉ user đã đăng nhập mới truy cập được các tính năng học/test).

Khả năng mở rộng (Scalability)

* Microservices và Docker giúp dễ dàng mở rộng từng service độc lập.
* Có thể triển khai nhiều instance cho từng service khi cần.

Độ tin cậy (Reliability)

* Có health check cho từng service.

Khả năng bảo trì (Maintainability)

* Code chia module rõ ràng, tách biệt frontend/backend, từng service có cấu trúc riêng.
* Sử dụng React component hóa giúp dễ bảo trì UI.

Khả năng sử dụng (Usability)

* Giao diện trực quan, có thông báo lỗi, loading, modal xác thực.
* Responsive design cho nhiều thiết bị.

Tính di động (Portability)

* Dockerfile cho cả frontend/backend giúp triển khai trên nhiều nền tảng.
* Sử dụng biến môi trường cho cấu hình.

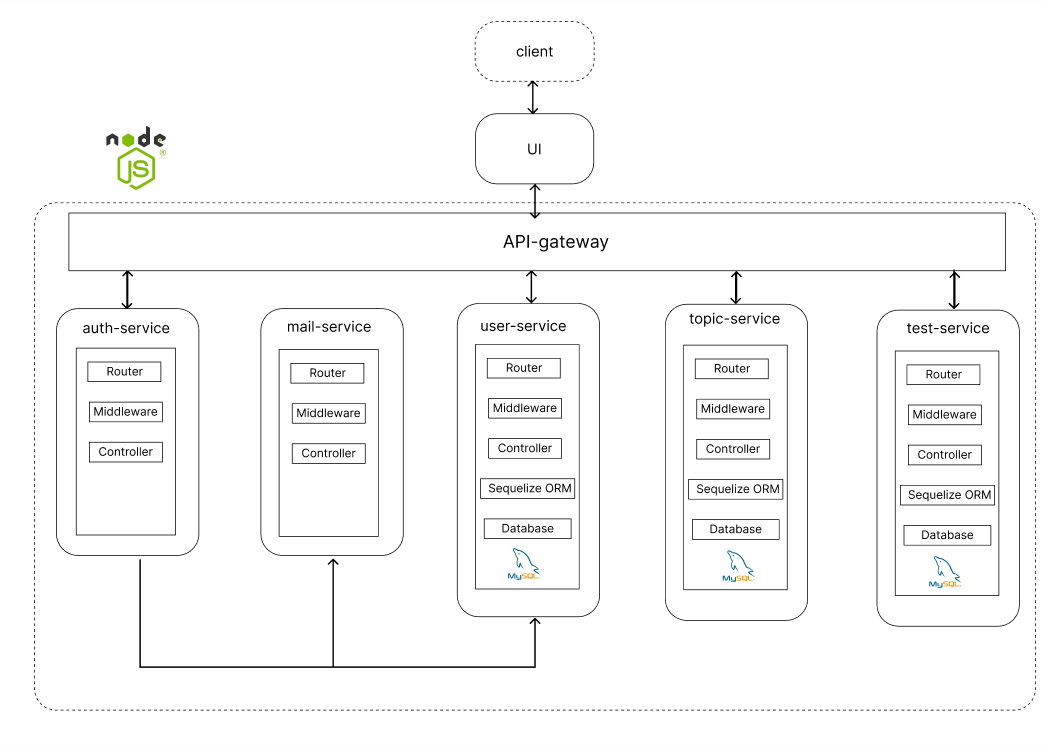
Khả năng tương thích (Compatibility)

* Sử dụng chuẩn REST API, JWT, React, Bootstrap – dễ tích hợp với các hệ thống khác.

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

## Kiến trúc hệ thống

### Kiến trúc



Hình 3.1 Sơ đồ kiến trúc hệ thống

Kiến trúc Microservice sử dụng Node.js kết hợp API Gateway và Sequelize ORM+MySQL. Đây là kiến trúc hiện đại giúp dễ dàng quản lý dự án theo từng phần riêng lẽ, mỗi service đảm bảo một chức năng duy nhất:

Client: Client là thiết bị người dùng, có thể là máy tính hoặc điện thoại di động, truy cập vào hệ thống thông qua trình duyệt web hoặc ứng dụng. Người dùng tương tác trực tiếp với giao diện để thực hiện các chức năng như đăng nhập, học từ vựng, làm bài kiểm tra,...

UI: Giao diện người dùng được phát triển bằng React.js chịu trách nhiệm hiển thị thông tin và xử lý các thao tác từ phía người dùng.

API Gateway: Là cổng trung gian tiếp nhận yêu cầu từ UI, xác thực và phân phối đến các micro-service tương ứng.

Các Microservice:

- auth-service: Xử lý đăng ký, đăng nhập, xác thực người dùng, cấp phát và kiểm tra token JWT. Đảm bảo chỉ người dùng hợp lệ mới truy cập được các chức năng bảo vệ.

- user-service: Quản lý thông tin của người dùng bao gồm tên đăng nhập, email đăng nhập.

- topic-service: Quản lý các chủ đề từ vựng, từ vựng theo từng chủ đề. Cung cấp API lấy danh sách chủ đề, chi tiết chủ đề, tìm kiếm từ vựng, thêm/sửa/xóa từ vựng.

- test-service: Quản lý các bài kiểm tra, câu hỏi, các câu trả lời trắc nghiệm, kết quả làm bài. Cung cấp API lấy danh sách bài test, chi tiết bài test, nộp bài, xem kết quả.

Database: Mỗi microservice sử dụng một database MySQL riêng biệt. Việc tách database giúp tăng tính độc lập, bảo mật, dễ mở rộng và bảo trì.

### Luồng hoạt động

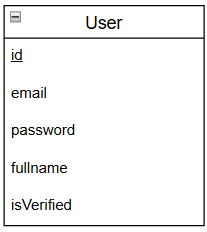
* Người dùng thao tác trên UI (ví dụ: đăng nhập, chọn chủ đề học, làm bài kiểm tra).
* UI gửi request đến API Gateway.
* API Gateway xác thực request (kiểm tra token, phân quyền).
* Gateway định tuyến request đến microservice phù hợp (auth, user, topic, test).
* Microservice xử lý nghiệp vụ, truy vấn database riêng, trả kết quả về Gateway.
* Gateway trả response về UI, UI hiển thị kết quả cho người dùng.

### Lợi ích của Microservice

* Tách biệt chức năng: Mỗi service đảm nhiệm một nhiệm vụ duy nhất, dễ bảo trì, phát triển và kiểm thử.
* Khả năng mở rộng: Có thể scale từng service độc lập theo nhu cầu thực tế, tối ưu tài nguyên.
* Bảo mật: API Gateway kiểm soát xác thực, phân quyền truy cập, bảo vệ các service phía sau.
* Dễ triển khai: Docker hóa từng service, dễ dàng triển khai trên nhiều môi trường (local, cloud).
* Khả năng phục hồi: Nếu một service gặp sự cố, các service khác vẫn hoạt động bình thường.

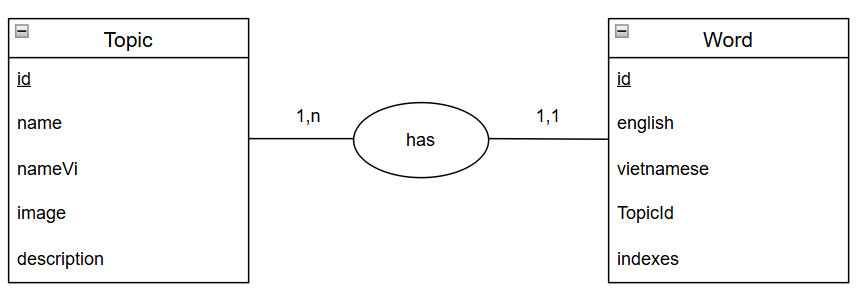
## Thiết kế cơ sở dữ liệu

### 3.2.1 Cơ sở dữ liệu user-service



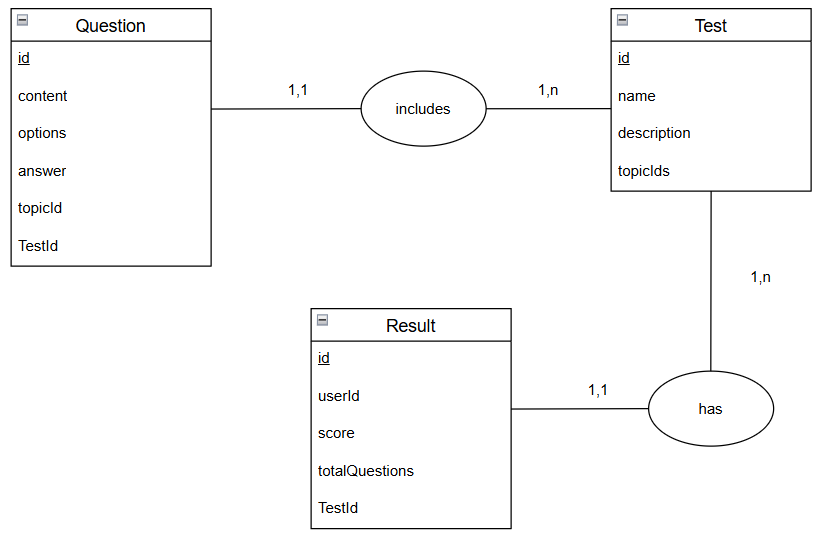
Hình 3.2 User

### 3.2.2 Cơ sở dữ liệu topic-service



Hình 3.3 topic-service

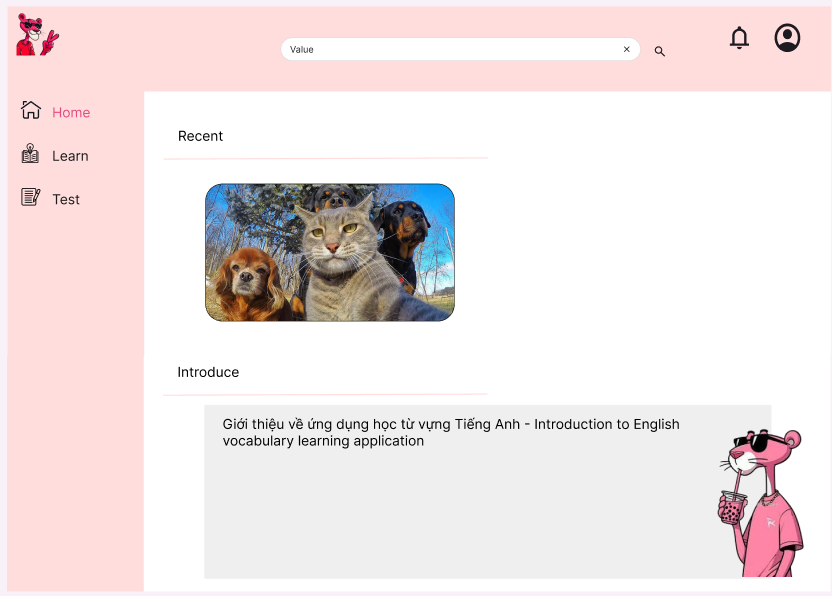
### 3.2.3 Cơ sở dữ liệu test-service



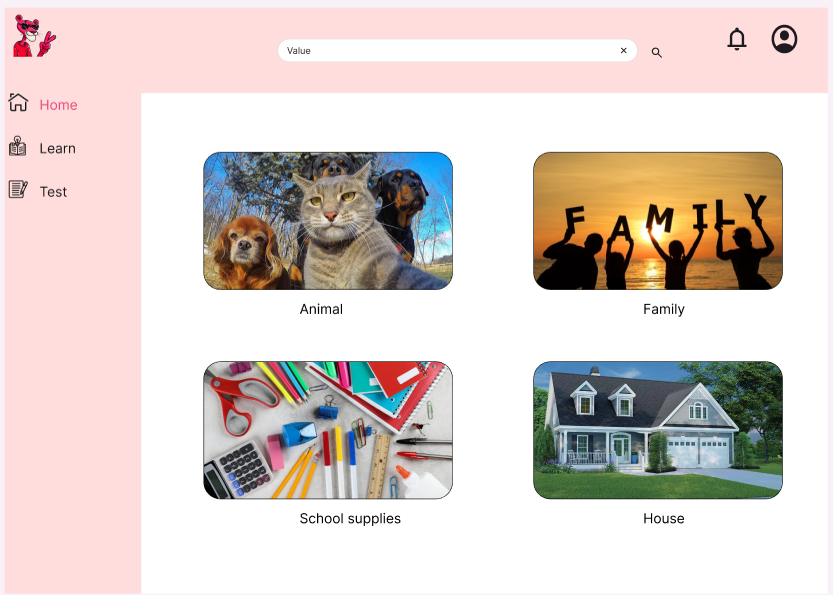
Hình 3.4 test service

## Thiết kế giao diện

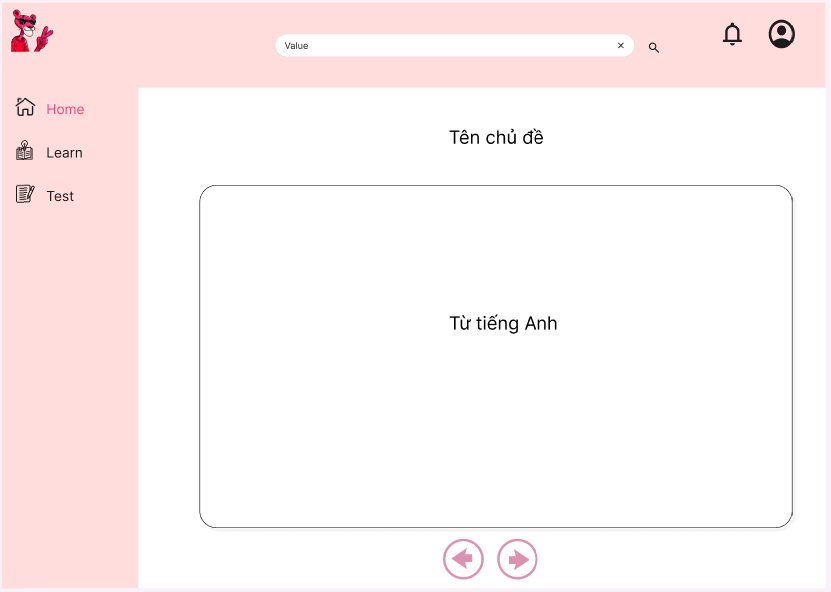
### 3.3.1 Thiết kế Figma



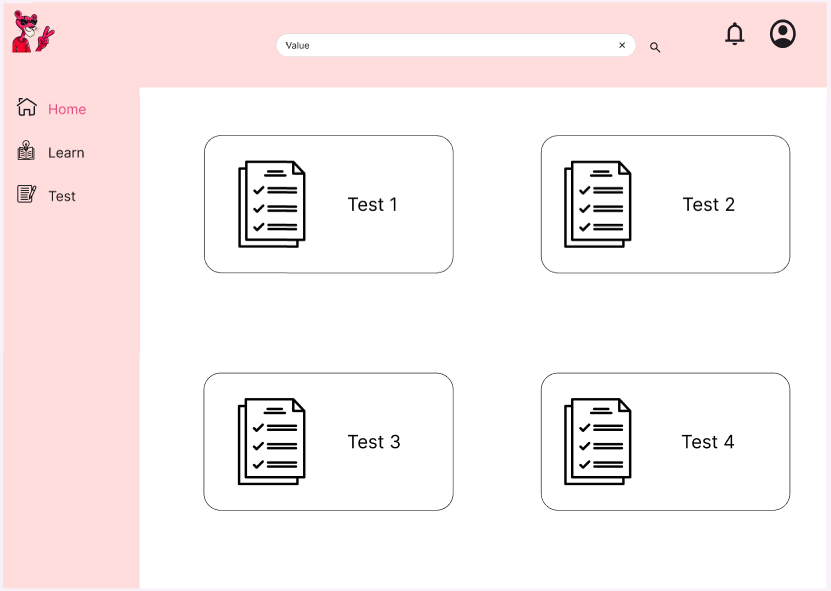
Hình 3.5 Giao diện Figma trang chủ



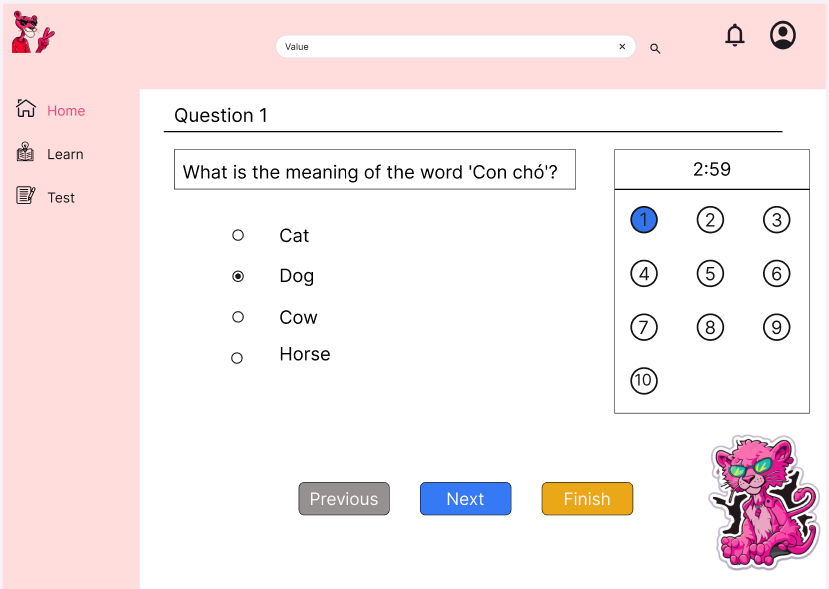
Hình 3.6 Giao diện Figma trang chủ đề từ vựng



Hình 3.7 Giao diện Figma trang học từ vựng

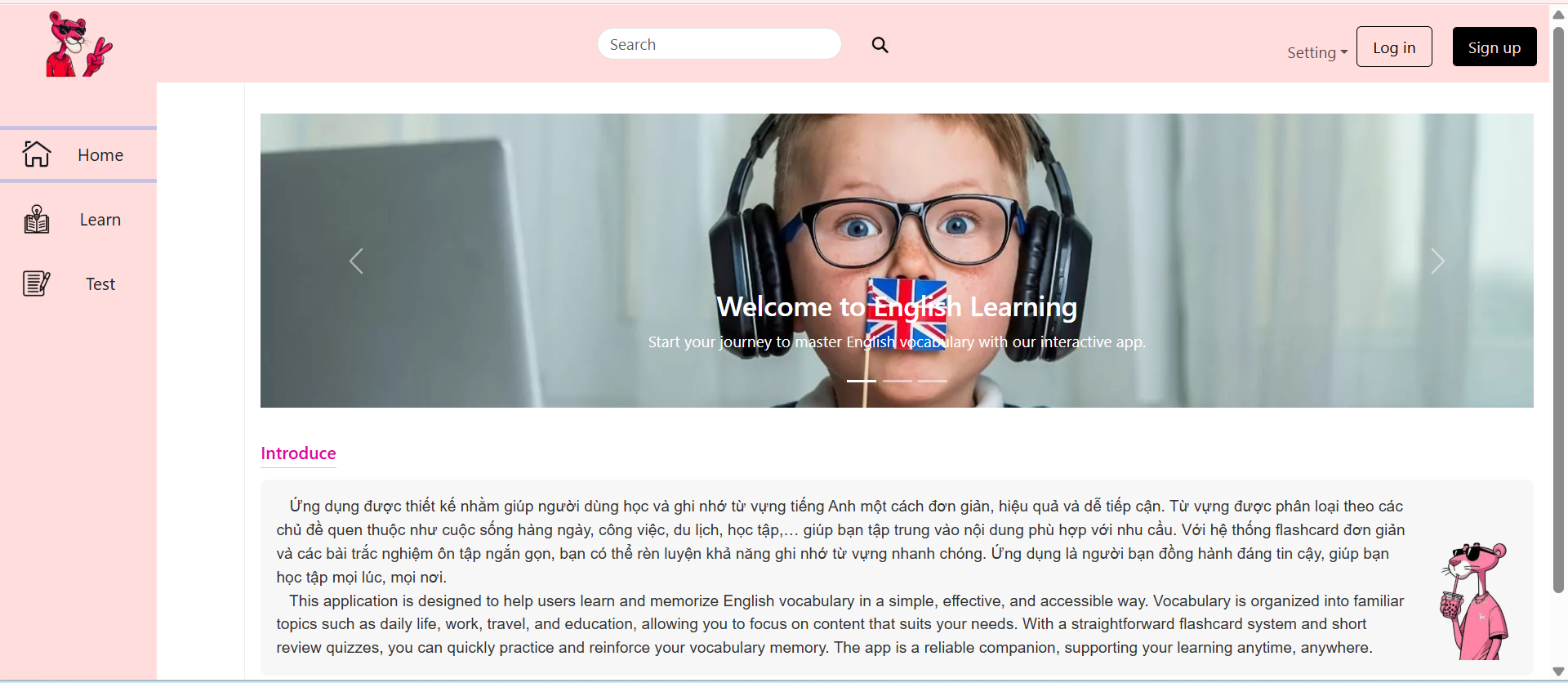


Hình 3.8 Giao diện Figma trang các bài test

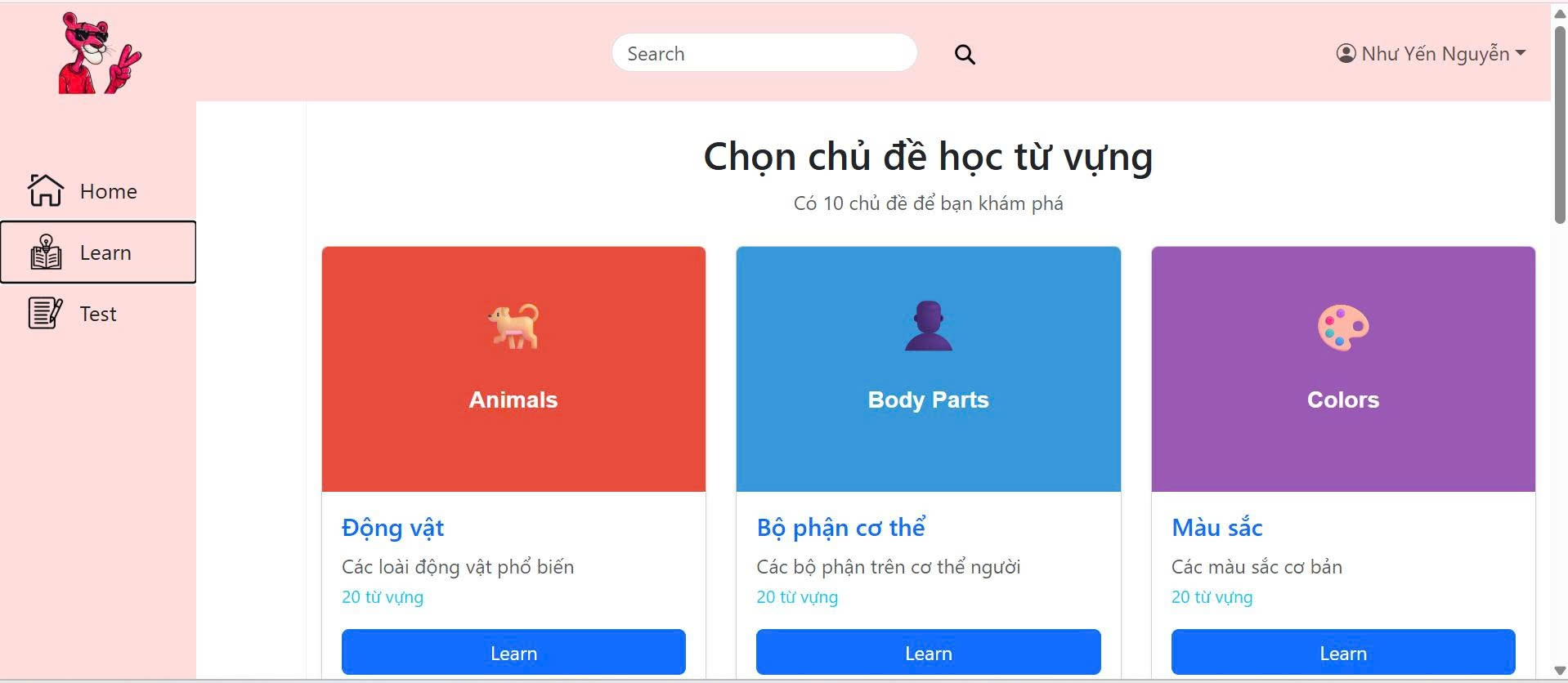


Hình 3.9 Giao diện Figma trang bài test

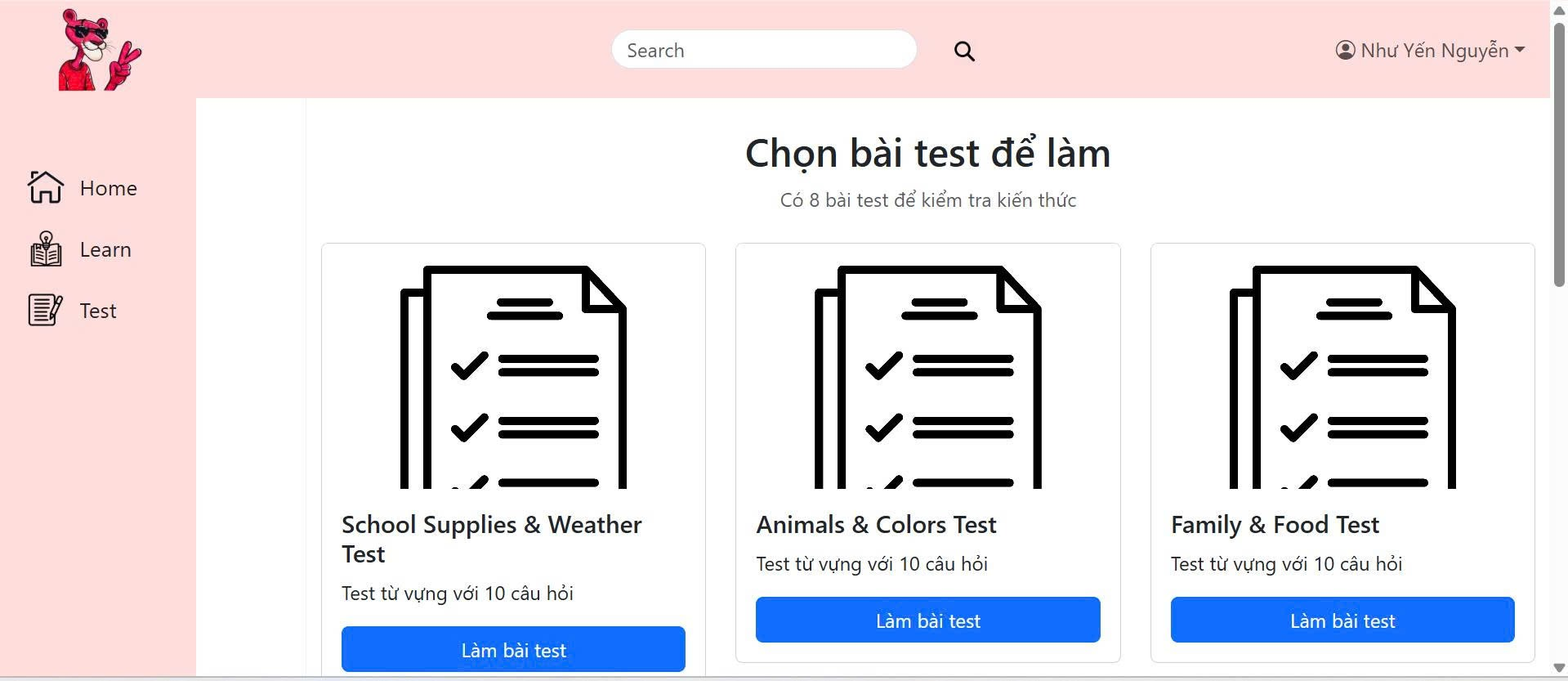
### 3.3.2 Thiết kế giao diện với React



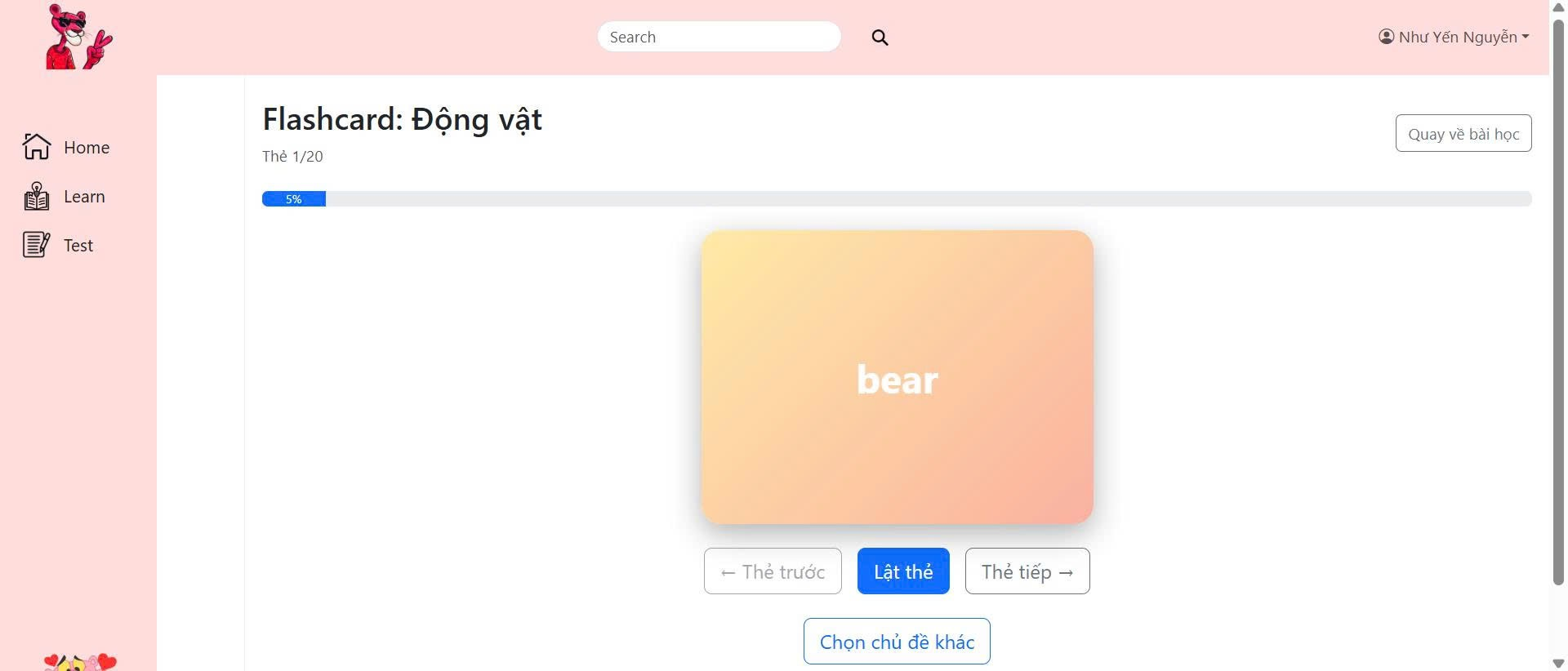
Hình 3.10 Giao diện trang chủ



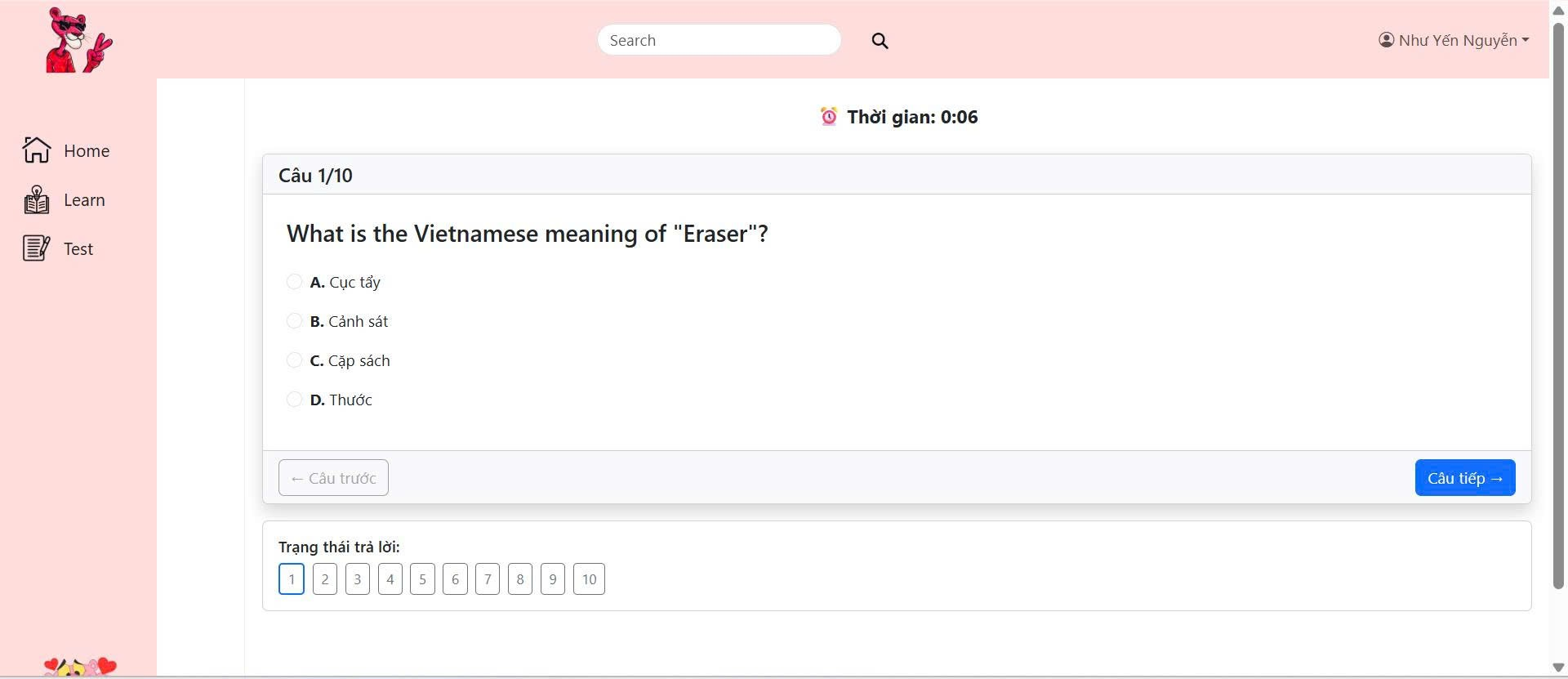
Hình 3.11 Giao diện trang chủ đề



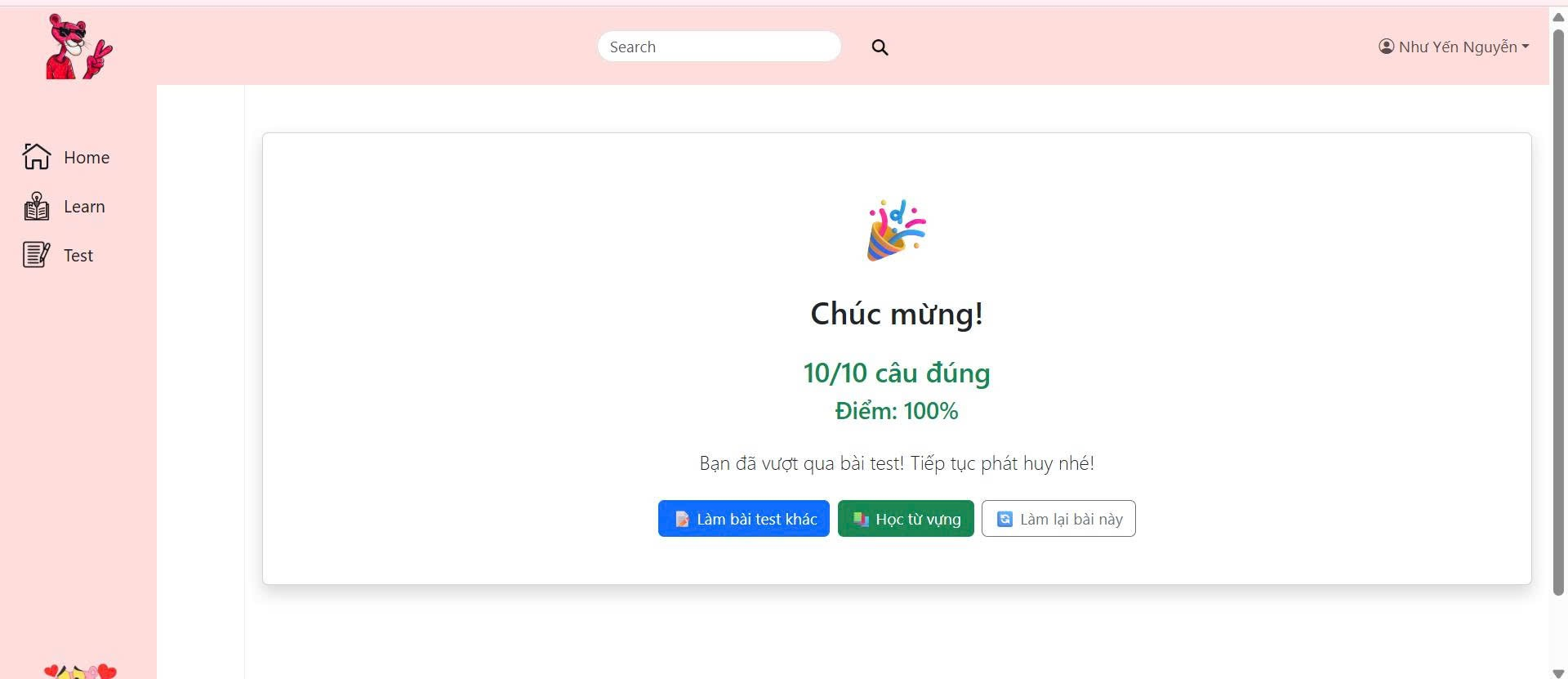
Hình 3.12 Giao diện các bài kiểm tra



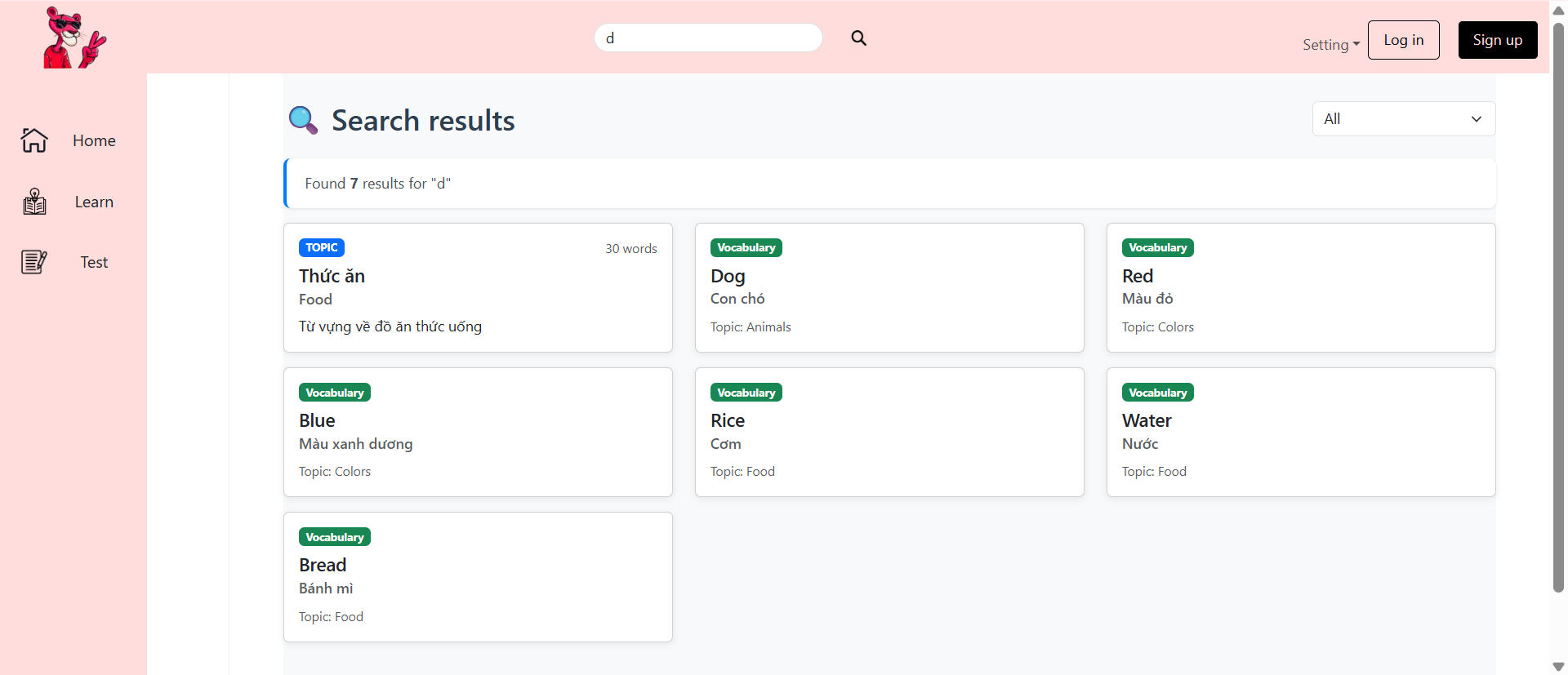
Hình 3.13 Giao diện học từ vựng



Hình 3.14 Giao diện làm bài



Hình 3.15 Giao diện hoàn thành bài kiểm tra



Hình 3.16 Giao diện kết quả tìm kiếm

CHƯƠNG 4: TRIỂN KHAI VÀ CÔNG NGHỆ

## 4.1 Danh sách các công nghệ đã sử dụng

Dự án đã áp dụng các công cụ/ công nghệ sau:

Jira: Quản lý các dự án, lập kết hoạch sprint và phân chia nhiệm vụ nhóm.

Figma: Sử dụng để thiết kế bản phát thảo giao diện của ứng dụng, các mối liên kết của các màn hình trong ứng dụng.

GitHub: Quản lý mã nguồn, quản lý nhánh main, kiểm soát và cập nhật phiên bản dự án.

GitHub Actions: Tự động chạy kiểm thử đơn vị (unit test) mỗi khi API được cập nhật lên GitHub.

Docker: Đóng gói và triển khai dự án trong môi trường container.

Swagger: Tạo tài liệu API tự động, giúp mô tả và kiểm thử các endpoint.

RestFUL API: Áp dụng nguyên tắc REST để xây dựng API rõ ràng, hiệu quả.

## 4.2 Quy trình CI/CD với GitHub Actions.

Khái niệm CI/CD:

Continuous Integration (CI) là quá trình liên tục tích hợp mã nguồn mới vào nhánh chính (main) và kiểm tra chất lượng của mã thông qua các bước tự động như: kiểm thử đơn vị, kiểm tra định dạng code, phân tích bảo mật, v.v.

Continuous Deployment (CD) là quá trình triển khai tự động ứng dụng sau khi vượt qua các bước kiểm thử, thường là đẩy lên môi trường staging hoặc production.

GitHub Actions là một dịch vụ CI/CD tích hợp trực tiếp trong GitHub, cho phép định nghĩa các tác vụ (workflow) tự động thực hiện khi xảy ra các sự kiện như push, pull\_request, hay release. Các workflow này được viết dưới dạng file YAML trong thư mục .github/workflows/.

Cấu hình quy trình CI/CD:

Trong đề tài, nhóm triển khai quy trình CI gồm các bước:

1. Mỗi khi có push hoặc pull request lên nhánh main, GitHub Actions sẽ tự động kích hoạt.
2. Mã nguồn được tải về một môi trường máy ảo (runner) chạy Ubuntu.
3. Thiết lập môi trường Node.js, cài đặt thư viện bằng npm install.
4. Chạy kiểm thử đơn vị với lệnh npm test.
5. (Tùy chọn) Nếu kiểm thử thành công, thực hiện build và triển khai đến môi trường staging hoặc production.

## 4.3 Cấu hình Docker và quy trình triển khai ứng dụng.

### 4.3.1 Cấu hình Docker

Ứng dụng được đóng gói trong container thông qua Docker để đảm bảo khả năng chạy đồng nhất trên mọi môi trường.

|  |
| --- |
| # Sử dụng image Node.js phiên bản 18 làm nền  FROM node:18  # Thiết lập thư mục làm việc trong container  WORKDIR /app  # Sao chép file cấu hình gói để cài đặt dependencies  COPY package\*.json ./  # Cài đặt các thư viện phụ thuộc  RUN npm install  # Sao chép toàn bộ mã nguồn vào container  COPY . .  # Mở cổng 8000 (ứng dụng sẽ chạy tại đây)  EXPOSE 8000  # Lệnh khởi chạy ứng dụng  CMD ["node", "index.js"] |

### 4.3.2 Quy trình triển khai ứng dụng trên Docker

Chuẩn bị Docker file cho từng microservices:

Mỗi dịch vụ (service) có một thư mục riêng chứa mã nguồn và file Dockerfile.

|  |
| --- |
| /user-service/Dockerfile  /mail-service/Dockerfile  /topic-service/Dockerfile  /test-service/Dockerfile  /auth-service/Dockerfile |

Viết file docker-compose.yml để chạy toàn bộ hệ thống:

|  |
| --- |
| version: '3.8'  services:  # ==================== FRONTEND ====================  frontend:  build:  context: ./frontend  dockerfile: Dockerfile  ports:  - "3000:3000"  environment:  - REACT\_APP\_API\_URL=http://localhost:8000/api  depends\_on:  - api-gateway  restart: unless-stopped  # ==================== API GATEWAY ====================  api-gateway:  build:  context: ./backend/api-gateway  dockerfile: Dockerfile  ports:  - "8000:8000"  environment:  - NODE\_ENV=production  - FRONTEND\_URL=http://localhost:3000  depends\_on:  - auth-service  - user-service  - topic-service  - test-service  - mail-service  restart: unless-stopped  # ==================== MICROSERVICES ====================  auth-service:  build:  context: ./backend/auth-service  dockerfile: Dockerfile  ports:  - "5001:5001"  environment:  - NODE\_ENV=production  - USER\_SERVICE\_URL=http://user-service:5004  - JWT\_SECRET=secretkey  - JWT\_EXPIRES\_IN=1d  depends\_on: # Auth service cần user-service hoạt động  - user-service  restart: unless-stopped  mail-service:  build:  context: ./backend/mail-service  dockerfile: Dockerfile  ports:  - "5002:5002"  environment:  - NODE\_ENV=production  - MAIL\_HOST=smtp.gmail.com  - MAIL\_PORT=587  - EMAIL=duongbaokhanh123tv@gmail.com  - EMAIL\_PASS=inrpcnnqellosqdk  - MAIL\_FROM="VocabMafia <noreply@vocabmafia.com>"  restart: unless-stopped  user-service:  build:  context: ./backend/user-service  dockerfile: Dockerfile  ports:  - "5004:5004"  environment:  - NODE\_ENV=production  - DB\_HOST=db-users  - DB\_USER=root  - DB\_PASSWORD=0000  - DB\_NAME=user\_db  - DB\_DIALECT=mysql  - DB\_PORT=3306  depends\_on:  db-users:  condition: service\_healthy # SỬA LỖI: Chờ cho đến khi DB sẵn sàng  restart: unless-stopped  topic-service:  build:  context: ./backend/topic-service  dockerfile: Dockerfile  ports:  - "5005:5005"  environment:  - NODE\_ENV=production  - DB\_HOST=db-topics  - DB\_USER=root  - DB\_PASSWORD=0000  - DB\_NAME=topic\_db  - DB\_DIALECT=mysql  - DB\_PORT=3306  depends\_on:  db-topics:  condition: service\_healthy  restart: unless-stopped  test-service:  build:  context: ./backend/test-service  dockerfile: Dockerfile  ports:  - "5006:5006"  environment:  - NODE\_ENV=production  - DB\_HOST=db-tests  - DB\_USER=root  - DB\_PASSWORD=0000  - DB\_NAME=test\_db  - DB\_DIALECT=mysql  - DB\_PORT=3306  depends\_on:  db-tests:  condition: service\_healthy # SỬA LỖI: Chờ cho đến khi DB sẵn sàng  restart: unless-stopped  # ==================== DATABASES ====================  db-users:  image: mysql:8.0  restart: always  environment:  MYSQL\_ROOT\_PASSWORD: "0000"  MYSQL\_ROOT\_HOST: "%"  MYSQL\_DATABASE: user\_db  ports:  - "3308:3306"  volumes:  - mysql\_users\_data:/var/lib/mysql  healthcheck: # SỬA LỖI: Thêm kiểm tra sức khỏe  test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-p0000"]  interval: 10s  timeout: 5s  retries: 5  db-topics:  image: mysql:8.0  restart: always  environment:  MYSQL\_ROOT\_PASSWORD: "0000"  MYSQL\_ROOT\_HOST: "%"  MYSQL\_DATABASE: topic\_db  ports:  - "3309:3306"  volumes:  - mysql\_topics\_data:/var/lib/mysql  healthcheck: # SỬA LỖI: Thêm kiểm tra sức khỏe  test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-p0000"]  interval: 10s  timeout: 5s  retries: 5  db-tests:  image: mysql:8.0  restart: always  environment:  MYSQL\_ROOT\_PASSWORD: "0000"  MYSQL\_ROOT\_HOST: "%"  MYSQL\_DATABASE: test\_db  ports:  - "3310:3306"  volumes:  - mysql\_tests\_data:/var/lib/mysql  healthcheck: # SỬA LỖI: Thêm kiểm tra sức khỏe  test: ["CMD", "mysqladmin", "ping", "-h", "localhost", "-u", "root", "-p0000"]  interval: 10s  timeout: 5s  retries: 5  volumes:  mysql\_users\_data:  mysql\_topics\_data:  mysql\_tests\_data: |

Build và chạy toàn bộ hệ thống:

Chạy lệnh:

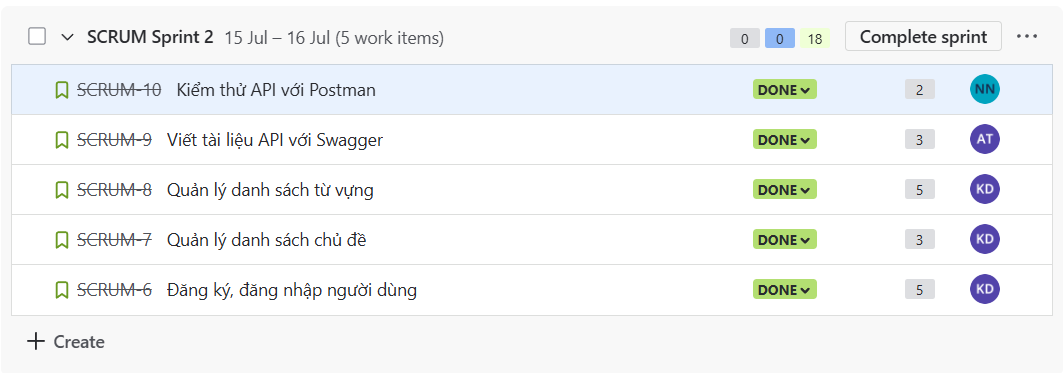
|  |
| --- |
| docker-compose up --build -d |

CHƯƠNG 5: QUẢN LÝ DỰ ÁN

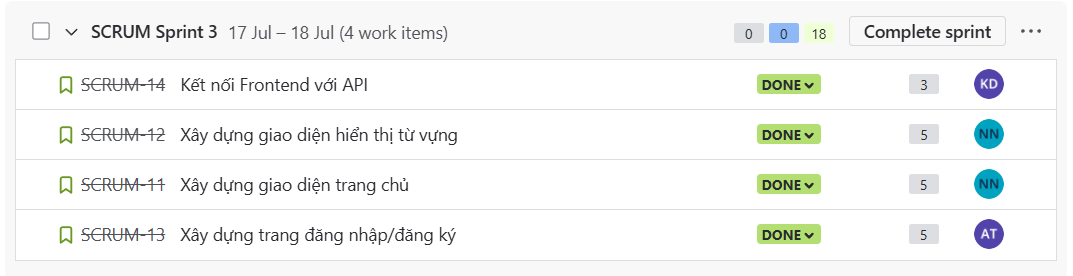
Phân công nhiệm vụ của từng thành viên trong nhóm bằng Jira:



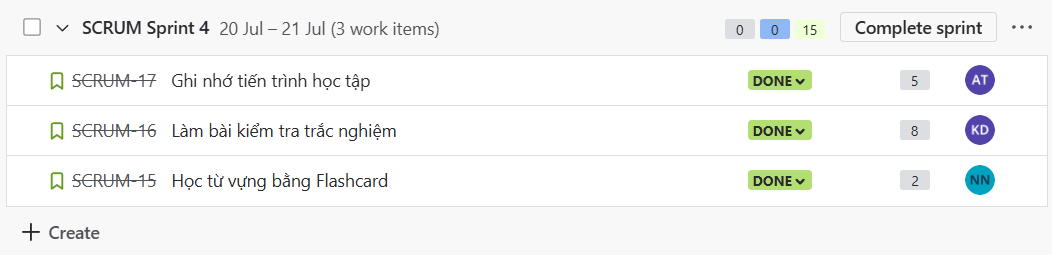
Hình 5.1 SCRUM Sprint 1



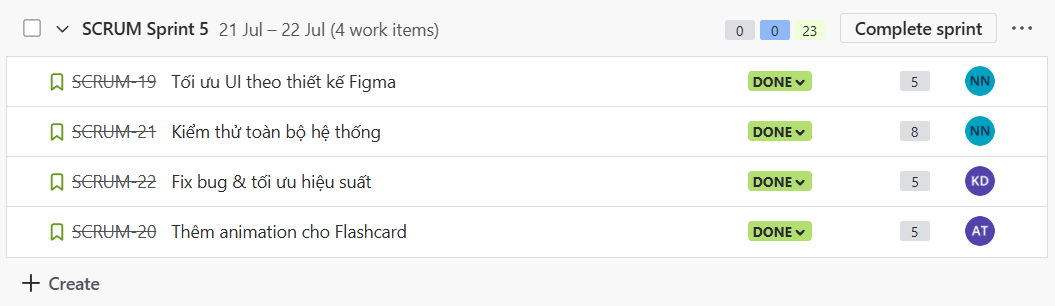
Hình 5.2 SCRUM Sprint 2



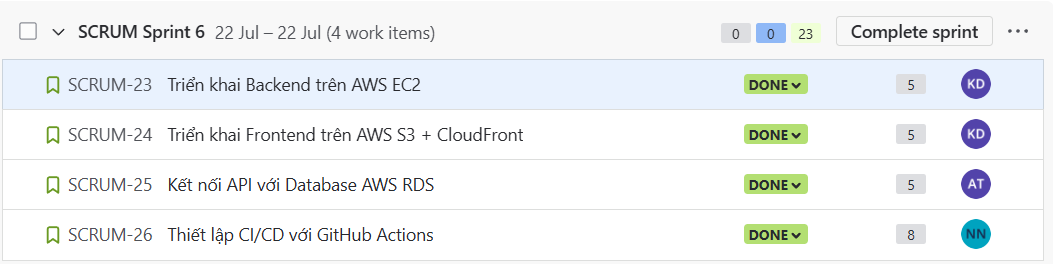
Hình 5.3 SCRUM Sprint 3



Hình 5.4 SCRUM Sprint 4



Hình 5.5 SCRUM Sprint 5



Hình 5.6 SCRUM Sprint 6

CHƯƠNG 6: KIỂM THỬ

## 6.1 Kiểm thử unit test

Trong đề tài này, nhóm áp dụng kiểm thử đơn vị (unit test) nhằm đảm bảo từng thành phần logic trong hệ thống hoạt động đúng yêu cầu kỹ thuật, và sử dụng GitHub Actions để tự động hóa quá trình kiểm thử mỗi khi có cập nhật mã nguồn.

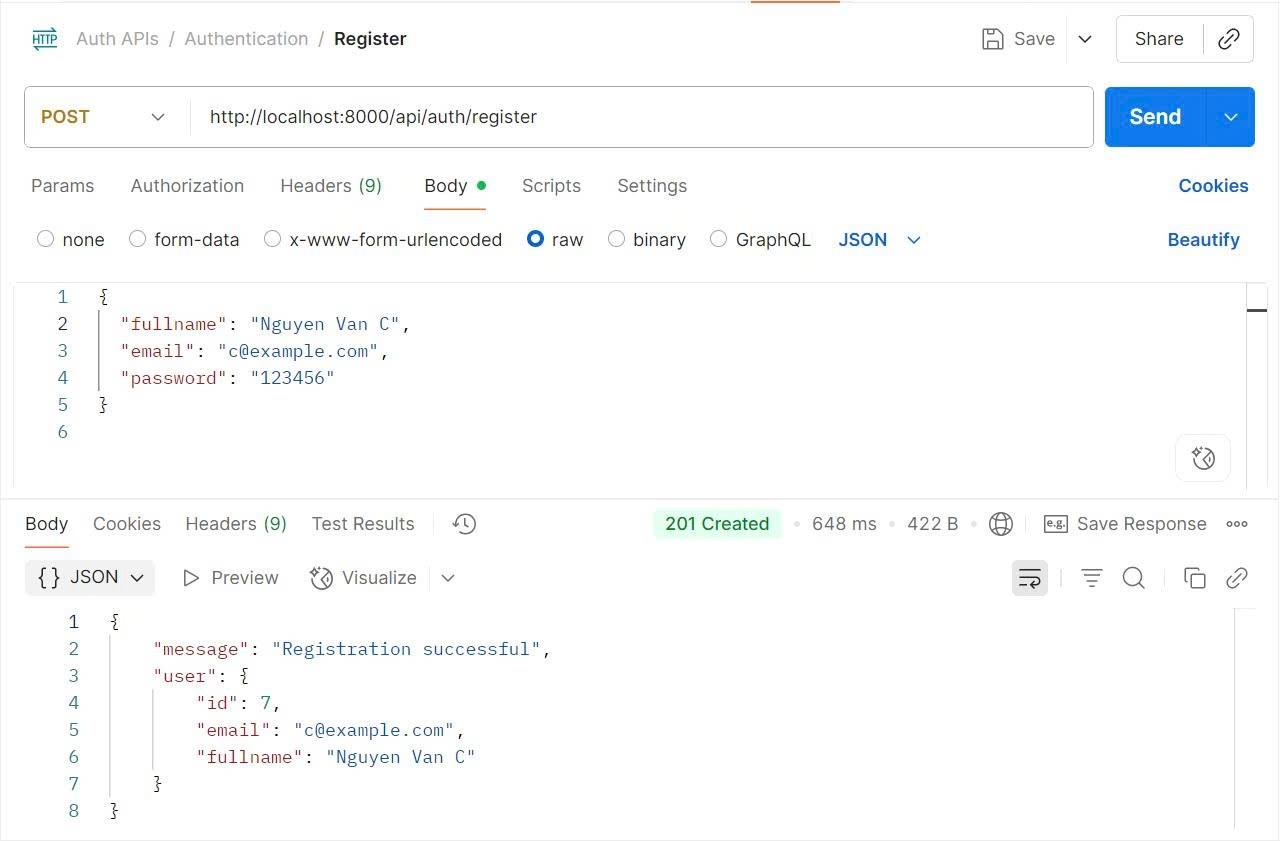
Kiểm thử đơn vị được thực hiện cho từng service độc lập trong hệ thống, chủ yếu tập trung vào các module xử lý nghiệp vụ như xác thực người dùng, chấm điểm bài thi, gửi email xác nhận và xử lý API nội bộ. Các bài test được viết bằng Jest – một framework kiểm thử phổ biến trong Node.js hỗ trợ test đồng bộ và bất đồng bộ, đo độ phủ mã nguồn (code coverage), và cho phép mock các thành phần phụ thuộc.

Cấu trúc test được đặt theo dạng \*.test.js đặt gần các controller tương ứng. Mỗi hàm được test độc lập với các phụ thuộc như cơ sở dữ liệu, dịch vụ mail hoặc JWT bằng cách sử dụng mock/stub. Các bài test được thực thi bằng lệnh npm test.

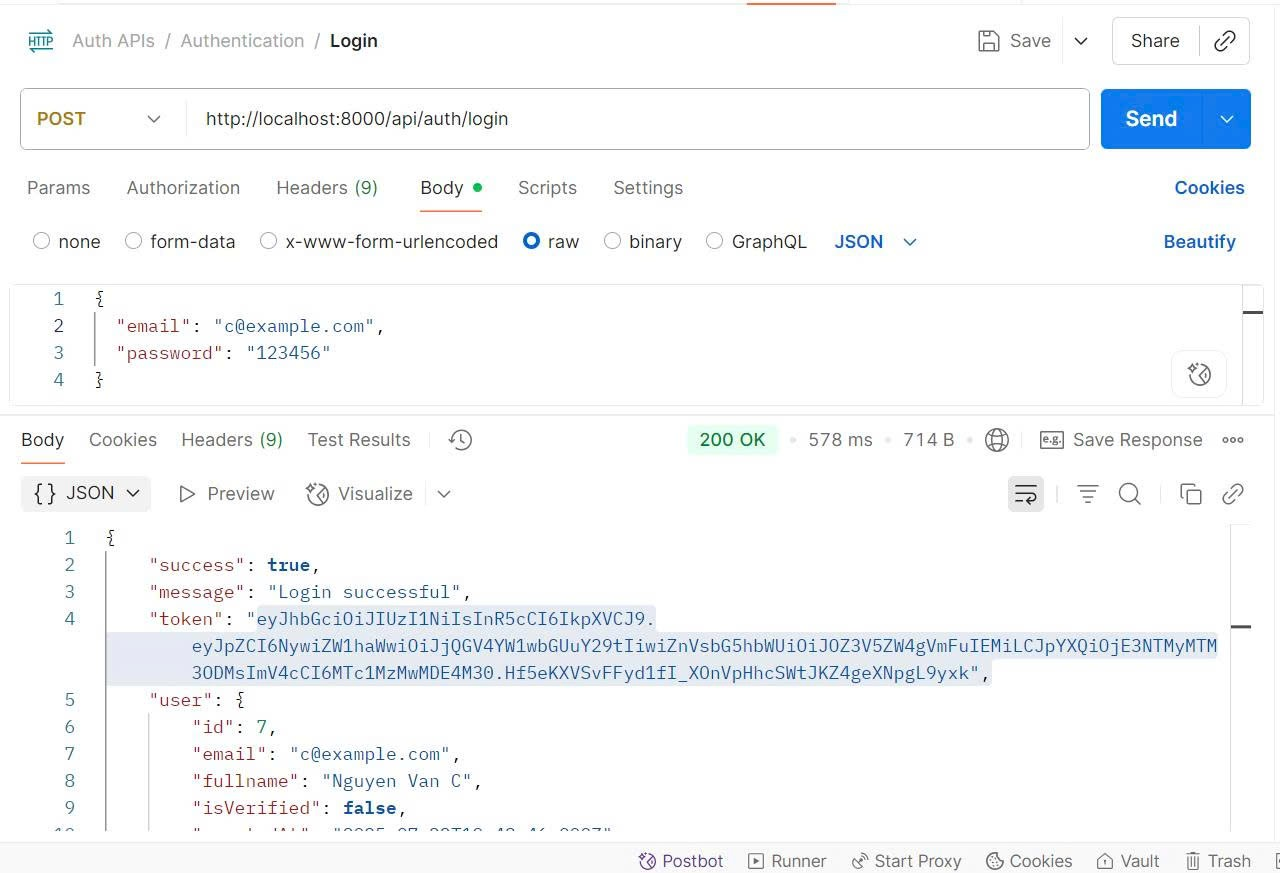
Để đảm bảo tính ổn định khi làm việc nhóm và giảm rủi ro tích hợp mã lỗi vào hệ thống chính, nhóm sử dụng GitHub Actions để thực hiện kiểm thử tự động. Mỗi lần có thao tác push hoặc pull request đến nhánh chính, một quy trình CI (Continuous Integration) sẽ được kích hoạt, thực hiện các bước:

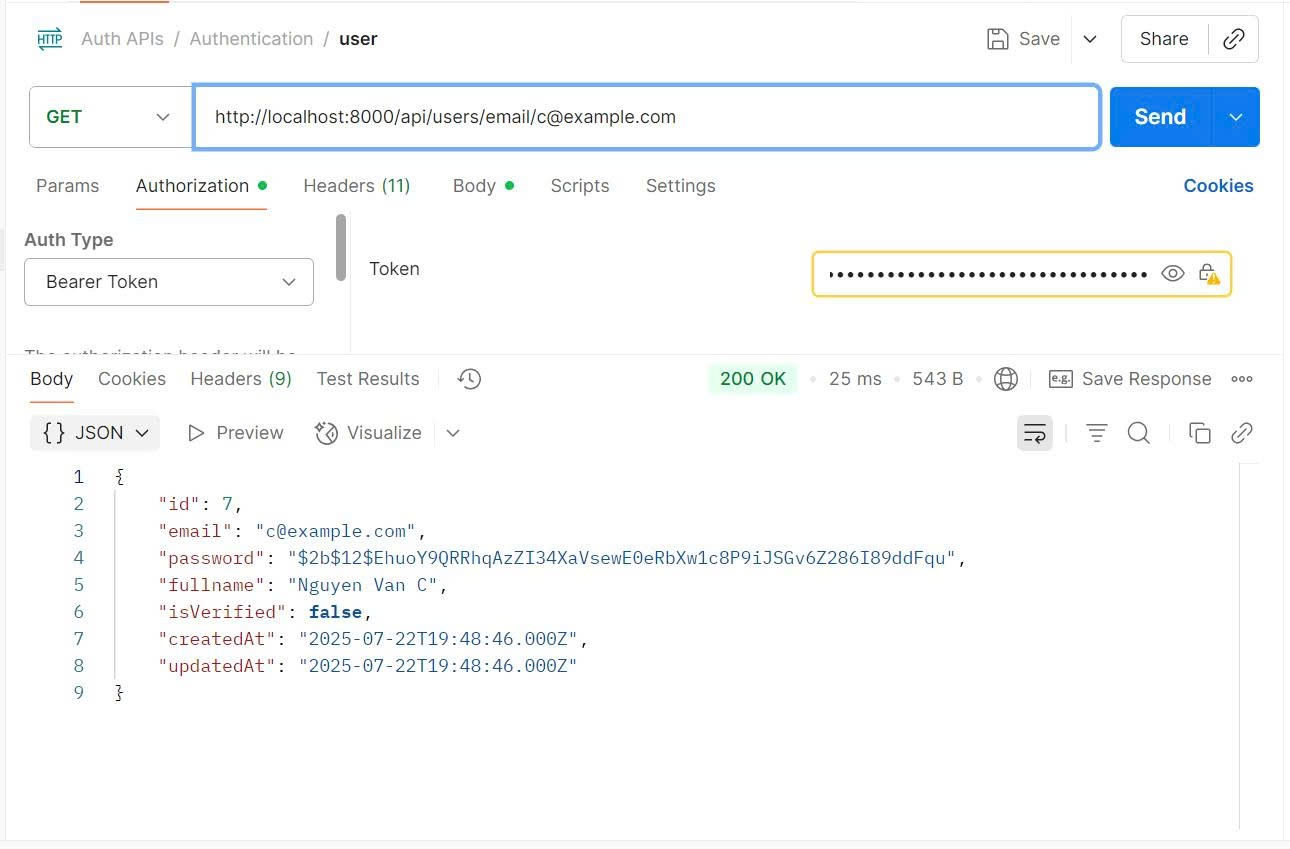
Quy trình này giúp phát hiện lỗi ngay khi đẩy code, tránh việc merge các đoạn mã gây lỗi runtime hoặc sai logic. Trong các hệ thống nhiều service, mỗi service có thể có workflow riêng biệt, giúp kiểm soát chất lượng theo từng phần độc lập.

## 6.2 Kiểm thử API với Postman:



Hình 6.1 Kiểm API đăng kí

Hình 6.2 Kiểm API đăng nhập



Hình 6.3 Lấy API đăng nhập

CHƯƠNG 7: ĐÁNH GIÁ VÀ KẾT LUẬN

Trong quá trình nghiên cứu và xây dựng dự án, nhóm đã xây dựng thành công các công việc mà mình đặt ra. Bên cạnh đó nhóm cũng mắc phải một số khó khăn về kỹ thuật, vấn đề lỗi mã phát sinh. Song, chúng em cũng đã cố gắng thay đổi và chỉnh sửa dự án để đưa ra một kết quả tốt nhất.

Chúng em đã tiếp thu được nhiều kiến thức bổ ích về quy trình xây dựng ứng dụng có kế hoạch rõ ràng, tiếp cận được nhiều công cụ và công nghệ khác nhau để áp dụng cho dự án. Cuối cùng cho ra được một ứng dụng học từ vựng tiếng Anh với các chức năng hữu dụng và dễ sử dụng. Ứng dụng còn có thể phát triển trong tương lai và có giá trị ứng dụng thực tiễn cao trong việc hỗ trợ người học nâng cao vốn từ vựng tiếng Anh một cách hiệu quả. Trong tương lai, nhóm kỳ vọng có thể tích hợp thêm các tính năng như học qua trò chơi, luyện phát âm qua AI để nâng cao trải nghiệm người dùng, tương thích với mọi thiết bị di động và mở rộng phạm vi ứng dụng.

**DANH MỤC TÀI LIỆU THAM KHẢO**

[1] D. Nguyễn, “Công nghệ phần mềm là gì? Và những thông tin quan trọng khác,” Soft Supplier, Apr. 23 2021. [Online]. Available: <https://softsupplier.com/tong-hop-ve-cong-nghe-phan-mem/>

[2] BAC, “Hướng dẫn chi tiết về mô hình Agile trong công nghệ phần mềm,” BACS.vn, n.d. [Online]. Available: <https://www.bacs.vn/vi/blog/kien-thuc/huong-dan-chi-tiet-ve-mo-hinh-agile-trong-cong-nghe-phan-mem-33846.html>

[3] V. Quốc, “Microservices là gì? Kiến trúc, lợi ích và cách triển khai chi tiết,” DevOps.vn, Sep. 29 2019. [Online]. Available: <https://devops.vn/posts/microservices-la-gi-kien-truc-loi-ich-va-cach-trien-khai-chi-tiet/>

[4] T. T. Hai, “Tổng quan về đảm bảo chất lượng phần mềm,” Viblo, May 29, 2019. [Online]. Available: https://viblo.asia/p/tong-quan-ve-dam-bao-chat-luong-phan-mem-al5XRBbLRqPe. [Accessed: Jul. 19, 2025]