LỜI MỞ ĐẦU

1. Lý do chọn đề tài

Trong bối cảnh chuyển đổi số đang diễn ra mạnh mẽ trên toàn cầu, thương mại điện tử ngày càng đóng vai trò quan trọng trong việc thay đổi phương thức kinh doanh truyền thống. Việc ứng dụng công nghệ số, đặc biệt là nền tảng di động và API vào hoạt động kinh doanh không chỉ giúp tiết kiệm chi phí, thời gian mà còn mở rộng phạm vi thị trường, tạo kết nối trực tiếp giữa người bán và người mua, từ đó nâng cao trải nghiệm người dùng và hiệu quả giao dịch.

Tại Việt Nam, nhu cầu mua sắm thiết bị công nghệ như điện thoại, laptop, phụ kiện ngày càng gia tăng, đặc biệt ở nhóm khách hàng trẻ và người tiêu dùng tại các thành phố lớn. Tuy nhiên, nhiều doanh nghiệp vẫn đang vận hành theo mô hình truyền thống, thiếu sự tích hợp công nghệ trong quá trình bán hàng, quản lý đơn hàng và tương tác khách hàng, dẫn đến những hạn chế về quy mô, tốc độ và tính minh bạch trong giao dịch.

Trong bối cảnh đó, việc xây dựng một ứng dụng bán thiết bị công nghệ trên nền tảng di động tích hợp với API từ hệ thống quản lý và thanh toán là một hướng đi thiết thực và cấp thiết. Ứng dụng này không chỉ giúp doanh nghiệp chủ động trong việc giới thiệu, bán sản phẩm mà còn tạo điều kiện cho người dùng tiếp cận thiết bị công nghệ chính hãng với thông tin rõ ràng, giá cả minh bạch và quy trình mua sắm tiện lợi. Đồng thời, việc tìm hiểu và khai thác API giúp tăng tính tự động hóa.

2. Mục đích nghiên cứu

Trong đồ án khóa luận tốt nghiệp này, tôi sẽ xây dựng một ứng dụng bán thiết bị công nghệ trên nền tảng di động tích hợp với API.

3. Đối tượng và phạm vi nghiên cứu

Người tiêu dùng: Là những khách hàng có nhu cầu tìm kiếm, tham khảo và mua sắm các thiết bị công nghệ như điện thoại, máy tính bảng, laptop, phụ kiện điện tử... thông qua ứng dụng di động một cách nhanh chóng, tiện lợi và an toàn.

Quản trị viên: Là người chịu trách nhiệm điều phối, giám sát toàn bộ hoạt động của hệ thống, duyệt sản phẩm, quản lý đơn hàng, hỗ trợ người dùng và đảm bảo sự ổn định của nền tảng.

4. Phương pháp nghiên cứu

- Phương pháp lý thuyết:

Tìm hiểu các khái niệm về thương mại điện tử, phát triển ứng dụng di động, kiến trúc client-server và quản lý cơ sở dữ liệu.

Nghiên cứu công nghệ sử dụng gồm: React Native, Node.js, MongoDB và API RESTful.

- Phương pháp thực nghiệm:

Phân tích nhu cầu người dùng và xác định các chức năng cốt lõi: xem sản phẩm, giỏ hàng, đặt hàng, thanh toán, theo dõi đơn.

Xây dựng giao diện bằng React Native đảm bảo trực quan, thân thiện.

Phát triển backend với Node.js, cung cấp API xử lý nghiệp vụ.

Lưu trữ dữ liệu bằng MongoDB.

Tích hợp hỗ trợ người dùng (chatbot hoặc live chat).

Kiểm thử hệ thống và đánh giá hiệu năng để đề xuất cải tiến.

LỜI CẢM ƠN

Tôi xin chân thành cảm ơn đến toàn thể quý Thầy, Cô Khoa Công nghệ Thông tin, Trường Kỹ thuật và Công nghệ đã đã hỗ trợ và tạo điều kiện thuận lợi để tôi hoàn thành đồ án này.

Tôi xin gửi lời cảm ơn đến thầy Trịnh Quốc Việt, nhờ sự hướng dẫn tận tình và kiên nhẫn tôi đã có thể hoàn thành đồ án một cách tốt nhất. Qua đó học hỏi được rất nhiều kiến thức và kỹ năng quý giá, đặc biệt là về kỹ năng xây dựng API và lập trình front end với React, React Native và back end với NodeJS.

Cuối cùng, tôi xin gửi lời cảm ơn sâu sắc đến bạn bè và gia đình đã luôn ủng hộ và khích lệ tôi trong suốt quá trình thực hiện đồ án. Tôi xin chân thành cảm ơn !

MỤC LỤC

NHẬN XÉT

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

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Giảng viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT
(Của cơ quan thực tập, *nếu có*)

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

UBND TỈNH TRÀ VINH
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
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: ..... MSSV: .....
Ngành: ..... Khóa: .....
Tên đề tài: .....
.....
.....
Họ và tên Giáo viên hướng dẫn:.....
Chức danh: ..... Học vị: .....

NHẬN XÉT

1. Nội dung đề tài:

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

2. Ưu điểm:

.....
.....
.....
.....

3. Khuyết điểm:

.....
.....
.....
.....
.....

4. Điểm mới đề tài:

.....
.....
.....
.....
.....

5. 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 tên và ghi rõ họ tên)

UBND TỈNH TRÀ VINH
TRƯỜNG ĐẠI HỌC TRÀ VINH

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
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 CÁC BẢNG, SƠ ĐỒ, HÌNH

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

API	Application Programming Interface
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
URL	Uniform Resource Locator
JWT	JSON Web Token
MVC	Model - View - Controller
MVP	Model - View - Presenter
PDF	Portable Document Format
PHP	Hypertext Preprocessor
UWP	Universal Windows Platform
NFC	Near Field Communication
JSON	JavaScript Object Notation
BSON	Binary JSON
CMS	Content Management System
OTP	One-Time Password

DANH MỤC BẢNG BIỂU

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Trong những năm gần đây, sự phát triển mạnh mẽ của công nghệ và xu hướng mua sắm trực tuyến qua thiết bị di động đã mở ra nhiều cơ hội mới cho lĩnh vực kinh doanh thiết bị công nghệ. Người tiêu dùng ngày càng ưu tiên sự tiện lợi, nhanh chóng và khả năng tiếp cận thông tin sản phẩm mọi lúc, mọi nơi thông qua các ứng dụng di động.

Vì vậy, việc tìm hiểu cách hoạt động của API và xây dựng một ứng dụng bán thiết bị công nghệ trên nền tảng di động là vô cùng cần thiết. Ứng dụng này không chỉ giúp cung cấp thông tin sản phẩm rõ ràng, cập nhật nhanh chóng mà còn mang đến trải nghiệm mua sắm thông minh, thuận tiện và hiện đại, góp phần nâng cao hiệu quả kinh doanh và đáp ứng nhu cầu ngày càng cao của khách hàng.

1.2 Mục đích nghiên cứu

Nghiên cứu này tập trung vào việc thiết kế và phát triển một ứng dụng di động hiện đại, chuyên biệt nhằm phục vụ việc giới thiệu và kinh doanh thiết bị công nghệ. Các mục tiêu cụ thể bao gồm:

Xây dựng một ứng dụng thân thiện với người dùng, giúp khách hàng dễ dàng tra cứu thông tin, so sánh và mua sắm thiết bị công nghệ ngay trên thiết bị di động.

Tích hợp các tính năng thiết yếu như tìm kiếm và lọc sản phẩm, đánh giá và bình luận, quản lý giỏ hàng, danh sách yêu thích, mã giảm giá và thông báo khuyến mãi.

Tìm hiểu và áp dụng các API trong quá trình phát triển, từ việc lấy dữ liệu sản phẩm, xử lý đơn hàng đến tích hợp thanh toán, nhằm nâng cao hiệu quả hoạt động của ứng dụng.

Nâng cao kỹ năng lập trình ứng dụng di động thông qua việc sử dụng React Native cho giao diện người dùng và Node.js cho backend, đáp ứng các yêu cầu kỹ thuật hiện đại và tối ưu hóa trải nghiệm người dùng.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Giới thiệu về NodeJS

2.1.1 Khái niệm NodeJS

NodeJS là một môi trường thực thi đa nền tảng và mã nguồn mở, cho phép phát triển các ứng dụng nhanh chóng và có khả năng mở rộng. Nó được xây dựng trên nền tảng "V8 JavaScript Engine" của Google Chrome. Cùng với đó là cấu trúc I/O non-block và mô hình event-driven (hướng sự kiện) giúp tăng hiệu suất và xử lý các ứng dụng thời gian thực một cách hiệu quả [5].

2.1.2 Ứng dụng của NodeJS

Ứng dụng thời gian thực (Real-time Applications): NodeJS rất phù hợp để xây dựng các ứng dụng cần kết nối và cập nhật liên tục giữa server và client, nhờ cơ chế event-driven và WebSocket.

API RESTful và GraphQL: NodeJS thường được sử dụng để xây dựng các API RESTful hoặc GraphQL, cung cấp dữ liệu cho ứng dụng frontend hoặc mobile.

Ứng dụng Single Page Application (SPA): NodeJS giúp xử lý yêu cầu và dữ liệu phía server hiệu quả, tạo nền tảng tốt cho các SPA.

Ứng dụng IoT (Internet of Things): NodeJS là một lựa chọn tuyệt vời để phát triển các hệ thống IoT nhờ khả năng xử lý sự kiện nhanh và sử dụng ít tài nguyên.

Truyền phát dữ liệu và nội dung (Streaming Applications): NodeJS hỗ trợ truyền phát dữ liệu theo luồng (streaming) hiệu quả, giúp xây dựng các ứng dụng phát trực tiếp hoặc xử lý file lớn.

Ứng dụng thương mại điện tử (E-commerce Applications): Nhờ khả năng xử lý đồng thời hàng nghìn yêu cầu, NodeJS phù hợp với các trang web thương mại điện tử cần tốc độ cao.

Ứng dụng máy chủ Proxy (Proxy Server Applications): NodeJS có thể được sử dụng như một máy chủ proxy để xử lý các yêu cầu giữa client và server khác, đặc biệt khi cần xử lý nhiều dịch vụ hoặc các API bên thứ ba.

Ứng dụng đa nền tảng (Cross-platform Applications): Node.js kết hợp với các công cụ như Electron có thể phát triển ứng dụng desktop chạy đa nền tảng.

2.1.3 Nguyên lý hoạt động của NodeJS

NodeJS hoạt động dựa trên mô hình event-driven (hướng sự kiện) và sử dụng vòng lặp sự kiện (event loop) để xử lý các tác vụ bất đồng bộ. Thay vì tạo nhiều luồng như các mô hình truyền thống (multi-threaded), NodeJS sử dụng một luồng đơn (single-threaded) để xử lý tất cả các yêu cầu. Điều này được thực hiện thông qua cơ chế non-blocking I/O, giúp NodeJS có thể xử lý nhiều kết nối đồng thời mà không bị nghẽn.

Khi một tác vụ I/O (ví dụ: đọc file, truy vấn cơ sở dữ liệu, gọi API bên ngoài) được gửi đi, NodeJS không chờ tác vụ đó hoàn thành. Thay vào đó, nó đăng ký callback và tiếp tục xử lý các yêu cầu khác. Khi tác vụ hoàn thành, một sự kiện được đưa vào hàng đợi (event queue) và được event loop xử lý, gọi callback tương ứng.

NodeJS được xây dựng trên Google Chrome V8 Engine, cho phép biên dịch mã JavaScript trực tiếp thành mã máy, giúp tăng tốc độ thực thi.

Mô hình hoạt động của NodeJS gồm các thành phần chính:

Call Stack: nơi lưu trữ các lời gọi hàm.

Event Queue: hàng đợi các sự kiện cần xử lý.

Event Loop: cơ chế trung tâm, liên tục kiểm tra call stack và event queue, đảm bảo xử lý các callback khi call stack trống.

Worker Threads / Thread Pool: phía dưới tầng NodeJS, các tác vụ nặng như I/O có thể được đưa vào thread pool (do thư viện libuv cung cấp), sau đó trả kết quả về qua callback.

Nhờ mô hình này, NodeJS rất phù hợp với các ứng dụng yêu cầu hiệu suất cao và xử lý bất đồng bộ, đặc biệt là các ứng dụng thời gian thực như chat, streaming hoặc API server.

2.1.4 Ưu nhược điểm của NodeJS

Ưu điểm:

- Hiệu suất cao
- Xử lý thời gian thực tốt
- Sử dụng cùng một ngôn ngữ
- Hệ sinh thái phong phú
- Dễ dàng mở rộng

Nhược điểm:

- Hiệu suất kém với các tác vụ CPU nặng
- Không phù hợp với ứng dụng đơn luồng phức tạp
- Quản lý callback phức tạp
- Không có tính nhất quán
- Bảo mật

2.2 Giới thiệu về ExpressJS

2.2.1 Khái niệm ExpressJS

Express.js là một framework phổ biến được sử dụng để xây dựng ứng dụng web và API thông qua Node.js. Nền tảng được xem là một phương thức xử lý các yêu cầu HTTP, quản lý các tuyến đường, xử lý phần mềm trung gian và nhiều tính năng khác để phát triển hiệu quả ứng dụng web.

Express.js tập trung vào công việc tối ưu hóa việc xây dựng web ứng dụng bằng cách cung cấp một cấu trúc hoạt động và chỉ định rõ ràng việc xử lý yêu cầu và phản hồi. Nền tảng cũng hỗ trợ tích hợp các phần mềm trung gian bên ngoài để mở rộng chức năng của ứng dụng.

Với cộng đồng lớn và phổ biến, Express.js đã trở thành một trong những lựa chọn phổ biến để phát triển ứng dụng web và API trên nền tảng Node.js [6].

2.2.2 Tính năng của ExpressJS

Router tích hợp: Hỗ trợ định tuyến URL rõ ràng và có tổ chức, giúp xây dựng các ứng dụng có cấu trúc, hỗ trợ các phương thức HTTP (GET, POST, PUT, DELETE) để xử lý yêu cầu từ client.

Quản lý API dễ dàng: Được tối ưu hóa để xây dựng API RESTful và GraphQL, cung cấp cú pháp đơn giản để tạo các điểm cuối API (endpoints).

Tích hợp tốt với Node.js và các công nghệ khác: Express.js hoạt động trơn tru với Node.js, cơ sở dữ liệu và các thư viện bên ngoài thông qua npm.

2.2.3 Ưu nhược điểm của ExpressJS

Ưu điểm:

- Nhẹ và linh hoạt
- Dễ học và sử dụng
- Tốc độ phát triển nhanh
- Hiệu suất cao
- Hỗ trợ ứng dụng đa dạng

Nhược điểm:

- Không phù hợp với dự án lớn
- Quá phụ thuộc vào middleware
- Cần viết nhiều mã
- Bảo mật không tích hợp sẵn
- Quản lý lỗi không hiệu quả

2.2.4 Nguyên lý hoạt động của ExpressJS

ExpressJS hoạt động như một lớp trung gian giữa máy chủ NodeJS và các yêu cầu HTTP, đóng vai trò tổ chức và xử lý luồng dữ liệu đi qua hệ thống. Khi một yêu cầu từ phía client được gửi đến máy chủ, Express sẽ:

Tiếp nhận yêu cầu (request) và phân tích nội dung (method, URL, headers, body…).

Truy tìm tuyến đường (route) phù hợp thông qua hệ thống định tuyến (router).

Xử lý tuần tự các middleware tương ứng (gồm middleware mặc định và do người dùng định nghĩa).

Trả về phản hồi (response) thích hợp đến client.

Middleware trong Express đóng vai trò then chốt: mỗi middleware có thể truy cập, chỉnh sửa hoặc kết thúc quá trình xử lý yêu cầu, tạo nên một pipeline xử lý tuyến tính nhưng linh hoạt. Cơ chế này giúp Express dễ dàng mở rộng chức năng, ví dụ như xác thực người dùng, kiểm tra dữ liệu, ghi log, xử lý lỗi, v.v.

2.2.5 Một số ứng dụng phổ biến của ExpressJS

ExpressJS được sử dụng rộng rãi trong nhiều loại dự án, từ nhỏ đến trung bình. Một số ứng dụng tiêu biểu bao gồm:

- Xây dựng hệ thống API RESTful: Express được sử dụng làm nền tảng để tạo các API phục vụ frontend hoặc mobile app.

- Ứng dụng web có cấu trúc MVC: Express có thể kết hợp với các công cụ như EJS, Handlebars để phát triển các trang web động theo mô hình MVC.

- Hệ thống xác thực và phân quyền: Nhờ cơ chế middleware, Express hỗ trợ triển khai các chức năng bảo mật như JWT, OAuth dễ dàng.

- Ứng dụng thời gian thực khi kết hợp với Socket.io: Express thường được dùng làm lớp HTTP phía dưới khi xây dựng các ứng dụng chat, game, hoặc công cụ cộng tác.

- Hệ thống thương mại điện tử: Với khả năng tổ chức tuyến đường rõ ràng và khả năng mở rộng, Express được dùng trong nhiều hệ thống quản lý đơn hàng, giỏ hàng và thanh toán.

2.3 Giới thiệu về API

2.3.1 Khái niệm API

API là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của Application Programming Interface – giao diện lập trình ứng dụng. API cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng

2.3.2 Ứng dụng của API

Web API: là hệ thống API được sử dụng trong các hệ thống website. Hầu hết các website đều ứng dụng đến Web API cho phép bạn kết nối, lấy dữ liệu hoặc cập nhật cơ sở dữ liệu. Ví dụ: Bạn thiết kế chức nằng login thông Google, Facebook, Twitter, Github… Điều này có nghĩa là bạn đang gọi đến API của. Hoặc như các ứng dụng di động đều lấy dữ liệu thông qua API.

API trên hệ điều hành: Windows hay Linux có rất nhiều API, họ cung cấp các tài liệu API là đặc tả các hàm, phương thức cũng như các giao thức kết nối. Nó giúp lập trình viên có thể tạo ra các phần mềm ứng dụng có thể tương tác trực tiếp với hệ điều hành.

API của thư viện phần mềm hay framework: API mô tả và quy định các hành động mong muốn mà các thư viện cung cấp. Một API có thể có nhiều cách triển khai khác nhau và nó cũng giúp cho một chương trình viết bằng ngôn ngữ này có thể sử dụng thư viện được viết bằng ngôn ngữ khác. Ví dụ bạn có thể dùng Php để yêu cầu một thư viện tạo file PDF được viết bằng C++.

2.3.3 Phân loại API

Open API (Public API): Được công bố công khai, cho phép bất kỳ ai cũng có thể sử dụng. Ví dụ: API thời tiết, bản đồ Google Maps.

Internal API (Private API): Dùng nội bộ trong hệ thống tổ chức để các thành phần nội bộ giao tiếp với nhau.

Partner API: Chỉ được chia sẻ với đối tác cụ thể, thường yêu cầu xác thực và giới hạn quyền truy cập.

Composite API: Gộp nhiều API con thành một gọi duy nhất, giúp giảm số lượng request và cải thiện hiệu suất.

2.3.4 Ưu, nhược điểm của API

Ưu điểm:

Tăng khả năng tái sử dụng mã nguồn.

Giảm chi phí phát triển và bảo trì hệ thống.

Hỗ trợ tích hợp hệ thống nhanh chóng.

Nhược điểm:

Cần đảm bảo bảo mật, xác thực và phân quyền hợp lý.

Phụ thuộc vào bên thứ ba nếu sử dụng API ngoài.

Phải duy trì tài liệu và kiểm tra tương thích phiên bản.

2.4 Giới thiệu về RESTful API

2.4.1 Khái niệm RESTful API

RESTful API (Representational State Transfer API) là một kiểu kiến trúc cho các API (Application Programming Interface) được sử dụng để truyền tải và trao đổi dữ liệu giữa các ứng dụng web. RESTful API sử dụng giao thức HTTP để truyền tải dữ liệu giữa máy chủ và máy khách và sử dụng các phương thức HTTP như GET, POST, PUT và DELETE để thực hiện các thao tác trên tài nguyên.

RESTful API sử dụng các URL dễ đọc và dễ hiểu và sử dụng các định dạng dữ liệu như JSON hoặc XML để trao đổi thông tin giữa máy chủ và máy khách. RESTful API cũng có tính khả di động cao, cho phép các ứng dụng khác nhau có thể truy cập và sử dụng các tài nguyên một cách dễ dàng [7].

2.4.2 Các nguyên tắc của RESTful API

RESTful API được xây dựng dựa trên một tập hợp các nguyên tắc cốt lõi, giúp đảm bảo tính hiệu quả và linh hoạt trong thiết kế.

Client-Server Architecture (Kiến trúc Client-Server)

Statelessness (Không lưu trạng thái)

Cacheable (Có thể lưu cache)

Layered System (Hệ thống phân lớp)

Uniform Interface (Giao diện thống nhất)

Code on Demand (Mã được tải về) *(Không bắt buộc)*

2.4.3 Các thành phần chính của RESTful API

Resource (Tài nguyên): Mỗi tài nguyên có một URL định danh duy nhất.

HTTP Methods:

GET: Lấy dữ liệu.

POST: Tạo mới dữ liệu.

PUT: Cập nhật toàn bộ dữ liệu.

PATCH: Cập nhật một phần.

DELETE: Xoá dữ liệu.

Status Code: Trạng thái phản hồi (200 OK, 404 Not Found, 500 Server Error…).

Headers & Body: Chứa metadata và nội dung dữ liệu gửi/nhận.

2.4.4 Lợi ích của RESTful API

Tính đơn giản và tiêu chuẩn hóa: Dựa trên HTTP: RESTful API sử dụng các phương thức HTTP cơ bản (GET, POST, PUT, DELETE) để thực hiện các thao tác CRUD (Create, Read, Update, Delete).

Khả năng tương thích đa nền tảng: RESTful API có thể được sử dụng bởi bất kỳ ứng dụng nào có khả năng gửi yêu cầu HTTP, như trình duyệt, ứng dụng di động, hoặc các dịch vụ backend.

Tính mở rộng (Scalability): Kiến trúc REST cho phép các hệ thống mở rộng dễ dàng nhờ vào khả năng xử lý tải trọng cao mà không làm gián đoạn các thành phần khác.

Khả năng tái sử dụng (Reusability): API được thiết kế theo RESTful dễ tái sử dụng trong các dự án khác nhau, giảm thời gian và chi phí phát triển.

Tách biệt giữa client và server: RESTful API cho phép tách biệt hoàn toàn phần giao diện người dùng (frontend) và logic xử lý (backend), giúp việc phát triển đồng thời hai phần này trở nên dễ dàng hơn.

Tính linh hoạt và tương lai: RESTful API có thể cung cấp dữ liệu ở nhiều định dạng khác nhau (JSON, XML, HTML), thích hợp với nhu cầu khác nhau của ứng dụng.

2.4.5 Những hạn chế của RESTful API

Không phù hợp với các hệ thống có mối quan hệ dữ liệu phức tạp.

Thiếu khả năng mô tả API tự động như GraphQL.

Không có quy chuẩn chính thức về cách đặt tên tài nguyên, dễ gây không thống nhất.

2.5 Giới thiệu về React Native

2.5.1 Khái niệm React Native

React Native là một framework mã nguồn mở được sáng tạo bởi Facebook. Nó được sử dụng để phát triển ứng dụng di động Android, iOS, Web và UWP bằng cách cho phép các nhà phát triển sử dụng React cùng với môi trường ứng dụng gốc[8].

2.5.2 Ứng dụng của ReactJS

Ứng dụng di động đa nền tảng (Android & iOS): React Native cho phép xây dựng ứng dụng chạy trên cả hai hệ điều hành từ một codebase JavaScript duy nhất. Ví dụ: Facebook, Instagram, Uber Eats.

Ứng dụng thương mại điện tử trên thiết bị di động: Phát triển các app bán hàng có giao diện đẹp, mượt mà, hỗ trợ tìm kiếm, lọc, giỏ hàng, thanh toán và quản lý đơn hàng. Ví dụ: Shopee, Amazon Mobile App.

Ứng dụng quản lý và theo dõi cá nhân: Tạo app theo dõi sức khỏe, lịch trình, công việc hoặc học tập. Ví dụ: app tập gym, học tiếng Anh, quản lý chi tiêu cá nhân.

Ứng dụng bản đồ và định vị: Tích hợp bản đồ, GPS và định vị thời gian thực để phục vụ các dịch vụ như giao hàng, gọi xe. Ví dụ: Grab, Gojek.

Ứng dụng giáo dục và học trực tuyến: Học từ vựng, video bài giảng, bài tập tương tác. Ví dụ: Duolingo, Monkey Stories.

2.5.3 Ưu nhược điểm của ReactJS

Ưu điểm:

Tối ưu chi phí và thời gian: Chỉ cần viết một lần bằng JavaScript, có thể chạy được trên cả Android và iOS, giảm công sức phát triển và bảo trì.

Cộng đồng lớn, nhiều thư viện hỗ trợ: Dễ dàng tìm kiếm giải pháp hoặc gói thư viện cho các tính năng phổ biến như định vị, camera, thanh toán.

Hot Reloading / Fast Refresh: Giúp lập trình viên thấy ngay kết quả sau khi chỉnh sửa mã mà không cần build lại ứng dụng.

Hệ sinh thái JavaScript: Có thể kết hợp dễ dàng với Redux, Axios, Firebase và các thư viện khác trong hệ sinh thái JavaScript.

Nhược điểm:

Debugging đôi khi phức tạp: Do sử dụng nhiều công nghệ kết hợp (JavaScript, native modules), việc tìm lỗi có thể mất thời gian.

Yêu cầu kiến thức đa nền tảng: Dù dùng JavaScript, nhưng đôi khi vẫn cần hiểu native Android (Java/Kotlin) hoặc iOS (Swift/Objective-C) để xử lý các vấn đề đặc thù.

Quản lý phiên bản thư viện khó khăn: Một số thư viện có thể không tương thích giữa các phiên bản React Native, dễ phát sinh lỗi khi cập nhật.

Chưa hỗ trợ tốt một số tính năng đặc biệt: Một số chức năng native như Bluetooth, NFC,... cần phải viết thêm module native thủ công.

2.5.4 Nguyên lý hoạt động của React Native

React Native hoạt động dựa trên cơ chế "bridge" – cầu nối giữa JavaScript và mã gốc (native). Khi ứng dụng được chạy, mã JavaScript sẽ được biên dịch bằng công cụ như JavaScriptCore (trên iOS) hoặc Hermes Engine (trên Android), sau đó truyền dữ liệu qua “bridge” đến các thành phần gốc của hệ điều hành.

Cụ thể:

Giao diện được xây dựng từ các component (nút, hình ảnh, danh sách…) dùng cú pháp React.

Khi một hành động xảy ra (như người dùng bấm nút), React Native gửi thông tin đó từ JavaScript sang native code qua cầu nối bất đồng bộ (asynchronous bridge).

Native code xử lý hành động và trả lại phản hồi cho JavaScript nếu cần.

Giao diện được cập nhật nhờ cơ chế Virtual DOM và sự tái sử dụng lại thành phần giống ReactJS, nhưng các thành phần này được render bằng UI gốc, không phải HTML/CSS.

Điểm khác biệt lớn của React Native với các giải pháp hybrid (như Cordova) là nó render giao diện bằng native components thay vì webview, nhờ đó mang lại trải nghiệm mượt mà hơn và hiệu năng tốt hơn.

2.5.5 Một số trường hợp sử dụng phổ biến của React Native

React Native phù hợp với nhiều loại ứng dụng trên thiết bị di động nhờ khả năng phát triển đa nền tảng:

Startup muốn ra mắt MVP nhanh: Với nguồn lực giới hạn, startup có thể dùng React Native để nhanh chóng phát triển phiên bản Android và iOS với cùng một mã nguồn.

Ứng dụng có giao diện đơn giản đến trung bình: Các app đọc tin, quản lý cá nhân, mạng xã hội, thương mại điện tử thường không yêu cầu các animation quá phức tạp nên React Native hoạt động tốt.

Doanh nghiệp muốn tái sử dụng code web: Nhờ sử dụng JavaScript và cùng triết lý component của ReactJS, đội ngũ web dễ dàng chuyển sang xây dựng app di động.

Ứng dụng cần cập nhật thường xuyên: React Native hỗ trợ hot update qua CodePush, cho phép cập nhật giao diện hoặc logic JavaScript mà không cần chờ xét duyệt lại từ App Store hoặc Google Play.

2.6 Giới thiệu về MongoDB

2.6.1 Khái niệm MongoDB

Mongodb là một loại database thiên hướng tài liệu và là một dạng NoSQL database. Chính vì vậy, Mongodb thường sẽ tránh đi cấu trúc table-based của relational database để có thể thích ứng được với mọi tài liệu như JSON có sẵn trong một schema rất linh hoạt và được gọi là BSON.

Mongodb Atlas là một giải pháp phần mềm Database as a Service Provider có chức năng và chi phí hoàn toàn phù hợp cho mọi doanh nghiệp từ nhỏ đến vừa và đến lớn. Khi đó, phần mềm MongoDB Atlas sẽ được đánh giá bởi tất cả các người dùng lẫn với chuyên gia trong các lĩnh vực Database Software [9].

2.6.2 Tính năng của MongoDB

Cấu trúc dữ liệu linh hoạt (Document-oriented): MongoDB lưu trữ dữ liệu dưới dạng document JSON/BSON, giúp dễ dàng biểu diễn các cấu trúc dữ liệu phức tạp và đa dạng mà không cần bảng hoặc mối quan hệ cứng nhắc như SQL.

Schema-less (Không có schema cố định): không yêu cầu cấu trúc dữ liệu cố định, cho phép thêm hoặc sửa đổi dữ liệu một cách linh hoạt mà không cần thay đổi schema.

Hỗ trợ sharding (Phân mảnh dữ liệu): hỗ trợ phân mảnh dữ liệu trên nhiều máy chủ, đảm bảo hiệu suất tốt khi xử lý lượng dữ liệu lớn hoặc lưu lượng cao.

Replication (Sao chép dữ liệu): MongoDB hỗ trợ replication thông qua Replica Sets, giúp tăng độ tin cậy và khả năng phục hồi sau sự cố.

Query và Aggregation Framework: MongoDB cung cấp các truy vấn linh hoạt, tìm kiếm toàn văn (full-text search) và khả năng phân tích dữ liệu mạnh mẽ với aggregation pipeline.

Khả năng tích hợp tốt: tích hợp với nhiều ngôn ngữ lập trình như JavaScript, Python, Java, Node.js và các công cụ phân tích dữ liệu như Tableau.

Hỗ trợ Geospatial Data: MongoDB có tính năng hỗ trợ dữ liệu không gian địa lý, phù hợp với các ứng dụng như bản đồ, giao vận, hoặc tìm kiếm địa điểm.

2.6.3 Ưu nhược điểm của MongoDB

Ưu điểm:

- Hiệu suất cao và mở rộng dễ dàng
- Dễ học và triển khai
- Linh hoạt trong quản lý dữ liệu
- Thích hợp cho dữ liệu phi cấu trúc và bán cấu trúc
- Cộng đồng lớn và tài liệu phong phú
- Khả năng xử lý dữ liệu phức tạp

Nhược điểm:

- Tiêu thụ nhiều bộ nhớ
- Không phù hợp cho các ứng dụng giao dịch
- Thiếu tính toàn vẹn dữ liệu
- Không tối ưu cho truy vấn phức tạp giữa các bảng
- Chi phí triển khai phân mảnh (sharding)

2.6.4 Một số ứng dụng phổ biến của MongoDB

MongoDB phù hợp với nhiều loại ứng dụng hiện đại, đặc biệt là các hệ thống xử lý dữ liệu linh hoạt, không đồng nhất, hoặc có yêu cầu mở rộng cao:

- Ứng dụng thương mại điện tử: Lưu trữ thông tin sản phẩm, đơn hàng, người dùng với cấu trúc thay đổi linh hoạt (Shopee, Amazon, Lazada...).

- Ứng dụng mạng xã hội: Xử lý dữ liệu không đồng nhất như bài đăng, bình luận, lượt thích, bạn bè... (Facebook, LinkedIn).

- Hệ thống phân tích dữ liệu lớn (Big Data Analytics): Làm nguồn lưu trữ cho các pipeline ETL và hệ thống BI nhờ khả năng tích hợp với Hadoop, Spark, Tableau...

- Ứng dụng theo dõi thời gian thực: Ví dụ như theo dõi vị trí xe (fleet tracking), giám sát hệ thống IoT.

- Ứng dụng quản lý nội dung (CMS): Lưu trữ bài viết, hình ảnh, video, metadata... mà không cần thiết kế bảng cố định như SQL.

- Ứng dụng di động & web hiện đại: Kết hợp tốt với Node.js, Express, React/React Native để xây dựng các hệ thống MERN hoặc MEAN stack.

2.7 Giới thiệu về thanh toán điện tử

2.7.1 Khái niệm thanh toán điện tử

Thanh toán điện tử, hay còn được gọi là thanh toán trực tuyến, là một hình thức thanh toán trên internet, cho phép thực hiện các giao dịch tài chính bằng cách sử dụng các công nghệ thông tin như internet hay các thiết bị di động, thay vì sử dụng tiền mặt hoặc thẻ tín dụng. Điều này mang lại nhiều lợi ích cho người dùng như tính tiện lợi, tốc độ nhanh chóng, an toàn và dễ dàng kiểm soát các giao dịch tài chính. Thanh toán điện tử đang trở thành xu hướng phổ biến trong thời đại công nghệ hiện nay và được sử dụng rộng rãi trong các lĩnh vực kinh doanh, thương mại điện tử và dịch vụ tài chính [10].

2.7.2 Các hình thức thanh toán điện tử

Thanh toán qua thẻ: Thẻ tín dụng (Credit Card**),** thẻ ghi nợ (Debit Card**),** thẻ trả trước (Prepaid Card)

Ví điện tử (E-wallet): Là ứng dụng trên điện thoại hoặc nền tảng trực tuyến cho phép lưu trữ tiền và thanh toán một cách nhanh chóng.

Internet Banking: Sử dụng giao diện web để chuyển tiền, thanh toán hóa đơn, nạp tiền điện thoại, v.v.

Mobile Banking: Ứng dụng ngân hàng trên điện thoại, cung cấp các dịch vụ tương tự Internet Banking với giao diện thân thiện hơn.

Thanh toán QR Code: Người dùng quét mã QR của người bán bằng ứng dụng ngân hàng hoặc ví điện tử để thực hiện giao dịch.

2.7.3 Lợi ích của thanh toán điện tử

- Tiện lợi và nhanh chóng

- Giảm phụ thuộc vào tiền mặt

- Tăng cường tính minh bạch
- An toàn và bảo mật
- Hỗ trợ giao dịch quốc tế
- Giảm chi phí vận hành
- Hỗ trợ tích hợp nhiều tiện ích
- Thân thiện với môi trường
- Thúc đẩy nền kinh tế số
- Khuyến mãi và ưu đãi

2.7.4 Thách thức và rủi ro của thanh toán điện tử

Dù mang lại nhiều lợi ích vượt trội, thanh toán điện tử vẫn tồn tại một số thách thức và rủi ro mà người dùng và doanh nghiệp cần lưu ý:

Rủi ro bảo mật: Các cuộc tấn công mạng như đánh cắp thông tin thẻ, tài khoản, mã OTP có thể gây thất thoát tài sản nếu không được bảo vệ đúng mức.

Lừa đảo và gian lận: Hình thức lừa đảo thông qua website giả mạo, tin nhắn rác (phishing), hoặc các ứng dụng mạo danh ngày càng tinh vi.

Phụ thuộc vào hạ tầng công nghệ: Mạng internet, server của ngân hàng hoặc ví điện tử bị gián đoạn có thể làm gián đoạn hoạt động thanh toán.

Thiếu kỹ năng số ở một bộ phận người dùng: Đặc biệt là người cao tuổi hoặc vùng nông thôn, gây khó khăn trong tiếp cận và sử dụng.

Chi phí giao dịch ẩn: Một số nền tảng có thể tính phí giao dịch hoặc chuyển tiền, ảnh hưởng đến trải nghiệm người dùng nếu không được minh bạch.

2.7.5 Một số nền tảng thanh toán điện tử phổ biến tại Việt Nam

Tại Việt Nam, thị trường thanh toán điện tử đang phát triển mạnh với nhiều nhà cung cấp uy tín, tiêu biểu gồm:

Momo: Ví điện tử phổ biến, hỗ trợ thanh toán hóa đơn, chuyển tiền, nạp tiền điện thoại và tích điểm thưởng.

ZaloPay: Tích hợp với hệ sinh thái Zalo, hỗ trợ thanh toán nhanh qua mã QR, chuyển tiền nội bộ chỉ bằng số điện thoại.

VNPay: Cung cấp giải pháp thanh toán QR code, liên kết với nhiều ngân hàng và doanh nghiệp.

ShopeePay: Gắn liền với sàn thương mại điện tử Shopee, hỗ trợ thanh toán đơn hàng, hoàn tiền, khuyến mãi hấp dẫn.

Internet/Mobile Banking của các ngân hàng lớn: Như Vietcombank, Techcombank, BIDV, MB Bank, hỗ trợ đầy đủ các tính năng từ chuyển khoản đến thanh toán QR, gửi tiết kiệm trực tuyến.

CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1 Khảo sát hiện trạng

Hiện nay, nhiều người có thói quen mua sắm các thiết bị công nghệ như điện thoại, laptop, tai nghe… qua ứng dụng trên điện thoại vì tiện lợi và nhanh chóng. Các ứng dụng lớn như Shopee, Tiki, Lazada đều có giao diện thân thiện, dễ sử dụng và cho phép người dùng tìm kiếm, đặt hàng và thanh toán dễ dàng.

Tuy nhiên, nhiều cửa hàng nhỏ chưa có ứng dụng riêng mà vẫn bán hàng qua Facebook, Zalo hoặc website đơn giản. Điều này gây khó khăn trong việc quản lý sản phẩm, đơn hàng và làm trải nghiệm người mua không được tốt.

Một giải pháp hiệu quả là xây dựng một ứng dụng bán hàng, có kết nối với API để lấy dữ liệu từ hệ thống (như danh sách sản phẩm, thông tin người dùng, đơn hàng…). Điều này giúp ứng dụng hoạt động nhanh hơn, dữ liệu được đồng bộ tốt hơn và dễ mở rộng sau này.

3.2 Mô tả bài toán

Đề tài yêu cầu xây dựng một ứng dụng di động bán thiết bị công nghệ, cho phép người dùng xem sản phẩm, đặt hàng và quản lý tài khoản. Ứng dụng sẽ kết nối với API để lấy và gửi dữ liệu với máy chủ (server).

Chức năng chính của ứng dụng:

Đối với người dùng:

- Xem danh sách các sản phẩm (điện thoại, laptop, phụ kiện...).
- Tìm kiếm và lọc sản phẩm theo loại.
- Xem chi tiết sản phẩm: tên, hình ảnh, giá, mô tả...
- Thêm vào giỏ hàng, đặt mua sản phẩm.
- Đăng ký, đăng nhập, quản lý tài khoản và đơn hàng đã mua.

Đối với người quản lý:

- Thêm, sửa, xóa sản phẩm (thông qua API hoặc giao diện quản lý).
- Xem và cập nhật trạng thái đơn hàng.

- Quản lý doanh thu, tồn kho,…

Công nghệ sử dụng:

Frontend (ứng dụng di động): React Native

Backend (server/API): Node.js (Express)

Cơ sở dữ liệu: MongoDB

API: Kết nối frontend và backend, giúp truyền dữ liệu (sản phẩm, người dùng, đơn hàng...).

Mục tiêu:

Biết cách xây dựng ứng dụng bán hàng trên điện thoại.

Hiểu và sử dụng API để truyền và nhận dữ liệu.

Tạo ra một ứng dụng đơn giản, dễ sử dụng, có thể demo được.

3.3 Mô tả tổng quan về website

Ứng dụng bán thiết bị công nghệ trên thiết bị di động là nền tảng thương mại điện tử được xây dựng dựa trên việc tích hợp và khai thác hiệu quả các API hiện đại. Ứng dụng cho phép người dùng truy cập nhanh chóng vào kho sản phẩm công nghệ đa dạng như điện thoại, laptop, phụ kiện, với thông tin chi tiết về thông số kỹ thuật, hình ảnh, giá cả và đánh giá người dùng. Giao diện tối ưu cho di động giúp người dùng dễ dàng tìm kiếm, lọc sản phẩm, thêm vào giỏ hàng và thực hiện thanh toán trực tuyến một cách an toàn và tiện lợi.

Ngoài ra, hệ thống API còn hỗ trợ người dùng đăng ký tài khoản, đăng nhập, quản lý đơn hàng và theo dõi trạng thái giao hàng theo thời gian thực. Ứng dụng còn tích hợp các API đánh giá sản phẩm và hiển thị bài viết tư vấn công nghệ giúp khách hàng có thêm thông tin trước khi đưa ra quyết định mua hàng. Việc xây dựng ứng dụng dựa trên API không chỉ giúp tăng tính linh hoạt mà còn nâng cao trải nghiệm người dùng trên nền tảng di động.

3.4 Các chức năng của website

Giới thiệu sản phẩm: cung cấp thông tin chi tiết về sản phẩm như giá bán, thương hiệu, thông tin,…

Mua sắm trực tuyến: cho phép người dùng thêm sản phẩm vào giỏ hàng và thanh toán trực tuyến.

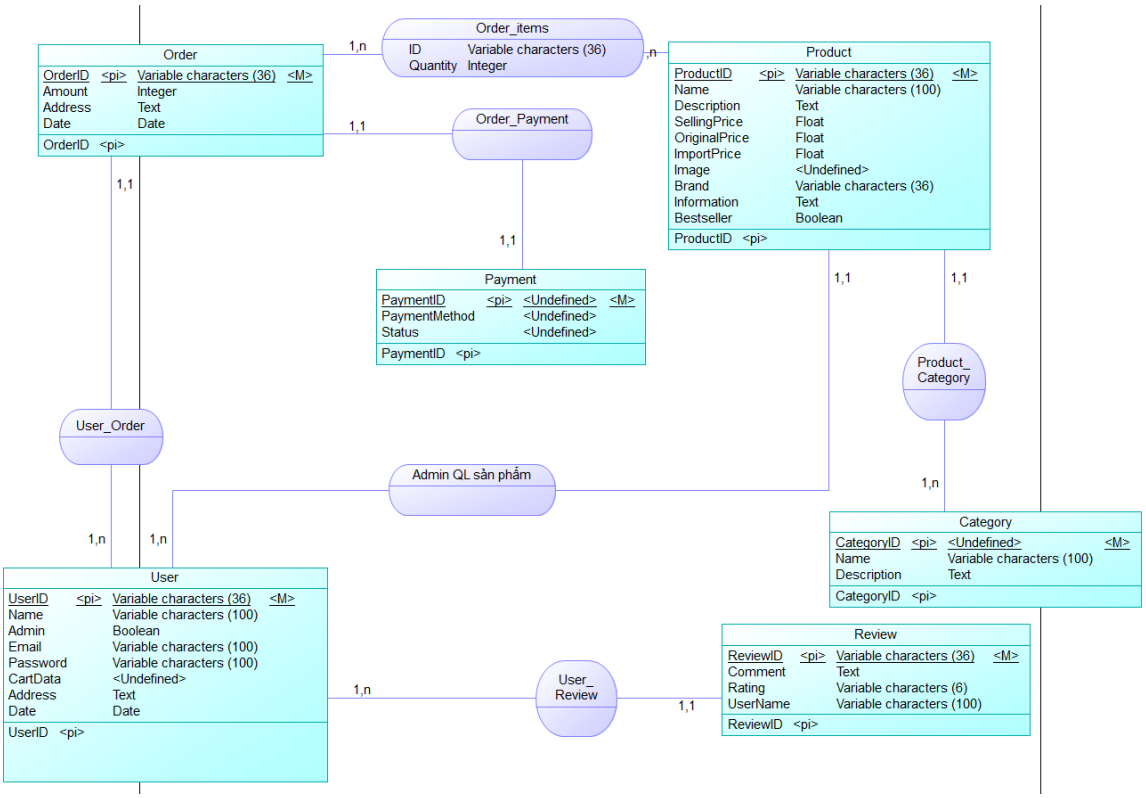
Tìm kiếm và lọc sản phẩm: hỗ trợ khách hàng tìm kiếm theo tên, loại sản phẩm, giá tiền, thương hiệu

Đánh giá, nhận xét sản phẩm: người dùng đánh giá và nhận xét sau khi mua hàng.

Tài khoản người dùng: người dùng có thể tạo tài khoản, quản lý đơn hàng, lưu sản phẩm yêu thích và theo dõi trạng thái giao hàng.

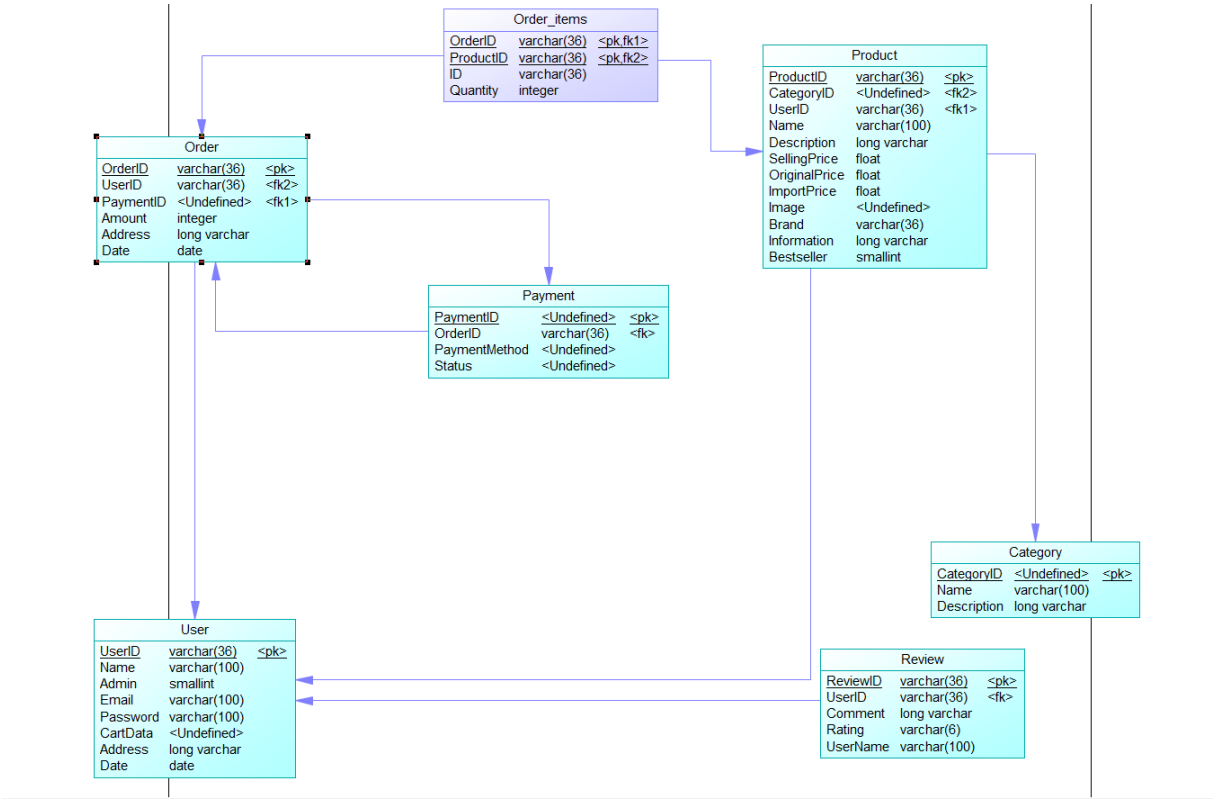
3.5 Xây dựng website

3.5.1 Mô hình dữ liệu mức quan niệm



Hình 1. Mô hình dữ liệu mức quan niệm

3.5.2 Mô hình dữ liệu mức logic



Hình 2. Mô hình dữ liệu mức logic

3.5.3 Các thực thể

Bảng 1. Bảng User

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	UserID	ID người dùng (khóa chính)	Varchar(36)
2	Name	Tên người dùng	Varchar(100)
3	Admin	Vai trò (1: Admin, 0: Customer)	Boolean
4	Email	Email người dùng	Varchar(100)
5	Password	Mật khẩu	Varchar(255)
6	CartData	Dữ liệu giỏ hàng	JSON/Undefined

7	WishData	Dữ liệu yêu thích	JSON/Undefined
8	Date	Ngày tạo tài khoản	Datetime

Bảng 2. Bảng Order

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	OrderID	ID đơn hàng (khóa chính)	Varchar(36)
2	PaymentID	ID thanh toán (khóa ngoại)	Varchar(36)
3	UserID	ID người dùng (khóa ngoại)	Varchar(36)
4	Amount	Tổng tiền đơn hàng	Int
5	Address	Địa chỉ giao hàng	Text
6	Date	Ngày đặt hàng	Datetime

Bảng 3. Bảng Order_items

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	OrderID	ID đơn hàng (khóa ngoại)	Varchar(36)
2	ProductID	ID sản phẩm (khóa ngoại)	Varchar(36)
3	Quantity	Số lượng sản phẩm	Int
4	ID	ID chi tiết đơn hàng	Varchar(36)

Bảng 4. Bảng Product

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	ProductID	ID sản phẩm (khóa chính)	Varchar(36)
2	UserID	ID người tạo sản phẩm (khóa	Varchar(36)

		ngoại)	
3	CategoryID	ID danh mục (khóa ngoại)	Varchar(36)
4	Name	Tên sản phẩm	Varchar(100)
5	Description	Mô tả sản phẩm	Text
6	SellingPrice	Giá bán sản phẩm	Float
7	OriginalPrice	Giá gốc sản phẩm	Float
8	ImportPrice	Giá nhập sản phẩm	Float
9	Image	Hình ảnh sản phẩm	JSON/Undefined
10	Information	Thông tin sản phẩm	Text
11	Brand	Thương hiệu sản phẩm	Varchar(36)
12	Bestseller	Sản phẩm bán chạy	Bit

Bảng 5. Bảng Category

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	CategoryID	ID danh mục (khóa chính)	Varchar(36)
2	Name	Tên danh mục	Varchar(100)
3	Description	Mô tả danh mục	Text

Bảng 6. Bảng Review

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	ReviewID	ID đánh giá (khóa chính)	Varchar(36)
2	UserID	ID người dùng (khóa ngoại)	Varchar(36)

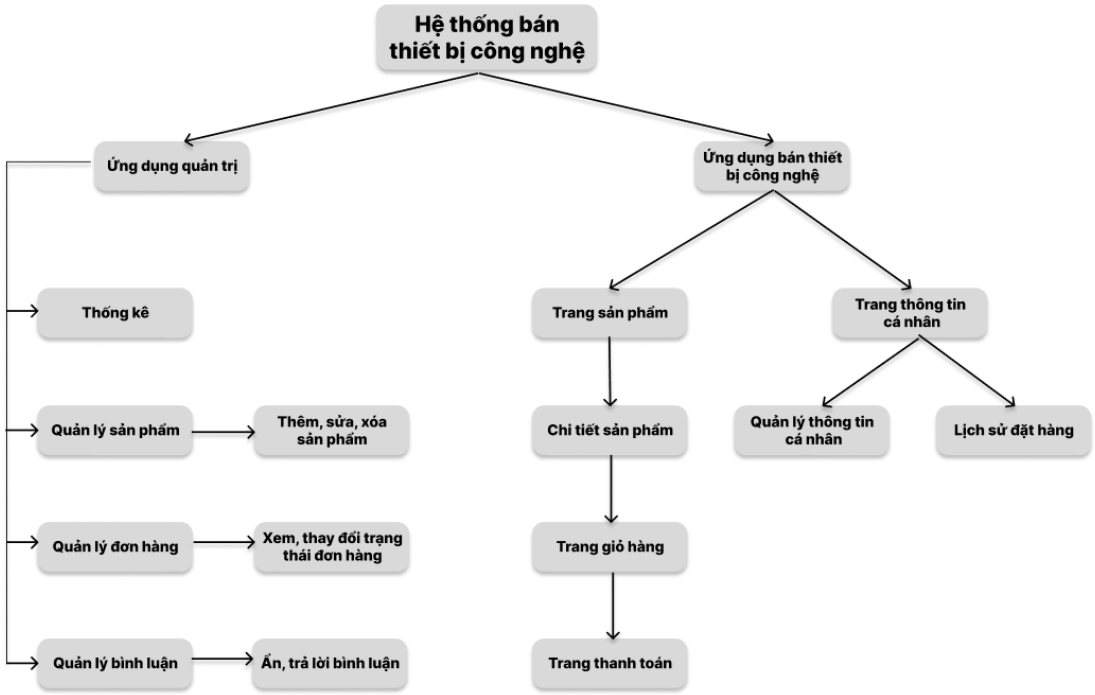
3	Comment	Bình luận	Text
4	Rating	Đánh giá (1-5 sao)	Int
5	NameUser	Tên người đánh giá	Varchar(100)

Bảng 7. Bảng Payment

STT	Tên thuộc tính	Mô tả	Kiểu dữ liệu
1	PaymentID	ID thanh toán (khóa chính)	Varchar(36)
2	OrderID	ID đơn hàng (khóa ngoại)	Varchar(36)
3	PaymentMethod	Phương thức thanh toán	Varchar(50)
4	Status	Trạng thái thanh toán	Varchar(50)

3.6 Thiết kế xử lý

3.6.1 Sơ đồ hệ thống



Hình 3. Sơ đồ hệ thống

Sơ đồ chức năng của hệ thống bán thiết bị công nghệ được chia thành hai phân hệ chính: ứng dụng quản trị và ứng dụng bán thiết bị công nghệ cho người dùng. Mỗi phân hệ đảm nhận các chức năng riêng biệt nhằm phục vụ cho từng vai trò trong hệ thống.

Ứng dụng quản trị

Đây là phân hệ dành cho quản trị viên, nhằm quản lý toàn bộ hoạt động của hệ thống, bao gồm:

Thống kê: Cung cấp số liệu tổng quan như doanh thu, số lượng đơn hàng, lượng người dùng và các sản phẩm bán chạy.

Quản lý sản phẩm: Cho phép thêm mới, chỉnh sửa hoặc xóa sản phẩm trong hệ thống.

Quản lý đơn hàng: Cho phép theo dõi đơn hàng và cập nhật trạng thái đơn hàng như: đã xác nhận, đang vận chuyển, hoàn tất hoặc hủy đơn.

Quản lý bình luận: Quản trị viên có thể ẩn hoặc trả lời các bình luận từ người dùng, nhằm đảm bảo môi trường mua sắm tích cực.

Ứng dụng bán thiết bị công nghệ (cho người dùng)

Đây là phân hệ dành cho người dùng cuối, phục vụ quá trình mua sắm trực tuyến và quản lý thông tin cá nhân. Bao gồm các chức năng:

Trang sản phẩm: Hiển thị danh sách các sản phẩm hiện có, có thể bao gồm chức năng tìm kiếm, lọc theo danh mục hoặc giá.

Chi tiết sản phẩm: Cung cấp thông tin chi tiết về sản phẩm như mô tả, thông số kỹ thuật, đánh giá và hình ảnh.

Trang giỏ hàng: Hiển thị danh sách sản phẩm đã chọn, cho phép cập nhật số lượng hoặc xóa khỏi giỏ hàng.

Trang thanh toán: Cho phép chọn thông tin giao hàng, chọn phương thức thanh toán và tiến hành đặt hàng.

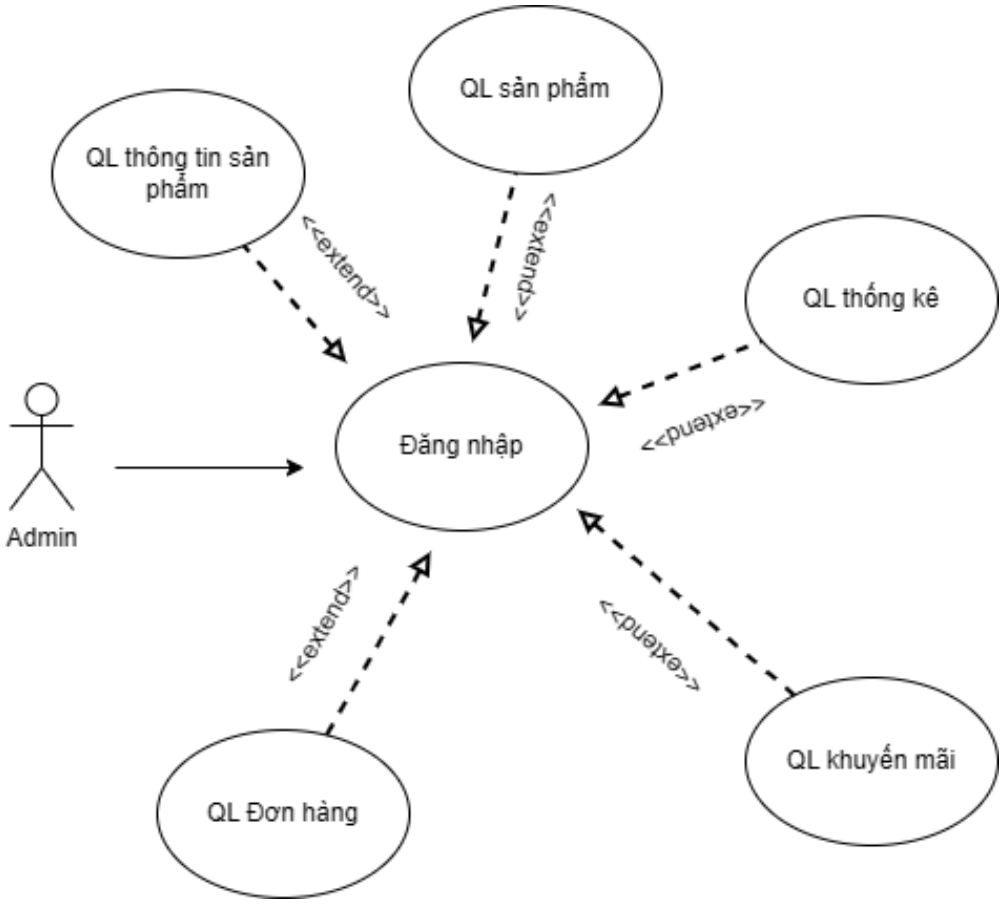
Trang thông tin cá nhân: Bao gồm:

Quản lý thông tin cá nhân: Cho phép người dùng cập nhật họ tên, email, địa chỉ, số điện thoại,...

Quản lý đơn hàng: Hiển thị danh sách các đơn hàng đã mua cùng với trạng thái từng đơn hàng.

3.6.2 Biểu đồ Usecase

Usecase tổng quát của tác nhân quản trị



Hình 4. Biểu đồ Use Case tổng quát của tác nhân quản trị

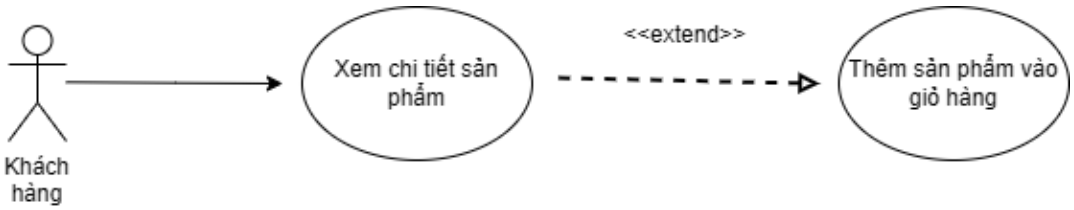
Mô tả: Người quản trị viên có chức năng quản lý địa thông tin sản phẩm, quản lý sản phẩm, quản lý thông kê, quản lý khuyến mãi, quản lý đơn hàng.

Usecase tổng quát của tác nhân khách hàng

Hình 5. Biểu đồ Use Case tổng quát của tác nhân khách hàng

Mô tả: Khách hàng có chức năng tìm kiếm sản phẩm, xam sản phẩm, xem chi tiết sản phẩm, đánh giá, góp ý, đặt hàng, đăng ký/đăng nhập, quản lý giỏ hàng.

Usecase khách hàng thêm sản phẩm vào giỏ hàng



Hình 6. Usecase khách hàng thêm sản phẩm vào giỏ hàng

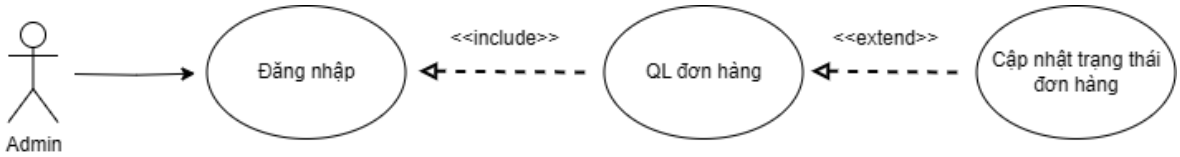
Mô tả: Khách hàng có thể xem chi tiết sản phẩm để tìm hiểu thêm thông tin. Từ đó, họ có tùy chọn thêm sản phẩm vào giỏ hàng nếu muốn mua hoặc thêm vào mục sản phẩm yêu thích.

Usecase admin quản trị sản phẩm

Hình 7. Usecase admin quản trị sản phẩm

Mô tả: Sơ đồ thể hiện quy trình Admin quản lý sản phẩm. Sau khi đăng nhập, Admin có thể truy cập vào chức năng quản lý sản phẩm, bao gồm các hành động như thêm sản phẩm, với các tùy chọn mở rộng là cập nhật hoặc xóa sản phẩm.

Usecase admin quản trị đơn hàng



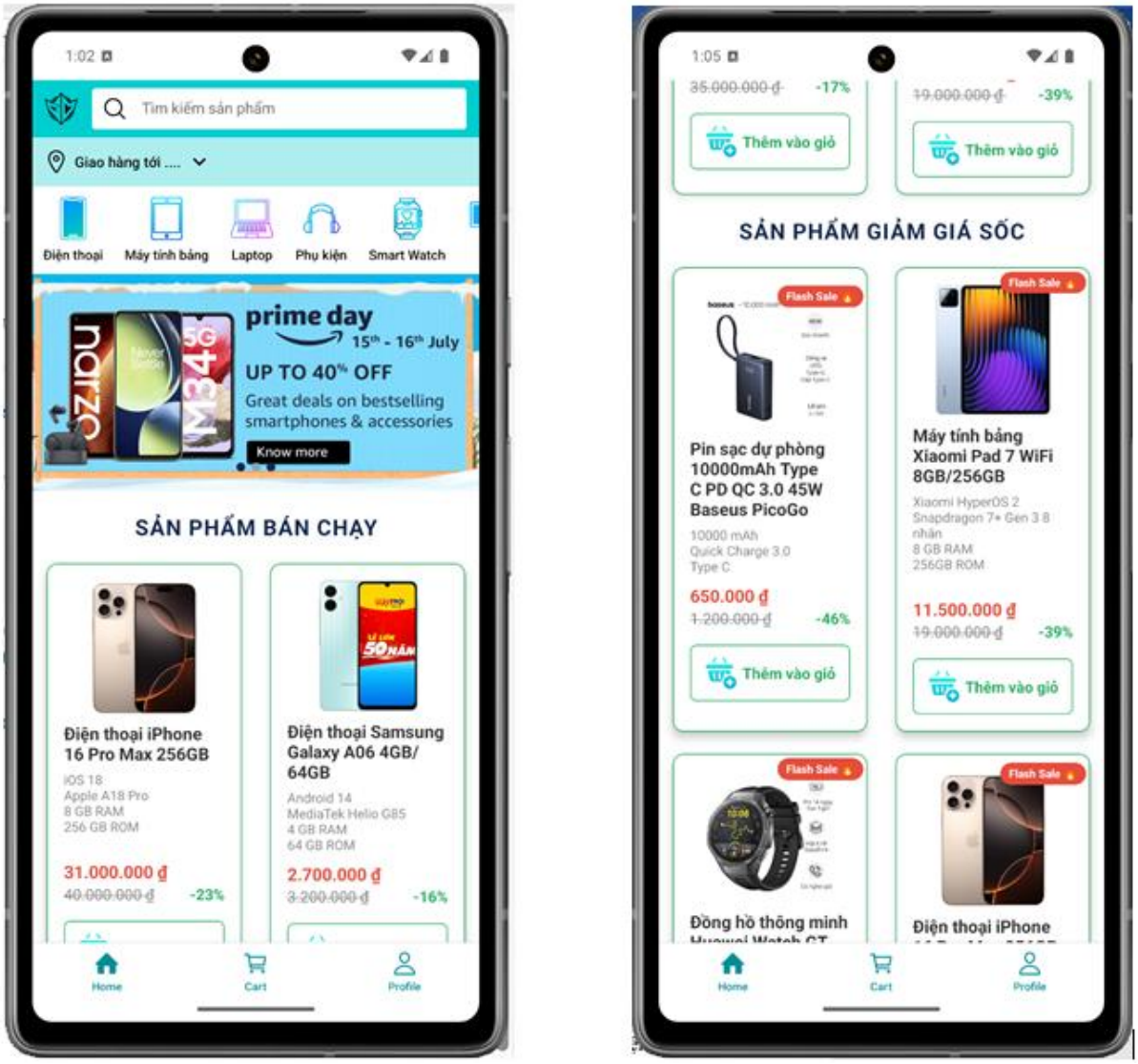
Hình 8. Usecase admin quản lý đơn hàng

Mô tả: Sơ đồ mô tả quy trình Admin quản lý đơn hàng. Admin bắt đầu bằng việc đăng nhập để truy cập chức năng quản lý đơn hàng, sau đó có thể thực hiện hành động cập nhật trạng thái đơn hàng.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Giao diện người dùng

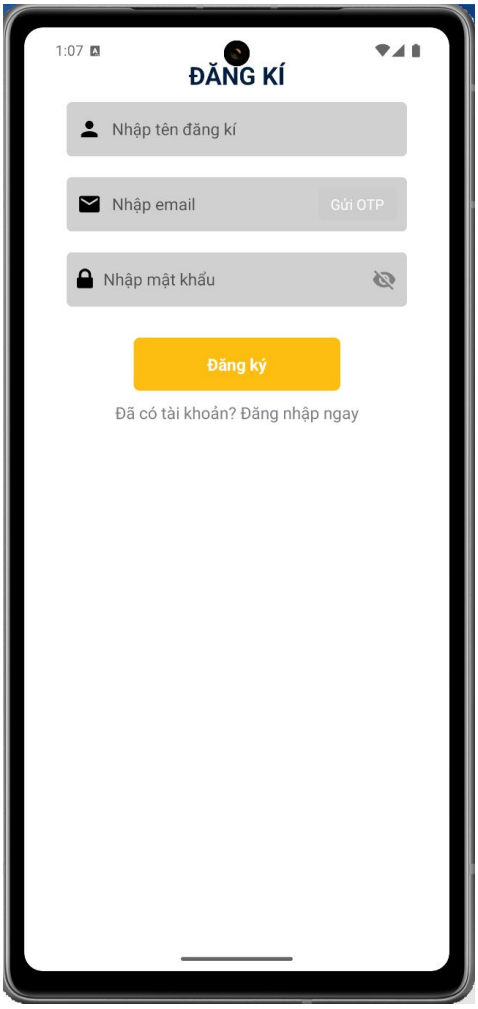
4.1.1 Giao diện trang chủ



Hình 9. Giao diện trang chủ

Mô tả: trang chủ là trang đầu tiên xuất hiện khi người dùng truy cập vào website. Nó được thiết kế với mục tiêu tạo ra một không gian dễ dàng và thuận tiện cho khách hàng khám phá các sản phẩm mới nhất, tìm kiếm những mặt hàng nổi bật và lựa chọn sản phẩm phù hợp với nhu cầu cá nhân.

4.1.2 Giao diện trang đăng ký



Hình 10. Giao diện đăng ký

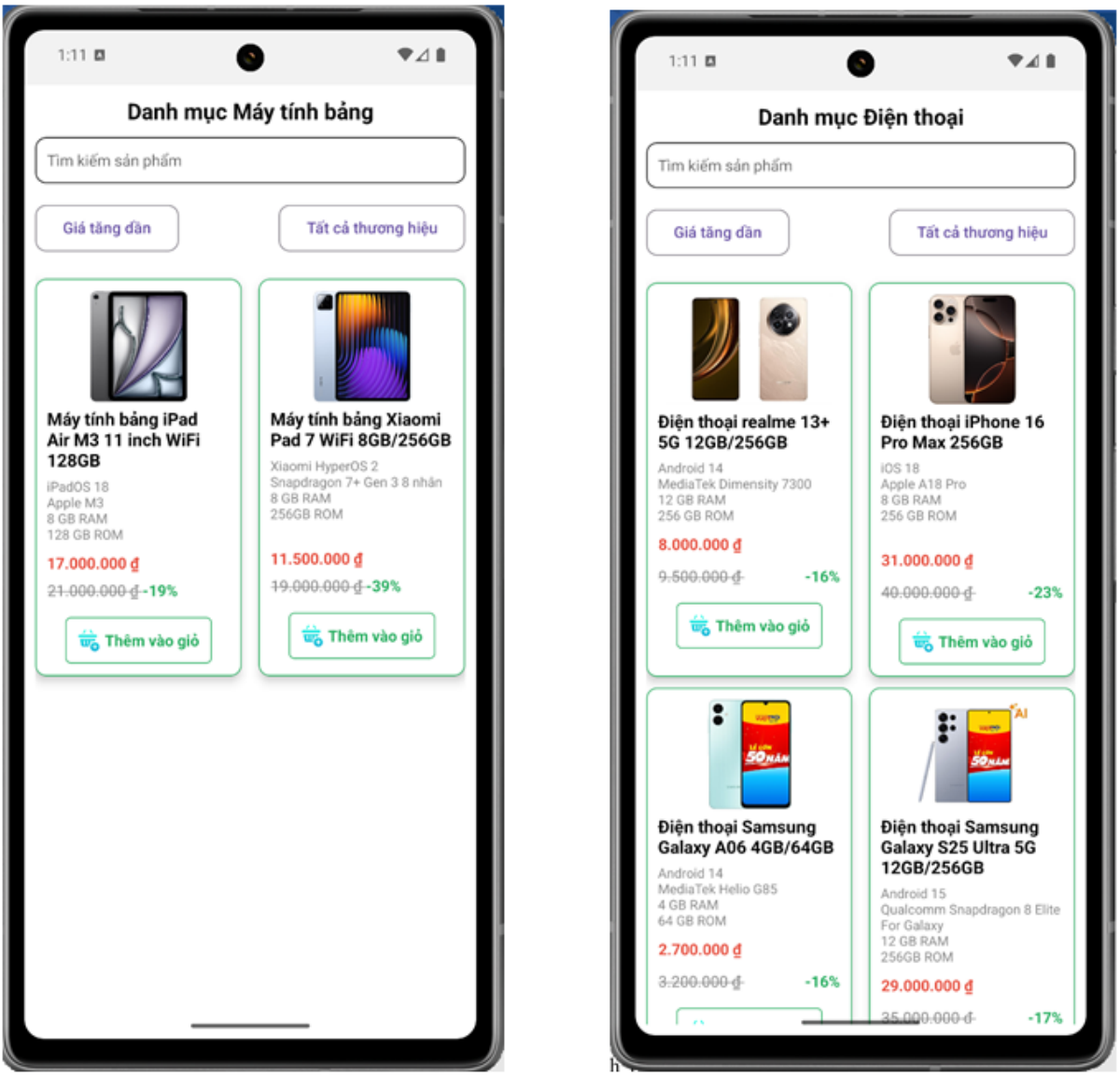
Mô tả: trang đăng ký cho phép người dùng tạo tài khoản mới trên hệ thống. Khi đăng ký, người dùng sẽ cung cấp các thông tin cần để tạo tài khoản và xác thực bằng mã OTP.

4.1.3 Giao diện trang đăng nhập

Hình 11. Giao diện đăng nhập

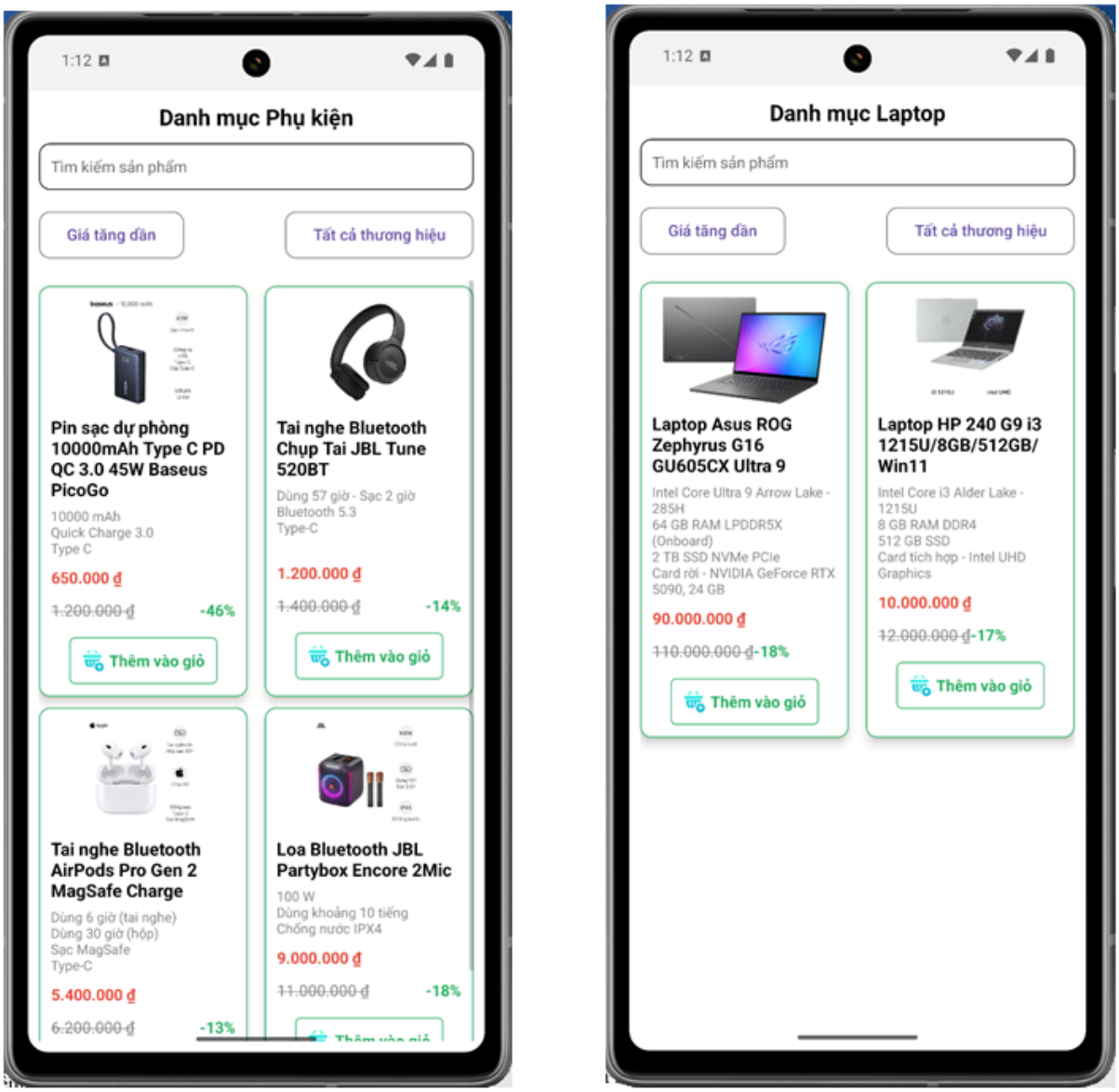
Mô tả: giao diện đăng nhập cho phép người dùng truy cập tài khoản cá nhân của mình nếu đã đăng ký tài khoản. Khi đăng nhập thành công, người dùng có thể sử dụng các tính năng của website.

4.1.4 Giao diện trang sản phẩm



Hình 12. Giao diện sản phẩm

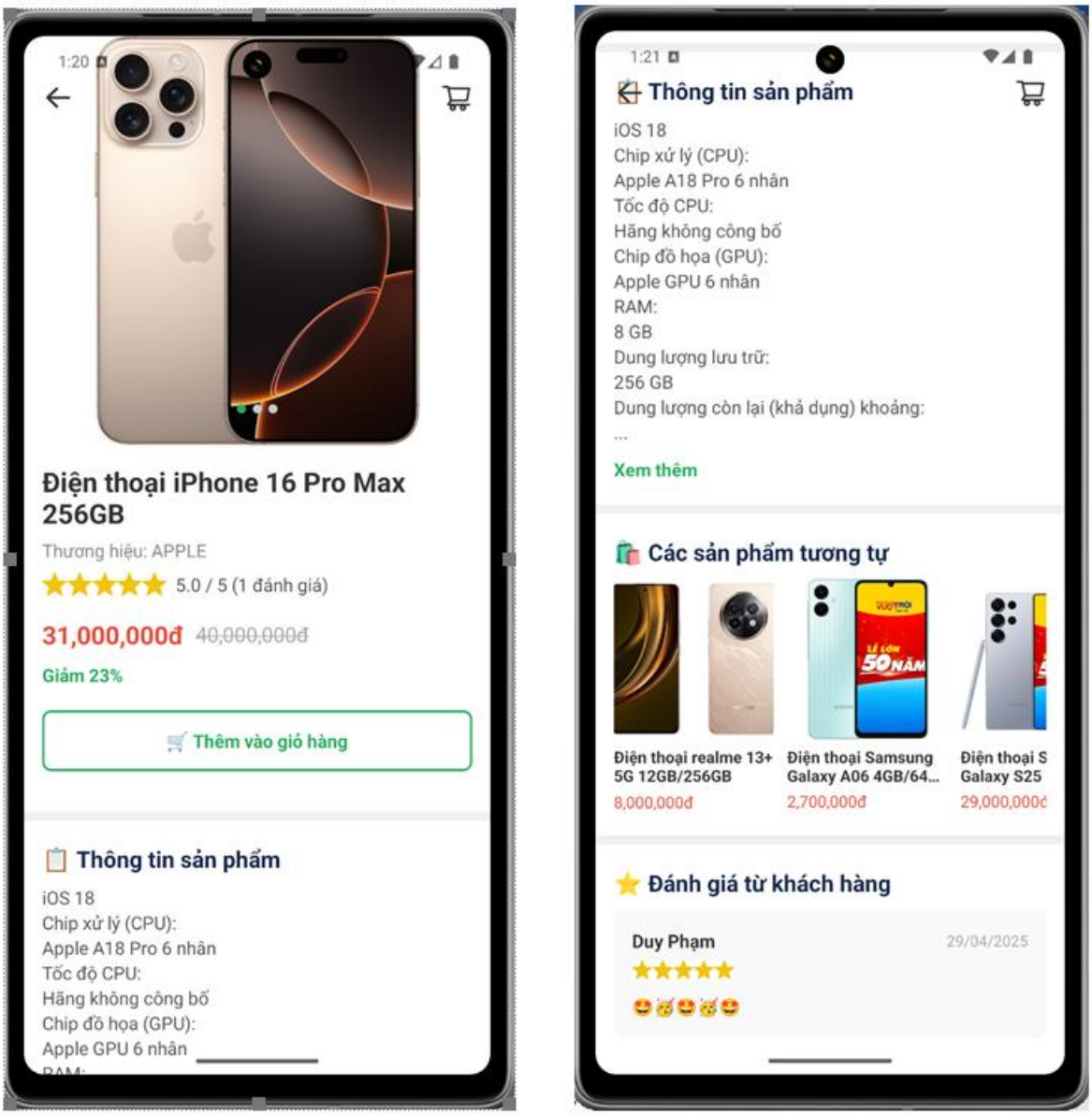
Mô tả: giao diện sản phẩm được thiết kế để giúp người dùng dễ dàng tìm kiếm và lựa chọn các sản phẩm mình mong muốn. Chức năng tìm kiếm cho phép người dùng nhanh chóng tìm ra sản phẩm bằng cách nhập tên sản phẩm vào ô tìm kiếm. Hệ thống sẽ trả về các kết quả liên quan ngay lập tức và chính xác.



Hình 13. Giao diện sản phẩm

Ngoài ra, giao diện còn hỗ trợ tính năng lọc sản phẩm theo thương hiệu sản phẩm, giúp người dùng tìm kiếm những sản phẩm phù hợp với nhu cầu cụ thể. Bên cạnh đó, người dùng có thể sắp xếp các sản phẩm theo giá từ thấp đến cao hoặc từ cao xuống thấp, giúp việc lựa chọn sản phẩm dễ dàng hơn, phù hợp với ngân sách và yêu cầu của khách hàng.

4.1.5 Giao diện trang chi tiết sản phẩm



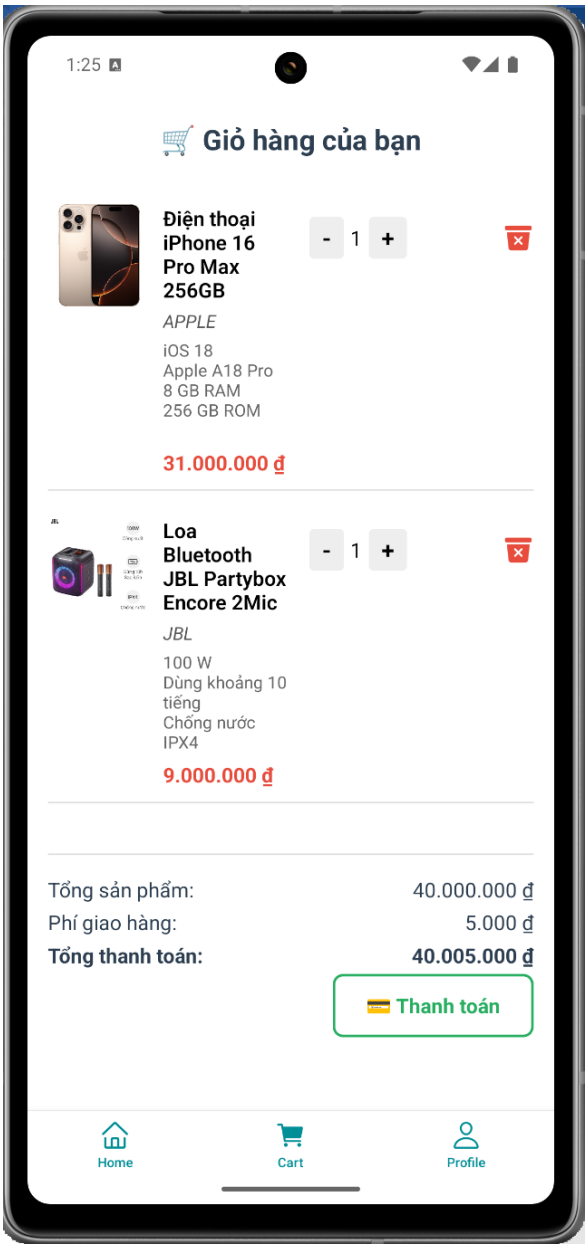
Hình 14. Giao diện chi tiết sản phẩm

Mô tả: khi người dùng chọn một sản phẩm, họ sẽ được chuyển đến trang chi tiết sản phẩm. Tại đây, người dùng có thể xem các hình ảnh chi tiết của sản phẩm, cùng với thông tin cụ thể sản phẩm và số sao đánh giá sản phẩm được tổng hợp từ người dùng. Điều này giúp người dùng nắm rõ về sản phẩm, từ đó đưa ra quyết định mua sắm chính xác.

Ngoài ra, người dùng có thể thêm sản phẩm vào giỏ hàng để dễ dàng tiếp tục mua sắm sau này. Giao diện cũng cho phép người dùng xem bình luận và đánh giá về

sản phẩm, chia sẻ trải nghiệm và cảm nhận của mình với cộng đồng, giúp những người mua khác có thêm thông tin trước khi quyết định mua sản phẩm.

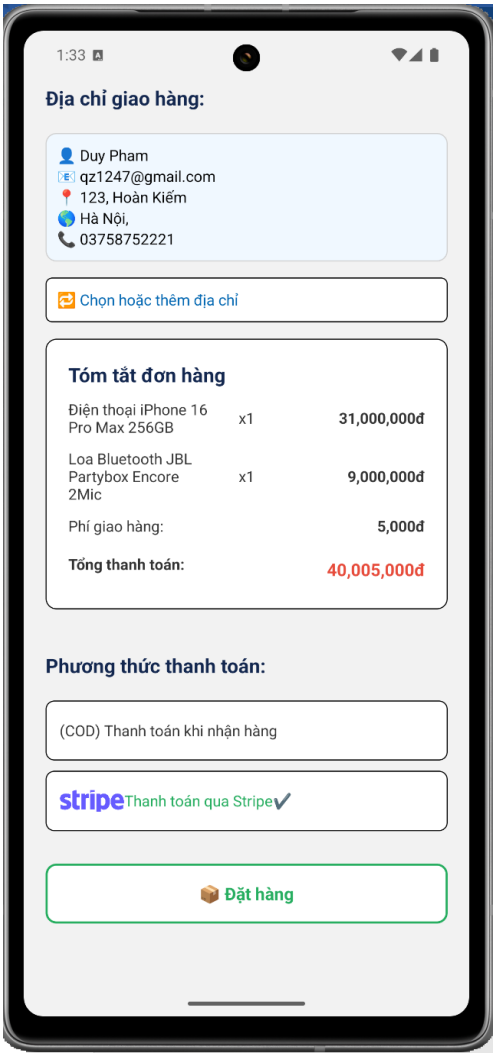
4.1.6 Giao diện trang giỏ hàng



Hình 15. Giao diện giỏ hàng

Mô tả: giao diện giỏ hàng giúp khách hàng theo dõi các sản phẩm đã chọn, điều chỉnh số lượng hoặc xóa sản phẩm khỏi giỏ. Tại đây, khách hàng cũng có thể kiểm tra tổng số tiền thanh toán trước khi tiếp tục các bước thanh toán.

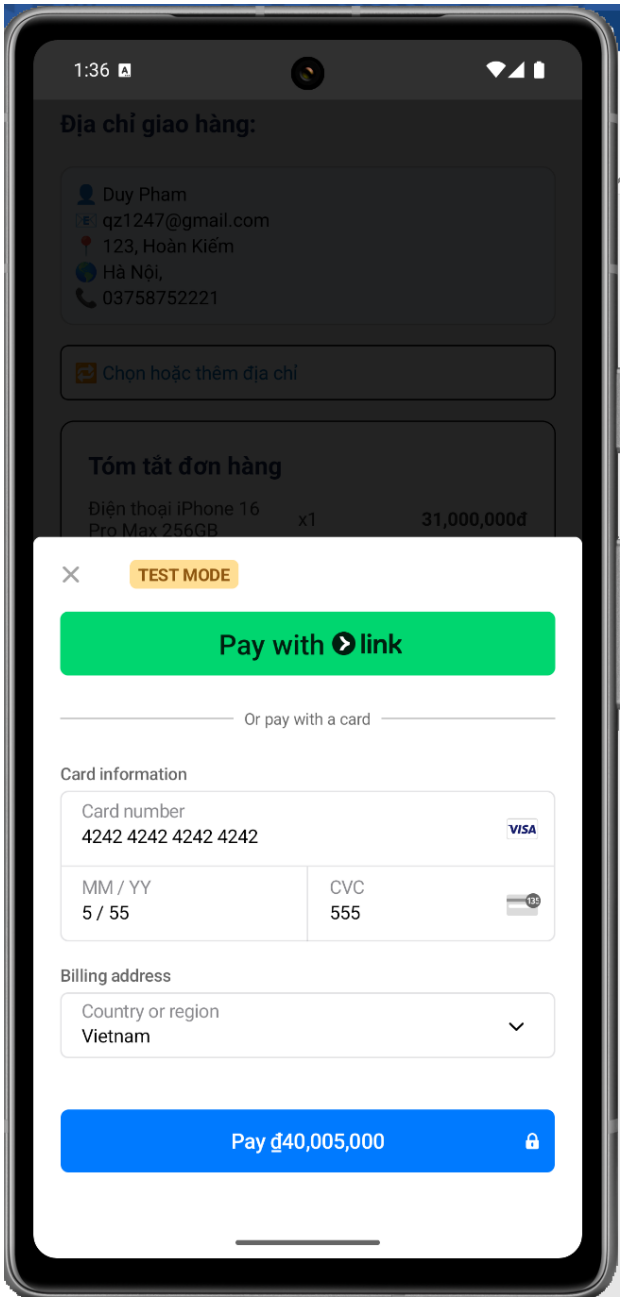
4.1.7 Giao diện đặt hàng



Hình 16. Giao diện đặt hàng

Mô tả: giao diện gồm địa chỉ giao hàng, tóm tắt đơn hàng và phương thức thanh toán bao gồm phần địa chỉ giao hàng yêu cầu khách hàng cung cấp thông tin cá nhân như họ tên, email, địa chỉ, số điện thoại để đảm bảo giao hàng chính xác. Phần tóm tắt đơn hàng hiển thị rõ tổng giá trị đơn hàng, phí vận chuyển và tổng tiền. Người dùng có thể chọn giữa thanh toán trực tuyến qua Stripe hoặc thanh toán khi nhận hàng (COD).

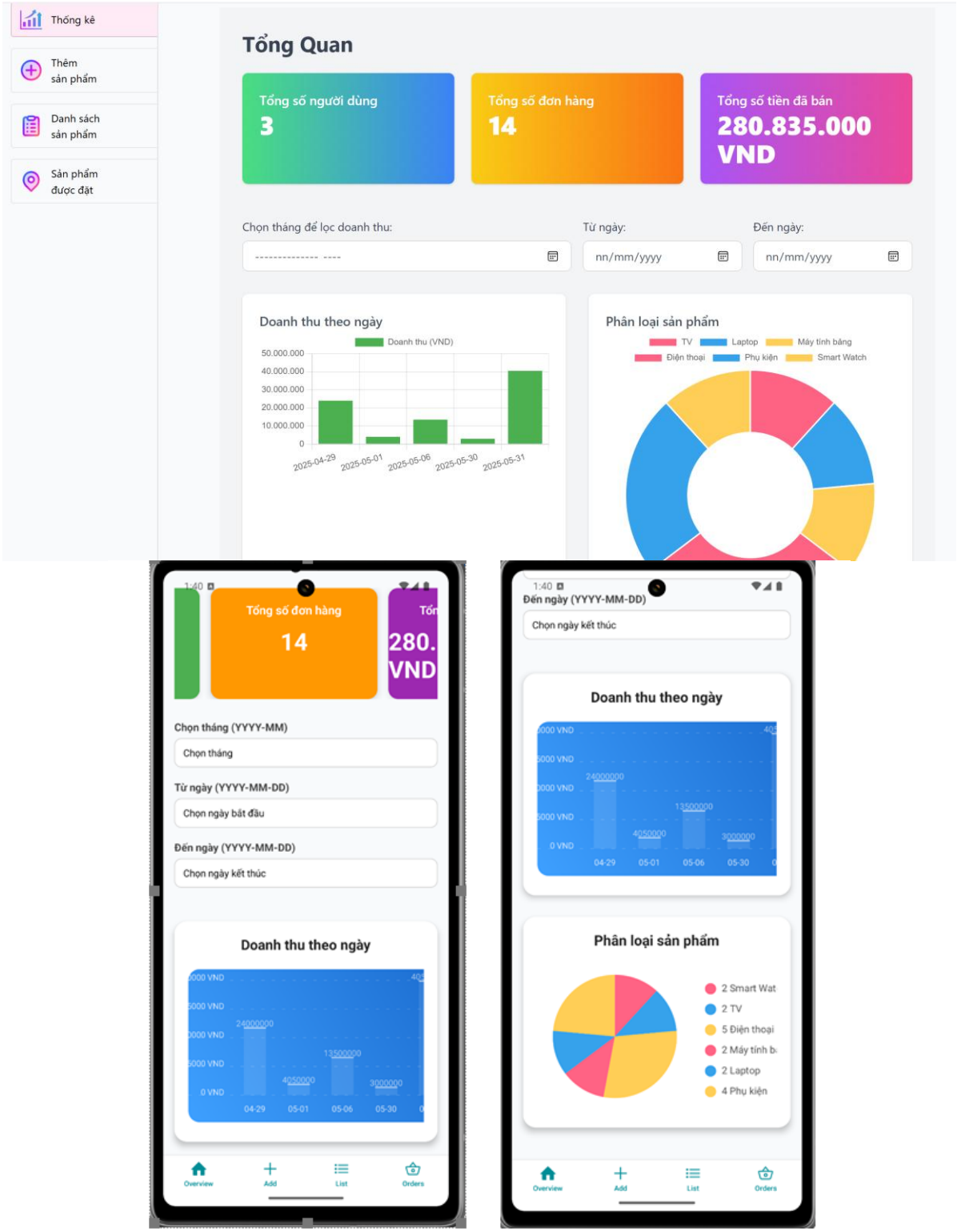
4.1.8 Giao diện thanh toán đơn hàng với Stripe



Hình 17. Giao diện thanh toán đơn hàng với Stripe

Mô tả: Stripe là một hình thức thanh toán trực tuyến an toàn và tiện lợi, được tích hợp vào hệ thống để hỗ trợ khách hàng thực hiện giao dịch một cách nhanh chóng. Người dùng có thể thanh toán trực tiếp bằng thẻ tín dụng hoặc thẻ ghi nợ quốc tế. Hệ thống đảm bảo bảo mật thông tin thanh toán nhờ vào các tiêu chuẩn mã hóa hiện đại, giúp người dùng yên tâm khi giao dịch. Đây là lựa chọn lý tưởng cho những khách hàng ưu tiên sự tiện lợi và mong muốn hoàn tất đơn hàng mà không cần sử dụng tiền mặt.

4.2 Giao diện quản trị



Hình 18. Giao diện quản trị

Mô tả: giao diện quản trị được xây dựng để tối ưu hóa việc quản lý và vận hành hệ thống. Người quản trị có thể dễ dàng thực hiện các thao tác như thêm mới, chỉnh sửa hoặc xóa dữ liệu liên quan đến sản phẩm và đơn hàng. Ngoài ra, hệ thống cung

cấp các công cụ thống kê chi tiết, giúp quản trị viên theo dõi tổng số đơn hàng, số lượng khách hàng, doanh thu và nhiều dữ liệu quan trọng khác, mang lại cái nhìn toàn diện về website.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Kết quả đạt được:

Về kiến thức và kỹ năng:

Lập trình front-end: Sử dụng React để xây dựng giao diện người dùng hiện đại, thân thiện với khách hàng.

Lập trình back-end: Xử lý dữ, quản lý server và xây dựng API liệu với NodeJS

Quản lý dự án: Lập kế hoạch chi tiết và thực hiện từng bước công việc từ nghiên cứu, thiết kế đến triển khai và báo cáo.

Xử lý vấn đề thực tế: Nâng cao kỹ năng giải quyết vấn đề và làm quen với cách phát triển một dự án thục tế.

Về ứng dụng:

Xây dựng thành công ứng dụng bán thiết bị công nghệ trên thiết bị di động, gồm cả frontend và backend, đáp ứng được đầy đủ các chức năng cốt lõi như tìm kiếm, giỏ hàng, thanh toán, đánh giá và xếp hạng sản phẩm, quản lý tài khoản và đơn hàng, bình luận.

Ưu điểm:

Giao diện thân thiện, hiện đại: Nhờ sử dụng React, ứng dụng mang lại trải nghiệm người dùng mượt mà và dễ sử dụng.

Chức năng đầy đủ và tiện ích: Hỗ trợ toàn bộ quy trình mua sắm trực tuyến từ tìm kiếm đến thanh toán và phản hồi người dùng.

Tính linh hoạt và mở rộng: Có thể dễ dàng mở rộng thêm tính năng như gợi ý sản phẩm, đa ngôn ngữ, hoặc tích hợp hệ thống thanh toán mới.

Kết nối giữa front-end và back-end hiệu quả: Nhờ API do Node.js xây dựng, đảm bảo trao đổi dữ liệu nhanh chóng, ổn định.

Khả năng ứng dụng thực tế cao: Phù hợp triển khai trong môi trường kinh doanh thực tế hoặc tiếp tục phát triển thành sản phẩm thương mại.

Hạn chế:

Chưa tối ưu hóa hiệu suất cho dữ liệu lớn: Khi số lượng người dùng và sản phẩm tăng cao, cần cải thiện hiệu suất hệ thống.

Chưa tích hợp đầy đủ các phương thức thanh toán phổ biến như ví điện tử nội địa, quét mã QR...

Thiếu tính năng bảo mật nâng cao: Cần bổ sung các cơ chế bảo vệ dữ liệu người dùng như mã hóa JWT, xác thực hai bước (2FA).

5.2 Hướng phát triển:

Tối ưu hiệu suất hệ thống: Nâng cao khả năng xử lý và phản hồi khi dữ liệu người dùng và số lượng sản phẩm tăng mạnh, đảm bảo ứng dụng hoạt động ổn định trong môi trường thực tế có lưu lượng cao.

Tích hợp AI: Ứng dụng trí tuệ nhân tạo để gợi ý sản phẩm phù hợp với nhu cầu và sở thích của khách hàng.

Mở rộng phạm vi: Xây dựng hệ thống đa ngôn ngữ để tiếp cận khách hàng quốc tế.

Phân tích dữ liệu: Sử dụng công cụ phân tích dữ liệu để cung cấp báo cáo chi tiết hơn về hành vi khách hàng và doanh thu.

DANH MỤC TÀI LIỆU THAM KHẢO

Sách/ Tài liệu:

[1] Đoàn Phước Miền, Phạm Thị Trúc Mai (2014), Thiết kế và lập trình Web, Trường Đại học Trà Vinh.

[2] Hà Thị Thúy Vi (2013), Cơ sở dữ liệu, Trường Đại học Trà Vinh.

[3] Phạm Minh Đương (2014), Phân tích và thiết kế hệ thống thông tin, Trường Đại học Trà Vinh.

Website:

[4] JavaScript là gì? Kiến thức cơ bản về JavaScript từ A-Z, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://mikotech.vn/javascript-la-gi/

[5] NodeJS là gì? NodeJS có phải ngôn ngữ lập trình, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://codegym.vn/blog/nodejs-la-gi-nodejs-co-phai-la-ngon-ngu-lap-trinh

[6] ExpressJS là gì?, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://fptshop.com.vn/tin-tuc/danh-gia/express-js-la-gi-174976

[7] RESTful API là gì? Các nguyên tắc của RESTful API, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://stringee.com/vi/blog/post/restful-api-la-gi

[8] ReactJS là gì?, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://freetuts.net/reactjs-la-gi-cac-khai-niem-co-ban-2343.html

[9] MongoDB là gì?, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://itnavi.com.vn/blog/mongodb-la-gi

[10] Thanh toán điện tử là gì?, Truy cập ngày: 21/06/2025, Nguồn tham khảo: https://www.vib.com.vn/vn/cam-nang/ngan-hang-so/tien-ich-va-trai-nghiem/thanh-toan-dien-tu

PHỤ LỤC