**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**



**ISO 9001:2015**

**XÂY DỰNG WEBSITE BÁN LAPTOP**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

Giảng viên hướng dẫn: **Ths. VÕ THÀNH C**

Sinh viên thực hiện: **NGUYỄN TÍN THÀNH**

Mã số sinh viên: **110121104**

Lớp: **DA21TTA**

Khoá: **2021**

**TRÀ VINH, NĂM 2025**

**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

A blue and white logo

AI-generated content may be incorrect.

**ISO 9001:2015**

**XÂY DỰNG WEBSITE BÁN LAPTOP**

**ĐỒ ÁN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

Giảng viên hướng dẫn: **Ths. VÕ THÀNH C**

Sinh viên thực hiện: **NGUYỄN TÍN THÀNH**

Mã số sinh viên: **110121104**

Lớp: **DA21TTA**

Khoá: **2021**

**TRÀ VINH, NĂM 2025**

**MỤC LỤC**

**LỜI MỞ ĐẦU**

**LỜI CẢM ƠN**

**NHẬN XÉT**

**(Của giảng viên hướng dẫn trong đồ án, khoá luận của sinh viên)**

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Giảng viên hướng dẫn**

(ký và ghi rõ họ tên)

UBND TỈNH TRÀ VINH **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TRÀ VINH Độc lập - Tự do - Hạnh phúc**

**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**

*(Của giảng viên hướng dẫn)*

Họ và tên sinh viên: Nguyễn Tín Thành MSSV: 110121104

Ngành: Công nghệ Thông tin Khóa: 2021

Tên đề tài: Xây dựng website bán laptop

Họ và tên Giáo viên hướng dẫn: Võ Thành C

Chức danh: Giảng viên Học vị: Thạc sĩ

**NHẬN XÉT**

1. Nội dung đề tài:

1. Ưu điểm:

1. Khuyết điểm:

1. Điểm mới đề tài:

1. Giá trị thực trên đề tài:

7. Đề nghị sửa chữa bổ sung:

8. Đánh giá:

Trà Vinh*, ngày tháng năm 20…*

Giảng viên hướng dẫn

*(Ký & ghi rõ họ tên)*

**NHẬN XÉT**

**(Của giảng viên chấm trong đồ án, khoá luận của sinh viên)**

**Giảng viên chấm**

(ký và ghi rõ họ tên)

UBND TỈNH TRÀ VINH **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TRÀ VINH Độc lập - Tự do - Hạnh phúc**

**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**

*(Của cán bộ chấm đồ án, khóa luận)*

Họ và tên người nhận xét:

Chức danh: Học vị:

Chuyên ngành:

Cơ quan công tác:

Tên sinh viên:

Tên đề tài đồ án, khóa luận tốt nghiệp:

**I. Ý KIẾN NHẬN XÉT**

1. Nội dung:

2. Điểm mới các kết quả của đồ án, khóa luận:

3. Ứng dụng thực tế:

**II. CÁC VẤN ĐỀ CẦN LÀM RÕ**

(Các câu hỏi của giáo viên phản biện)

**III. KẾT LUẬN**

(Ghi rõ đồng ý hay không đồng ý cho bảo vệ đồ án khóa luận tốt nghiệp)

*……………, ngày …… tháng …… năm 20…*

Người nhận xét

*(Ký & ghi rõ họ tên)*

**DANH MỤC HÌNH ẢNH**

**DANH MỤC BẢNG BIỂU**

**DANH MỤC TỪ VIẾT TẮT**

# GIỚI THIỆU ĐỀ TÀI

## Đặt vấn đề

Trong bối cảnh xã hội hiện đại không ngừng phát triển và hội nhập, sự bùng nổ của công nghệ thông tin đã kéo theo những thay đổi sâu sắc trong mọi lĩnh vực, đặc biệt là thương mại. Thương mại điện tử (TMĐT) đã trở thành một xu thế tất yếu, định hình lại cách thức giao dịch mua bán, trao đổi hàng hóa và dịch vụ. Sự tiện lợi, khả năng tiếp cận không giới hạn về không gian và thời gian, chỉ với một thiết bị thông minh có kết nối internet, đã biến TMĐT thành kênh phân phối hàng đầu, mang lại lợi thế cạnh tranh đáng kể cho doanh nghiệp và người tiêu dùng.

Trong thị trường TMĐT sôi động hiện nay, ngành hàng công nghệ - đặc biệt là laptop - luôn giữ một vai trò quan trọng nhờ nhu cầu sử dụng ngày càng cao từ học sinh, sinh viên đến nhân viên văn phòng, lập trình viên và doanh nhân. Nhu cầu này tạo ra một cơ hội lớn cho các doanh nghiệp triển khai hệ thống kinh doanh trực tuyến chuyên biệt, hiệu quả để tiếp cận và phục vụ khách hàng một cách tối ưu.

Tuy nhiên, để một website bán hàng thực sự hiệu quả và có khả năng cạnh tranh trong môi trường TMĐT ngày càng khốc liệt, việc chỉ cung cấp các tính năng cơ bản như giỏ hàng, thanh toán hay quản lý đơn hàng là chưa đủ. Người dùng hiện đại kỳ vọng một trải nghiệm mua sắm cá nhân hóa, thông minh và thuận tiện hơn. Trong bối cảnh đó, việc tích hợp Trí tuệ nhân tạo (AI) trở thành một giải pháp đột phá, giúp tối ưu hóa trải nghiệm người dùng thông qua các tính năng như hệ thống gợi ý sản phẩm, chatbot hỗ trợ khách hàng, phân tích hành vi mua sắm, và quản lý tồn kho thông minh.

Về mặt kiến trúc hệ thống, dự án sử dụng kiến trúc phân lớp kết hợp với các công nghệ hiện đại nhằm đảm bảo hiệu năng, bảo mật và khả năng mở rộng trong tương lai. Cụ thể:

* Backend được xây dựng bằng Node.js với framework Express, kết nối với MySQL để lưu trữ và truy xuất dữ liệu.
* Frontend sử dụng React.js, một thư viện JavaScript phổ biến để phát triển các ứng dụng web hiện đại, tối ưu hiệu suất và nâng cao trải nghiệm người dùng.
* Hệ thống còn tích hợp AI ở một số tính năng như gợi ý sản phẩm dựa trên lịch sử mua hàng, giúp tăng mức độ hài lòng và giữ chân khách hàng.
* RESTful API được thiết kế để dễ dàng mở rộng và tích hợp với các dịch vụ khác.

Từ những phân tích trên, đề tài “xây dựng Website bán laptop” được đề xuất với mục tiêu phát triển một hệ thống thương mại điện tử hiện đại, thông minh, đáp ứng kỳ vọng của người tiêu dùng trong thời đại số, đồng thời mang lại hiệu quả vận hành và lợi thế cạnh tranh cho doanh nghiệp.

## Mục đích

Đề tài “xây dựng Website bán laptop” được thực hiện với mục tiêu nghiên cứu và phát triển một nền tảng thương mại điện tử hiện đại, thông minh và có khả năng mở rộng. Trang web đảm bảo có khả năng đáp ứng các chức năng cơ bản của một hệ thống bán hàng trực tuyến để nâng cao trải nghiệm người dùng, tối ưu hóa hoạt động vận hành, đồng thời mang lại lợi thế cạnh tranh cho doanh nghiệp trong kỷ nguyên số.

### Đối với người dùng cuối

Người dùng là trung tâm của hệ thống, là nền tảng của một cửa hàng nếu muốn hoạt động ổn định, do đó đề tài hướng tới việc cung cấp một nền tảng mua sắm trực tuyến đáp ứng nhu cầu thực tế và ngày càng khắt khe của khách hàng hiện đại:

* **Tìm kiếm và lọc nâng cao**: Tối ưu giao diện và chức năng tìm kiếm, giúp người dùng dễ dàng tiếp cận sản phẩm mong muốn.
* **Giao diện trực quan, thân thiện**: Ứng dụng React.js để xây dựng giao diện hiện đại, phản hồi nhanh, dễ sử dụng trên cả desktop và thiết bị di động.
* **Tương tác cộng đồng và đánh giá sản phẩm**: Khuyến khích người dùng chia sẻ trải nghiệm, đóng góp nội dung và hỗ trợ các khách hàng khác thông qua chức năng bình luận, đánh giá.
* **Tăng cường gắn kết thông qua ưu đãi và mini game**: Thêm tính năng trò chơi quay thưởng, nhận mã giảm giá nhằm tạo sự hấp dẫn và giữ chân khách hàng lâu dài.

### Đối với người quản trị

Hệ thống cung cấp một bảng điều khiển chuyên nghiệp và đầy đủ tính năng, hỗ trợ quản lý vận hành hiệu quả:

* **Quản lý sản phẩm và danh mục linh hoạt**: Dễ dàng thêm, sửa, xóa, tìm kiếm và kiểm kê sản phẩm/danh mục.
* **Quản lý người dùng và nội dung cộng đồng**: Duyệt bình luận, xử lý báo cáo vi phạm, quản lý thông tin người dùng và quyền truy cập.
* **Theo dõi và xử lý đơn hàng**: Kiểm soát toàn bộ quá trình mua hàng từ đặt hàng, thanh toán, giao nhận.
* **Quản lý mã giảm giá và chương trình khuyến mãi**: Tạo, chỉnh sửa, theo dõi hiệu quả sử dụng mã khuyến mãi, tự động hủy mã hết hạn.

### Mục tiêu kỹ thuật

* **Kiến trúc linh hoạt, dễ mở rộng**: Ứng dụng kiến trúc phân lớp với RESTful API để hỗ trợ các dịch vụ trong tương lai như microservices.
* **Bảo mật người dùng:** Áp dụng các kỹ thuật bảo mật cơ bản như mã hóa mật khẩu bằng bcrypt, xác thực người dùng bằng JWT (JSON Web Token), và phân quyền truy cập theo vai trò (user, admin).
* **Tối ưu hiệu năng giao diện người dùng:** Sử dụng React.js để xây dựng giao diện nhanh, mượt mà, kết hợp lazy loading, phân trang, debounce tìm kiếm nhằm tăng tốc độ phản hồi và giảm tải server.
* **Tích hợp cơ sở dữ liệu quan hệ:** Sử dụng MySQL để lưu trữ dữ liệu có cấu trúc như sản phẩm, đơn hàng, người dùng,... kết hợp prisma ORM hoặc truy vấn SQL thuần tùy theo yêu cầu của hệ thống.

# CƠ SỞ LÝ THUYẾT

## Kiến trúc Microservices

### Tổng quan

Microservices Architecture (kiến trúc vi dịch vụ) là một mô hình kiến trúc phần mềm trong đó hệ thống được chia thành nhiều dịch vụ nhỏ, độc lập, mỗi dịch vụ đảm nhiệm một chức năng riêng biệt. Mỗi microservice có thể được phát triển, triển khai và mở rộng một cách riêng rẽ, cho phép tối ưu hóa hiệu năng, khả năng mở rộng và tính bảo trì của toàn hệ thống.

Việc lựa chọn xây dựng hệ thống theo hướng microservices, sử dụng Node.js làm nền tảng phát triển phía backend và MySQL làm hệ quản trị cơ sở dữ liệu cho từng dịch vụ sẽ giúp dễ dàng triển khai các dịch vụ mà không bị xung đột đối với các dịch vụ khác

#### Đặc điểm của kiến trúc Microservices

* **Tách biệt chức năng**: Mỗi microservice đảm nhiệm một chức năng cụ thể như quản lý người dùng, sản phẩm, đơn hàng, giỏ hàng,...
* **Triển khai độc lập**: Các dịch vụ có thể được build và triển khai riêng biệt mà không ảnh hưởng đến các dịch vụ khác.
* **Cơ sở dữ liệu riêng biệt**: Mỗi service có cơ sở dữ liệu riêng (ở đây là MySQL), giúp đảm bảo tính toàn vẹn dữ liệu và tránh phụ thuộc giữa các service.
* **Giao tiếp qua giao thức HTTP hoặc Message Queue:** Các service giao tiếp với nhau thông qua RESTful API hoặc các hàng đợi tin nhắn như RabbitMQ, Kafka trong các tình huống cần xử lý bất đồng bộ.

### Ưu điểm

* **Phân chia rõ ràng, dễ quản lý:** Kiến trúc Microservices giúp chia hệ thống lớn thành các dịch vụ nhỏ, độc lập, mỗi dịch vụ phụ trách một chức năng cụ thể. Việc này giúp quản lý code dễ dàng hơn, giảm độ phức tạp cho từng thành phần, và tăng khả năng làm việc theo nhóm.
* **Dễ dàng mở rộng (scalability):** Do mỗi dịch vụ được tách biệt, hệ thống có thể mở rộng linh hoạt từng service tùy theo nhu cầu về tài nguyên hoặc lưu lượng người dùng. Ví dụ, nếu service xử lý đơn hàng có nhu cầu cao, chỉ cần mở rộng riêng service này mà không ảnh hưởng đến toàn bộ hệ thống.
* **Triển khai độc lập:** Mỗi dịch vụ có thể được triển khai riêng biệt, không cần chờ các phần còn lại của hệ thống. Điều này hỗ trợ mạnh mẽ cho các quy trình phát triển theo DevOps và triển khai liên tục (CI/CD).
* **Tăng tính ổn định:** Một lỗi xảy ra trong một service sẽ không làm sập toàn bộ hệ thống, giúp hệ thống duy trì tính sẵn sàng cao hơn. Khả năng cô lập lỗi tốt hơn cũng giúp quá trình bảo trì và khắc phục sự cố dễ dàng hơn.
* **Dễ áp dụng công nghệ đa dạng:** Các service có thể sử dụng ngôn ngữ lập trình hoặc công nghệ riêng phù hợp với nghiệp vụ của chúng. Trong đề tài này, nhóm sử dụng Node.js và MySQL nhưng vẫn có thể tích hợp các công nghệ khác nếu cần thiết mà không ảnh hưởng đến tổng thể kiến trúc.

### Nhược điểm

* **Độ phức tạp cao**: Việc phân tách hệ thống thành nhiều service làm gia tăng độ phức tạp trong quản lý, triển khai và vận hành. Hệ thống cần các công cụ hỗ trợ theo dõi logs, giám sát, phân phối cấu hình, đồng bộ dữ liệu và đảm bảo giao tiếp ổn định giữa các service.
* **Khó khăn trong quản lý giao tiếp giữa các service:** Việc gọi API giữa các service (service-to-service communication) có thể dẫn đến vấn đề về độ trễ (latency), lỗi mạng, hoặc lỗi timeout. Nếu không xử lý tốt, các service sẽ dễ bị phụ thuộc vào nhau, đi ngược lại nguyên tắc tách biệt.
* **Yêu cầu cao về DevOps**: Microservices đòi hỏi môi trường DevOps mạnh mẽ để tự động hóa việc build, test, deploy và monitor từng service. Với những nhóm chưa có kinh nghiệm hoặc thiếu công cụ phù hợp, việc triển khai có thể gặp khó khăn.
* **Khó đảm bảo tính nhất quán dữ liệu (data consistency):** Vì mỗi service có cơ sở dữ liệu riêng, việc đảm bảo tính toàn vẹn dữ liệu khi thực hiện các thao tác liên quan đến nhiều service trở nên khó khăn. Việc triển khai các mô hình như Saga hoặc Eventual Consistency là cần thiết nhưng không đơn giản.
* **Chi phí tài nguyên cao hơn:** Việc vận hành nhiều service chạy song song có thể làm tăng tài nguyên hệ thống, yêu cầu nhiều container hoặc máy chủ hơn so với kiến trúc nguyên khối (monolithic), dẫn đến chi phí cao hơn nếu không tối ưu tốt**.**

## Restful API

### Tổng quan

**RESTful API** (Representational State Transfer) là một phong cách kiến trúc phần mềm được sử dụng phổ biến trong việc phát triển các hệ thống dịch vụ web, đặc biệt phù hợp với các hệ thống phân tán như **Microservices**. REST định nghĩa một bộ quy tắc để xây dựng API sử dụng các phương thức HTTP tiêu chuẩn như **GET**, **POST**, **PUT**, **DELETE** nhằm thao tác với tài nguyên (resources) được đại diện dưới dạng dữ liệu (thường là JSON).

Trong kiến trúc RESTful API, mỗi tài nguyên (ví dụ: sản phẩm, người dùng, đơn hàng...) được biểu diễn bởi một URL duy nhất. Các thao tác tương ứng sẽ được thực hiện thông qua HTTP method. Điều này giúp API trở nên đơn giản, dễ hiểu và thân thiện với lập trình viên front-end.

RESTful API đặc biệt phù hợp với **kiến trúc Microservices**, nơi mỗi dịch vụ (service) phụ trách một phần chức năng riêng biệt, và giao tiếp với nhau thông qua HTTP.

### Đặc điểm

* **Dựa trên giao thức HTTP:** gồm các phương thức là GET, POST, PUT, DELETE và PATCH
* **Mỗi tài nguyên có một URL duy nhất:** ví dụ như api/product là một URL duy nhất
* **Sử dụng JSON để trao đổi dữ liệu:** JSON là định dạng phổ biến để truyền dữ liệu giữa client và server trong RESTful API vì nó nhẹ, dễ đọc và dễ xử lý ở cả hai phía.
* **Stateless (phi trạng thái):** Mỗi request từ client tới server đều độc lập và không lưu trữ trạng thái giữa các request. Điều này giúp hệ thống dễ mở rộng và đơn giản trong việc xử lý.
* **Khả năng phân tách rõ ràng giữa client và server:** Client (giao diện người dùng) và server (logic xử lý) hoạt động độc lập, giao tiếp thông qua API, giúp tăng khả năng mở rộng và bảo trì hệ thống.
* **Dễ dàng tích hợp với hệ thống khác:** RESTful API là chuẩn phổ biến và được hỗ trợ rộng rãi nên dễ tích hợp với các hệ thống, dịch vụ bên ngoài hoặc app mobile.

### Ưu điểm

* **Đơn giản và dễ hiểu**: RESTful API sử dụng các phương thức HTTP cơ bản như GET, POST, PUT, DELETE nên rất trực quan và dễ tiếp cận đối với các lập trình viên, kể cả người mới bắt đầu.
* **Tính linh hoạt cao:** REST không bị ràng buộc bởi ngôn ngữ lập trình hay công nghệ cụ thể, giúp API có thể triển khai bằng bất kỳ backend nào (Node.js, Java, Python...) và dễ dàng tích hợp với nhiều client khác nhau như web, mobile, IoT,...
* **Dễ mở rộng:** RESTful API giúp chia nhỏ hệ thống thành các tài nguyên độc lập, phù hợp với kiến trúc microservices. Việc thêm mới, cập nhật hoặc xoá dịch vụ không ảnh hưởng đến toàn hệ thống.
* **Hiệu suất cao và nhẹ:** RESTful thường sử dụng JSON làm định dạng dữ liệu nên nhẹ hơn so với XML, giúp tiết kiệm băng thông và cải thiện hiệu suất truyền tải dữ liệu.
* **Dễ kiểm thử và debug:** Do RESTful API sử dụng chuẩn HTTP và tài nguyên có URL rõ ràng, nên dễ dàng kiểm thử qua công cụ như Postman, cURL, hoặc tích hợp CI/CD để tự động hoá việc kiểm tra.

### Nhược điểm

* **Thiếu tính chuẩn hóa nghiêm ngặt:** REST chỉ là một phong cách kiến trúc nên không có quy chuẩn chặt chẽ, điều này dẫn đến các dự án RESTful API có thể triển khai không nhất quán nếu không có định hướng hoặc quy ước rõ ràng từ đầu.
* **Không hỗ trợ tốt các thao tác phức tạp**: RESTful API có thể gặp khó khăn khi xử lý các yêu cầu có logic nghiệp vụ phức tạp, chẳng hạn như thao tác đồng thời trên nhiều tài nguyên (transactional logic), vì REST thường chỉ thao tác từng tài nguyên một cách độc lập.
* **Không có tính trạng thái (Stateless):** Mặc dù stateless giúp đơn giản hóa và mở rộng dễ dàng, nhưng đồng thời cũng gây khó khăn khi cần lưu trữ phiên người dùng hoặc các thao tác liên quan đến trạng thái dài hạn.
* **Quản lý version phức tạp:** Khi hệ thống mở rộng và thay đổi, việc quản lý version API (v1, v2,...) cần có chiến lược rõ ràng nếu không sẽ gây rối loạn hoặc xung đột cho các client đang sử dụng.

## Ngôn ngữ backend

### Nodejs

Node.js là một môi trường thực thi JavaScript phía máy chủ (server-side), được xây dựng dựa trên công cụ V8 JavaScript Engine của Google. Được phát triển lần đầu vào năm 2009 bởi Ryan Dahl, Node.js cho phép JavaScript – vốn là một ngôn ngữ chỉ chạy trên trình duyệt – có thể được sử dụng để xây dựng các ứng dụng backend mạnh mẽ, hiệu quả và dễ mở rộng.

Node.js áp dụng mô hình **non-blocking I/O** (nhập xuất không đồng bộ) và **event-driven** (dựa trên sự kiện), giúp tối ưu hoá hiệu suất xử lý trong các ứng dụng có nhiều kết nối đồng thời, như API backend, hệ thống chat, hoặc ứng dụng real-time.

Nhờ sử dụng cùng một ngôn ngữ JavaScript cho cả frontend và backend, Node.js giúp giảm sự phân mảnh về công nghệ trong một dự án, đồng thời tăng tốc độ phát triển và bảo trì hệ thống.

Một số ưu điểm nổi bật của Node.js:

* Hiệu năng cao: Nhờ vào engine V8 và cơ chế I/O không đồng bộ.
* Dễ mở rộng: Thích hợp cho các kiến trúc như microservices hoặc serverless.
* Hệ sinh thái phong phú: Có sẵn hàng nghìn package thông qua npm (Node Package Manager).
* Tính nhất quán về ngôn ngữ: Giúp các đội ngũ frontend có thể dễ dàng làm việc với backend.

Node.js hiện là một trong những công nghệ backend phổ biến nhất, được sử dụng bởi nhiều công ty lớn như Netflix, LinkedIn, Uber và PayPal.

### Express.js

Express.js là một framework tối giản và linh hoạt dành cho Node.js, giúp xây dựng các ứng dụng web và API một cách nhanh chóng. Express hỗ trợ đầy đủ các tính năng để xây dựng một hệ thống RESTful API chuyên nghiệp và là một lựa chọn phổ biến trong các dự án Node.js.

Ưu điểm của Express:

* **Cú pháp đơn giản, dễ dùng**: Giúp khởi tạo nhanh các endpoint HTTP (GET, POST, PUT, DELETE).
* **Middleware mạnh mẽ:** Cho phép xử lý logic giữa các bước như xác thực, logging, kiểm tra lỗi,...
* **Dễ dàng mở rộng**: Có thể tích hợp các thư viện khác như JWT, multer, morgan,... để mở rộng chức năng.
* **Tối ưu cho Microservices**: Dễ chia nhỏ thành các service độc lập với khả năng mở rộng linh hoạt.

## Ngôn ngữ frontend

### Reactjs

ReactJS là một thư viện JavaScript mã nguồn mở được phát triển và duy trì bởi Facebook từ năm 2013. React được thiết kế nhằm hỗ trợ việc xây dựng giao diện người dùng (UI) một cách hiệu quả, đặc biệt trong các ứng dụng web có tính tương tác cao. Với triết lý component-based (dựa trên thành phần), React giúp chia nhỏ giao diện thành các thành phần độc lập, có thể tái sử dụng và dễ bảo trì.

Khác với các framework toàn diện như Angular, React chỉ tập trung vào tầng View trong mô hình MVC (Model – View – Controller). Tuy nhiên, nhờ hệ sinh thái phong phú (React Router, Redux, Zustand, React Query, Material UI,...), React có thể dễ dàng mở rộng để đáp ứng đầy đủ nhu cầu phát triển các ứng dụng web hiện đại.

#### Ưu điểm của ReactJS:

* Component hóa mạnh mẽ: React cho phép chia nhỏ giao diện thành nhiều component nhỏ, độc lập, có thể tái sử dụng, từ đó tăng khả năng quản lý và bảo trì mã nguồn.
* Virtual DOM: React sử dụng cơ chế Virtual DOM để cập nhật giao diện một cách tối ưu, giúp tăng hiệu suất và giảm chi phí tái render không cần thiết.
* Khả năng tái sử dụng và mở rộng cao: Các component React có thể dễ dàng được tái sử dụng trong nhiều phần khác nhau của ứng dụng hoặc thậm chí trong các dự án khác.
* JSX – cú pháp mở rộng: JSX kết hợp giữa JavaScript và HTML giúp việc viết UI trở nên trực quan, đồng thời tăng tính mô-đun và dễ kiểm soát logic giao diện.
* Cộng đồng lớn mạnh và tài liệu phong phú: React có một cộng đồng người dùng rộng lớn cùng với tài liệu chính thức chi tiết, hỗ trợ quá trình học tập và phát triển ứng dụng dễ dàng hơn.
* Thích hợp với SPA (Single Page Application): React rất phù hợp để xây dựng các ứng dụng một trang, nơi người dùng có thể tương tác mà không cần tải lại trang.
* Tích hợp tốt với RESTful API hoặc GraphQL: React có thể kết nối dữ liệu dễ dàng thông qua các thư viện như Axios, Fetch, hay Apollo Client, phù hợp với các kiến trúc backend hiện đại.

#### Nhược điểm của ReactJS:

* Chỉ tập trung vào View: React không cung cấp sẵn các giải pháp về routing, state management, hay form validation, buộc lập trình viên phải chọn thư viện bên ngoài, dẫn đến sự phân mảnh và cần thời gian để lựa chọn công nghệ phù hợp.
* Cập nhật thường xuyên: React và hệ sinh thái đi kèm cập nhật liên tục. Mặc dù giúp cải tiến tính năng và hiệu năng, nhưng cũng đòi hỏi nhà phát triển phải luôn cập nhật kiến thức, nhất là khi có thay đổi lớn (ví dụ như hooks, concurrent mode).
* Khó khăn cho người mới: Việc kết hợp giữa JavaScript, JSX, hooks, context và các thư viện ngoài như Redux có thể khiến người mới bắt đầu cảm thấy khó khăn khi tiếp cận và xây dựng ứng dụng React hoàn chỉnh.
* Quản lý state phức tạp: Trong các ứng dụng lớn, việc quản lý trạng thái (state) toàn cục có thể trở nên phức tạp nếu không có kiến trúc rõ ràng.

### Material UI library

**Material UI** (thường được viết tắt là MUI) là một thư viện giao diện người dùng (UI) nổi bật dành cho React, được xây dựng dựa trên triết lý thiết kế Material Design của Google. Thư viện này cung cấp một tập hợp các component UI phong phú, dễ sử dụng, có tính nhất quán cao và khả năng tuỳ biến linh hoạt, giúp tăng tốc quá trình xây dựng giao diện người dùng hiện đại cho các ứng dụng web.

Không giống như việc tự thiết kế từng thành phần từ đầu, Material UI mang đến một bộ công cụ thiết kế hoàn chỉnh với các thành phần như nút bấm, thanh điều hướng, hộp thoại, biểu mẫu, bảng dữ liệu, biểu tượng, và nhiều hơn nữa – tất cả đều tuân thủ chặt chẽ chuẩn thiết kế Material Design. Với việc tích hợp sâu sắc vào React và khả năng mở rộng mạnh mẽ, MUI trở thành lựa chọn phổ biến trong cộng đồng phát triển front-end.

#### Ưu điểm:

Một trong những điểm mạnh lớn nhất của Material UI là khả năng **tăng tốc độ phát triển giao diện**. Các component đã được tối ưu sẵn về mặt giao diện lẫn hành vi, giúp lập trình viên không phải tốn thời gian "mò thiết kế" từ đầu, đồng thời vẫn đảm bảo tính thẩm mỹ và sự thống nhất trên toàn hệ thống.

Material UI cũng **hỗ trợ tuỳ biến rất tốt**. Nhờ hệ thống theme linh hoạt, nhà phát triển có thể dễ dàng thay đổi màu sắc, kiểu chữ, kích thước, hình dạng component, thậm chí override style chi tiết bằng hệ thống CSS-in-JS (Emotion hoặc Styled Components). Điều này giúp MUI phù hợp với cả các dự án cá nhân lẫn các hệ thống thương mại có yêu cầu cao về thương hiệu (branding).

Bên cạnh đó, Material UI đi kèm với **bộ icon phong phú** (Material Icons), hỗ trợ hiệu ứng chuyển động mượt mà (animation), khả năng responsive cao, và tích hợp tốt với các thư viện quản lý trạng thái như Redux hoặc Context API.

Một điểm cộng nữa là **cộng đồng người dùng rộng lớn**, tài liệu rõ ràng, ví dụ dễ hiểu và có sự hỗ trợ từ đội ngũ phát triển chuyên nghiệp. Việc cập nhật phiên bản và các tính năng mới cũng diễn ra thường xuyên, giúp đảm bảo tính ổn định lâu dài.

#### Nhược điểm:

Tuy sở hữu nhiều ưu điểm, Material UI cũng tồn tại một số điểm hạn chế nhất định. Thứ nhất, **kích thước bundle tương đối lớn** nếu chỉ sử dụng một vài component, điều này có thể ảnh hưởng đến hiệu suất nếu không cấu hình tree-shaking hoặc lazy load hợp lý.

Thứ hai, việc **tuỳ biến nâng cao** đôi khi yêu cầu người dùng phải hiểu rõ cấu trúc bên trong của component MUI, đặc biệt khi cần override sâu CSS hoặc áp dụng các kỹ thuật như slot customization. Điều này có thể gây khó khăn cho người mới bắt đầu hoặc những ai chưa quen với hệ thống CSS-in-JS.

Cuối cùng, **Material Design** tuy rất phổ biến, nhưng đôi khi không phù hợp với một số sản phẩm yêu cầu thiết kế mang tính cá nhân hoá cao hoặc đi theo hướng "phá cách", bởi phong cách Material khá đặc trưng và có phần ràng buộc về mặt hình thức.

Tóm lại, Material UI là một lựa chọn mạnh mẽ cho việc xây dựng giao diện React hiện đại, giúp giảm thiểu thời gian phát triển, tăng tính nhất quán trong UI, đồng thời vẫn giữ được sự linh hoạt trong tuỳ biến và mở rộng.

## Cơ sở dữ liệu MySQL

### Khái niệm

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System – RDBMS) mã nguồn mở được phát triển ban đầu bởi công ty MySQL AB, sau đó được Sun Microsystems mua lại và hiện nay thuộc quyền sở hữu của Oracle Corporation. MySQL nổi bật nhờ hiệu suất cao, dễ sử dụng và khả năng mở rộng tốt, là lựa chọn phổ biến cho các hệ thống web, đặc biệt là trong kiến trúc LAMP (Linux – Apache – MySQL – PHP/Python/Perl).

Không giống như cái tên dễ gây nhầm lẫn là "MySQL server" ám chỉ phần cứng, MySQL thực chất là một phần mềm hệ quản trị cơ sở dữ liệu. Nó hoạt động như một “máy chủ dữ liệu” có nhiệm vụ tiếp nhận các yêu cầu từ ứng dụng (client), xử lý truy vấn và trả về kết quả tương ứng. Việc này cho phép các ứng dụng như hệ thống quản lý, trang web thương mại điện tử, blog, hoặc nền tảng phân tích dữ liệu vận hành trên một cơ sở dữ liệu tập trung, nhanh chóng và an toàn.

MySQL sử dụng ngôn ngữ SQL (Structured Query Language) để thao tác và truy vấn dữ liệu. Đây là một ngôn ngữ chuẩn, phổ biến trên hầu hết các hệ quản trị cơ sở dữ liệu quan hệ. Dù không có hệ mở rộng như T-SQL của SQL Server, MySQL vẫn hỗ trợ đầy đủ các tính năng cần thiết như stored procedures, triggers, views và transaction để xử lý các nghiệp vụ phức tạp trong thực tế.

### Cấu trúc

MySQL có kiến trúc mô-đun rõ ràng, giúp dễ dàng mở rộng và tối ưu hóa, bao gồm các thành phần chính như sau:

* **Connection Manager & Protocol Layer**: Là tầng đầu tiên tiếp nhận kết nối từ client. MySQL hỗ trợ nhiều giao thức như TCP/IP, socket Unix, hoặc Named Pipes (trên Windows). Tầng này đảm bảo giao tiếp hiệu quả giữa client và server.
* **SQL Layer**: Đây là nơi xử lý tất cả các truy vấn SQL. Các chức năng chính bao gồm phân tích cú pháp, lập kế hoạch thực thi (query optimization), kiểm tra quyền truy cập, và xử lý các biểu thức SQL.
* **Storage Engine Layer**: Đây là tầng cốt lõi phụ trách việc lưu trữ, truy xuất và ghi dữ liệu vào ổ cứng. MySQL hỗ trợ nhiều loại storage engine như:
  + InnoDB: Engine mặc định, hỗ trợ ACID, khóa ngoại (foreign key), transaction, và row-level locking.
  + MyISAM: Engine truyền thống, nhẹ hơn nhưng không hỗ trợ transaction hay khóa ngoại.
  + Ngoài ra còn có MEMORY, ARCHIVE, CSV, v.v., tùy thuộc vào nhu cầu sử dụng.
* **File System Layer**: Là tầng giao tiếp cuối cùng giữa MySQL và hệ điều hành, đảm nhận nhiệm vụ đọc/ghi dữ liệu vào tệp vật lý, quản lý bảng, chỉ mục, log file, và các dữ liệu hệ thống khác.

Một đặc điểm quan trọng của MySQL là thiết kế kiến trúc plug-in storage engine, cho phép người dùng lựa chọn hoặc thậm chí phát triển engine phù hợp với mục đích sử dụng cụ thể.

## Docker

### Khái niệm

Trong bối cảnh phát triển phần mềm hiện đại, nơi các ứng dụng cần được triển khai linh hoạt và nhanh chóng trên nhiều môi trường khác nhau, Docker nổi lên như một nền tảng mã nguồn mở mạnh mẽ, cho phép tự động hóa việc đóng gói, phân phối và triển khai ứng dụng dưới dạng các container. Với triết lý “viết một lần, chạy ở mọi nơi”, Docker đã trở thành một công cụ không thể thiếu trong hệ sinh thái DevOps và triển khai phần mềm.

Về bản chất, Docker ứng dụng công nghệ container hóa – một phương pháp đóng gói toàn bộ mã nguồn, thư viện, môi trường runtime và các phụ thuộc khác của ứng dụng vào một đơn vị độc lập gọi là container. Chính khả năng đóng gói này đảm bảo rằng ứng dụng sẽ hoạt động nhất quán bất kể môi trường triển khai là máy tính cá nhân, máy chủ cục bộ, hay đám mây.

Một trong những thành phần cốt lõi của Docker là Docker Engine – phần mềm nền tảng giúp tạo, quản lý và chạy các container. Docker Engine bao gồm ba thành phần chính: đầu tiên là Docker Daemon, một tiến trình chạy nền có nhiệm vụ xử lý các thao tác khởi tạo, dừng, quản lý và xóa container. Tiếp theo là Docker CLI (Command Line Interface), giao diện dòng lệnh cho phép lập trình viên và quản trị viên hệ thống tương tác với Docker một cách hiệu quả thông qua các lệnh đơn giản. Cuối cùng là Docker API, cung cấp khả năng truy cập và điều khiển Docker thông qua các giao thức HTTP, giúp tích hợp Docker vào các hệ thống hoặc công cụ khác.

Bên cạnh Docker Engine, khái niệm Docker Container giữ vai trò trung tâm trong toàn bộ kiến trúc. Mỗi container là một môi trường tách biệt, nơi một ứng dụng có thể chạy độc lập cùng với tất cả các thành phần cần thiết đi kèm. Mặc dù các container này chia sẻ chung hạt nhân hệ điều hành (kernel), chúng vẫn được cách ly về mặt quy trình, hệ thống tệp và tài nguyên, từ đó đảm bảo tính an toàn và ổn định trong vận hành.

Để tạo ra các container, Docker sử dụng Docker Image – một bản mẫu không thay đổi, chứa toàn bộ hệ điều hành, cấu hình, mã ứng dụng và các phần mềm hỗ trợ cần thiết. Các image này được xây dựng từ một tập tin có tên Dockerfile, nơi định nghĩa các bước cần thiết để thiết lập môi trường chạy cho ứng dụng. Dockerfile có thể được ví như một “công thức” để xây dựng image, cho phép tái tạo container một cách nhất quán và có thể kiểm soát được.

Ngoài ra, Docker còn cung cấp một nền tảng chia sẻ và phân phối các image thông qua Docker Hub – một kho lưu trữ trực tuyến cho phép người dùng tải về các image đã được chuẩn bị sẵn, hoặc chia sẻ những image do họ tự xây dựng. Docker Hub đóng vai trò như một chợ ứng dụng, giúp tiết kiệm thời gian và tối ưu quy trình phát triển.

### Lợi ích

Một trong những lợi thế nổi bật của Docker chính là khả năng khởi tạo và dừng container một cách nhanh chóng chỉ trong vài giây, giúp tăng tốc quá trình phát triển và triển khai ứng dụng. Không những vậy, Docker có thể chạy nhất quán trên nhiều hệ điều hành và nền tảng khác nhau, từ máy cá nhân đến hệ thống máy chủ hay nền tảng đám mây, nhờ kiến trúc container hóa độc lập với môi trường. Đặc biệt, Docker cho phép thiết lập môi trường làm việc một cách dễ dàng và ổn định, khi toàn bộ cấu hình và các phụ thuộc chỉ cần cài đặt một lần duy nhất trong file cấu hình (Dockerfile hoặc docker-compose), từ đó loại bỏ hoàn toàn nhu cầu cài lại phần mềm hay thư viện mỗi khi chuyển sang máy khác hoặc làm việc nhóm.

## Github Action

### Khái niệm

GitHub Actions là một nền tảng CI/CD tích hợp sẵn trên GitHub, cho phép tự động hóa toàn bộ quy trình phát triển phần mềm từ xây dựng, kiểm thử, đến triển khai. Với GitHub Actions, bạn có thể tạo các workflow (quy trình làm việc) để tự động thực hiện các tác vụ khi có các sự kiện cụ thể xảy ra trong repository của bạn.

### Các thành phần chính

Workflow: Là một tập hợp các quy trình tự động hóa được định nghĩa trong repository GitHub, giúp quản lý và thực hiện các công việc như build, test, và deploy phần mềm. Workflows được cấu hình thông qua các tệp YAML và lưu trữ trong thư mục .github/workflows của repository

Jobs: Là các tập hợp các bước (steps) cần thực hiện trong workflow. Mỗi job chạy trên một runner riêng biệt, có thể chạy song song hoặc theo thứ tự phụ thuộc lẫn nhau.

Runners: Là các máy chủ hoặc máy ảo nơi các jobs trong workflow được thực thi. GitHub cung cấp runners mặc định (hosted runners) và bạn cũng có thể tự thiết lập runners của riêng mình (self-hosted runners).

Actions: Là các nhiệm vụ (tasks) được thực thi trong một workflow. Một action có thể là một lệnh shell hoặc một phần mềm được đóng gói sẵn giúp thực hiện các nhiệm vụ phổ biến.

Events: Là các sự kiện kích hoạt workflows. GitHub cung cấp nhiều sự kiện khác nhau, chẳng hạn như khi có push, pull request, hoặc issue mới.

# CHƯƠNG 3 HIỆN THỰC HÓA NGHIÊN CỨU

## Mô tả bài toán

## Yêu cầu chức năng của hệ thống

### Đối với người quản trị

### Đối với người dùng

## Yêu cầu phi chức năng của hệ thống

## Phân tích thiết kế hệ thống

### Đặc tả hệ thống

## Thiết kế giao diện

## Phương pháp xây dựng hệ thống gợi ý

# CHƯƠNG 4 KẾT QUẢ NGHIÊN CỨU

# CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

### Kết quả và đóng góp của đồ án

### Hạn chế

## Hướng phát triển

**PHỤ LỤC**

**DANH MỤC TÀI LIỆU THAM KHẢO**