

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC DUY TÂN**



**Vũ Tiến Thành**

**PHÁT TRIỂN ỨNG DỤNG KẾT NỐI DOANH NGHIỆP VÀ KHÁCH HÀNG QUA NỀN TẢNG TRAO ĐỔI SẢN PHẨM TRỰC TUYẾN TÍCH HỢP ĐA CỬA HÀNG VỚI FLUTTER, NODEJS, EXPRESS VÀ MONGODB**

CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM

**TÓM TẮT BÁO CÁO**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Nguyễn Hữu Phúc

Đà Nẵng, tháng 5 năm 2025

# MỤC LỤC

[MỤC LỤC 1](#_Toc198284124)

[MỞ ĐẦU 1](#_Toc198284127)

[1. Lý do chọn đề tài 1](#_Toc198284128)

[2. Mục đích và ý nghĩa của đề tài 1](#_Toc198284129)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc198284130)

[4. Phương pháp nghiên cứu 2](#_Toc198284131)

[5. Kết quả dự kiến 2](#_Toc198284132)

[6. Bố cục đề tài 2](#_Toc198284133)

[CHƯƠNG I 3](#_Toc198284134)

[TỔNG QUAN VỀ CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN 3](#_Toc198284135)

[1.1. GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH DART 3](#_Toc198284136)

[1.1.1 Dart là gì? 3](#_Toc198284137)

[1.1.2. Ưu và nhược điểm của Dart 3](#_Toc198284138)

[1.2. GIỚI THIỆU VỀ FLUTTER 4](#_Toc198284139)

[1.2.1. Flutter là gì? 4](#_Toc198284140)

[1.2.2. Ưu và nhược điểm của Flutter 4](#_Toc198284141)

[1.3. GIỚI THIỆU VỀ NODEJS 5](#_Toc198284142)

[1.3.1. Nodejs là gì? 5](#_Toc198284143)

[1.3.2. Ưu và nhược điểm của Nodejs 5](#_Toc198284144)

[1.4. Giới thiệu về Express 6](#_Toc198284145)

[1.4.1. Express là gì? 6](#_Toc198284146)

[1.4.2. Ưu và nhược điểm của Express 6](#_Toc198284147)

[1.5. Giới thiệu về MongoDB 7](#_Toc198284148)

[1.5.1. MongoDB là gì? 7](#_Toc198284149)

[1.5.2. Ưu và nhược điểm của MongoDB 7](#_Toc198284150)

[1.6. Giới thiệu về dự án 8](#_Toc198284151)

[CHƯƠNG II 9](#_Toc198284152)

[TỔNG QUAN HỆ THỐNG 9](#_Toc198284153)

[2.1. Khảo sát thực trạng 9](#_Toc198284154)

[2.2. Các đối tượng tương tác với hệ thống 9](#_Toc198284155)

[2.3. SƠ ĐỒ USECASE SỬ DỤNG (USECASE DIAGRAM) 10](#_Toc198284156)

[2.3.1. Use case tổng quát 10](#_Toc198284157)

[2.4. Cơ sở dữ liệu 11](#_Toc198284158)

[2.4.1. Lựa chọn cơ sở dữ liệu 11](#_Toc198284159)

[2.4.2. Lược đồ cơ sở dữ liệu 12](#_Toc198284160)

[CHƯƠNG III 13](#_Toc198284161)

[USER INTERFACE DESIGN 13](#_Toc198284162)

[2.1. Giao diện người dùng 13](#_Toc198284163)

[2.1.1. Giao diện thiết kế bắt đầu, đăng ký, đăng nhập 13](#_Toc198284164)

[2.1.2. Giao diện trang chủ 14](#_Toc198284165)

[2.1.3. Giao diện giỏ hàng, chỉnh sửa thông tin 15](#_Toc198284166)

[2.2. Giao diện admin 16](#_Toc198284167)

[2.2.1. Giao diện admin DashBoard 16](#_Toc198284168)

[2.2.2. Giao diện admin đơn hàng 17](#_Toc198284169)

[2.2.3. Giao diện danh sách người dùng 18](#_Toc198284170)

[CHƯƠNG III 19](#_Toc198284171)

[DEMO CHƯƠNG TRÌNH 19](#_Toc198284172)

[3.1. Giao diện người dùng 19](#_Toc198284173)

[3.1.1. Giao diện đăng ký, đăng nhập 19](#_Toc198284174)

[3.1.2. Giao diện trang chủ, danh mục sản phẩm 20](#_Toc198284175)

[3.1.3. Giao diện giỏ hàng, thêm yêu thích 21](#_Toc198284176)

[3.1.4. Giao diện thanh toán 22](#_Toc198284177)

[3.1.6. Giao diện cửa hàng 23](#_Toc198284178)

[3.2. Giao diện nhà bán hàng 24](#_Toc198284179)

[3.2.1. Giao diện người bán thống kê, tải sản phẩm 24](#_Toc198284180)

[3.2.2. Giao diện chỉnh sửa sản phẩm 25](#_Toc198284181)

[3.2.3. Giao diện người bán đơn hàng 26](#_Toc198284182)

[3.3. Giao diện admin 27](#_Toc198284183)

[3.3.1. Giao diện admin quản lý người dùng 27](#_Toc198284184)

[3.3.2. Giao diện admin đơn hàng 27](#_Toc198284185)

[3.3.3. Giao diện admin upload danh mục hàng 28](#_Toc198284186)

[CHƯƠNG V 29](#_Toc198284187)

[KẾT LUẬN 29](#_Toc198284188)

# MỤC LỤC HÌNH

[Hình 1: Use case tổng quát 11](#_Toc198284266)

[Hình 2: Giao diện thiết kế bắt đầu, đăng ký, đăng nhập 13](#_Toc198284267)

[Hình 3: Giao diện thiết kế trang chủ 14](#_Toc198284268)

[Hình 5: Giao diện giỏ hàng, chỉnh sửa thông tin 15](#_Toc198284269)

[Hình 6: Giao diện admin DashBoard 16](#_Toc198284270)

[Hình 8: Giao diện admin danh sách người dùng 18](#_Toc198284271)

[Hình 11: Giao diện đăng ký, đăng nhập 19](#_Toc198284272)

[Hình 12: Giao diện trang chủ, danh mục sản phẩm 20](#_Toc198284273)

[Hình 13: Giao diện giỏ hàng, thêm yêu thích 22](#_Toc198284274)

[Hình 14: Giao diện thanh toán 23](#_Toc198284275)

[Hình 16: Giao diện cửa hàng 23](#_Toc198284276)

[Hình 17: Giao diện người bán thống kê, tải sản phẩm 24](#_Toc198284277)

[Hình 18: Giao diện chỉnh sửa thông tin 26](#_Toc198284278)

[Hình 19: Giao diện người bán đơn hàng 26](#_Toc198284279)

[Hình 21: Giao diện admin quản lý người dùng 27](#_Toc198284280)

[Hình 22: Giao diện admin quản lý đơn hàng 28](#_Toc198284281)

[Hình 23: Giao diện admin thêm danh mục 28](#_Toc198284282)

# MỤC LỤC BẢNG

[Bảng 1: Các đối tượng tương tác với hệ thống 10](#_Toc198284290)

[Bảng 2: bảng cơ sở dữ liệu 12](#_Toc198284291)

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Trong thời đại công nghệ 4.0 hiện nay, nhu cầu sử dụng các ứng dụng mobile để kết nối doanh nghiệp với khách hàng ngày càng tăng cao. Các nền tảng trao đổi sản phẩm trực tuyến đang trở thành xu hướng phổ biến nhằm cung cấp một phương thức tiện lợi, hiệu quả và nhanh chóng cho việc mua bán và trao đổi sản phẩm. Việc tích hợp đa cửa hàng trong cùng một nền tảng không chỉ giúp mở rộng đối tượng khách hàng mà còn tạo điều kiện thuận lợi cho việc quản lý và vận hành của các doanh nghiệp.

Vì vậy, việc nghiên cứu và phát triển một ứng dụng mobile tích hợp đa cửa hàng sử dụng các công nghệ tiên tiến như Flutter, Nodejs, Express và MongoDB là cần thiết và có ý nghĩa thực tiễn cao. Ứng dụng này không chỉ cung cấp trải nghiệm mượt mà cho người dùng mà còn đảm bảo tính bảo mật, mở rộng và hiệu quả trong việc quản lý dữ liệu.

## 2. Mục đích và ý nghĩa của đề tài

1. Mục đích

* Xây dựng một ứng dụng mobile đa nền tảng (Android và iOS) sử dụng Flutter.
* Kết nối ứng dụng với backend được xây dựng bằng Nodejs, Express và MongoDB.
* Cung cấp nền tảng trao đổi sản phẩm trực tuyến tích hợp đa cửa hàng, cho phép người dùng dễ dàng mua bán sản phẩm và các doanh nghiệp quản lý sản phẩm của mình hiệu quả.

1. Ý nghĩa

* Cung cấp một nền tảng ứng dụng mới giúp doanh nghiệp tiếp cận khách hàng một cách thuận tiện, nhanh chóng và hiệu quả.
* Tăng cường tính cạnh tranh của doanh nghiệp trong lĩnh vực thương mại điện tử.
* Góp phần nâng cao trải nghiệm của người dùng trong việc trao đổi và mua bán sản phẩm.

## 3. Đối tượng và phạm vi nghiên cứu

1. Đối tượng

* Các công nghệ phát triển ứng dụng mobile đa nền tảng (Flutter)
* Các công nghệ backend (Nodejs, Express).
* Cơ sở dữ liệu NoSQL (MongoDB).

1. Phạm vi

* Phát triển ứng dụng mobile tích hợp đa cửa hàng, bao gồm: giao diện người dùng, hệ thống quản lý sản phẩm, giỏ hàng, thanh toán.
* Xây dựng API backend để kết nối ứng dụng mobile với cơ sở dữ liệu.
* Triển khai hệ thống trên môi trường thử nghiệm và đánh giá hiệu quả.

## 4. Phương pháp nghiên cứu

* Tìm hiểu lý thuyết về Flutter, Nodejs, Express và MongoDB.
* Phân tích các yêu cầu chức năng của hệ thống.
* Thiết kế và xây dựng hệ thống dựa trên các công nghệ đã chọn.
* Thử nghiệm và đánh giá hiệu quả của ứng dụng.
* Góp ý của các chuyên gia.

## 5. Kết quả dự kiến

* Ứng dụng mobile đa nền tảng hoàn thiện với các chức năng cơ bản của hệ thống quản lý đa cửa hàng.
* Backend hỗ trợ đầy đủ các API cần thiết cho ứng dụng mobile.
* Đánh giá tính hiệu quả, ổn định và bảo mật của hệ thống.

## 6. Bố cục đề tài

* Nội dung của đề tài gồm các phần:
* Chương 1: Tổng quan về công cụ và môi trường phát triển.
* Chương 2: Phân tích và thiết kế hệ thống.
* Chương 3: Cài đặt và kiểm thử.
* Kết luận và hướng phát triển.

# CHƯƠNG I

# TỔNG QUAN VỀ CÔNG CỤ VÀ MÔI TRƯỜNG PHÁT TRIỂN

## GIỚI THIỆU NGÔN NGỮ LẬP TRÌNH DART

##### 1.1.1 Dart là gì?

Dart là một ngôn ngữ lập trình mã nguồn mở, được phát triển bởi Google, tập trung vào việc xây dựng các ứng dụng frontend đa nền tảng. Dart được thiết kế để hỗ trợ phát triển giao diện người dùng (UI) với cú pháp đơn giản, hiệu suất cao, và tích hợp chặt chẽ với Flutter để phát triển ứng dụng mobile, web và desktop.

##### 1.1.2. Ưu và nhược điểm của Dart

Ưu điểm:

Cú pháp đơn giản và dễ học: Dart có cú pháp tương tự JavaScript và Java, giúp các lập trình viên quen thuộc với các ngôn ngữ này dễ dàng làm quen.

Hiệu suất cao: Dart hỗ trợ biên dịch JIT (Just-In-Time) để phát triển nhanh và AOT (Ahead-Of-Time) để tạo mã máy native, đảm bảo hiệu suất tốt cho ứng dụng.

Tích hợp chặt chẽ với Flutter: Dart được thiết kế để hoạt động tối ưu với Flutter, giúp phát triển giao diện người dùng mượt mà và hiệu quả.

Hỗ trợ lập trình hướng đối tượng: Dart cung cấp các tính năng như class, interface, và mixin, phù hợp cho phát triển ứng dụng phức tạp.

Cộng đồng phát triển từ Google: Dart được Google hỗ trợ mạnh mẽ, đảm bảo sự ổn định và cập nhật thường xuyên.

Nhược điểm:

Ít phổ biến ngoài Flutter: Dart chủ yếu được sử dụng với Flutter, nên phạm vi ứng dụng và cộng đồng hỗ trợ nhỏ hơn so với các ngôn ngữ như JavaScript hay Python.

Thiếu tài nguyên học tập: So với các ngôn ngữ lập trình phổ biến, tài liệu và tài nguyên học Dart còn hạn chế, đặc biệt với các lập trình viên mới.

Hiệu suất không tối ưu cho backend: Dart không được thiết kế cho phát triển backend, nên ít được sử dụng trong các ứng dụng phía server so với JavaScript (NodeJS) hay Python.

## GIỚI THIỆU VỀ FLUTTER

##### Flutter là gì?

Flutter là một bộ công cụ giao diện người dùng (UI toolkit) mã nguồn mở do Google phát triển, cho phép các lập trình viên xây dựng các ứng dụng đa nền tảng (mobile, web, và desktop) từ một codebase duy nhất. Flutter sử dụng ngôn ngữ lập trình Dart và cung cấp một bộ sưu tập các widget (thành phần giao diện) phong phú, cho phép tạo ra các giao diện đẹp mắt, mượt mà và hiệu suất cao, gần tương đương với ứng dụng native.

##### Ưu và nhược điểm của Flutter

Ưu điểm:

Đa nền tảng từ một codebase duy nhất: Flutter cho phép phát triển ứng dụng chạy trên nhiều nền tảng (Android, iOS, web, desktop) mà không cần viết lại mã, giúp tiết kiệm thời gian và chi phí.

Hiệu suất cao: Nhờ biên dịch AOT (Ahead-Of-Time) sang mã máy native, Flutter mang lại hiệu suất gần tương đương với ứng dụng native, đặc biệt trong việc xử lý giao diện phức tạp và hoạt ảnh.

Tính năng Hot Reload: Flutter hỗ trợ Hot Reload, cho phép lập trình viên xem thay đổi giao diện ngay lập tức mà không cần biên dịch lại toàn bộ ứng dụng, tăng tốc độ phát triển.

Giao diện tùy chỉnh linh hoạt: Flutter cung cấp hệ thống widget phong phú, cho phép tùy chỉnh giao diện theo ý muốn mà không phụ thuộc vào các thành phần giao diện mặc định của hệ điều hành.

Cộng đồng lớn và tài liệu phong phú: Flutter có cộng đồng phát triển mạnh mẽ, với nhiều tài liệu, thư viện, và công cụ hỗ trợ từ Google và các lập trình viên trên toàn cầu.

Nhược điểm:

Kích thước ứng dụng lớn: Do Flutter tích hợp cả engine riêng (Skia) và các widget, kích thước ứng dụng thường lớn hơn so với các ứng dụng native, đặc biệt với các ứng dụng nhỏ.

Hạn chế trong tích hợp native: Mặc dù Flutter hỗ trợ tích hợp với mã native (qua Platform Channels), việc này đòi hỏi kiến thức sâu về nền tảng (Android, iOS) và có thể phức tạp trong một số trường hợp.

Hỗ trợ thư viện chưa đầy đủ: So với các framework lâu đời như React Native, hệ sinh thái thư viện của Flutter vẫn đang phát triển, một số tính năng hoặc thư viện bên thứ ba có thể chưa hoàn thiện.

Khúc học tập ban đầu: Đối với lập trình viên mới, việc làm quen với Dart và hệ thống widget của Flutter có thể mất thời gian, đặc biệt nếu chưa quen với lập trình declarative (lập trình khai báo).

## GIỚI THIỆU VỀ NODEJS

##### Nodejs là gì?

NodeJS là một môi trường runtime mã nguồn mở, cho phép chạy JavaScript ở phía server, được xây dựng trên nền tảng V8 JavaScript Engine của Google Chrome. NodeJS được thiết kế để phát triển các ứng dụng web hiệu suất cao, không đồng bộ (asynchronous), và có khả năng mở rộng tốt, đặc biệt phù hợp cho các ứng dụng thời gian thực và xử lý I/O nặng.

##### Ưu và nhược điểm của Nodejs

Ưu điểm:

Không đồng bộ và hiệu suất cao: NodeJS sử dụng mô hình không đồng bộ (event-driven), cho phép xử lý nhiều yêu cầu đồng thời mà không bị chặn, rất phù hợp cho các ứng dụng thời gian thực.

Hệ sinh thái phong phú: Với npm (Node Package Manager), NodeJS có hệ sinh thái thư viện lớn nhất thế giới, giúp lập trình viên dễ dàng tích hợp các công cụ và thư viện.

Dễ học với JavaScript: NodeJS sử dụng JavaScript – ngôn ngữ phổ biến nhất trong lập trình web, giúp lập trình viên frontend dễ dàng chuyển sang phát triển backend.

Khả năng mở rộng: NodeJS hỗ trợ mở rộng theo chiều ngang (horizontal scaling) thông qua các công cụ như PM2 hoặc container hóa (Docker).

Cộng đồng lớn: NodeJS có cộng đồng phát triển rộng lớn, với nhiều tài liệu, hướng dẫn và hỗ trợ từ các công ty lớn như Netflix, PayPal, và Walmart.

Nhược điểm:

Không tối ưu cho tác vụ CPU nặng: Do sử dụng mô hình single-threaded, NodeJS không hiệu quả trong các tác vụ tính toán nặng (như xử lý dữ liệu phức tạp hoặc mã hóa), dễ bị chặn nếu không quản lý tốt.

Callback hell: Mô hình không đồng bộ có thể dẫn đến mã phức tạp (callback hell) nếu không sử dụng các công cụ như Promises hoặc async/await.

Hiệu suất không ổn định với dữ liệu lớn: Với các ứng dụng yêu cầu xử lý dữ liệu lớn và phức tạp, NodeJS có thể gặp khó khăn nếu không được tối ưu hóa tốt.

Khúc học tập với mô hình không đồng bộ: Đối với lập trình viên quen với mô hình đồng bộ (như PHP hoặc Java), việc làm quen với lập trình không đồng bộ của NodeJS có thể mất thời gian.

## Giới thiệu về Express

##### Express là gì?

Express là một framework mã nguồn mở, nhẹ và nhanh, được xây dựng trên nền tảng NodeJS, chuyên dùng để phát triển các ứng dụng web và API RESTful. Express cung cấp các công cụ và thư viện cần thiết để xử lý các yêu cầu HTTP, định tuyến (routing), và tích hợp middleware, giúp lập trình viên xây dựng backend một cách đơn giản và hiệu quả.

##### Ưu và nhược điểm của Express

Ưu điểm:

Tối giản và linh hoạt: Express là một framework nhẹ, không áp đặt cấu trúc cố định, cho phép lập trình viên tự do thiết kế ứng dụng theo ý muốn.

Hỗ trợ mạnh mẽ cho API RESTful: Express cung cấp các công cụ định tuyến và xử lý HTTP đơn giản, rất phù hợp để xây dựng API RESTful.

Tích hợp middleware: Express hỗ trợ middleware (như body-parser, cors), giúp dễ dàng xử lý các tác vụ như phân tích dữ liệu, xác thực, hoặc logging.

Tích hợp tốt với NodeJS: Là framework dựa trên NodeJS, Express tận dụng toàn bộ sức mạnh của NodeJS, từ mô hình không đồng bộ đến hệ sinh thái npm.

Cộng đồng lớn: Express có cộng đồng phát triển rộng lớn, với nhiều tài liệu, hướng dẫn và thư viện hỗ trợ.

Nhược điểm:

Thiếu tính năng tích hợp sẵn: So với các framework đầy đủ như Django hoặc Laravel, Express thiếu các tính năng tích hợp sẵn (như ORM, authentication), đòi hỏi lập trình viên phải tự cài đặt thêm thư viện.

Khúc học tập với middleware: Việc sử dụng middleware có thể phức tạp với người mới, đặc biệt khi cần xử lý nhiều middleware trong một ứng dụng lớn.

Không tối ưu cho ứng dụng phức tạp: Với các ứng dụng lớn và phức tạp, Express có thể thiếu cấu trúc rõ ràng, khiến việc quản lý mã nguồn trở nên khó khăn nếu không được tổ chức tốt.

Phụ thuộc vào NodeJS: Express kế thừa các hạn chế của NodeJS, như hiệu suất kém với các tác vụ CPU nặng.

## Giới thiệu về MongoDB

##### MongoDB là gì?

MongoDB là một cơ sở dữ liệu NoSQL mã nguồn mở, lưu trữ dữ liệu dưới dạng tài liệu (document) JSON (hoặc BSON), thay vì bảng như các cơ sở dữ liệu quan hệ (SQL). MongoDB được thiết kế để xử lý dữ liệu không đồng nhất, có khả năng mở rộng cao và phù hợp cho các ứng dụng yêu cầu tốc độ truy vấn nhanh và khả năng xử lý dữ liệu lớn.

##### Ưu và nhược điểm của MongoDB

Ưu điểm:

Linh hoạt với dữ liệu không đồng nhất: MongoDB không yêu cầu schema cố định, cho phép lưu trữ dữ liệu với cấu trúc khác nhau trong cùng một collection, rất phù hợp cho các ứng dụng có dữ liệu thay đổi thường xuyên.

Khả năng mở rộng cao: MongoDB hỗ trợ sharding (phân mảnh dữ liệu) và replication (sao chép dữ liệu), giúp mở rộng cơ sở dữ liệu theo chiều ngang dễ dàng khi lượng dữ liệu tăng.

Hiệu suất truy vấn nhanh: Với thiết kế dựa trên tài liệu và hỗ trợ index, MongoDB cung cấp tốc độ truy vấn nhanh, đặc biệt với dữ liệu lớn.

Tích hợp tốt với JavaScript: MongoDB lưu trữ dữ liệu dưới dạng JSON/BSON, tích hợp mượt mà với JavaScript (NodeJS) và các framework như Express thông qua thư viện Mongoose.

Cộng đồng lớn và tài liệu phong phú: MongoDB có cộng đồng phát triển rộng lớn, với nhiều tài liệu, hướng dẫn và công cụ hỗ trợ.

Nhược điểm:

Không tối ưu cho mối quan hệ phức tạp: MongoDB không hỗ trợ tốt các mối quan hệ phức tạp (join) như cơ sở dữ liệu quan hệ (SQL), đòi hỏi lập trình viên phải xử lý mối quan hệ thủ công.

Dung lượng lưu trữ lớn: Do lưu trữ dữ liệu dưới dạng tài liệu JSON/BSON, MongoDB có thể tiêu tốn nhiều dung lượng lưu trữ hơn so với cơ sở dữ liệu quan hệ.

Tính toàn vẹn dữ liệu thấp hơn: Vì không có schema cố định, MongoDB dễ dẫn đến dữ liệu không đồng nhất nếu không được quản lý tốt.

## Giới thiệu về dự án

Trong đề tài này, Dart và Flutter được sử dụng để phát triển ứng dụng mobile đa nền tảng, đảm bảo giao diện mượt mà và hiệu suất cao trên cả Android và iOS. NodeJS và Express được dùng để xây dựng API backend RESTful, cung cấp các endpoint để xử lý yêu cầu từ ứng dụng mobile. MongoDB được chọn làm cơ sở dữ liệu để lưu trữ và quản lý thông tin đa cửa hàng, từ sản phẩm, người dùng, đến đơn hàng, nhờ tính linh hoạt và khả năng mở rộng của nó.

# CHƯƠNG II

# TỔNG QUAN HỆ THỐNG

## 2.1. Khảo sát thực trạng

Hiện nay, nhu cầu phát triển các nền tảng trao đổi sản phẩm trực tuyến đang ngày càng gia tăng do xu hướng chuyển đổi số trong thương mại điện tử. Các hệ thống hiện tại chủ yếu được xây dựng để phục vụ một cửa hàng hoặc một nền tảng bán hàng cụ thể, chưa có nhiều giải pháp tích hợp đa cửa hàng trên cùng một ứng dụng.

Bên cạnh đó, việc sử dụng các công nghệ cũ, thiếu tính mở rộng và tích hợp khiến cho việc quản lý sản phẩm và giao dịch của các doanh nghiệp trở nên phức tạp. Việc xây dựng một hệ thống sử dụng các công nghệ hiện đại như Flutter, Nodejs, Express và MongoDB sẽ giúp khắc phục các hạn chế này và mang lại một giải pháp tối ưu hơn.

Vì vậy, việc nghiên cứu và phát triển một ứng dụng mobile tích hợp đa cửa hàng là cần thiết để giải quyết các vấn đề trên, đồng thời tạo ra một nền tảng tiện ích cho cả doanh nghiệp và khách hàng.

## Các đối tượng tương tác với hệ thống

Hệ thống có các tác nhân: Người dùng chưa đăng nhập, Người dùng đã đăng nhập và quản trị viên, người bán hàng. Vai trò của các tác nhân được thể hiện trong bảng như sau:

|  |  |  |
| --- | --- | --- |
| STT | Tên tác nhân | Mô tả |
| 1 | Người dùng đã đăng nhập | Người dùng đã tạo tài khoản và đăng nhập thành công, có thể thực hiện các thao tác như mua sản phẩm, xem chi tiết sản phẩm, đánh giá sản phẩm, thêm vào giỏ hàng và thanh toán. |
| 2 | Quản trị viên | Người quản lý hệ thống, có quyền quản lý người dùng, sản phẩm, đơn hàng và các báo cáo. |
| 3 | Người bán hàng | Các doanh nghiệp hoặc cá nhân đăng ký bán hàng trên nền tảng, có thể đăng sản phẩm và quản lý các sản phẩm của mình. |

Bảng 1: Các đối tượng tương tác với hệ thống

## SƠ ĐỒ USECASE SỬ DỤNG (USECASE DIAGRAM)

### Use case tổng quát



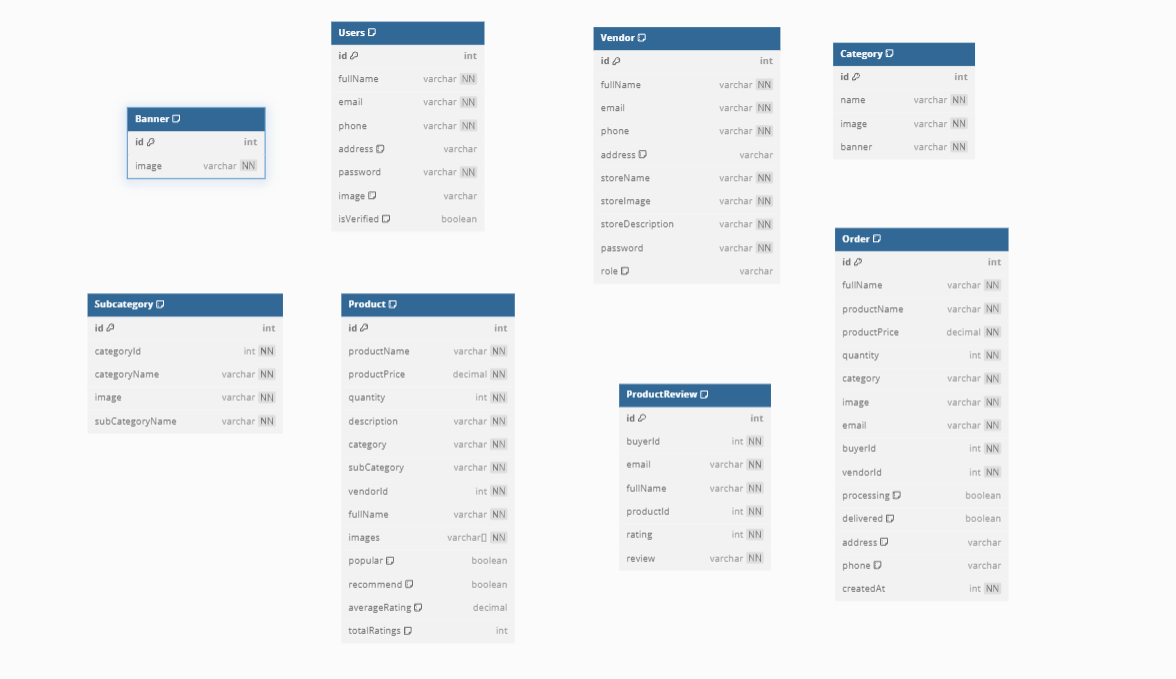
###### Hình 1: Use case tổng quát

## Cơ sở dữ liệu

### 2.4.1. Lựa chọn cơ sở dữ liệu

Trong quá trình phát triển ứng dụng mobile tích hợp đa cửa hàng, việc lựa chọn cơ sở dữ liệu phù hợp đóng vai trò rất quan trọng nhằm đảm bảo tính ổn định, mở rộng và quản lý dữ liệu hiệu quả. Sau khi nghiên cứu và so sánh các hệ quản trị cơ sở dữ liệu phổ biến, đã quyết định sử dụng MongoDB làm cơ sở dữ liệu chính cho hệ thống.

### 2.4.2. Lược đồ cơ sở dữ liệu



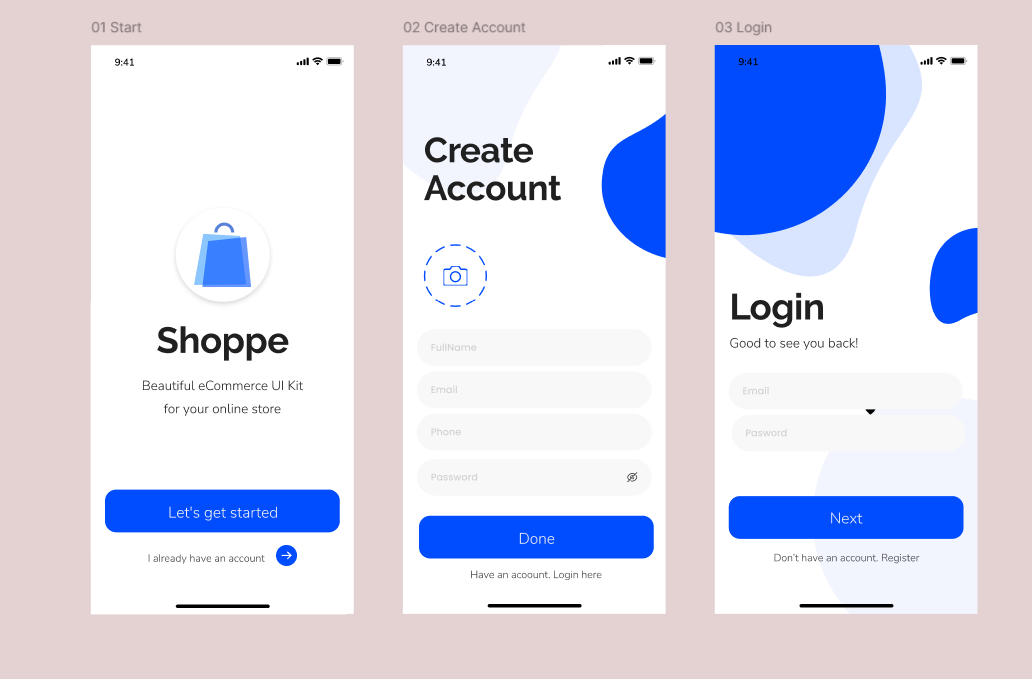
Bảng 2: bảng cơ sở dữ liệu

# CHƯƠNG III

# USER INTERFACE DESIGN

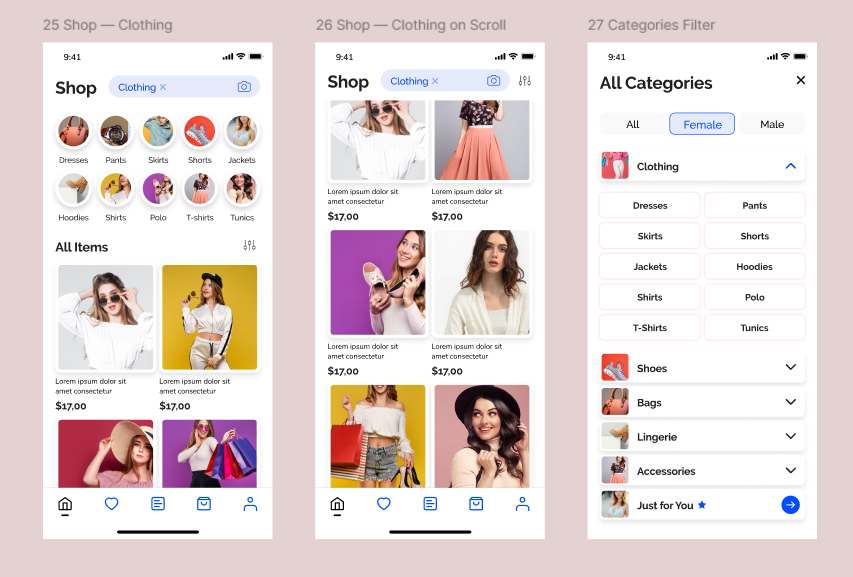
## Giao diện người dùng

### 2.1.1. Giao diện thiết kế bắt đầu, đăng ký, đăng nhập



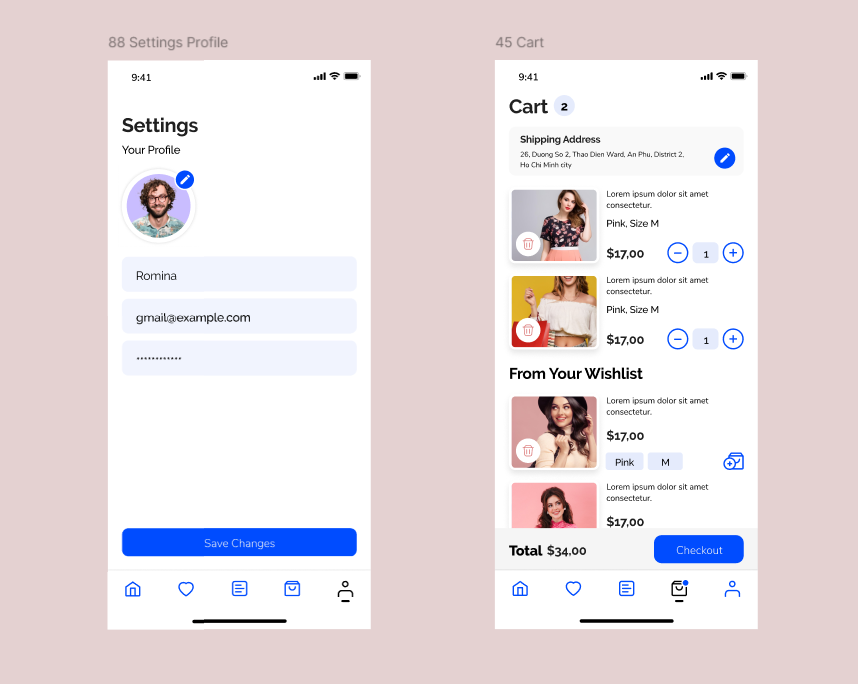
###### Hình 2: Giao diện thiết kế bắt đầu, đăng ký, đăng nhập

### 2.1.2. Giao diện trang chủ



###### Hình 3: Giao diện thiết kế trang chủ

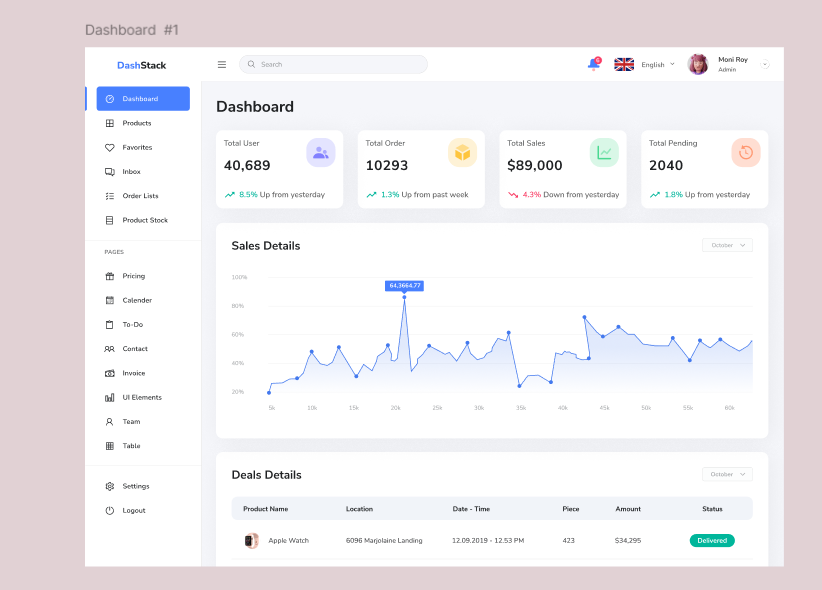
### 2.1.3. Giao diện giỏ hàng, chỉnh sửa thông tin



Hình 5: Giao diện giỏ hàng, chỉnh sửa thông tin

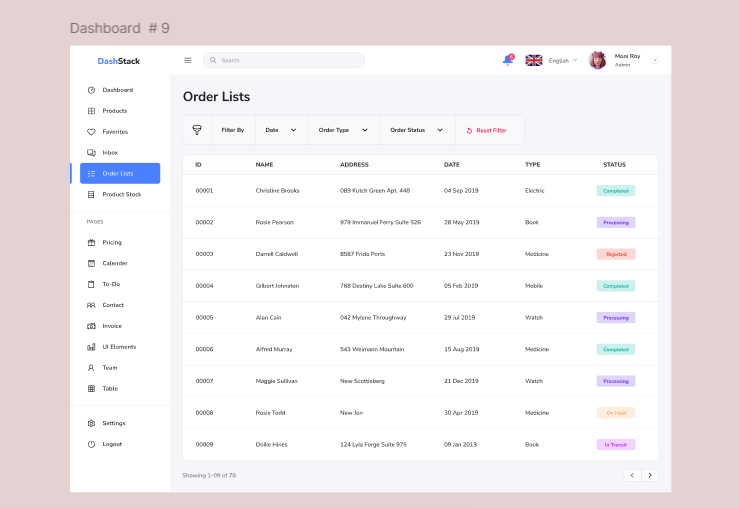
## 2.2. Giao diện admin

### 2.2.1. Giao diện admin DashBoard

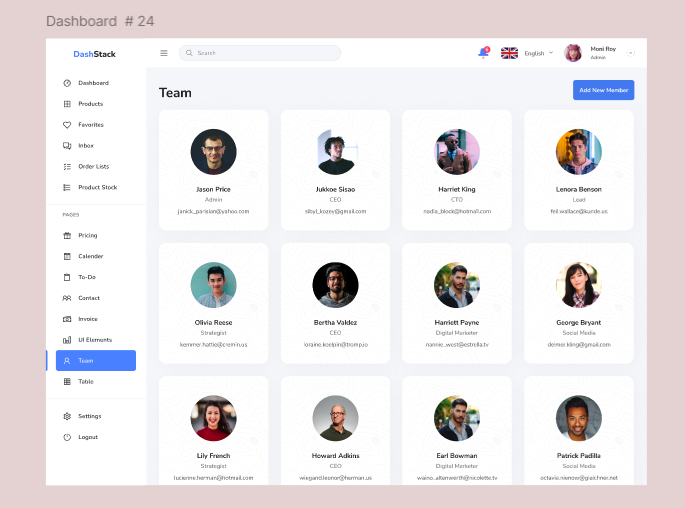


Hình 6: Giao diện admin DashBoard

### 2.2.2. Giao diện admin đơn hàng

Hình 7: Giao diện admin đơn hàng

### 2.2.3. Giao diện danh sách người dùng



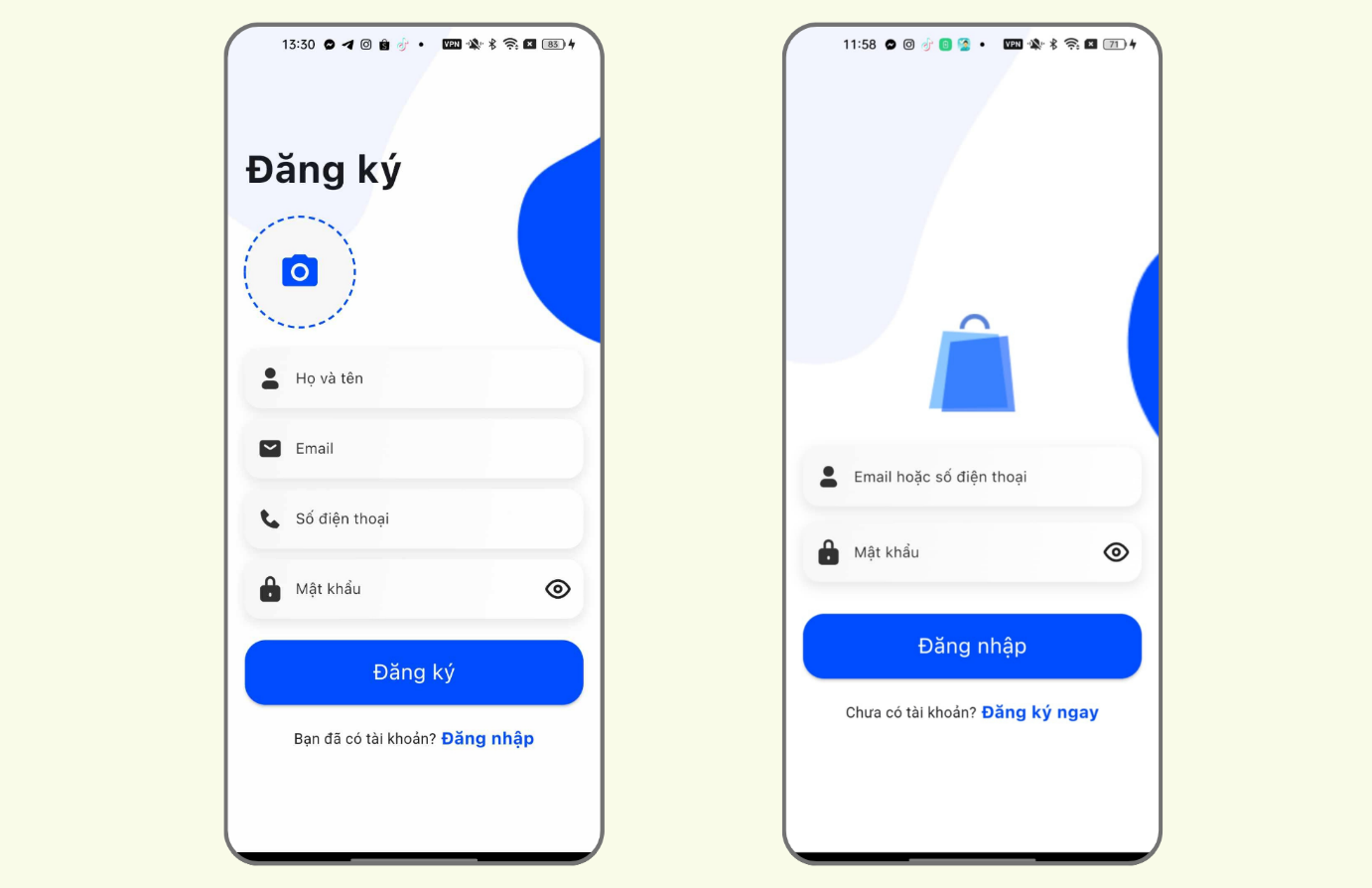
###### *Hình 8: Giao diện admin danh sách người dùng*

# CHƯƠNG III

# DEMO CHƯƠNG TRÌNH

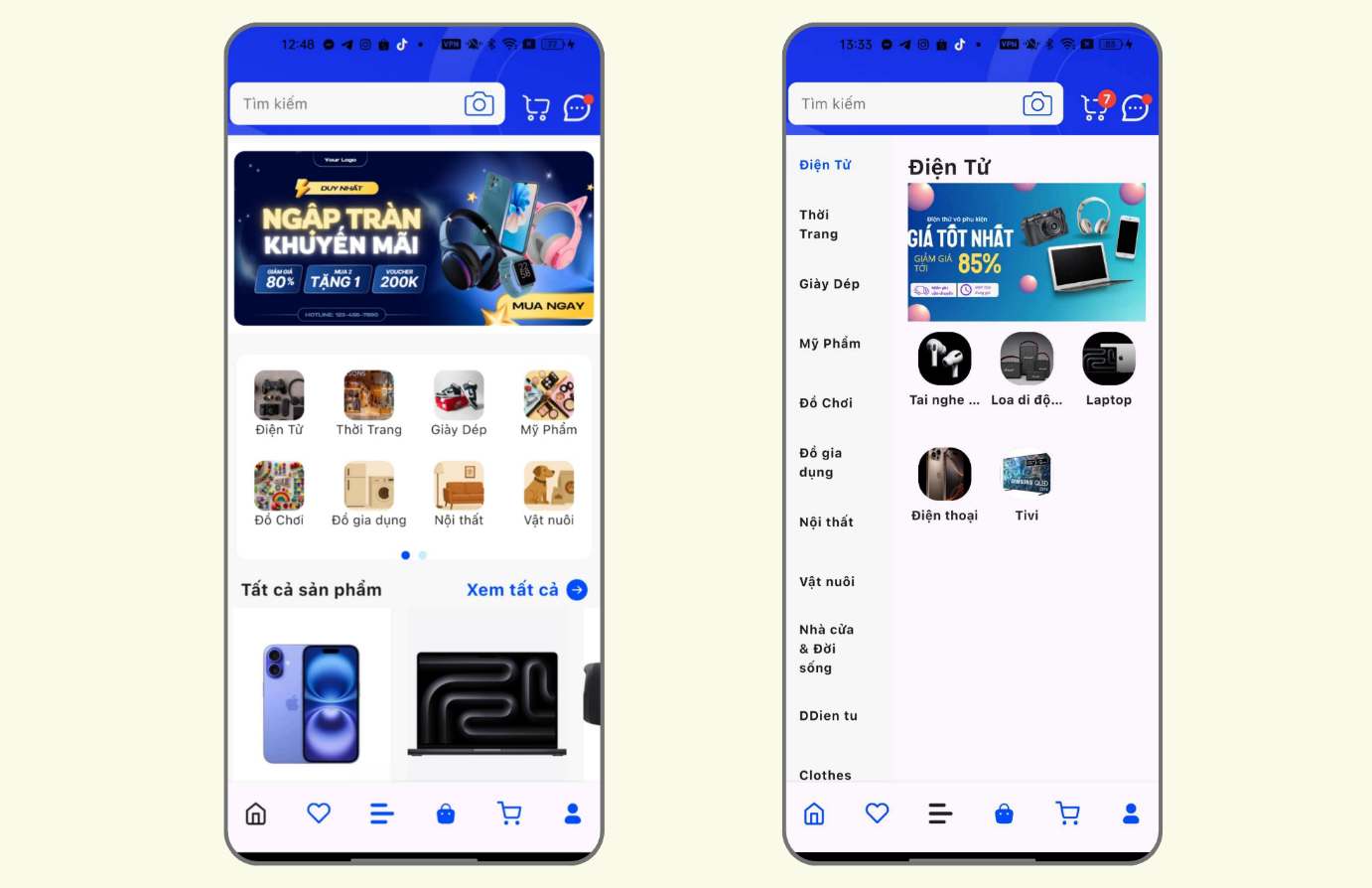
## Giao diện người dùng

### 3.1.1. Giao diện đăng ký, đăng nhập



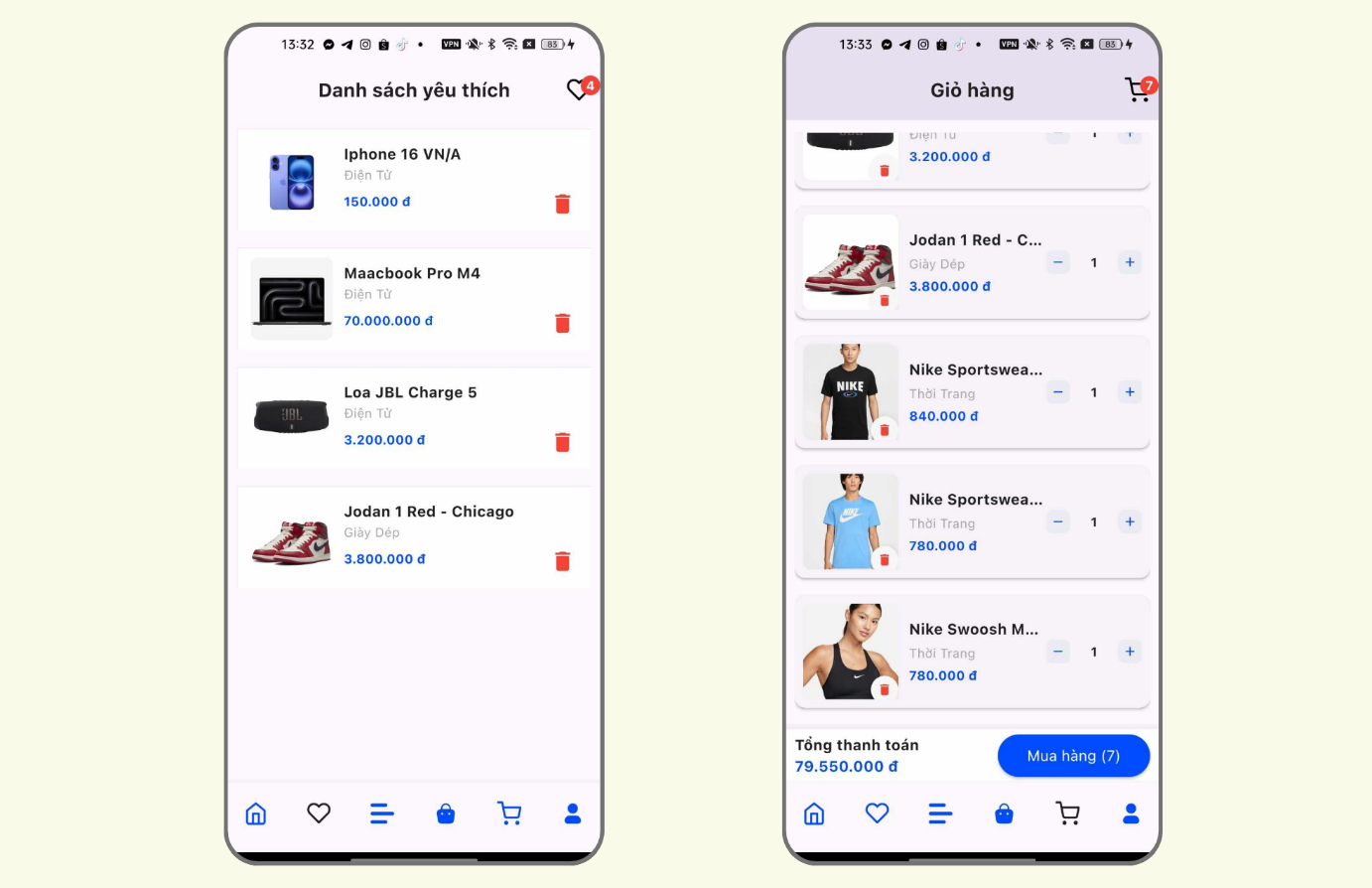
###### Hình 11: Giao diện đăng ký, đăng nhập

### 3.1.2. Giao diện trang chủ, danh mục sản phẩm



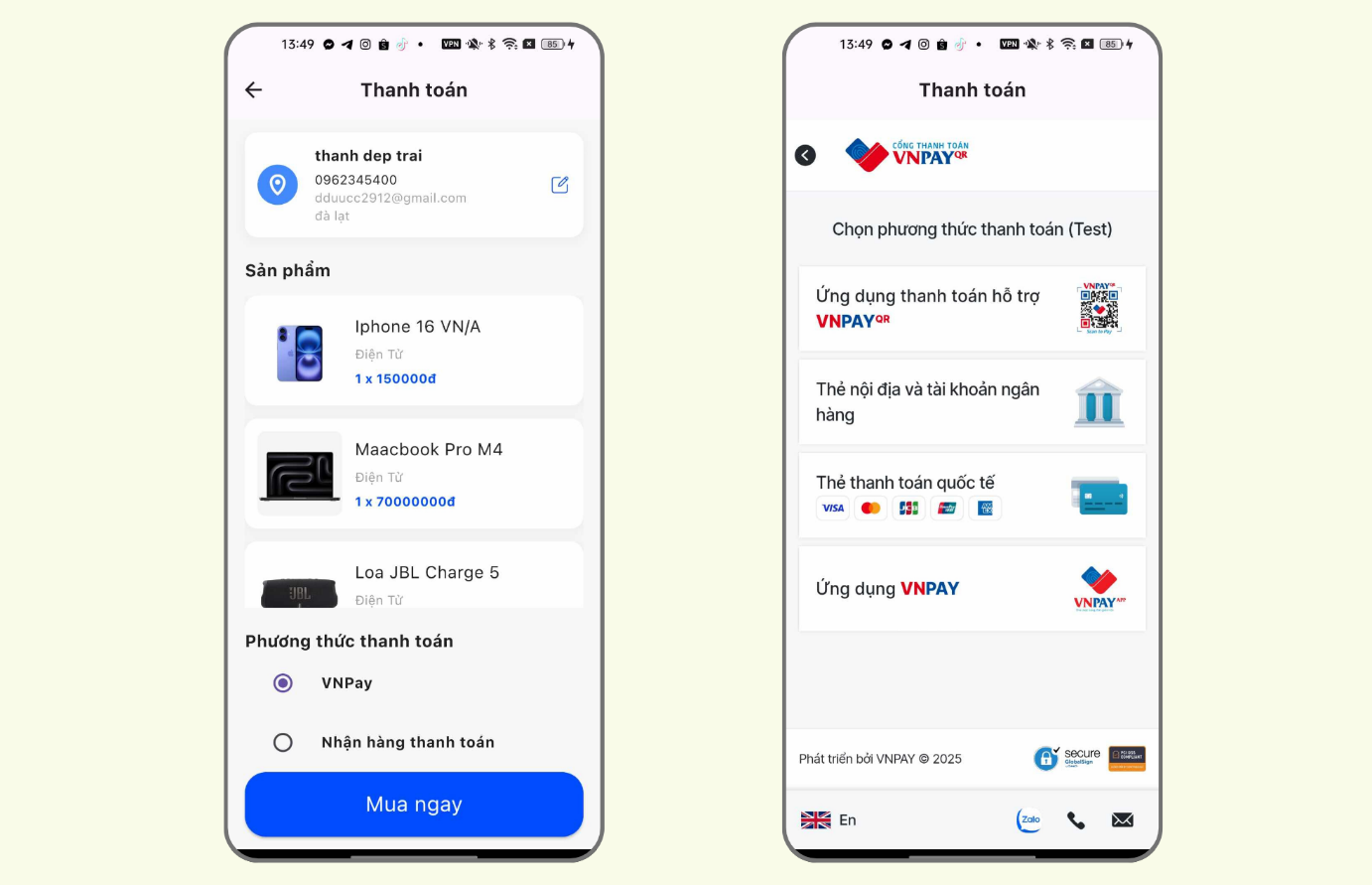
###### Hình 12: Giao diện trang chủ, danh mục sản phẩm

### 3.1.3. Giao diện giỏ hàng, thêm yêu thích



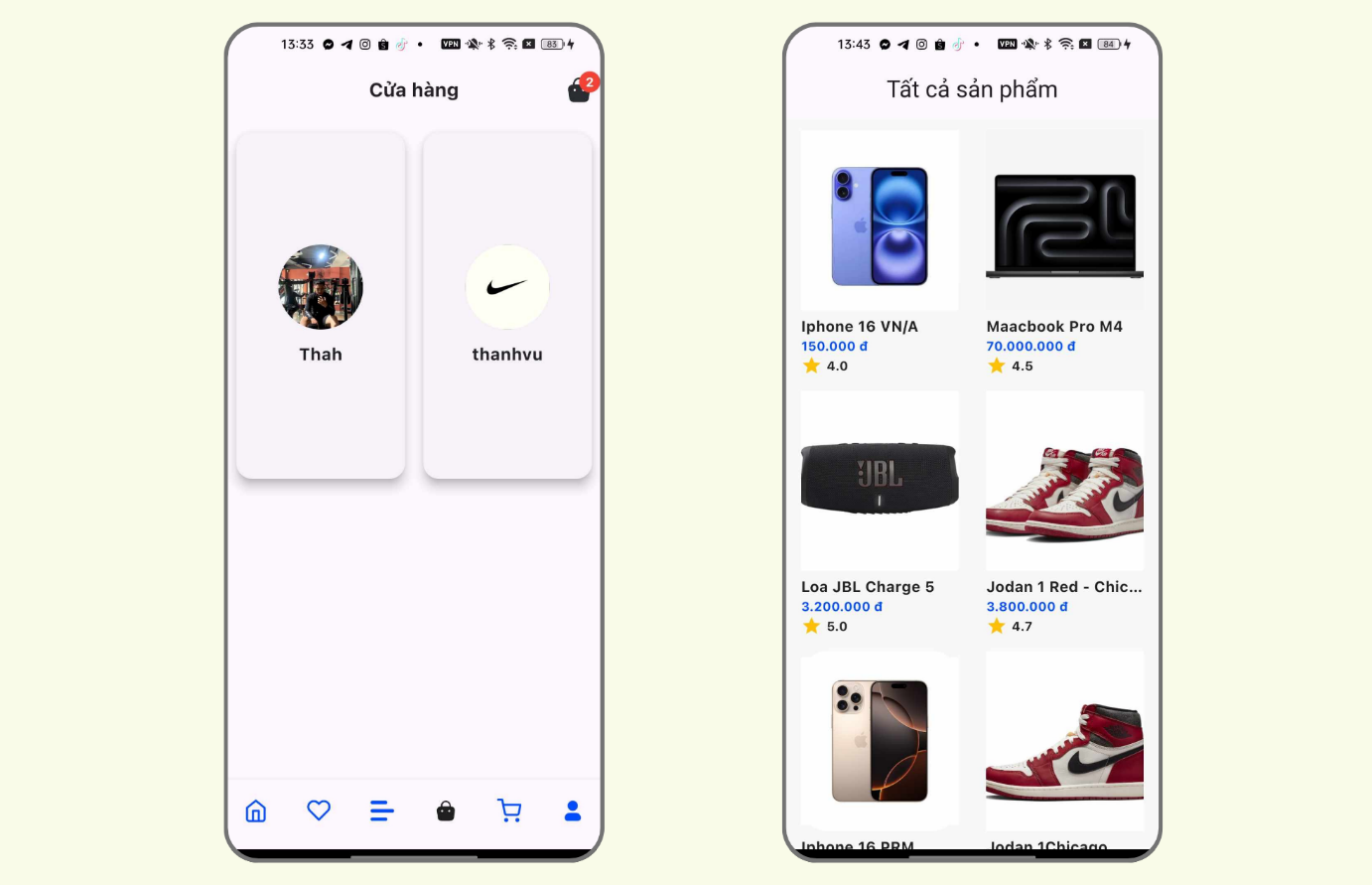
###### Hình 13: Giao diện giỏ hàng, thêm yêu thích

### 3.1.4. Giao diện thanh toán



###### Hình 14: Giao diện thanh toán

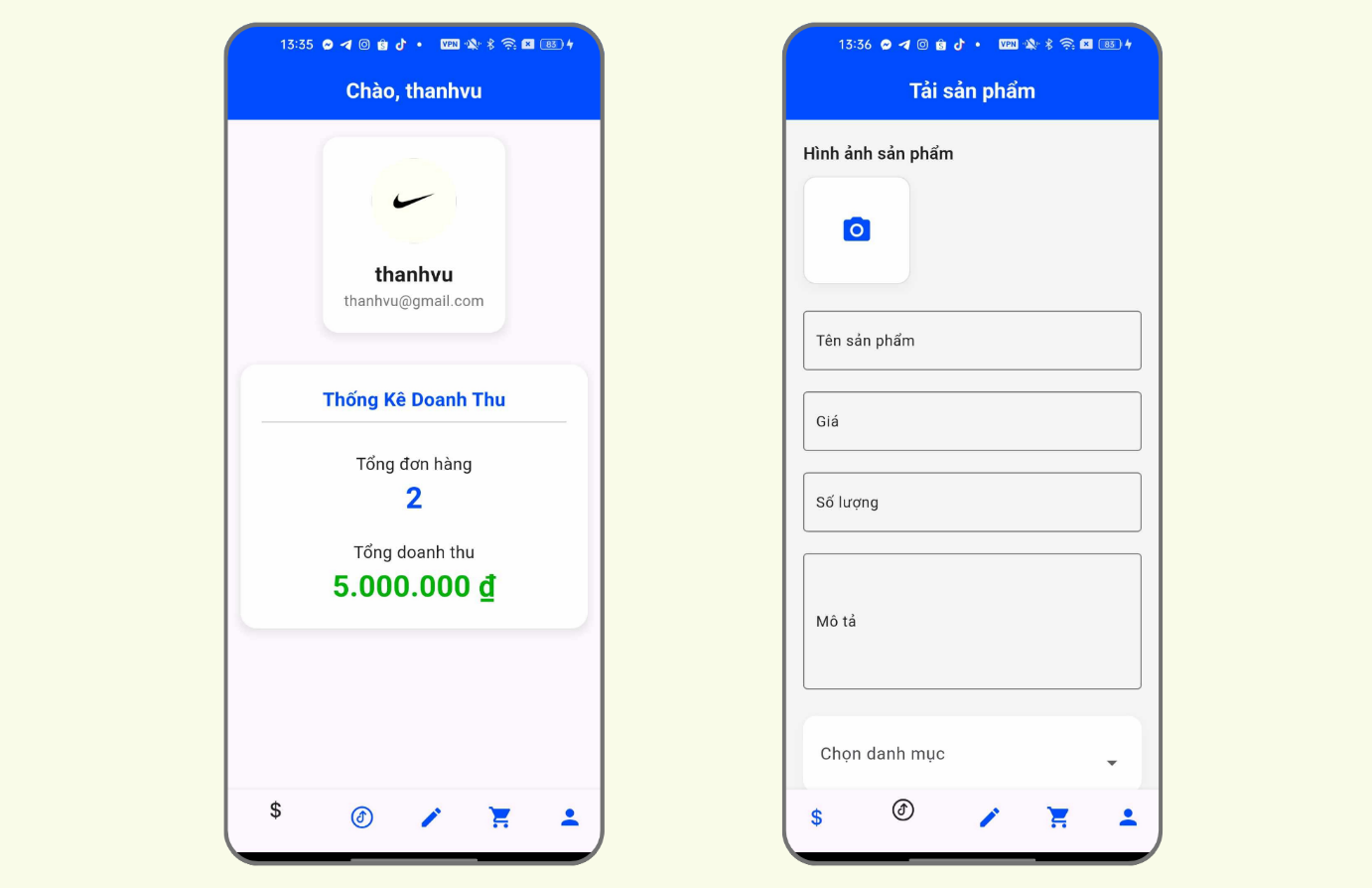
### 3.1.6. Giao diện cửa hàng



###### Hình 16: Giao diện cửa hàng

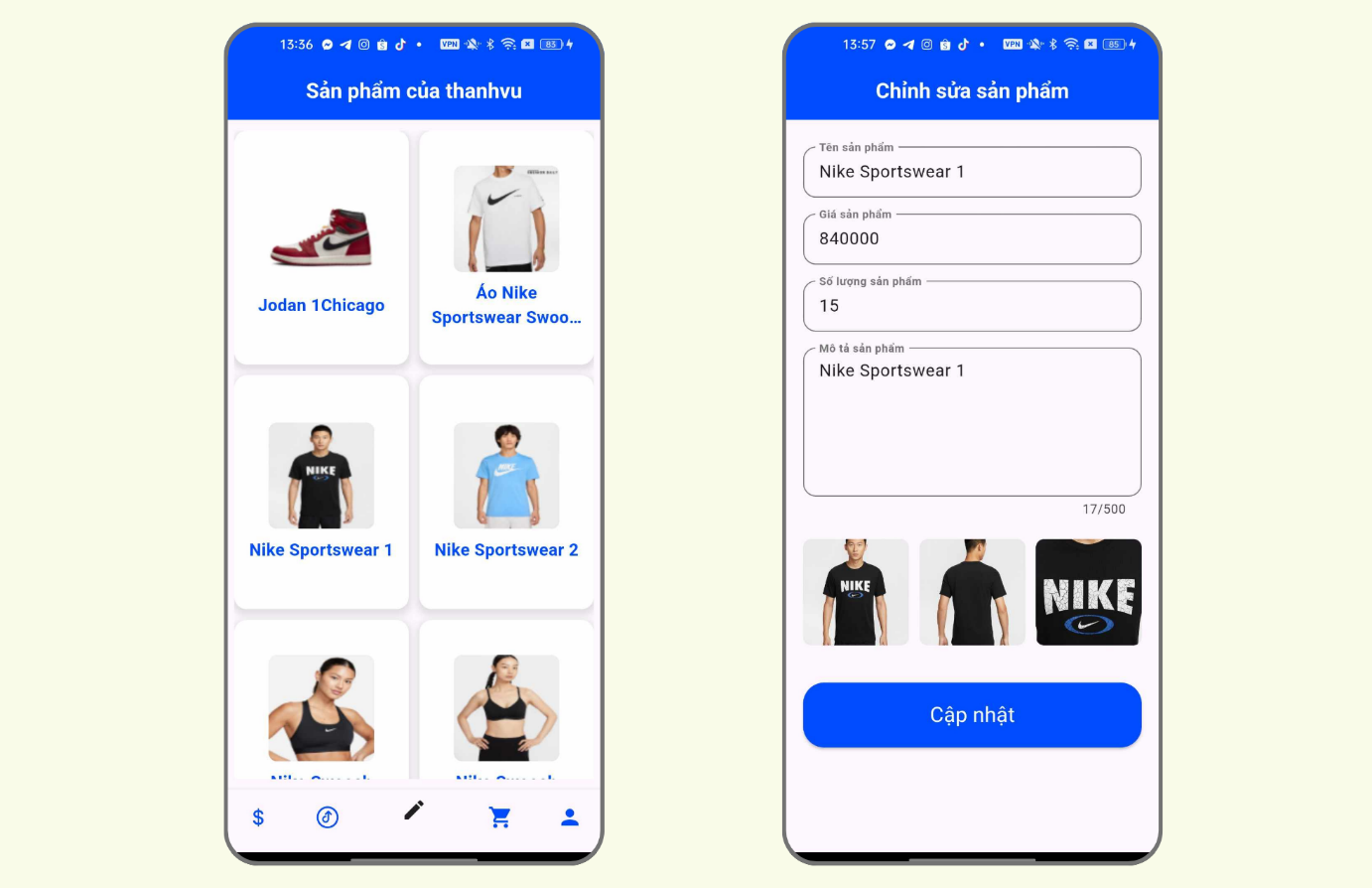
## Giao diện nhà bán hàng

### 3.2.1. Giao diện người bán thống kê, tải sản phẩm



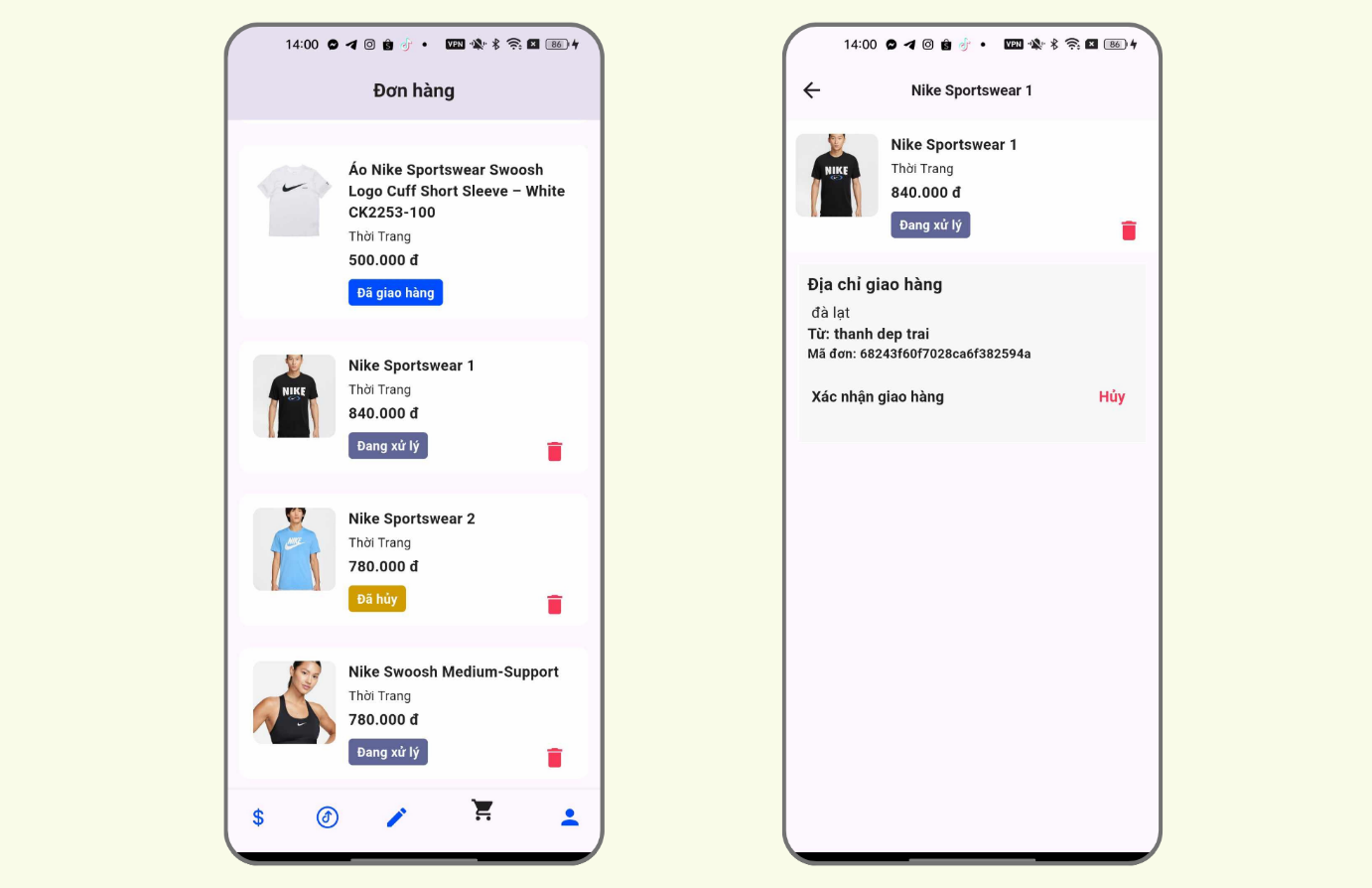
###### Hình 17: Giao diện người bán thống kê, tải sản phẩm

### 3.2.2. Giao diện chỉnh sửa sản phẩm



###### Hình 18: Giao diện chỉnh sửa thông tin

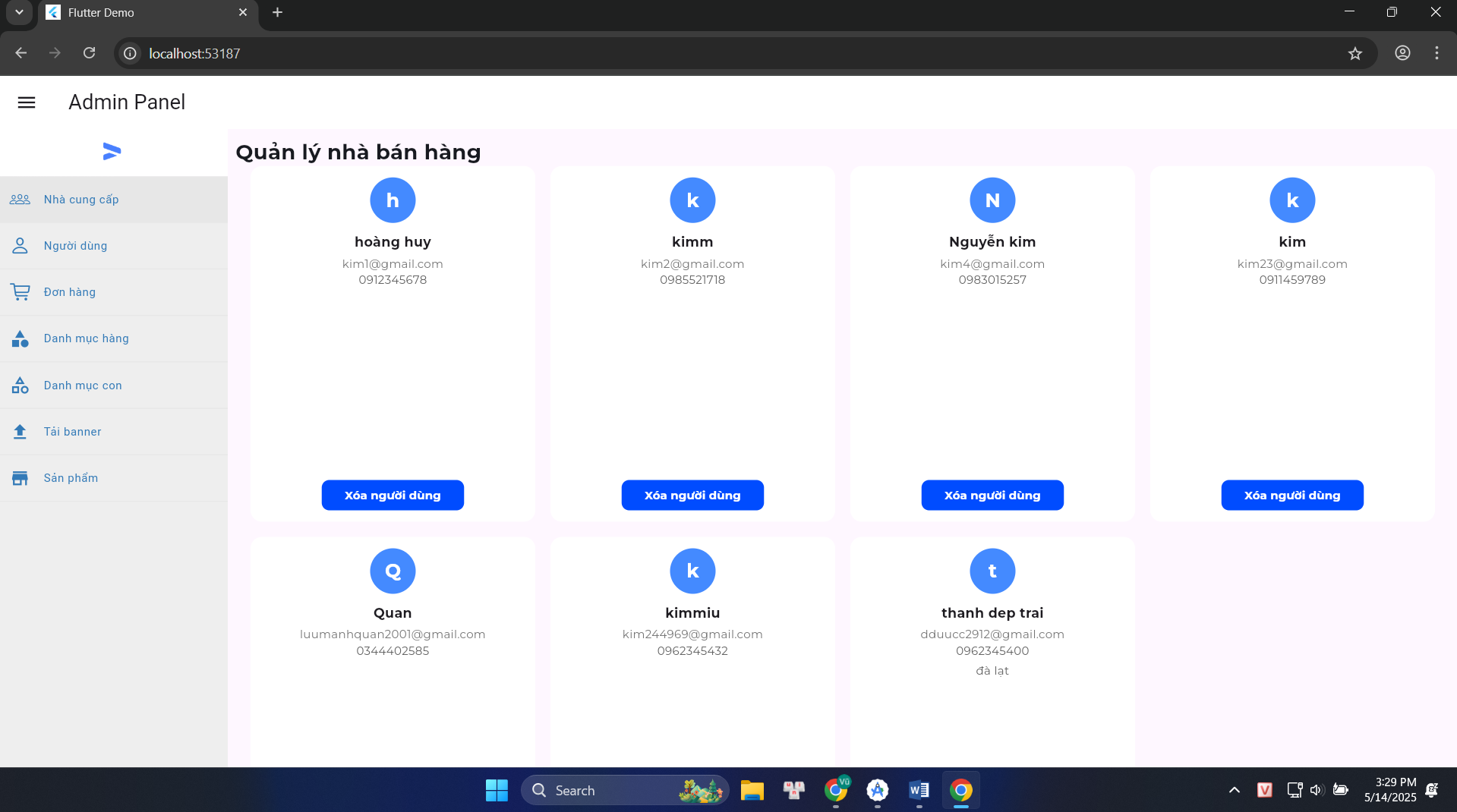
### 3.2.3. Giao diện người bán đơn hàng



###### Hình 19: Giao diện người bán đơn hàng

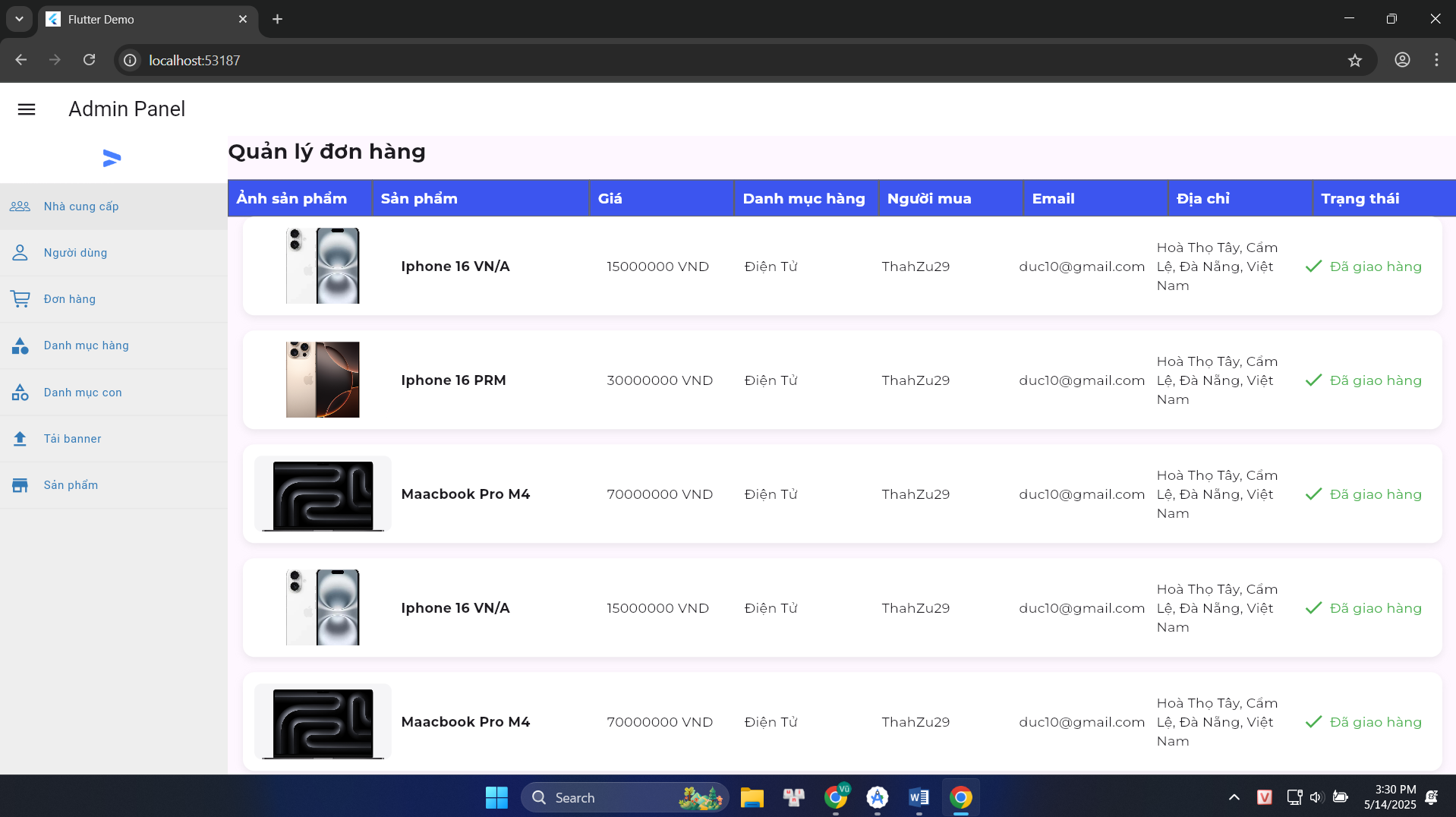
## Giao diện admin

##### 3.3.1. Giao diện admin quản lý người dùng



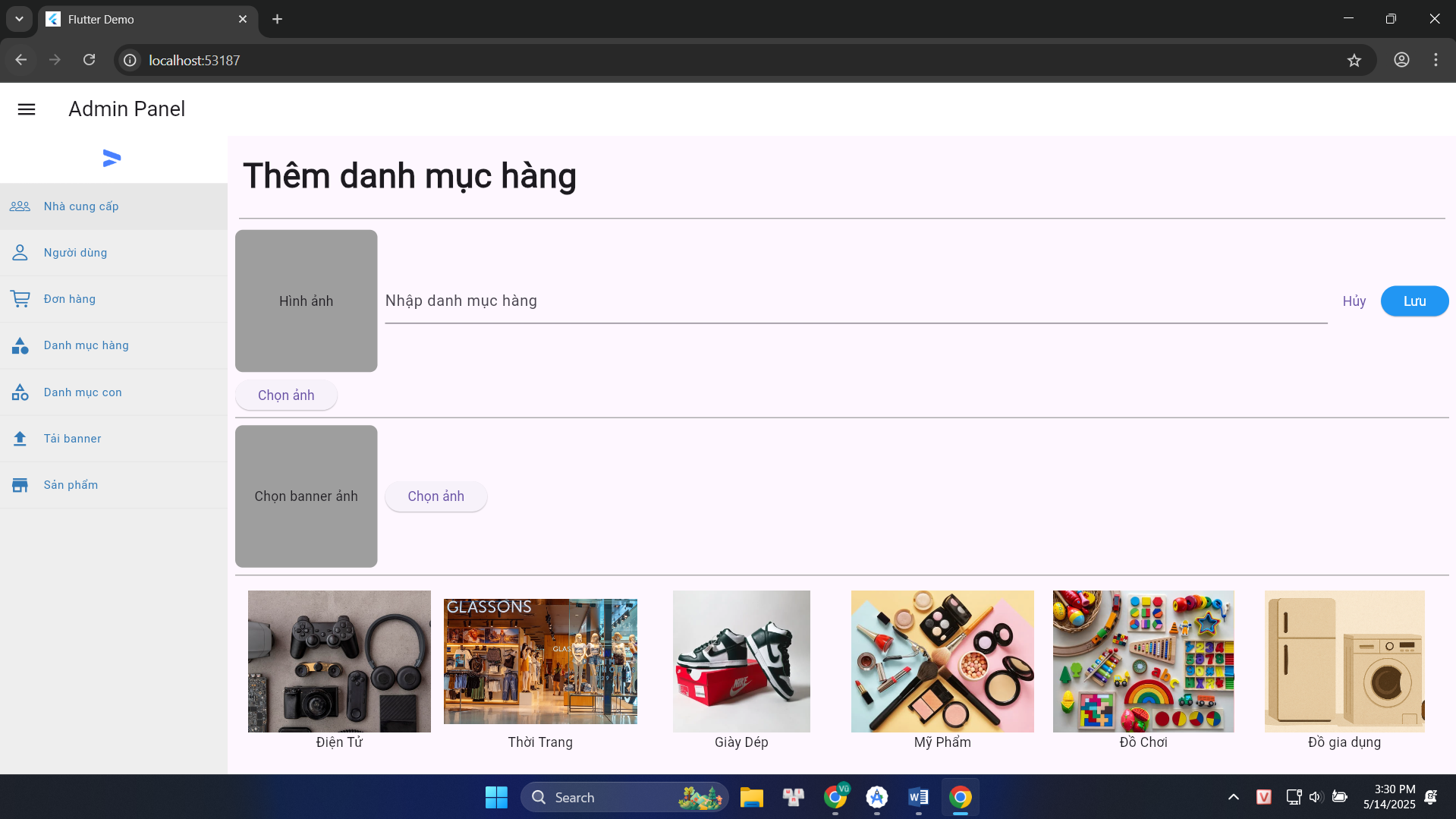
###### Hình 21: Giao diện admin quản lý người dùng

##### 3.3.2. Giao diện admin đơn hàng



###### Hình 22: Giao diện admin quản lý đơn hàng

##### 3.3.3. Giao diện admin upload danh mục hàng



###### Hình 23: Giao diện admin thêm danh mục

# CHƯƠNG V

# KẾT LUẬN

Qua quá trình nghiên cứu và thực hiện luận văn "Phát triển ứng dụng kết nối doanh nghiệp và khách hàng qua nền tảng trao đổi sản phẩm trực tuyến tích hợp đa cửa hàng với Flutter, NodeJS, Express và MongoDB", dự án đã đạt được một số kết quả đáng ghi nhận, đồng thời cũng nhận thấy những hạn chế cần khắc phục và định hướng phát triển trong tương lai.

Ưu điểm

• Xây dựng nền tảng cơ bản cho ứng dụng đa cửa hàng: Dự án đã hoàn thiện một ứng dụng mobile đa nền tảng (Android và iOS) sử dụng Flutter, kết hợp với backend NodeJS, Express và cơ sở dữ liệu MongoDB, đáp ứng các chức năng cơ bản như đăng ký/đăng nhập, xem/tìm kiếm sản phẩm, thêm vào giỏ hàng, mua hàng, quản lý đơn hàng, và quản lý sản phẩm. Điều này tạo ra một nền tảng kết nối hiệu quả giữa doanh nghiệp và khách hàng.

• Phân tích và thiết kế hệ thống chi tiết: Đã thực hiện đầy đủ các bước phân tích yêu cầu và thiết kế hệ thống, bao gồm các sơ đồ Use Case sơ đồ tuần tự, sơ đồ hoạt động. Các thiết kế này đảm bảo tính logic và hỗ trợ tốt cho việc triển khai ứng dụng.

• Ứng dụng công nghệ hiện đại: Việc sử dụng Flutter giúp ứng dụng đạt hiệu suất cao và giao diện đồng nhất trên các nền tảng. Backend với NodeJS và Express cung cấp API RESTful ổn định, trong khi MongoDB đáp ứng tốt yêu cầu lưu trữ dữ liệu linh hoạt cho mô hình đa cửa hàng.

• Tăng cường trải nghiệm người dùng: Ứng dụng đã triển khai các tính năng như nhắn tin giữa người dùng và nhà bán hàng, đánh giá sản phẩm, và thống kê thu nhập, góp phần nâng cao trải nghiệm người dùng và hỗ trợ doanh nghiệp quản lý hiệu quả.

• Khả năng mở rộng: Thiết kế cơ sở dữ liệu với MongoDB và kiến trúc API của backend cho phép hệ thống dễ dàng mở rộng khi số lượng cửa hàng và người dùng tăng lên.

Hạn chế

• Giao diện người dùng còn đơn giản: Giao diện ứng dụng vẫn ở mức cơ bản, chưa được tối ưu về mặt thẩm mỹ và trải nghiệm người dùng (UX/UI), điều này có thể ảnh hưởng đến sự thu hút đối với người dùng.

• Hiệu suất cần cải thiện: Với lượng dữ liệu lớn từ nhiều cửa hàng, hệ thống chưa được tối ưu hóa hoàn toàn về truy vấn cơ sở dữ liệu và xử lý tải, dẫn đến nguy cơ chậm trễ khi số lượng người dùng tăng cao.

• Chưa đảm bảo bảo mật toàn diện: Các biện pháp bảo mật như mã hóa dữ liệu nhạy cảm, xác thực hai yếu tố (2FA), hoặc kiểm tra bảo mật định kỳ chưa được áp dụng triệt để, có thể gây rủi ro cho hệ thống.

• Kiểm thử chưa toàn diện: Việc kiểm thử chỉ được thực hiện ở mức cơ bản trên một số chức năng chính, chưa bao quát toàn bộ hệ thống, dẫn đến khả năng tồn tại lỗi chưa được phát hiện.

Hướng phát triển

• Bổ sung các tính năng nâng cao: Triển khai các chức năng như báo cáo doanh thu theo thời gian thực, tích hợp nhiều phương thức thanh toán đa dạng, và xây dựng hệ thống quản lý giao hàng hiệu quả để đáp ứng tốt hơn nhu cầu thực tế.

• Cải thiện giao diện và trải nghiệm người dùng: Thiết kế lại giao diện với phong cách hiện đại, tối ưu hóa UX/UI để tăng tính hấp dẫn và thân thiện với người dùng trên cả Android và iOS.

• Tối ưu hóa hiệu suất và mở rộng hệ thống: Áp dụng các kỹ thuật như indexing trong MongoDB, sử dụng caching (ví dụ: Redis) để tăng tốc độ truy vấn, và triển khai kiến trúc microservices để đảm bảo hệ thống hoạt động ổn định khi quy mô tăng.

• Tăng cường bảo mật: Tích hợp các biện pháp bảo mật tiên tiến như mã hóa dữ liệu, xác thực hai yếu tố, và thực hiện kiểm tra bảo mật định kỳ để bảo vệ thông tin người dùng và doanh nghiệp.